Electrochemical Double Layered Capacitor Development and Implementation System

By

Gavin P. Strunk

Submitted to the graduate degree program in Department of Mechanical Engineering and the Graduate Faculty to the University of Kansas in parital fulfillment of the requirements for the degree of Doctor of Philosophy.

_____
Dr. Terry N. Faddis, Chair

_____
Dr. Carl Luchies, Committee Member

_____
Dr. James Miller, Committee Member

_____
Dr. Sara Wilson, Committee Member

_____
Dr. Xinmai Yang, Committee Member

Date Defended:

The Dissertation Committee for Gavin P. Strunk certifies that this is the approved version of the following dissertation:

Electrochemical Double Layered Capacitor Development and Implementation System

_____
Dr. Terry N. Faddis, Chair

_____
Dr. Carl Luchies, Committee Member

_____
Dr. James Miller, Committee Member

_____
Dr. Sara Wilson, Committee Member

_____
Dr. Xinmai Yang, Committee Member

Date Approved:

ABSTRACT

Electrochemical Double Layered Capacitors (EDLC's) are becoming a more popular topic of research for hybrid power systems, especially vehicles. They are known for their high power density, high cycle life, low internal resistance, and wider operating temperature compared to batteries. They are rarely used as a standalone power source; however, because of their lack of energy density compared to batteries and fuel cells. Researchers are now discovering the benefits of using them in hybrid systems. The increased complexity of a hybrid power source presents many challenges. A major drawback of this complexity is the lack of design tools to assist a designer in translating a simulation all the way to a full scale implementation.

A full spectrum of tools was designed to assist designers at all stages of implementation including: single cell testing, a multi-cell management system, and a full scale vehicle data acquisition system to monitor performance. First, the full scale vehicle data acquisition is described. The system is isolated from the electric shuttle bus it was tested on to allow the system to be ported to other vehicles and applications. This was done to modularize the system to characterize a wide variety of full scale applications. Next, a single cell test system was designed that allows the designer to characterize cell specifications, as well as, test control and safety systems in a controlled environment. The goal is to ensure safety systems can be thoroughly tested to ensure robustness as the bank is scaled up. This system also includes simulation models that provide examples of using the simulation to predict the behavior of a cell and the test system to validate the results of the simulation. This information is then used by the designer to more effectively design sensor ranges for the bank.

Finally, a multi-cell EDLC management system was designed to implement a bank. It incorporates 12 series EDLC cells per control module, and the modular design allows expandability in parallel and series to fit any application and number of cells required. Lastly, test procedures were run to validate the proper operation of the systems.

ACKOWLEDGEMENTS

I would like to extend a special thanks to my graduate advisor, Dr. Terry N. Faddis, for his guidance and support during this research. He has truly given me an unmeasurable amount of valuable experience and knowledge. I would also like to thank my committee members, Dr. Carl Luchies, Dr. James Miller, Dr. Sara Wilson, and Dr. Xinmai Yang. They have all given excellent advice and support throughout my time at the University of Kansas.

Next, I would like to thank the University of Kansas Transportation Research Institute and the University of Kansas Medical Center for providing me with funding for my research.

Finally, I would like to thank my family and friends whose support has been vital to my success. I would like to give special thanks to my parents, Richard and Benita, for their advice, support, and encouragement. To my sister Sara, your support was also very much appreciated. Thanks.

TABLE OF CONTENTS

LIST OF FIGURES

NONMENCLATURE

A/D	Analog to Digital
CAN	Controller Area Network
D/A	Digital to Analog
DRAM	Dynamic Random-Access Memory
EDLC	Electrochemical Double Layered Capacitor
FPGA	Field Programmable Gate Array
I^2C	Inter-Integrated Circuit
KSPS	Kilo Samples Per Second
LDO	Low-Dropout
MSPS	Million Samples Per Second
OTG	On-The-Go
PCB	Printed Circuit Board
PCI	Peripheral Component Interface
SPI	Serial Peripheral Interface
TTL	Transistor Transistor Logic
UART	Universal Asynchronous Receiver/Transmitter

1. INTRODUCTION

1.1 Hybrid Electric Vehicles and Control

Alternative fuel vehicles have been ever increasing in interest and performance with companies such as Tesla leading the way. The need to reduce vehicle emissions, improve vehicle efficiency, reduce road noise, and fossil fuel dependency are all reasons that alternative fuel vehicles are being researched by companies and academia alike [1]. Pure electric vehicles have not been widely accepted; however, because they typically suffer from one of the following shortcomings: insufficient range, insufficient power for large vehicles, or high cost. To combat these issues, many companies and researchers have come up with the intermediate solution of hybrid vehicles. Hybrid vehicles have the ability to have longer ranges, deliver more power as compared to a typical vehicle grade electric motor with batteries, and reduced cost by not requiring as many batteries, which comprises one-third of the vehicle cost [2]. One example of this is using ultracapacitors and batteries to satisfy the peak power needs thus no longer requiring the battery pack to be oversized for these situations [3][4]. This reduces the cost, as well as, improving the cycle life of the batteries extending the maintenance time of the vehicle [4]. Typical current commercially available hybrids involved batteries and a gasoline or diesel motor to extend the driving range of the vehicle. These hybrid vehicles address some of the issues such as increasing the vehicle range and efficiency, but do not eliminate the use of fossil fuels. Therefore, they are serving as an intermediate step to a more favorable solution to the problem.

The commercially available hybrid vehicles typically use a simplistic control strategy by using strictly the available battery power until it is depleted and then switching to the motor. More sophisticated blended strategies have been a focus of many current researchers to further improve efficiency [5]. A variety of studies have shown improvements in range by using

blended strategies suggesting there is room for efficiency increases using intelligent control methods not only in this style of hybrid vehicle but in others as well [5].

Along with the infancy of the control strategies, current vehicles typically provide very little information to the driver regarding power consumption or potentially hazardous conditions to the vehicle components. Current gasoline cars have implemented features such as displaying the real time miles per gallon to the driver in an attempt to have more informed drivers to increase driver efficiency. Equivalent features have not been as successfully integrated in the commercial hybrid vehicle technologies.

Safety measures are also in a relatively new state for these vehicles. There have been various accidents involving battery packs melting or exploding in vehicles without warning. This drives the assumption that there is also a need for improved safety systems implemented in the hardware as well as the vehicle controller. It is typical that the pack has to be physically removed and disassembled to find the cause of the failure. This approach does not always reveal the true cause of the failure; however, as many times the pack is destroyed beyond allowing troubleshooting. This is due to minimal or no data being recorded and communicated to a system outside the pack itself. The failure modes also have room for improvement in not only detection, but preemptive action to attempt to stop the failure from occurring. An example is the recent Tesla fire [6]. The vehicle warned the drive to pull over and exit the vehicle before the vehicle caught on fire resulting in no casualties as a result of the failure. Many power systems monitor the basic voltage, current, and temperature readings, but these are typically described in a binary state of safe or not safe. A better approach is to realize these limits are being approached and the velocity at which they are being approached, so preventative actions can be implemented to attempt to correct the failing conditions. Despite these shortcomings, hybrid

vehicles represent a huge opportunity to increase efficiency and reduce carbon emissions. A newer style of hybrid involving all electrochemical components has also started receiving more research attention. These vehicles typically involve some combination of batteries, electrochemical double layered capacitors (EDLC), and/or fuel cells [7].

1.2 Current Power Sources

The three most common electrochemical power sources are batteries, EDLCs, and fuel cells. They vary in properties, cost, size, and availability. Batteries have not only been around the longest commercially, but are also the most popular choice of power source in pure electric and hybrid electric vehicles. Currently lithium based batteries are the most popular choice. Lithium batteries have a good balance of energy and power properties by having good energy density and good power density. They also have a good balance of cost [8][9][10]. For these reasons they are the best choice for most single source applications. There is still considerable room for improvement in batteries to have higher capacities and ability to deliver more power. Current batteries also suffer from low cycle-life compared to other energy forms, especially when subjected to pulse current loads like those in vehicle applications. Typically batteries also require much longer recharge times than current gasoline tanks, which can be an annoyance for the user [11][12][13].

Fuel cells are the newest of the power sources discussed here but offer some advantages over batteries. They have very good energy density compared to lithium batteries as well as the ability to be refilled much quicker than current batteries charge times [14]. This is very appealing in vehicle applications where vehicle range is directly related to energy density. Recharge time is also a variable that has to be considered in vehicle applications because this relates to the user experience. We have become accustom to refilling gasoline vehicles in a few

minutes, so having to wait hours for batteries to charge seems like a large burden to many vehicle owners. Fuel cells have the ability to be refilled in a similar fashion as current gasoline vehicles, which would solve this problem. Fuel cells are not without their own disadvantages though. They have very poor power density and are damaged easily when excessive power is drawn from them resulting in fuel starvation and irreversible damage to the system [15]. Fuel cells also suffer from low efficiency at very low power levels and are unidirectional so they cannot recapture regenerative breaking energy. Finally, they have to have been charged before they can deliver power so they display very slow dynamics [15]. It has also been shown that peak power conditions cause a significant voltage drop resulting in inconsistent power [16]. This is a large problem when running an electric motor because electric motors require a large spike of current at the start of their response. Future breakthroughs will likely address many of these problems, but in their current state fuel cells are not viable power sources for vehicle applications by themselves.

The final power source is electrochemical double layered capacitors. These are also commonly referred to as ultracapacitors or supercapacitors. EDLC's have excellent power density and are capable of charging and discharging very quickly due to their low internal resistance [17][18]. EDLC's also typically have several orders of magnitude higher cycle life than batteries. This makes them ideal in applications that either require or deliver bursts of power, such as capturing regenerative breaking energy and power balancing [19]. Another advantage of an EDLC is a wider temperature operating range than batteries. In addition to operating over a wider range, they are also much more consistent in the efficiency of their output over their thermal operating range [20]. They do however suffer from low energy density compared to batteries, which rarely allows them to be a viable option as a sole power source.

Despite this, EDLCs have the ability to improve performance of both battery and fuel cell hybrid vehicles. When combined with batteries the EDLCs would be able to absorb the power spikes seen from electric motors, deliver more stable power, as well as reduce the harsh conditions the battery pack sees. This would improve the overall lifetime of the battery system and increase the overall power density of the hybrid source [21][22]. In tandem with fuel cells, EDLCs would again be able to make up for the poor power density as well as allowing the fuel cell time to charge while power is delivered from the EDLC bank. These advantages make EDLC an ideal candidate for hybrid sources when combined with batteries or fuel cells.

1.3 Problem Identification

A significant amount of research effort is focused on component research on the previously mentions sources. There has also been some research on proper implementation of the battery systems, but significantly less revolving around EDLCs. The need exists for development tools to not only characterize EDLC cells, but then provide a means to scale the cells up to a full size bank without sacrificing safety or reliability. Trends have shown that EDLCs are rarely a viable single source of power especially in vehicles. Therefore, in addition to implementing these systems, the design work needs to be conducted in a way that allows the final bank to be easily integrated with other power sources, such as batteries or fuel cells. This lays the groundwork for a layered approach to hybrid power source control, which not only makes the complexity more manageable but modular as well. Modularity in a power source management design is a must because of the rapid rate of research progression in the area and the varying requirements from application to application.

Current efforts in this area have been focused on algorithms and simulations. As this is typically the first step, this research has laid some ground work as to a variety of approaches to

implementing such a system [5][7][9]. The major problem with the simulations is that they lack the depth of accounting for many losses associated with physical implementation of such a system. Examples of this include losses through power electronics and thermal models. They also typically do not account for subsystems such as cooling or heating systems that are required. These losses do not typically change the effectiveness of an algorithm, but they cannot be overlooked when implementing a physical system as they directly affect the safety and reliability of the bank. There have been a couple of researchers working on concepts of how to implement systems that provide information feedback from a vehicle, but these systems are normally a means to test algorithms with less emphasis on robust implementation for a commercial application [23][24].

1.4 Research Objective

The overall research objective is to design development tools for EDLCs that allow an implementer to test all phases of a design from a single cell all the way to a complete bank with a management system, as well as a platform to perform these tests to validate and improve existing models. This is accomplished in three parts: a full vehicle test platform, a single cell test system, and a multi-cell management system.

The full vehicle test platform will have to be versatile and modular as well so testing a variety of systems can be conducted. The vehicle will also need a data acquisition system that is capable of characterizing the systems to demonstrate improvements or faults in the designs [25]. This system will provide the driver with feedback as well as show the information about the rest of the vehicle so any dependent behavior can also be identified and proper comparisons between systems can be made without bias from other vehicle systems.

The single test cell system will need to support a large variety of tests and testing environments so it will also be modular in nature. This will include a modular power supply, EDLC connection, and modular load. In addition, the measurements will be isolated from the power path. This keeps the measurement system from affecting the behavior of the cell. This will also protect the measurement circuitry, which allows the user to test fault conditions as well as push cells to the point of failure without damaging the test system. This will also ensure the measurement system will continue to operate even after a cell has failed to allow the user to identify the conditions that caused the failure. This provides a huge advantage by allowing safety systems to be tested to ensure robustness [26].

Finally, a multi-cell management system will be developed. This system needs to be self-contained in terms of power and scalable in terms of the number of cells. This design will provide a generic controller that can be used to build any combination of cells in series or parallel to provide the desired amount of capacity and power density. By drawing the power from the bank itself, the management system can be packaged with the bank without requiring addition wiring to operate. It should, however, also allow an additional external power source to be added as a backup in case of a pack failure. This, like the test system, would allow the management system to continue to operate even if there is a failure in the pack. This offers several safety advantages, such as the ability to continue operating systems that may still be running. For example, if a short in the bank causes a thermal spike that melts a cell the cooling system may continue to be operational and the continued cooling operation could prevent additional cells from damage.

1.5 Future Impact

The solution to the proposed problem holds a variety of positive future impacts for hybrid vehicle development. First, it will decrease time and complexity of implementing an EDLC bank into a power source. This will encourage more designers to implement this technology when appropriate. It will allow researchers to conduct simultaneous research at all levels of integration moving forward. This ultimately will allow more customization of hybrid electric power sources to properly fit the application, rather than having the user accept the shortcomings of the system. The modularity of the system will allow for integration with current technologies as well as leave room to support future power sources that have not been developed or commercialized yet. These systems allow the designer to test operating and extreme conditions, which will aid in designing more efficient packaging and cooling systems. This in turn will reduce the overall size of the system and the parasitic power consumption. The ability for robust testing will inevitably lead to more intelligent safety systems that not only improve the way failures are handled, but take preemptive action to avoid the failure conditions. Finally, further implementation and testing will reveal flaws in the cells and can be passed down to component researchers to help design better cells at the component level.

1.6 References

[1] Li, Q., et al. (2012). "Energy management strategy for fuel cell/battery/ultracapacitor hybrid vehicle based on fuzzy logic." International Journal of Electrical Power & Energy Systems **43**(1): 514-525.

[2] Tie, S. F. and C. W. Tan (2013). "A review of energy sources and energy management system in electric vehicles." Renewable and Sustainable Energy Reviews **20**(0): 82-102.

[3] Adib, E. and H. Farzanehfard (2009). "Soft switching bidirectional DC–DC converter for ultracapacitor–batteries interface." Energy Conversion and Management **50**(12): 2879-2884.

[4] He, H., et al. (2013). "Energy management strategy research on a hybrid power system by hardware-in-loop experiments." Applied Energy **112**(0): 1311-1317.

[5] Chen, Z., et al. "Energy Management of a Power-Split Plug-in Hybrid Electric Vehicle

Based on Genetic Algorithm and Quadratic Programming." Journal of Power Sources(0).
[6] Musk, Elon. (2013). “Model S Fire.” http://www.teslamotors.com/blog/model-s-fire
[7] Yu, Z., et al. (2011). "An innovative optimal power allocation strategy for fuel cell, battery and supercapacitor hybrid electric vehicle." Journal of Power Sources **196**(4): 2351-2359.
[8] Sharma, P. and T. S. Bhatti (2010). "A review on electrochemical double-layer capacitors." Energy Conversion and Management **51**(12): 2901-2912.
[9] Chao, C.-H. and J.-J. Shieh (2012). "A new control strategy for hybrid fuel cell-battery power systems with improved efficiency." International Journal of Hydrogen Energy **37**(17): 13141-13146.
[10] Payman, A., et al. (2008). "Energy control of supercapacitor/fuel cell hybrid power source." Energy Conversion and Management **49**(6): 1637-1644.
[11] Erdinc, O., et al. (2009). "A wavelet-fuzzy logic based energy management strategy for a fuel cell/battery/ultra-capacitor hybrid vehicular power system." Journal of Power Sources **194**(1): 369-380.
[12] Paladini, V., et al. (2007). "Super-capacitors fuel-cell hybrid electric vehicle optimization and control strategy development." Energy Conversion and Management **48**(11): 3001-3008.
[13] Yuchen, L., et al. (2007). Adaptive Control of an Ultracapacitor Energy Storage System for Hybrid Electric Vehicles. Electric Machines & Drives Conference, 2007. IEMDC '07. IEEE International.
[14] Thounthong, P., et al. (2006). "Control strategy of fuel cell/supercapacitors hybrid power sources for electric vehicle." Journal of Power Sources **158**(1): 806-814.
[15] Lin, W.-S. and C.-H. Zheng (2011). "Energy management of a fuel cell/ultracapacitor hybrid power system using an adaptive optimal-control method." Journal of Power Sources **196**(6): 3280-3289.
[16] Thounthong, P., et al. (2009). "Energy management of fuel cell/battery/supercapacitor hybrid power source for vehicle applications." Journal of Power Sources **193**(1): 376-385.
[17] Jung, D. Y., et al. (2003). "Development of ultracapacitor modules for 42-V automotive electrical systems." Journal of Power Sources **114**(2): 366-373.
[18] Farzanehfard, H., et al. (2008). "A bidirectional soft switched ultracapacitor interface circuit for hybrid electric vehicles." Energy Conversion and Management **49**(12): 3578-3584.
[19] Wu, C. H., et al. (2012). "On-line supercapacitor dynamic models for energy conversion and management." Energy Conversion and Management **53**(1): 337-345.
[20] Gualous, H., et al. (2010). "Supercapacitor ageing at constant temperature and constant voltage and thermal shock." Microelectronics Reliability **50**(9–11): 1783-1788.
[21] Burke, A. and M. Miller (2010). "Testing of electrochemical capacitors: Capacitance, resistance, energy density, and power capability." Electrochimica Acta **55**(25): 7538-7548.
[22] Burke, A. and M. Miller (2011). "The power capability of ultracapacitors and lithium batteries for electric and hybrid vehicle applications." Journal of Power Sources **196**(1): 514-522.
[23] Jianfeng, H., et al. (2008). Application and study of novel electronic technologies on vehicle control system of fuel cell bus. Vehicle Power and Propulsion Conference, 2008. VPPC '08. IEEE.
[24] Zhu, D. and H. Xie (2008). Control strategy optimization of the hybrid electric bus based on

remote self-learning driving cycles. Vehicle Power and Propulsion Conference, 2008. VPPC '08. IEEE.
[25] Baker, B. C., et al. (2012). Electric transit bus for variable grade terrain. Electric Vehicle Conference (IEVC), 2012 IEEE International.
[26] Strunk, G. K., M; Baker, B.; Faddis, T (2012). Ultracapacitor Test System. Electric Vehicle Symposium. Los Angeles, CA.

2. Electric Shuttle Bus Data Acquisition Retrofit

2.1 Introduction

Research into hybrid electric and pure electric vehicles has been a major focus of many groups around the world for a significant period of time. More commercial entities have also started offering hybrid vehicles as an alternative to gasoline vehicles with many beginning to offer viable pure electric vehicles for everyday use [1]. These vehicles typically offer increased gas mileage, decreased environmental footprint, and reduced road noise. These have all been factors that have driven consumers and researchers to pursue this technology with vigilance [2]. Despite the increasing maturity of small cars, progress in applying this same technology has been slower in larger vehicles [3]. This is due to a variety of issues including: large amounts of electrical storage required, larger power output, and higher cost to develop. Because it is expensive to implement this technology in a shuttle bus sized vehicle, it is useful to develop a means for numerous people to test ideas on a single platform [4].

This is the aim at the University of Kansas Intelligent Systems and Automation Lab. A shuttle bus has already been acquired that is currently in a pure electric configuration implementing lead acid batteries. The shuttle bus is shown in Figure 2.1.1.

Figure 2.1.1: University of Kansas Electric Shuttle Bus

This will serve not only as the validation system for the data acquisition system, but provide a full scale platform to continue research into large vehicle electrification [4].

2.2 *Design Requirements*

The first step in maturing this vehicle platform is to develop a data acquisition system that can characterize the current technology implemented and track the improvements of more advanced technology as upgrades are made. There are several requirements to implement an effective system that will support not only the current needs of the research, but include future plans as well. These include: isolation from the vehicle systems, ease of use, ease of maintainability, expandable, high performance, and portable to another vehicle. These serve as high level requirements to the system design. More specific requirements are discussed later. It was also vital to consider that the system should be able to feed a mathematical model used to predict a variety of characteristics to aid in making design decisions in the future [4]. This means it is also important to have deterministic and time aligned sampling so as not to introduce timing errors when calculating models. Next, further justification for these individual requirements will be given.

The data acquisition system needs to be isolated from the vehicle systems for several reasons. First, it is desirable that the system simply observes the behavior of the vehicle, but does not influence its behavior. If the data acquisition system is electrically coupled to that of the vehicle it will introduce extra loads that would not be present in a production setting giving less accurate results. An isolated system is also beneficial because it can more easily be changed, removed, or implemented into a different vehicle. This criterion also affects the portability of the system. As various components are upgraded, it is imperative that the new

components can easily be incorporated into the system. This will allow much faster implementation time and help to ensure a more consistent system when making comparisons.

Ease of use and maintainability are also vital since the system is likely to be supported for a considerable period of time. A system that is easy to use from an interface standpoint allows researchers to perform tests without the need for an intimate knowledge of the data acquisition system. This will also aid in increasing the efficiency and speed at which research is performed. Undoubtedly, the system requirements are going to evolve with the research so a system that is simple to learn and maintain is crucial.

High performance and expandable can mean a variety of things, but for our purposes it is referring to throughput of sampling and the number of samples capable of being taken. Again with the anticipation of a changing project is the need to allow room for growth in the system in the form of extra channels of acquisition for common sensors. Next, the two systems that were considered will be introduced as well as more detailed specifications of the systems.

2.3 Data Acquisition System

In general, the system is broken down into two pieces. First is the embedded system that takes care of interfacing to the various sensors and doing some preliminary signal processing. This embedded system then transmits the data to a computer that decodes the data, displays it on three monitors mounted in the rear of the vehicle, and saves the data to a file for post processing. Two versions of the data acquisition system were developed both on the computer portion and the embedded portion. The first was based around a custom designed PCB built around a PIC32 microcontroller and a custom interface via USB programmed in JAVA on the computer side. The second was built around a CompactRIO from National Instruments and LabVIEW software on the computer side.

The first system to be discussed is the embedded system that was designed around the PIC32MX460F512L microcontroller. This microcontroller includes a variety of serial communication interfaces including SPI, UART, I2C, parallel interface, and several memory options. It also included 512 KB for program memory, 32 KB RAM, 12 KB of auxiliary flash for configuration setting storage, and a USB OTG interface. These were all desirable features as the wide selection of interfaces matches those that are typically found in automotive sensors. Two that were not present were a sufficient A/D converter and CAN. The PIC32 includes a 16 channel 10 bit 1 MSPS onboard A/D converter. This was not used because 10 bits did not provide enough resolution to measure the analog sensors to the desired precision. Although CAN is one of the most popular automotive interfaces, none of the sensors that were implemented in the system communicated via the CAN bus. Therefore, a CAN bus was not implemented in the system. It is relatively straight forward to add one by selecting from a variety of transceivers that communicated via another serial protocol such as UART or SPI.

The MAX1132 was the A/D converter chosen for the system. It is a 16 bit 200 KSPS bipolar chip. This provides several advantages over other options. It operates from a single 5 V supply, but can measure either 0 to 12 V or $\pm$12 V ranges. This eases the implementation circuitry by not requiring an external resistor divider and voltage bias circuits. It also provided the sufficient precision of 0.366 mV/bit for the $\pm$12 V range. To increase the number of available channels, a DG406 analog multiplexor was placed in front of the A/D to provide 16 single ended channels. It was anticipated that some current shunts would be used to measure a variety of low voltage bus currents, so the system also included four AD8293G160 fixed gain amplifiers. This instrument amplifier had excellent noise properties and rail-to-rail operation with a fixed gain of 160. This gain was chosen to allow shunts up to $\pm$ 31.25 mV output to be

used in the system. This does not make use of the full scale range of the A/D, but there was a trade-off between board space of adding additional amplifiers for each of the four current measurement channels and the reduced precision. For a shunt rated to measure 10 A and output 31.25 mV results in a resolution of 1.464 mA/bit, which is sufficiently precise for all the measurements taken on the vehicle.

Next on the board was a MAX6675 cold joint compensated type K thermocouple digital converter. This allowed type K thermocouples to be easily converter to a digital value with guaranteed accuracy to 3 ºC. It reads with a resolution of 0.25 ºC from 0 to 700 ºC. The high end of the measurement range is well beyond the operating conditions that would ever be seen in the vehicle, but the low end may cause some problems. As this vehicle operates in Lawrence, KS it is not improbable that temperatures would drop below 0 ºC in the winter months. This does not cause a component failure, but it does need to be understood that the guaranteed accuracy does not extend to temperatures that cold. Again, a single thermocouple interface is not sufficient for the entire vehicle, so an ADG709 quadruple differential multiplexor was placed in front of the MAX6675 to allow up to four thermocouples to be connected to the system. This multiplexor has very low resistance (< 3 Ohms), which is important to switch the thermocouples because their raw voltage output is 41 μV/ºC. The low resistance minimizes the voltage drop through the switch reducing the noise introduced into the measurement caused by the multiplexor.

Several different voltage rails were required to supply power to the various components. The MAX743 supplied $\pm$ 12 V to power the bipolar multiplexor to the A/D converter. The +5 V rail was supplied by LM2940-5.0 LDO regulator. That rail also fed a MAX828 which inverted

the rail and supplied -5 V for the current shunt instrument amplifiers. Finally, a MC33269 supplied +3.3 V for the PIC32 and other peripherals.

The communication interface to the computer system was USB as this is a common standard. A FT232RL UART to USB chip was used to not only ease the interface from the embedded side, but can be detected as a serial COMM port making the computer side interface easier as well. This interface is typically limited to a maximum baud rate of 115,200 baud so that was the chosen communication rate.

This design had a variety of appealing advantages, such as, very cost effective (<$200) compared to off-the-shelf solutions costing in the thousands of dollars. The custom board also had a small form factor fitting in a 5” x 2.5” footprint as seen in Figure 2.3.1.

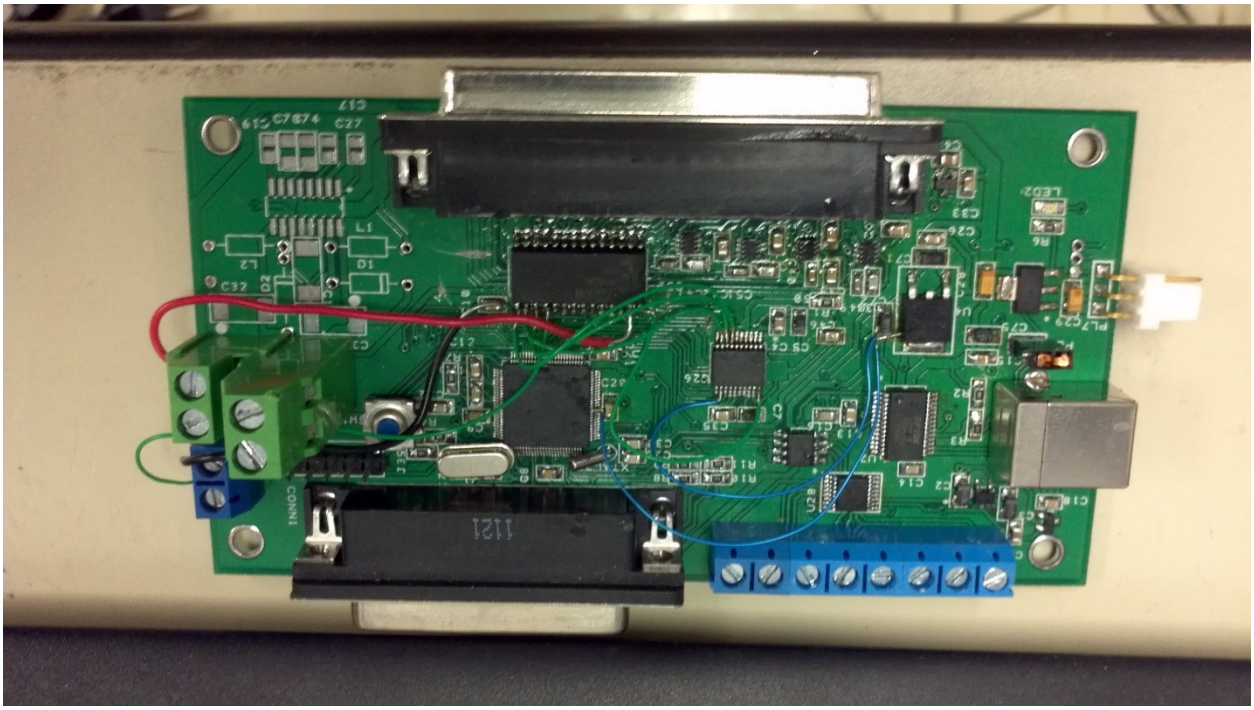

Figure 2.3.1: Custom Designed Electric Bus Data Acquisition System

It did present some disadvantages compared to an off-the-shelf solution including: more difficult to implement the software, more difficult to change to include unforeseen features, and more difficult to maintain software. The custom solution required several applications, all the device

drivers, and a communication schema be developed. This resulted in a more significant programming effort than an off-the-shelf product that includes some code and drivers. Also, the board was designed to support extra features without needing revision, however if for instance a CAN sensor was implemented it would require designing a new board to incorporate it because there is not a CAN transceiver on the custom board. This would require more effort to update the schematic, re-layout a PCB, and could require multiple revisions to get new features working robustly. Although this solution is still inexpensive in terms of hardware cost, this requires a significant amount of researcher time which would ultimately slow the ability of the system to adapt to changes. Finally, this system is harder to maintain because making firmware changes require an in-depth knowledge of the current application to implement new features without sacrificing the performance and reliability of the current features. Again this is not preferred because the system should support basic changes with minimal effort to allow more researchers to use the system and focus on their research rather than programming. Ultimately, it was decided that this system simply did not meet enough of the original design criteria. Therefore, commercial options were explored and a product line from National Instruments was chosen.

The CompactRIO was designed to allow users to have maximum flexibility when choosing peripherals for their system. It is also aimed at making this an integrated and easy process, as well as providing rugged systems that can stand up to a wide variety of environments [5]. The cRIO-9014 was the controller chosen as the heart of the second system and is shown in Figure 2.3.2.

Figure 2.3.2: NI cRIO-9014 Controller with 8 Slot Chassis
http://www.dbm.com.vn/?page=products&catid=211

It is an ultra-rugged, high performance, controller that included a 400 MHz PowerPC hard core processor, 2 GB nonvolatile memory, 128 MB DRAM, and an 8-slot Virtex-5 LX 30 FPGA for the reconfigurable peripheral portion. To provide communication the cRIO-9014 has 10/100 BASE-T Ethernet, RS-232 port, and a USB Host port for external memory devices. A major benefit to this system was the ability to buy I/O modules and simply plug in the desired modules to the chassis. This connected through a PCI bus on the backplane of the chassis and could be controller with the real-time controller, the FPGA, or a combination of both.

The analog module chosen was the NI 9205. It is a 32 channel 16-Bit 250 KSPS converter that has a configurable gain to adjust the input range from $\pm$ 200 mV to $\pm$ 10 V. This was a much needed feature because it allowed the gain to be programmatically turned up for the current shunt channels to increase the precision of the measurement by utilizing more of the modules full scale range without requiring any external circuitry. The digital module was the NI 9403. It was a 32 channel 5V TTL bidirectional module. This module was used for basic switching I/O functions as well as connecting serial interfacing sensors. This module provided flexibility to connect to any sensor that a microcontroller would be capable of connecting to. Among other options, a CAN transceiver was available as one of the modules. It was not implemented do to the lack of requirement to have one, but as there is a reasonably high probability that it will be needed at some point and it will be easy to buy the module and simply

plug it into the chassis. Finally, the NI 9211 was used to interface the thermocouples. It is a four channel thermocouple input module that supports a variety of thermocouple types. The module is also accurate to 1.7 ºC over the range of -40 to 70 ºC, which is an improvement not only in guaranteed accuracy but the range extended below 0 ºC. The final component that needed to be addressed was the onboard RS-232 port. This was used to communicate with the GPS module, as will be discussed later, but the GPS unit was unable to directly connect to the port as it followed the RS-232 standard and the GPS module communicated with a +3.3 V TTL UART. Therefore, a simply interface board was build that included a MAX232 chip to bridge the GPS UART output and the cRIO RS-232 input. Next, the various sensors that were implemented in the vehicle will be discussed.

2.4 Sensor Selection

The first group of sensors installed was various current measurements. Hall effect sensors provide a means to obtain a non-contact current measurement. They work on the principal of sensing the induced inductance created by the current flowing through a wire. Therefore, these sensors are installed with the wire running through them, but they are not part of the circuit. Two hall effect sensors were installed on two phases of the induction motor. The third phase does not require a measurement because it can be calculated from the other two phases. Another hall effect sensor was used to measure the main battery pack current. The compressor for the air system was the original and suspected to be a significant load on the battery pack so a hall effect sensor was installed to measure the current drawn by the compressor. Two more hall effect sensors were placed in the inputs to the DC/DC converters. A major reason for this selection is to be able to obtain an estimate of the efficiency of the converters. Finally, the bus front panel gauges and lights were the originals and require

significantly more current than today's LED lights and low power gauges. To track the gains new gauges will provide, a hall effect sensor was also installed on the main power line leading to all of the front panel systems. Using hall effect sensors was a necessity to keep high current and voltage systems decoupled from the data acquisition system for protection.

For low voltage and lower current readings current shunts were used. These sensors do require the circuit to be broken and placed in series in the circuit. This violated one of the original design criteria by not being completely decoupled from the vehicle itself. Due to the lower cost of the current shunts, these were selected as a tradeoff. They could easily be upgrade to hall effect sensors to achieve the original criteria. Current shunts were installed to measure the output current of both DC/DC converters that step down the battery pack voltage to +13.8 V used for subsystems in the vehicle. The vehicle also has a 12 V auxiliary battery to start the vehicle, so a current shunt was also used to measure that batteries current. Another current shunt was used to measure the data acquisition systems batteries. This was used to detect if power to the data acquisition system was dropping to low. In the event of this happening, the system was designed to enter a controlled shutdown to make sure the data was saved and closed in a known manner. The final use of this measurement was to empirically find the power consumption of the data acquisition system as a whole and calculate the needed amount of battery capacity to run for a desired amount of time.

Next, the main battery pack voltage needed to be measured. This was done by using a high power resistor to step down the voltage and then another type of hall effect sensor was used to measure that voltage. Careful attention to the heat dissipation of this sensor was vital. The entire sensor was installed in a custom aluminum box with the power resistor connected to the box with thermal compound. Bolt patterns were also drilled into the box to support a small fan

in case forced cooling was required. It was shown that forced cooling was not required with the aluminum box.

In addition to voltages and currents, three temperatures were measured all with type K thermocouples. These were outside temperature, inside temperature, and the motor compartment temperature. The outside temperature was measured not only to provide data to compare tests with ambient temperature, but also to eventually use this measurement to make smarter control decisions. The inside temperature was primarily taken for use with the plans to upgrade the air conditioning and heating systems. This ensures that the inside vehicle temperature can be controlled to keep the vehicle comfortable for passengers. Finally, a thermocouple was placed in the motor compartment near the inverter. This was done because a serious pitfall of the current vehicle is an inverter over-heating problem. We wanted to deduce whether this problem was contained to purely the inner components of the inverter, or if the entire motor compartment was increasing in temperature. This was important information because two solutions arose to either increase the liquid cooling system size for the inverter or increase the air flow in the motor compartment. This will be discussed further later in this paper.

Other sensors include a GPS module to track where the vehicle was and had been, as well as, used to coarsely calculate speed of the vehicle. Lastly an inclinometer was installed because a major design goal for the vehicle was the ability to climb hills without a significant reduction of speed.

2.5 Software

The software for the embedded system and the computer system were programmed in LabVIEW. The embedded system took advantage of the new LabVIEW FPGA module to implement the serial protocols to interface the various sensors. This provided faster access to the

digital module than using the real-time controller, which allowed for performance gains on the order of 10 times faster using the FPGA. Once the device drivers were programmed, the rest of the embedded system was programmed in the real-time host environment to support deterministic timing. This data was collected, processed, and then streamed over the Ethernet connection to the computer.

The computer side then grabbed the data packets and decoded the packets. These were then streamed into shared memory between three monitors. The three monitors were all mounted in the rear of the vehicle as shown below in Figure 2.5.1.

Figure 2.5.1: University of Kansas Shuttle Bus Data Acquisition System

The middle monitor was designed to look like a normal gauge panel a driver would see when driving the vehicle. This included a speedometer, tachometer, temperature gauges, and the main controllers to start and stop acquisition. The center monitor also includes the main battery pack voltage because this can be used as an estimated gas gauge for the lead acid batteries. Finally, the incline information is display by a sliding bar.

The left monitor was programmed to show detailed information about the sensor data streams in several forms. First, there is an option to see the data streams as either graphs or as

gauges with green, yellow, and red zones. The graphs allow the researcher to observe the data as it is streaming over the last approximate minute. This feature makes it easier to detail fast events or hold enough information if a researcher is looking for a particular event to occur. The gauges provided a higher level view of the data and by properly setting the green, yellow, and red zones make it easy to see values that are approaching damaging levels. This view was necessary due to the large amount of information being streamed continuously.

The right monitor displays information regarding location, slower occurring events, and a customizable graph. First, the GPS data latitude and longitude are parsed and plotted in a graph on the right monitor. The idea is a future improvement would be to connect this data to a google maps interface and plot the real route as it is occurring. Surrounding the coordinates graph is a variety of fill gauges. These are used for displaying slower occurring information such as battery capacity for the main pack, auxiliary battery, and the data acquisition battery. Finally, on the right monitor is a customizable graph. The graph supports a drop down menu for both the x and y axis that contains a list of all of the data streams. These can then be selected and it will update by plotting any two selected variables against each other. This allows the researcher to view potential correlations in variables. Plotting against time is also an option on the x axis only for completeness, but is a little redundant as most variables are plotted on the left screen against time.

The final feature implemented on the computer side of the system was saving all of the raw data to a text file for post processing. The next section will discuss in more detail the tests that were run and present some examples of this data.

2.6 Test Procedures

Two test tracks were used to test the vehicle. This allowed tests to be conducted with as consistent routes as possible. Because drive cycles have variability even when driving the same route, driving different routes would have added more complexity to the analysis when attempting to compare hardware performance. The first track was a short square loop track. It was approximately 1200 feet total and had very low inclines (< 3% grade). This track was used to make a large number of loops, also helping to average out the variability between individual laps.

A second course mapped was an approximate 4000 feet rectangle that included a short but steep hill with a grade of 14%. This course had a few advantages of over the first course. First, it had two longer straight sections that allowed the vehicle to be driven at speeds that correspond with normal city driving conditions. Second, the hill provided data on the vehicles ability to climb a relatively steep grade. Thirdly, driving the course in reverse allowed the vehicle to be coasted down the hill and strong regeneration data was attainable. This is an important future requirement as there are plans to recapture more regeneration energy.

In general, the test procedure for acquiring data remained the same. First, the data acquisition system was powered on and recording was started. Then the vehicle was turned on and the test was run. Then the vehicle was parked and turned off. Finally, the data acquisition stopped and the data was saved to an external memory device for post processing. The results from the tests presented in this paper are primarily for operational validation of the data acquisition system.

2.7 Results

The data below is shown in its raw form. These tests were meant to evaluate the overall operation of the system before adding any extra post processing software filtering. By characterizing the system this way the first upgrades will be to help improve the hardware. This makes the system more robust, and after that then software filters can be applied to smooth the data. The data below is a trial run from the shorter course driven for five laps. Additional data from other tests can be found in Appendix A. Figure 2.7.1 shows the main battery pack voltage during the test.

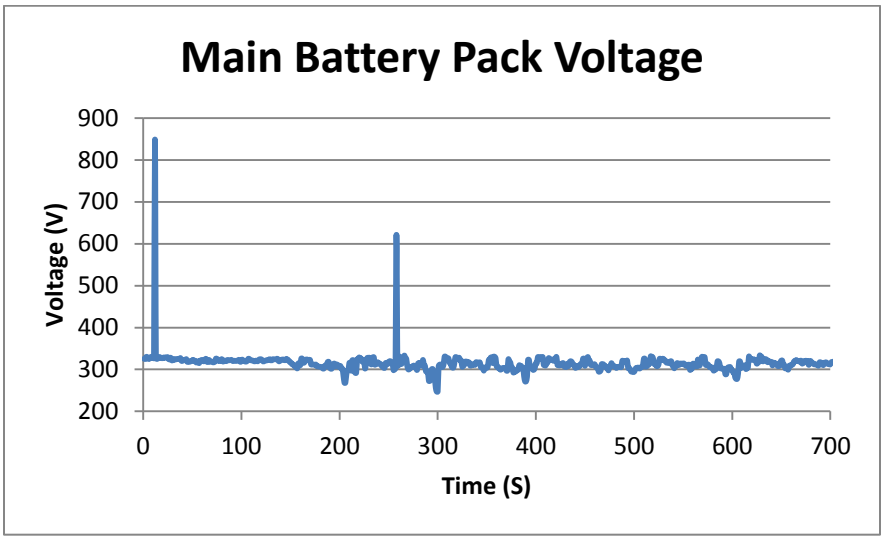

Figure 2.7.1: Main Battery Pack Voltage

It can be seen that there were two spikes in the battery pack voltage measurement. This is likely to be due to noise in the wires from the sensor. With the exception of those two points, the data follows an expected trend. Overall, the data is trending downward since these are lead acid batteries that have a faster voltage drop as they are discharged compared to newer technologies. The final voltage averaged 12.7 V lower than the starting voltage value. The dips in the data are also expected because as the system comes under high load, such as when the electric motor

starts, the voltage dips momentarily. The Figure 2.7.2 shows the main battery pack current measurement.

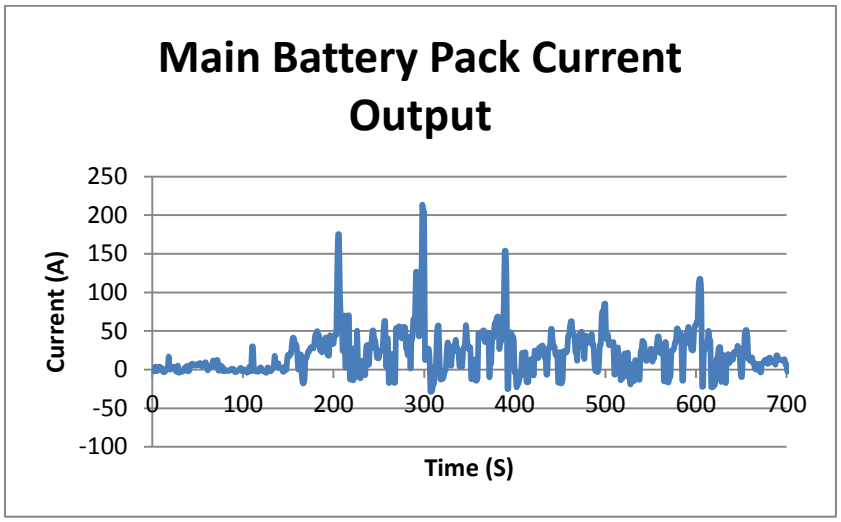

Figure 2.7.2: Main Battery Output Current

The battery pack current provides several interesting results. First, it can be seen that negative current is occurring, so when the vehicle is coasting it is regenerating energy. It can also be seen from the graphs this regeneration is a relatively low amount of recaptured energy. A second observation is the spikes in the current values. Several of these values have multiple non-repeating data points in the region, which suggest they are neither outliers nor noise. This observation is further support below as the spikes in the battery pack current are time aligned with the spikes in the motor current shown in Figure 2.7.3.

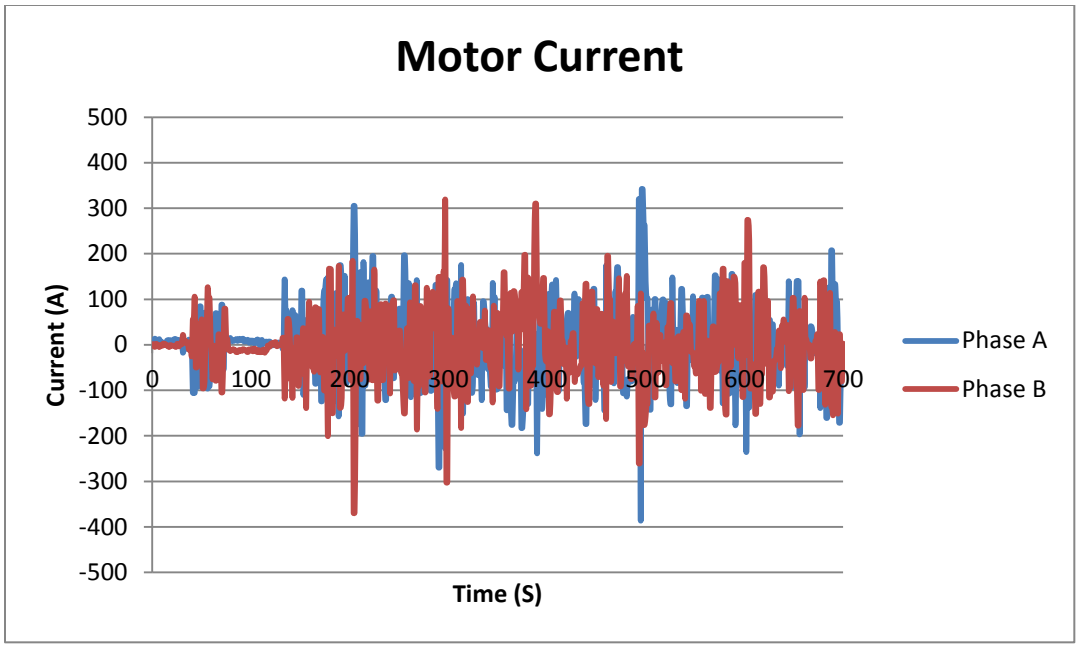

Figure 2.7.3: Phase A and B Motor Current

The motor current appears at first to have a significant amount of noise. It is important to remember that the sampling rate of the system is 10 Hz, so we are not even close to 10 times the motor current frequency required to accurately measure the current frequency and amplitude according to Nyquist frequency and Shannon's sampling theory. Therefore, it was not expected to be able to recreate perfectly the current waveform. This measurement is taken to get a general average of motor current usage after the inverter and potentially to be used for control in the future with a faster acquisition rate. Figure 2.7.4 below is a zoomed in window of values taken from the beginning of the test.

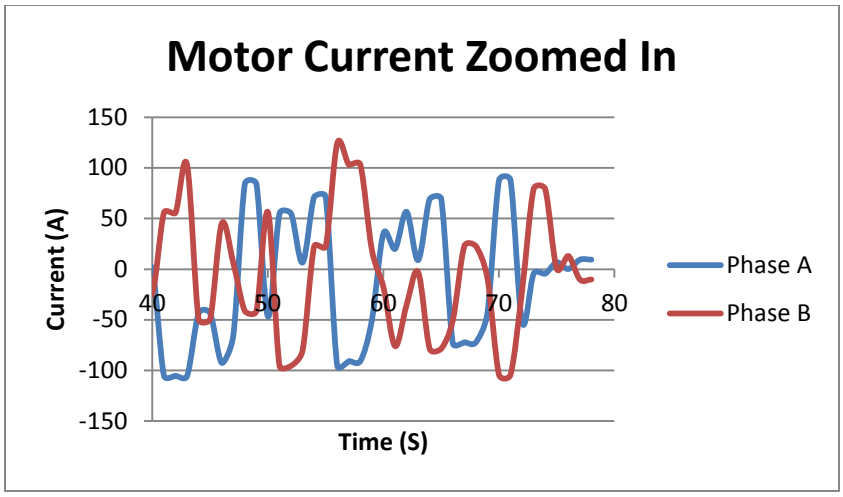

Figure 2.7.4: Zoomed in Motor Current

To validate the measurement, the observation can be made that the behavior of the two phases does seem to be similar but shifted in phase. That is exactly what was expected. Again, the waveforms do not follow precisely due to the low sampling rate.

Next, the DC/DC current input measurements are displayed in Figures 2.7.5 and 2.7.6.

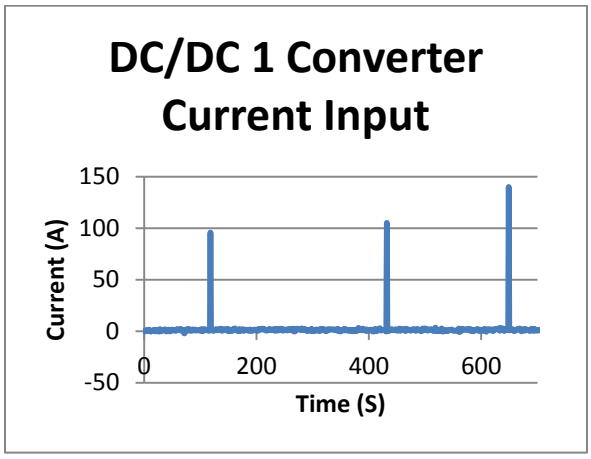

Figure 2.7.5: DC/DC 1 Input Current

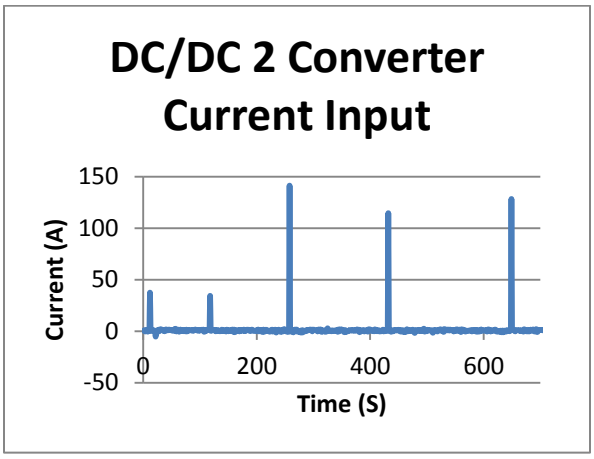

Figure 2.7.6: DC/DC 2 Input Current

Again like the motor current the DC/DC converters show several large spikes. It was noticed throughout testing that when the compressor for the compressed air system starts a current spike can be seen. It is believed that the two spikes are outliers; however, because they are not time aligned with each other and both are bussed together. Three spikes are time aligned which would suggest that there is possibly a vehicle condition that is causing a large current draw,

likely due to the compressor starting. The output side of the converters is shown in Figures 2.7.7 and 2.7.8.

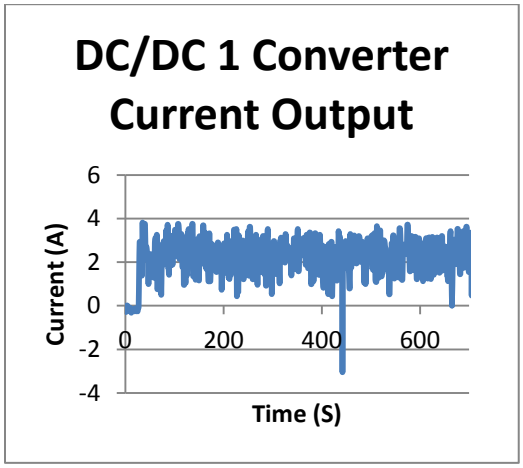

Figure 2.7.7: DC/DC 1 Output Current

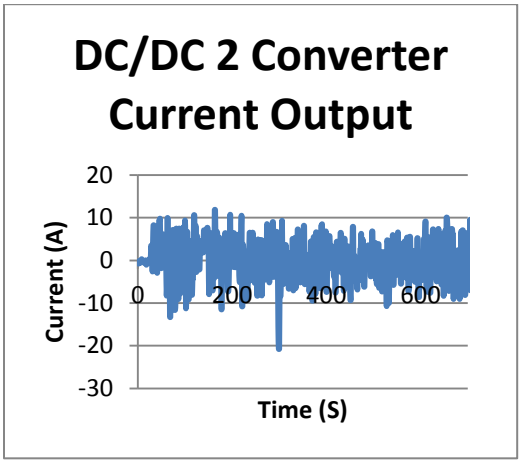

Figure 2.7.8: DC/DC 2 Output Current

The output side of the DC/DC converters shows a relatively steady current response. The measurement does appear to have significant amount of noise. This is likely due to the fact that these sensors are the current shunts rather than the hall effect sensors. This causes noise in the measurement because it is a low voltage signal (± 50 mV max), and it is connected to the data acquisition with 5 to 8 feet of wire that run through the vehicle. This signal is then amplified by the data acquisition system, so any noise introduced from the sensor to the system is also amplified. Several alternatives to help alleviate some of this noise will be discussed in the recommendations.

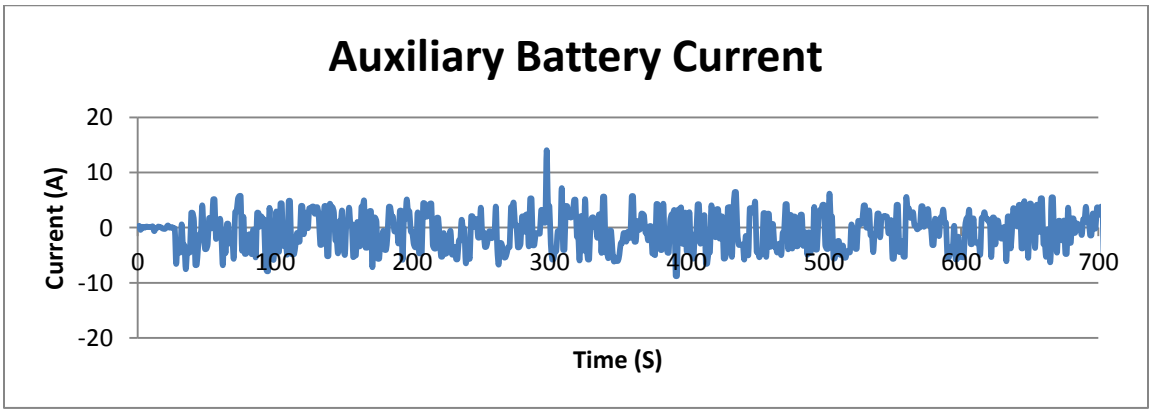

Figure 2.7.9: Auxiliary Battery Current

The auxiliary battery behavior, in Figure 2.7.9, displays similar noise characteristics as the DC/DC converters output, and this is also likely due the current shunt sensor. The behavior of this signal varies depending on the charge level of the battery. Because the auxiliary battery is bussed together directly with the DC/DC converters it can be in one of several "modes". First, is full charged (around 14 V) and the graph will show positive current leaving the battery. The second state is almost full charged where the open circuit voltage of the battery is very close to the output of the DC/DC converters. In this case, the battery will either be float charging or if the load on the bus suddenly increases it will help make up the current draw for the voltage dip. Thirdly is a partially or mostly discharged battery state. During this state, the auxiliary battery has enough capacity to start the vehicle and as soon as it does the DC/DC converters begin charging the battery and the data is seen as negative current flowing into the battery. This becomes important because the DC/DC converters need to be slightly oversized when designing to account for the fact that they may not only be supporting the worst case load on the low voltage system, but may also be charging the auxiliary battery. This test had a state that was lower than the bus voltage because the average current was -0.62 A. The next measurement was the compressor input current shown in Figure 2.7.10.

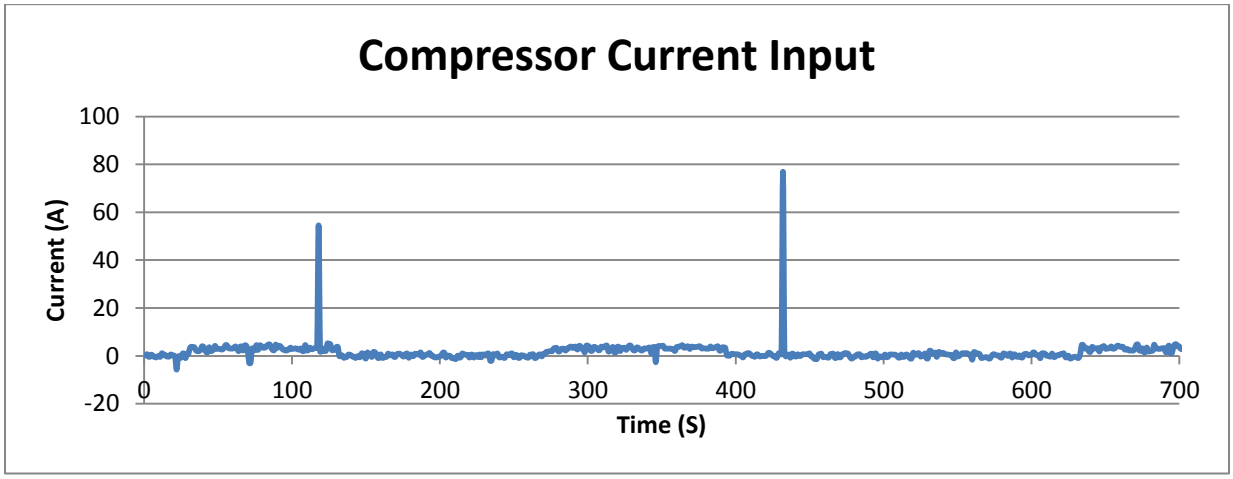

Figure 2.7.10: Compressor Input Current

The compressor is used to keep the air system charged for the air brakes and suspension system. Not surprisingly current spikes are observed when the compressor turns on. The relatively low current draw is deceiving because it is powered from the high voltage side, so the overall power consumption is substantial. Another parasitic load on the system is the front panel gauges and display. Their current input graph is shown below in Figure 2.7.11.

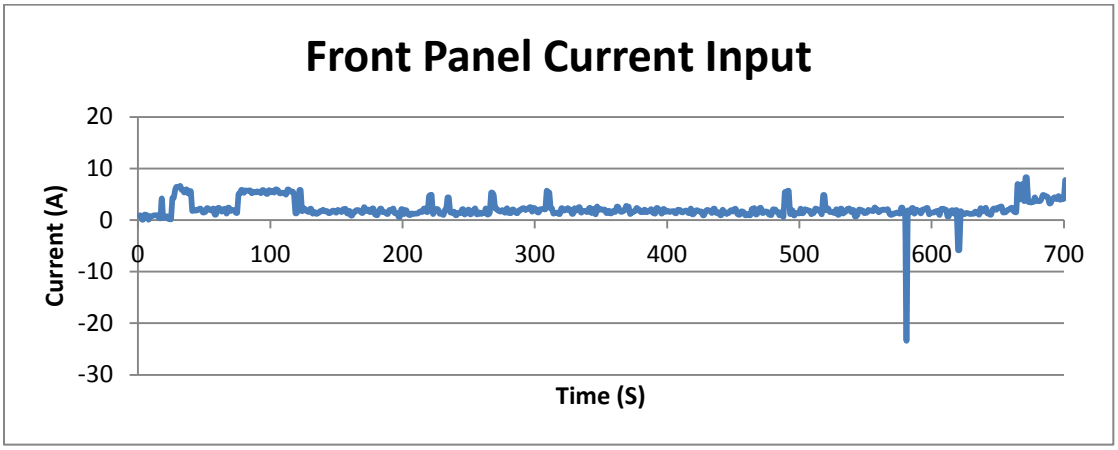

Figure 2.7.11: Front Panel Input Current

It was expected that the front panel gauges would consume a large amount of energy because they are older technology lights, gauges, and peripherals. It was decided that it was unnecessary to have finer grain measurements than simply the total current consumption because these will be replaced with low energy lights and gauges in the future, and it is only interesting to know the overall energy savings. The front panel does show two low spikes in the data that have no explanation. Because the front panel consists of only parasitic loads, it is not possible for the front panel to put energy back into the system. Although this measurement is taken with a hall effect sensor it does run almost the entire length of the bus (22 feet), so there are additional noise reduction considerations to try and eliminate those erroneous spikes. These considerations will be discussed in the recommendations section.

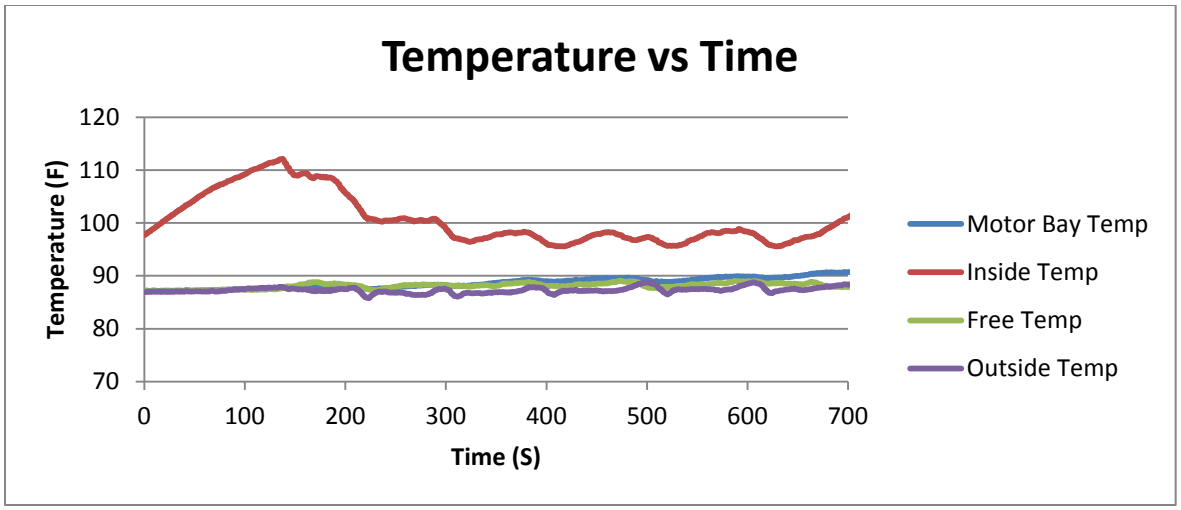

Figure 2.7.12: Temperature Measurements

The four thermocouples were measured and the results are shown in Figure 2.7.12. They behaved as expected and the readings were validated with weather data corresponding to the time and location. A very important observation is the original hypothesis of the motor bay heating being the cause of the inverter overheating can be disproven from this data as the temperature did not increase significantly. This shows for certain that the overheating is contained within the inverter, likely from the IGBTs. A final note would be the vehicle would benefit from an air conditioning system as the temperature suggests this was not a particularly comfortable riding experience.

Now that the internal measurements have been discussed the measurements pertaining to the vehicle as a whole will be discussed. This mainly consists of data that is read and parsed from the GPS module. The first chart, Figure 2.7.13, shows the vehicle speed as measured by the GPS.

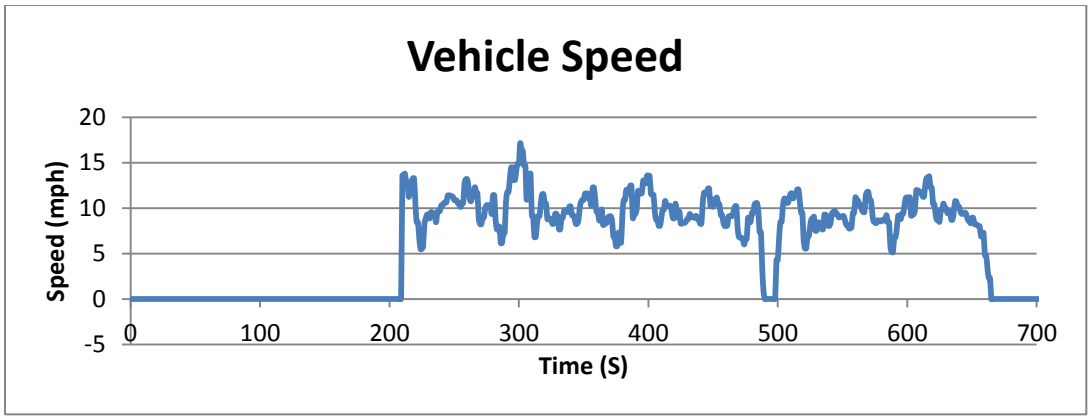

Figure 2.7.13: Vehicle Speed

Right away it is important to mention that this test as described previously began taking data before moving the vehicle. This is relevant because the vehicle started in a metal garage, and the GPS was unable to lock a satellite until it left the garage. After leaving the garage, the GPS module still required a significant amount of time before finally locking a sufficient satellite signal. Despite the pitfalls, the GPS module was accurate to within a few miles per hour over its update rate of 1 Hz. This was validated by simultaneously watching the speedometer and the GPS read out for dynamic scenarios. To provide a more scientific validation, the vehicle was put into cruise control on the flattest portion of road and the GPS value was compared to the vehicle speed and was found to be within 4 mph at all times. This resolution is sufficient to build drive cycles to feed a simulation model. The next measurement captured from the GPS module was the course of the vehicle in degrees as seen in Figure 2.7.14.

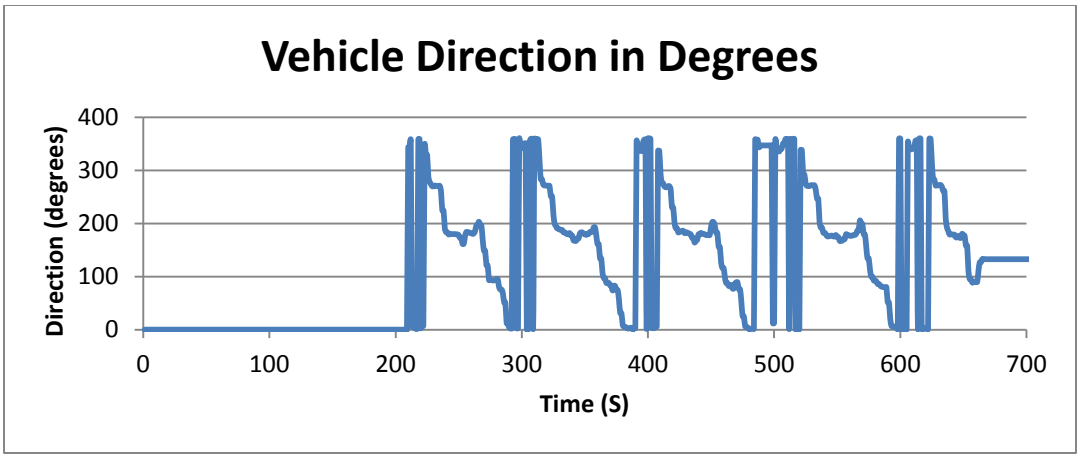

Figure 2.7.14: Vehicle Direction

The course is measured where compass north is at 0 or 360 degrees. This measurement was primarily used to provide the compass direction indication that many vehicles now incorporate. It was captured as it may be useful in future research if the speed and direction of the vehicle are required.

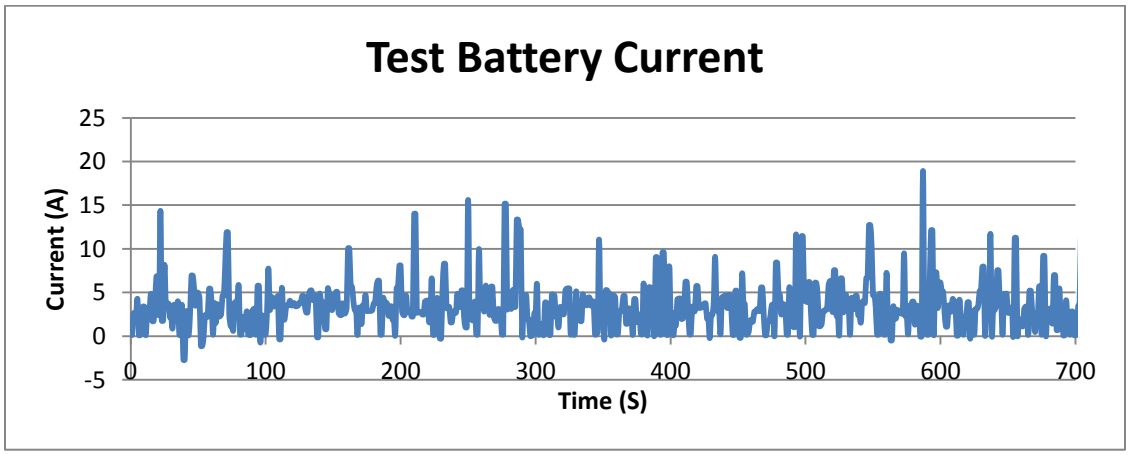

Figure 2.7.15: Test Battery Current

The last measurement is the test battery current, shown above in Figure 2.7.15, which is decoupled from the vehicle electrical system. It can be seen that the power draw from the test battery was significant. This is due to the fact that it was powering three monitors, a computer, and the data acquisition system. A major update needed is a high capacity source for the test system. It was consistently seen that the data acquisition system power ran out before the vehicle power ran out. This does not allow endurance testing on the vehicle if the data

acquisition system cannot outlast the vehicle. Fortunately this measurement provides the necessary data to calculate the amount of energy needed to fulfill that requirement, and additional batteries can easily be added in parallel to support the desired run time.

2.8 Conclusions

There were two options that were explored throughout the development of the data acquisition system for the electric shuttle bus. The first was a custom embedded system built around the PIC32 microcontroller. This system offered a large amount of flexibility and significant cost reduction over off-the-shelf solutions. Despite these advantages, the custom system suffered from increased difficulty to implement, maintain, and use. Therefore, a CompactRIO from National Instruments was chosen as the platform to design the system around.

The system also completely satisfies the original design requirements. It is able to support current and future research because there are extra analog, digital, and thermocouple channels for use on future sensors. The system is easy to implement and maintain because it is written in LabVIEW, which is a common language and relatively easy to learn. The system is simple to expand because the CompactRIO chassis allows modules to be plugged in to add additional functionality and then leverage LabVIEW for quick application development. The system has sufficient performance being able to sample with a throughput of 10 Hz. The ability to feed a model with deterministic data is handled through the use of the real-time module.

The only two requirements that are not fully met is isolation from the vehicle, which in turn does not make the system completely portable from one vehicle to another without being invasive. The data acquisition system only violates this criterion with the current shunts on the low voltage side current measurements. These were chosen over hall effect sensors for cost reasons, but can very easily be upgraded to fully meet the isolation requirement. Although the

system meets the design goals, there are some areas of improvement that will help the system be more robust in measurement quality.

2.9 Recommendations

The current shunts have the highest amount of noise in their measurement so they are addressed first. Ideally, the solution is to replace these with hall effect sensors because they are more immune to noise and would isolate the measurement. If that is not possible then twisted pair wire with a shield could be used to connect the shunt and the system. The shield will help reduce the amount of induced noise, and the twisted pair wire will typically see noise on both wires which makes filtering it out with a differential measurement more effective. The same wire scheme can be used to reduce the noise in the front panel measurement since it has a relatively long wire length. The inclinometer originally chosen was a magnetometer that uses the magnetic field of the earth to detect orientation in two axes. There was an unanticipated major flaw in this sensor because it assumes the largest magnetic field is the earth. The sensor was installed at the back of the vehicle so the motor was creating a stronger magnetic field and causing huge errors in the sensor. The inclinometer was switched to a fluid version, and future tests will validate that this completely resolved the problem.

The GPS module worked well once it locked a satellite signal, which in many cases took several minutes. Therefore, it is recommended to add an additional external antenna and mount it on top of the vehicle to maximize the antenna gain and minimize the objects blocking line of site of the antenna. This will help the GPS lock a signal faster. The battery system powering the data acquisition system was also found to be insufficient to power the system longer than the vehicle battery system. Therefore, a future upgrade will include calculations to identify the required capacity of the batteries and implement the additional batteries to allow endurance

testing of the vehicle. Finally, the system relied too heavily on the communication with the computer. Occasionally the real time module would lose communication with the computer causing it to lock up and would require a reset to bring it out of lock up. This presented a problem because we lost the data collected when this happened because the computer program also crashed. To fix this problem a buffer should be implemented on a future software version to store data until the communication returns on the real time module side. The computer side should then run the communication in its own thread and the file I/O operations in a separate thread so these are decoupled. This way if communication is locked the data is not lost, and if a restart is required the file can be saved.

2.10 References

[1] Qiang, S. and L. Chenguang (2012). "Data Acquisition System for Electric Vehicle's Driving Motor Test Bench Based on VC++." Physics Procedia **33**(0): 1725-1731.

[2] Baghli, L., et al. (2012). Optimal hybrid vehicle, embedded data acquisition and tracking. Environment Friendly Energies and Applications (EFEA), 2012 2nd International Symposium on.

[3]Hairr, M. E. G., Paul; Bailey, J. Ronald; Madden, Woodlyn (2009). Data Acquisition System for Electric- and Hybrid-Electric Buses. EVS24. Stavanger, Norway.

[4] Baker, B. C., et al. (2012). Electric transit bus for variable grade terrain. Electric Vehicle Conference (IEVC), 2012 IEEE International.

[5] Coosemans, T. B., Ricardo; Timmermans, Jean-Marc; Mulders, Frederik Van; Mierlo, J. Van (2009). Data Acquisition System for Optimization of Series Hybrid Propulsion Systems. EVS24. Stavanger, Norway.

3. EDLC TEST SYSTEM

3.1 Introduction

Researchers have discovered a variety of applications that show improvements in energy efficiency and storage with the addition of Electric Double Layered Capacitors (EDLC) [1-6]. EDLCs are more commonly known by the commercial names of ultracapacitor or supercapacitor. Despite the inconsistency in naming convention these are all referring to the same thing. EDLCs have several properties that are advantageous over batteries, such as, high power density, longer life cycles, faster charge and discharge times, and lower cost per amount of power output as compared to high performance batteries [2,3,6].

One of the difficulties of designing an electric vehicle is finding the balance between power output, energy density, weight and cost. To complicated the problem further these requirements vary in each application. The overall focus of this research will be targeting a retrofit of a larger electric vehicle, more specifically a shuttle bus. Larger electric vehicles present a different variety of problem compared to a car. Cars are typically designed to be very lightweight to minimize weight and reduce the amount of power and energy required to have acceptable performance and run time. When retrofitting a vehicle, the luxury of designing the vehicle from the ground up with lightweight strong materials is not an option. Therefore, a major problem is the amount of power output of the energy system to achieve comparable performance to the gasoline equivalent of the bus.

EDLCs make an excellent supplement to batteries, which have high energy density and lower leakage current. It is important to note that currently EDLCs do not have high enough energy density to be used in many applications as a standalone energy source. They do; however, provide a variety of advantages. A major advantage is they help protect the battery

system by acting as a power buffer. Electric motors have a characteristic large current spike when first starting and this can shorten the life of a battery pack [7]. An EDLC bank is capable of sourcing this short burst reducing the strain on the batteries. EDLCs are also capable of rapidly charging, which make them an ideal candidate to recover regenerative energy. They also have a cycle life on the order of 1000 times that of a current battery pack [8]. This reduces the maintenance time and cost of the vehicle, while also improving the performance.

The test system presented serves a few deficiencies in EDLC implementation tools currently including the ability to verify manufacturer specifications and test control and safety system design. Many designers treat EDLCs and batteries with the same type of control and protection strategy. Although they do have some similar properties, they exhibit very different operational behavior and should have control and safety strategies that are tailored for their operation. The test system will allow a designer to characterize the cell to identify the best strategy to properly implement the EDLC bank, because like batteries they can be damaged if they are not correctly managed. This information will provide the needed data to design a full scale bank to be implemented in the retrofit of the electric shuttle bus shown in Figure 3.1.1.

Figure 3.1.1: Electric Mass Transit Vehicle – Design Target

Characterizing cells will not only allow designers to validate manufacturing specifications, but validate simulation models. Finally, the test system will allow the freedom to test protection

circuitry and charge and discharge strategies in a safer environment before implementing them on the full scale bank. To design a system that meets these requirements it is first important to understand the properties of EDLCs that are of concern.

3.2 EDLC Properties

It can be seen that EDLCs have many operational differences compared to batteries, but something they do have in common is misuse can cause accelerated degradation. To prevent this degradation the cell voltage, current, and temperature need to be monitored [4,9,10,11]. Failing to observe the cell's specifications can cause permanent damage and even cause catastrophic failures in extreme cases.

3.2.1 EDLC Voltage

Cell voltage is the first critical characteristic of an EDLC. As expected, exceeding the rated voltage can cause damage, but simply charging to a lower voltage also comes at a cost. The voltage level has a squared relationship to the overall energy stored in the cell, so undercharging the cell significantly for safety reasons greatly reduces the amount of energy that will be stored in each cell. Therefore, a goal of an ideal system is to charge to the maximum rated voltage without exceeding the value as this reduces the life of the cell [1].

In addition to the voltage level of an individual cell, the voltages of each cell in a pack need to be balanced with respect to each other. Unbalanced cells will cause inconsistent current draw when the cells are discharged and overcharging of some cells when charging. If left uncontrolled, this behavior can worsen over time until cells begin failing [4]. Therefore, it is important to monitor cell voltage not only to ensure safe operating conditions, but to track the degradation and establish pack health over time.

3.2.2 EDLC Current

One of the major benefits of EDLCs over batteries is their ability to charge and discharge significantly higher current without damaging the cells. Typical cells in the 1000 Farad range are able to source and sink hundreds of amps continuously and on the order of thousands of amps for short periods of time (<30 seconds). This is the reason they are excellent in applications that have power spikes or need smoothing. In an electric vehicle application, this equates to reduced stress on the battery pack, which is typically the most expensive component in the system so extending their life greatly reduces the maintenance cost of the vehicle. It also allows the vehicle to have a smaller battery pack and still successfully meet the power requirements, reducing the cost of the vehicle [4].

Though it is possible to apply too much current to or draw too much current from an EDLC this is the least probable fault condition due to their high tolerance for short bursts. It is also likely that the application of an excessive amount of current will cause a different fault condition, such as an over temperature, to occur. One potentially unexpected fault condition is a short circuit fault. Because EDLCs have a very low internal resistance, which allows them to source and sink a large amount of current, some short circuit protection circuitry will think the EDLC is shorted until it reaches a voltage around 0.35V [10]. This voltage value depends heavily on the specific resistance characteristics of the cell, but many times the cells are not discharged below this value during operation because extra steps need to be taken to ensure the short circuit protection does not shutdown due to a fault [10]. In addition to safety, it is beneficial to measure current in and out of the cell to provide feedback to the charge circuitry. Like a standard electrolytic capacitor, EDLCs have an exponential charge time when using a

constant voltage charge source. This can be improved by using a constant current charge, which typically requires the current measurement to accurately regulate.

3.2.3 EDLC Temperature

Another advantage of EDLCs is their relatively stable capacity over a wider temperature range than typical batteries. EDLCs typical operating range is -40ºC to 65ºC, which extends beyond batteries capabilities on the cold end. One concern with this range is the upper range has the potential to be exceeded in a hot environment as the cell is used and heats up. The cell naturally heats up during use, like batteries, due to non-ideal energy conversion and the excess is released as heat [10]. This heat production can vary depending on the amount of current passing through the cell and the number and rate of charge and discharge cycles. This makes monitoring the temperature of the cell critical as well. Since it is not practical to put a thermal sensor inside the cell, the temperature is typically taken on one of the cell's leads as close to the cell as possible. This provides an acceptable approximation of the internal temperature of the cell [11].

Failing to properly regulate the cell's temperature below the maximum rating can quickly cause the cell to begin swelling and breaking down. If the temperature continues to increase the cell will eventually reach an internal pressure that will cause it to vent in an attempt to prevent explosion. If the temperature continues to increase beyond this range the cell will be in danger of melting or in more rare cases explode [11]. These can be potentially very dangerous conditions for people in a vehicle. With proper controls and temperature measurements, it is relatively simply to maintain the cell's temperature within safe operating conditions and avoid hazardous conditions and failures.

3.3 Simulation

A valuable asset when designing a power source is an accurate simulation to predict the results of various configurations. These results can provide high level information such as predicted run time for a system and maintenance cost by predicting degradation over time. This allows the designer to not only test various pack configurations to meet design requirements, but the affect different control strategies have on that pack. This helps the designer optimize the hardware and software before investing a significant amount of money in a prototype. The simulations that accompany the EDLC test system have the added benefit of being done using a free Simulation Program with Integrated Circuit Emphasis (SPICE) program developed by Linear Technologies (LTSPICE IV). This was selected to avoid the requirement of obtaining an expensive software license to future users.

LTSPICE contains an extensive built in standard capacitor model, so this was used as a starting point for modeling the EDLC. There have been several research groups that have worked on refining more precise mathematical equations to model and EDLC, but this is beyond the scope of this research [2,3,5,9]. This system is directed toward designers and implementers who are attempting to more effectively design an EDLC bank, rather than specialized tools to better predict the electrochemical behavior that occurs during energy transfer. A primary goal of this approach is to remain modular so as modeling makes advances, these more complex and more accurate models can be incorporated into this system without affecting the hardware configuration.

3.3.1 Constant Voltage Model

The simplest model to construct is a constant voltage (CV) model for a single cell. The CV model had several aims including: verifying the EDLC cell model, predict behavior with a

CV source, and to validate the accuracy of the model against the data provided by the test system. LTSPICE did not contain an EDLC specific model so the capacitor models were constructed using a standard polarized capacitor model, with a parallel resistor and a series resistor as show in the Figure 3.3.1.1, and data from the datasheets of the specific cells intended to be tested.

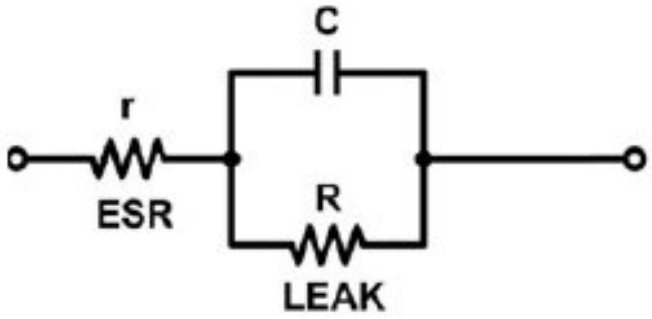

Figure 3.3.1.1: EDLC Circuit Model
http://www.design-reuse.com/articles/34551/tradeoffs-of-ldo-architectures-and-the-advantages-of-advanced-architecture-capless-ldos.html

It is important to note that this circuit configuration will be represented by just a polarized capacitor in the models discussed.

The first cell modeled had a capacity of 0.47 F and rated for 2.7 V. This was data from the UM series EDLC from EVerCAP. The model used a constant +5 V source and a voltage divider to charge the cell to +2.5 V with a limited current. This circuit configuration allows the proper operation of the test system to be verified without the need for protection circuitry because the physical setup will not allow the cell to charge beyond the rated 2.7 V. In addition, the current limiting resistors will keep the current in safe levels, and the combination of relatively low current, single cycle, and slow cycle rate will not cause excessive heating in the cell, so a thermal failure will not be an issue. The schematic of the model is shown in Figure 3.3.1.2.

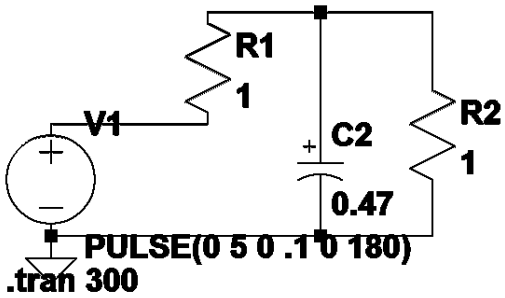

Figure 3.3.1.2: Constant Voltage Model

Next, the model was run through a simulation in which the source was turned on for 5 seconds and then shut off allowing the cell to discharge through the power circuit. The model shows the characteristic RC circuit behavior that was expected from this configuration. It also predicts an approximate charge time of 100 mS to reach 90% of its final rated voltage, but taking a full 5 seconds to reach 98.9% of the final voltage. The output of the simulation is shown in Figure 3.3.1.3 below.

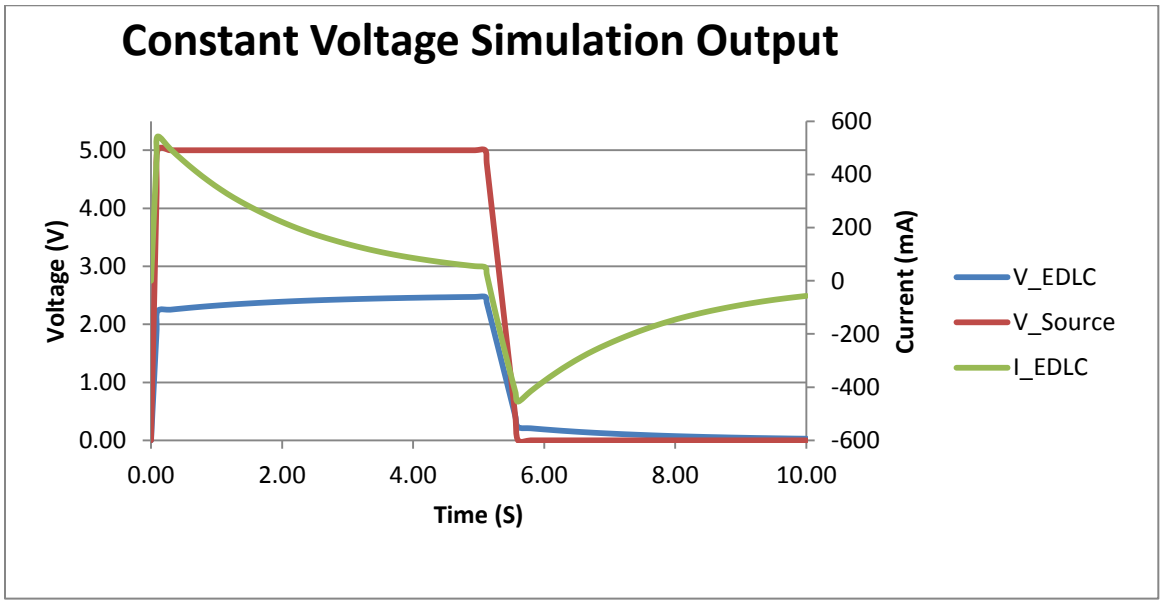

Figure 3.3.1.3: Constant Voltage Simulation Output

This setup will be used to validate the operation of the physical setup. In addition, this will provide a means to evaluate the accuracy of the capacitor model and the overall accuracy of the simulated circuit.

Next, the same circuit configuration was simulated replacing the EDLC with a 100 F 2.7 V cell from IOXUS. This not only provides predictions for the higher capacity cell, but gives insight into the generality of the model for capturing the behavior of different manufacturer's cells. If differences in modeling various manufacturers cells are required these features can be identified and incorporated into the model to support these parameters. The same voltage, current and thermal restrictions are present in this model as the previous. The new model is shown in Figure 3.3.1.4.

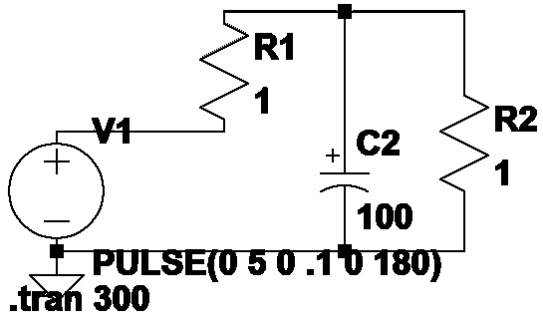

Figure 3.3.1.4: Constant Voltage 100F Cell Model

The output of the simulation is shown in Figure 3.3.1.5 below. As anticipated, the 100 F cell requires a considerable increase in time to charge and discharge. The model predicts it will take 180 seconds to reach 2.5 V.

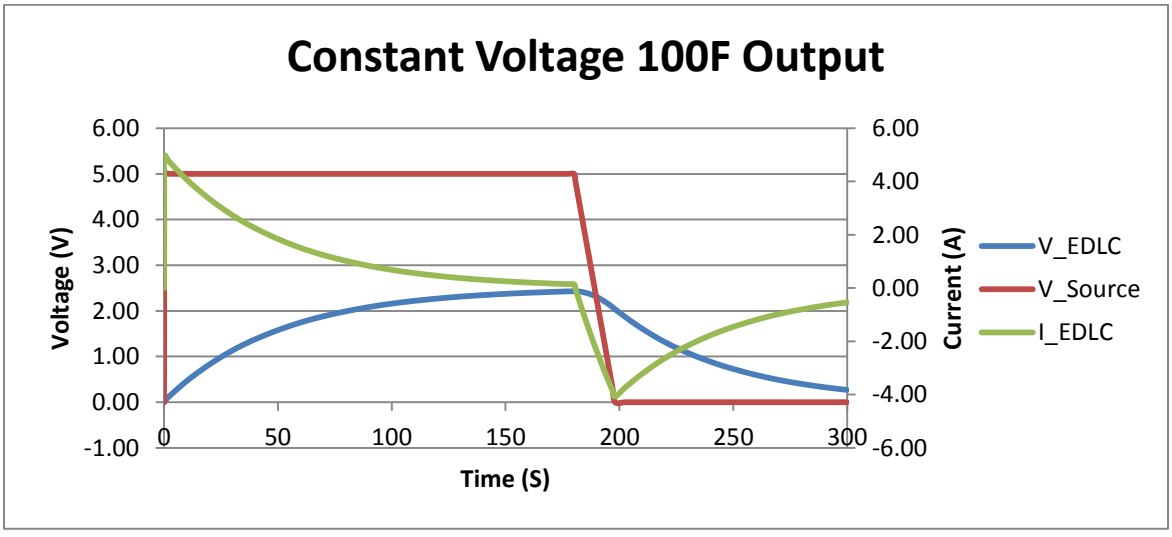

Figure 3.3.1.5: Constant Voltage Simulation 100 F Output

This model can be easily scaled to larger cells and remove the restriction on the current limitation to get the full performance of the cell under a constant voltage source. The constant voltage model is also the simplest to verify with a physical system due to the minimal components required to build such a setup. However, to build a high performance bank of cells a faster means of charging and discharging the cells is required so exponential time is not required.

3.3.2. Constant Current Model

The next model constructed was a constant current (CC) model. CC sources provide many improvements over the CV source in terms of decreasing the amount of time required to charge the cell. The reason for the charging time improvement is a linear charge behavior rather than an exponential behavior as seen with a constant voltage source. This can be shown in the derivation of charging a capacitor with the two different sources. The equation for the voltage across the EDLC with a constant voltage source is shown in Equation 1, while the EDLC voltage with a constant current source is shown in Equation 2.

$$V_c(t) = V_{in}e^{-\frac{1}{RC}t} \tag{3.3.2.1}$$

$$Vc(t) = \frac{1}{C}It \tag{3.3.2.2}$$

For a full derivation for a constant voltage and constant current source refer to Appendix A.

It is important to realize that the CC model allows the source voltage to exceed the suggested rating of the cell in order to keep the current constant. Without proper controls in place, this could lead to overcharging the cell. Fortunately, the physical test system was designed to monitor and prevent this condition from occurring, but the modelling software makes capturing this behavior with built in functionality significantly more complex. To simplify the simulation circuit the time required to charge the cell was found empirically. The model was run

and the charge time was found that would not result in the cell exceeding the rated voltage. This value was then used to drive the on time of the source in subsequent simulation runs. The circuit schematic incorporating the 0.47 F cell is shown in Figure 3.3.2.1.

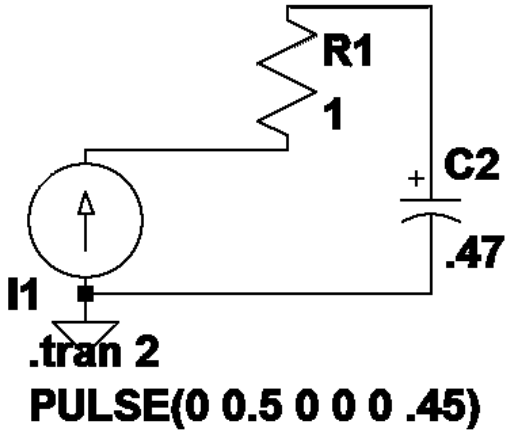

Figure 3.3.2.1: Constant Current Model

The simulation results shown in Figure 3.3.2.2 demonstrate the significant decrease in charge time due to the constant current source. The EDLC only requires 0.5 second with the CC model versus 5 seconds with the CV model to reach its maximum voltage. It can also be seen that the voltage source voltage is lower than the constant voltage model. This is caused because the constant current model only has a single current limiting resistor and does not have the voltage divider that is found in the constant voltage model.

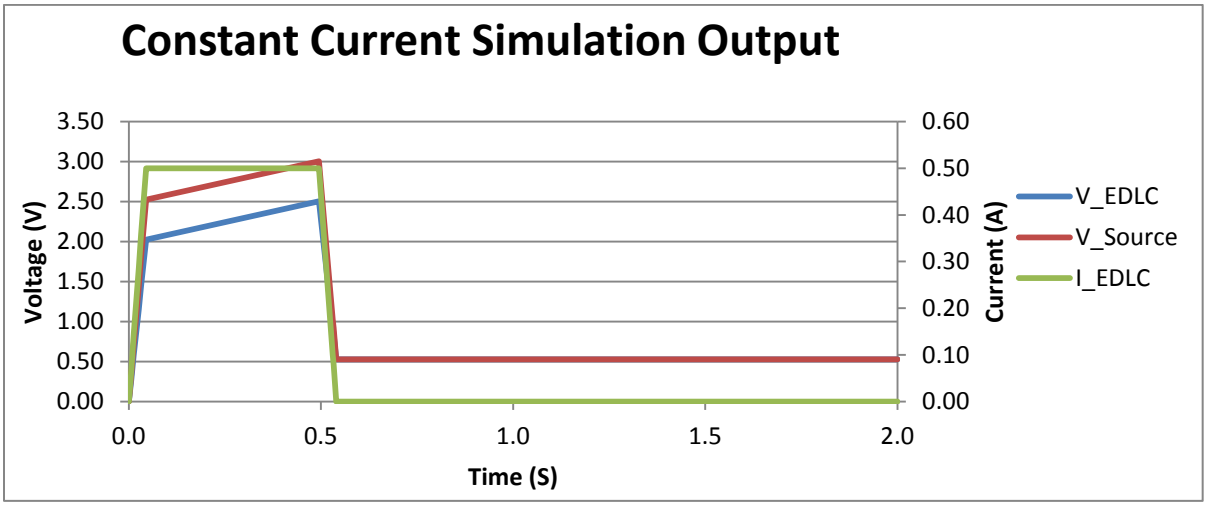

Figure 3.3.2.2: Constant Current Simulation Output

As with the CV model, the CC model was reconfigured with the 100 F cell and rerun. The output from that simulation is shown in Figure 3.3.2.3.

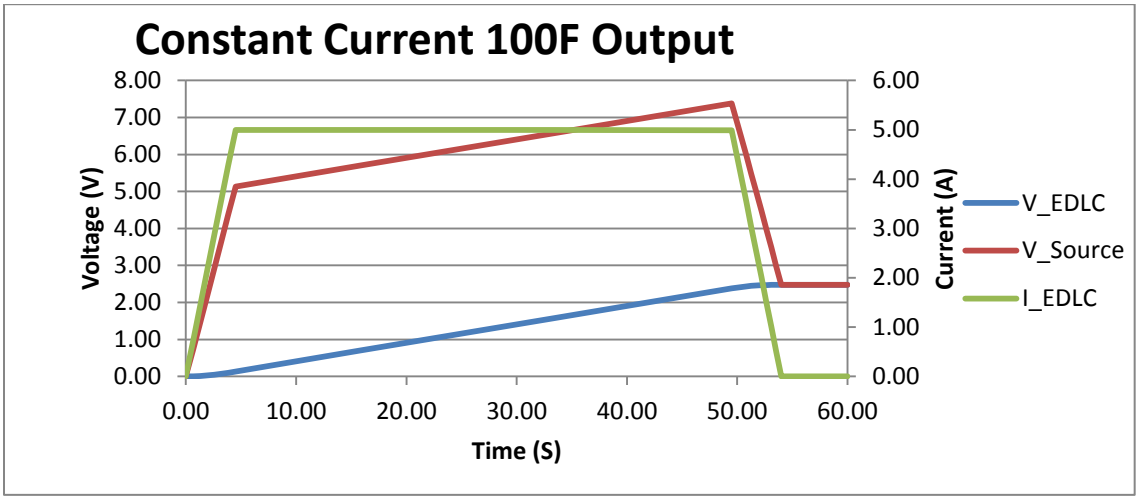

Figure 3.3.2.3: Constant Current 100 F Output

It is also worth noting that these schematics will not be as accurate representing the discharge characteristics of the EDLC if the power supply does not prevent backflow current as this behavior is not modelled in the simulation. This can be quickly fixed by installing a switch or diode to prevent backflow current once the cell is charged in the physical system. Regardless, the primary goal of the CC simulation is to predict the advantage of using a constant current source to charge the EDLC. A decrease in charge time of the 100 F cell is seen to go from 180 seconds for the CV source to 49.5 seconds for the CC source, which is a 3.636 times reduction. This clearly makes a CC source an advantageous choice to test with the physical system.

In addition to demonstrating the power of the CC source, these models provide an approximation for the measurement range that will be required for the various sensors to be built into the test system. Also, the methodology of comparing different sources and even sinks has been demonstrated and can be applied to future tests such as a constant power source or a constant current followed by a constant voltage charge algorithm.

3.4 Physical Test System

So far, the simulation models serve as a tool to estimate the behavior of the EDLCs, but to validate the accuracy of the models and evaluate the real cells a test system has been designed and implemented. The modular design contains one part for handling the data acquisition to monitor the characteristics of interest, and another portion that is capable of driving a source for the EDLC. This configuration allows a tester to change a measurement sensor, the EDLC cell, source, or load all independently of each other. This allows for the widest range of tests and maximum flexibility. A modular configuration also supports test to be run to evaluate specific properties or the cell as a whole without affecting the rest of the system. Another feature of the design is isolated measurements from the test circuit. This is an imperative feature to protect the measurement circuitry from the potentially large current in the test circuit. By incorporating isolated measurements, the measurement side can also support onboard sensors as well as external sensors, which allow this system to change the operating range easily and scales from small cells to cells over 1000 F with only minor modifications.

3.4.1 Hardware Configuration

The test system is based around the Parallax Propeller microcontroller. The Propeller chip is a 32 bit microcontroller with 8 processing cores incorporating 32KB ROM and 32KB RAM shared memory and 2KB RAM local memory per core. It handles communication with the analog to digital converter as well as the USB to serial interface to stream the data packet to the computer.

Isolated power and analog input interfaces were implemented using the TI DCP011515 and TI ISO124P chips respectively. These chips provided the necessary isolated ground to be connected to the test circuit loop and the ability to support the onboard sensors or external shunts

and voltage measurements. The isolated power also provided the power and reference voltage for the MAX1270 A/D chip. The chip was selected because it has 12 bit resolution, 110 KSPS sampling rate, and 8 bipolar input channels. The bipolar inputs are necessary to handle the positive current to charge the cell and negative current when discharging the cell. Without this circuitry built into the A/D chip, external components would be required to scale the voltage as well as provide a voltage bias. Another appealing feature is it operates on a single +5 V supply, which means no additional regulation for a split supply is required for the A/D.

Next, circuitry had to be added to connect the sensors to the A/D converter. A simple divider circuit using high valued resistors (> 1 MOhm) with a ½ gain was used to connect the measurement to the A/D converter. This is done because the source voltage could be a constant current source and the voltage will likely be above rating of the EDLC to maintain the constant current output. By choosing high valued resistors, the effect of the current through the sensing system is minimized. There is also an interface that bypasses the divider to allow other voltage reductions to be used with external components.

Connected to the A/D converter is circuitry to measure a shunt resistor. The measurement is connected as a differential measurement and is passed through an instrumentation amplifier built from generic low noise opamps and tuned to a gain of 10. This sets the range of the shunt to $\pm$ 20 A for the input range of $\pm$ 5 V. The external interface bypasses the instrumentation amplifier to directly feed the differential measurement of the A/D. This was done because an external shunt is going to need to calculate the proper gain to use the full scale range of the A/D input based on the anticipated current. Therefore, if an external current shunt is used it is important to remember that any filtering or voltage scaling also has to happen external to the test system.

A thermocouple circuit was built around the LT1025 cold joint compensator and a low noise opamp. This circuit converts the typical output of a variety of types of thermocouples to a consistent 10 mV/ºC. The chip also acquires the cold joint measurement and applies the correction based on that measurement. This configuration allows the user to select a thermocouple that best suits their needs by simply connecting the leads to the correct pins for the thermocouple type without requiring a change in the rest of the circuitry.

Controlling the test circuit loop is done from two large relays that are driven by transistors. This allows the source and load control to be connected or disconnect independently of each other. This also supports an open circuit configuration to measure the leakage current of the cell.

3.4.2 Software Configuration

The software is implemented in two pieces: firmware and computer code. The firmware code handles the embedded system including: read the sensor information, filling a byte buffer, sending the buffer through the USB port, and reading back any control commands from the computer and converter those commands to output actions. Assembly was used to implement the SPI communication to the A/D converter, and the proprietary SPIN language was used to implement the rest of the logic.

After receiving the measurements, they are encoded in the form "%xxxx*yy". Where the percent sign denotes the start of a data packet, xxxx is the A/D value in decimal form, * denotes the start of the identifier characters, and yy is the two character code to identify which measurement the value is referring to. This form is generated for each measurement and concatenated together to form the transmitted data packet.

The computer side that receives this packet is implemented in LabVIEW CVI. It receives the buffer at 250 KBaud, decodes the packet, and places the information into corresponding variables. Then the variables are displayed on the front panel gauges and charts for a real time view of the data and stored in a file for offline analysis. The software was implemented using three threads. The main thread handled the user interface and configuring and starting the other threads. It also ensured proper cleanup was completed, such as ensuring threads had processed all the data in their buffers, before exiting the application. The primary thread was also responsible for taking user input to control the system, such as to turn on the power source or load, and sending that data to the serial communication thread to send to the embedded system.

The second thread handled the serial communication by inserting the incoming data into a circular FIFO buffer. It then detected when an entire packet had been received, parsed the packet, converted the values from the integer A/D output to the proper units, packed the data into a structure, and finally put the structure into a thread safe queue for the user interface and the file handling thread. The third thread was the file I/O thread. It was responsible for read the data structure from the queue and writing it to the .tdms file. This file can then be converted to an Excel or other format offline. Figure 3.4.2.1 below shows the front panel view.

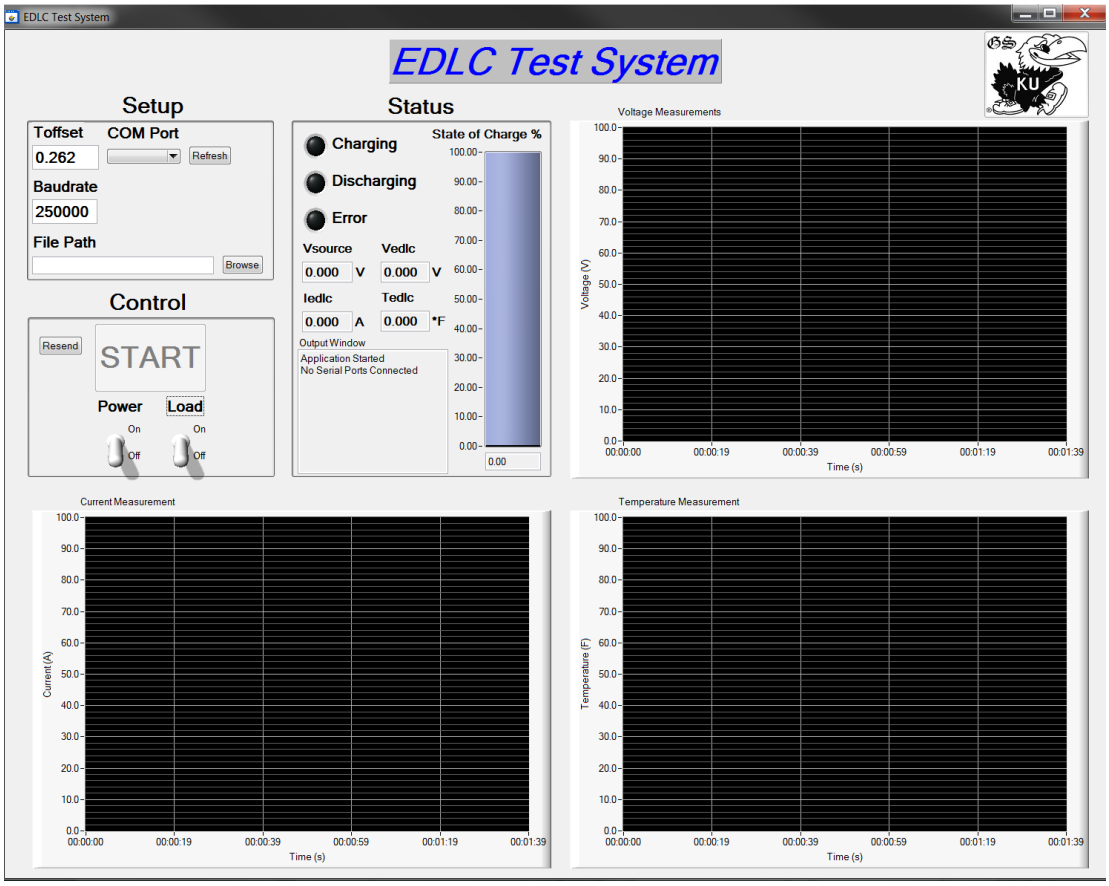

Figure 3.4.2.1: LabVIEW CVI User Interface for EDLC Test System

3.4.3 Measurements

There are four measurements taken by the test system including: source voltage, EDLC voltage, EDLC current, and EDLC temperature. The source voltage measurement allows the system to identify and track the type of charging algorithm being applied. This measurement is also useful in aligning the characteristics of the EDLC with the state of the source and the loads.

The EDLC voltage is taken to prevent an overvoltage condition and provide feedback information so charging algorithms can be tested and closed loop. The EDLC voltage is also used to calculate the State of Charge (SOC) of the cell. This is useful for providing a "fuel" gauge equivalent for the cell. The EDLC current measurement is taken to again ensure that an overcurrent condition does not occur. In addition to protection, it is foreseeable that an

intelligent charge controller would be used and need the current measurement to provide more flexibility than a rigid hardware opamp feedback control loop.

Finally, a type K thermocouple is connected to the EDLC with high temperature glue. This maximizes the contact with the cell to provide a more accurate measurement without the risk of the glue melting and the thermocouple becoming detached. The type K thermocouple provides a sufficient measuring range compared to the operating range of the cell and is relatively inexpensive for its accuracy.

3.4.4 Programmable Load

External to the test system was a programmable load. This was designed to support a large variety of load tests including constant voltage, constant current, constant resistance, and constant power; but the primary purpose was to test the constant current discharge behavior of the EDLC. This will allow the linear discharge behavior to be characterized for the various cells, and fed back to improve the model predictions. This programmable load is shown in Figure 3.4.4.1.

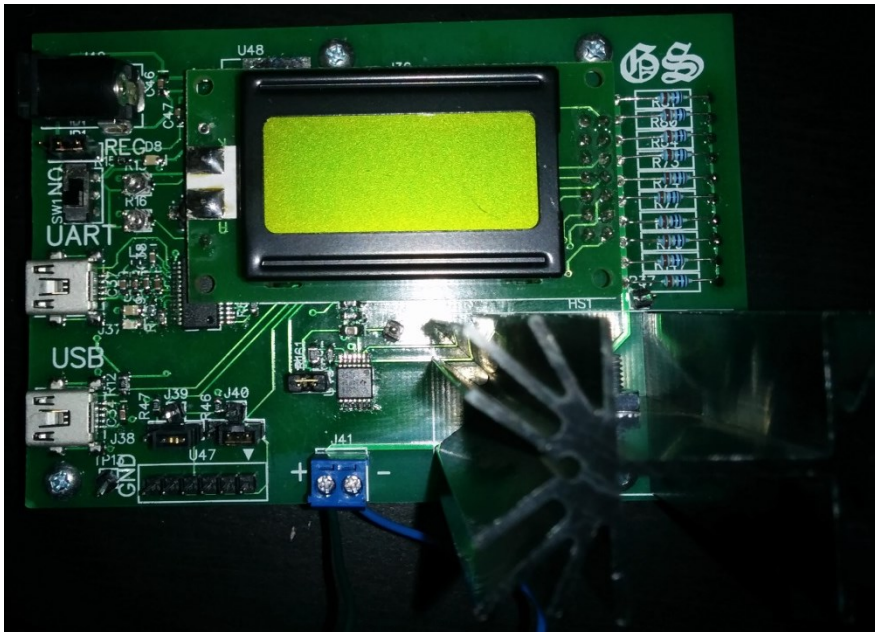

Figure 3.4.4.1: Programmable Load

The programmable load is a custom designed PCB built around the PIC16F1459. The PIC16F1459 is a middle range 8 bit microcontroller with a variety of built in serial protocols including: UART, I^2C, and SPI. It also includes an onboard A/D converter that is a 10 bit, 9 channels, and 125 KSPS module. This provided a sufficient resolution and sampling rate so no external A/D was required. The PIC also includes an onboard PWM module that has a selectable frequency and duty cycle. The carrier frequency was chose to be 19.53 KHz because this was the highest frequency with 10 bits of resolution. This also provided sufficient resolution for controlling the current load.

The current loop of the circuit consists of a N channel MOSFET with a 1 Ohm resistor connecting the source of the MOSET and ground. The drain of the MOSFET is then connected to the positive terminal of the positive side of the source being connected to it. The gate of the MOSFET is then connected to the output of an opamp. This portion of the circuit can be seen in Figure 3.4.4.2.

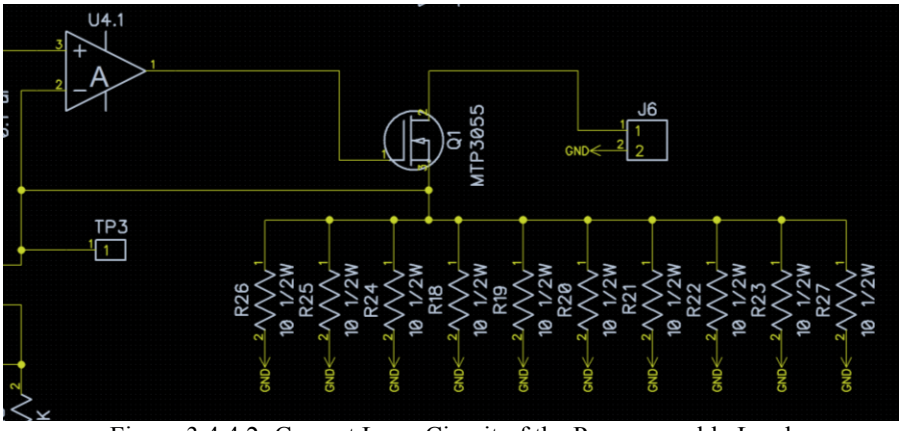

Figure 3.4.4.2: Current Loop Circuit of the Programmable Load

The first observation that can be made is there are 10 resistors rather than 1. This is done because buying a 1 Ohm power resistor is not only relatively expensive, but they are typically not as accurate. To make the current computation simple and fast this value needs to be very close to 1 Ohm, which will be explained in more detail shortly. To solve these shortcomings,

using 10 Ohm resistors not only requires them to handle 1/10 of the power of using a single resistor, but the inaccuracy in resistors tends to cancel each other out in this configuration. This allows cheaper, more accurate, and lower power resistors to be used, while requiring a comparable amount of layout space.

The programmable load fundamentally works off the principle that an opamp will keep the voltage between the inverting and non-inverting inputs equal. As seen in the previous circuit, the inverting input is connected to the resistor bank and the source of the MOSFET. The result of this is the output of the opamp is going to drive the gate voltage to try and match the voltage seen at the source of the MOSFET to the input of the non-inverting input. Because the resistor bank was chosen to be 1 Ohm this means that the current through the loop will be equal to the voltage applied to the non-inverting input of the opamp. To drive the opamp input, an analog voltage needed to be generated. One method of doing this would be a D/A converter. The PIC chosen for the application did not include one onboard, so the PWM module was used. The output of the PWM module was first run through a voltage divider set to ½. This was then put through a low pass filter that smoothed the pulses to create an analog voltage. This configuration sets the operating range of the load to 0 – 2.5 A with a resolution of 2.44 mA based on the 10 bit PWM module. Finally, to provide feedback the MOSFET source voltage was fed to another non-inverting opamp circuit with a gain of 1.636. This sets the voltage to 4.09V, which is very close to the 4.096V reference of the onboard A/D converter. Although this measurement is not required to operate the system as a constant current load, it is required if other modes of operation are desired.

The current settings are displayed by the 2X8 segmented LCD located on the board as well as through the USB connection labeled UART. The UART connected uses a FT232RL

USB to Serial converter to make communication with a computer very simple. This port allows the user to not only get the current settings, but adjust them and turn the load on or off. Finally, to allow the system to reach the maximum current load a relatively large heatsink was connected to the MOSFET. The resistor bank does not need heatsinking as it will only have to withstand 2.5 W maximum and by choosing 1/2W resistors the bank is rated to 5W.

3.5 Results

After constructing the physical system the simulation model was tweaked to more accurately represent the physical components. This consisted of the addition of a resistor of the same value as the current shunt and inputting the measured values of the real resistors composing the voltage divider. The model continued to assume the power supply had an ideal 5 V output. The final configuration used to compare against the physical data is shown in Figure 3.5.1.

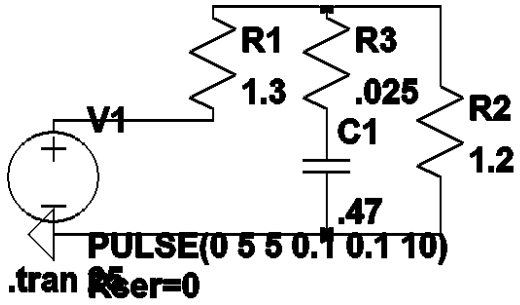

Figure 3.5.1: Test System Constant Voltage Simulation Model

The simulation was run using the 0.47 F EDLC with a 5 second delay, then 10 seconds of power on, and finally off for 10 seconds. The predicted results of the system are shown in Figure 3.5.2.

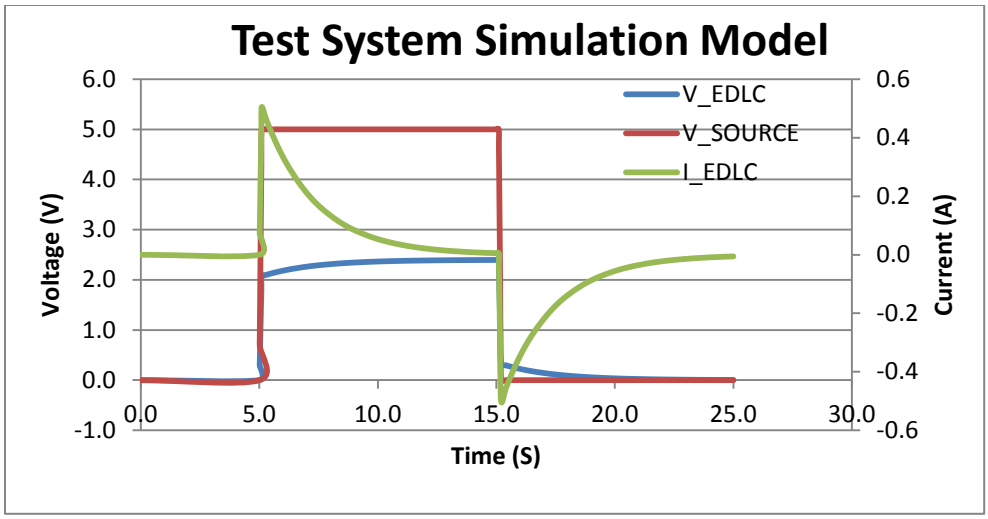

Figure 3.5.2: Simulation Results for Test System Model

The physical system was then configured and run with the same power conditions as the simulation. The measurements were taken at 100 Hz throughput rate. The voltage measurement of the source is shown below in Figure 3.5.3.

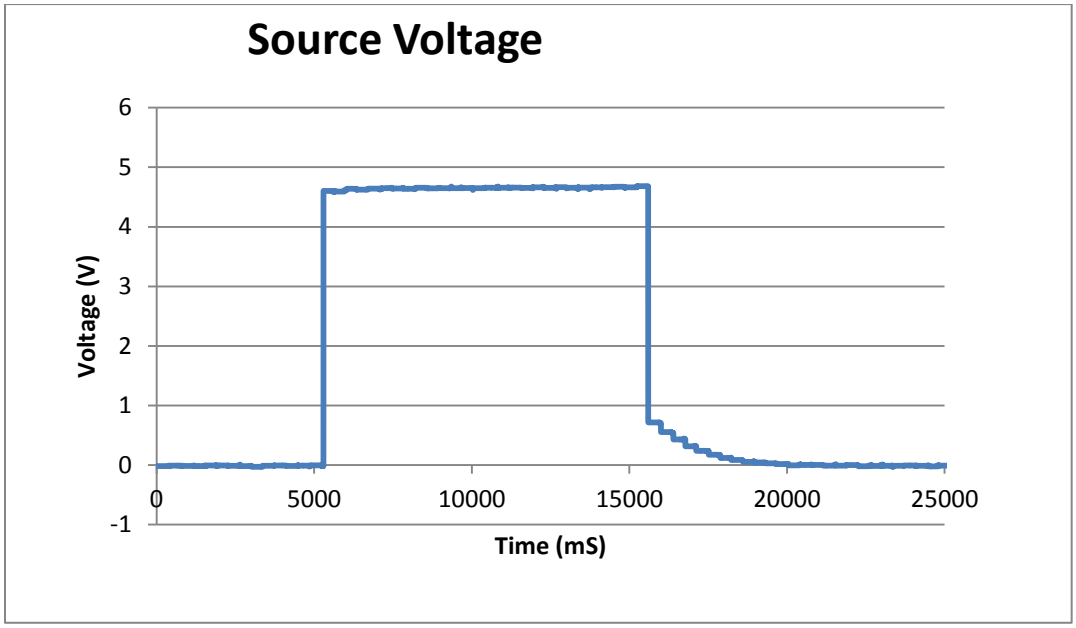

Figure 3.5.3: Source Voltage for Constant Voltage Validation Test

Next, the voltage of the EDLC was graphed and shown in the Figure 3.5.4.

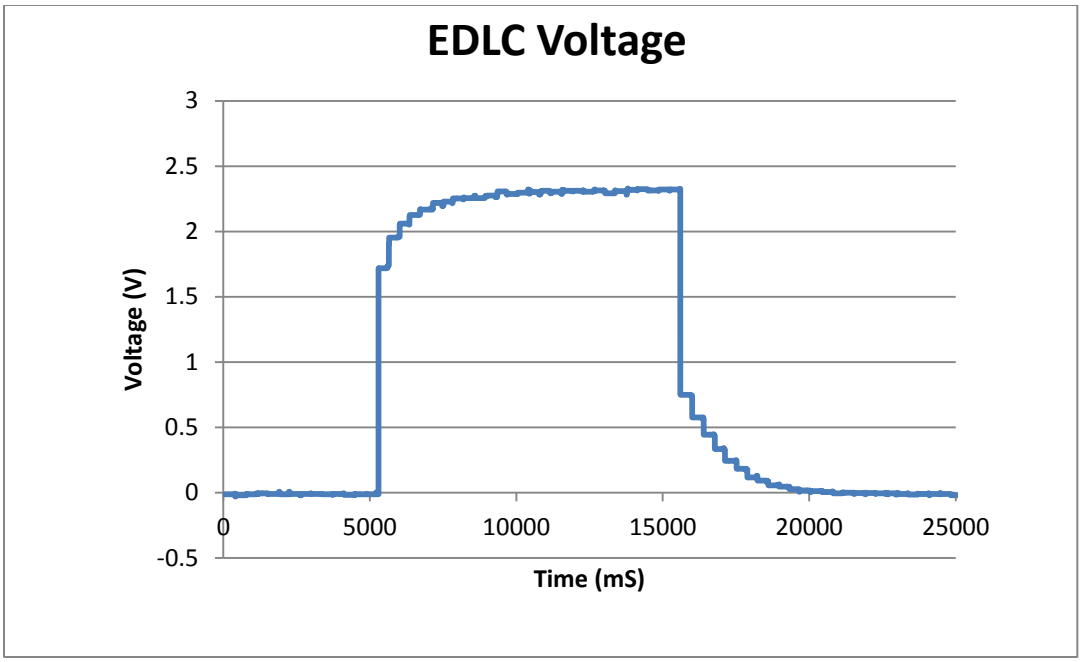

Figure 3.5.4: EDLC Voltage Measurement

Figure 3.5.5 shows the current measurement from the resistor, converted to amperage, providing the current into and out of the EDLC.

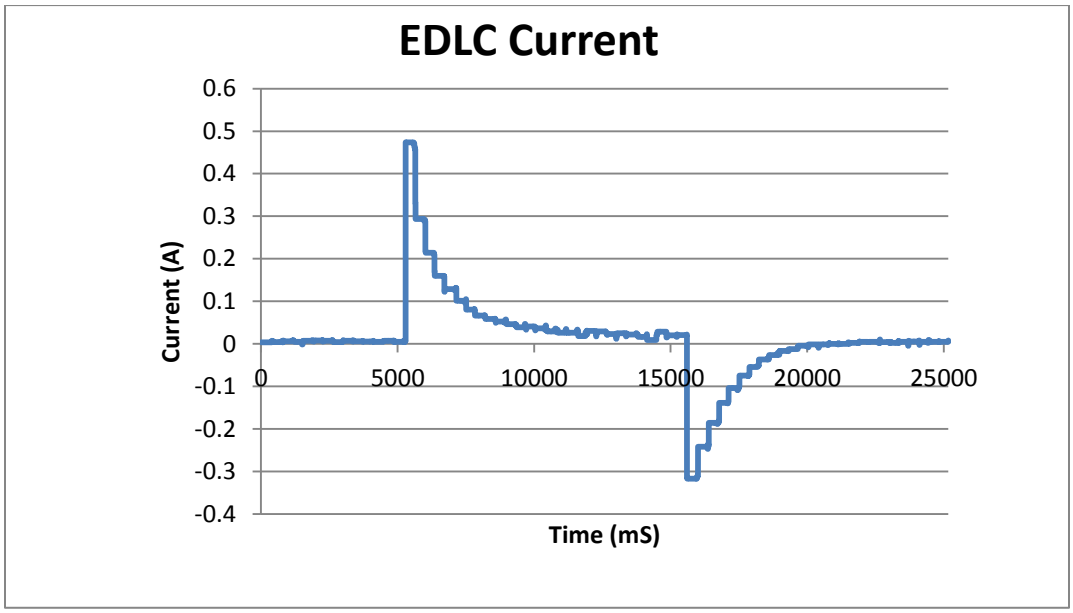

Figure 3.5.5: EDLC Current Measurement

Finally, the EDLC temperature measurement reading using the type K thermocouple is shown in Figure 3.5.6.

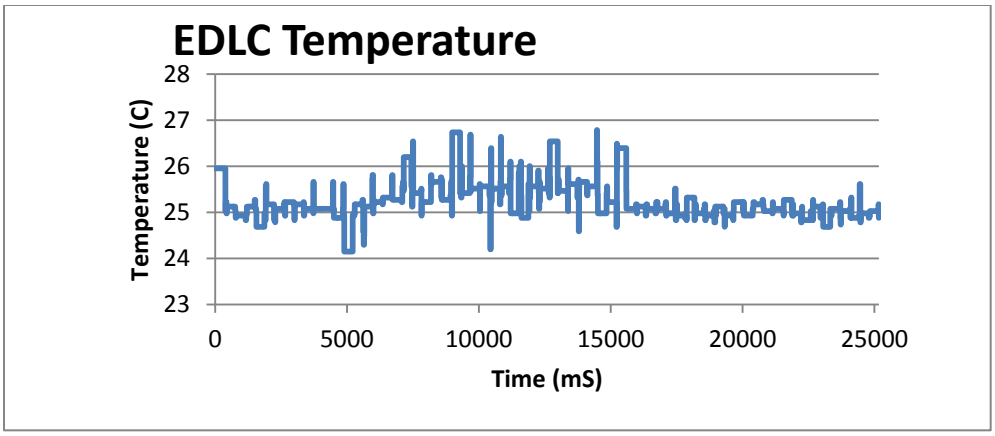

Figure 3.5.6: EDLC Temperature Measurement

Next, tests were run using the constant current programmable load. This was done to test that the linear behavior seen from the constant current charging can be seen when discharging. The test was run by charging the 0.47 F EDLC with a constant voltage source, then given several seconds to come to an equilibrium voltage and then connected to the constant current load set to draw 100 mA. Figure 3.5.7 below shows the voltage measurement of the EDLC.

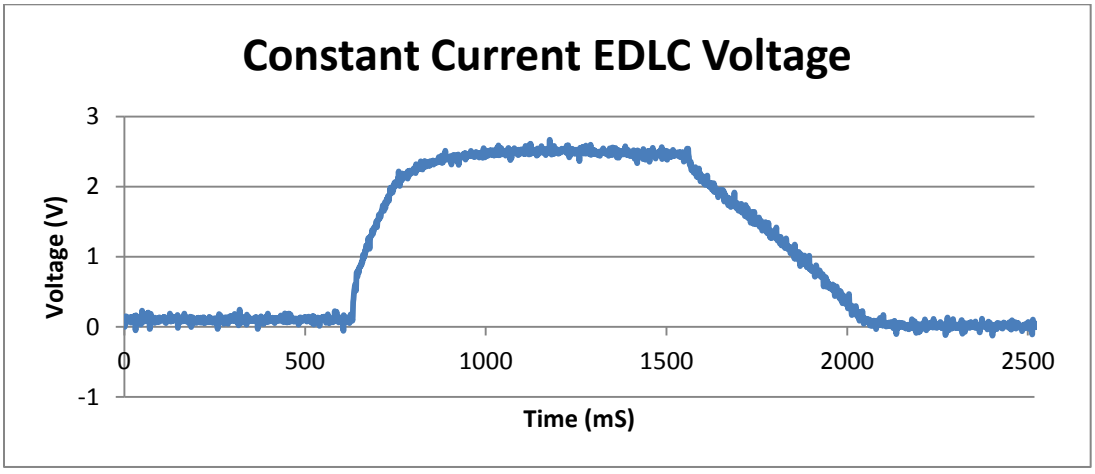

Figure 3.5.7: Constant Current EDLC Voltage Measurement

The current was taken using an external shunt connected to the external shunt circuitry. The results are shown in Figure 3.5.8.

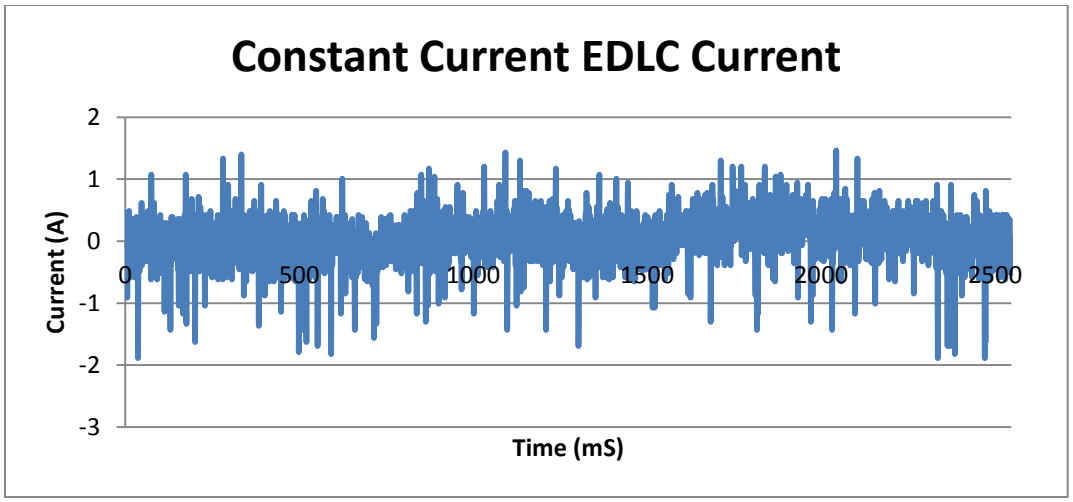

Figure 3.5.8: Constant Current Discharge EDLC Current Measurement

Finally, the temperature of the cell was also measured and is shown in Figure 3.5.9.

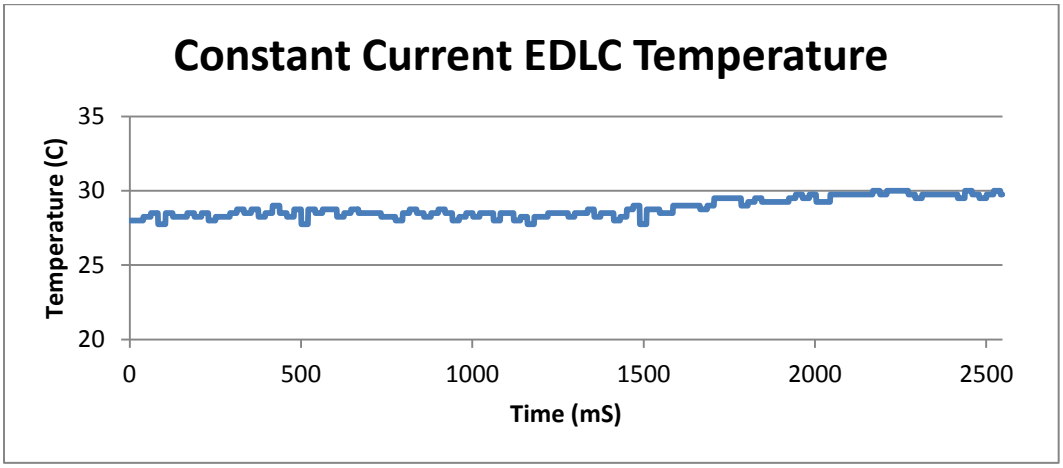

Figure 3.5.9: Constant Current Discharge EDLC Temperature

All of the measurement data in the figures are raw measurements without any software filtering. This was done to identify noise in the measurement system in addition to verifying the operation of the system. This allows adjustments to be made to the sensors or hardware to improve the quality of the measurement before applying any additional real-time or post-processing software filtering.

3.6 Conclusion

It can be seen from the previous figures that the physical system is properly measuring the values of interest and the expected EDLC behavior was observed. It can also be seen that the

constant voltage model closely predicted the true behavior of the system. The EDLC voltage after 10 seconds was measured to be 2.412 V. The model predicted it would be 2.398 V resulting in an error of only 0.58%. Despite the acceptable accuracy, the model assumed a power supply voltage slightly higher than the actual output, but it predicted a lower ending voltage. A future improvement to the power supply model would more realistically predict the power supply behavior.

The EDLC current was measured to peak at 473 mA, while the model predicted a maximum of current of 505 mA. This resulted in 6.3% error between the system and the model. It is again expected that this is a by-product of an idealistic power supply model. The model does; however, provide valuable information as to the approximate maximum current expected as it is better that the model is slightly on the conservative side. This information will help properly size components and measurement ranges to ensure safe operation for larger capacity cells and multiple cell banks.

The final measurement of interest for the constant voltage test was the temperature. It is seen that the average temperature only increased around 1 ºC. The operation of the thermocouple was checked against an infrared temperature sensor and a separate thermocouple measurement system using a hot plate and the system was verified to be reading properly within the accuracy claimed by the manufacture of the IC ($\pm$ 0.5 ºC). This means that the level of current being sourced to the EDLC and the single charge and discharge cycle was not high enough to significantly increase the internal temperature. Future tests could include more aggressive current supplies to find the temperature limits of a particular EDLC.

The constant current load test also performed as expected. The discharge voltage decreased in a linear manner as expected. It can also be seen from the temperature plot that an

increase of 2 ºC was seen. Compared to the single cycle constant voltage test the cell temperature increased around twice as much. This would suggest that the fast high pulse of current that occurs with a constant voltage charging and discharging causes a quick temperature increase that is then quickly dissipated as the current drops exponentially. For the constant current, the large current spikes do not occur but the current stays higher for longer causing a smoother increase in temperature, but it does not have time to dissipate. Therefore, it is important to recognize if the cell is going to experience a large number of successive constant current charging and discharging it will also be important to monitor the rate of the temperature increase.

The current measurement using the external shunt interface had a significant amount of noise. The current load was set to 100 mA, but noise 10 times the signal strength was seen on the measurement. This is likely due to the fact that external shunt interface does not include onboard filtering or signal conditioning, only isolation. This means that not only the external signal conditioning is very important to obtain a quality measurement, but the connection to the board is also susceptible to noise. This could be combated in one of three ways. First, an interface that would interface to a hall effect sensor could be added that would take an isolated measurement and then onboard filtering could be designed based on a 0 – 5 V input, with a 2.5 V bias, could be designed. This is a common output of a hall effect so it would provide ample hall effect sensor choices that could measure up to hundreds of amps. The second option would be to change the connector to a low noise connection and then add some low pass filtering to remove at a minimum the high frequency noise. Finally, the filtering could be done in software in post processing, but with a signal to noise ratio of the signal being so low the performance of this method is difficult to predict.

Overall, the system is operating properly and provides the much needed information to continue EDLC testing and designing larger EDLC banks that operate safely and optimally.

3.7 References

[1] Ashtiani, C., R. Wright, et al. (2006). "Ultracapacitors for automotive applications." Journal of Power Sources 154(2): 561-566.

[2] Burke, A. and M. Miller (2011). "The power capability of ultracapacitors and lithium batteries for electric and hybrid vehicle applications." Journal of Power Sources 196(1): 514-522.

[3] Henson, W. (2008). "Optimal battery/ultracapacitor storage combination." Journal of Power Sources 179(1): 417-423.

[4] Jung, D. Y., Y. H. Kim, et al. (2003). "Development of ultracapacitor modules for 42-V automotive electrical systems." Journal of Power Sources 114(2): 366-373.

[5] Lin, W.-S. and C.-H. Zheng (2011). "Energy management of a fuel cell/ultracapacitor hybrid power system using an adaptive optimal-control method." Journal of Power Sources 196(6): 3280-3289.

[6] Thounthong, P., S. Raël, et al. (2009). "Energy management of fuel cell/battery/supercapacitor hybrid power source for vehicle applications." Journal of Power Sources 193(1): 376-385.

[7] Kuperman, A. and I. Aharon (2011). "Battery-ultracapacitor hybrids for pulsed current loads: A review." Renewable and Sustainable Energy Reviews 15(2): 981-992.

[8] Yan, X. and D. Patterson "Novel power management for high performance and cost reduction in an electric vehicle." Renewable Energy 22(1-3): 177-183.

[9] Lee, D. H., U. S. Kim, et al. (2008). "Modelling of the thermal behaviour of an ultracapacitor for a 42-V automotive electrical system." Journal of Power Sources 175(1): 664-668.

[10] Burke, A. and M. Miller (2010). "Testing Of electrochemical capacitors: Capacitance, resistance, energy density, and power capability." Electrochimica Acta 55(25): 7538-7548.

[11] Gualous, H., R. Gallay, et al. "Supercapacitor ageing at constant temperature and constant voltage and thermal shock." Microelectronics Reliability 50(9-11): 1783-1788.

4. EDLC MANAGEMENT SYSTEM

4.1 Introduction

The advantages of EDLCs in applications that require high power output are becoming very well known. Battery technology is constantly improving in energy density and power density, but they still have not reached the power density of EDLCs [1]. Despite this knowledge, the application of hybrid power systems incorporating batteries and EDLCs has remained mostly limited to research divisions of universities, labs, and companies. To implement such a system, requires a variety of components working well in tandem. First, the battery system requires a management system to control and protect the batteries. Second, the EDLCs require a similar system that controls and protects them. Thirdly, some form of power electronics topology is used to electrically connect the two systems. Finally, an overall control system needs to be implemented to control the power routing between the systems [2,3]. This configuration becomes a complicated implementation that still has not considered other real world variables, such as, cost, size, storage capacity, and desired power output. These criteria also need to be considered and balanced to build an optimal power source [4]. Once that is completed, the designer still needs to select from the huge variety of battery technologies, companies, and specific cell, as well as, select an EDLC cell.

This level of complexity and design effort appear to have been a large inhibiting factor of EDLC/battery hybrid systems. When breaking down the complexity of this system it can be seen that there has been a large amount of research involving the implementation of battery packs and they are found in a commercially available vehicle by almost all major auto makers. There has also been a large amount of research that explores different power topologies and control system strategies to control a hybrid power source at universities around the world [5,6,7]. The missing

piece is a management system for the EDLC bank that is able to scale well and is tunable to support various applications. To accomplish this goal the system also needs to be self-contained and modular. The need to support scalable packs is a necessity for such a system to be successful as each application will potentially implement a different number of cells to meet the requirements of the application [8]. The system should also run solely on the power from the EDLC bank and not require additional external power. This configuration allows the bank to run stand alone or as part of a hybrid system. Support for the hybrid system will be accomplished by providing data about the bank status through a common standard interface. Because this system is targeted for the automotive industry the interface is CAN [9,10]. The system then has to be able to be configured on the fly by the higher level control system that is routing power, as well as, provide imperative feedback information that allows the power routing controller to operate properly [1]. Finally, the system needs to be easy to design and build. Ideally, a few simple circuit parameters are tuned to achieve the desired cell balancing rate and expected current sourcing and sinking. Once those parameters are chosen to meet the application the designer should be able to simply plug the cells into the system, attach the temperature and current sensors, and be ready to operate.

This level of simplicity in implementing a hybrid system involving an EDLC bank would encourage the use of these very beneficial systems. To accomplish all the features described above, the system design needs a balance between a fast acquisition system to ensure safe operation of the cells while minimizing the power consumption of the management system because it does draw directly from the bank. Next, the design and hardware used to realize this system will be discussed.

4.2 Hardware System

The nature of an EDLC management system is performing a relatively small number of operations on multiple cells. To accommodate this naturally parallel operation and the performance required to properly control the cells, an FPGA was chosen as the brain of the system. FPGAs not only have the benefit of being able to perform many parallel operations, but they are capable of having a soft core processor be built in logic that allows the designer to program a portion of the chip as a microcontroller. The Altera Cyclone IV EP4CE22F17C6N was the specific FPGA chosen because it has numerous I/O, high enough logic fabric (22,320 logic elements) to satisfy the application, and it is optimized for low cost and low power applications.

To power the FPGA and various peripherals, several different voltage rails are required. First, a TI LM2576 buck regulator is used to make a constant +5 V output. This chip was chosen because it has a very wide input voltage range of +7 – 40 V. This means with 12 EDLCs connected to the pack the system only requires an average voltage of 0.583 V per cell to operate. This voltage rating for the chosen EDLC cell equates to 4.67% charge required to operate. It is typical to only discharge the cells down to 10% during operation so this selection safely covers that operating condition. For a fully charged 12 cell pack, the voltage could reach 32.4 V, so the system is also capable of handling a fully charged bank with an added factor of safety. The circuit is also capable of supplying up to 3A of current, which is greater than the worst case expected current draw plus a factor of safety of 2. In addition to being used for a variety of peripherals, the +5 V feeds a MAX743 chip that creates $\pm$ 15 V. This voltage is used to supply the multiplexors and opamps used in the data acquisition circuit. Finally, the +5 V rail feeds a

NCP1117 LDO regulator that outputs +3.3 V. This feeds two LP5900SD regulators that create +2.5V and +1.2V used to power the FPGA core.

There are several external memory chips that interface to the FPGA. First, an Altera EPCS64, 64 Mb flash memory, holds the hardware and software images, as well as, the bootloader that configures the FPGA on power up. A 2 Kb I^2C EEPROM serves as an extra non-volatile memory for general purpose use. A 32MB SDRAM provides additional off-chip RAM for the soft processing core. Programming the FPGA and flash memory is done via a USB Blaster circuit. This communicates via JTAG to the various devices in the chain. To communicate to the computer an FTDI FT232RL USB to UART chip is used at 115,200 Baud. The last communication type is the CAN protocol to interface with external hardware. A MCP2515 CAN controller from Microchip converts a SPI protocol to a CAN driver output. That connection then feeds an Analog Devices ADM3053 isolated CAN transceiver. This chip provides power and signal isolation from the bank to support chaining EDLC banks without creating ground loops within the banks. This provides the final CAN compliant output that can be connected to other banks or to the controller in a hybrid system.

The soft core processor implemented in the FPGA fabric was the Altera Nios II fast core. The Nios II is clocked at 50 MHz and handles the higher level functionality of the system, such as communication, data acquisition rate control, and error detection. The core is configured with floating point hardware acceleration, 4 KB instruction and data cache, an internal interrupt handler, and 28 KB of internal RAM. The peripherals in the system have then been connected as memory mapped slaves to the Nios II Avalon memory mapped master port. Address space is then assigned to communicate to each of the peripherals.

The first peripheral is the A/D converter. The A/D chose was the AD7655 from Analog Devices. It incorporates 16 bit resolution, 1 MSPS, and 4 channels. Channels are assigned to the following measurements: EDLC voltage, EDLC temperature, EDLC current, and voltage reference. The voltage reference measurement provides a known voltage that allows the system to validate proper operation when it powers on by checking the known value. The other three channels have passive low pass filters with a cutoff frequency of 10 MHz to reduce unwanted aliasing effects of high frequency noise. Feeding the low pass filters are unity buffers that ensures the A/D input is driven with the proper signal resistance. The first channel buffer is connected to the 12 EDLC voltage measurements. They are first multiplexed with three MAX379 high-voltage analog multiplexors. These not only incorporate fault detection, so they are unharmed if less than 12 EDLCs are connected, but they have 4 differential channels rated to $\pm$ 60 V. The outputs of the three multiplexors are wired in parallel, so only one multiplexor is allowed to be on at a time to provide a valid measurement. These outputs are then connected to an AD629 high common-mode voltage difference amplifier. This chip takes the voltage of each cell and references it to the A/D ground reference so it can be measured.

The second A/D channel buffer is connected to the thermistor interface. Thermistors were chosen as the temperature sensor because they provide an absolute temperature measurement with simpler and cheaper implementation cost than thermocouples. The 12 thermistor's specification includes the following: 10 KOhm, $\pm$ 3% accuracy, and B25/100 value of 3988K. These are connected to two CD74HCT4051 analog multiplexors. Like the EDLC voltage measurement the outputs from the multiplexors are connected in parallel so only one can be enabled at a time for a valid measurement.

The third A/D channel is connected to the current shunt interface to measure the EDLC current. An AD8210 high voltage bidirectional current shunt monitor was chosen to allow a designer to implement the shunt on either the high or low side of the pack without requiring any additional circuitry. The circuit is configured with a reference voltage bias of +2.5 V to support the bidirectional current flow. The chip also contains an internal fixed gain of 20, which allows the shunt to have a voltage of up to $\pm$125 mV at the peak positive and negative current.

The last portion of the circuit is the EDLC cell balancing. A large number of cell balancing circuits have been suggested in a variety of researcher's work [12]. These circuits typically all have pros and cons depending on the specific purpose they were trying to serve. The two basic types are active and passive. Passive cell balancing typically involves putting a load across the cell that gradually discharges the cell at some percentage of its' capacity that over time equalizes the cells. This approach is typically very cost effective to implement, but wastes a significant amount of energy [12]. Active cell balancing typically uses a transistor and a load: resistive, capacitive, or inductive. The load is then switch across the cells transferring current from the higher potential cells to the lower ones. This application required a balance between efficient operation, cost, and pcb footprint to implement. The circuit implementation is shown in Figure 4.2.1.

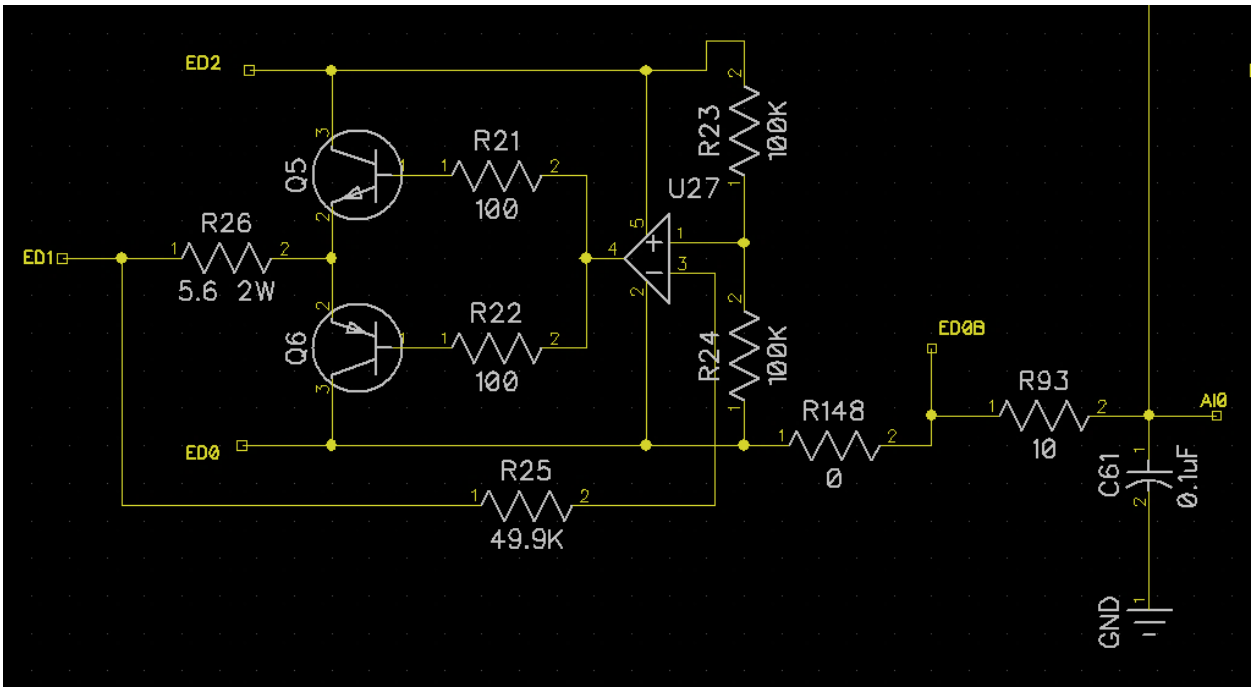

Figure 4.2.1: Cell Balancing Circuit in EDLC Management System

The circuit is based on a resistive load that can be varied by the selection of R26. The circuit uses a comparator circuit to compare the voltage of adjacent cells and switch one of the transistors if there is a voltage difference between neighboring cells. This is accomplished by dividing the combined voltage of the current cell and the next higher potential cell in the series string in half and comparing that to the current cell voltage. If these two voltages are equal then both cells have the same voltage. To balance a bank, the circuit will continue to open and close connections to adjacent cells until all the cells voltages are the same. This circuit is relatively inexpensive to implement in terms of cost and the pcb footprint. The circuit also incorporates the ability to bypass the cell balancing to allow a designer to implement their own scheme. To summarize the hardware configuration a flowchart is shown in Figure 4.2.2.

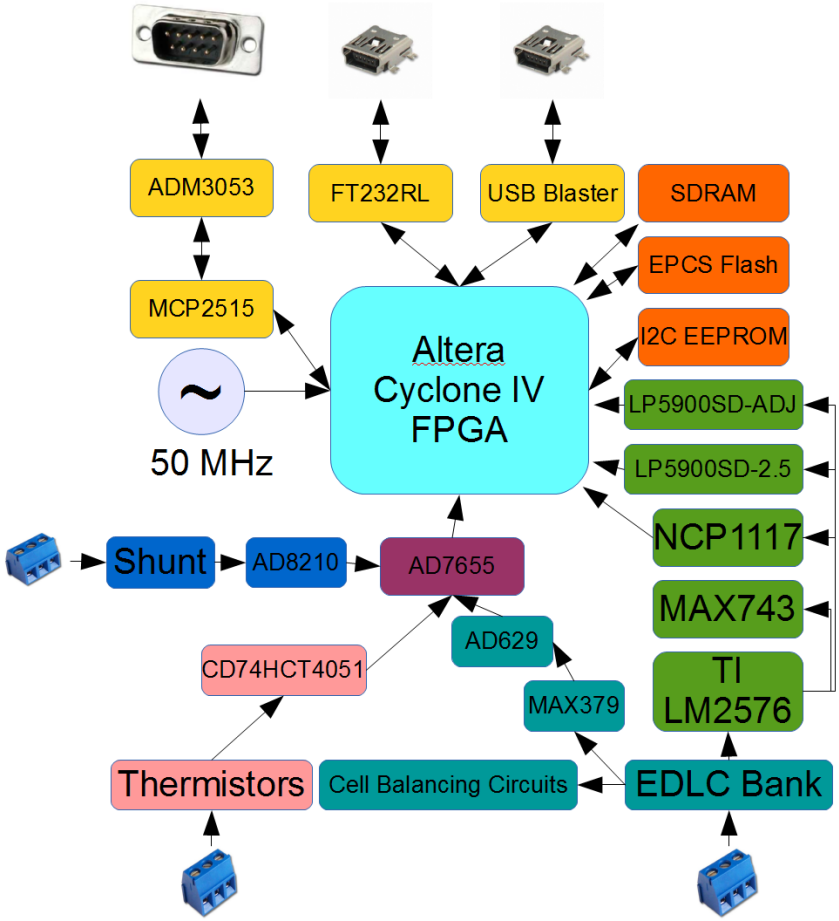

Figure 4.2.2: EDLC Management System Hardware Flowchart

4.3 Software System

The software to operate the EDLC management system is in three pieces: FPGA code, Nios II code, and computer code. The FPGA code is written in VERILOG and responsible for the low level bit manipulation and timing for the various peripherals. These code blocks are also connected to the Avalon memory mapped protocol so the Nios II process can read and write to the associated registers. Next, the Nios II firmware is written in C and controls the overall embedded system. It is responsible for reading in the sensor data, updating statuses, and error checking based on the sensor values. The Nios II then calculates an XOR checksum for the data packet.

Once the data is received, the new status is update, and the checksum is calculated the processor builds the data packet to send to the computer. The data packet is encoded in the form "%SSSSSSSS,XX,XX,XX,…*YY". The percent sign denotes the beginning of a packet. The start of packet is immediately following with a hexadecimal representation of the 32 bit status register. This allows for up to 28 error states to be set by assign one to each bit and the most significant 4 bits report the number of cells attached to the system. This scheme error reporting scheme was chosen over unique codes because multiple errors can be detected in a single packet by statically assigning an error to each bit. Using error codes would require multiple packets or redundant codes to represent multiple simultaneous errors, which would increase the latency between the error and when it is logged by the computer or handled by a hybrid power routing controller. Following the status register is the comma delimited data that reports the temperature, voltages, current, and reference voltage. After the data load, the "*" character denotes the end of the data packet and the start of the 1 byte checksum.

This packet is then sent over the USB connection to the computer application at 115,200 baud. The computer application that receives the packet is implemented in LabVIEW CVI. The application is constructed using three threads. The first handles the user interface and configuring and launching the other two threads. It is also responsible for reading a data queue once the data acquisition begins and displaying the data on the real time display. Finally, the first thread ensures that the application exits safely. This includes shutting down the USB communication thread, then waiting until all buffered data is written to the file, and the file is closed properly before closing the display. This helps ensure that data is not corrupt if the user exits the application without properly stopping the data acquisition thread.

The second thread is responsible for the USB communication. It starts by connecting to the embedded system and reading the data via an interrupt routine into a buffer. As the data is buffered, the thread also constantly checks to see if the buffer contains the start of packet character, the end of data packet character, and two more characters representing the checksum. Once, it detects all of these it proceeds to read that packet out of the buffer. Next, it calculates a XOR checksum on the packet in between and including the start of packet and end of data characters. This value is then compared to the checksum that was received. If the checksum does not match then NULL values are written in to the data buffers to represent a checksum mismatch. If the checksums match then the packet is passed to the parsing algorithm that strips the data, converts then strings to doubles, applies the conversion from the integer A/D value to real world units, and builds a structure that is passed to the display and file queues.

The final thread controls the file operations involved with logging the data. This thread begins by checking if the user defined file exists. If it does then the file name is appended with "_1", so it is not possible to overwrite a data file. This operation is performed recursively until a valid non-existing file name is found. Then the file is opened and the data structure is constantly read from the queue as it becomes available. The values are then written into a text file with space delimiters and a carriage return at the end of the each data structure. This makes the file easy to import into MATLAB, EXCEL, or a variety of other post processing software.

The user interface provides the real time view of the management system and is shown in Figure 4.3.1.

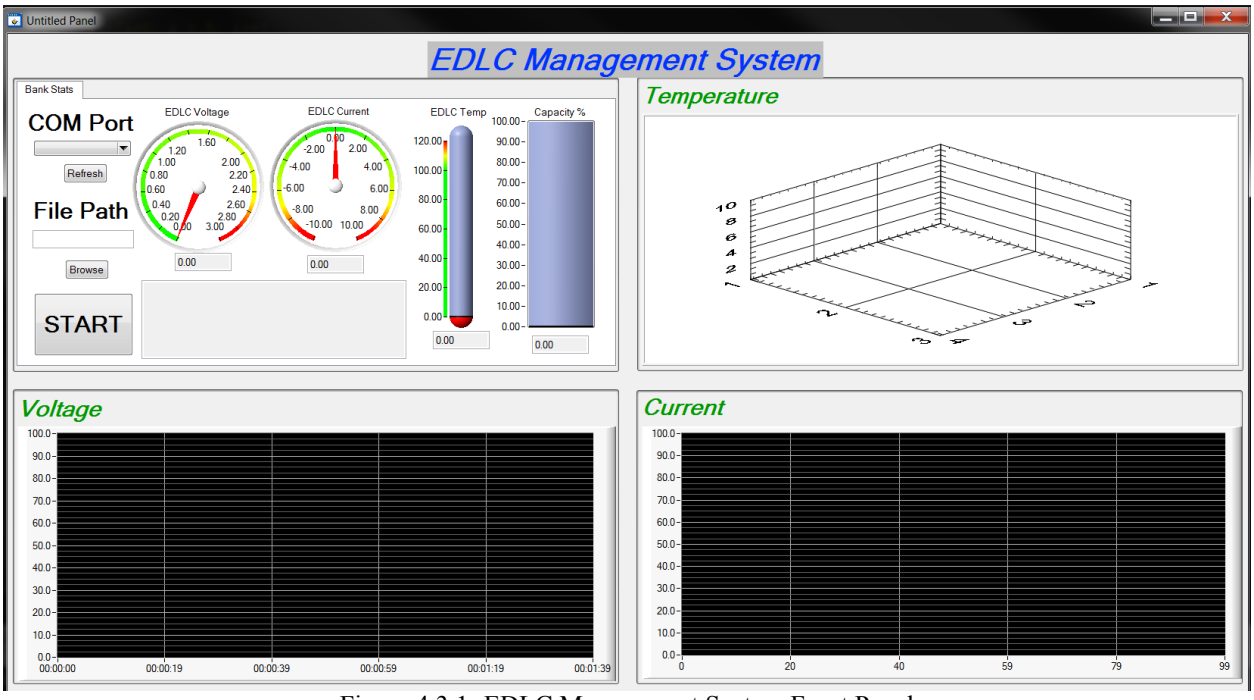

Figure 4.3.1: EDLC Management System Front Panel

The controls for the management system are very simple since it is designed to run autonomously. The computer interface requires the user to select the COM port and a file path to store the data. Then once the start button is clicked the source and average EDLC voltages are plotted in the lower left graph. The current is plotted in the lower right, and the temperature values are plotted on a surface plot in the upper right. The surface plot shows the cells in a 3 X 4 array, but it is important to remember that for a single pack they are connected in series. This was done to shown that this plot could scale to support banks connected in parallel. In the upper left quadrant, the maximum EDLC voltage and current are shown in colored dials to easily identify if hazardous conditions are starting to occur. The thermometer also displays the maximum single cell temperature for the same reason. The capacity of the bank is shown in the fill tank to act as a gas gauge equivalent. Finally, a display box is used to display the translation of status register to inform the user of any errors reported by the management system.

4.4 Results

The first test ran was to validate the setup. During this test, it was discovered that the fastest acquisition rate that could be attained without data loss through the serial port due to overrunning the computer buffer was 10 Hz. Therefore, the controller was reduced to this rate for the tests discussed here. In practice, instead of sending every piece of data the serial port was meant to send a subset of information that represents the overall state of the bank would be sent so it does not reduce the performance of the controller loop rate. A constant voltage source was set to +30 V and connected to the bank with 20 Ohms of current limiting resistors also connected in series. Once the voltage stabilized for a few seconds the power supply was shutoff and the bank was allowed to discharge until the bank voltage dropped below the drop out voltage and the controller shut down. The bank was also configured with the cell balancing in bypass mode, so no active or passive balancing was performed. This was done to provide a baseline to compare the results with the cell balancing circuit active. Finally, the results are present as the raw data and no software post processing filtering has been applied. This was done to evaluate the noise of the acquisition system and the quality of the hardware filtering.

To report consistent results between tests the following quantities have been plotted: EDLC bank voltage, EDLC bank current, temperature (average, max, and min), and voltage difference between the highest and lowest potential cells. The first plot, Figure 4.4.1, shows the bank voltage.

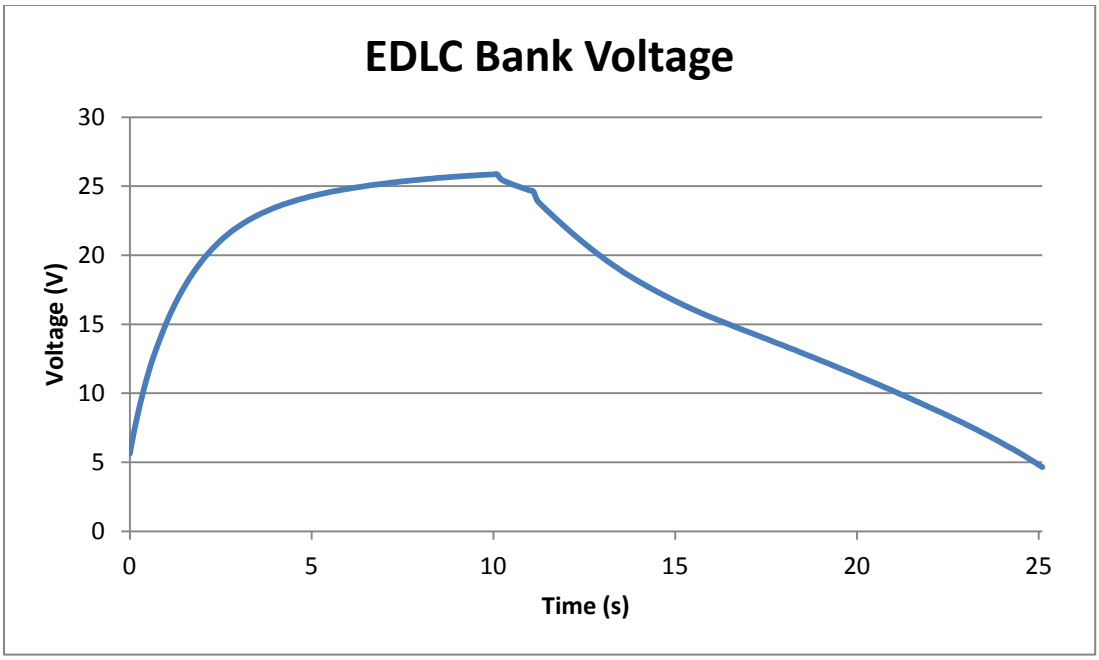

Figure 4.4.1: EDLC Bank Voltage

As expected with a constant voltage source, the exponential charge curve was observed. The first observation that was made was the first stage buck converter operated below the rated 7V minimum. The converter acted like a pass through between around 5V and 6V, and then began regulating the voltage at around 6V. The rest of the circuitry was able to handle this temporary overvoltage condition and operate normally. Another apparent behavioral outlier is the voltage dip that can be seen when the charging is complete. It turns out this is the start of the voltage settling that occurs as the circuit reaches a steady state. This test would suggest that the power supply was turned off before the circuit was completely allowed to reach this steady state behavior. This can also be seen in the EDLC current shown in Figure 4.4.2.

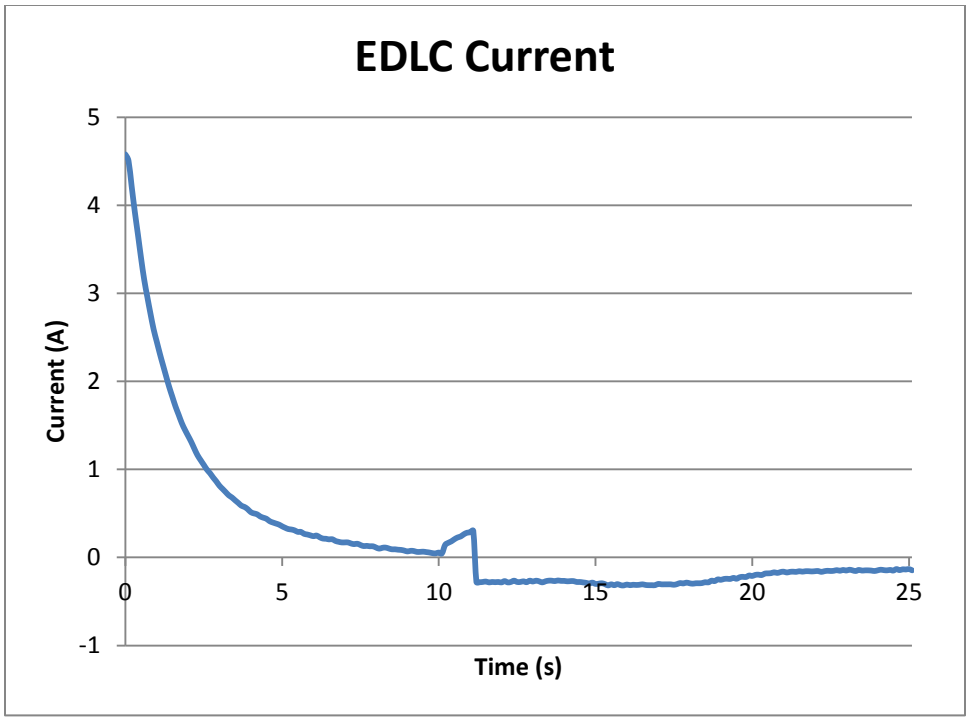

Figure 4.4.2: EDLC Bank Current

The cause of the voltage dip is very apparent from the current behavior as it is time aligned with the spike seen at the bottom of the charge curve. Seemingly, an outlier in the current behavior is the lack of a clean exponential discharge. This was a result of the system setup. Once the power supply was turned off, the circuit did not have a significant load to discharge the bank, so it discharged through the power supply and the power drawn by the management system. The combination of these two loads is not purely resistive so the pure exponential discharge is not seen. It is expected if a lower resistance load was connected in the circuit to discharge more current the exponential behavior would be seen as this would over-power the load of the management system.

A very important quantity is the difference between the maximum potential cell and the minimum. This represents the cell imbalance in the bank. This is plotted in Figure 4.4.3 below.

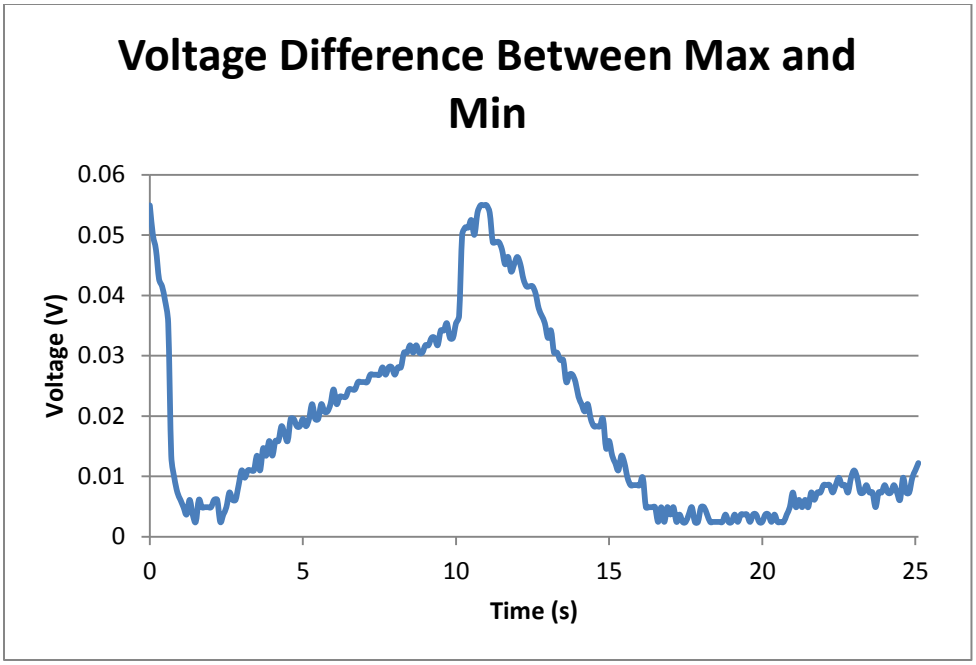

Figure 4.4.3: Voltage Difference Between Max and Min EDLCs

The cell imbalance after a single cycle is not expected to be significant as this is a condition that tends to degrade overtime. The results of the test were surprising showing that after a single charge cycle without cell balancing there was already a 2.2% difference between the highest and lowest potential cells. The discharge cycle brought this difference down significantly, but there still remained an imbalance of around 0.4%. This does not seem significant, but for a behavior that is expected to degrade over time this would be a cause for concern and a justification that cell balancing is required.

Finally, the temperature of the bank was monitored. Figure 4.4.4 shows the average temperature of the 12 EDLCs, the highest temperature EDLC, and lowest temperature EDLC.

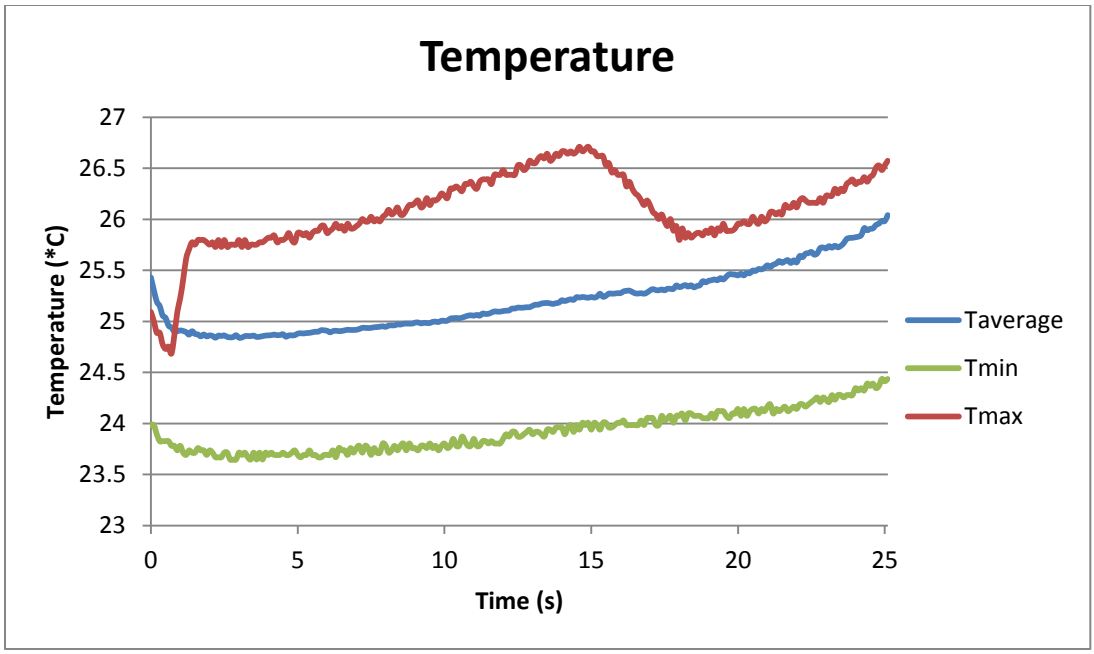

Figure 4.4.4: Temperature of the EDLC Bank

Almost humorous behavior is the maximum cell temperature is initially below the average. This occurs because this value represents the EDLC that has the highest temperature sum over the entire acquisition period. This was done because the EDLC with the highest temperature at a single point in time varied, and this value is meant to represent the "hottest" overall cell in the bank. Starting with the expected behavior, the average temperature starts by spiking when the current is very high and then dissipates as the current falls rapidly. Then, the bank tends to heat up gradually as it is discharged. The unexpected behavior was seen in the EDLC that had the highest temperature. It behaves like the average experiencing an initial spike and then dissipates, but it then quickly experiences another step change in temperature. After the step change, it continues to gradually heat in a similar fashion to the bank average, but once the discharge begins it experience another step change. This behavior was observed to varying degrees of severity starting around the middle cell up to the highest potential cell. This would suggest that this is a true system behavior rather than faulty measurement. The explanation for this behavior is the data is showing the filling behavior of the bank. The initial temperature is below the

average because the lower potential cells are filling first, and then rather quickly the current begins filling the entire pack which causes the first step change in temperature. The bank then continues to fill relatively evenly and the gradual heating behavior is seen. Then once the bank has filled the higher potential cells are able to dissipate as the current into these cells is reduced and the circuit starts to reach a steady state. This causes the second step change as the cell cools closer to the same temperature of the rest of the bank.

The bank was located in open air environment with no additional cooling, so it would be incorrect to assume this same behavior would necessarily be seen when the cells are located an enclosure or have additional cooling. A main goal of measuring each cell is to not only attain a more reliable average of the bank, but to identify hot spots when the system is implemented in an application to assist the designer in designing a cooling system that focuses on the hotter portions of the bank.

The second test that was run followed the same procedure as the previous test, but allowed the bank to rest longer after charging to provide more insight into the steady state behavior. The associated figures, Figures 4.4.5, 4.4.6, 4.4.7, 4.4.8, are shown below.

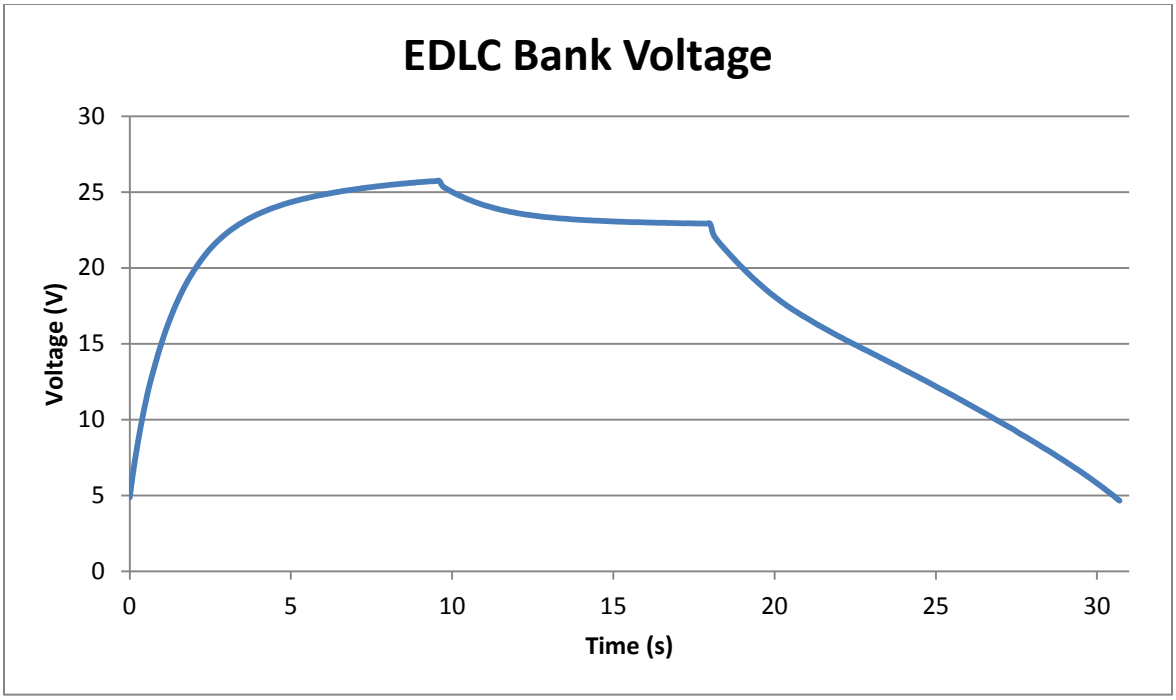

Figure 4.4.5: EDLC Bank Voltage Test 2

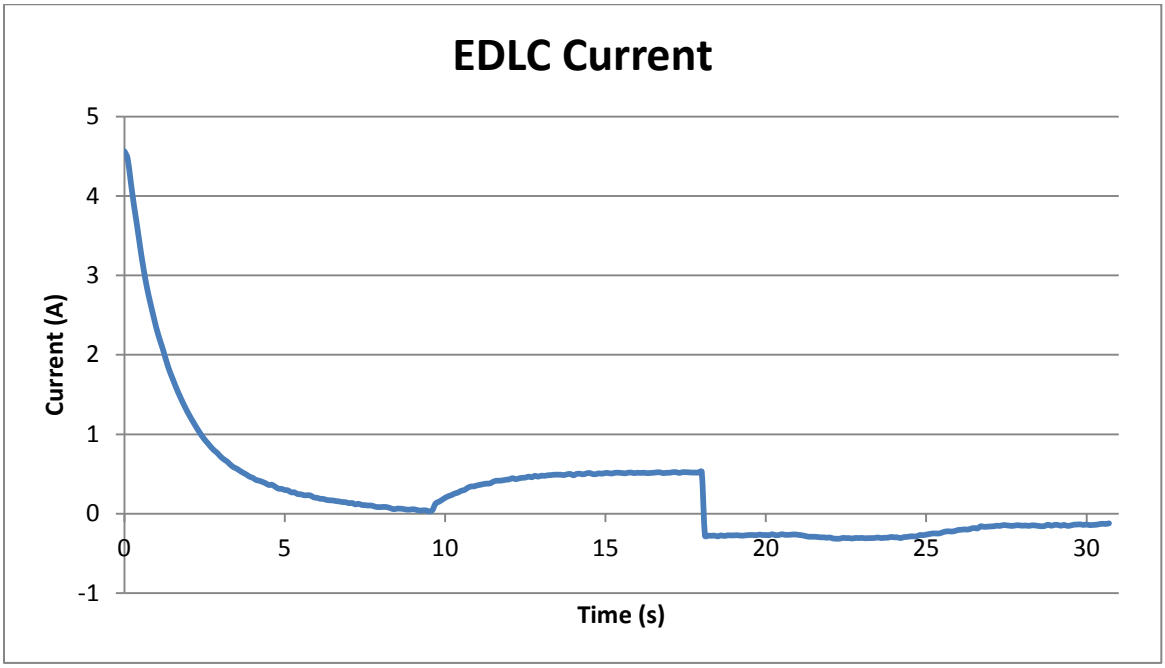

Figure 4.4.6: EDLC Current Test 2

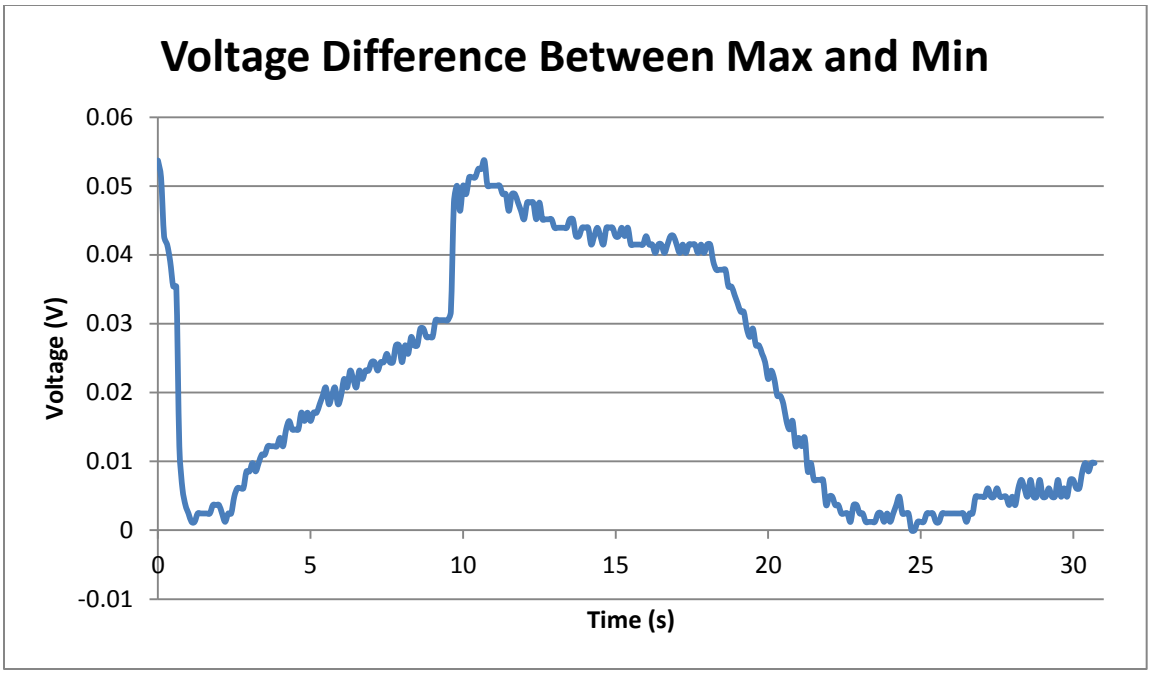

Figure 4.4.7: Voltage Difference Between Max and Min EDLC Test 2

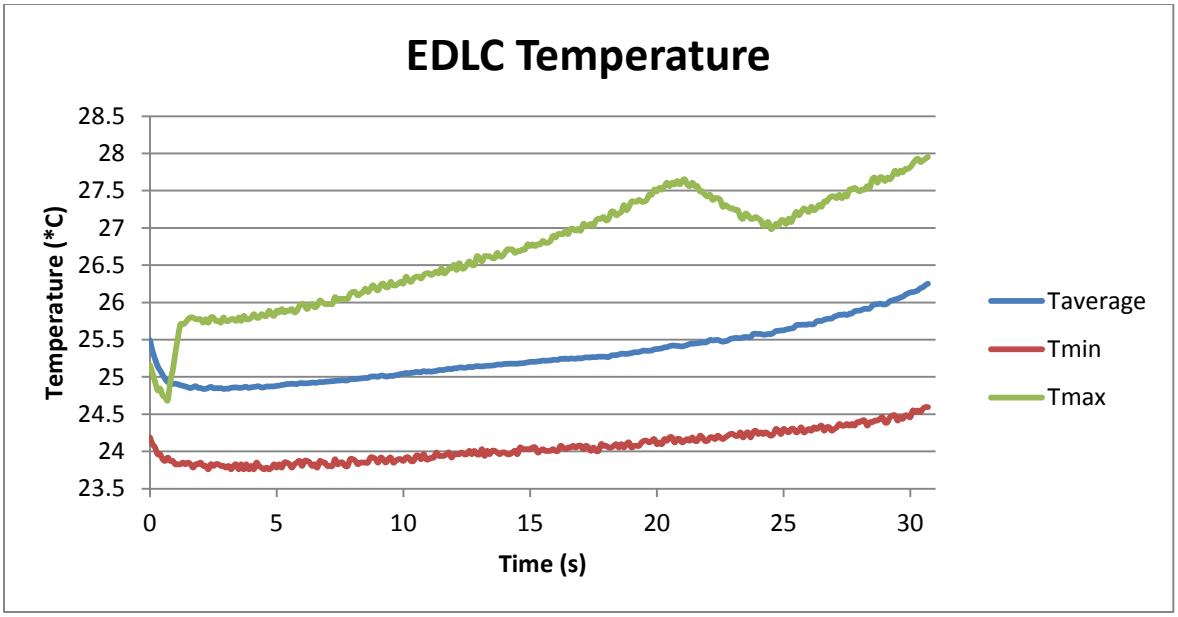

Figure 4.4.8: EDLC Temperature Test 2

The results of the second test were very similar to the first test as expected. The main point of emphasis is the behavior that occurs after the bank reaches the maximum voltage. The assumption was correct that circuit was settling to a steady state response. It can be seen in the voltage and current plots that once the bank is charged it comes to an impedance match with the

power supply and begins to share the load of powering the management system. This load sharing flattens out until the power supply is turned off and the bank begins to solely power the management circuit and the power supply acts like an additional load on the system.

The next test that was performed followed the same steps as the previous test, but the cell balancing circuits where switched from bypass to on. The EDLC bank voltage is shown in Figure 4.4.9.

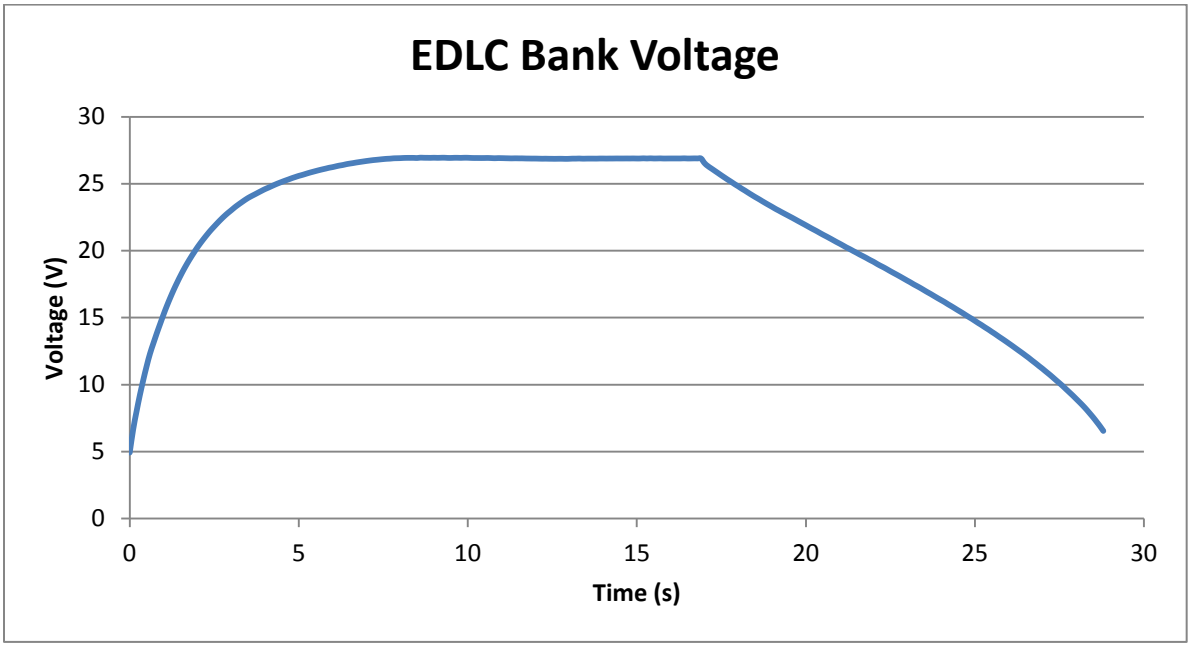

Figure 4.4.9: EDLC Bank Voltage Basic Test with Cell Balancing

Several differences were observed compared to the test without cell balancing. First, the voltage does not experience the settling effect that was previously present. Secondly, the maximum voltage reached increased from around +25.7V to +26.89V. Finally, it was observed that the buck regulator dropped out at +7V instead of working down to +5V like the previous test. The higher ending charge voltage can be explained by the addition of the resistors used in the active balancing. When the switches are on, the resistance of the bank increases. This causes the voltage to increase relative to the current limiting resistors because they are not changing in value. The flattened voltage after the charging is also explained by the balancing resistors. The

power supply continues to be applied during this period and the current that would normally be redirect to powering the controller is now dissipated across the balancing resistors. The flat behavior is due to the fact that the resistors will be switched as cells have a potential difference, so the current will be dissipated across the resistance of the cells with the highest potential. The buck converter having an increased drop out voltage is likely due to the resistors acting as a load on the input side of the regulator. Previously only the EDLCs were connected to the regulator so they were acting like a power buffer. The addition of the cell balancing introduces a resistive load with lower resistance than the input of the buck regulator, so the current is sunk through the resistors rather than continuing to power the buck regulator. These results are supported by the data seen in the current measurement shown in Figure 4.4.10.

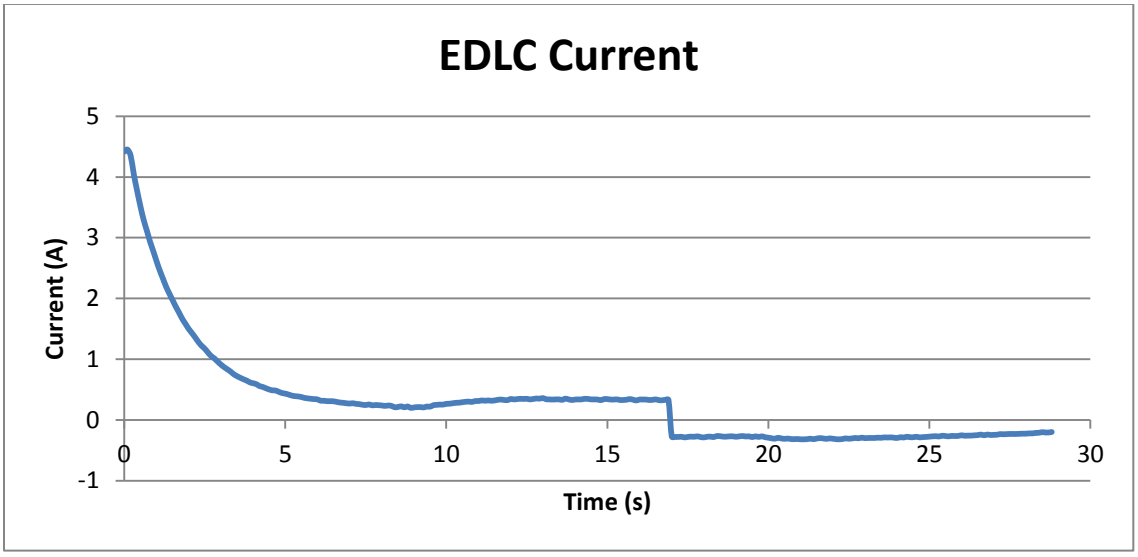

Figure 4.4.10: EDLC Current Basic Test with Cell Balancing

There are only a few minor differences in the current behavior. The major differences are the current does not drop to 0 A during the charging cycle and the steady state current is reduced. These are both explained by the same rational behind the voltage behavioral differences. Figure 4.4.11 shows the plot of most interest for this test displaying the voltage differential between the highest and lowest EDLCs.

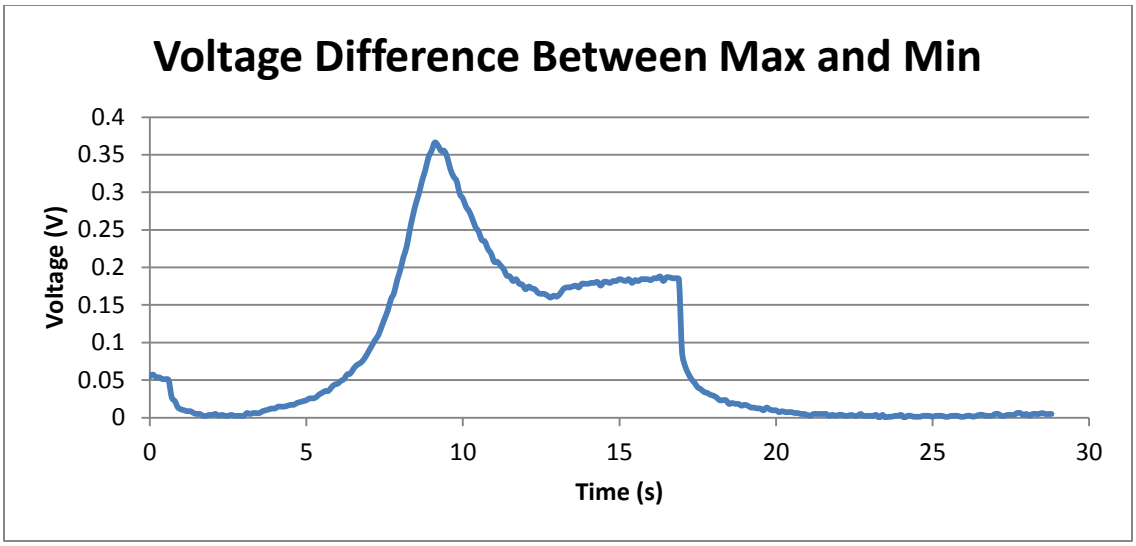

Figure 4.4.11: Voltage Difference Between Max and Min Basic Test with Cell Balancing

A major difference was seen in the maximum differential that was seen during the test compared to the previous test. A peak difference was 0.366V compared to 0.053V without balancing. This is due to the resistors slowing down the natural response of the EDLCs to balance among themselves. This behavior was expected, but the amount that is occurred was surprising. The other difference was seen in the ending difference. Previously the voltage difference was 0.01V, but with the balancing it was reduced to 0.003V. This shows that the cell balancing circuit is working properly.

The temperature data is show in Figure 4.4.12. There was not a significant change in behavior between the two tests, so the addition of the cell balancing does not have an effect on the heating of the cells.

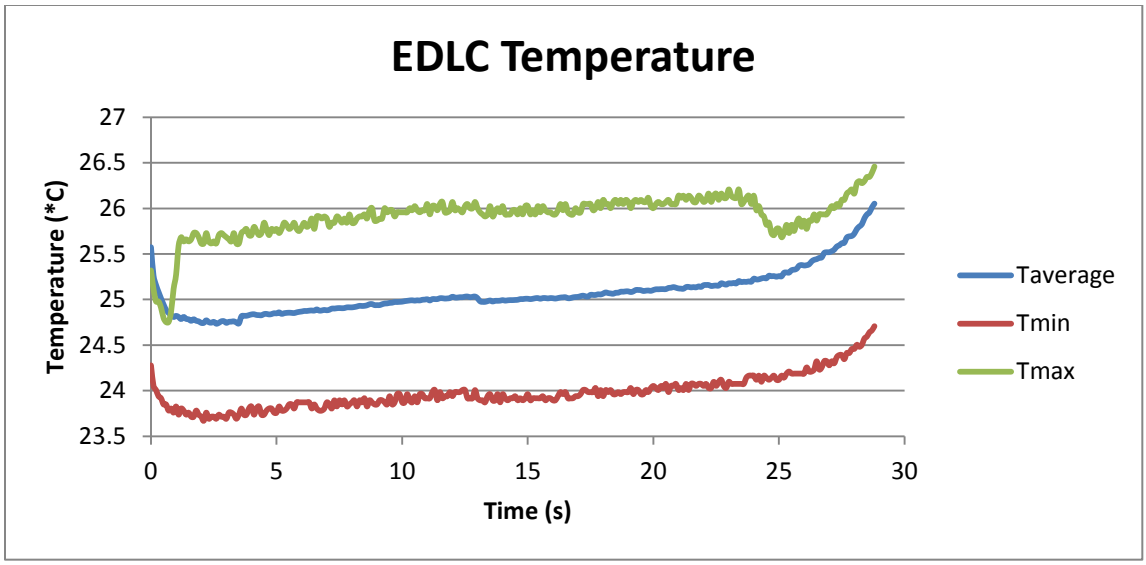

Figure 4.4.12: EDLC Temperature Basic Test with Cell Balancing

The final test presented here was a cyclic test. The bank was charged with a constant voltage source of +30V. Once it reached a stable voltage, the supply was turned off and the cells were allowed to discharge. When they reached around +12V power was reapplied until they reached a stable voltage again. This was repeated for five cycles with the cell balancing circuits active. This test was to further test the ability of the cell balancing circuit to maintain the voltage balancing across several cycles, as well as, to observe the thermal behavior under cyclic loading. The bank voltage is seen in Figure 4.4.13.

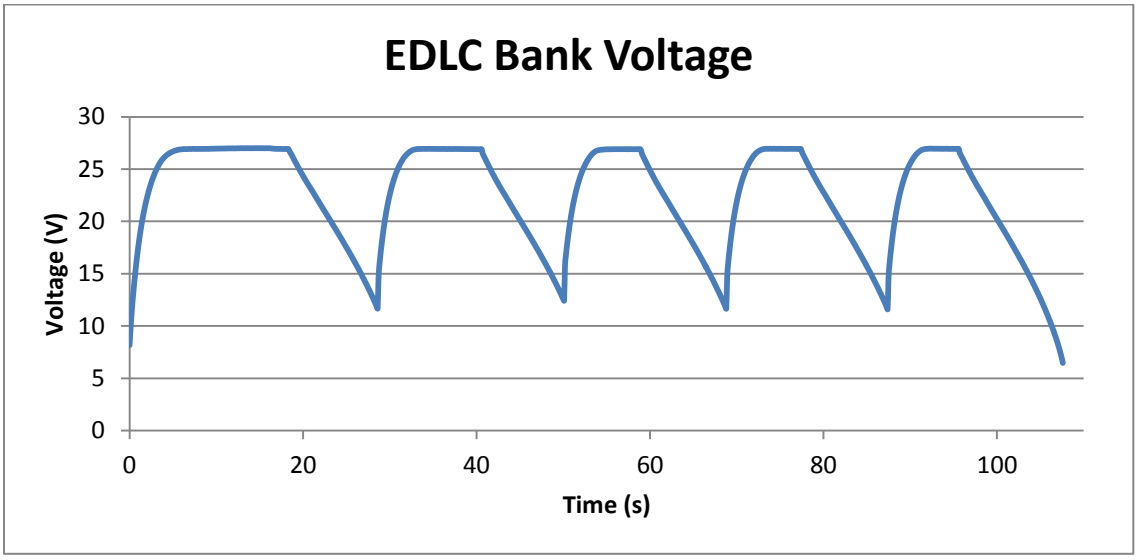

Figure 4.4.13: EDLC Bank Voltage Cyclic Test with Cell Balancing

There were not any surprises in the voltage behavior. The bank consistently returned to the same maximum voltage with a repeatable behavior seen in a single cycle. The lack of consistency in the pulse length was due to the manual cycling of the power rather than a controlled source. The current data shown in Figure 4.4.14 was also as expected.

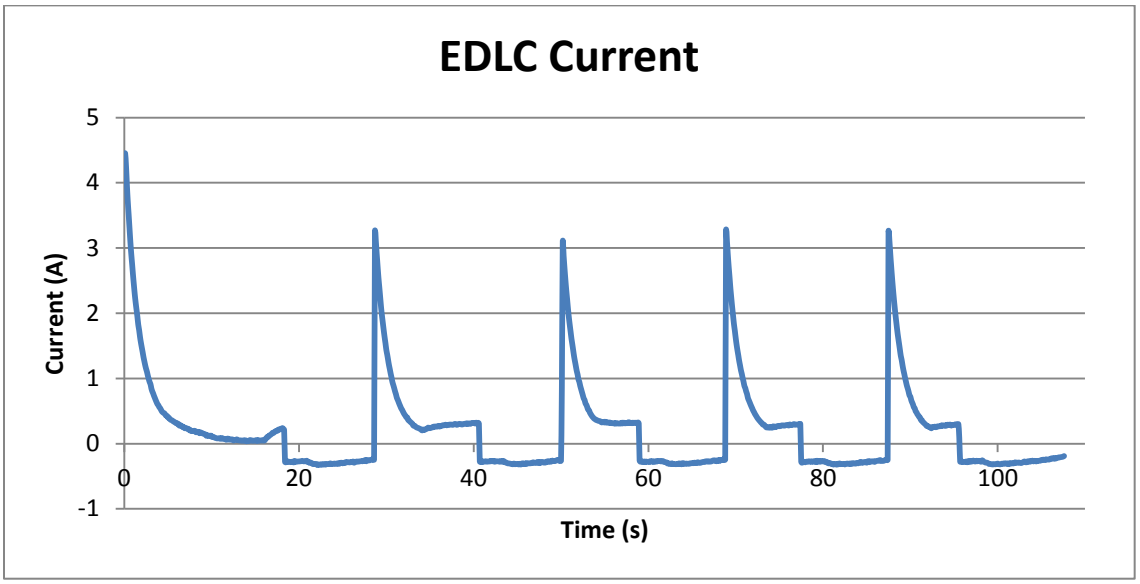

Figure 4.4.14: EDLC Current Cyclic Test with Cell Balancing

The slight variation in current peaks is also due to manual power cycling as the bank voltage was not exactly +12V each time when the supply was powered on. Other than that observation, the behavior was also very consistent with the single cycle test.

Figure 4.4.15 shows the voltage difference, and this has several interesting characteristics. It is seen that the voltage difference varies between cycles. The behavior tends to follow a similar trend, but the third cycle behavior more closely resembled that of the first cycle rather than the subsequent cycles. A likely explanation for this behavior is when the power was applied for the first and third cycles more of the transistors were switched open which resulted in a higher resistance of the bank. This in turn caused the smoother behavior that is seen compared to the other cycles. The more important piece of the information is the ending difference between the cycles. The voltage difference very consistently returns to the 0.003V

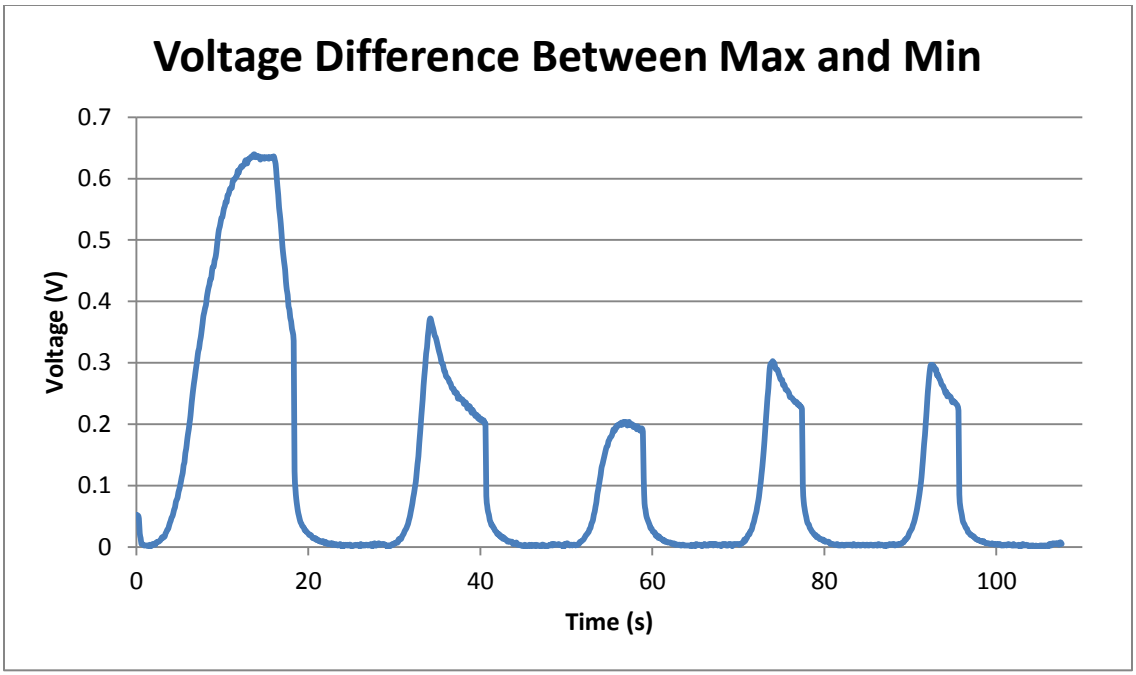

Figure 4.4.15: Voltage Difference Between Max and Min Cyclic Test with Cell Balancing

that was observed in the single cycle test. The final ending voltage was 0.0024V, which was even a slight improvement over the single cycle. This suggests that although the behavior during initial charging may exhibit some variation, overall the cell balancing circuit is performance very well. Lastly, the temperature data is displayed in Figure 4.4.16.

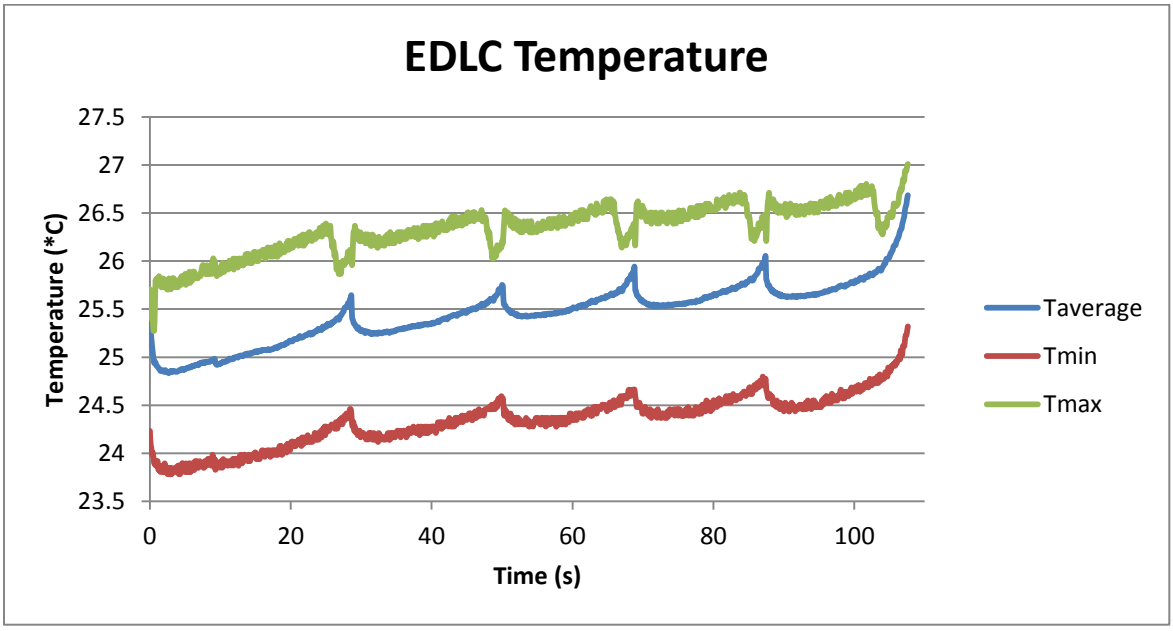

Figure 4.4.16: EDLC Temperature Cyclic Test with Cell Balancing

Again a very important behavior was observed during this test. The temperature behavior shows the same pulse behavior that was demonstrated in the single cycle test, but the overall trend is gradually up. This was the anticipated behavior of a cyclic charge and discharge sequence. This data supports the idea that temperature rate measurement could be attained by averaging the data over the cycle.

4.5 Conclusion

The primary goal of this research was to design, construct and validation an EDLC management system that would be capable of operating as a standalone controller or as part of a hybrid power system. The management system also needed to be relatively simple for a designer to implement and scalable. Ideally, the designer simply needs to select a fixed number of tunable parameters regarding the specifics of the application and the cells and thermistors directly plug into the system and are ready to operate. The system need to support a series string of 12 EDLCs per control board implemented. Finally, the system needed to be self-contained in terms of power usage to satisfy the above requirements. The proposed design was built around an Altera Cyclone IV FPGA, which was chosen for its superior parallel processing capabilities compared to a microcontroller. The high level functionality of the system was implemented in a soft core Nios 2 fast core processor built in the FPGA. This allowed advanced functionality to be programmed user higher level languages.

Several tests were run that validated the proper operation of the system and even exposed some previously unknown behavior of an EDLC bank. These tests validated that all of the design criteria were met with this design. The tests also showed the necessity of a cell balancing circuit, as well as, proving the active resistance scheme chosen performed quite well. Some

further improvements in performance and implementation were identified after further analysis of the data.

First, the cell balancing circuit drew a significant amount of energy, which is likely a direct cause for the impressive performance. To reduce the amount of wasted energy, the comparator could be removed and the bases of the transistors could be connected to the FPGA. This could allow the balancing scheme to be programmable. An immediate improvement that could be seen from this configuration is to disable the balancing while the EDLCs are being charged as it was observed they naturally balanced themselves better during charging without the added resistance. After the bank reaches a defined percentage of full, the balancing scheme could be switched on to maintain the flatten behavior that was seen in the tests with the cell balancing active. Finally, the balancing scheme could be modulated once the cells are balanced within some threshold to reduce the energy loss. It is also conceivable that further improvements could be seen with more robust balancing algorithms than the simple solution suggested here. Another way to reduce the power draw from the management system would be to shutdown the processor when the bank is in a steady state. This would then require some wakeup criteria that could be threshold based or time based, but this would reduce the power consumption of the system regardless.

An improvement that can be made to the current limiting structure is the implementation of a thermal rate controller. It was seen that on average during the cyclic test the temperature demonstrated a relatively predictable temperature increase. It would be beneficial to monitor this rate because it would provide a prediction as to when a hazardous condition will be reached. Based on this, the control strategy can be adjusted to keep the bank from reaching the condition

in a proactive scheme rather than turning off the pack in a reactive scheme. This would improve the robustness of the system during operation.

To improve the test acquisition, the amount of data could be reduced by only reporting the values that were plotted in the results section rather than every piece of data. This would allow the control to increase its speed without exceeding the bottleneck of the communication bandwidth. To further improve the communication, a higher baud rate for the serial communication could be implemented provided this does not result in data corruption. The XOR checksum is not a robust enough check if there begins to be a significant amount of corrupt data packets. This could be combatted by a more robust CRC calculation being performed and resending packets, but it would be best if this situation is avoided. Another means to speed up the acquisition is to convert the acquisition to be based on the CAN communication. The CAN communication has higher data rates and better noise reduction compared to implemented serial communication. This is not an ideal situation unless there is a configurable setting to turn off the CAN messages when using serial because it was found that the CAN bus had to be disabled during acquisition because there was not a system to remove messages from the bus so it crashed. It is also not ideal because there becomes a coupling between acquisition or debugging and an application and this should also be avoided, so the bank can be observed via the serial port while the CAN communicates with a hybrid controller. Overall, the system performed very well and provides a much easier means to implement an EDLC bank than currently available systems.

4.6 References

[1] Auer, J. M., Dr. John "Ultracapacitor-based energy management strategies for eCVT hybrid vehicles."

[2] Saadi, R. B., M. et al (2013). Energy Management of Fuel Cell/ Supercapacitor Hybrid Power Sources Based on The Flatness Control. 4th International Conference on Power Engineering, Energy and Electrical Drives. Istanbul, Turkey.

[3] Dixon, J. O., Micah; Moreno, Jorge "DSP Based Ultracapacitor System for Hybrid-Electric Vehicles."

[4] Fernandes Neto, T. and P. Mutschler (2014). "Ultracapacitor Storage and Management System for Short Primary Linear Drives Applied in Automated Handling Applications." Industry Applications, IEEE Transactions on **PP**(99): 1-1.

[5] Hata, K., et al. (2013). A series or parallel changeover system using battery with EDLC for EV. Power Electronics and Applications (EPE), 2013 15th European Conference on.

[6] Sibo, W., et al. (2008). Fuzzy logic energy management strategy for supercapacitor-based energy saving system for variable-speed motor drives. Electrical Machines and Systems, 2008. ICEMS 2008. International Conference on.

[7] Hicks, J. A., et al. (2007). Ultracapacitor Energy Management and Controller Developments for a Series-Parallel 2-by-2 Hybrid Electric Vehicle. Vehicle Power and Propulsion Conference, 2007. VPPC 2007. IEEE.

[8] Gualous, H., et al. (2005). Power management of an embedded fuel cell - supercapacitor APU. Power Electronics and Applications, 2005 European Conference on.

[9] Florescu, A., et al. (2012). Results concerning ultracapacitor-based energy management strategy within electric vehicles. System Theory, Control and Computing (ICSTCC), 2012 16th International Conference on.

[10] Rengui, L., et al. (2007). "Super-Capacitor Stacks Management System With Dynamic Equalization Techniques." Magnetics, IEEE Transactions on **43**(1): 254-258.

[11] Chieh, S.-W. H., Wen-Hsien, et al (2013). Implementation and Study of Super-capacitor Cell Power Management System. Progress in Electromagnetics Research Symposium. Taipei.

[12] Sheng, C., et al. (2011). Implementation of cell balancing with super-capacitor for robot power system. Intelligent Control and Automation (WCICA), 2011 9th World Congress on.

5. CONCLUSIONS AND RECOMMENDATIONS

EDLCs have been gaining popularity in research environments for the high power density and high cycle life compared to current battery technologies. EDLCs are an excellent choice for a hybrid because they can act like a power buffer by sourcing and sinking large current spikes. This significantly reduces the stress applied to the battery pack, which is typically the most expensive component of a hybrid vehicle. EDLCs are also an excellent candidate for capturing regenerative energy because their low internal resistance allows them to charge with much higher current rates than current batteries. This means more energy can be put back into the system rather than dissipated through a power resistor or the chassis. Capturing more energy more efficiently also results in longer runtime and potentially a smaller capacity requirement. The addition of EDLCs to batteries also reduces the battery pack size while still meeting the power requirements of the application. Both of these benefits results in a less expensive system.

All of these benefits are not without a few disadvantages as well. A hybrid power system is more complex to implement and less tools and systems are currently available to implement an EDLC bank compared to a battery pack. EDLC's also exhibit behavior similar in concept to batteries, but they have significantly different operational behavior than batteries meaning they require their own management and safety systems to operate safely. One of the main differences is EDLCs have a faster dynamic response than batteries as a result of their lower internal resistance. There the safety and management systems need to have higher performance than those used for a battery pack. Currently, sufficient tools to test a single cell, manage a bank, and implement and test a bank of a full scale system do not exist. These tools are desperately needed to reduce the complexity of implementing a hybrid system.

The tools presented here meet these requirements. First, a data acquisition system was designed that could be ported from different vehicles and would characterize a current system, as well as, track improvements made as upgrades are performed. The system was designed in a modular fashion making it easy to expand and maintain. Finally, it was designed around commercially available NI cRIO hardware, which was also naturally designed to be modular and expandable to meet a wide variety of application needs. Next, a single cell test system was developed around a custom PCB and Parallax Propeller microcontroller. This system was designed to characterize a cells behavior, as well as, provide a test bench to implement control algorithms and safety systems on small scale. This reduces the risk of experimentation and allows a designer to thoroughly test the safety systems in a controlled environment to ensure robustness. Accompanying this test is some simulation examples based on free opensource software. These simulations help the designer anticipate sensor ranges and overall response of a bank. The performance of the cell model can also be verified with the physical test system to ensure accurate results from the simulation.

Finally, a multi-cell management system was developed. It was built around an Altera Cyclone IV FPGA and implements 12 series EDLC per control board. It is self-contained by running only on the power provided by the EDLC bank. It is designed to be expandable in series and parallel to support any configuration of cells necessary to fit the application. It was then tested with a series of EDLCs to ensure proper operation. Overall, this collection of tools will simplify the daunting task of implementing a hybrid power system and will hopefully encourage its use. As a result, this opens up a broader spectrum of opportunities to electrify vehicles and other mechanical systems.

Some recommendations for future work to further improve upon these systems include switching the shunt resistors used to measure current in the electric shuttle bus to hall effect sensors. This will completely electrically decouple the measurement system from the vehicle and satisfy the only requirement that was not completely met. Next, is to convert the large relays used to switch the power and load on the test system to IGBTs or MOSFETs. These have much higher cycle life than relays, and will allow for endurance cycling tests to be run without concern of reaching the usable life of the component nearly as quickly. A new feature to add to the test system would be a direct feedback between the test system and the simulations that would essentially allow a designer perform an automated system identification on the EDLC. An upgrade to make to the test system would be to convert the Propeller to a Cyclone IV FPGA as well. This serves several advantages. The FPGA is a higher performance chip, but more importantly this allows algorithms and safety systems that are developed on the test system to be directly implemented on the management system without any porting required to the code. An improvement to the management system would be to decouple the cell balancing circuit from the control board. This would allow more freedom when designing the enclosure of a bank by placing the cell balancing circuit very close to the EDLC and wiring the circuits to the control board located farther away. Second, adding a cooling system control interface to the management system would remove the requirement to have a separate controller to handle external cooling for the bank. Finally, it would be convenient to convert CVI interfaces to a free C++ implementation, so a designer does not require a license to modify the code. This would allow others to implement custom plugins to view other data relevant to their application.

APPENDIX A: EDLC CHARGE DERIVATION

A.1 Constant Voltage Source

Start by assuming we have a simple series resistor and EDLC with a constant voltage source.

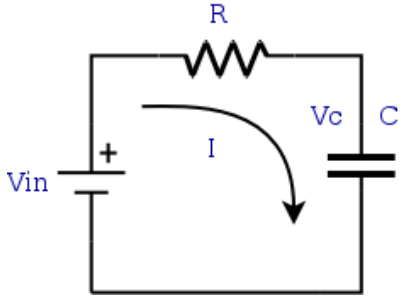

Figure A.1.1: RC Circuit
http://www.zilogic.com/blog/zkit-51-cap-meter.html

Write the differential equation for this circuit

$$R * I(t) + \frac{1}{C}\int I(t)\,dt = Vin(t) \tag{A.1.1}$$

Take the time derivative and set the derivative of the input voltage to 0 because it is constant

$$R * \frac{dI}{dt} + \frac{I(t)}{C} = 0 \tag{A.1.2}$$

Rearrange the equation

$$\frac{\left(\frac{dI}{dt}\right)}{I(t)} = -\frac{1}{RC} \tag{A.1.3}$$

The solution takes the following form:

$$\ln(I(t)) = \frac{d}{dt}\left(-\frac{1}{RC}\right) + a0 \tag{A.1.4}$$

Solve for I(t)

$$I(t) = e^{-\frac{1}{RC}t + a0} \tag{A.1.5}$$

Solve for a0 with the initial condition that I(0) = Vin/R

$$I(0) = \frac{Vin}{R} = e^{0+a0} \tag{A.1.6}$$

Therefore,

$$a0 = \ln\left(\frac{Vin}{R}\right) \tag{A.1.7}$$

Substituting back into (A.1.5) and reducing the equation results in:

$$I(t) = \frac{Vin}{R} e^{-\frac{1}{RC}t} \tag{A.1.8}$$

Now we solve for Vc(t) using the following equation

$$Vc(t) = \frac{1}{C}\int I(t)dt \tag{A.1.9}$$

Substituting and integrating gives:

$$Vc(t) = V_{in}e^{-\frac{1}{RC}t} + a1 \tag{A.1.10}$$

Solving for a1 with the initial condition that Vc(0) = 0 means a1 = 0, so the final equation is

$$Vc(t) = V_{in}e^{-\frac{1}{RC}t} \tag{A.1.11}$$

A.2 Constant Current Source

The derivation for the EDLC voltage is much easier for a constant current source. Start with the equation of the EDLC voltage.

$$Vc(t) = \frac{1}{C}\int I(t)dt \tag{A.2.1}$$

Simply integrate remembering I(t) is a constant

$$Vc(t) = \frac{1}{C}It + a1 \tag{A.2.2}$$

Now solve for the constant assuming the initial condition Vc(0) = 0. This makes a1 = 0 and the equation becomes:

$$Vc(t) = \frac{1}{C}It \tag{A.2.3}$$

APPENDIX B: EDLC TEST SYSTEM DESIGN

B.1 Schematics

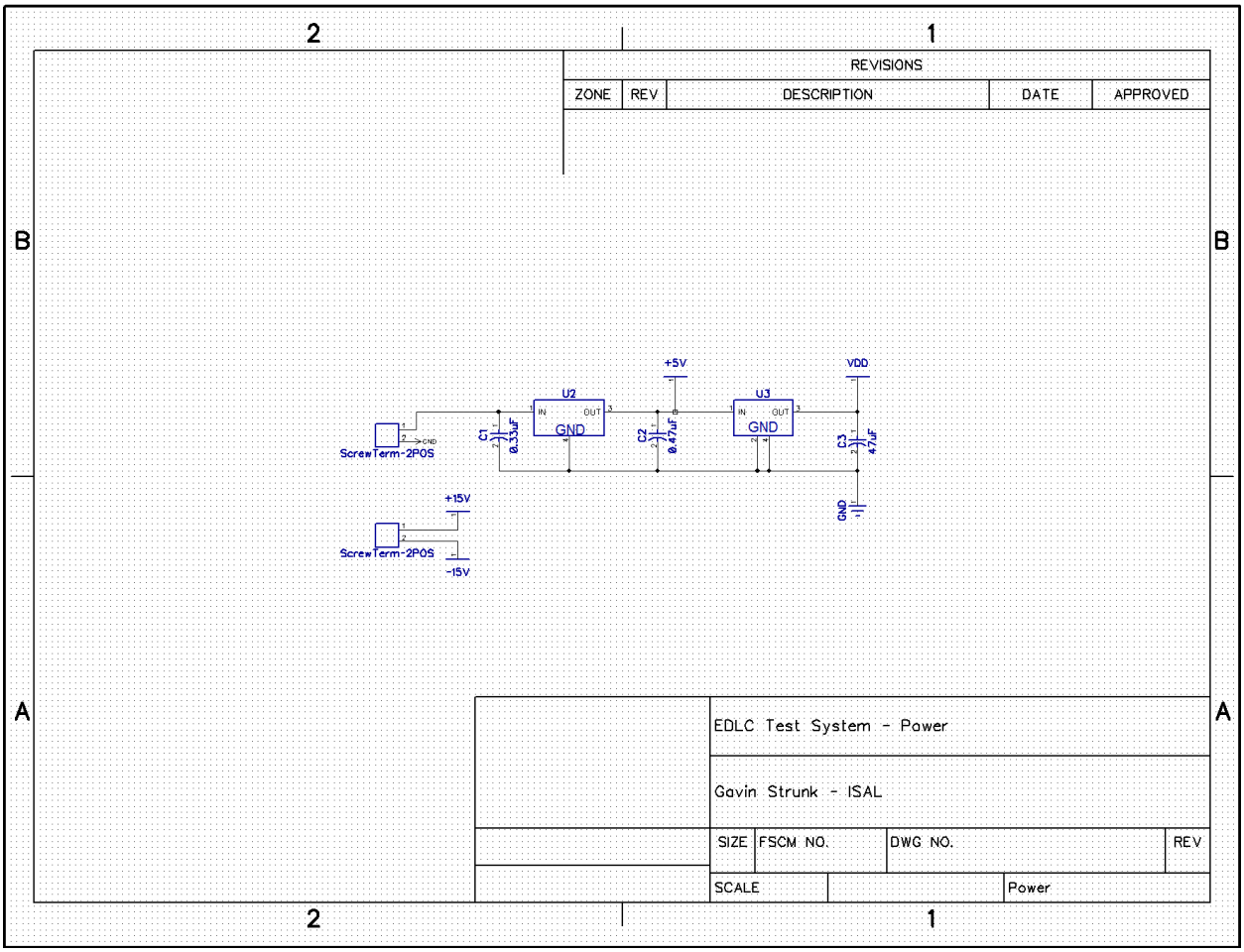

Figure B.1.1: Power Schematic for EDLC Test System

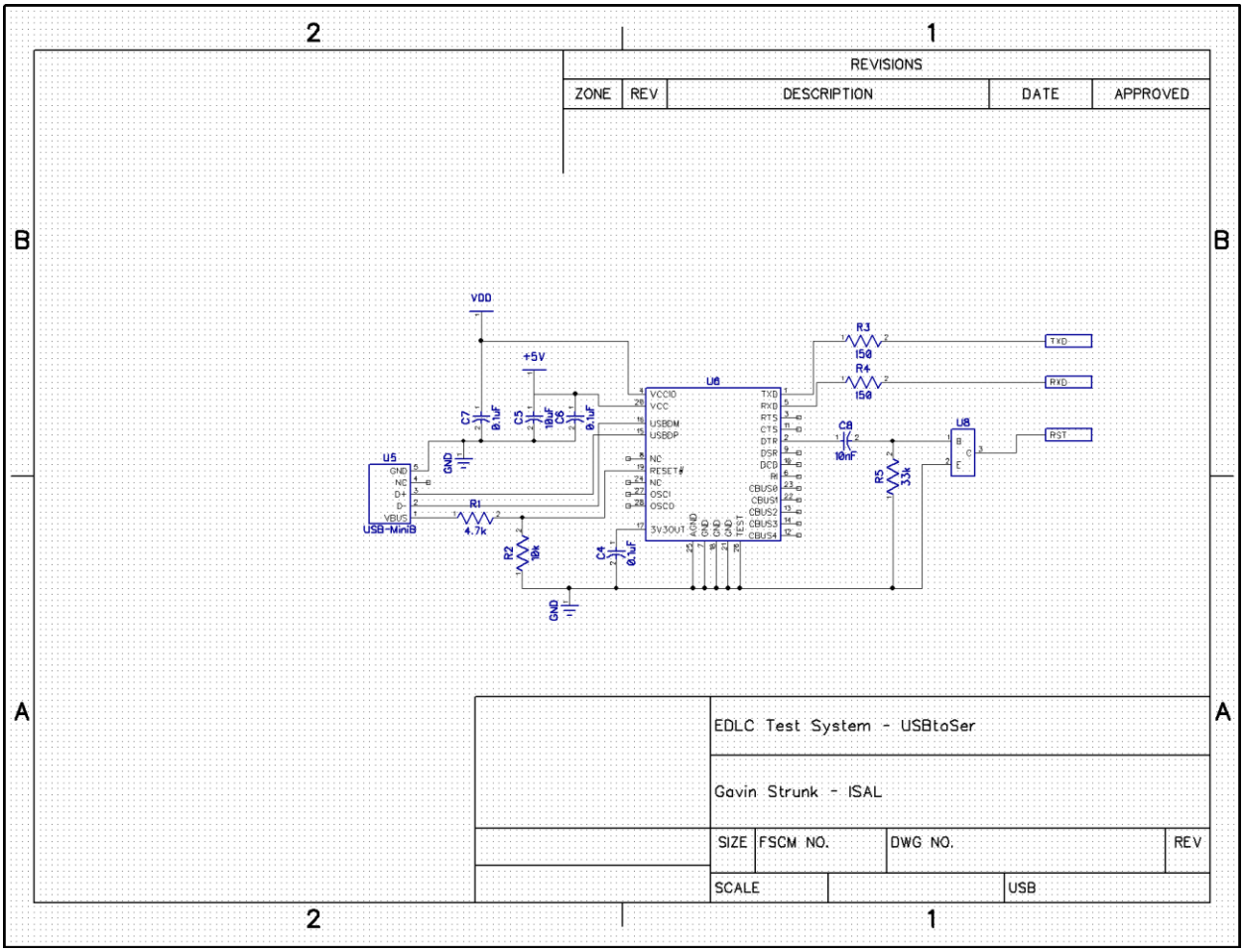

Figure B.1.2: USB to Serial Circuit for EDLC Test System

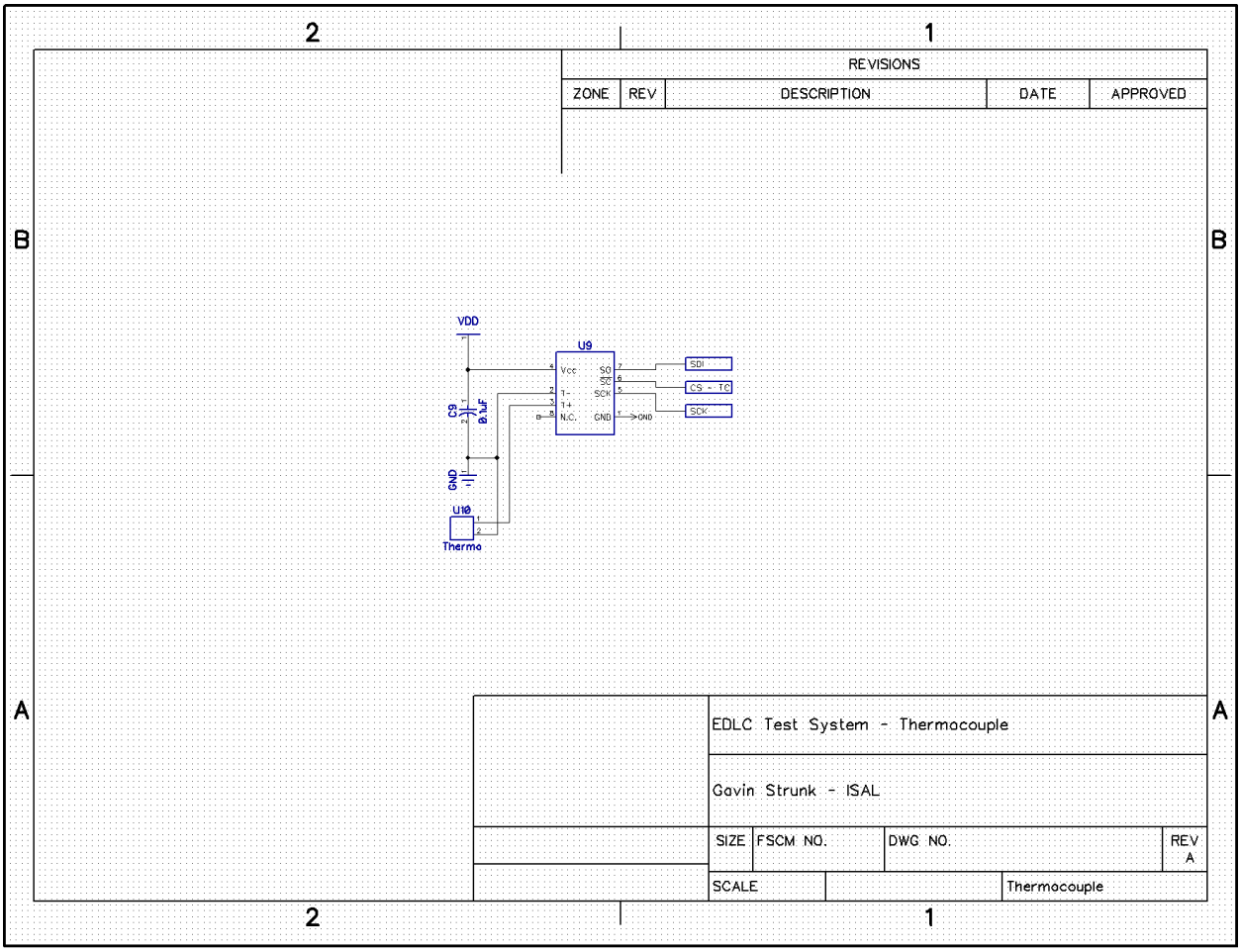

Figure B.1.3: Thermocouple Circuit for EDLC Test System

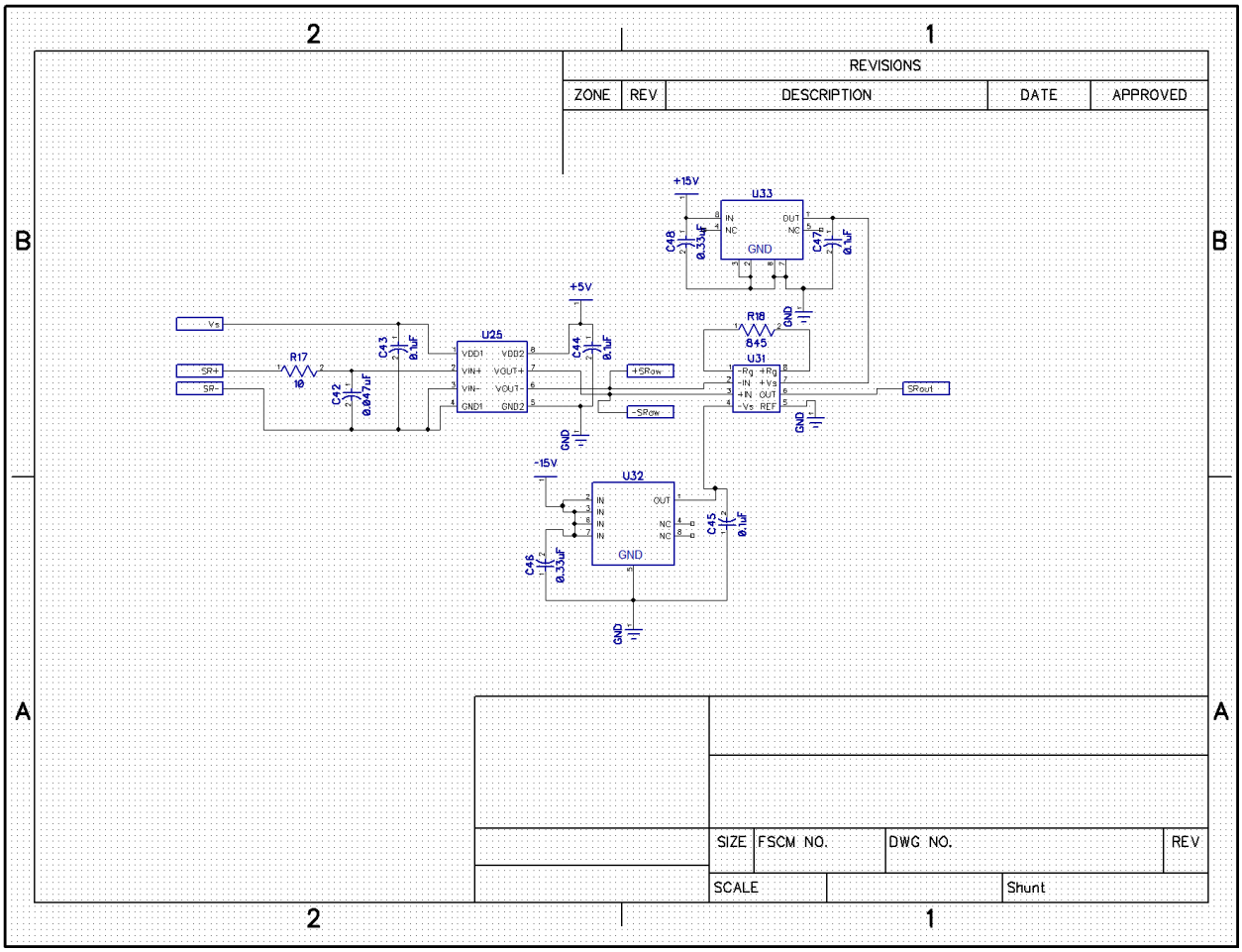

Figure B.1.4: Current Shunt Interface Circuit for EDLC Test System

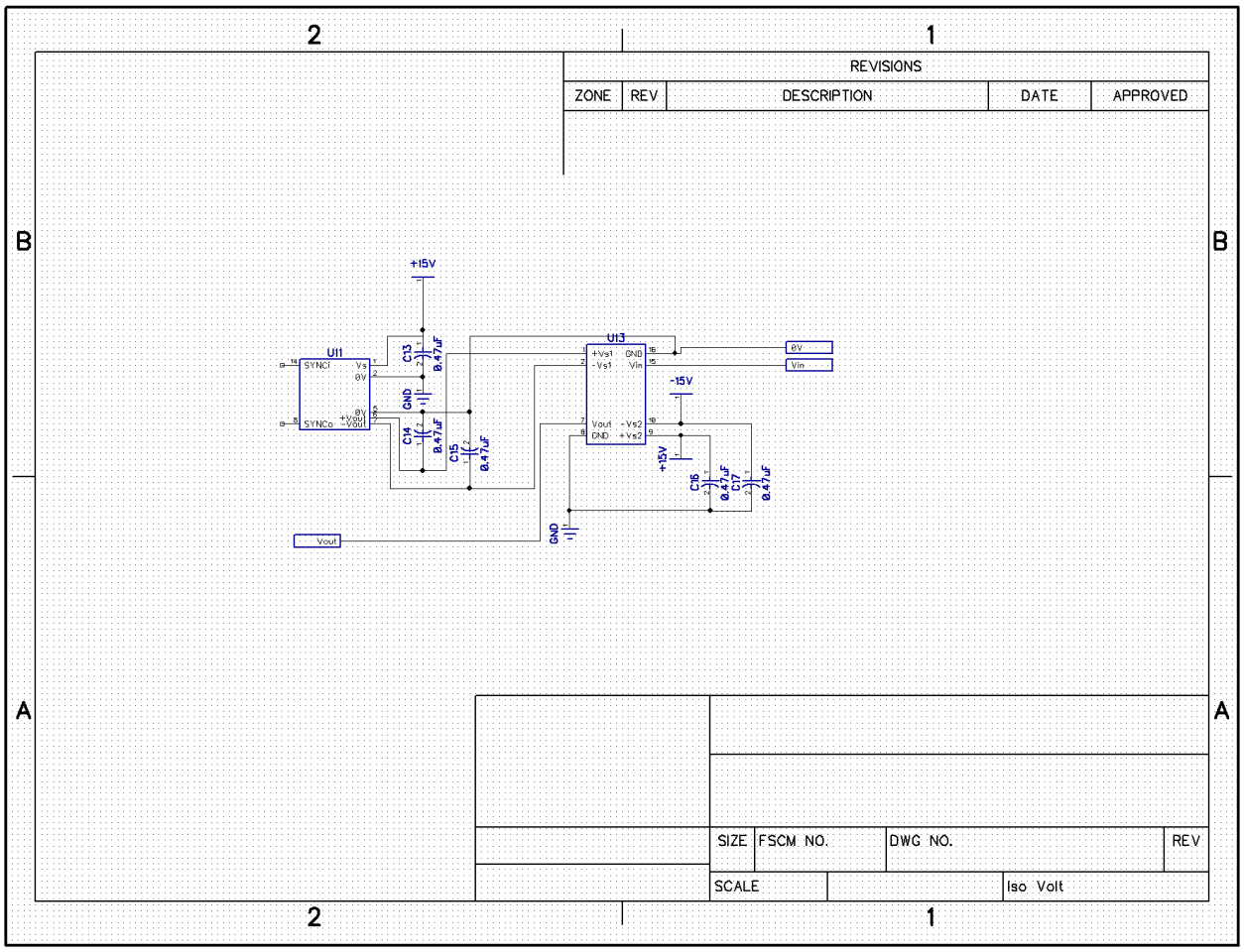

Figure B.1.5: Voltage Isolation Circuit for EDLC Test System

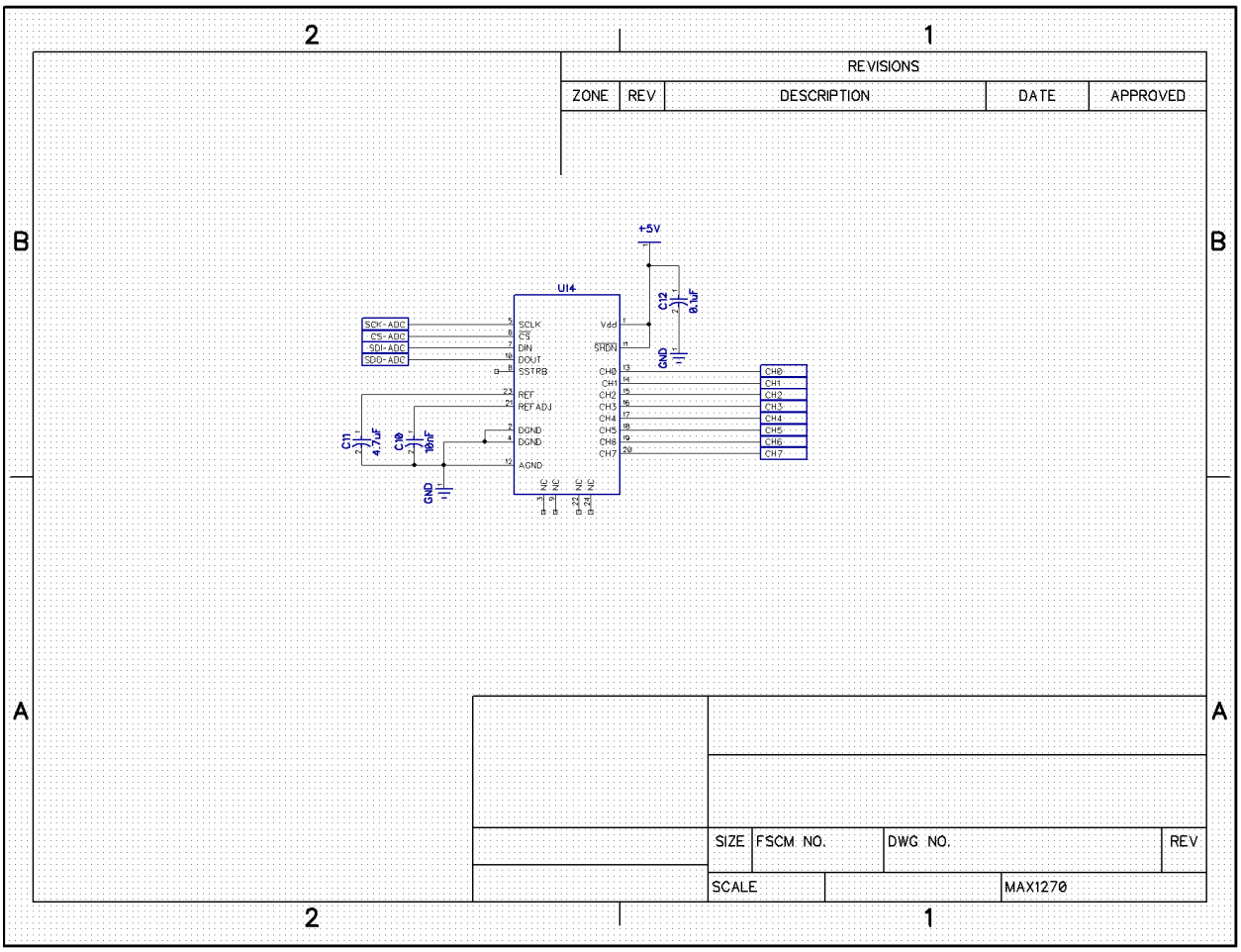

Figure B.1.6: MAX1270 A/D Circuit for EDLC Test System

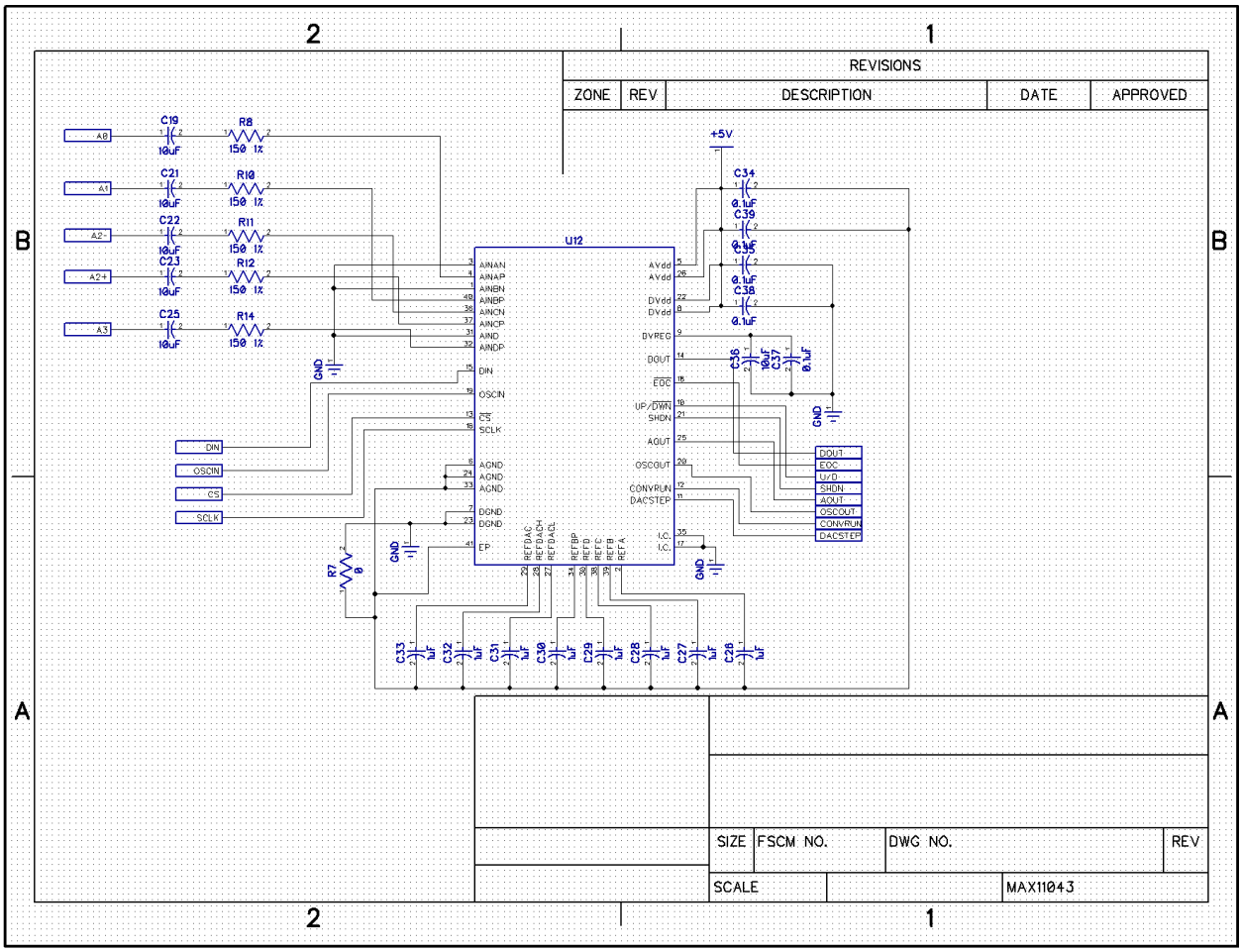

Figure B.1.7: MAX11043 A/D Converter Circuit for EDLC Test System

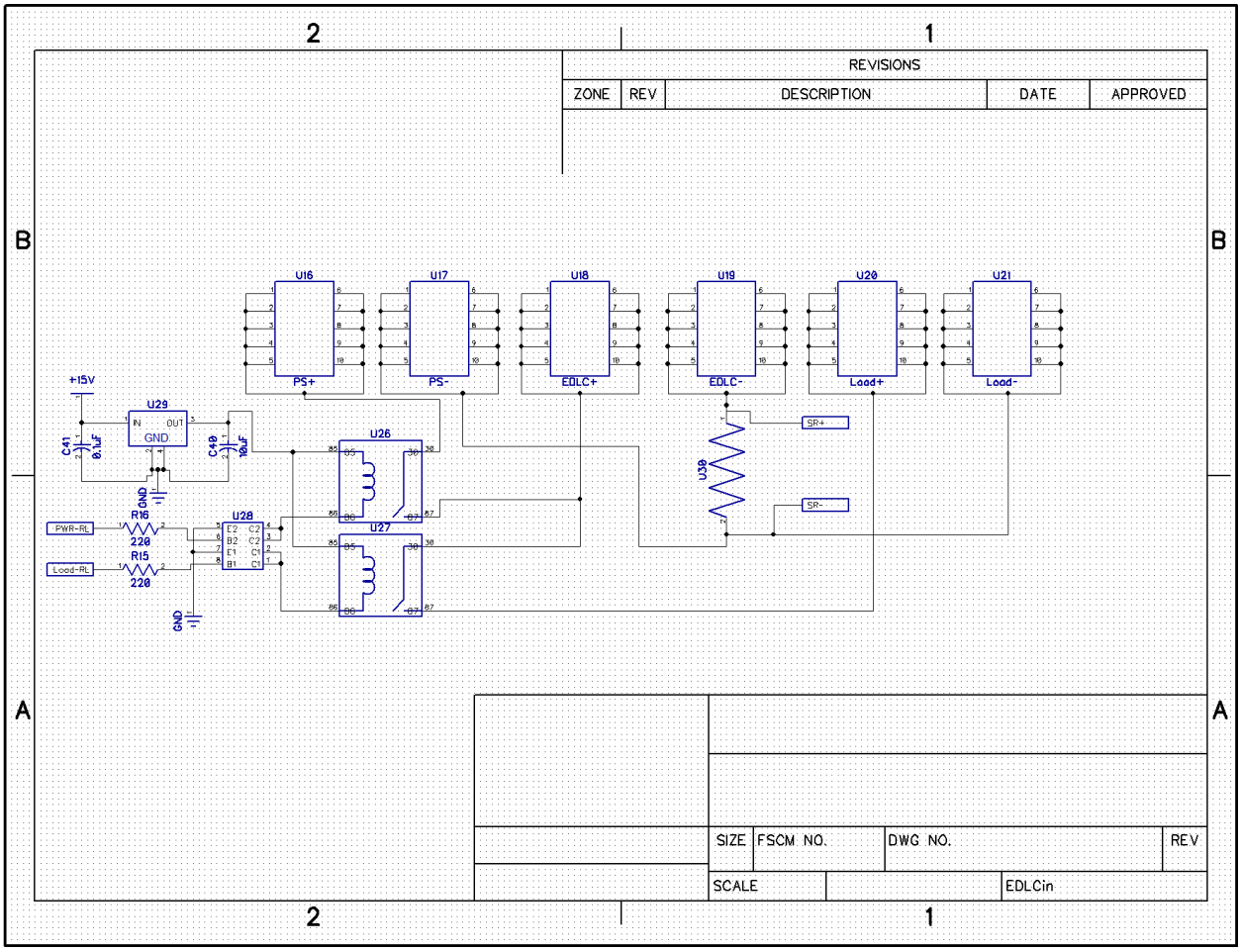

Figure B.1.8: EDLC, Power Source, and Load Interface for EDLC Test System

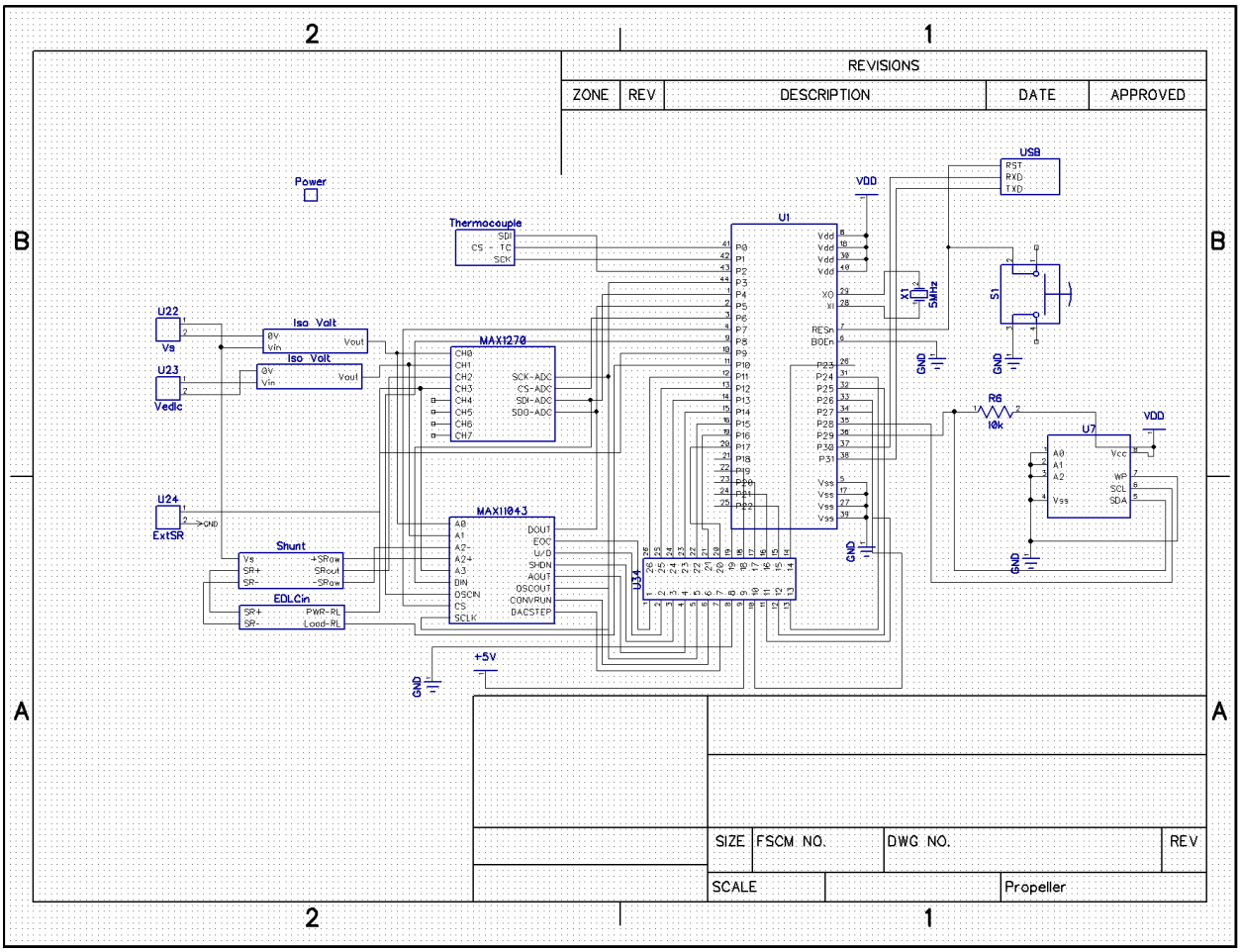

Figure B.1.9: Propeller Circuit for EDLC Test System

B.2 Layout Diagrams

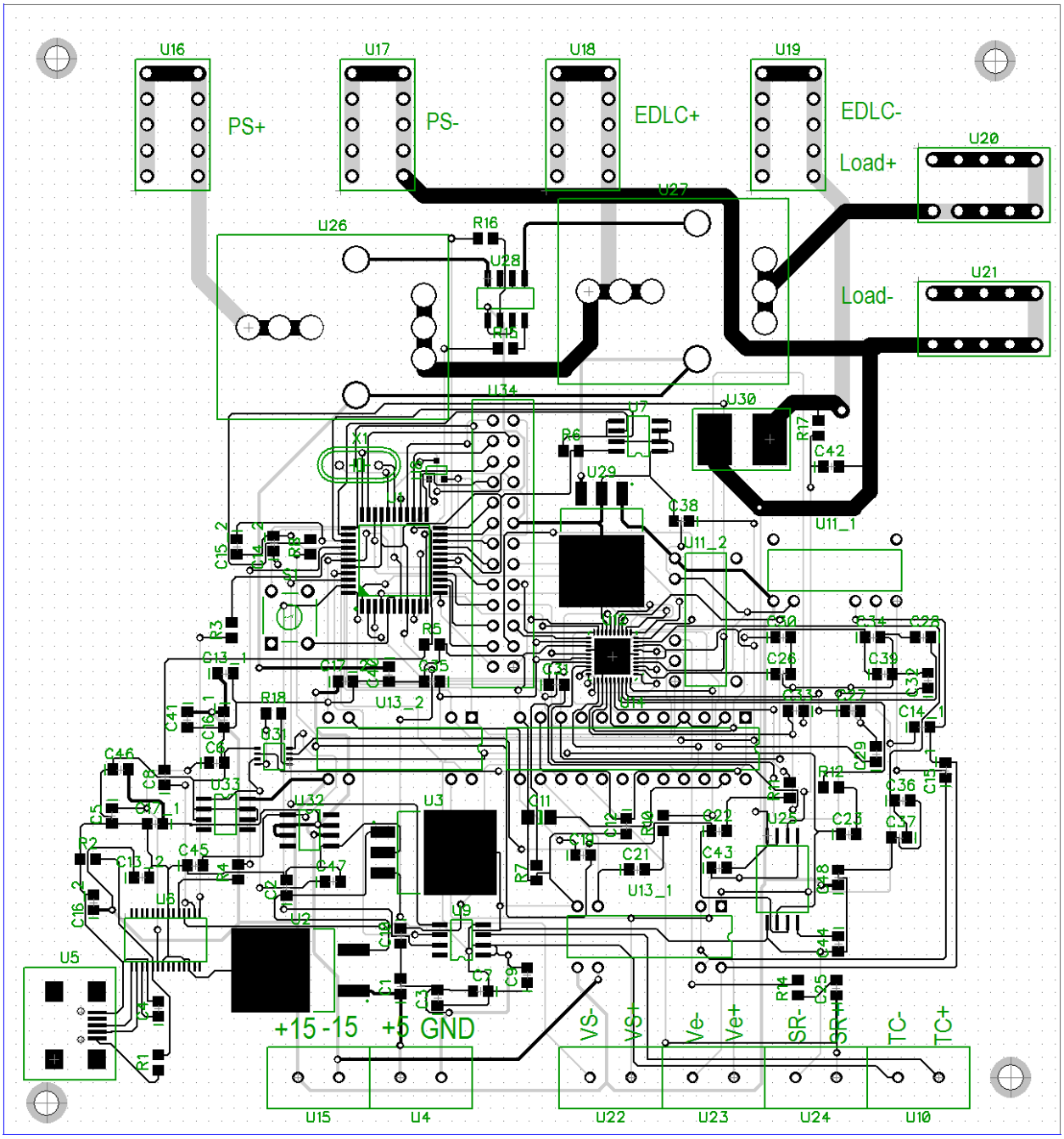

Figure B.2.1: Top Layer Layout for EDLC Test System

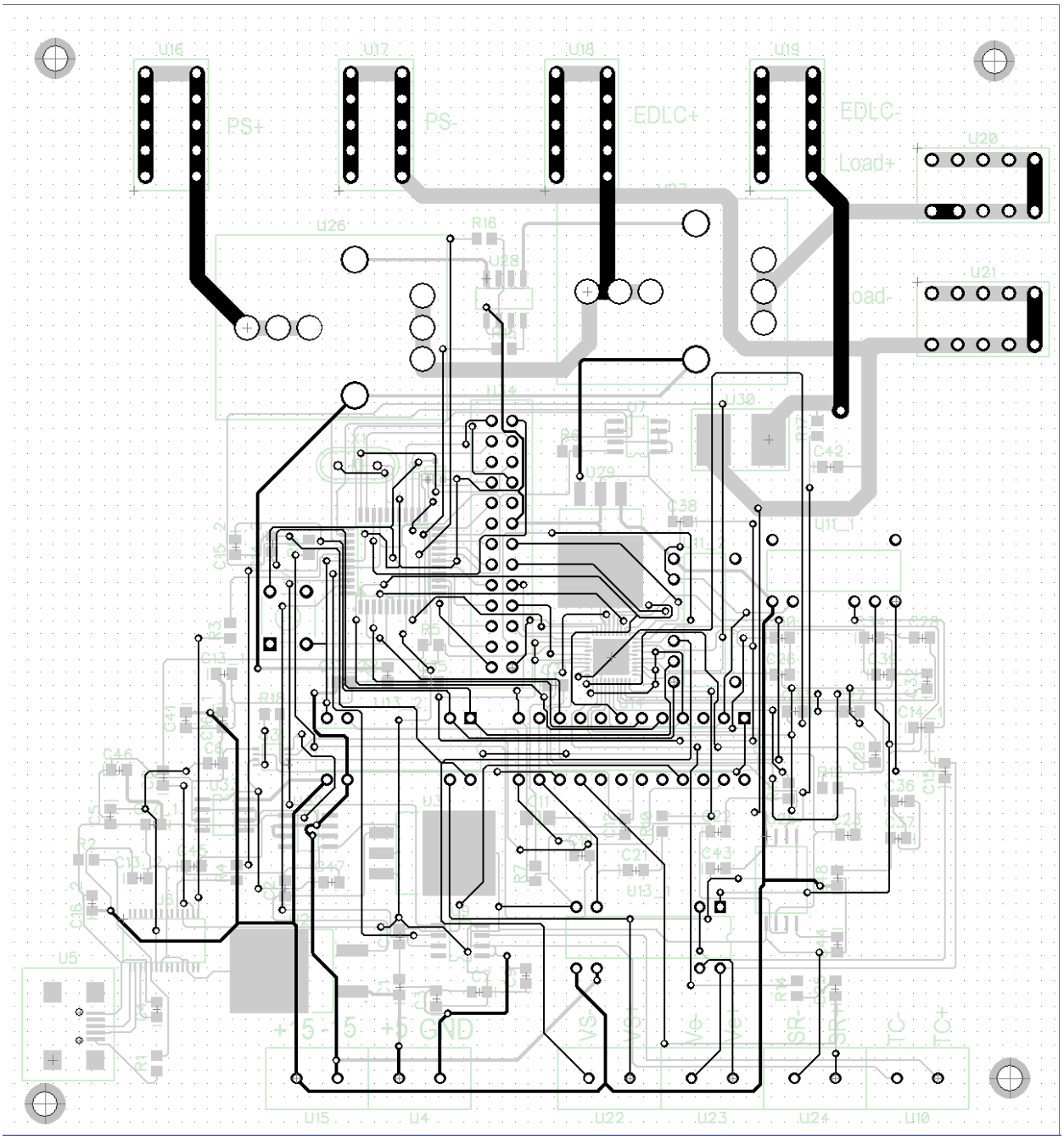

Figure B.2.2: Bottom Layer Layout for EDLC Test System

B.3 Code

B.3.1 MAX1270 A/D Driver

```
{{*****************************************************************************
 GS_MAX1270_ASM_V1.1.spin
 ========================================
 Author : Gavin Strunk
 Date   : 10 August 2011
 University of Kansas
 Intelligent Systems and Automation Lab
 ========================================

 Description : This is the assembly driver for the MAXIM 1270 A/D because
             straight spin was entirely too slow! 1.4ms come on get
             serious!  Uses external continuous clock no power-down
             supported.

 Pinout :

 Revision History
 ========================================
 V1.1 8/10 : add support for other channels for 25 clock cycle spi
           verified reads all 8 channels very stably /GS
 V1.0 8/10 : setup parameter list and assign to values
           verified working operation for ch 0  /GS

 *****************************************************************************
}}

CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

  'Constant Definitions

  'Pin Definitions

VAR
'4 million years of bad luck is waiting for anyone who changes these variables literally!
  LONG SCLK,CS,DIN,DOUT
  LONG CBYTE[8]
  LONG adval[8]
OBJ
  'dbg   :       "PASDebug"
PUB Main(_SCLK,_CS,_DIN,_DOUT)
```

```
  SCLK := _SCLK
  CS   := _CS
  DIN  := _DIN
  DOUT := _DOUT

  Initialize

  cognew(@Start,@SCLK)
  'dbg.Start(31,30,@Start)

PUB Initialize
'Setup control bytes
  CBYTE[0] := %1000_0101_0000_0000_0000_0000_0000_0000
  CBYTE[1] := %1001_0101_0000_0000_0000_0000_0000_0000
  CBYTE[2] := %1010_0101_0000_0000_0000_0000_0000_0000
  CBYTE[3] := %1011_0101_0000_0000_0000_0000_0000_0000
  CBYTE[4] := %1100_1101_0000_0000_0000_0000_0000_0000
  CBYTE[5] := %1101_1101_0000_0000_0000_0000_0000_0000
  CBYTE[6] := %1110_1101_0000_0000_0000_0000_0000_0000
  CBYTE[7] := %1111_1101_0000_0000_0000_0000_0000_0000


PUB Read_CH(chnum)

  return adval[chnum]


PUB RangeSel(_ch,_val)
'This function serves the purpose of configuring the control bytes for the channels
'ignore for now



DAT

            org       0
Start

'   long $34FC1202,$6CE81201,$83C120B,$8BC0E0A,$E87C0E03,$8BC0E0A
'   long $EC7C0E05,$A0BC1207,$5C7C0003,$5C7C0003,$7FFC,$7FF8
'assign definitions
            mov       t0,     par
            rdlong    t1,     t0
            mov       clkpin, #1
            shl       clkpin, t1
```

```
            add     t0,    #4
            rdlong  t1,    t0
            mov     cspin, #1
            shl     cspin, t1

            add     t0,    #4
            rdlong  t1,    t0
            mov     dipin, #1
            shl     dipin, t1

            add     t0,    #4
            rdlong  t1,    t0
            mov     dopin, #1
            shl     dopin, t1

            add     t0,    #4
            mov     cbptr, t0

            add     t0,    #32
            mov     adptr, t0

'setup pin states
            andn    OUTA,  clkpin
            or      OUTA,  cspin

'setup pin directions
            or      DIRA,  clkpin
            or      DIRA,  cspin
            or      DIRA,  dipin
            andn    DIRA,  dopin

'Init - to guarentee conversation
'bring cs low and clock 18 zeros
            andn    OUTA,  cspin
            mov     t2,    #18
zeros       or      OUTA,  clkpin
            andn    OUTA,  dipin
            andn    OUTA,  clkpin
            djnz    t2,    #zeros

'get control byte
            mov     chtemp, #8

Ctrbyte     rdlong  t3,    cbptr
            mov     adtemp, #0
            mov     t2,    #25
```

```
:rrl        rol     t3,    #1
            test    t3,    #1 wc
            rol     adtemp, #1

:inloop     andn    OUTA,  clkpin
            muxc    OUTA,  dipin
            or      OUTA,  clkpin
            test    dopin, ina wc
            addx    adtemp, #0
            djnz    t2,    #:rrl

            wrlong  adtemp, adptr

            add     cbptr, #4
            add     adptr, #4
            djnz    chtemp, #Ctrbyte

            sub     cbptr, #32
            sub     adptr, #32
            mov     chtemp, #8
            jmp     #Ctrbyte


t0          res     1
t1          res     1
t2          res     1
t3          res     1
t4          res     1
adtemp      res     1
chtemp      res     1

clkpin      res     1
cspin       res     1
dipin       res     1
dopin       res     1

adptr       res     1
cbptr       res     1
```

B.3.2 MAX6675 Thermocouple Driver

```
CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

VAR
```

```
  WORD temp
  LONG stack[50]

obj
  serial : "FullDuplexSerial"

PUB Main(CS,CLK,SI)

  cognew(Start(cs,clk,si),@stack)

PUB Start(cs,clk,si)|t

   temp := 0
  'initialize pins
  dira[cs]~~
  dira[clk]~~
  dira[si]~

  outa[clk]~
  outa[cs]~~

  'pulse cs to clear conversations
  outa[cs]~
  outa[cs]~~

  repeat
   t := 0
   outa[cs]~

    repeat 16
     outa[clk]~~
     t := (t << 1) + ina[si]
     outa[clk]~

   outa[cs]~~
   'serial.dec((t>>3))
   'serial.str(string(13))
   temp := (t>>3)
   waitcnt(clkfreq/2 + cnt)

pub GetTemp

  return temp
```

B.3.3 MAX11043 A/D Driver

```
CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

  CS   = 3
  CLK  = 4
  DIN  = 5
  DOUT = 6

OBJ
  serial : "Fullduplexserial"

VAR
  long ADC

PUB Start

  serial.start(31,30,0,115200)
  waitcnt(clkfreq*3 + cnt)

  serial.str(string("ADC Started",13))

  Init

  repeat
    SPI_READ($00)
    serial.dec(ADC)
    waitcnt(clkfreq/3 + cnt)

PUB Init
  'set pin states
  DIRA[CS]~~
  DIRA[CLK]~~
  DIRA[DIN]~~
  DIRA[DOUT]~

  OUTA[CS]~~
  OUTA[CLK]~

PUB SPI_WRITE(addr, value)| temp

  'format temp for the write
  temp := addr << 2
```

```
  OUTA[CS]~

  repeat 8
    OUTA[DIN] := temp & $80
    OUTA[CLK]~~
    OUTA[CLK]~
    temp <<= 1

  repeat 8
    OUTA[DIN] := value & $80
    OUTA[CLK]~~
    OUTA[CLK]~
    value <<= 1

  OUTA[CS]~~

PUB SPI_READ(addr)| temp

  ADC := 0

  'format temp for read
  temp := (addr << 2) + 2

  OUTA[CS]~

  repeat 8
    OUTA[DIN] := temp & $80
    OUTA[CLK]~~
    OUTA[CLK]~
    temp <<= 1

  repeat 16
    OUTA[CLK]~~
    ADC := (ADC << 1) + INA[DOUT]
    OUTA[CLK]~

  OUTA[CS]~~
```

B.3.4 Main Embedded Application

```
{{GS_UltraCapacitorTestBed_V5.0.spin
  ============================
  Author: Gavin Strunk
  Date  : 4 June 2014
  University of Kansas ISAL
```

```
  Description:This code reads the sensor network and converts the readings into the string format used to
          transmit the data into LabVIEW.

  PINOUTS:

  Revision History:
  \GS 6/15/11 V1.0 : First Draft
  \GS 8/13/11 V2.0 : Added maxim bipolar a/d
  \GS 4/22/12 V3.0 : Added a load and relays to control charge and discharge flow, MAX6675 now does temperature,
                     also changed the encoding format to reduce overhead
  \GS        V4.0 : Added support for the upgraded hardware
  \GS 6/4/14  V5.0 : Removed support for upgraded hardware and added support for new CVI computer software
}}
CON
  _clkmode = xtal1 + pll16x
  _xinfreq = 5_000_000

  'Thermocouple SPI
  T_CS   = 0
  T_SCLK = 1
  T_DI   = 2

  'ADC SPI
  SCLK  = 3
  SDI   = 4
  SDO   = 5
  CS    = 6

  Power_Relay = 9
  Load_Relay  = 10

  VS  = $2000
  VC  = $4000
  IC  = $6000
  TC  = $8000
  TML = $A000
  TMH = $C000

  Baud = 9600

OBJ
  serial : "FullDuplexSerial"
  ADC    : "GS_MAX1270_ASM_V1.1"
```

```
  TEMP   : "GS_MAX6675_SPIN_V1.0"
VAR
  LONG _vs,_vc,_ic,_tc
  LONG time
  BYTE data[40]
  long running
PUB Start | t,state

  running := 0
  dira[8]~~
  dira[7]~~
  DIRA[Power_Relay]~~
  DIRA[Load_Relay]~~

  OUTA[Power_Relay]~
  OUTA[Load_Relay]~

  serial.start(31,30,0,Baud)
  ADC.Main(SCLK,CS,SDI,SDO)
  TEMP.Main(T_CS,T_SCLK,T_DI)
  waitcnt(3*clkfreq + cnt)

  repeat
    if (running == 0)
      state := serial.rxtime(1)
      if(state <> -1)
        serial.tx(state)
        SetState(state)
    else
      Read_ADC
      Build_STR(_vs,_vc,_ic,_tc)
      serial.str(@data[0])
      'serial.str(string(","))
      'serial.dec(_vs + VS)
      'serial.str(string(","))
      'serial.str(string(13))
      'serial.dec(_vc + VC)
      'serial.str(string(","))
      'serial.str(string(13))
      'serial.dec(_ic + IC)
      'serial.str(string(","))
      'serial.str(string(13))
      'serial.dec(_tc + TC)
      serial.str(string(13))
      state := serial.rxtime(1)
      if(state <> -1)
```

```
      SetState(state)
    else
      outa[8]~
    waitcnt(clkfreq/100 + cnt)



PRI Build_STR(a,b,c,d)|t1,t2,t3,i,j,k, check
  '%###,####,####,####*YY
  bytefill(@data[0], 0, 40)
  data[0] := "%"

  i  := 0
  j  := 1
  check := 0 ^ data[0]
  repeat i from 0 to 3
    t1 := 1000
    k := 0
    t3 := a
    case i
      0:
        t3 := a
      1:
        t3 := b
      2:
        t3 := c
      3:
        t3 := d

    repeat until (t1 == 0)
      t2 := t3/t1
      if(t2 > 0)
        data[j] := t2 + "0"
        check ^= data[j]
        k := 1
        j += 1
      elseif((t2 == 0) and (k == 1))
        data[j] := "0"
        check ^= data[j]
        j += 1
      elseif((t1 == 1) and (k == 0))
        data[j] := "0"
        check ^= data[j]
        j += 1
      t3  //= t1
      if(t1 == 1)
```

```
        t1 := 0
       else
        t1 /= 10
     if(i < 3)
      data[j] := ","
      check ^= data[j]
      j += 1
     else
      data[j] := "*"
      check ^= data[j]
   ConvertIntToHex(check, 2, @data[j + 1])

PRI Read_ADC
   _vs := ADC.Read_CH(0)
   _vc := ADC.Read_CH(1)

   'turn off if above 2.5V
   if((_vc => 1024) and (_vc <2047))
    SetState(52)

   _ic := ADC.Read_CH(3)
   _tc := TEMP.GetTemp

PRI SetState(button)
  'serial.dec(button)
  'set the states of the relays but don't let them both be on at the same time
  outa[8]~~
  case (button)
   48: 'stop
    running := 0
    OUTA[Power_Relay]~
    OUTA[Load_Relay]~
    outa[7]~
   49: 'start
    running := 1
    OUTA[Power_Relay]~
    OUTA[Load_Relay]~
    outa[7]~
   50: 'power
    if(running == 1)
     OUTA[Load_Relay]~
     OUTA[Power_Relay]~~
     outa[7]~
   51: 'load
    if(running == 1)
     OUTA[Power_Relay]~
```

```
      OUTA[Load_Relay]~~
   52: 'off
     if(running == 1)
      OUTA[Power_Relay]~
      OUTA[Load_Relay]~
      outa[7]~
   other:
     outa[7]~

PRI ConvertIntToHex(value, digits, buf)| i
  i := buf
  value <<= (8 - digits) << 2
  repeat digits                                 'do it for the number of hex digits being transmitted
    byte[i] := (lookupz((value <-= 4) & $F : "0".."9", "A".."F"))
    i += 1
```

B.3.5 CVI Application

```
//=====================================================================
==========
//
// Title:          EDLCTestGui
// Purpose:                A short description of the application.
//
// Created on:  5/31/2014 at 8:02:40 AM by jaycomp.
// Copyright:    . All Rights Reserved.
//
//=====================================================================
==========

//=====================================================================
==========
// Include files

#include <ansi_c.h>
#include <cvirte.h>
#include <userint.h>
#include <stdlib.h>
#include <string.h>
#include <utility.h>
#include "EDLCTestGui.h"
#include "toolbox.h"
#include "pathctrl.h"
#include "LocateCOMCVI.h"
#include "SerialComms.h"
#include "TDMSFileIo.h"
```

```
#include "ProjectDefs.h"

//Main Application
static          int panelHandle = 0;
static          int running = 0;
volatile        int quit = 0;
static          char currentcmd = 0;

static char*    InitMsg = "Application Started\n";
static char*    STARTLABEL = "START";
static char*    STOPLABEL  = "STOP";

static CmtTSQHandle datatsqhdl;
static CmtTSQHandle filetsqhdl;
static CmtTSQHandle cmdtsqhdl;
//struct DATAPACK dataqueue;
//struct DATAPACK filequeue;
//static int                cmdqueue;

//TDMS
static struct FileInfo    fileinf;

//Main
struct CMDPACK cmd = {0};
void startDimControl(void);
void startLogic(void);
void stopLogic(void);
void CVICALLBACK readDataQueue(int queueHandle, unsigned int event, int value, void
*callbackData);


//Debug Functions
void printInt(int value);
void printDouble(double value);
void printHex(int value);
void printString(char *value);

//DAQ Thread
static int daqhandle;
static CmtTSQCallbackID datacbid;
int CVICALLBACK DaqThread(void *functionData);
double convertAdcValue(double data);
void CVICALLBACK readCmdQueue(int queueHandle, unsigned int event, int value, void
*callbackData);

//File IO Thread
```

```
static int filehandle;
static CmtTSQCallbackID filecbid;
int     CVICALLBACK FileIOThread(void *functionData);
void    CVICALLBACK readFileQueue(int queueHandle, unsigned int event, int value, void
*callbackData);


int main (int argc, char *argv[])
{
        int error = 0;

        /* initialize and load resources */
        nullChk (InitCVIRTE (0, argv, 0));
        errChk (panelHandle = LoadPanel (0, "EDLCTestGui.uir", PANEL));

        //Set initial states of the panel controls
        SetCtrlAttribute(panelHandle, PANEL_START, ATTR_DIMMED, 1);
        SetCtrlVal(panelHandle, PANEL_MSG, InitMsg);

        getAvailableComPorts(panelHandle, PANEL_COM, PANEL_MSG);

        //add panel control
        NewPathCtrl(panelHandle, PANEL_FILE, 8, 1);

        //init the thread queues
        CmtNewTSQ(1, sizeof(struct DATAPACK), OPT_TSQ_DYNAMIC_SIZE,
&datatsqhdl);
        CmtNewTSQ(1, sizeof(struct DATAPACK), OPT_TSQ_DYNAMIC_SIZE,
&filetsqhdl);
        CmtNewTSQ(1, sizeof(char), OPT_TSQ_DYNAMIC_SIZE, &cmdtsqhdl);

        //install read callback
        CmtInstallTSQCallback(datatsqhdl, EVENT_TSQ_ITEMS_IN_QUEUE, 1,
readDataQueue, 0, CmtGetCurrentThreadID(), NULL);

        /* display the panel and run the user interface */
        errChk (DisplayPanel (panelHandle));
        errChk (RunUserInterface ());

Error:

        CmtDiscardTSQ(datatsqhdl);
        CmtDiscardTSQ(filetsqhdl);
        CmtDiscardTSQ(cmdtsqhdl);
        if (panelHandle > 0)
                DiscardPanel (panelHandle);
```

```
	return 0;
}

//Thread Functions
//==============================================================================
==========
int CVICALLBACK DaqThread(void *functionData)
{
	int comnum, baud;
	BOOL ret = FALSE;
	struct DATAPACK data;

	CmtInstallTSQCallback(cmdtsqhdl, EVENT_TSQ_ITEMS_IN_QUEUE, 1,
readCmdQueue, 0, CmtGetCurrentThreadID(), &datacbid);

	GetCtrlVal(panelHandle, PANEL_COM, &comnum);
	GetCtrlVal(panelHandle, PANEL_BAUDRATE, &baud);
	openComPort(comnum, baud, panelHandle, PANEL_MSG);

	//writeCharCommand('1');
	while(!quit)
	{
		ProcessSystemEvents();
		ret = isDataAvailable();
		//SetCtrlVal(panelHandle, PANEL_CHARGE, TRUE);
		if(ret == TRUE)
		{
			data = getData();
			SetCtrlVal(panelHandle, PANEL_MSG, "Data Received\n");

			//convert the data to real values
			data.vsrc = convertAdcValue(data.vsrc);
			data.vedlc = convertAdcValue(data.vedlc);
			data.iedlc = (convertAdcValue(data.iedlc)*10)/8.2;
			data.tedlc = (data.tedlc / 4);

			CmtWriteTSQData(datatsqhdl, &data, 1, TSQ_INFINITE_TIMEOUT, 0);
			//PlotStripChartPoint(panelHandle, PANEL_TEMPGRAPH, data.tedlc);
			//SetCtrlVal(panelHandle, PANEL_CHARGE, FALSE);
		}
	}

	//writeCharCommand('0');
	closeComPort();
	CmtUninstallTSQCallback(cmdtsqhdl,datacbid);
	return 0;
```

```
}

void CVICALLBACK readCmdQueue(int queueHandle, unsigned int event, int value, void
*callbackData)
{
	char cmd;
	CmtReadTSQData(cmdtsqhdl, &cmd, 1, TSQ_INFINITE_TIMEOUT, 0);
	CmtFlushTSQ(cmdtsqhdl, TSQ_FLUSH_ALL, NULL);
	writeCharCommand(cmd);
}

double convertAdcValue(double data)
{
	//convert from bipolar binary to +-
	if(data > 2047)
	{
		data = 4096 - data;
		data *= -1;
	}

	return ((data/2048)*5.0);
}

int CVICALLBACK FileIOThread(void *functionData)
{
	BOOL val = FALSE;

	CmtInstallTSQCallback(filetsqhdl, EVENT_TSQ_ITEMS_IN_QUEUE, 1,
readFileQueue, 0, CmtGetCurrentThreadID(), &filecbid);
	while(!quit)
	{
		SetCtrlVal(panelHandle, PANEL_DISCHARGE, val);
		if(val == FALSE)
			val = TRUE;
		else
			val = FALSE;
		Delay(1);
	}

	CmtUninstallTSQCallback(filetsqhdl, filecbid);
	return 0;
}

void CVICALLBACK readFileQueue(int queueHandle, unsigned int event, int value, void
*callbackData)
{
```

```
}

//==========================================================================
==========
// UI callback function prototypes

/// HIFN Exit when the user dismisses the panel.
int CVICALLBACK panelCB (int panel, int event, void *callbackData,
		int eventData1, int eventData2)
{
	if (event == EVENT_CLOSE)
	{
		if(running == 1)
			stopLogic();
		QuitUserInterface (0);
	}
	return 0;
}

int CVICALLBACK handleBrowse (int panel, int control, int event,
                                                        void *callbackData, int eventData1, int
eventData2)
{
	char filebuf[512];

	switch (event)
	{
		case EVENT_COMMIT:
			if(FileSelectPopup("", "*.tdms", "*.tdms", "Enter TDM Streaming File",
VAL_OK_BUTTON,
                                0,0,1,1,filebuf) > 0)
                                SetCtrlVal(panelHandle, PANEL_FILE, filebuf);

                        startDimControl();
                        break;
	}
	return 0;
}

int CVICALLBACK handleStart (int panel, int control, int event,
                                                        void *callbackData, int eventData1, int
eventData2)
{
	switch (event)
	{
```

```
		case EVENT_COMMIT:
			if(running == 0)
			{
				startLogic();
				currentcmd = '1';
				CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
			}
			else
			{
				currentcmd = '0';
				CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
				stopLogic();
			}
			break;
	}
	return 0;
}

int CVICALLBACK handlePowerSwitch (int panel, int control, int event,
                                   void *callbackData, int
eventData1, int eventData2)
{
	char currentcmd;
	switch (event)
	{
		case EVENT_COMMIT:
			if(running == 1)
			{
				GetCtrlVal(panelHandle, PANEL_POWERSW,
&cmd.powerswitch);
				if(cmd.loadswitch == 1)
				{
					cmd.loadswitch = 0;
					SetCtrlVal(panelHandle, PANEL_LOADSW,
cmd.loadswitch);
					currentcmd = '2';
					CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
				}
				else
				{
					currentcmd = '4';
					CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
```

```
                        }
                    }
                    break;
        }
        return 0;
}

int CVICALLBACK handleLoadSwitch (int panel, int control, int event,
                                  void *callbackData, int eventData1,
int eventData2)
{
        switch (event)
        {
            case EVENT_COMMIT:
                if(running == 1)
                {
                    GetCtrlVal(panelHandle, PANEL_LOADSW, &cmd.loadswitch);
                    if(cmd.powerswitch == 1)
                    {
                        cmd.powerswitch = 0;
                        SetCtrlVal(panelHandle, PANEL_POWERSW,
cmd.powerswitch);
                        currentcmd = '3';
                        CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
                    }
                    else
                    {
                        currentcmd = '4';
                        CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
                    }
                }
                break;
        }
        return 0;
}

int CVICALLBACK handleFile (int panel, int control, int event,
                            void *callbackData, int eventData1, int
eventData2)
{
        switch (event)
        {
            case EVENT_VAL_CHANGED:
                startDimControl();
```

```
                        break;
        }
        return 0;
}

int CVICALLBACK handleRefresh (int panel, int control, int event,
                                                        void *callbackData, int eventData1, int
eventData2)
{
        switch (event)
        {
                case EVENT_COMMIT:
                        DeleteListItem(panelHandle, PANEL_COM, 0, -1);
                        getAvailableComPorts(panelHandle, PANEL_COM, PANEL_MSG);
                        startDimControl();
                        break;
        }
        return 0;
}


//FUNCTIONS
//=========
void startDimControl(void)
{
        int item;
        GetCtrlIndex(panelHandle, PANEL_COM, &item);
        GetCtrlVal(panelHandle, PANEL_FILE, fileinf.filePath);

        if((fileinf.filePath[0] != NULL) && (item != -1))
                SetCtrlAttribute(panelHandle, PANEL_START, ATTR_DIMMED, 0);
        else
                SetCtrlAttribute(panelHandle, PANEL_START, ATTR_DIMMED, 1);
}
void startLogic(void)
{
        quit = 0;
        GetCtrlVal(panelHandle, PANEL_FILE, fileinf.filePath);
        CmtScheduleThreadPoolFunction(DEFAULT_THREAD_POOL_HANDLE,
DaqThread, NULL, &daqhandle);
        CmtScheduleThreadPoolFunction(DEFAULT_THREAD_POOL_HANDLE,
FileIOThread, NULL, &filehandle);
        printString("Start Logic\n");
        SetCtrlAttribute(panelHandle, PANEL_START, ATTR_LABEL_TEXT, STOPLABEL);
        running = 1;
}
```

```
void stopLogic(void)
{
	quit = 1;
	CmtWaitForThreadPoolFunctionCompletion(DEFAULT_THREAD_POOL_HANDLE,
daqhandle, 0);
	CmtWaitForThreadPoolFunctionCompletion(DEFAULT_THREAD_POOL_HANDLE,
filehandle, 0);
	SetCtrlAttribute(panelHandle, PANEL_START, ATTR_LABEL_TEXT,
STARTLABEL);
	running = 0;
}

void CVICALLBACK readDataQueue(int queueHandle, unsigned int event, int value, void
*callbackData)
{
	SetCtrlVal(panelHandle, PANEL_MSG, ".");
	struct DATAPACK data;

	CmtReadTSQData(datatsqhdl, &data, 1, TSQ_INFINITE_TIMEOUT, 0);

	double y[2];
	y[0] = data.vsrc;
	y[1] = data.vedlc;

	PlotStripChart(panelHandle, PANEL_VOLTGRAPH, y, 2, 0, 0, VAL_DOUBLE);
	PlotStripChartPoint(panelHandle, PANEL_CURRENTGRAPH, data.iedlc);
	PlotStripChartPoint(panelHandle, PANEL_TEMPGRAPH, data.tedlc);
}

//Debug Functions
void printDouble(double value)
{
	char buf[50];
	sprintf(buf, "%f\n", value);
	SetCtrlVal(panelHandle, PANEL_MSG, buf);
}

void printHex(int value)
{
	char buf[50];
	sprintf(buf, "%X\n", value);
	SetCtrlVal(panelHandle, PANEL_MSG, buf);
}

void printInt(int value)
```

```
{
	char buf[50];
	sprintf(buf, "%i\n", value);
	SetCtrlVal(panelHandle, PANEL_MSG, &buf[0]);
}

void printString(char *value)
{
	SetCtrlVal(panelHandle, PANEL_MSG, value);
}


int CVICALLBACK handleResend (int panel, int control, int event,
                                                        void *callbackData, int eventData1, int
eventData2)
{
	switch (event)
	{
		case EVENT_COMMIT:
			CmtWriteTSQData(cmdtsqhdl, &currentcmd, 1,
TSQ_INFINITE_TIMEOUT, 0);
			break;
	}
	return 0;
}
```

APPENDIX C: PROGRAMMABLE LOAD DESIGN

C.1 Schematics

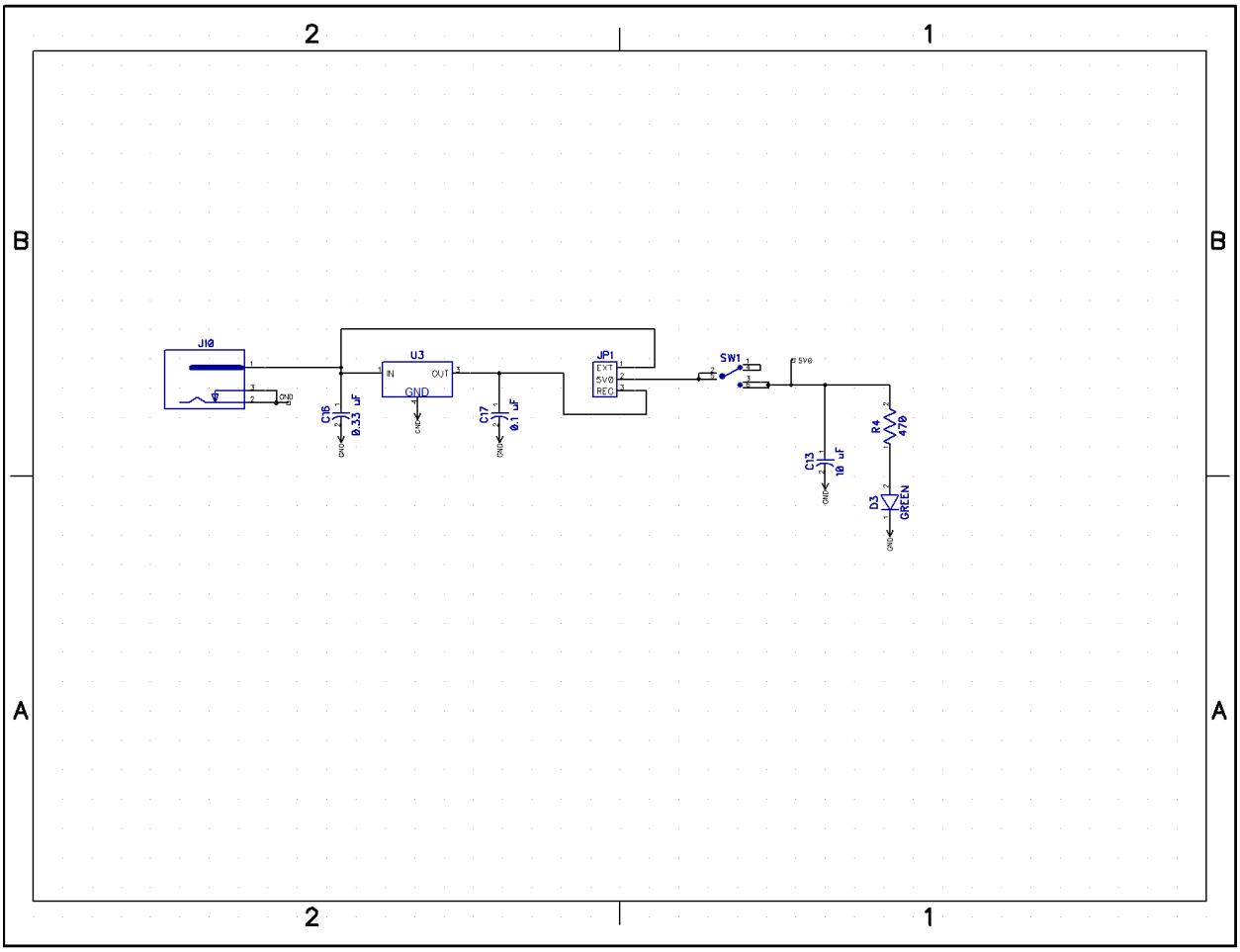

Figure C.1.1: Power Circuit for Programmable Load

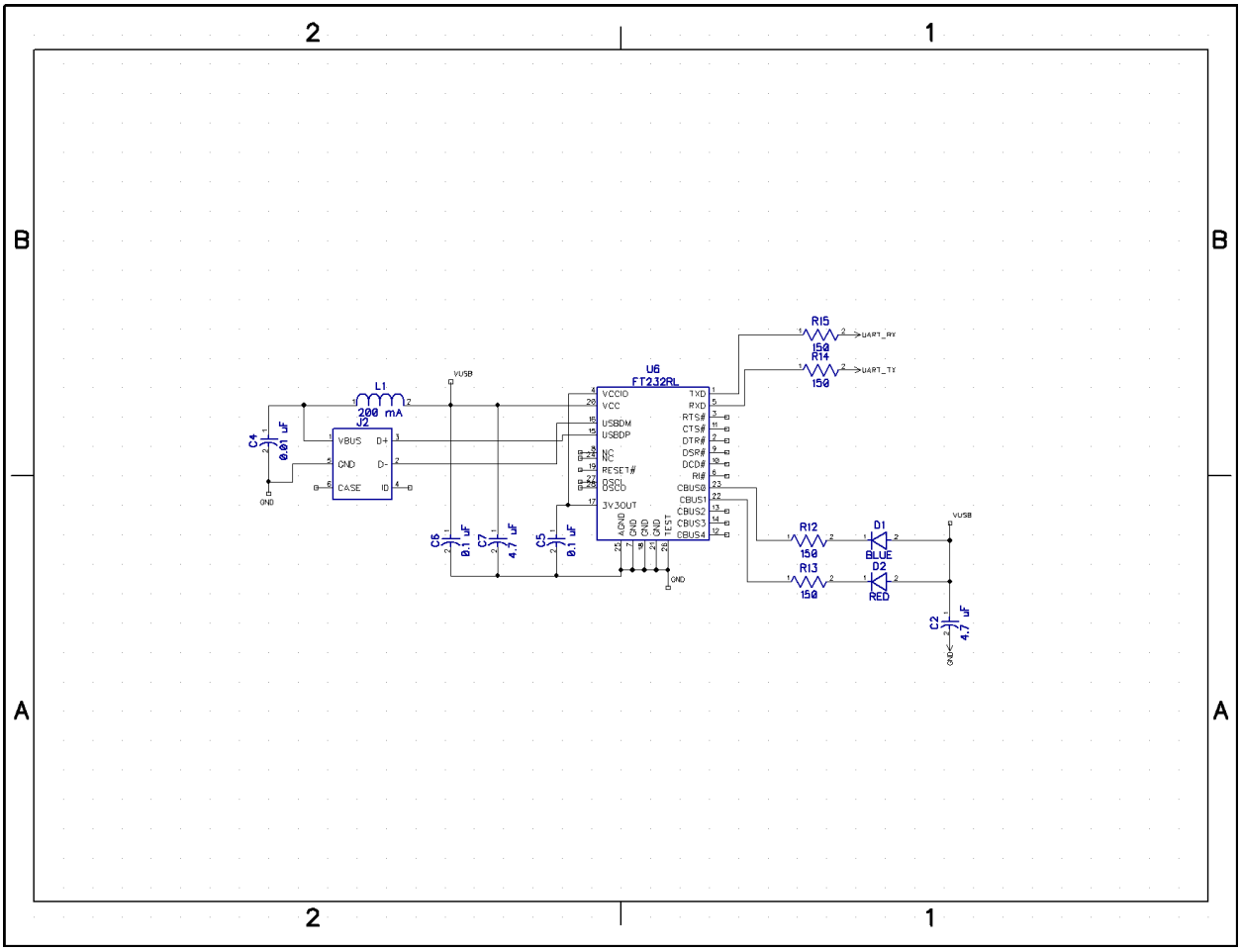

Figure C.1.2: USB to Serial Circuit for Programmable Load

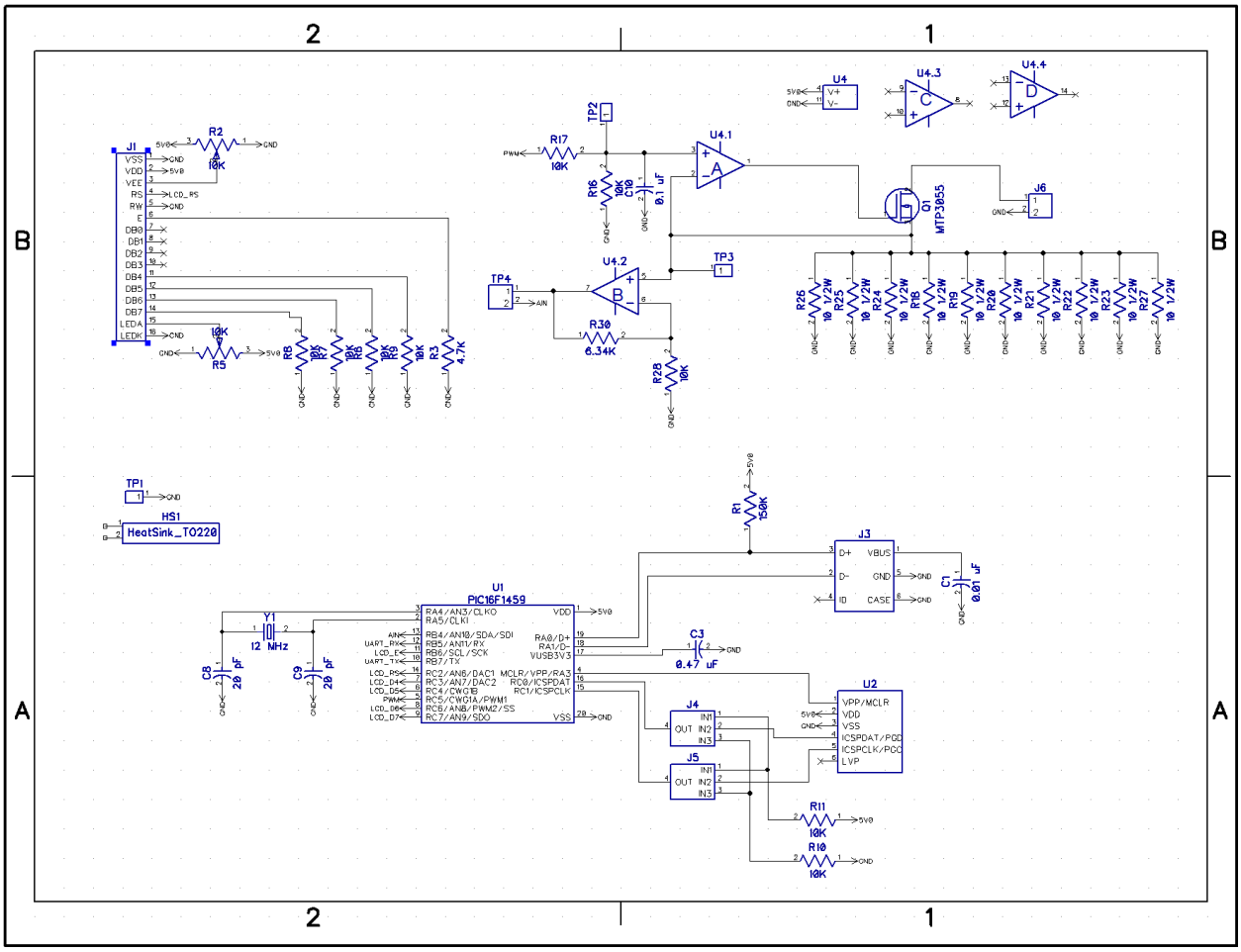

Figure C.1.3: PIC, LCD, and Circuit Loop Circuits for Programmable Load

C.2 Layout Diagrams

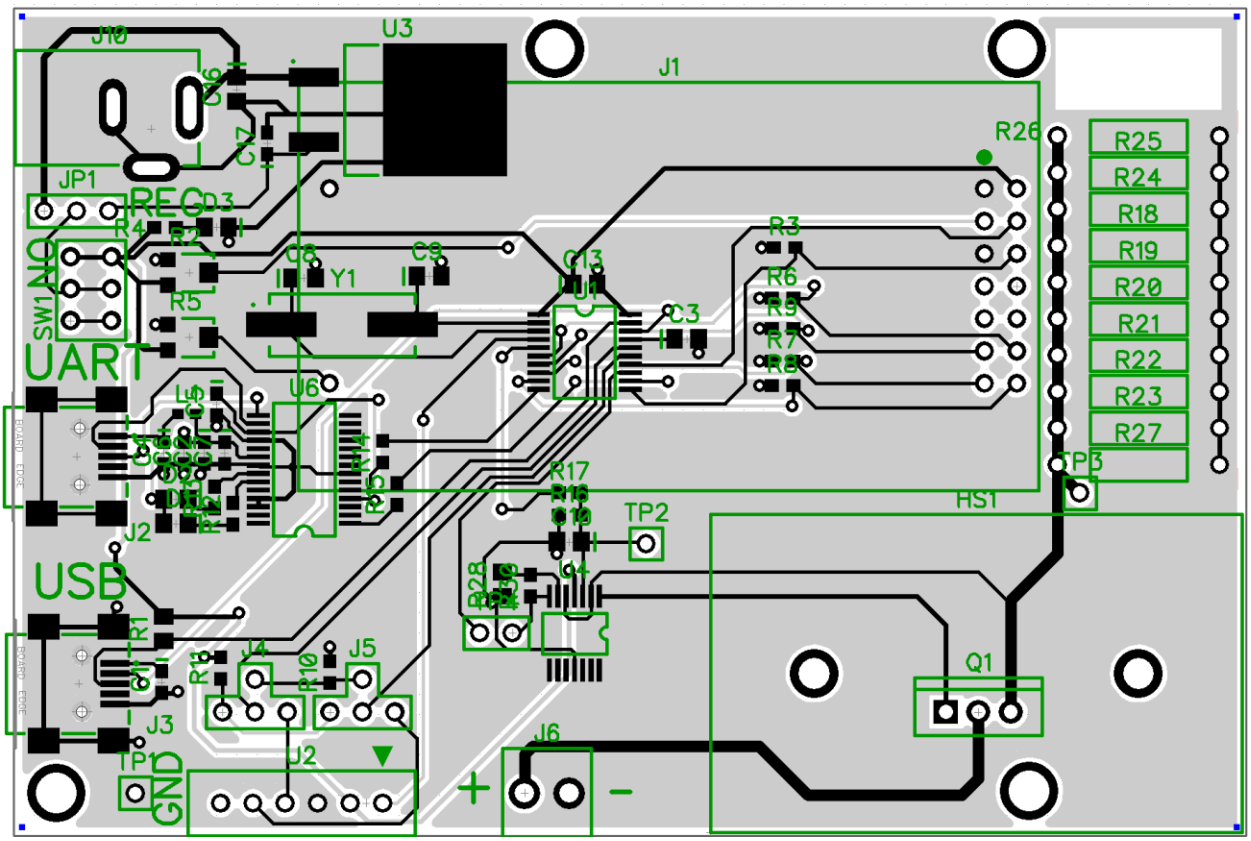

Figure C.2.1: Top Layer Layout for Programmable Load

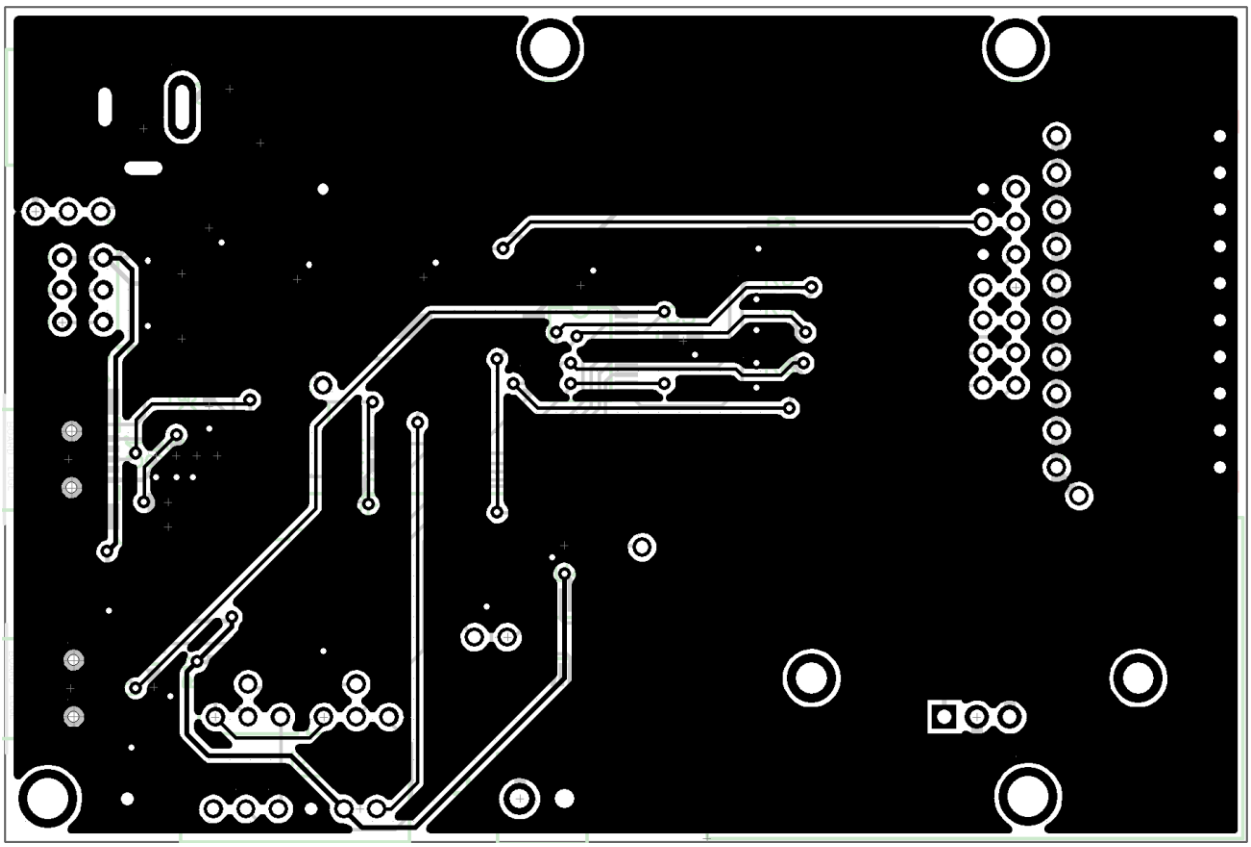

Figure C.2.2: Bottom Layer Layout for Programmable Load

C.3 Bill of Materials

RefDes	Value	Company	Part #	Quantity
C1, C4	0.01 uF	Digikey	445-1311-1-ND	2
C2, C7	4.7 uF	Digikey	445-7482-1-ND	2
C3	0.47 uF	Digikey	587-1282-1-ND	1
C5, C6, C17	0.1 uF	Digikey	445-1316-1-ND	3
C8, C9	20 pF	Digikey	399-8033-1-ND	2
C10	0.1 uF	Digikey	1276-1007-1-ND	1
C13	10 uF	Digikey	445-1371-1-ND	1
C16	0.33 uF	Digikey	445-7462-1-ND	1
D1	BLUE	Digikey	160-1645-1-ND	1
D2	RED	Digikey	475-1415-1-ND	1
D3	GREEN	Digikey	475-1410-1-ND	1
HS1	HeatSink_TO220	Digikey	FA-T220-64E-ND	1
J1	LCDHeader_2x8			1
J2, J3	USB-MINI	Digikey	ED2992CT-ND	2
J4, J5	HEADER_3to1			2
J6				1
J10	PJ-002AH	Digikey	CP-002AH-ND	1
JP1	Jumper	Digikey	609-3470-ND	1
L1	200 mA	Digikey	445-2179-1-ND	1
Q1	MTP3055	Digikey	MTP3055VLFS-ND	1
R1	150K	Digikey	P150KACT-ND	1
R2, R5	10K	Digikey	TC33X-103ECT-ND	2
R3	4.7K	Digikey	RMCF0603JT4K70CT-ND	1
R4	470	Digikey	RMCF0603FT470RCT-ND	1
R6, R7, R8, R9, R10, R11, R16, R17	10K	Digikey	RMCF0603JT10K0CT-ND	8
R12, R13, R14, R15	150	Digikey	P150GCT-ND	4
R18, R19, R20, R21, R22, R23, R24, R25, R26, R27	10 1/2W	Digikey	PPC10.0XCT-ND	10
R28	10K	Digikey	P10.0KCCT-ND	1
R30	6.34K	Digikey	P6.34KHCT-ND	1
SW1	SPDT	Digikey	EG1918-ND	1
TP1, TP2, TP3				3
TP4				1
U1	PIC16F1459-SS	Microchip	PIC16F1459-I/SS	1
U2				1
U3	L7805A	Digikey	497-7255-1-ND	1

U4		**Digikey**	**296-9545-1-ND**	**1**
U6		**Digikey**	**768-1007-1-ND**	**1**
Y1	**12 MHz**	**Digikey**	**XC1289CT-ND**	**1**

C.4 Code

C.4.1 Analog to Digital Driver

```
//libAnalog.h
#ifndef LIBANALOG_H
#defineLIBANALOG_H

#ifdef  __cplusplus
extern "C" {
#endif

   #include "HardwareConfig.h"

   //Function Prototypes
   //Public
   void initADC();
   int  readADC();


   //Private

#ifdef  __cplusplus
}
#endif

#endif  /* LIBANALOG_H */

//libAnalog.c
#include "libAnalog.h"

void initADC()
{
   //configure ADC module
   ADFM  = 1;
   ADCON1bits.ADCS = 6;

   //make pin input
   AIN_DIR = 1;
   ANSELB = 0x10;

   //select channel
   ADCON0bits.CHS = 10;
   ADCON0bits.ADON = 1;

}
```

```
int readADC()
{
   int value = 0;

   __delay_us(10);
   ADCON0bits.ADGO = 1;

   while(ADCON0bits.ADGO == 1);

   value = (ADRESH << 8);
   value += ADRESL;
   return value;
}
```

C.4.2 LCD Driver

```
//libLCD4Bit.h
#ifndef LIBLCD4BIT_H
#defineLIBLCD4BIT_H

#ifdef  __cplusplus
extern "C" {
#endif

   #include "HardwareConfig.h"

   //Definitions
   #define INTERFACE 0b00101100

   //Functions Prototypes
   //PUBLIC
   void initLCD();


   //PRIVATE
   void checkBusy();
   void initWriteCmd(unsigned char cmd);
   void writeCommand(unsigned char cmd);
   void writeChar(unsigned char dat);
   void writeString(unsigned char *dat);
   void LcdWrite(unsigned char data);
   void LcdDir(unsigned char dir);

#ifdef  __cplusplus
}
#endif
```

```
#endif  /* LIBLCD4BIT_H */


//libLCD4Bit.c
#include "libLCD4Bit.h"

void initLCD()
{
   //set pin direction
   E_DIR   = 0;
   RS_DIR  = 0;
   LcdDir(0x00);

   //init the state
   E  = 0;
   RS = 0;
   LcdWrite(0x00);

   //wait for the LCD to initialize
   __delay_ms(15);

   //4 bit initialization
   initWriteCmd(0x03);
   __delay_ms(5);
   initWriteCmd(0x03);
   __delay_ms(5);
   initWriteCmd(0x03);
   __delay_ms(5);
   initWriteCmd(0x02);
   checkBusy();

   //set interface length
   writeCommand(INTERFACE);
   checkBusy();

   //Turn off display and clear
   writeCommand(0x08);
   checkBusy();
   writeCommand(0x01);
   checkBusy();

   //set cursor behavior
   writeCommand(0x06);
   checkBusy();
```

```
    //turn on display and cursor
    writeCommand(0x0E);
    checkBusy();
}

void checkBusy()
{
    __delay_us(1000);
}

void initWriteCmd(unsigned char cmd)
{
    RS = 0;
    __delay_us(1);
    E = 1;
    LcdWrite(cmd);
    __delay_us(1);
    E = 0;
    checkBusy();
}

void writeCommand(unsigned char cmd)
{
    RS = 0;
    __delay_us(1);
    E = 1;
    LcdWrite((cmd >> 4) & 0x0F);
    __delay_us(1);
    E = 0;
    __delay_us(1);
    E = 1;
    LcdWrite(cmd & 0x0F);
    __delay_us(1);
    E = 0;
    checkBusy();
}

void writeChar(unsigned char dat)
{
    RS = 1;
    __delay_us(1);
    E = 1;
    unsigned char temp = (dat >> 4) & 0x0F;

    LcdWrite((dat >> 4) & 0x0F);
    __delay_us(1);
```

```
    E = 0;
    __delay_us(1);
    E = 1;
    temp = dat & 0x0F;
    unsigned char temp2 = temp & 1;
    unsigned char temp3 = (temp >> 3) & 1;
    LcdWrite(dat & 0x0F);
    __delay_us(1);
    E = 0;
    checkBusy();
}

void writeString(unsigned char *dat)
{
    while(*dat != 0)
    {
        writeChar(*dat);
        dat++;
    }
}

void LcdWrite(unsigned char data)
{
    D4 = data & 1;
    D5 = (data>>1) & 1;
    D6 = (data>>2) & 1;
    D7 = (data>>3) & 1;
}

void LcdDir(unsigned char dir)
{
    D4_DIR = dir & 1;
    D5_DIR = (dir>>1) & 1;
    D6_DIR = (dir>>2) & 1;
    D7_DIR = (dir>>3) & 1;
}
```

C.4.3 Uart Driver

```
//libUart.h
#ifndef LIBUART_H
#defineLIBUART_H

#include <string.h>

#ifdef  __cplusplus
```

```
extern "C" {
#endif

   #include <stdbool.h>
   #include "HardwareConfig.h"

   //Function Prototypes
   //Public
   void initUart(long baud, bool invert);
   void uartWrite(unsigned char data);
   void uartWriteString(char *data, int leng);
   unsigned char uartRead();
   void uartRestart(long baud, bool invert);

   //Private


#ifdef  __cplusplus
}
#endif

#endif  /* LIBUART_H */


//libUart.c
#include "libUart.h"

void initUart(long baud, bool invert)
{
   unsigned int baudval;

   switch(baud){
         case 1:
         case 1200:
                  baudval = 9999;
                  break;
         case 2:
         case 4800:
                  baudval = 2499;
                  break;
         case 3:
         case 9600:
                  baudval = 1249; //working
                  break;
         case 4:
         case 19200:
```

```
                baudval = 624;
                break;
        case 5:
        case 38400:
                baudval = 312;
                break;
        case 6:
        case 57600:
                baudval = 207;
                break;
        case 7:
        case 115200:
                baudval = 103;
                break;
      default: baudval = 0;
   }

   //set pin directions
   RX_DIR = 1;
   TX_DIR = 0;

   BRGH  = 1;
   BRG16 = 1;
   SYNC  = 0;
   SPEN  = 1;

   if(invert == true)
      SCKP = 1;

   SPBRGH = (baudval>>8) & 0xFF;
   SPBRGL = baudval & 0xFF;

   //enable interrupts
   RCIE = 1;
   GIE = 1;
   PEIE = 1;


   CREN = 1;
   TXEN = 1;
}

void uartWrite(unsigned char data)
{
   while(!TXIF) continue;
   TXREG = data;
```

```
}

void uartWriteString(char *data, int leng)
{
   while(leng > 0)
   {
      uartWrite(*data);
      data++;
      leng--;
   }
}

unsigned char uartRead()
{
   return RCREG;
}

void uartRestart(long baud, bool invert)
{
   SPEN = 0;
   CREN = 0;
   TXEN = 0;

   initUart(baud,invert);
}
```

C.4.4 PWM Driver

```
//libPWM.h
#ifndef LIBPWM_H
#defineLIBPWM_H

#ifdef  __cplusplus
extern "C" {
#endif

   #include "HardwareConfig.h"

   //Function Prototypes
   //Public
   void initPWM(int speed, int duty);
   void setPWMFrequency(int freq);
   void setPWMDutyCycle(int duty);
   void startPWM();
   void stopPWM();
```

```
   //Private


#ifdef  __cplusplus
}
#endif

#endif  /* LIBPWM_H */


//libPWM.c
#include "libPWM.h"

unsigned char pr;
unsigned char prescale;

void initPWM(int speed, int duty)
{
   long dutyval = 0;

   switch(speed){
         case 0:
         prescale = 64;
         pr = 0xFF;
         break;
         case 1:
         prescale = 4;
         pr = 0xFF;
         break;
   case 2:
         prescale = 1;
         pr = 0xFF;
         break;
         case 3:
         prescale = 1;
         pr = 0x3F;
         break;
         case 4:
         prescale = 1;
         pr = 0x1F;
         break;
         case 5:
         prescale = 1;
         pr = 0x17;
         break;
      default:
```

```
        prescale = 1;
        pr = 0xFF;
        }

    //disable by setting tris
    PWM_DIR = 1;

    //clear the pwm1con
    PWM1CON = 0;

    //load pr2
    PR2 = pr;

    //clear dch dcl
    PWM1DCH = 0;
    PWM1DCL = 0;

    dutyval = (duty*(4*((long)pr+1)))/100;
    PWM1DCL = dutyval & 0x03;
    PWM1DCH = (dutyval >> 2) & 0xFF;

    //configure and start timer 2
    TMR2IF  = 0;
    T2CKPS0 = prescale & 0x01;
    T2CKPS1 = prescale & 0x02;
    TMR2ON  = 1;

    //enable and wait for overflow
    PWM1EN = 1;
    while(!TMR2IF);

    PWM_DIR = 0;
    PWM1OE = 1;

}

void setPWMFrequency(int freq)
{

}

void setPWMDutyCycle(int duty)
{
        long dutyval = 0;
        dutyval = (duty*(4*((long)pr+1)))/100;
    PWM1DCL = dutyval & 0x03;
```

```
   PWM1DCH = (dutyval >> 2) & 0xFF;
}

void startPWM()
{
   PWM1OE = 1;
}

void stopPWM()
{
   PWM1OE = 0;
}
```

C.4.5 Main Application

```
//HardwareConfig.h
#ifndef HARDWARECONFIG_H
#defineHARDWARECONFIG_H

#ifdef  __cplusplus
extern "C" {
#endif

#include <stdio.h>
#include <stdlib.h>
#include <xc.h>

#include "libLCD4Bit.h"
#include "libUart.h"
#include "libPWM.h"
#include "libAnalog.h"

#define    _XTAL_FREQ  48000000

#define    RX      PORTBbits.RB5
#define    RX_DIR  TRISBbits.TRISB5
#define    TX      PORTBbits.RB7
#define    TX_DIR  TRISBbits.TRISB7
#define    PWM     PORTCbits.RC5
#define    PWM_DIR TRISCbits.TRISC5
#define    AIN_CH  10
#define    AIN     PORTBbits.RB4
#define    AIN_DIR TRISBbits.TRISB4
#define    E       PORTBbits.RB6
#define    E_DIR   TRISBbits.TRISB6
#define    RS      PORTCbits.RC2
```

```
#define    RS_DIR  TRISCbits.TRISC2
#define    D4      PORTCbits.RC3
#define    D4_DIR  TRISCbits.TRISC3
#define    D5      PORTCbits.RC4
#define    D5_DIR  TRISCbits.TRISC4
#define    D6      PORTCbits.RC6
#define    D6_DIR  TRISCbits.TRISC6
#define    D7      PORTCbits.RC7
#define    D7_DIR  TRISCbits.TRISC7


#ifdef  __cplusplus
}
#endif

#endif  /* HARDWARECONFIG_H */


//main.c
#include <string.h>
#include "HardwareConfig.h"

//Configuration
// CONFIG1
#pragma config FOSC = HS        // Oscillator Selection Bits (HS Oscillator, High-speed
crystal/resonator connected between OSC1 and OSC2 pins)
#pragma config WDTE = OFF       // Watchdog Timer Enable (WDT disabled)
#pragma config PWRTE = OFF      // Power-up Timer Enable (PWRT disabled)
#pragma config MCLRE = ON       // MCLR Pin Function Select (MCLR/VPP pin function is
MCLR)
#pragma config CP = OFF         // Flash Program Memory Code Protection (Program memory
code protection is disabled)
#pragma config BOREN = ON       // Brown-out Reset Enable (Brown-out Reset enabled)
#pragma config CLKOUTEN = OFF   // Clock Out Enable (CLKOUT function is disabled. I/O
or oscillator function on the CLKOUT pin)
#pragma config IESO = ON        // Internal/External Switchover Mode (Internal/External
Switchover Mode is enabled)
#pragma config FCMEN = ON       // Fail-Safe Clock Monitor Enable (Fail-Safe Clock Monitor
is enabled)

// CONFIG2
#pragma config WRT = OFF        // Flash Memory Self-Write Protection (Write protection off)
#pragma config CPUDIV = NOCLKDIV// CPU System Clock Selection Bit (NO CPU system
divide)
#pragma config USBLSCLK = 48MHz // USB Low SPeed Clock Selection bit (System clock
expects 48 MHz, FS/LS USB CLKENs divide-by is set to 8.)
```

```
#pragma config PLLMULT = 4x     // PLL Multipler Selection Bit (4x Output Frequency
Selected)
#pragma config PLLEN = ENABLED  // PLL Enable Bit (3x or 4x PLL Enabled)
#pragma config STVREN = ON      // Stack Overflow/Underflow Reset Enable (Stack Overflow
or Underflow will cause a Reset)
#pragma config BORV = LO        // Brown-out Reset Voltage Selection (Brown-out Reset
Voltage (Vbor), low trip point selected.)
#pragma config LPBOR = OFF      // Low-Power Brown Out Reset (Low-Power BOR is
disabled)
#pragma config LVP = ON         // Low-Voltage Programming Enable (Low-voltage
programming enabled)

#define NUMITERS 20

void initialize();

int main() {
   initialize();

         //try using notepad
   __delay_ms(2000);
   uartWriteString("start", 5);
   setPWMDutyCycle(4);

   double value;
   double current;
   char buf[20];
   int strleng,i;
   while(1)
   {
      for(i=0; i < NUMITERS; i++)
         value += (double) readADC();
      value /= NUMITERS;
      //value = (double) readADC();
      value = (value / 1024.0)*5.0;
      current = value / 1.634;
      strleng = sprintf(buf, "%f\t", value);
      uartWriteString(buf, strleng);
      strleng = sprintf(buf, "%f\r", current);
      uartWriteString(buf, strleng);
      __delay_ms(200);
   }
   return 0;
}

void initialize()
```

```
{
    //setup pins
    //disable analog pins
    ANSELA = 0x00;
    ANSELB = 0x00;
    ANSELC = 0x00;

    PWM_DIR = 0;

    //Delay 100 ms to let the LCD initialize
    __delay_ms(100);
    initLCD();
    initUart(9600, false);
    initPWM(2, 10);
    initADC();
}
```

APPENDIX D: EDLC MANAGEMENT SYSTEM DESIGN

D.1 Schematics

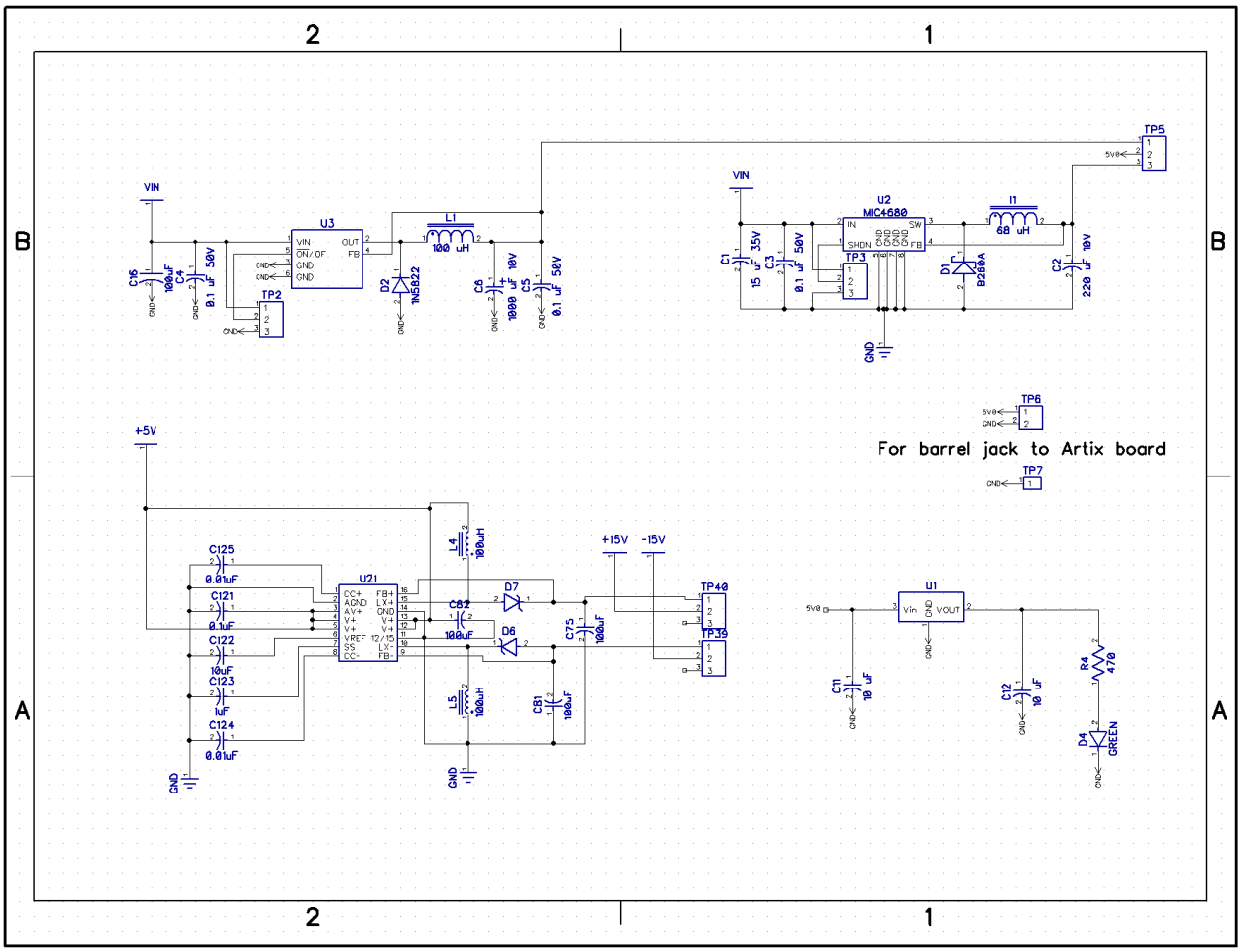

Figure D.1.1: Power Circuit for EDLC Management System

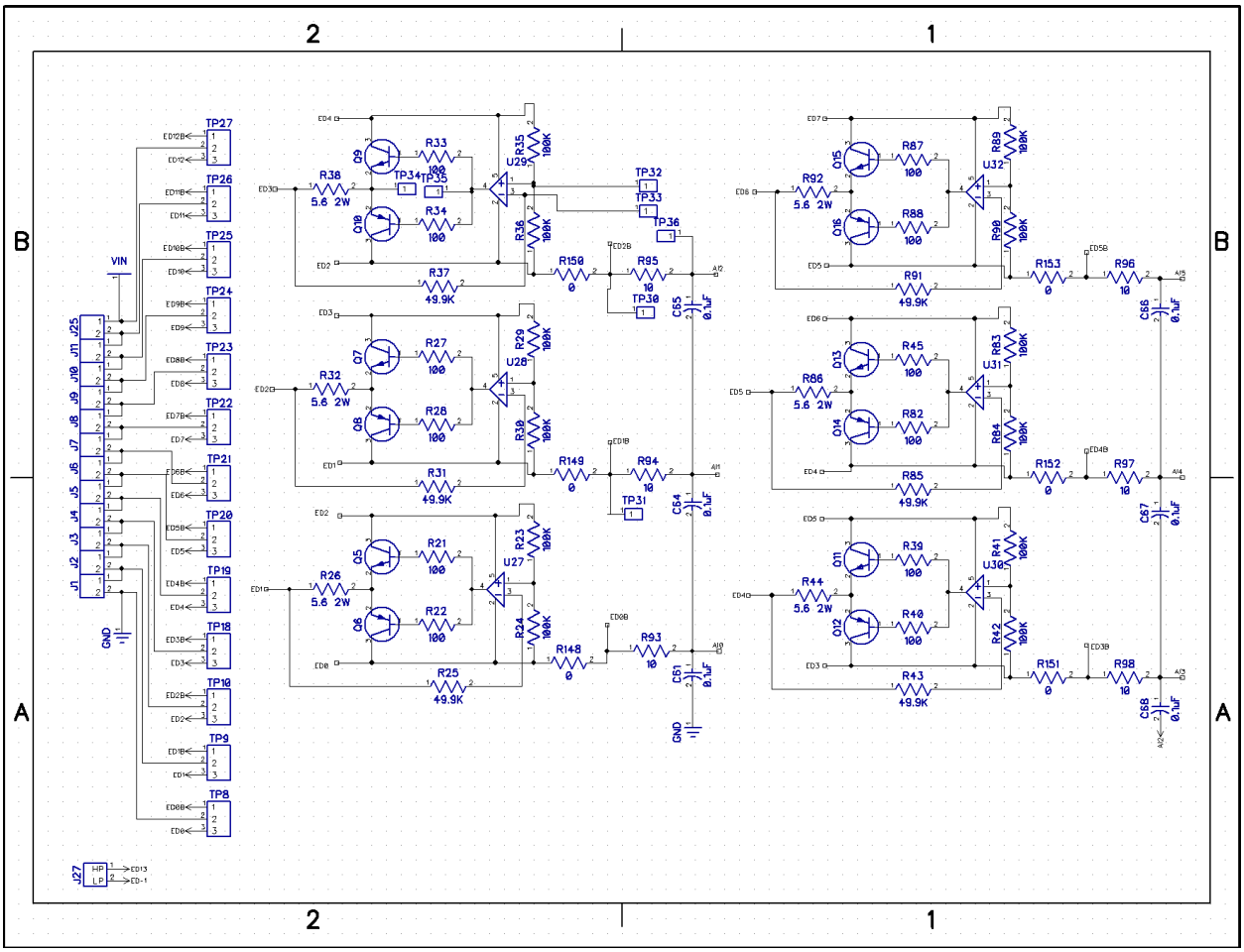

Figure D.1.2: EDLC Interface and Cell Balancing Circuit for EDLC Management System

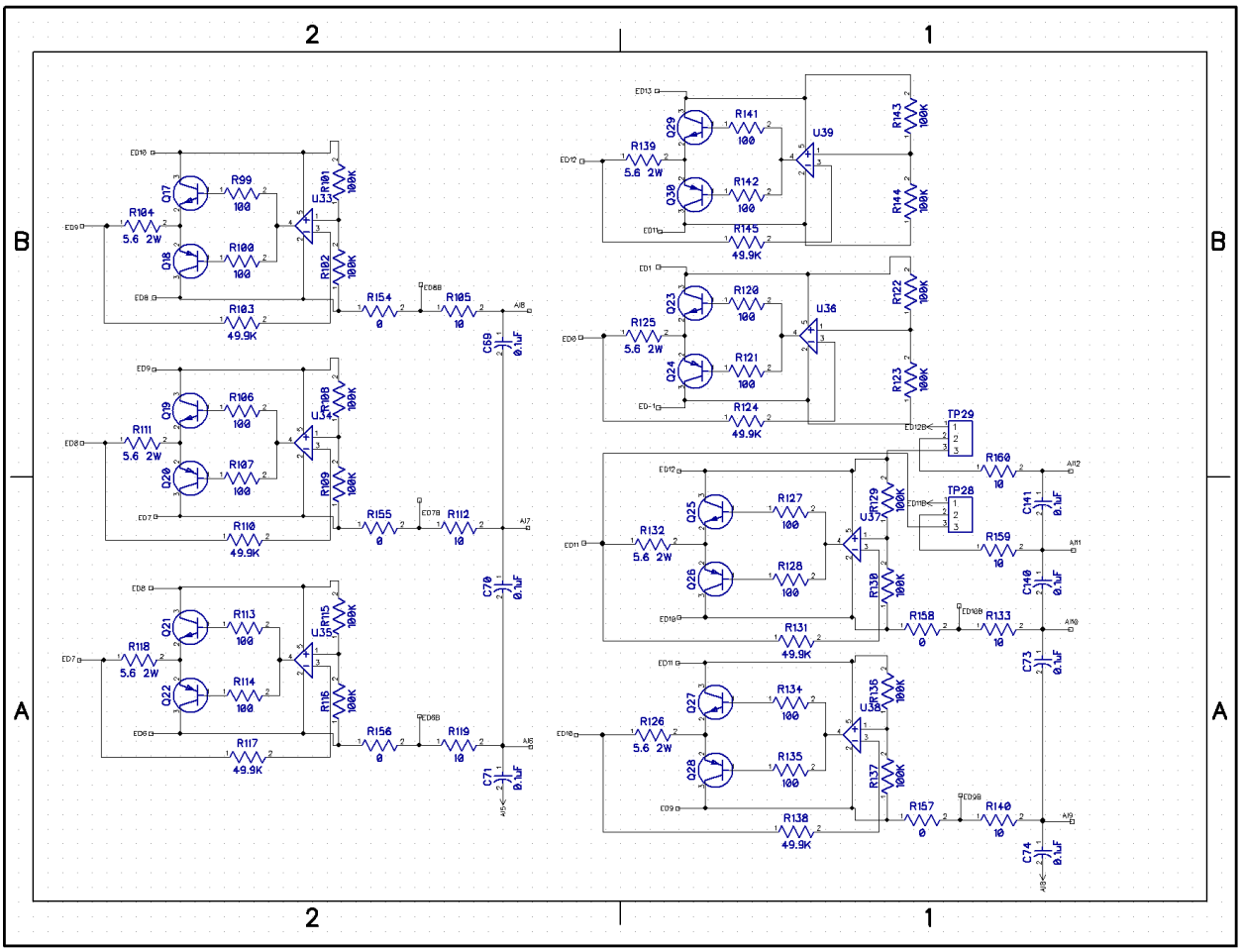

Figure D.1.3: Cell Balancing Circuit Cont. for EDLC Management System

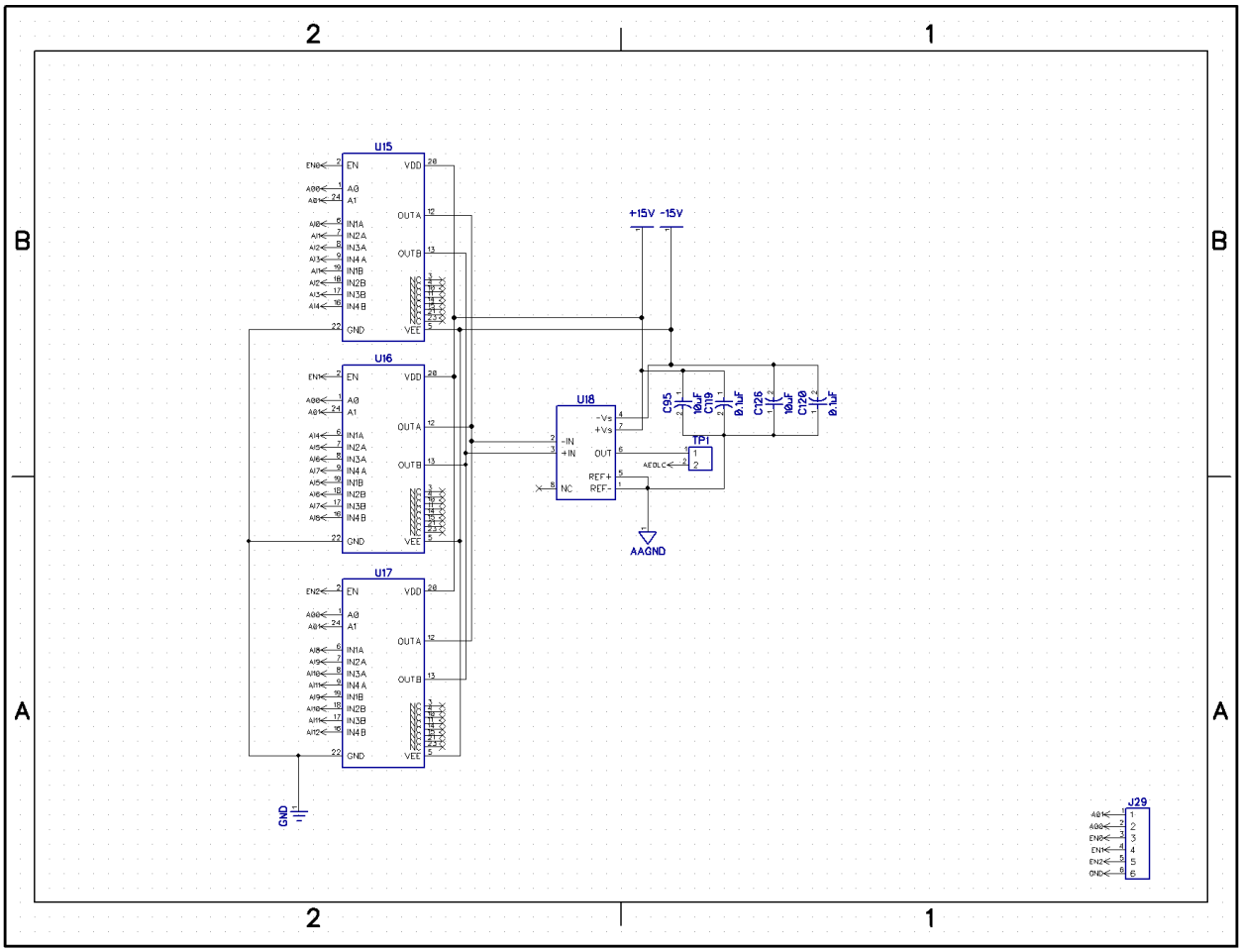

Figure D.1.4: EDLC Multiplexor Circuit for EDLC Management System

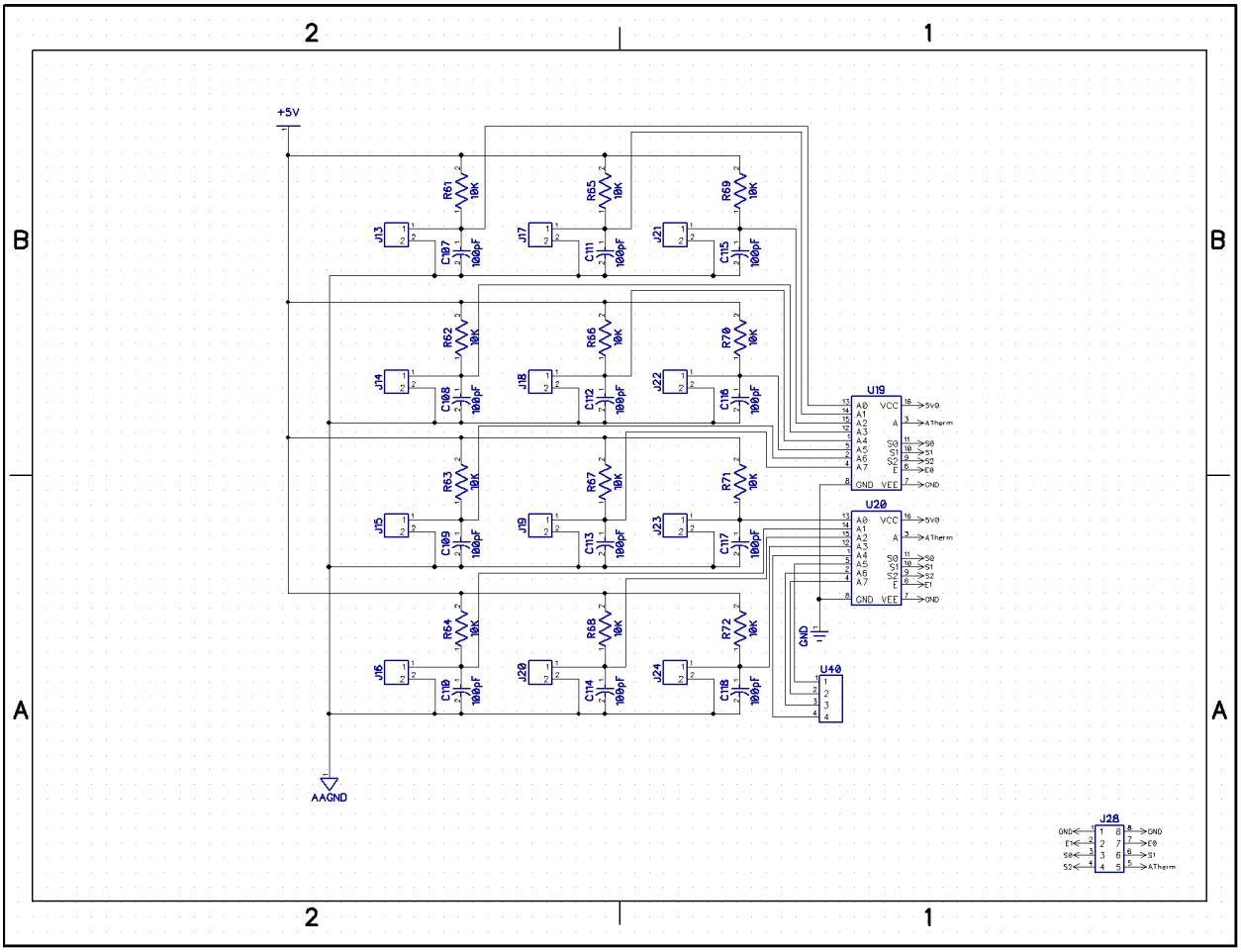

Figure D.1.5: Thermistor Interface for EDLC Management System

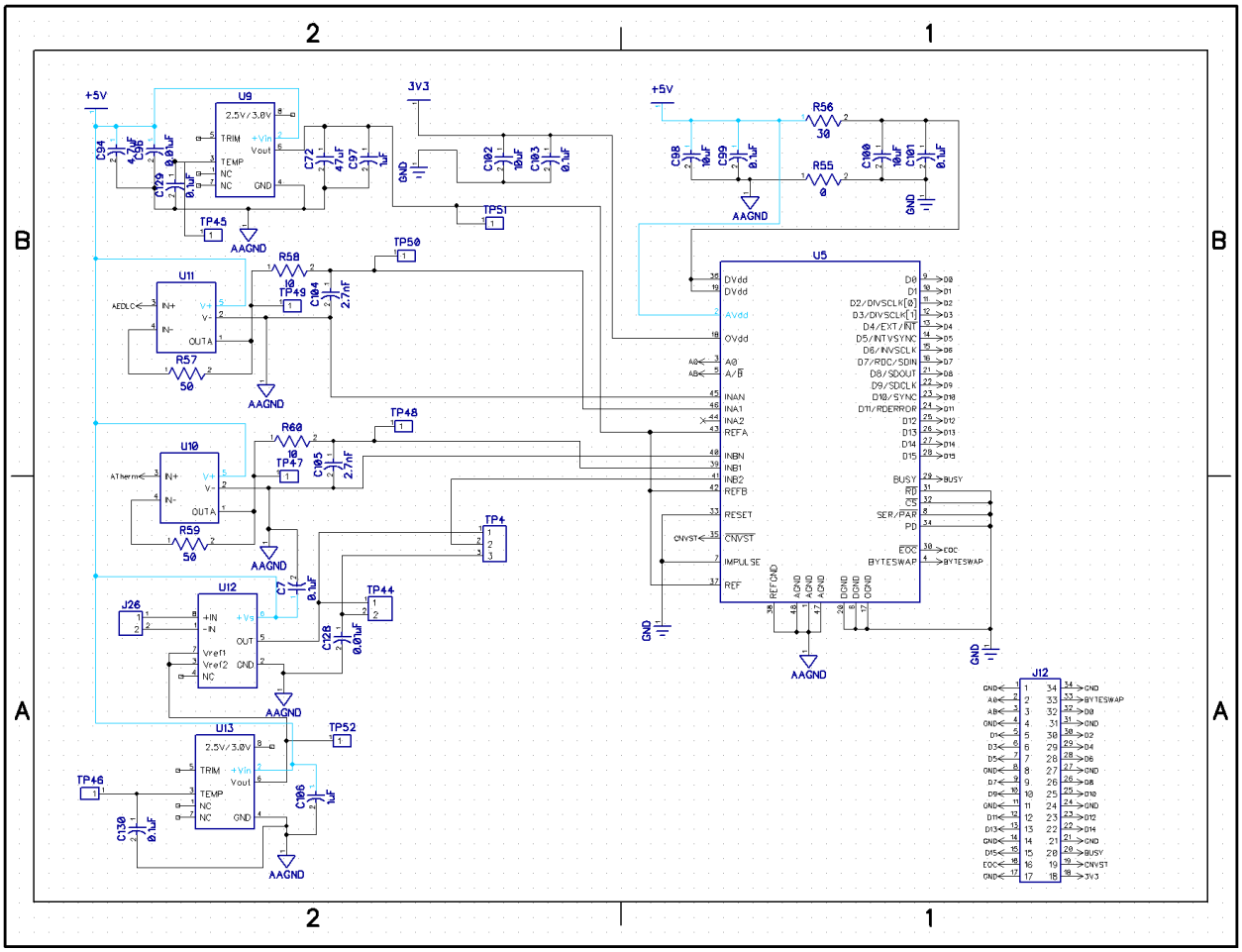

Figure D.1.6: AD7655 A/D Circuit for EDLC Management System

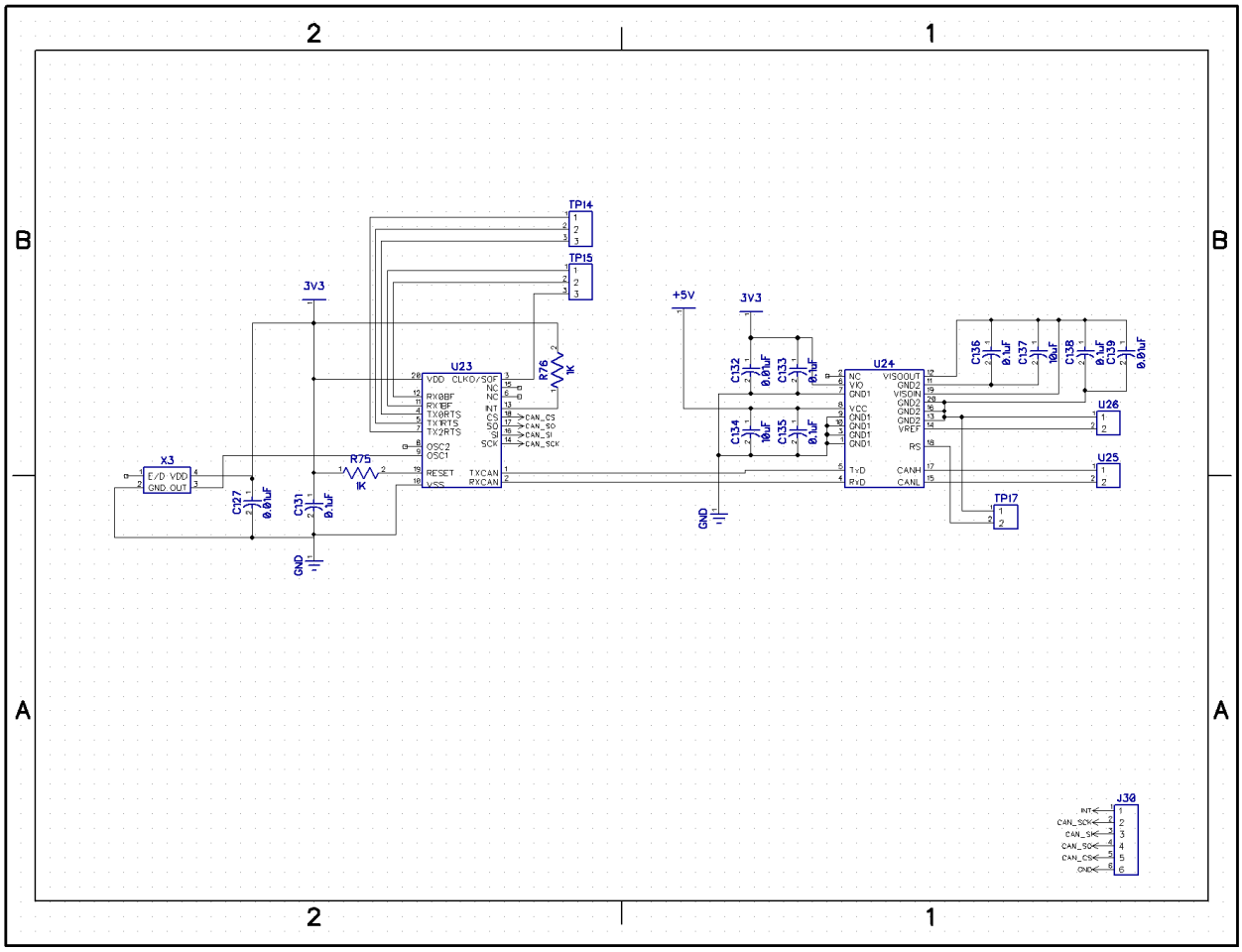

Figure D.1.7: CAN Circuit for EDLC Management System

D.2 Layout Diagrams

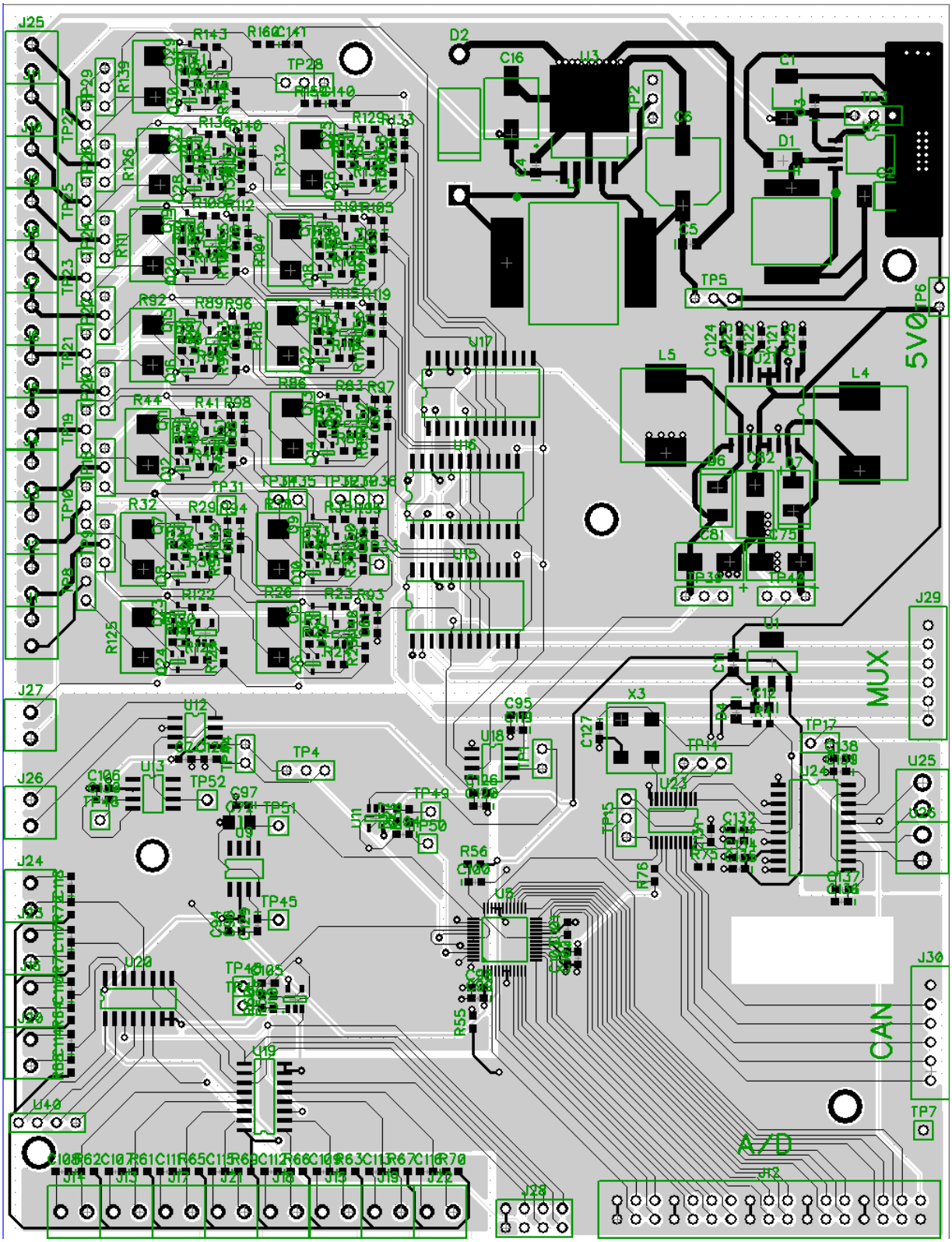

Figure D.2.1: Top Layer Layout for EDLC Management System

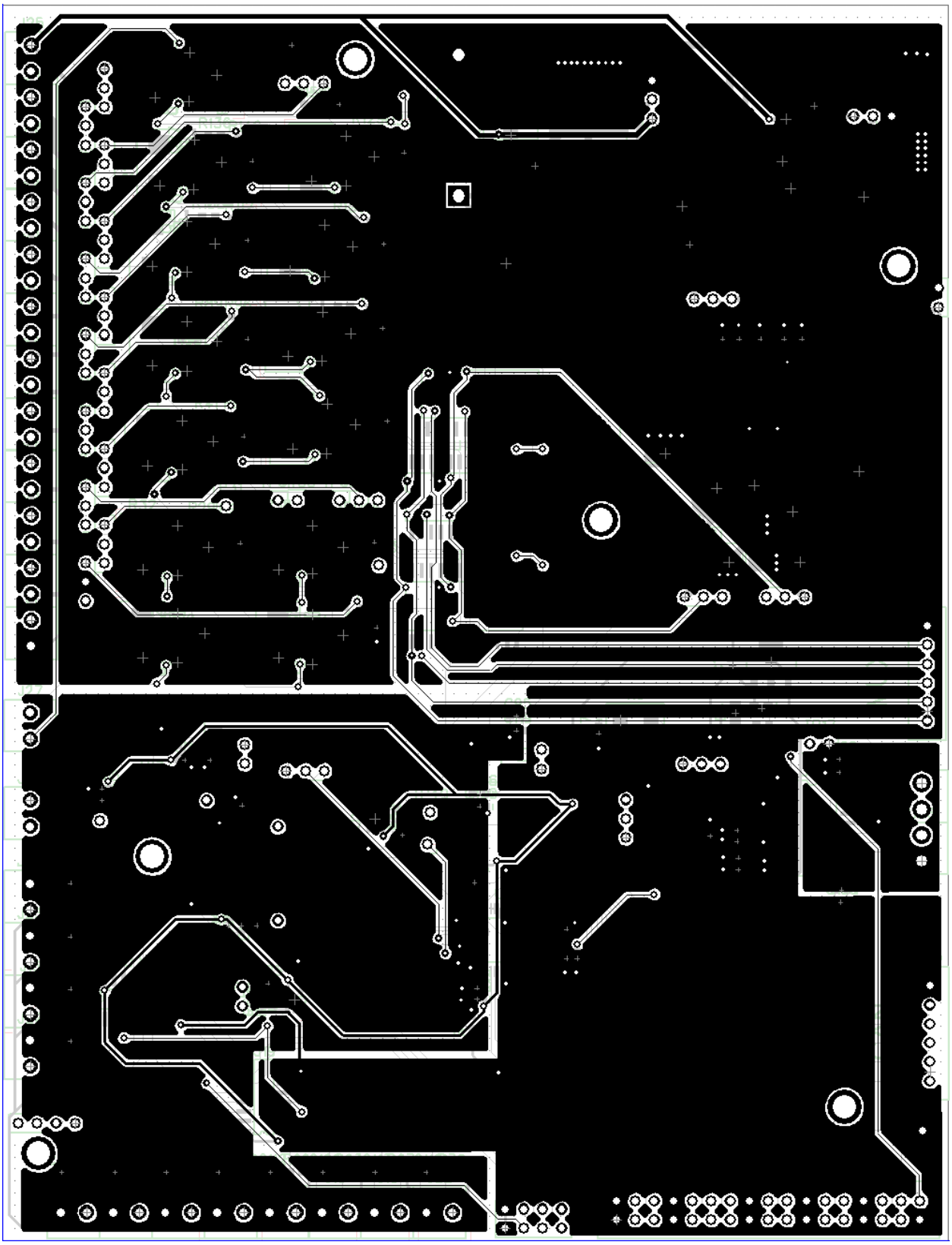

Figure D.2.2: Bottom Layer Layout for EDLC Management System

D.3 Bill of Materials

RefDes	Value	Company	Part #	Quantity
C1	15 uF 35V	Digikey	478-4101-1-ND	1
C2	220 uF 10V	Digikey	478-4110-1-ND	1
C3, C4, C5	0.1 uF 50V	Digikey	1276-1007-1-ND	3
C6	1000 uF 10V	Digikey	PCE4567CT-ND	1
C7, C61, C64, C65, C66, C67, C68, C69, C70, C71, C73, C74, C99, C101, C103, C119, C120, C121, C129, C130, C131, C133, C135, C136, C138, C140, C141	0.1uF	Digikey	399-1096-1-ND	27
C16	100uF	Digikey	PCE3917CT-ND	1
C72	47uF	Digikey	490-5528-1-ND	1
C75, C81, C82	100uF	Digikey	399-5175-1-ND	3
C94	4.7uF	Digikey	490-3302-1-ND	1
C95, C98, C100, C102, C122, C126, C134, C137	10uF	Digikey	445-4112-1-ND	8
C96, C124, C125, C127, C128, C132, C139	0.01uF	Digikey	311-1572-1-ND	7
C97, C106, C123	1uF	Digikey	1276-1041-1-ND	3
C104, C105	2.7nF	Digikey	399-7890-1-ND	2
C107, C108, C109, C110, C111, C112, C113, C114, C115, C116, C117, C118	100pF	Digikey	1276-1008-1-ND	12
D1	B260A	Digikey	B260A-FDICT-ND	1
D2	1N5822	Digikey	1N5822-TPMSCT-ND	1
D6, D7		Digikey	641-1014-1-ND	2
I1	68 uH	Digikey	513-1083-1-ND	1
J1, J2, J3, J4, J5, J6, J7, J8, J9, J10, J11, J13, J14, J15, J16, J17, J18, J19, J20, J21, J22, J23, J24, J25, J26, J27		Digikey	A98036-ND	26
J12	Header_2x17	Digikey	S7120-ND	1
J28	Header_2x4	Digikey	A26454-ND	1
J29, J30	HEADER_6PIN	Digikey	609-3256-ND	2
L1	100 uH	Digikey	513-1093-1-ND	1
L4, L5	100uH	Digikey	513-1526-1-ND	2
Q5, Q7, Q9, Q11, Q13,		Digikey	MMBT2222AWT1GOS	13

Q15, Q17, Q19, Q21, Q23, Q25, Q27, Q29			**CT-ND**	
Q6, Q8, Q10, Q12, Q14, Q16, Q18, Q20, Q22, Q24, Q26, Q28, Q30		**Digikey**	**MMBT2907AWT1GOSCT-ND**	**13**
R21, R22, R27, R28, R33, R34, R39, R40, R45, R82, R87, R88, R99, R100, R106, R107, R113, R114, R120, R121, R127, R128, R134, R135, R141, R142	**100**	**Digikey**	**CR0603-FX-1000ELFCT-ND**	**26**
R23, R24, R29, R30, R35, R36, R41, R42, R83, R84, R89, R90, R101, R102, R108, R109, R115, R116, R122, R123, R129, R130, R136, R137, R143, R144	**100K**	**Digikey**	**A102199CT-ND**	**26**
R25, R31, R37, R43, R85, R91, R103, R110, R117, R124, R131, R138, R145	**49.9K**	**Digikey**	**RMCF0603FT49K9CT-ND**	**13**
R26, R32, R38, R44, R86, R92, R104, R111, R118, R125, R126, R132, R139	**5.6 2W**	**Digikey**	**A103600CT-ND**	**13**
R55, R148, R149, R150, R151, R152, R153, R154, R155, R156, R157, R158	**0**	**Digikey**	**P0.0GCT-ND**	**12**
R56	**30**	**Digikey**	**P30.0HCT-ND**	**1**
R57, R59	**50**	**Digikey**	**FC0603-50BFCT-ND**	**2**
R58, R60, R93, R94, R95, R96, R97, R98, R105, R112, R119, R133, R140, R159, R160	**10**	**Digikey**	**RMCF0603JT10R0CT-ND**	**15**
R61, R62, R63, R64, R65, R66, R67, R68, R69, R70, R71, R72	**10K**	**Digikey**	**P10.0KHCT-ND**	**12**
R75, R76	**1K**	**Digikey**	**P1.0KGCT-ND**	**2**
TP1, TP2, TP3, TP6, TP17, TP44				**6**
TP4, TP5, TP8, TP9, TP10, TP14, TP15, TP18, TP19, TP20, TP21, TP22, TP23, TP24, TP25, TP26, TP27, TP28, TP29, TP39, TP40				**21**
TP7, TP30, TP31, TP32, TP33, TP34, TP35, TP36,				**16**

TP45, TP46, TP47, TP48, TP49, TP50, TP51, TP52				
U2	**MIC4680**	**Digikey**	**576-1221-1-ND**	**1**
U3	**LM2576D2T-5**	**Digikey**	**LM2576SX-5.0/NOPBCT-ND**	**1**
U5		**Digikey**	**AD7655ASTZ-ND**	**1**
U9, U13		**Digikey**	**AD780BRZ-ND**	**2**
U10, U11		**Digikey**	**AD8605ARTZREEL7CT-ND**	**2**
U12		**Digikey**	**AD8210YRZ-ND**	**1**
U15, U16, U17		**Digikey**	**MAX379CWG+-ND**	**3**
U18		**Digikey**	**AD629BRZ-ND**	**1**
U19, U20		**Digikey**	**296-9276-5-ND**	**2**
U21		**Digikey**	**MAX743EWE+-ND**	**1**
U23		**Digikey**	**MCP2515-I/ST-ND**	**1**
U24	**ADM3053**	**Digikey**	**ADM3053BRWZ-ND**	**1**
U25, U26	**A98036-ND**	**Digikey**	**A98036-ND**	**2**
U27, U28, U29, U30, U31, U32, U33, U34, U35, U36, U37, U38, U39		**Digikey**	**296-10500-1-ND**	**13**
U40				**1**
X3	**20MHz**	**Digikey**	**631-1054-1-ND**	**1**

D.4 Code

```
#include <stdio.h>
#include <stdint.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <unistd.h>
#include "system.h"
#include "altera_avalon_pio_regs.h"
#include "altera_avalon_spi_regs.h"

#define		THERMCH		1
#define		EDLCCH		0
#define		ICH		2
#define		REFCH		3
#define		CONVERSION	3.3/4096


uint16_t	dtherm[16]	= {0};
uint16_t	dedlc[16]	= {0};
uint16_t	dcurr		= 0;
uint32_t	status		= 0;
uint32_t	ctrlvolt	= 0;

char		CLS[]		= "\033[2J";

inline void	writeRawAdc(int channel);
inline uint32_t readRawAdc();

int			calcCheckSum(char *buf, int datalen);
void		updateDisplay();
uint32_t	readAdc(int channel);
int		switchAmux(int channel);
int		switchTmux(int channel);
void		readAllMeasurements(uint16_t *therm, uint16_t *edlc, uint16_t *iedlc);

int main()
{
  printf("ADC Read Test\n");
  printf("=============\n\n");

  //open the uart port
  int fd = open("/dev/uart", O_RDWR | O_NONBLOCK);

  char buf[100] = {0};
  ssize_t bytes;
```

```
  int checksum = 0;
  while(1)
  {
          checksum = 0;
          readAllMeasurements(&dtherm[0], &dedlc[0], &dcurr);
          ctrlvolt = readAdc(REFCH);
          //updateDisplay();
          sprintf(&buf[0],"%%%i,%i,%i,%i,%i,%i,%i,%i,%i,%i,%i,%i",

dedlc[0],dedlc[1],dedlc[2],dedlc[3],dedlc[4],dedlc[5],dedlc[6],dedlc[7],dedlc[8], \
                          dedlc[9],dedlc[10],dedlc[11]);
          checksum = calcCheckSum(buf, strlen(buf));
          write(fd, &buf, strlen(buf));
          usleep(50000);
          sprintf(&buf[0], ",%i,%i,%i,%i,%i,%i,%i,%i,%i,%i,%i,%i", \
                          dtherm[0], dtherm[1], dtherm[2], dtherm[3], dtherm[4],dtherm[5],
dtherm[6], dtherm[7], \
                          dtherm[8], dtherm[9], dtherm[10], dtherm[11]);
          checksum ^= calcCheckSum(buf, strlen(buf));
          write(fd,&buf[0], strlen(buf));
          usleep(50000);
          sprintf(&buf[0], ",%i,%i*", dcurr, ctrlvolt);
          checksum ^= calcCheckSum(buf, strlen(buf));
          sprintf(&buf[strlen(buf)], "%02X\n\r", checksum);
          bytes = write(fd, &buf, strlen(buf));
          //printf("Bytes written = %i\n", bytes);
          usleep(50000);
  }

  close(fd);
  return 0;
}

int calcCheckSum(char *buf, int datalen)
{
          unsigned char temp = 0;
          while(datalen > 0)
          {
                  temp ^= *buf;
                  buf++;
                  datalen--;
          }
          return temp;
}

void updateDisplay()
```

```
{
	int i;
	printf(CLS);
	printf("Reference Voltage = %lu\t%f\n\n", ctrlvolt, ((double)ctrlvolt)*CONVERSION);
	printf("EDLC #\tTherm\t\tVolt\n");
	for(i=0; i<12; i++)
	{
		printf("%i\t%i\t%f\t%i\t%f\n", i, dtherm[i],
((double)(dtherm[i]))*CONVERSION, dedlc[i], ((double)dedlc[i])*CONVERSION);
	}
	printf("Current = %i\t%f\n", dcurr, ((double)dcurr)*CONVERSION);
	printf("Status  = %lu\n", status);
}

void writeRawAdc(int channel)
{
	uint32_t control = 0;
	control = (channel << 11);
	IOWR(ADC_SPI_BASE, ALTERA_AVALON_SPI_TXDATA_REG, control);
	while(!(IORD(ADC_SPI_BASE, ALTERA_AVALON_SPI_STATUS_REG) & 0x20));
}

uint32_t readRawAdc()
{
	return IORD(ADC_SPI_BASE, ALTERA_AVALON_SPI_RXDATA_REG);
}

uint32_t readAdc(int channel)
{
	uint32_t control = 0;
	control = (channel << 11);

	IOWR(ADC_SPI_BASE, ALTERA_AVALON_SPI_TXDATA_REG, control);
	while(!(IORD(ADC_SPI_BASE, ALTERA_AVALON_SPI_STATUS_REG) & 0x20));

	IOWR(ADC_SPI_BASE, ALTERA_AVALON_SPI_TXDATA_REG, control);
	while(!(IORD(ADC_SPI_BASE, ALTERA_AVALON_SPI_STATUS_REG) & 0x20));

	return IORD(ADC_SPI_BASE, ALTERA_AVALON_SPI_RXDATA_REG);
}

int switchAmux(int channel)
{
	uint32_t pinmux = 0;

	if(channel > 11)
```

```
		return -1;

	uint8_t ch = channel / 4;
	uint8_t ao = channel % 4;

	pinmux = (1 << (ch + 2));
	pinmux += ao;

	IOWR_ALTERA_AVALON_PIO_DATA(AMUX_BASE, pinmux);
	usleep(1000);
	return pinmux;
}

int switchTmux(int channel)
{
	uint32_t pinmux = 0;

	if(channel > 15)
		return -1;

	if(channel > 7)
	{
		pinmux = 0x01;
		channel -= 8;
	}
	else
		pinmux = 0x02;

	pinmux = pinmux ^ (channel << 2);
	IOWR_ALTERA_AVALON_PIO_DATA(TMUX_BASE, pinmux);
	usleep(1000);
	return pinmux;
}

void readAllMeasurements(uint16_t *therm, uint16_t *edlc, uint16_t *iedlc)
{
	//this is a algorithm to minimize the number of dummy writes and reads
	int i = 0;

	//set the muxes
	switchTmux(i);
	switchAmux(i);


	//write channel 0
	writeRawAdc(THERMCH);
```

```
        i++;
        readRawAdc();
        writeRawAdc(EDLCCH);
        *therm = readRawAdc();
        therm++;

        while(i < 12)
        {
                switchTmux(i);
                writeRawAdc(THERMCH);
                *edlc = readRawAdc();
                switchAmux(i);
                writeRawAdc(EDLCCH);
                *therm = readRawAdc();
                edlc++;
                therm++;
                i++;
        }

        writeRawAdc(ICH);
        *edlc = readRawAdc();
        writeRawAdc(0);
        *iedlc = readRawAdc();
}
```