NAU NORTHERN ARIZONA UNIVERSITY
School of Informatics, Computing, and Cyber Systems

Team Lithium Lumberjacks

Capstone Final Report

April 29, 2022

Project Sponsor: John Lehman, MSEE
Project Faculty Mentors: Robert Severinghaus, Ph. D and Mahsa Keshavarz, MSEE
Team members: Hamad Aldossary, Hunter Browning, Sean Conlin, Darby DeGan

Overview: The purpose of this document is to provide an overview of our capstone project

Table of Contents

Introduction **3**

Design Process **4**

Functional Decomposition 4

Research 5

Prototyping 7

User Interface 7

Cell Balancing and Charging Algorithms 8

Cell Discharge Hardware 11

Charging Hardware 11

Final Design **12**

Results **15**

Important Test Results 16

Analysis of Results 19

Conclusion **19**

Important Requirements 19

Lessons Learned 21

Acknowledgements 21

User Manual **22**

Introduction 22

Installation 24

Configuration and Use 25

LCD Screen 25

Interface Buttons 25

Fault Conditions 26

Maintenance 26

Troubleshooting Operations 26

Code Not Loading 26

No Power to a Subsystem 27

Conclusion 27

Appendix A **28**

Appendix B **29**

Appendix C **31**

Appendix D **32**

Appendix E **34**

Appendix F **36**

Appendix G **41**

Introduction

This project is sponsored by Dataforth Corporation, a company that provides signal conditioning, data acquisition, and data communication hazard protection solutions to the ever-enlarging factory automation markets [1]. Our liaison for Dataforth was John Lehman, the Vice President of Product Development. Throughout the year our team met with Mr. Lehman weekly to discuss progress and ideas about the product we were developing for Dataforth. The primary objective of this project is to provide Dataforth with a product that is able to utilize and showcase the data acquisition capabilities of the company's MAQ20 line of products. In seeking to achieve this, a battery management system has been proposed.

The Dataforth Corporation has numerous products that can assist in monitoring the necessary data. This data includes cell voltage levels, cell current levels, and battery cell temperature readings. For this project in particular, the charger must be able to safely and efficiently charge and monitor a four-cell lithium-ion polymer (LiPo) battery. In addition, the charger must also be able to check and account for fault conditions such as a defective cell or an incorrect connection to one of the battery cells. For the purpose of accessibility, the battery charger will implement an intuitive user interface; one that utilizes both simple controls and an informative display that allows the user to effectively use the device with relative ease. The finished product is expected to be able to outperform similar battery chargers in one or more areas of cost, efficiency, or size. In addition, the project will have a clearly documented path towards applying the concept of this system towards scaled applications such as battery charging for electric vehicles or energy storage systems in homes and other buildings. The battery charger must use the Texas Instruments (TI) MSP430 microcontroller as the primary controlling circuit of the device. The charging, discharging, cell balancing, and fault correction algorithms will all be stored in the MSP430 microcontroller. In addition, the battery charging device must utilize Dataforth products when applicable.

While the premise of this project is well traversed, it provides goals that make it a unique and relevant task for undergraduate students. This is in addition to giving students the opportunity to work with a major player in industrial electronics. Having Dataforth as a client in this project provides access to expertise from numerous experts such as the vice president for product development, John Lehman. The Dataforth Corporation also gives the students involved in this project opportunity to work with state of the art data acquisition equipment, such as the MAQ20 data acquisition modules. Finally, working with Dataforth in a trade setting will give students experience in presenting a finished product to colleagues and experts in the field, as

the team will have to provide all of the periphery needed to supplement the project, including a user manual for proper operation of the battery charger, a presentation outlining the functions of the charger and the associated Dataforth products, and a display suitable for large scale presentations during trade shows.

Design Process

This section will go into detail about the team's design process over the full year of our capstone project. The process started with researching battery charger and battery management system documentation to learn as much as we could about how to correctly and safely develop our own battery management system (BMS). Once the team had gathered enough information the team began to form a design idea of how we would create our system. Once a design idea had been decided on the team began prototyping each of the subsystems that were in the design. After prototyping the team used the prototype findings to begin to design the final products' subsystems. The last part of the design process was documenting the final product. Each of these steps will be explained in this section.

Functional Decomposition

Using the requirements that were developed by the team in conjunction with the team sponsor, a more detailed description of the operation of the charger was created. Upon device startup and connection of the battery, the battery management system is meant to begin charging at a default changing rate of 1C. While the battery charges, the system needs to monitor multiple parameters of each of the cells via the data acquisition module (MAQ20) from Dataforth, including cell voltage, cell temperature, and overall charging current. In the event that the data acquisition module records an unacceptably overcharged, undercharged, or overheated cell, the system processor immediately disconnects the charging circuit from the battery to protect both the cells and the battery management system. The data sent from the data acquisition module is also displayed to the user via the LCD screen. As the battery charges, the user can select which data to view, the status of the charging, and select one of two charging rates: the default 1C charge rate and a 3C fast charge rate. Once the cells are charged, the charging circuit is deactivated and any cells overcharged are discharged to the nominal voltage value. Once all the cells are at nominal voltage, the battery is considered chemically balanced and is disconnected from the system. The user is also notified of a completed charge via the LCD screen.

Research

In order to establish a thorough understanding of the intricacies of designing a battery management and cycling system, extensive research was done. This research comes in the form of a literature review. For this review, members of the team searched for a variety of documents related to battery charging. Some examples include reports on previous projects, sets of standards established by reputable organizations such as the Institute for Electrical and Electronics Engineers (IEEE) and datasheets describing the specifications for batteries, voltage relays, and other components that will be utilized in the project.

The first document for review concerns a project report for a novel universal battery charger. This charger is capable of charging batteries of many different chemistries, including nickel cadmium (NiCd), nickel metal hydride (NiMH), lithium ion and LiPo batteries [2]. This charger is able to achieve this through the combined use of a current control loop and a voltage control loop. The current control loop works by utilizing a difference amplifier to drive the gate of a power PMOS transistor, then drives the current through a resistor and then the battery [2]. Meanwhile, the voltage control loop is meant to keep the current flow at the maximum charge rate until the voltage of the battery reaches its maximum meaning it has been fully charged. The voltage control loop does this by acting as a feedback loop, allowing for high gain to limit the voltage, and a small unity gain bandwidth window to operate in conjunction with the current control loop [2]. Upon a battery cell(s) reaching its nominal value, the current is then reduced exponentially until it reaches a slow trickle charge rate of 5.5 mA [2]. A high precision analog to digital converter (ADC) is used to detect when the cell has reached its nominal voltage. This work is relevant to the present project because there have been discussions about making this Dataforth battery charger system compatible with multiple battery chemistries as well. At minimum the current and voltage control loop could serve as an adequate model for the battery charger to test for safe and efficient charging capability.

The second work involves a proposal for a single-phase onboard battery charger for electric vehicles (EVs). This charger in particular is meant to have three modes and be capable of charging both a high voltage battery system and a low voltage battery system. The first mode is meant to charge the high-voltage battery of the EV when connected to a charging station or to a power grid. The second mode is meant to discharge the high-voltage battery in the event of excess charge being present. Finally, the third mode is meant to charge the low-voltage battery of the EV through transferring power from the high voltage battery [3]. The overall circuit for this onboard battery charger can be broken down into three separate components. The first

stage involves a full bridge AC-DC converter. This stage is used to convert the electrical energy into a usable form for both battery systems. The second stage, consisting of a dual active bridge DC-DC converter, which is meant to help filter out any excess noise in the circuit. Finally, a dual functional circuit is added to serve as both the charging circuit for the low-voltage battery as well as an active power decoupler. The active power decoupler portion of the circuit uses two capacitors in series to help alleviate issues such as power ripple [3].

The third work that was reviewed was focused on a fast charging technique for batteries. This paper was looking into lithium ion batteries, but the concepts of a fast charging algorithm could assist our team if we decided to allow fast charging in our battery charger. Batteries can be charged in different ways. The standard charging techniques for lithium-ion batteries are trickle charge (TC), constant current (CC), and constant voltage (CV) [4]. These charging methods can also be used on Lipo batteries which helps connect this paper to the project the our team is working on. Each of these charging methods is utilized in different ways when charging a battery. Trickle charging is charging at the same rate as the battery discharging rate allowing for the battery to stay at the charge it was at when trickle charging was started. Constant current and constant voltage charging methods are used at different times when charging a battery and charge at different rates, with constant current being the faster of the two modes [4]. The team from this paper wanted to use the batteries internal resistances and voltages to make the battery believe it needed to be using constant current charging for longer periods of time than it should [4]. This would allow the battery to charge faster by being in the fastest charging mode longer than any other mode. After simulating and testing the team found that “the extension of constant current mode makes the charger charge the battery with faster speed” [4]. Seeing this successful fast charging technique can help our team if we decide to utilize fast charging in our design. This review also helped show how algorithms are thought out and created, useful for our team as we will need to create normal charging algorithms for our battery charger.

Of all the previous documentation the team reviewed, any documents outlining the standards for lithium ion batteries and their charging units were by far the most important. This is because the standards documents provided the team with a set of guidelines that would define a safe and efficient battery charger. One such document provided a detailed description of charging standards for lithium based batteries. In this document, there was a decent amount of terminology that helped the team define how the system architecture would work. Essentially, the standards document described the module that would be responsible for providing power to the battery as a power conversion system (PCS) [5]. This PCS would be responsible for providing DC power to the battery, thus acting as a charger, a rectifier, and an inverter.

Additionally, the document went on to describe a BMS. The BMS serves two major functions in monitoring the status of the battery, including voltage levels, current levels, and temperature. The system would then be able to take action if it notices a fault in these qualities for any one of the cells. For example, if one of the cells were to lose voltage completely, the BMS would recognize this and deactivate the charging relay of that cell. For this project, the Dataforth data acquisition modules will act in conjunction with the MSP430 microcontroller as the primary control for the BMS.

Prototyping

This section will go into detail about the process of prototyping the different subsystems of our team's design. Prototyping is a necessary step in the design process. It allows for products to be tested for any major design flaws without spending as much money or time as would happen during the final design. Prototyping will allow the team to test specific parts of our design that we believe to be the most important to better understand devices we will be working with, and to find any flaws in the overall design at this time. The specific subsystems that were prototyped were chosen based on how important the system was in the design and if we needed to learn more about the system. Each member of the team was assigned a subsystem to prototype and all members of the team learned about the system via the prototyping process. This section will explain each subsystem and how it was prototyped during the prototyping stage of this project.

User Interface

One subsystem that needed to be prototyped was the user interface. This is an integral part of our design since the interface will allow the user to see details about the charging battery, and will be able to provide inputs to change how the battery is charging. A well designed user interface would create ease of use for our users and make the overall project look and feel more professional. The user interface was designed on an arduino using a 16x2 LCD display and a rotary encoder. Figure 1 shows the prototype of this subsystem. The prototype allowed the team to learn more about coding LCD displays and how to function a rotary encoder. This knowledge will be utilized when the final design of the user interface is created. The prototype already allowed the team to see some chances for changes within the design. The LCD screen did not have enough space, nor did it look as professional as the team would like, so for the final design the team planned on changing to a bigger and higher definition LCD screen.

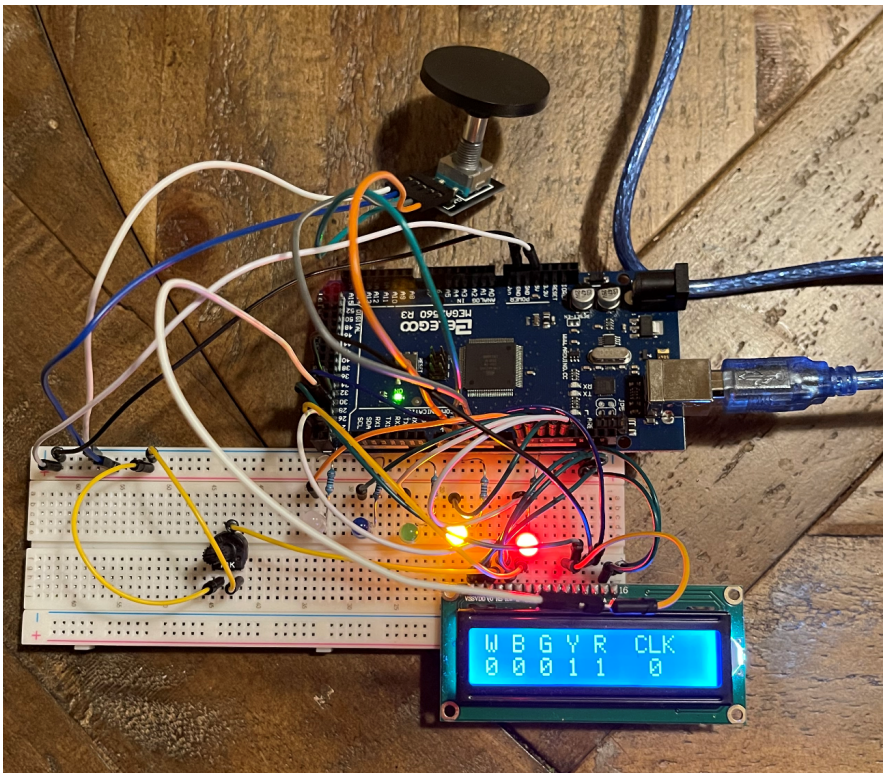

Figure 1: Completed user interface prototype

This prototype was successful. It allowed the team to gain knowledge that will be useful in the further development of the product. The prototype was a chance to test how to code a menu that can be changed and scrolled through using a rotary encoder. While the menu was not as in depth as the final design it still had scrolling features, and allowed the user to press a button to change the menu. These were the results the team was looking to achieve when it was decided that this subsystem would be prototyped. Since the prototyping phase this subsystem has started to be worked on to get closer to the final product wanted by the team. With it being such an important part of the design it was good to see a successful prototype that allows for implementation faster than if the prototype was not successful and required more in depth testing and changes.

Cell Balancing and Charging Algorithms

The initial portion of planning out a prototype for the cell balancing submodule was focused on determining how it would be implemented. In other words, the team needed to figure out what kind of hardware or software would need to be created as a relevant cell balancing submodule for the battery charger. The best resource to aid in this question was the project sponsor, Mr.

John Lehman. He was a key figure in this stage of planning because he knew which hardware components would be required to include in the battery charger. After meeting with Mr. Lehman, he had told us the hardware Dataforth was providing us, the MAQ20 data acquisition system, was able to read voltage, current, and temperature values of the battery cells directly. In addition, the MAQ20 was able to communicate directly with our central microcontroller for the project, the MSP430, via UART to Serial communication. With that in mind, the prototyping of a cell balancing submodule would be solely focused on software, namely programming the MSP430 to react to the data received from the MAQ20. The first step in designing a prototype was to create an algorithm for the MSP430 to execute and command other modules to help balance the cells. Thus, the flowchart, as seen in Figure 3, was created to help aid in visualizing this algorithm.

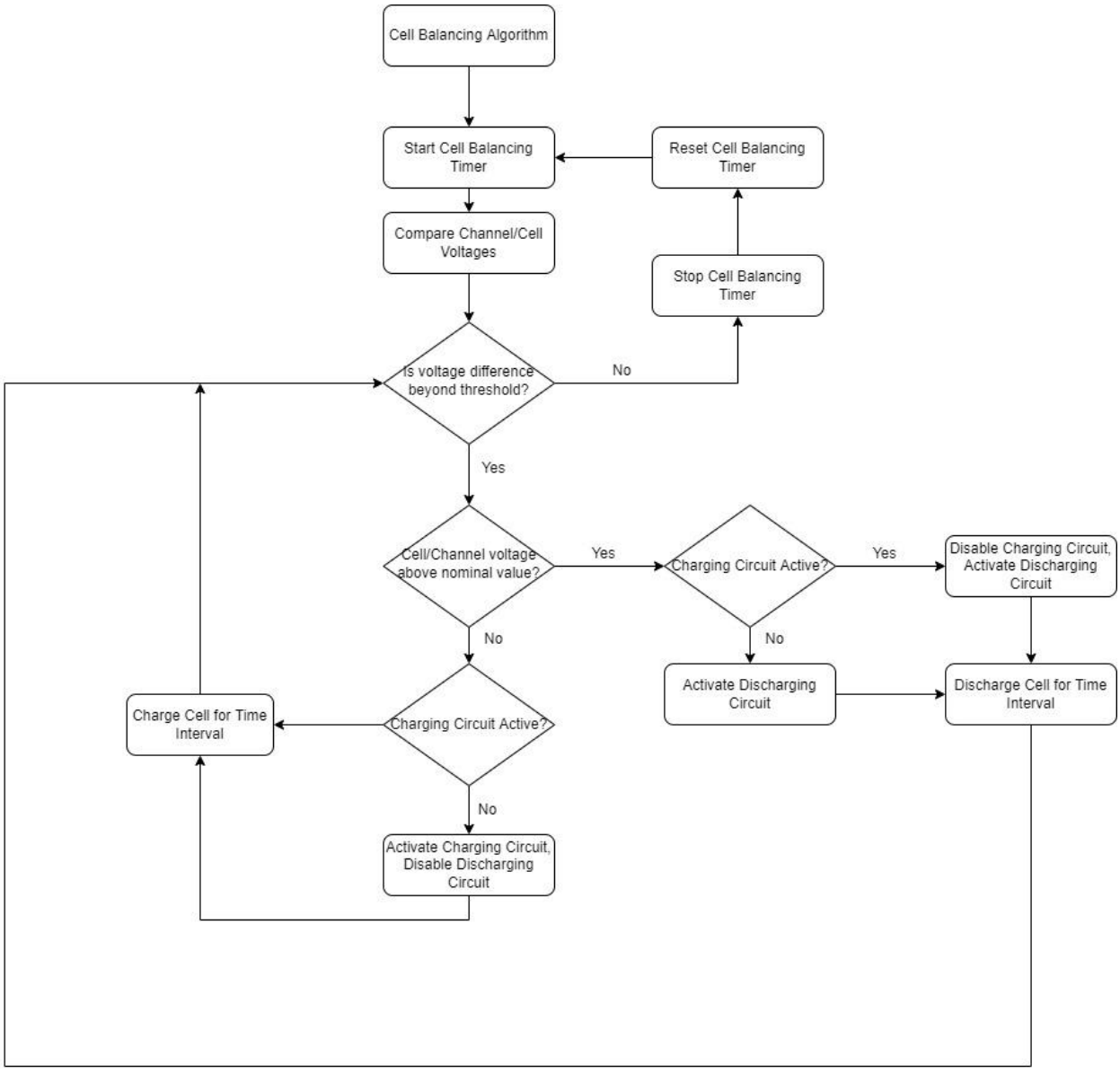

Figure 3: Charging algorithm flowchart

The algorithm is based off of a timer that resets each time before checking the cell voltages. Should the algorithm find that the difference between one cell and the other cells exceeds the threshold (0.1V is the test value), the algorithm then proceeds to check where the cell voltages are in relation to their nominal value. In this case, the nominal voltage of our six-cell LiPo battery at full charge is 3.4V. Should the cell be below the nominal value, the algorithm then ensures that the charging circuit is active for that cell. If the cell is overcharged, or above the

nominal value, the algorithm will then activate the discharging circuit. This charging or discharging will be repeated over a set of time intervals until the cell has matched with the other cells at the nominal voltage rating.

Cell Discharge Hardware

In balancing each cell, discharging current is necessary to level the differing cell chemistries. Conceptually, the best thing to discharge a battery safely is an incandescent light bulb or functional equivalent. This is because the incandescent bulb will vary its resistance as a function of current going through it. As the filament heats up the resistance increases and therefore the voltage drop across the filament can be normalized to a value related to the wattage of the bulb.

While the design includes a varistor, the prototype of our discharging circuit was done via incandescent bulb to an expected outcome. In testing the prototype an ammeter was used to determine the current going through the bulb, which varied as the filament got brighter and warmer. The incandescent bulb prototype was tested to withstand up to three of our LiPo battery cells before the rated wattage was exceeded.

Charging Hardware

Our charging circuit prototype was more conceptual; gathering from research into safety standards for lithium battery charging and battery charging theory. Concepts such as constant current and constant voltage regulation were explored in detail. It was concluded that, in the interest of battery longevity and safety, the entire battery could be charged at a rate of 1C (amp-hours) at the rated voltage of the pack.

This led to the final subsystem design goal of providing a nominal 1.25A at 22.2V to the battery pack terminals. While the prototype of this circuitry was not completed on schedule, the design requirements have brought to light many issues regarding high-performance battery packs. In particular the amount of current every module needs to handle and the control of that current for safety and ease of maintenance [A].

Final Design

From a hardware perspective, the overall structure of the battery charger remained consistent throughout the design process. The hardware block diagram below features a detailed visual of each section of the hardware.

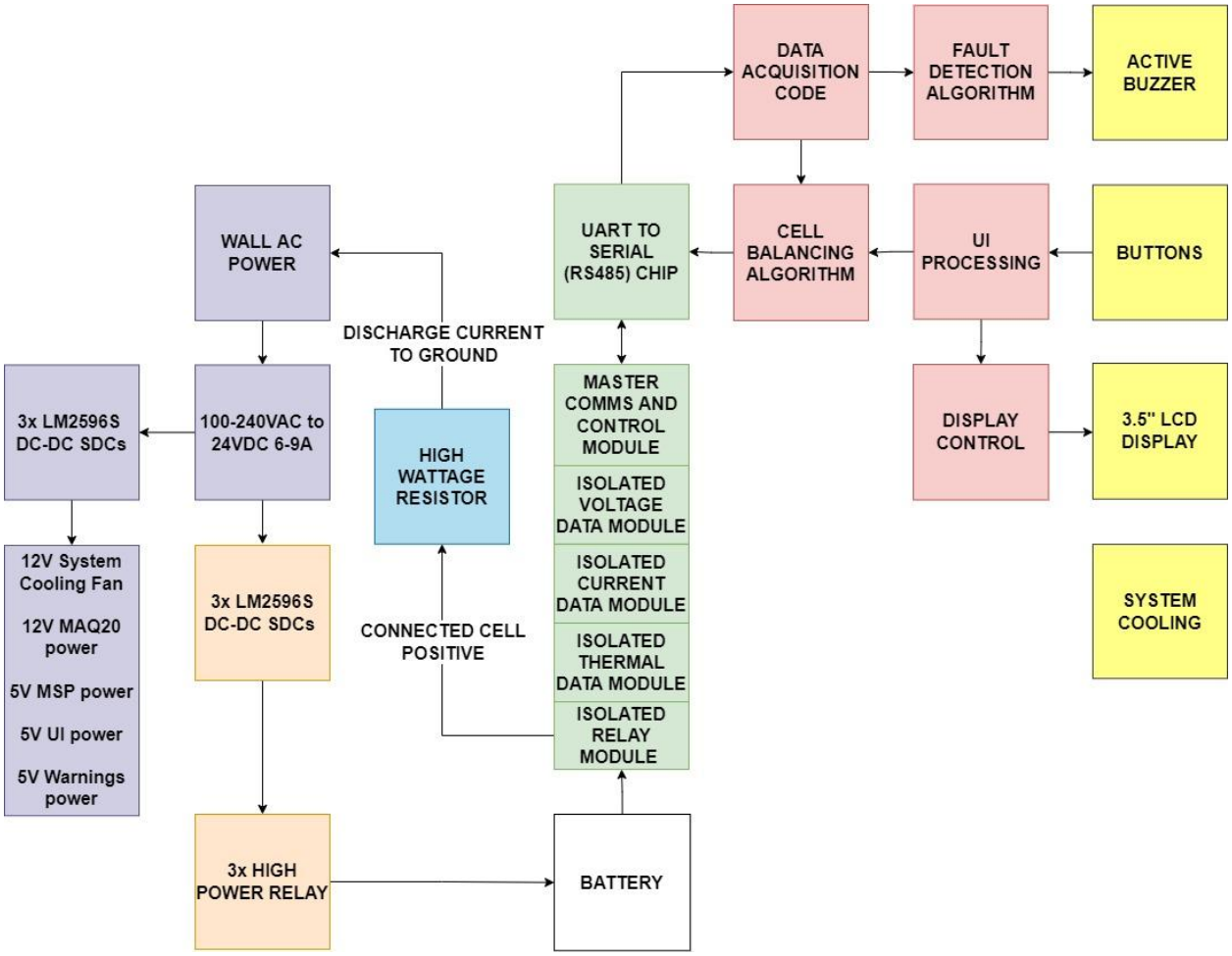

Figure 4: System Architecture - Hardware Block Diagram

Power Supply

The hardware block diagram was color coded to highlight the major hardware components individually. The purple blocks on the left represent the power supply components of the device, which includes a 24VDC Supply Module that can convert the power from a typical 120VAC outlet into 24VDC. This 24V is then fed into an array of DC-DC step-down converters, lowering the voltages to 12V for powering the system cooling fans and MAQ20 data acquisition system, and 5V for powering the MSP430 microcontroller, user interface LCD screen and buttons, and warning buzzers.

Charging Circuit

The blocks highlighted in orange represent the charging circuit of the device. The DC-DC step down converters are meant to reduce the voltage down to approximately 17V. The charging circuit is then switched on or off through the high power relays. These relays are controlled through digital input/output pins from the MSP430 microcontroller.

Data Acquisition & Discharge

The MAQ20 data acquisition system components are represented by the green blocks in the center of the diagram. The MAQ20 includes five separate components. At the top is the communication module, known as the COM4. This module is responsible for retrieving the data from the peripheral modules mentioned below and sending it to the MSP430 for processing. This is followed by the voltage module (ISOV2) which records and stores the voltage values of each cell. The current module (ISOI1) monitors the current being sent into the battery. The thermo data module (JTC) monitors the temperature of each cell. Finally, the isolated relay module individually connects the cells to the discharging resistor in the case of an overcharged cell. The green block above the main MAQ20 modules represents the intermediary device that allows the MAQ20 to communicate effectively with the MSP430. This device was necessary in order to convert the RS485 signals to UART signals that the MSP430 can recognize.

User Interface

The blocks in yellow on the right denote the user interface. The user interface features a 3.5” LCD screen that displays the status of the battery being charged, the voltage, current, and temperature of each cell, and any system warnings. Warnings are additionally noted through the active buzzer, which is activated through a digital output pin from the MSP430. In order for the user to operate the charger effectively, with an array of five buttons and a system power switch. There are two buttons to switch between menu screens, two buttons to alternate between the 1C slow charge and 3C fast charge, and a system reset button.

MSP430 Firmware

The firmware section of the project is represented by the red blocks. The firmware functions include communication with the MSP430, cell balancing, fault detection, user interface commands, and the LCD display control. It was this portion of the system that was the most challenging to implement, and thus underwent the most changes. In the final algorithm, our team decided on a voltage threshold of ±0.05V. Thus, if any cell is between 3.65V and 3.75V, it is considered fully charged. In any overvoltage condition where the voltage of a cell is between 3.7V and 4.2V, the discharge circuit is activated to bring the cell down to the nominal value of

3.7V. In addition, the 1C charging rate is specified as the default charging rate. Finally, conditions specifying the two different charging rates are added. The algorithm of the charger is shown in the flowchart below.

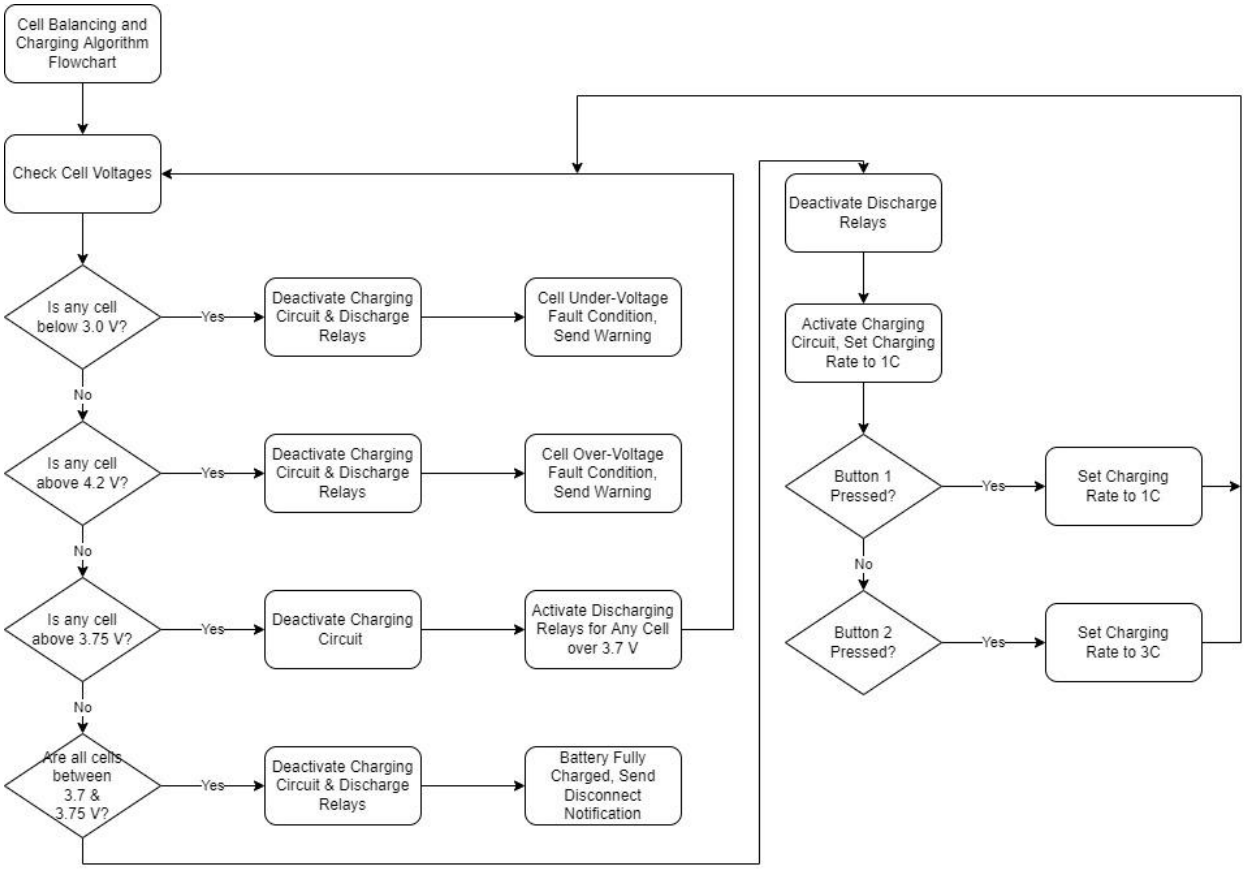

Figure 5: Final Algorithm for Charging, Cell-Balancing, and Fault Conditions

MAQ20 Communication Code

The code required to communicate with the MAQ20 proved to be the most challenging portion of our project. This was due to understanding how to effectively send commands to the MAQ20 to read values of voltage, current and temperature. The MAQ20 data acquisition system utilizes what is known as the MODBUS RTU protocol. This protocol is typically used for industrial electronic devices for communication, which is what Dataforth specializes in. In order to communicate with the MAQ20, the device need to be sent multiple bytes as commands, the first byte denotes the identity of the slave device, followed by the function the device will be performing, two bytes for the address, two bytes for data, and then two bytes for a cyclic redundancy check, which checks for any errors in the transmission of serial data. The example code communicating with the MAQ20 can be found in Appendix E.

SPI Communication Code

Another challenge in the firmware was implementing an effective program to communicate with the LCD screen. The 3.5” screen chosen utilizes a three-wire serial peripheral interface (SPI) configuration, which involves data in, data out, and clocking lines. However, the real challenge was finding a suitable library that could accurately send data to the screen, as the screen is based on the ILI9488 chip set. This is a chipset that is rarely used in conjunction with the MSP430, thus there was no direct library available to establish communication. However, there was one library that allowed communication to some degree in a direct memory access (DMA) SPI library, which is detailed below in Appendix F.

Results

This section will analyze the results of our capstone project. During the testing phase of the project the team gathered data and information from each test to show if the design was working to the requirements our team and the client set. All tests were designed to verify if requirements were met. Figure 3 shows the teams requirements and what type of test was done to test if that requirement was met. The four types of tests performed include: step-by-step and matrix unit tests, an integration test, and acceptance test. The colors in Figure 3 show the status of the test with green being complete and requirement met, yellow being the test was not fully complete, and red showing the test failed.

	Type of Test	Status	Req #	Requirement
4				
5	UTM		1.1*	1C minimum charge rate
6	UTM		1.2	3C maximum charge rate
7				
8	UTS		2.1	Monitor voltage drop across battery
9				
10	Integrate		3.1	Temperature of system side should not exceed 70C
11	Integrate		3.2*	Temperature of battery should not exceed 60C
12				
13	Integrate		4.1*	Monitor and regulate voltage drop across each cell
14				
15	Integrate		5.1	process data from battery pack data acquisition system
16	Integrate		5.2	process data from on-board power circuits
17	Integrate		5.3*	acquire control and response data from dataforth module
18	Inspect		5.4	respond to user input
19	Integrate		5.5	control the battery charger
20	Integrate		5.5.1	use appropriate charging algorithm
21	Inspect		5.6	control display

Figure 3: Tested requirements with type of test completed

Important Test Results

The team had three major subsystem tests during the testing phase of this project. Each subsystem is a major component of the overall design so the test allows the team to see design flaws in the most important parts of the design.

The first major subsystem test was the test for the user interface. This test was to show that the user interface was working and accepting user input. The test objectives were to show the main display with battery voltage, temperature, and charge percent. Show a secondary menu with the individual cell voltage values and show the fault detected screen when a fault was created. Ultimately this test did not work as the LCD screen never reached a functioning state due to Code Composer Studio (CCS) bugs and struggles. While trying to code the LCD many problems arose for the team. CCS was an integrated development environment software that no team member had experience with so the process was slow to learn. Along with the coding the team switched LCD screens to have a screen that could use SPI connectivity to reduce the amount of pins needed on the MSP430. While the LCD not working caused the overall test to fail, the test did allow the team to see that user input was being seen by CCS. Using a button the team could

get input to the computer showing that while the LCD was not working the buttons still were as shown by the button waveforms in appendix A.

This test was a UTS test to walk a user through using the user interface. The first step was to power on the charger and have the LCD turn on. With no battery plugged into the charger the LCD should have displayed that there was no signal from a battery detected. The user would then go to the second menu to see that the LCD showed that all cell voltages were 0V since no battery was detected still. The next step was to plug in a fully charged battery and go back to the main menu. The LCD was supposed to display the voltage of the charged battery which should be between 3.7V-4.2V. The next step would be to go to the second menu to check that all cell voltages display values between 3.7V-4.2V as well. The next step of the test was fault detection. To accomplish this the test called for returning to the main menu and then heating the battery to a temperature above 60°C with a heat gun. The LCD was supposed to display the fault detected screen along with the audible buzz of the active buzzer. The last step was to clear the fault to make sure the LCD returned to the main menu correctly. Again this test was not successful due to LCD problems.

The second major test the team conducted was the charging step down converters matrix test. This test was a black box test since the step down converters were purchased and the team did not know how they would work. The test was to make sure that the step down converters could successfully step down the 24V DC that the power supply provides. Appendix C shows the DC power supply voltage waveforms on both start up and shut down of the charger. The test involved supplying an LM2596 step down converter with the 24V DC, and checking that it could step down to all values needed by the system. For this test all inputs were >13V DC from a power supply. The first expected result was to step down to 12V. The first test was a pass and the step down converter correctly stepped the voltage down to approximately 12V DC. The second test was to step down to the 5V needed by the peripherals such as the LCD screen and buttons for the user interface. This test was also a pass. The next test was to step down to 3.3V for peripherals as well. This test also passed. The next test the team chose the value of 17V just to make sure that the step down converter could step down to the voltage for battery charging. The test passed and the step down converter successfully stepped the 24V to 17V. The next two tests were both to step down to 17V but now there would be amperage delivered under load to make sure the LM2596 could handle stepping down and charging. The first test expected result was 17V with a load of 1.75A which is the load for 1C charging. This test passed meaning the team can charge at 1C. The last test was to step down to 17V with a 3.75A load to test for 3C charging. The test showed closer to 3.1A, but still passed since 3.1A was close enough to the

value that the team expected. Appendix B shows the noise seen through the step down converters which could be a cause for the slight deviation from the expected result.

The final major test the team ran was the MAQ20 signal test. This was a UTS test that focused on the system being able to communicate with the MAQ20 and that the communication could travel to and from the MAQ20.

The first step of this test was to establish that there was communication with the COM4 module. This was meant to be done by powering the MAQ20 system featuring the COM4 module alone and connecting it to the MSP430 via the ISO3086 Serial to UART converter. Once this test proved successful, the peripheral modules would then be individually tested, making sure their respective values, including voltage, current, and temperature, could be measured and sent to the user in a readable format.

However, a significant oversight was made in not having a full understanding of how the UART communication procedure was executed within the MSP430. As a result, time was instead spent catching up on how to effectively send and receive signals via UART from the MSP 430. These preliminary tests included using the UART Tx pin to send signals that represented characters and string values to the terminal. First, the ASCII character ‘A’ was printed out to a terminal via the UART Tx pin. This test was used to understand how to set up the registers required for UART Tx operation and how to send data through the pin. This proved successful, as the oscilloscope reading in Figure 8 shows the reversed value of 0x41 in hex, the ASCII code for the character ‘A’. The second UART Tx test involved using an interrupt in order to send string values to the terminal of CCS. Here, specific registers, along with a vector and interrupt service routine, had to be set up properly to allow “Hello World” to be printed to the terminal via user input with a button. This test proved to be partially successful, as “Hello World4” was printed, as shown in Figure 9. This is likely due to a modulation issue, as the Tx pin records data slightly after the stop bit.

Finally the UART Rx Buffer register was tested by sending a character to the register, in this case ‘t’. This was done using an interrupt as well. In Figure 10, the value 0x72 can be seen in the UCA1RXBUF, the hex value for the character ‘t’. With basic understanding of the UART Tx and Rx pins of the MSP430, substantial progress needs to be made to communicate with the MAQ20 efficiently.

Analysis of Results

The team ran into many unforeseen software problems during the testing phase of this project. This had a major impact on testing and getting results that could be analyzed. With only one successful test out of the three subsystem tests the results the team got were not expected. It also is difficult to correctly analyze the test results when the integration test also could not be run due to the major parts of the integration test failing. While the testing did not go as planned there still was a successful test showing that the charging and discharging could be done, and the system could handle the 24V DC that the power supply provides. It does not provide the team with much, but it can show that on a small scale that the project can perform under the test conditions.

For the major test that failed the team still learned about the devices and code composer studio. More testing is required to get to the client standards but the team has talked with MR. Lehman and talked through redundancy plans if the current software problems persist. More testing will help us deliver a product that meets the standards of the team and Dataforth.

Conclusion

This section will summarize the important requirements of this project and the results of those requirements. The team will also explain the lessons learned of the time we have spent working on this project.

Important Requirements

All requirements for this project are important to meet, but there are select requirements that are considered more important to the team because these requirements are the most important to meet the goals of the client. These important requirements are marked by asterisks in Figure 2.

Requirement 1.1: The battery should be able to charge at a minimum charge rate of 1C. This requirement is important due to the main goal of the project being to charge a battery. The overall goal is to produce a device that can charge a four-cell LiPo battery, and if the device cannot charge at 1C then the battery will not charge in a sufficient timespan. This requirement is also important for the safety of the battery charger. Charging at 1C is the rate that is considered safe for charging all lithium-ion chemical base batteries; and with safety being an important factor for this project, charging at the correct rates is important. This requirement

being successfully met also allows the team to then test fast charging at higher rates, which started as a stretch goal but grew to a separate project requirement. Requirement 1.1 was successfully met by the team. We could successfully charge a battery at a rate of 1C. The team actually was able to charge the battery at multiple rates including up to 3C, going above the original requirement the team and client had set.

Requirement 3.2: The temperature of the LiPo battery should not exceed 60°C while charging. This requirement is considered important for device, client, and team safety. If the battery exceeds 60°C while charging then safety hazards become more probable, and harm could be caused to the team during testing, or the client with the final product. If the battery overheats it could cause a fire, or in some cases could explode. Both these cases need to be avoided by the team, meaning that keeping the battery in a safe temperature range, which has been determined by IEEE standards to be under 60°C, is an important requirement to meet for the team. The team successfully met requirement 3.2 while charging and discharging the battery. The battery did not get higher than the designated 60°C. This meant the team met the requirement and was able to show we could charge a battery while maintaining safe temperatures.

Requirement 4.1: The device should monitor and regulate voltage drop across each cell and discharge if the voltage meets or exceeds the nominal value of 3.7V/cell. This requirement is important to maintain the integrity of the battery and not cause damage from overcharging. The requirement also shows that the discharging circuitry and algorithms are working correctly. Discharging correctly will also allow the battery to stay on the charger without needing to remove the charger as soon as the battery is fully charged as the battery will continuously discharge to maintain nominal levels. The team partially met requirement 4.1. The team could partially measure the voltage across each cell, but the team did not get the MAQ20 communication to work making this requirement only partially met.

Requirement 5.3: Acquire control and response data from the Dataforth MAQ20 data acquisition module. This requirement is extremely important for the team. The client has specifically asked for a battery charger that will show Dataforth products in use to be able to take to trade shows and demos. Not meeting this requirement would mean that the team would not be producing a product that is up to the clients primary specifications. The team was given freedom on the design but Dataforth integration is needed for minimum satisfaction. This requirement was not met by the team. The team had difficulty combining CCS UART communication with the MAQ20 communication protocols. Although the team continuously

worked on this requirement with our clients guidance, there was not successful communication resulting in this requirement not being met.

Lessons Learned

The team learned many lessons from our shortcomings in the first round of testing our project. During both step tests the team learned that software can cause problems that hold up other parts of the project.

The team learned that hardware swaps late in a project can cause problems due to small nuances between different hardware components. The team also learned that when one thing does not work as expected then other important tests or devices will not work as well. This caused a failed test and hours reexamining the project to try and fix the problems and continue moving on.

With the help of Mr. Lehman the team has made progress on the problems we faced, and also learned that redundancy plans are a good idea in the instance something cannot be fixed in the amount of time allotted. Due to this the team implemented plans as backups.

Acknowledgements

We as team Lithium Lumberjacks thank our client John Lehman who represents Dataforth, proposed the project initially, and provided ample resources toward the projects' success. We would also like to thank Dr. Severinghaus and Mahsa Keshavarz for guiding us through the project and mentoring us along the way.

User Manual

Introduction

Thank you for choosing the Dataforth Battery Charger and we hope you enjoy what we have developed for you. The Lithium Lumberjacks are glad to support you in your business needs. As requested our team has developed a custom battery charger and battery management system to help you demonstrate the power of Dataforth products for your clientele. Our product features many customized systems to meet the high standards of Dataforth products Including:

- A user display
- Full Integration of the Dataforth MAQ20 modules
- Safety systems to provide safe use of our product
- A portable design for easy travel to tradeshows and demos

This user's manual will walk you through how to not only use our product, but also walk you through maintenance and troubleshooting of our product for continued use. Our goal is for you to have a lasting product that provides benefits and profits for your business.

Your need for a battery charger that integrated Dataforth products to show clients the many uses of your data acquisition and communication devices prompted our team to research and develop a high functioning device that can utilize different Dataforth products. Our team used the Dataforth MAQ20 system as the focal point of our design. This allowed the team to show the many uses of the MAQ20. The team also kept the MAQ20 on the exterior of the design so that at demos and tradeshows clients will be able to fully see the MAQ20 putting Dataforth products at the forefront of the demo, with the overall product being focused on showing off the Dataforth product. Not only did the team recognize the need to show how useful Dataforth products are, but we also designed our product to meet the safety standards set forth by IEEE. Creating a safe functioning product was important for the team. Our team recognized Dataforth's focus on safety and integrated that same high safety standard into our product. The device charges batteries without exceeding high temperatures. This keeps the battery, and all devices inside of our system safe. Our team focused on your needs to develop a safe functional system that complimented your products to allow for you to display your products and their uses to clientele.

Our system is made up of five main subsystems. The first subsystem is the user interface. This subsystem consisted of a high definition LCD screen, five buttons, and an active buzzer. The user interface was made to be simple for anyone to use. Each button has a designated function, the active buzzer allows for an audible alarm if a fault is detected in the system. The large LCD screen allows for easy viewability for the user. The second subsystem is the TI MSP430 microprocessor. This subsystem is the brains of our device. The MSP430 was chosen due to the relationship Dataforth has with Texas Instruments. The third subsystem is the charging circuit. This subsystem is created with step-down converters and relays to allow for the charging of the different battery cells. The fourth subsystem is the discharging circuit. This circuit is also created with relays, but the important part of this subsystem is a high wattage 4 ohm resistor. This resistor being high wattage is important for heat dissipation in the case where all cells discharge at once. The fifth and final subsystem is the MAQ20 system along with the necessary communication modules to use the MAQ20. This subsystem is the main communication between the battery and the MSP430. This subsystem is important because it monitors all the battery cells and communicates with the rest of the subsystems to allow for each subsystem to complete their designated task. This subsystem also displays the functionality of Dataforth

products. Each of our subsystems were designed with both safety and your company needs in mind. Our team included safety details and device protection into all our subsystems. Consistent grounding and electromagnetic interference protection was a focal point in all subsystems where it was important. Attention to all details was important to our team knowing Dataforth also has a focus on precision and detail in their products.

Over the course of the year our team was focused on designing a product that would meet your business needs. We followed the engineering design process and focused on your needs to create a product we felt met your standards and needs. Starting from in depth research of different battery chargers and battery charging algorithms our team developed our system using innovative charging algorithms. Our prototyping was done with a focus on how we could meet your required specifications along with trying to create a device that was complete, and provided functions above those requested. Using a larger, high definition LCD screen was a change made to allow for ease of use. We understood that a goal was to show how Dataforth products can be used so the team not only integrated the MAQ20, but kept it in view of the user so that Dataforth was a focal point when the product was in use. Finally, we developed our product with systems that could withstand being transported long distances, allowing for our device to be taken anywhere to be demonstrated.

The rest of this user's manual will walk through how to use our product to its fullest potential. We again hope you are satisfied with the product we developed for you to use in demonstrating Dataforth product power. We also would like to thank you for your continued support to our team throughout this project.

Installation

This section of the manual will provide you with an in depth understanding of how to correctly set up our product before use. Our product set up was made to be as simple as possible for easy installation and set up at tradeshows and client demonstrations.

To install our product you will need access to 115-125VAC wall power. To install our product you must plug into wall power. After plugging into wall power the user will need to flip the red power switch located on the back of the device next to the power cord. After ensuring the device is plugged in and the switch is in the on position the device is ready for use.

To use the device plug in your four-cell LiPo battery into the yellow XT60 connector located on the front of the device. Once the battery is connected the device has been correctly set up for use.

Configuration and Use

The Dataforth Battery Charger allows for ease of use through the user interface. Upon startup, the device will check the charging status of your battery by evaluating the voltages of each of the cells. Once the battery charger determines the battery needs to be charged, it begins charging at the slow rate of 1C. This serves as the default charging rate upon startup.

LCD Screen

The LCD Screen provides all of the information the user needs during charging. Information such as the voltage of each cell, the current the battery is being charged at, and the temperature of each cell. The LCD screen will also alert the user of any faults that occur while charging.

Interface Buttons

The user interface of the charger features five buttons to actuate multiple functions on the device, such as switching between menus, selecting the charge rate, and performing a system reset.

Switching Between Menus - Utilize the two black buttons on the right side of the device to switch between menus of the display screen. The order is as follows

- Main Menu - Displays charging percentage, current charging rate, and overall battery temperature
- Secondary Menu - Displays the voltage and temperature information for each of the individual LiPo cells.

Selecting the Charging Rate - There are two buttons for setting the charge rate. The central black button is used to select the slow charging rate of 1C. This charging rate is automatically applied upon system startup and after system reset. The red button selects the fast charging rate of 3C.

System Reset - The button on the very right of the interface is for a user initiated system reset. The reset procedure will deactivate the device from charging or discharging by opening the relay circuits of the charging and discharging circuits, thus disconnecting the battery. The system

will then be able to reconnect to the charging and discharging circuits by choosing a charging rate of 1C or 3C.

Fault Conditions

Cell Fault is most likely to happen due to plugging the battery in the wrong direction, as the battery connector itself has no discernable directional plug for the cell balance bus. It may also occur due to a reverse charged cell, which would be a cause for battery disposal.

Battery Overtemp can happen due to battery overcharge and failure of the charging circuitry within the system. It may also be caused by an imbalance in the chemistry such that charging or discharging the cells causes thermal runoff.

System Overtemp may happen due to a passive component burning. While unlikely, it may also be caused by overheated discharge or charge wires as they may carry up to 10A of current.

Maintenance

The Dataforth Battery Charger is meant to provide reliable charging over many charging cycles. However, to ensure users are able to properly maintain their device properly, links to charger components have been provided below.

Charging Circuit Relays: Minimum of 100,000 cycles before needing replacement
https://www.mouser.com/ProductDetail/Panasonic-Industrial-Devices/JW1AFSN-DC5V-F?qs=bpFJJ1fyfoDGhqtQ7d1JMw%3D%3D

3.5” LCD Screen:
https://www.amazon.com/gp/product/B08QYNMSZM/

Troubleshooting Operations

While no product is flawless, the team has built this system to be exceptionally robust. In this section we will cover issues that may occur and how to correctly troubleshoot them.

Code Not Loading

One issue that could occur is the device code not staying loaded on the MSP430 on shutdown. While the code should stay on the MSP430, if it ever does not reload on startup simply open the

face of the chassis by removing the four screws on the chassis face. Once removed the chassis should be able to open. Inside you will find the MSP430. Using a USB to Micro-USB cable connect a computer or laptop to the MSP430 and reload the device code. Once done, put the chassis back together and the device should be functioning properly again.

No Power to a Subsystem

If any portion of the product stops receiving power follow these steps to troubleshoot. First step would be to check the fuse located next to the power switch on the back of the device. If the fuse is not blown then the chassis will need to be opened. To open the chassis remove the four screws on the chassis face and the chassis should now open. Once opened, locate all seven step down converters. A reference step down converter is shown in Figure 1. Each step down converter has three LEDs located on the right of the device. Find the output indicator LED labeled “OK”. If that LED is solid red then that step down converter is no longer working. If this occurs refer to the maintenance section of the manual to order a new step down converter. If all step down converters are functioning the next step would be to test for any loose connections within the device with a continuity test. If a loose connection is found, reconnect and put the device back together. If no loose connection is found the device that is not receiving power may no longer be working. Refer to the maintenance section to order a replacement part.

Conclusion

The Lithium Lumberjacks would like to thank you for your decision to work with us on the development of this product. Our team is happy to have worked with your company and hope our product continues to be of use to you for years to come. While our team will be moving forward into our professional careers in the coming months feel free to reach out to us at our team email lithium.lumberjacks@nau.edu. Thank you again for your time and business.

Appendix A

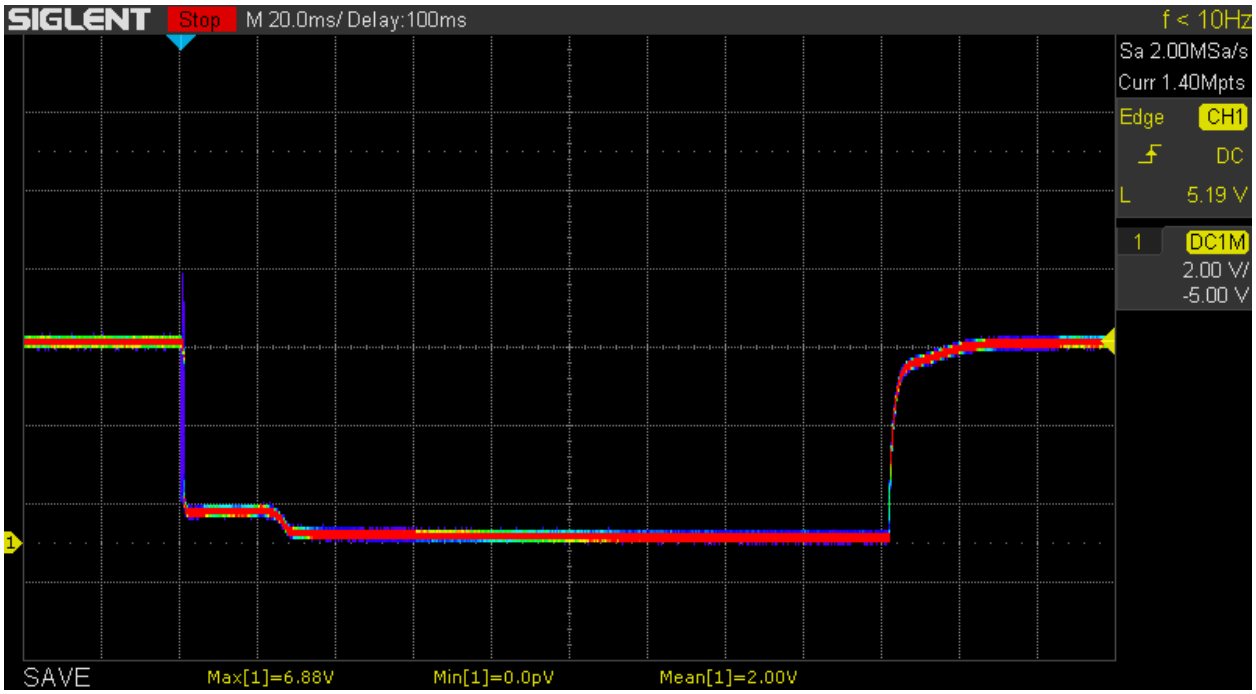

Figure 1: Bounce and rise of the reset switch. Voltage spike on left at first contact of the switch will require the use of decoupling capacitors.

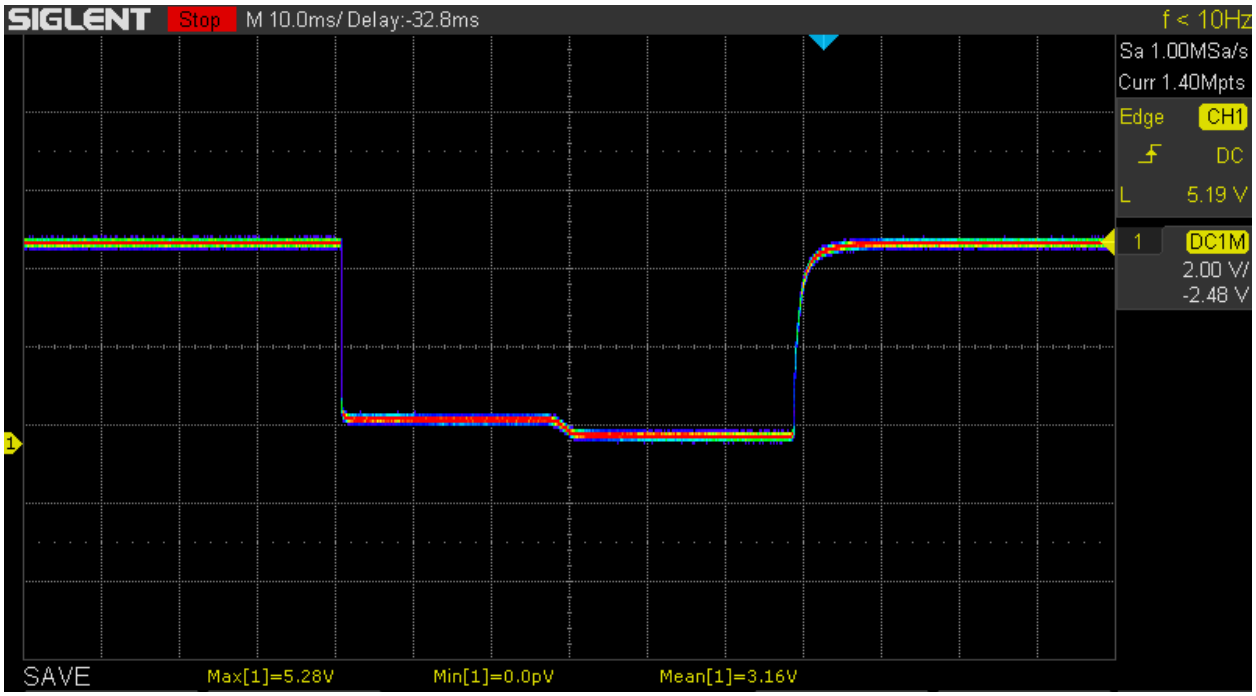

Figure 2: Other buttons have less bounce. Smaller bypass capacitors would be acceptable.

Appendix B

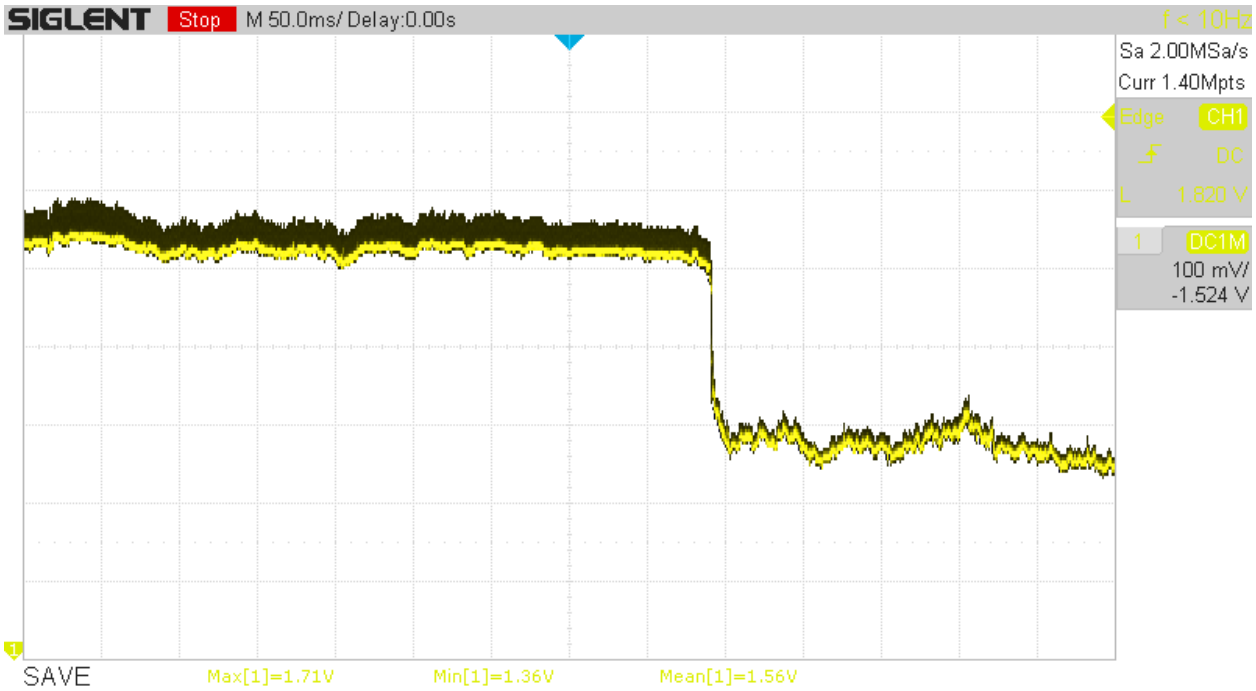

Figure 1: Approx. 0.5V noise off the step down converters shown in the variation of the line.

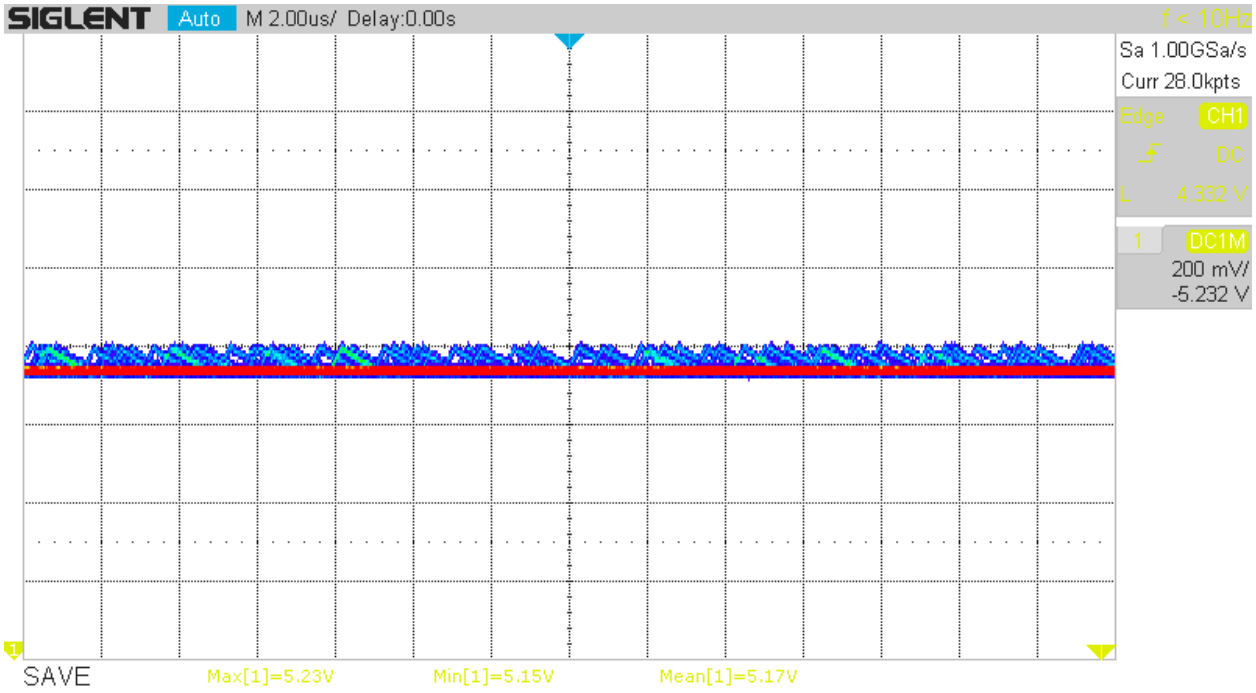

Figure 2: More congruent noise on a moderately consistent voltage line.

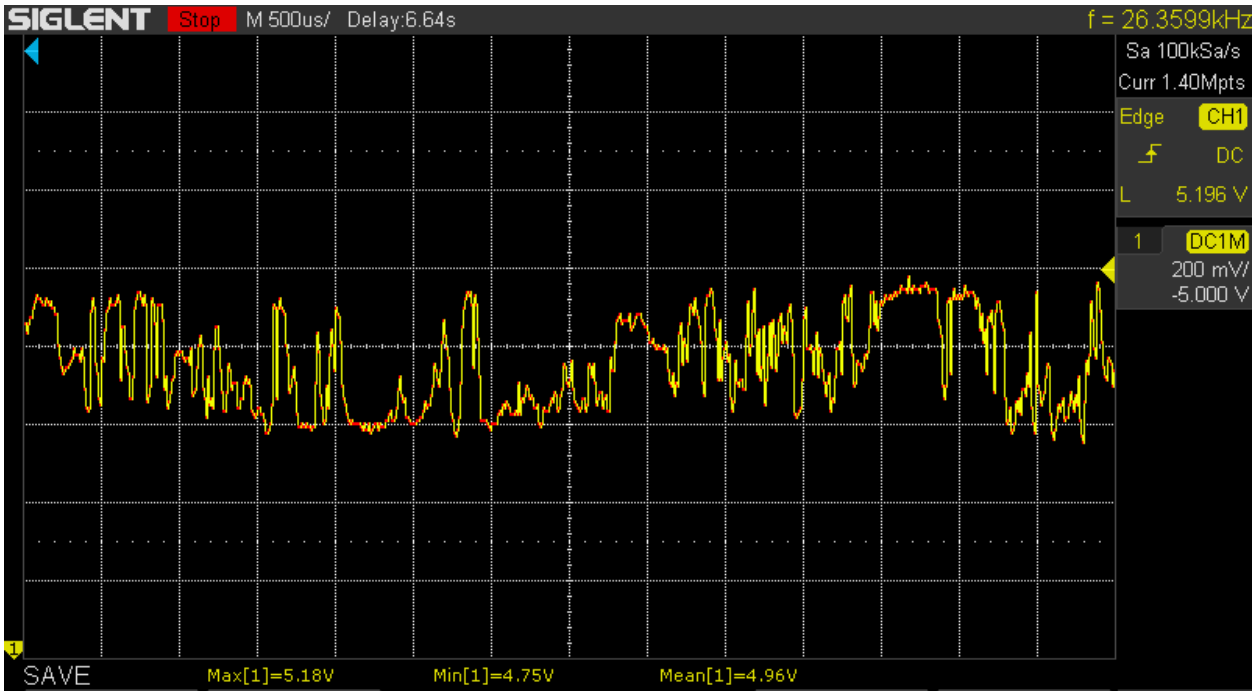

Figure 3: Severe noise on the output of a step down converter.

Appendix C

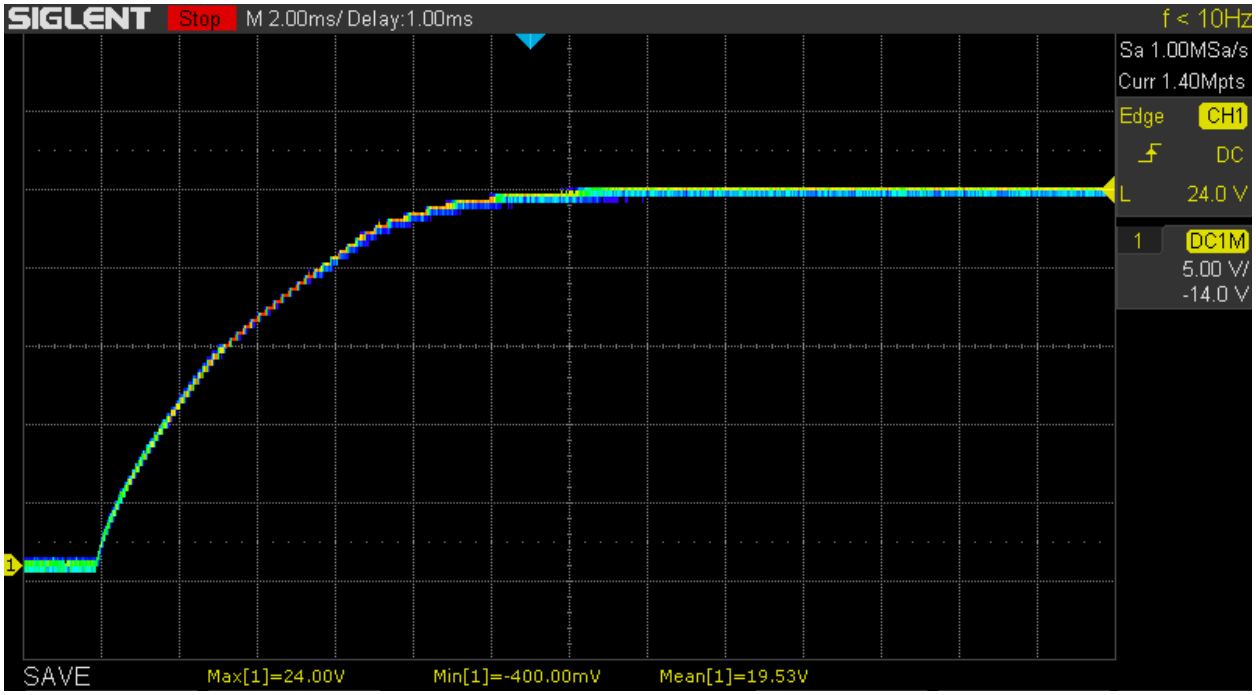

Figure 2: Rise signal for the primary power module upon startup.

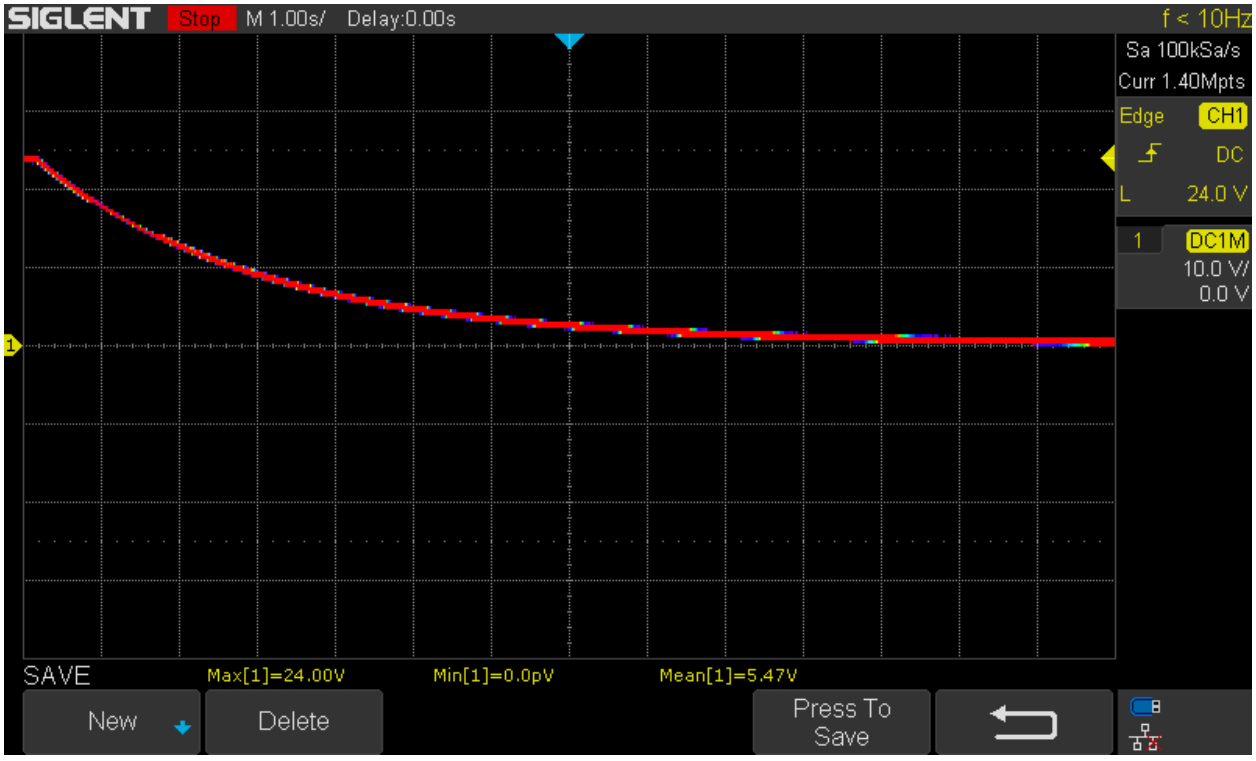

Figure 2: Falling voltage of the primary power module upon shutdown.

Appendix D

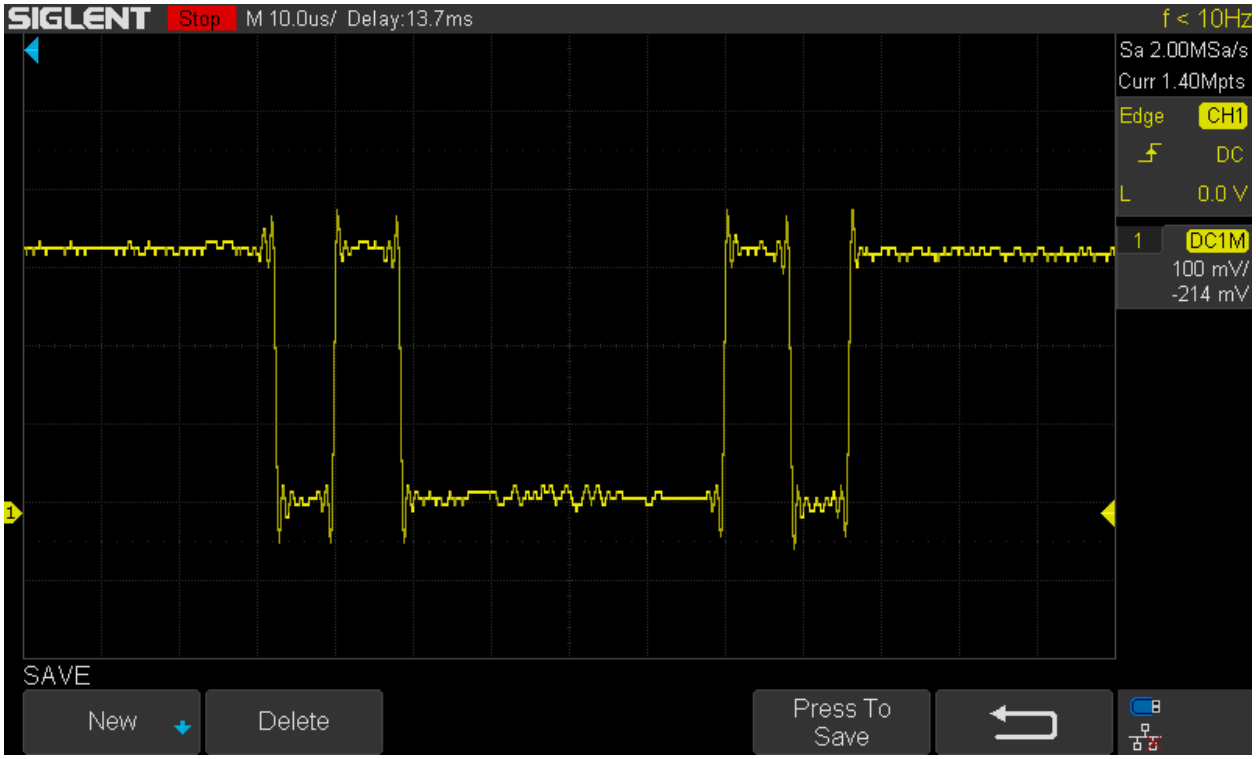

Figure 1: Signal Waveform of the ASCII Character 'A' (0x41) from the MSP430 TX Output

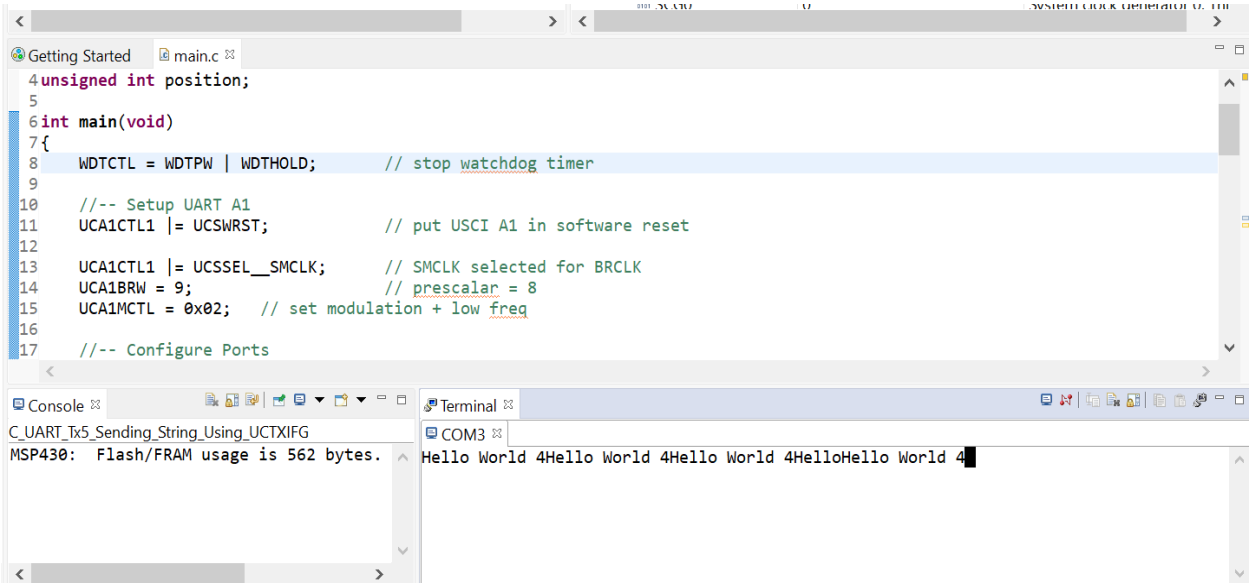

Figure 2: Terminal Output of the String 'Hello World' from MSP430 TX Output with Interrupt

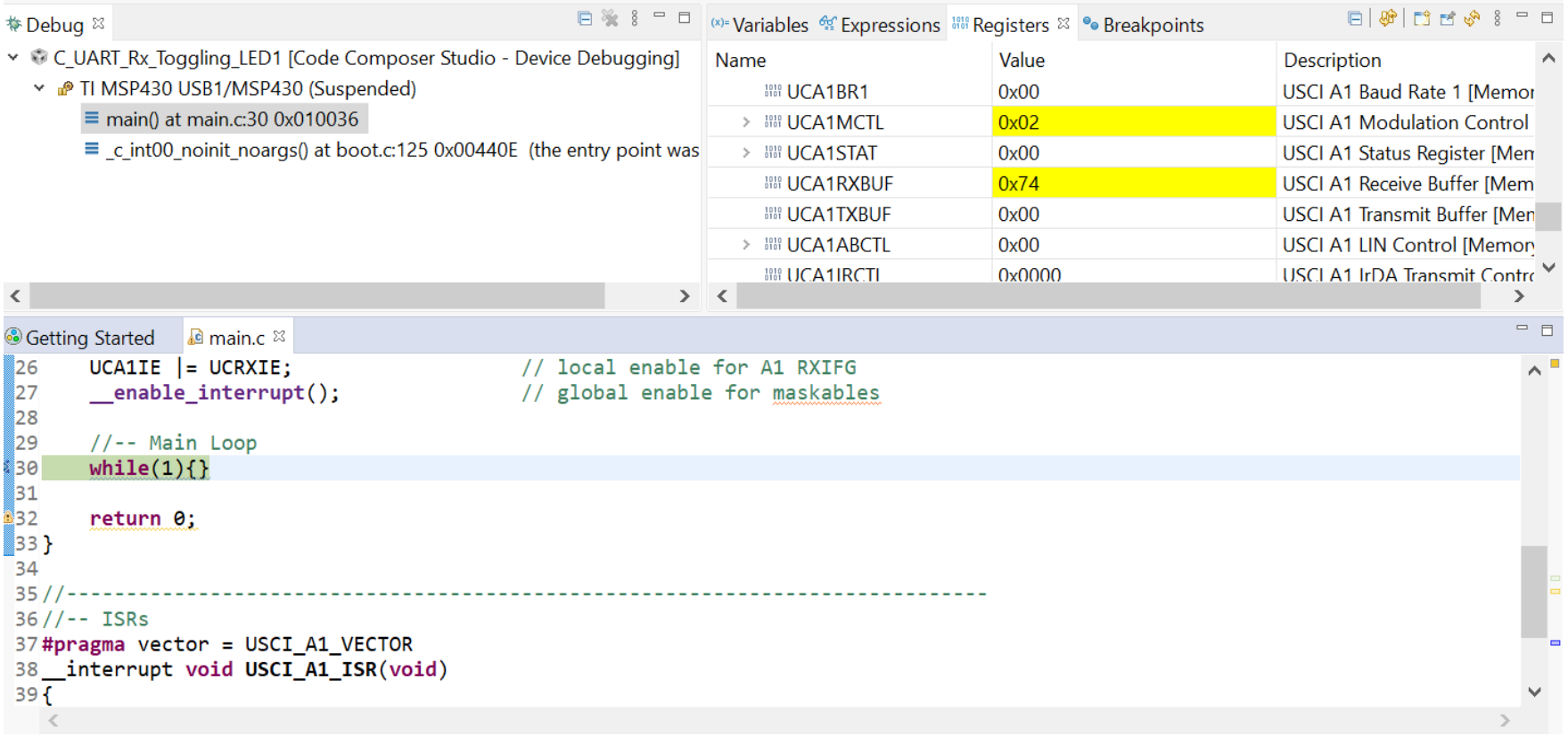

Figure 3: RX Buffer Register with Input Character 't'

Appendix E

```
// Sean Conlin
// header files
#include <msp430.h>
#include "src_CRC16.h"

// global constants
int WRITEREGISTER[] = [0x10 0x06 0x15 0x00 0x00 0x01 0x08
0xFF];
int READVOLTAGE[] = [0x10 0x04 0x1E 0x00 0x00 0x01 0x08 0xFF];
unsigned int position;

// main function
int main(void)
{
    WDTCTL = WDTPW | WDTHOLD;        // stop watch-dog timer

    //-- Setup UART A1
    UCA1CTL1 |= UCSWRST;             // put USCI A1 in software
reset

    UCA1CTL1 |= UCSSEL__SMCLK;       // SMCLK selected for BRCLK
    UCA1BRW = 9;                     // pre-scalar = 9
    UCA1MCTL = 0x02;   // set modulation + low frequency

    //-- Configure Ports

    P2DIR &= ~BIT1;                   // Pin 2.1 Set to Input
(S1)
    P2REN |= BIT1;                    // Enable resistor
    P2OUT |= BIT1;                    // Set resistor to pull up
    P2IES |= BIT1;                    // Make IRQ sense H to L
edge

    P4SEL |= BIT4;                    // Pin 4.4 function set to
UART A1 TX
```

```
    UCA1CTL1 &= ~UCSWRST;                   // Take USCI A1 out of
software reset

    //-- Enable IRQs

    P2IE |= BIT1;                           // Enables S1 IRQ
    P2IFG &= ~BIT1;                         // Clears interrupt flag
    __enable_interrupt();                   // Global en IRQs

     for (position = 0; position < sizeof(READVOLTAGE);
position++)
     {
          UCA1TXBUF = READVOLTAGE[position];

    return 0;
}
```

Appendix F

```
// Sean Conlin
// header files
#include "driverlib.h"

// global constants
char TxString;
char RxString;

// main function
void main (void)
{
    //Initialize Watchdog in Interval Timer Mode
    /* Base Address of WDT module
     * Use ACLK as clock source to WDT module
     * Divide ACLK by 32k to assert interval every 1 second
     */
    WDT_A_initIntervalTimer(WDT_A_BASE,
        WDT_A_CLOCKSOURCE_ACLK,
        WDT_A_CLOCKDIVIDER_32K);

    //Enable Watchdog Interupt
     SFR_clearInterrupt(WDTIE);
    SFR_enableInterrupt(WDTIE);

    //Clear P1.0 LED off
    GPIO_setOutputLowOnPin(

        GPIO_PORT_P1,
        GPIO_PIN0
        );

    //Set P1.0 to output direction
    GPIO_setAsOutputPin(
        GPIO_PORT_P1,
        GPIO_PIN0
        );
```

```
    //P3.4,3 & P2.7 option select
    GPIO_setAsPeripheralModuleFunctionInputPin(
        GPIO_PORT_P3,
        GPIO_PIN4 + GPIO_PIN3
        );
    GPIO_setAsPeripheralModuleFunctionInputPin(
        GPIO_PORT_P2,
        GPIO_PIN7
        );

     //Initialize Master
     USCI_A_SPI_initMasterParam param = {0};
    param.selectClockSource = USCI_A_SPI_CLOCKSOURCE_ACLK;
    param.clockSourceFrequency = UCS_getACLK();
    param.desiredSpiClock = 500000;
    param.msbFirst = USCI_A_SPI_MSB_FIRST;
    param.clockPhase =
USCI_A_SPI_PHASE_DATA_CAPTURED_ONFIRST_CHANGED_ON_NEXT;
    param.clockPolarity =
USCI_A_SPI_CLOCKPOLARITY_INACTIVITY_HIGH;
    USCI_A_SPI_initMaster(USCI_A0_BASE, &param);

     //Enable SPI module
    USCI_A_SPI_enable(USCI_A0_BASE);

    //Wait for slave to init
    __delay_cycles(100);

    //USCI_A0 TX buffer ready?
    while (!USCI_A_SPI_getInterruptStatus(USCI_A0_BASE,
UCTXIFG)) ;

    //Initialize and Setup DMA Channel 0
    /*
     * Configure DMA channel 0
     * Configure channel for single transfer
```

```
     * DMA transfers will be disabled and interrupt flag will
be set after every
     *   1 transfer
     * Use DMA Trigger Source 17 (UCA0TXIFG)
     * Transfer Byte-to-byte
     * Trigger transfer on signal held high
     */
    DMA_initParam param0 = {0};
    param0.channelSelect = DMA_CHANNEL_0;
    param0.transferModeSelect = DMA_TRANSFER_SINGLE;
    param0.transferSize = 1;
    param0.triggerSourceSelect = DMA_TRIGGERSOURCE_17;
    param0.transferUnitSelect = DMA_SIZE_SRCBYTE_DSTBYTE;
    param0.triggerTypeSelect = DMA_TRIGGER_HIGH;
    DMA_init(&param0);
    /*
     * Configure DMA channel 0
     * Use TxString as source
     * Increment source address after every transfer
     */
    DMA_setSrcAddress(DMA_CHANNEL_0,
        (uint32_t)(uintptr_t)&TxString,
        DMA_DIRECTION_INCREMENT);
    /*
     * Configure DMA channel 0
     * Use SPI TX Buffer as destination
     * Don't move the destination address after every transfer
     */
    DMA_setDstAddress(DMA_CHANNEL_0,

USCI_A_SPI_getTransmitBufferAddressForDMA(USCI_A0_BASE),
        DMA_DIRECTION_UNCHANGED);

    //Initialize and Setup DMA Channel 1
    /*
     * Configure DMA channel 1
     * Configure channel for single transfer
```

```
     * DMA transfers will be disabled and interrupt flag will
be set after every
     *   1 transfer
     * Use DMA Trigger Source 16 (UCA0RXIFG)
     * Transfer Byte-to-byte
     * Trigger transfer on signal held high
     */
     DMA_initParam param1 = {0};
    param1.channelSelect = DMA_CHANNEL_1;
    param1.transferModeSelect = DMA_TRANSFER_SINGLE;
    param1.transferSize = 1;
    param1.triggerSourceSelect = DMA_TRIGGERSOURCE_16;
    param1.transferUnitSelect = DMA_SIZE_SRCBYTE_DSTBYTE;
    param1.triggerTypeSelect = DMA_TRIGGER_HIGH;
    DMA_init(&param1);

    /*
     * Configure DMA channel 1
     * Use SPI RX Buffer as source
     * Don't move the source address after every transfer
     */
    DMA_setSrcAddress(DMA_CHANNEL_1,
        USCI_A_SPI_getReceiveBufferAddressForDMA(USCI_A0_BASE),
        DMA_DIRECTION_UNCHANGED);

    /* Configure DMA channel 1
     * Use RxString as destination
     * Increment destination address after every transfer
     */
    DMA_setDstAddress(DMA_CHANNEL_1,
        (uint32_t)(uintptr_t)&RxString,
        DMA_DIRECTION_INCREMENT);

    //Clear TxString && RxString
    TxString = RxString = 0;

    //Enter LPM3 w/ interrupts
    __bis_SR_register(LPM3_bits + GIE);
```

```
    //For Debugger
    __no_operation();
}

//Trigger DMA0 & DMA1 block transfer.
#if defined(__TI_COMPILER_VERSION__) ||
defined(__IAR_SYSTEMS_ICC__)
#pragma vector=WDT_VECTOR
__interrupt
#elif defined(__GNUC__)
__attribute__((interrupt(WDT_VECTOR)))
#endif
void WDT_A_ISR (void)
{
    if (TxString - 1 == RxString){
        //set P1.0
        GPIO_setOutputHighOnPin(

            GPIO_PORT_P1,
            GPIO_PIN0
            );
    } else   {
        //Clear P1.0 LED off
        GPIO_setOutputLowOnPin(

            GPIO_PORT_P1,
            GPIO_PIN0
            );
    }
    //Increment TxString counter
    TxString++;

    //Enable DMA transfers on channel 1
    DMA_enableTransfers(DMA_CHANNEL_1);

    //Enable DMA transfers on channel 0
    DMA_enableTransfers(DMA_CHANNEL_0);
}
```

Appendix G

[1] Dataforth Corporation, “About Us,” Accessed: Nov. 2021. [Online]. Available: https://www.dataforth.com/about-us.aspx

[2] F. Lima *et al*., "A novel universal battery charger for NiCd, NiMH, Li-ion and Li-polymer," *ESSCIRC 2004 - 29th European Solid-State Circuits Conference (IEEE Cat. No.03EX705)*, 2003, pp. 209-212, doi: 10.1109/ESSCIRC.2003.1257109.

[3] H. V. Nguyen and D. Lee, "Single-phase multifunctional onboard battery chargers with active power decoupling capability," *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, 2018, pp. 3434-3439, doi: 10.1109/APEC.2018.8341597.

[4] Chia-Hsiang Lin *et al*., "Fast charging technique for Li-Ion battery charger," *2008 15th IEEE International Conference on Electronics, Circuits and Systems*, 2008, pp. 618-621, doi: 10.1109/ICECS.2008.4674929.

[5] "IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications," in IEEE Std 1679.1-2017 , vol., no., pp.1-47, 31 Jan. 2018, doi: 10.1109/IEEESTD.2018.8262521.

[5] Panasonic Corporation, “Power relays (Over 2 A) - JW Relays,” JW1AFSN-DC5V-F datasheet, [Revised May 2021].

[6] Littelfuse, “MLA Varistor Series - Surface Mount Multilayer Varistors (MLVs),” V3.5MLA0603NA datasheet, [Revised May 2021].

[7] Dataforth Corporation, “MAQ 20 data acquisition systems - Communications Modules,” MAQ20 COM4 datasheet, 1995 [Revised 2017].

[8] Dataforth Corporation, “MAQ 20 data acquisition systems - Analog Input Modules: Process Voltage & Process Current,” MAQ20 ISOV2/I1 datasheet, 1995 [Revised 2017].

[9] Dataforth Corporation, “MAQ 20 data acquisition systems - Analog Input Modules: Thermocouple,” MAQ20 JTC datasheet, 1995 [Revised 2017].

[10] Dataforth Corporation, “MAQ 20 data acquisition systems - Discrete Output Module: Relay,” MAQ20 DORLY20 datasheet, 1995 [Revised 2017].

[A] "IEEE Guide for the Characterization and Evaluation of Lithium-Based Batteries in Stationary Applications," in IEEE Std 1679.1-2017 , 31 Jan. 2018, doi: 10.1109/IEEESTD.2018.8262521. [Online][Accessed Sept 2021].