ACTIVE FLYBACK BASED BATTERY MANAGEMENT SYSTEM WITH PROPORTIONAL BALANCING FOR USE IN AN ELECTRIC RACE CAR

by

MATTHEW MARTIN

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON
August 2016

To all of my friends and family that supported me throughout my scholastic career, including my father Doug, my mother Betty, Kyrsten, Isaac, David, and Donald.

ACKNOWLEDGEMENTS

First off, I would like to thank my supervising professor Dr. David Wetz for his support in this endeavor and dedication to me as a graduate student. None of this would have been possible if it were not for his dedication to making sure I had all of the proper tools, parts, software, and most importantly time to keep the project moving forward.

I would also like to extend a special thanks to several of my lab mates who, at some point or another, dedicated some time to helping me work through a problem or decide on a direction. Isaac Cohen, Donald Dodson, Chris Williams, and Clint Gnegy-Davidson, thank you all for the help and support! I literally could not have done it without you guys.

I would also like to thank Dr. Bob Woods and the entire Formula SAE racing team here at The Univeristy of Texas at Arlington for motivating me to become a better engineer and for being an inspiration and an outlet that breeds creativity.

Lastly I would like to thank my family for sticking by my decisions and supporting me to the very end. I could not have made it to where I'm at if it were not for my father Doug, my mother Betty, and Kyrsten. Thank you guys!

July 29th, 2016

ABSTRACT

MATTHEW MARTIN, M.S.

The University of Texas at Arlington, 2016

Supervising Professor: David A. Wetz

Battery management systems are responsible for many tasks, most important of which is maintaining balance amongst the cells in a series string to prevent excess aging and unsafe operating conditions [1][2]. In lithium-ion battery packs, traditionally, and almost exclusively in the world at large today, balancing is achieved by switching a resistor in parallel with high cells to bleed energy out of them and dissipate it as heat. This tactic is not only 100 percent inefficient, but it is slow due to power dissipation restrictions typically thermally imposed within a system, and it is only effective at preventing over-voltage conditions. This thesis serves to outline the design, simulation, and implementation of an active battery management system for deployment in an electric race car, such as to minimize battery requirements (and therefore weight), optimize aging, and maximize the usable capacity of a given battery pack. High level simulations were performed from a pure energy transfer perspective to assess the potential of such a system, showing a strong inclination towards the potential effectiveness of the system. Given the unique constraints imposed on the isolated DC/DC converters used to transfer energy within the battery string, low level simulations were performed to validate the feasibility from a hardware perspective. Hardware that supports up to 2.5A balance currents was designed, constructed, and objectively validated against the simulations. An embedded operating system was developed, with a modular hardware abstraction layer, a flexible scheduler, and many support functions, on which an active battery management state machine was built. Lastly, tests were performed in order to compare real world results to high and low level simulation results. Findings have shown that active balancing is not only more effective at bleeding off high cells during recharge imbalance, but balancing can be performed throughout the entire cycle to supplement weak cells in order to extend the capacity of the battery as well as to offset the aging rate towards the stronger cells.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1 Lithium Ion Battery Parameters and Behavior

Before diving in to the whys and hows of active battery management, it is necessary to cover some of the basics of lithium-ion batteries and their relevant terminology. Below is a list of some of the most relevant terms and their functional description as it pertains to high level system design, as well as few key points regarding degradation in lithium-ion cells.

Capacity: Capacity is nothing more than a metric used to communicate how much energy is stored within the cell. Capacity is represented using the units of 'Ah', representing the current-time capabilities of the battery. A 1Ah battery will last approximately 1 hour with a 1A continuous discharge rate. The same cell would only last one tenth of an hour at a 10A discharge rate. Obviously then, higher capacity cells have more energy stored, but it is very important to understand that higher energy stored does not directly equate to the cell being capable of supplying the energy faster.

State of Charge: Most often abbreviated to SOC, state of charge is used to represent the percent of energy remaining in a given battery. Numerically, it is a normalized representation of the capacity remaining divided by the total full charge capacity. It provides a convenient means of describing a battery's capacity independent of the actual capacity of the battery in question. For example, a battery with a full charge capacity of 6Ah that has had 3Ah removed would be at 50 percent SOC.

Similarly, a 10Ah battery with 8Ah removed would be at 20 percent SOC.

$$SOC = \frac{Ah_{Remaining}}{Ah_{Total}} \tag{1.1}$$

Power Density: Sometimes known as specific power when calculated with mass, power density is an absolute metric that is used to show how much relative power a specific cell can provide as a function of it's volume. This is a key metric of study when sourcing batteries for a high pulsed power applications, such as an electric race car. Typically speaking, batteries are not as power dense as some other storage elements, such as super-capacitors or ultra-capacitors, but recent developments in high power capable lithium-ion chemistries have began closing this gap. The equation for power density is shown in 1.2

$$P_{Density} = \frac{A_{Max} * V_{Nominal}}{Volume} \tag{1.2}$$

Energy Density: Sometimes known as specific energy when calculated with mass, the energy density of a battery is a numerical representation of the amount of energy stored in relation to it's volume. Energy density is another specially useful metric for comparison in a sizing study of an electric race car battery. Maximizing energy density means you are maximizing the amount of energy stored on the vehicle while keeping weight to a minimum. The typical order of operation for a sizing study is to narrow down your choices based on power requirements, and them finalize based on energy density, though other things such as cell geometry and cooling considerations should also take a role. The equation for energy density is shown in 1.3.

$$E_{Density} = \frac{Ah_{Nominal} * V_{Nominal}}{Volume} \tag{1.3}$$

Equivalent Series Resistance: Commonly referred to as 'ESR', it is a single lumped parameter used to simply describe the electrical and thermal behavior of the battery. It is the effective operational resistance of the battery, responsible for the voltage rise and drop under load as well as the joule heating of the cell as a function of current. Typically, ESR is measured and specified by manufacturers by superimposing a small (5mV-20mV) 1kHz sinusoidal voltage on the open circuit (at rest) potential of the battery and measuring the subsequent current, and then using the applied voltage and current waveforms to calculate an average resistance. This is referred to as the 1kHz ESR. A more practical value for the ESR however is the DC ESR, which is also sometimes included in data sheets, and depicts the actual resistance of the cell in DC operational conditions. For example, if a designer knows they are going to be discharging the cell at approximately 1A constant current, and the cell has an internal DC ESR of 100m , the power dissipated in the cell would be 100mW. They also know that the voltage drop within the cell would be 100mV. DC ESR is typically characterized by performing two full discharges of the cell, one at a rate low enough to cause practically no joule heating, and one at a nominal rate from the data sheet, and dividing the average voltage difference by the average current difference, as can be seen in Equation 1.4.

$$ESR_{DC} = \frac{\overline{V_{diff}}}{\overline{I_{diff}}} \tag{1.4}$$

C Rate: C rate is a unitless parameter used to communicate a current as a function of a particular cell's capacity (Equation 1.5. For example, if someone told you to discharge a 1Ah cell at 1C, you would discharge it at 1A, and the discharge would be complete in approximately 1 hour. If they said to discharge the same cell at 60C, you would discharge it at 60A, and the discharge would complete in approximately 1 minute. This allows manufacturers to easily specify maximum and minimum parameters of their cells in terms that scale with the pack size, and allows designers to immediately gauge stress levels placed on a cell without having to look up the exact particulars of that cell. For example, an engineer should immediately know that a 1C discharge for a lithium-ion cell is relatively relaxed, while a 10C discharge is a stressful operating scenario.

$$CRate = \frac{A_{Applied}}{Capacity} \tag{1.5}$$

Capacity Fade: Commonly referred to as 'aging' in lithium-ion cells, capacity fade is the main method of degradation in lithium-ion cells. For lithium-ion cells, "end of life" (EOL) is usually classified at a 20 percent reduction in capacity. Capacity of any given cell is reduced over time due to many different mechanisms: age, operating rate, and operating temperature. Age alone is the most negligible capacity degradation factor in a lithium-ion cell, typically on the order of fractions of a percent of the cell's capacity per year. This mechanism is typically ignored. Operational rate, however, can be a very large part of the degradation of a cell as time progresses. Higher operational C-rates will degrade cells much faster than lower C-rates, and degradation rate typically increases exponentially with operational C-rate. For example, a cell rated for 10,000 cycles per the manufacturer could be expected to operate to the full 10,000 cycles if operated at 1C or less, but if operated at 10C, it would be unreasonable to expect it to operate much past 100 cycles, unless the chemistry is specifically engineered for operating at very high operational rates. Temperature is also a very prominent degradation mechanism that designers should consider, though it is a bit of a 'dark art' to predict the behavior at a given temperature. As a rule of thumb, operating a cell in ambient conditions at or below half of the manufacturers recommended maximum operating temperature will yield negligible temperature dependent aging over the life of the cell. However, even just storing a lithium-ion cell at ambient temperatures at or near the cell's recommended maximum operating temperature will severely impact the capacity of the cell. In fact, tests have been performed at UTA that show that just four weeks of storage at 70°C for a commercial off-the-shelf (COTS) Lithium-Iron Phosphate cell rated for a maximum operating temperature of 70°C will bring the cell to EOL, as shown in Figure 1.1. This will vary from cell to cell, but the overall effect is still largely dominant.

ESR Growth: ESR growth is another aging mechanism that changes battery behavior as a function of the same influential parameters as capacity fade. The main difference is that capacity fade in and of itself majorly impacts the functionality of the

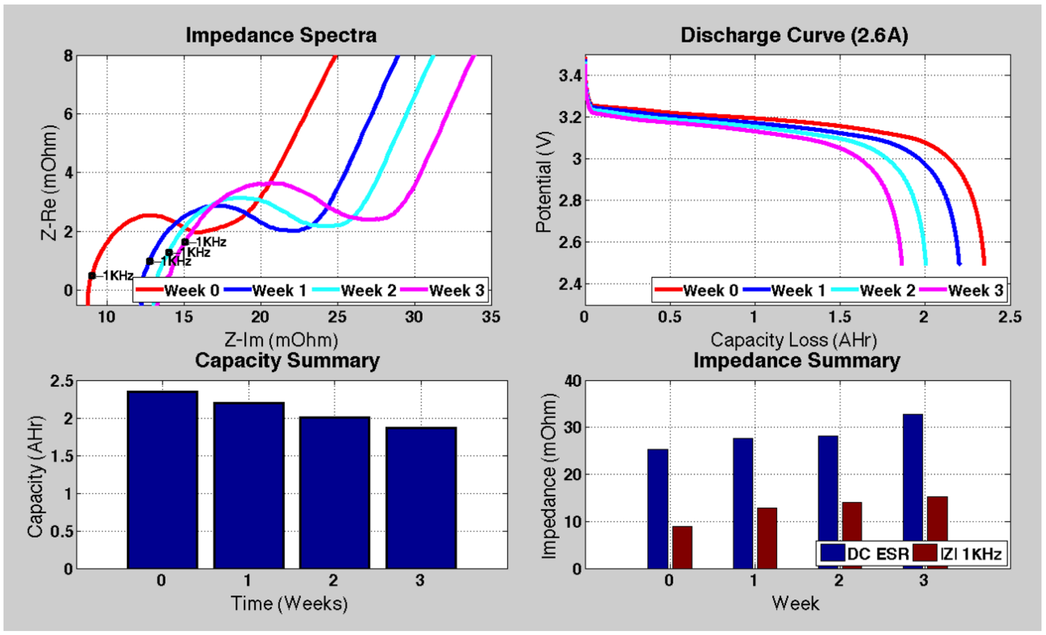


Figure 1.1: Example of the effects of storage temperature on capacity, ESR, and frequency response of lithium-ion cells stored at 70°C

cell within the target system, whereas growth of the ESR typically does not directly influence the operation of the system. The main detriment to growth of the cell's ESR is an increase in joule heating, which in turn further increases the capacity fade due to ever increasing operating temperatures. Generally, a system designer does not need to consider growth of the ESR as a function of time, unless the system is operated at extremely high rates, such that the voltage drop across the ESR is significant enough to further reduce the effective capacity due to premature voltage cutoff.

1.2 Electric Racecar Battery Requirements

For the purpose of this thesis, a battery management system specific to an electric race car, which will compete in the 2017 Formula SAE electric car competition, will be designed. This is an important consideration, because one of the main sources of constraints for such a car is the current year rules, and these rules change frequently. In the 2016 rules, the sections relevant to the design of the battery management system are: EV1.1.1, EV1.1.2, EV1.2.7, EV1.2.8, EV3.6.2, EV3.6.3, EV3.6.4, EV3.6.5, and EV3.6.6 [3].

EV1.1.1 and EV1.1.2: These rules imply two things, namely that when the high voltage system (tractive system) is deactivated, the maximum voltage permitted anywhere on the vehicle is 60V DC, and that when high voltage is active, the maximum voltage permitted anywhere on the vehicle is 300V DC. In other words, the maximum battery voltage for the car is 300V DC, and it must be able to be broken down into 60V segments. Assuming that the full charge cell voltage is either 3.6V or 4.2V (depending on the exact chemistry), the segments are limited to no more than 16 or 14 cells in series respectively. In order to design a universally applicable system, it will be assumed that the maximum number of cells in a given segment will be 10, limiting segment voltage to no more than 36V or 42V respectively, both of which are well within the rules.

EV1.2.7 and EV3.6.5: These rules state that the low voltage and high voltage system must be completely galvanically isolated. In the case of the battery management system, both systems will be present on the same board, so it is imperative that the design and PCB layout account for complete galvanic isolation (with at least 3/8” gap or a rated component) between the two systems. A previous BMS design at UTA has already proven the feasibility of such a configuration, so for the purpose of this thesis, only the operational functionality will be considered. Final layout will occur in the future when the next team finalizes a battery pack design.

EV1.2.8: This rule states that regardless of their location within the high voltage system, all components in the high voltage system must be rated to the full voltage. In other words, each BMS board must be designed to operate on either the bottom (most negative) segment, or on the top (most positive segment), without failure of the galvanic isolation. It is important here to remember that, per the FSAE organization, the main intent of the rules is to portray intent. Following explicitly, this rule requires that every single component on each and every one of the BMS boards must be rated to over 200V, but that is not the intention. The rule is written as though the team will be buying some form of isolated BMS system, so it is therefore only a requirement that the BMS system as a whole be designed to operate at the full voltage of the high voltage system, and not each and every component within.

EV3.6.2, EV3.6.3 and EV3.6.6: These rules state that the battery management system (BMS) is also responsible for thermal management of the battery cells. In order to prevent an extremely dangerous event where the cells are critically over their maximum temperature rating, it is necessary to monitor the temperature of the hottest part of the cells. If an over temperature occurs, it is the responsibility of the BMS to open the main relays and isolate the battery pack from the rest of the system. Per these rules, the BMS must monitor at least 30 percent of the cells, and the measurement points must be either on the negative terminal (typically, but not exclusively, the hottest part of a given cell due to the thermal resistance resulting from the construction method of the cells) or on the negative bus bar of a given cell, within 10mm of the terminal. Per the rules then, and given the design decision to design around 10-cell segments, there needs to be no less than three temperature sensor inputs per segment in the final system. Temperature measurements are nothing more than voltage measurements of a buffered and possibly amplified/conditioned temperature sensor, so for the purpose of this thesis, temperature measurements will be left out. It will be the responsibility of the future team that is implementing this BMS to account for the additional analog inputs, though the final firmware will have support for it.

EV3.6.4: This rule states that, for any BMS where a single board is to be measuring and managing more than 1 cell, all of the inputs from the battery segment must be protected by either a fuse or a fusible link wire. Per the rule, fusing must occur either in the conductor, wire, or PCB trace that is directly connected to the cell. In other words, if there are connectors, distribution blocks, or any other kinds of nodes between the cell connection and the BMS connection, fusing must occur before the intermediate node. For the purpose of this design, it will be assumed that, just as in the past, the BMS will directly interface with the battery bus bars via spring loaded terminals. The BMS board will mount directly to the top of the segment, and will have contacts protruding from the bottom of the board that sit on the battery bus bars directly. Fusing then will occur on the BMS board via properly rated SMD fuses. The fuses will be specified such that they open below the maximum rating of any of the traces or discrete components in the circuit.

Once the rules have been addressed, and a plan put in place for conformance, it is necessary to address the other constraints of the system's design that are imposed by specifics of operating in a race car environment. Many other factors will influence the design decisions of the BMS, such as peak balance current required to be effective at the expected operating rates, required battery capacity for a given event or series of events, and additional weight added via BMS components vs energy storage weight savings. While these may seem like very complex considerations, they are actually relatively simple to estimate based on previously calculated data, and can only be optimized via data collection once implemented into an actual vehicle due to the nature of the calculated data. For starters, required balance current and required capacity have been previously calculated by the UTA racing team using track data from a previous year's combustion car. Their findings indicate that the average current expected from the battery packs as a whole will be approximately 81A, which during a 20 lap endurance event with 60s average lap times, corresponds to 25Ah of battery capacity required, as shown in Equation 1.6.

$$CapacityReq = I_{avg} * RunTime = 81A * \frac{20Laps * 60sec}{3600} = 25Ah \tag{1.6}$$

Obviously this assumes many things, first and foremost that the weight of the car as well as the car's dynamics will be close enough to the combustion car's that the throttle input will be directly relatable. However, it is expected that, due to the increased weight of the electric car, the average current will be higher than this. Lithium-ion batteries typically experience no more than a 1 percent capacity and/or ESR discrepancy for cells manufactured within the same batch, so it is reasonable to design the maximum balance current such that it can compensate for a 1 percent imbalance in capacity throughout a cycle. If we assume a worst case scenario average battery current of 100A, that would correspond to a maximum average balance current of only 1A at the 1 percent mark. Average currents much beyond 2.5A would likely require much heavier and/or robust components within the BMS, so it is desirable to minimize the target average balance current as much as possible, but the weight/complexity cost up to the 2.5A point is relatively constant, so a maximum balance current of 2.5A was chosen. This means that, at the 1 percent discrepancy mark, the BMS could support up to a 250A system, and that at the worst case scenario average current of 100A, the BMS can account for up to a 2.5 percent discrepancy in the cells. It is important to realize that this weight/complexity cost is primarily driven by the selection of available isolated flyback transformers and ultra-low gate threshold MOSFETs, and that with a high volume custom solution, scaling would be much more linear.

The last thing to consider goes hand in hand with what has been previously stated regarding cell discrepancies, though it is still important to discuss because it highlights the true beauty of an active BMS that can not only balance during charge, but can also balance during discharge. Assume that as an absolute worst case scenario, you have a two cell string of batteries, and one battery is a 10Ah cell while the other is a 5Ah cell. Without balancing, the pack as a whole would only be a 5Ah pack, because they would never reach the limits at the same time due to the lower capacity cell always dictating the lower and upper limits. Also note that a traditional balancing system could easily get the cells in unison at 100 percent state of charge, but as soon as the discharge begins, the cells will separate and the charge-only balance algorithm will not do anything to compensate. This also means that both cells will always see the same aging due to current, but the weaker cell will see much greater aging due to the depth of discharge it is seeing as compared to the second. In an active BMS system, as soon as discharge begins, and the cells begin to separate, the BMS will begin moving energy from the higher capacity cell to the lower capacity cell. This does two things: it offsets the effective current between the two cells, and it alters the effective capacity of each cell. The higher capacity cell will see a higher average current, and therefore will age faster as a function of discharge current, and it will also see a higher depth of discharge as its energy is moved into the weaker cell, also increasing the aging effects. The opposite is true for the weaker cell, and the process should continue until the cells are matched, though in this extreme example, that would correspond to over a 50 percent reduction of capacity for the higher cell, and it would be deemed dead. An active BMS then becomes an extremely valuable tool in maintaining battery health!

1.3 Existing Systems and Topologies and their Short Comings

Many battery management system topologies exist which all accomplish the same thing: passive balancing via a pass transistor and/or a bleeding resistor. These range anywhere from simple ‘dumb’ zener diode biased transistor bleed circuits, to more complex comparator driven over-voltage detectors that bleed energy using a MOSFET and dissipation resistor, some of which are exampled in Figure 1.2. While some of these, like the zener biased pass transistor, do have some dynamic response qualities in that the balance current becomes proportionate to the amount of over-voltage, none of the all analog systems are smart enough to tailor the balance current to the level of imbalance among the cells. At best, a current proportional to the voltage over the threshold can be produced, but this is still a static threshold that does not move. Even if the cells are perfectly balanced once the threshold is reached, all of the balancing circuits will be active, and the charger must be smart enough to account for this ‘extra’ current when performing the CC-CV charge algorithm and subsequent cut off.

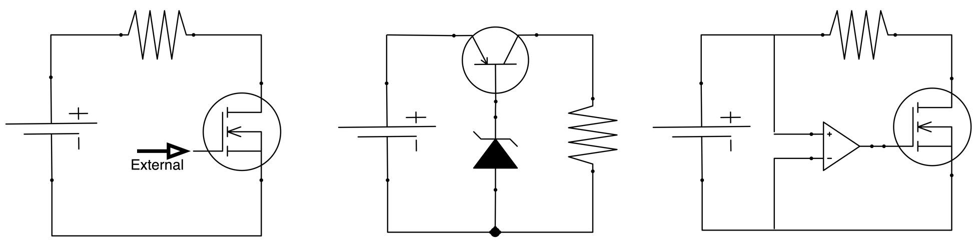


Figure 1.2: Example of passive balancing topologies. (Left) Bleed resistor with externally driven MOSFET. (Middle) Pass transistor biased with zener diode. (Right) Bleed resistor with locally driven MOSFET using comparator

The shortcomings of passive charge-only balancing schemes and topologies are no secret [4]. Discharge currents are typically orders of magnitude higher than recharge currents in high power systems, so most of the imbalance takes place during discharge where the balancing circuitry is not active. Even when the balance circuitry is active, for most passive designs, the balance current is binary: it's either on or off, and only varies proportionately with cell voltage. This means that higher levels of imbalance in some cells are treated no differently than lower levels of imbalance. It becomes obvious that an active system that can not only move energy from one cell to other cells in the pack, but can do so proportionately, both during charge and discharge, would be a much more efficient and effective means of maintaining balancing in the system. There is vast amounts of literature related to such active battery management schemes, ranging from capacitively coupled charge shuttling systems [5], to coupled inductor hybrid DC-DC converter systems [6]. Several of these topologies are exampled in Figure 1.3.

The inductor based boost mode balancer uses a boost converter topology to charge the inductor using the bottom switch, and the body diode in the top switch allows for charge to transfer to the top cell once the bottom switch is turned off.

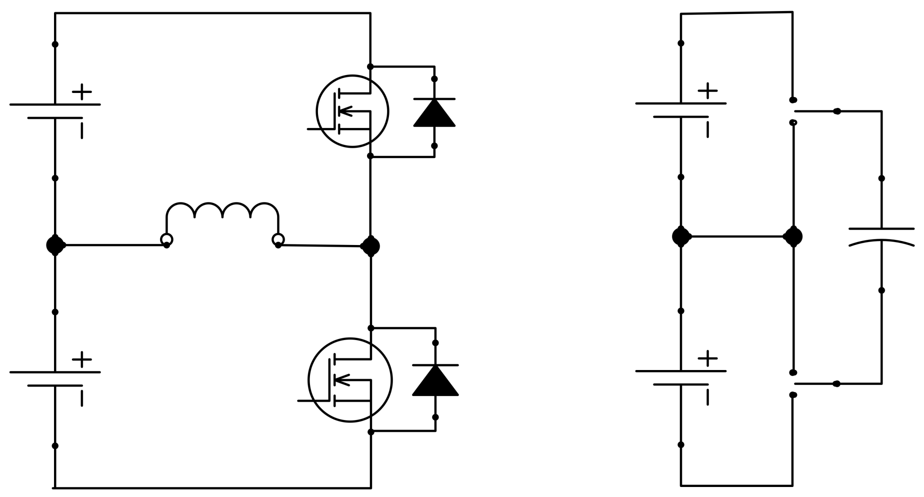


Figure 1.3: Example of active balancing topologies. (Left) Inductor based boost mode active balancing. (Right) Capacitor based charge shuttle active balancing

While this topology is very convenient in that parts count is low, and the energy transfer rate can be very high, this is a very limited scheme. Energy can only be shuttled one cell at a time, and the system becomes much more complex when the topology is extended for bidirectional energy flow [7]

The capacitor based charge shuttling topology, while extremely simple to implement, and arguably the most efficient of the active topologies, is also severely limited. Balancing is achieved by operating all of the switches simultaneously, each either in the high position or the low position. Cells that are higher than the capacitor voltage when switched will charge the capacitor, and cells that are lower than the capacitor when switched will discharge the capacitor, effectively moving charge from higher cells to lower cells until perfect balance is achieved. Balancing therefore is inherently uncontrollable, in that every cell in the stack is operated on identically and balancing is either on or off, and the energy transfer rate is directly proportional to the size of the capacitors and the switching frequency [8][9]. It is also increasingly less effective as the voltage on the cells equalize, and energy transfer is extremely "elastic" in that it is very slow to moved charge from many cells into a single weak cell.

There are many other active topologies as well, and each has its shortcomings, whether it be the power limitations in capacitor based systems, to the complexity and weight limitations in hybrid DC-DC converter based systems. Even systems specifically synthesized to minimize parts count, such as the single switch Sepic topology [10] come with the trade-offs of relatively slow balancing and very complex control schemes. One system stands out though: the isolated flyback system [11].

1.4 FlyBMS Overview

This thesis then focuses on the design and implementation of an isolated flyback based active battery management system, with a novel proportional balance control scheme unique to this design. The isolated flyback based active balance system is inherently isolated, inherently modular, requires no custom components, and can easily be operated in open loop mode (though some protection was found to be paramount in overall robustness, as will be discussed later). A simple flyback based balancer is exampled in Figure 1.4.

In this system, the transformer primaries are all tied directly to each cell, and the secondaries are all in parallel across the entire battery stack. Switches on the primary side (MOSFETs) charge the transformer, and switches on the secondary side (diodes) rectify the transformer output. Energy taken out of a single cell is boosted to the full pack voltage and dumped back into the entire string. Using a properly

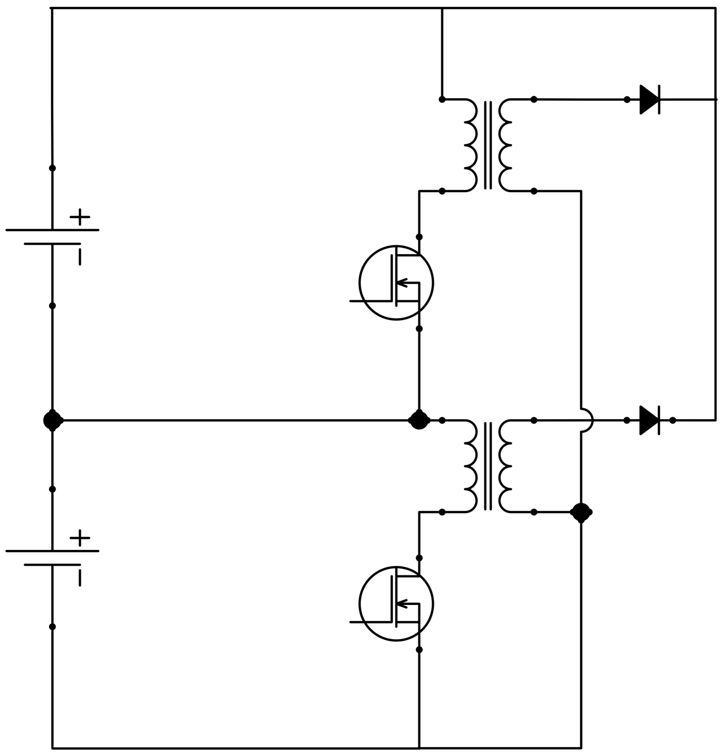


Figure 1.4: Example of an active flyback based balancing topology

designed control system, energy can be taken from a specific subset of cell(s) and moved into other specific subsets of cell(s), giving the designer complete freedom to manage the state of charge of every cell in the pack all the while dissipating orders of magnitude less energy as heat than a typical passive bleed system. One other key aspect of such a system is that the duty cycle of the driving signal can be varied in order to vary the balance current, so the designer not only has the freedom to move energy from and to anywhere in the stack, they can do so at virtually any rate, giving them the ability to target specific corrections. A system can be configured to only target minor voltage imbalance, voltage imbalance coupled with state of charge imbalance with tracking, and even a mode where overall health is not considered unless a cell has become a major limiting factor in overall pack capacity.

CHAPTER 2

MODELING AND SIMULATION

2.1 SOC Based Active Balancing MATLAB Simulation

In order to study the feasibility of an active balance system that can move energy from a single cell into the entire stack, a high level energy transfer simulation was developed using MATLAB. The goal of the simulation was to study the electrical behavior of the cells from a strict energy transfer point of view, as well as to early identify any potential pitfalls with such a balance algorithm. The simulation is built around a 10-cell battery pack, has the ability to cycle the pack under a constant current profile or a constant power profile, uses a lookup table built from a lithium-ion charge and discharge curve, and can be run with or without the balancing algorithm. The simulation was built on an integration loop with a 10ms timing, with a three operation state calculation set per iteration of the simulation. The inputs to the simulation are initializations for cell voltage, cell stage of charge, cell balance duty cycle, cell ESR, and external current set point.

The first loop operation is to calculate the net change in overall pack capacity. For this operation, external current is integrated over the loop timing to yield a net capacity change for the pack as a whole, which is then applied to the state of charge of each individual cell's SOC tracking vector (Equation 2.1). For this step, external current is either represented by a constant in the case of a constant current charge/discharge mode, or by a current dictated by the total pack voltage and the power set point in the case of constant power charge/discharge mode.

$$SOC(i+1) = SOC(i) + \frac{I_{ext} * T_{step}}{3600} \tag{2.1}$$

The second operation calculates net changes in each cell's individual SOC as a function of the balance current for each individual channel, where balance current is defined as the isolated current being sourced out of an individual cell. Since each cell has an individual flyback converter capable of up to 2.5A, the maximum balance current was set to a constant 2.5A. Each converter's duty cycle is also variable, and can operate at any current between 0A and 2.5A, so it was necessary to form a function that could determine the duty cycle of each cell's converter based on some parameter, and then convert that to a balance current. It was decided that, since the lowest cell will always be the limiting factor during charge and discharge, the balance current for a given cell should be proportionate to the voltage difference between said cell and the lowest cell in the pack. Equation 2.2 shows this relationship, and also serves to show that the balance current for the lowest voltage cell in the pack will always be zero. Duty cycle is then translated to current as a function of max balance current, as shown in Equation 2.3.

$$Duty(i, cellnum) = getDuty(Voltage(i, cellnum), Vmin(i)) \tag{2.2}$$

$$CellCurrent = (-1 * Duty(i, cellnum) * MaxBalanceCurrent) \tag{2.3}$$

The translation from cell voltage discrepancy to duty cycle is a very dynamic relationship that depends on many things, such as voltage resolution in the final system, the noise floor in the final system, and hysteresis effects due to instantaneous changes in cell voltage due to the ESR. As will be discussed later, in order to make the system truly universal, balance currents must be applied “softly” when the voltage discrepancies are small, because large jumps in balance current will result in large drops across the cell’s ESR, which will then cause race conditions (software oscillatory condition where previous decision directly impact the outcome of subsequent decision, independent of the actual system) in the balance algorithm and subsequent oscillations in the BMS state machine. However, it is also necessary to ramp the balance current quickly as the voltages deviate further in order to be able to react to real changes in the cell voltages should one cell begin separating quickly from the average voltage. For these reason, it was decided to use an exponentially increasing duty cycle, that ramps balance current exponentially as a cell’s voltage descrepancy climbs. In order to prevent 10ms oscillations in the applied duty cycle and subsequent balance current, a simple exponential mapping was used to map voltage deviations to duty cycles, as depicted in Table 2.1. The final form of this table was narrowed down after many tests were performed within the simulation, and further refined in hardware during the hardware testing phase. The mapping function also accounts for simple limiting, where duty cycle is set to zero in the event that a cell is completely dead in order to prevent cell damage.

The final operation in the simulation loop is to relate the newly calculated SOC vector to a new voltage for each cell, to be used in the next iteration of the simulation

Table 2.1: Mapping of voltage discrepancy ranges to applied duty cycle for each channel's flyback converter

Range1 = .001V	Duty1 = .05
Range2 = .0015V	Duty2 = .11
Range3 = .002V	Duty3 = .18;
Range4 = .0025V	Duty4 = .25;
Range5 = .003V	Duty5 = .32;
Range6 = .0035V	Duty6 = .39;
Range7 = .004V	Duty7 = .46;
Range8 = .0045V	Duty8 = .53;
Range9 = .005V	Duty9 = .6;
Range10 = .006V	Duty10 = .67;
Range11 = .008V	Duty11 = .74;
Range12 = .010V	Duty12 = .81;
Range13 = .013V	Duty13 = .88;
Range14 = .016V	Duty14 = .95;
Range15 = .020V	Duty15 = 1;

loop. The lookup table for this SOC relationship was formed by performing a very slow C/10 discharge and charge of an existing lithium-ion cell, from 100-0 percent SOC and 0-100 percent SOC respectively. The extremely low rate was chosen in order to obtain curves that are effectively independent of voltage variations due to the cell's ESR. The functions that perform these mappings are shown in Equations 2.5 and 2.7 respectively.

$$index = find(CHARGESOC(:,2) == SOC, 1); \tag{2.4}$$

$$Voltage = CHARGESOC(index, 1); \tag{2.5}$$

$$index = find(DISCHARGESOC(:,2) == SOC, 1); \tag{2.6}$$

$$Voltage = DISCHARGESOC(length(DISCHARGESOC) - index, 1); \tag{2.7}$$

The 10ms simulation loop runs continuously over a predefined period of time. Charge and discharge are defined simply by the cell voltage and SOC boundaries. If any cell reaches its maximum voltage or 100 percent SOC during the simulation, the external current applied becomes negative to represent a discharge, and if any cell reaches its minimum voltage or 0 percent SOC during the simulation, the external current applied becomes positive to represent a charge. The simulation also contains an aging parameter that linearly increases the ESR of the cells by a predefined amount to simulate aging within the cells. This scaling (shown in Equation 2.8) is proportionate not only to the number of cycles, but also by the external current applied, and is applied upon completion of every complete cycle of the pack, and is useful for observing BMS behavior over not only short term voltage discrepancies due to individual differences, but over long term discrepancies that become more prominent over time as the cells age and the voltage developed across the ESR during cycling grows.

$$ESR = ESR * I_{external} * AgeFactor \tag{2.8}$$

The high level simulation results were paramount in fine tuning the overall functionality of the system. Beyond the fundamental validation of feasibility of an active system, many parameters were optimized to improve overall performance, as well as to ensure extensibility of the system to any power level and/or lithium chemistry. Parameters considered during this testing were: control bandwidth of the BMS state machine, effects of different chemistries and their respective voltage curves, mapping of the voltage differentials to balance currents, and the effects of varying ESRs both statically as well as over time.

To start with, overall feasibility of the active system was validated. Control bandwidth was set to 10Hz (100ms), aging parameters were set to zero, duty cycle was mapped linearly to voltage imbalance, battery capacity was set to 6Ah, and the simulation was set to run in a constant current cycle mode with a set point of 6A (1C). All cells except for cell 10 were initialized with a matched ESR of 10mΩ, and all but cell 10 were initialized to 100 percent SOC. Cell 10 was initialized to 50 percent SOC and 15mΩ in order to force a full balance current initially as well as to force a permanent cell voltage discrepancy, such that BMS behavior could be easily validated both against capacity imbalance, as well as against ESR mismatch. After much trial and error with functionality of the simulation, a 10 hour simulation was run, both with and without active balancing. As is shown in Figure 2.1, the active system has proven to be functional. In the top plot, the battery pack is cycled within the limitations of the cells. The capacity window (window of usable capacity due to SOC imbalance in the pack) is equal to 3Ah due to the fact that one of the cells was initially at 50 percent SOC, and without balancing, the discrepancy is constant. In the bottom plot, however, the BMS corrects for the initial capacity discrepancy within the first two cycles, and the ESR mismatch is accounted for on a cycle-by-cycle basis to maximize the usable capacity.

The next step towards utilizing the simulations for system tuning was to tune the balance current (duty cycle) mapping and the control bandwidth of the BMS state machine. During initial testing of the simulation, it was found that with lower

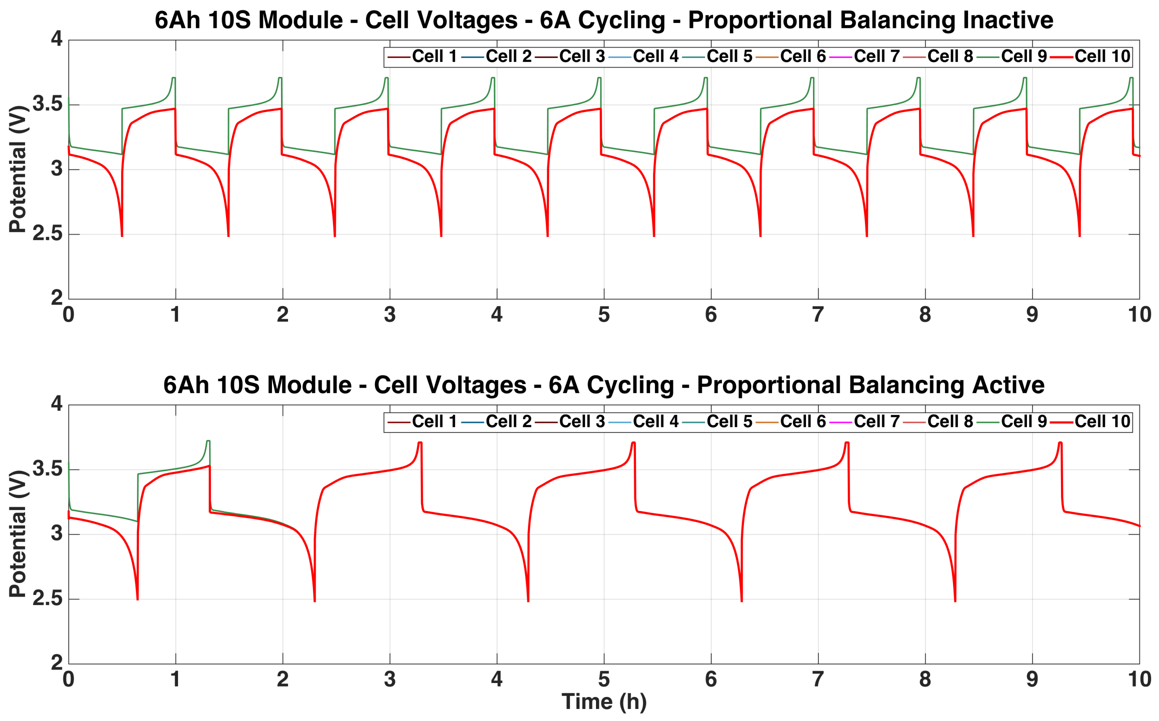


Figure 2.1: MATLAB simulation result that validates the active flyback BMS system.

power cells (ESR $> 20\text{m}\Omega$) if the loop timing was too low, or the balance current was too high at small voltage discrepancies, the BMS state machine would enter an oscillatory condition. Every iteration of the state machine would toggle the balance duty cycle between 0 percent and 100 percent. To counteract this, two changes were made. The first change was to redefine the duty cycle mapping from a linear function to an exponential function. Initially, the duty of each cell was increased by 10 percent (150mA) for every 5mV that particular cell was above the lowest cell. This meant that for every 5mV of voltage discrepancy, an additional 150mA of load current was placed on a given cell by the balance circuitry, which would then cause additional voltage drop across that cell's ESR, which explained why it was more prominent on higher ESR cells. To counteract this, duty cycle mapping was reconfigured to increase exponentially with the voltage discrepancy, such that small imbalances only resulted in very small balance currents, but large imbalances could still be reacted with full balance current. This duty cycle remapping, accompanied by a slower 4Hz control bandwidth that allowed the battery's natural voltage hysteresis to settle once balance current was enabled, made the system much more robust and tolerant to high ESR cells. Figures 2.2 and 2.3 example these changes and their effects respectively.

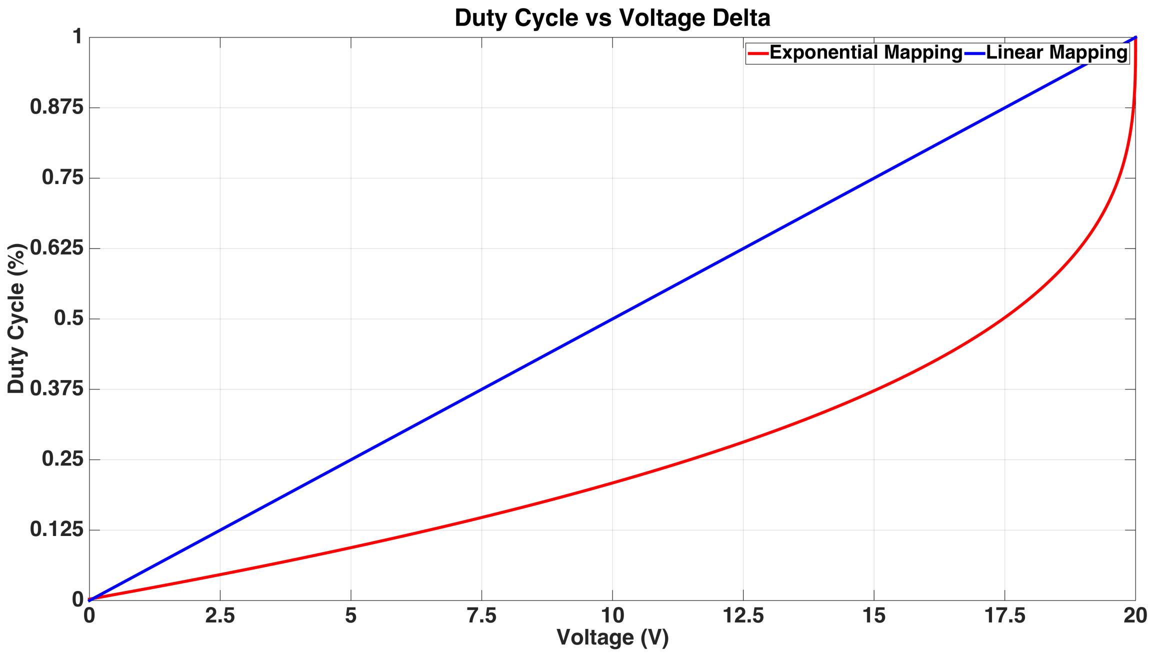


Figure 2.2: Mapping of voltage imbalance to applied duty cycle for each cell

The last parameter that was evaluated with the simulation was chemistry, ESR, capacity, and cycle rate independence. Many different lithium-ion chemistries exist, and each has a unique voltage curve. For testing purposes, I chose to evaluate curves from the Lithium Iron Phosphate (LFP) chemistry and the and the Lithium Manganese (IMR) chemistries respectively, because they have the most contrasting curve

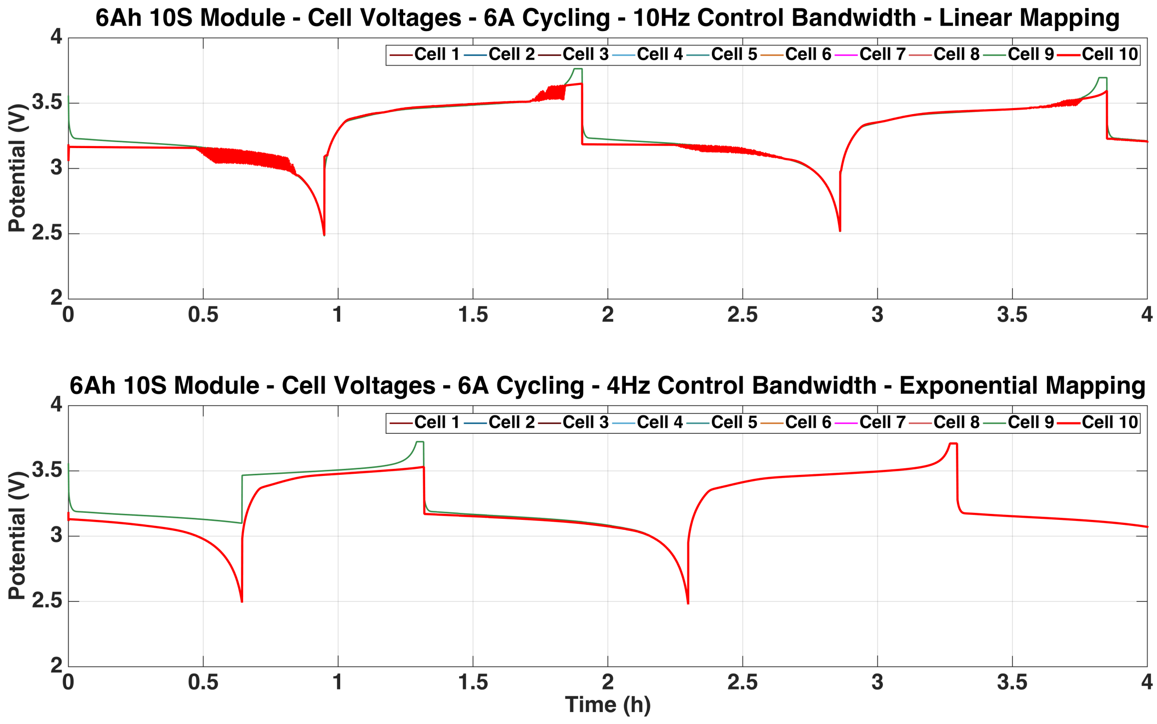


Figure 2.3: MATLAB simulation result showing oscillations with original loop timing and duty cycle mapping

of any chemistry. LFP has the flattest voltage curve of any chemistry, only slightly varying in voltage in the 80-20 percent SOC range, while IMR has a very linear voltage curve in the same range. Testing was performed by simply discharging one of each of these chemistries at a low rate, building a normalized voltage vs SOC charge and discharge lookup table for each, and redefining the look up tables for battery voltage within the simulation. It was quickly found that, for the more linear IMR chemistry, if the balance current was not applied softly initially, oscillations similar to the high bandwidth oscillations would occur. Changing the exponential duty cycle mapping from a 2nd order exponential to a 4th order exponential in order to further "soften" the activations of the balance system almost completely remedied this, and the remaining instability will be addressed in the final embedded system should they arise. Results from a sweep of capacities, cycle rates, and cell ESR are shown in Figure 2.4, and shows that the BMS operation is not compromised across a host of various cell parameters. The next step in system design was to design and construct the flyback balancer stage.

2.2 Simple mathematical model of a flyback converter

The flyback topology is a simple topology, with relatively straight forward solutions in both continuous conduction mode (CCM - transformer primary current never goes to zero) as well as discontinuous conduction mode (DCM - transformer allowed to fully desaturate between switching). Closed loop control of a flyback converter, however, can be much more challenging. The designer must consider not only the means of which to measure output voltage and/or current in an isolated manner, but the control loops have to also consider transformer saturation for robust performance. For the sake of simplicity, an open loop control mode will be implemented, and the transformer will be operated in DCM in order to avoid saturation of primary inductance of the transformer. In other words, rather than measuring output voltage and/or current on a cycle by cycle basis and compensating for the error by adjusting the duty cycle, output current will be mapped in software to a specific duty cycle at a specific input voltage.

Figure 2.5 shows the basic topology that will be implemented. There are many variants of the flyback topology, and many great resources available regarding their design. The Art of Electronics [12] is an example of a great reference material for any engineer with a strong background in electronics, and it was the main reference for the flyback design presented here. In the figure, CIN represents the input filter capacity of the flyback stage, LL, LP, and LS represent the leakage, primary, and secondary inductance of the transformer respectively, D1 and DZ make up the clamp circuit that will protect the MOSFET, M1 represent the MOSFET that is the main switch, D2 represent that output rectification diode, and COUT represents the output filter. The required specifications for theses components as well as many other relevant parameters are solved for next, followed by explanations for each component choice in the system. The first step towards solving for the parameters of any system is to identify your constraints as well as your constants.

In order to simplify the design of the flyback balancing stage, a decision was made to start by choosing a transformer, since there were not many options available for small yet relatively high power density transformers with a gain of at least 4. Most of the part selection justification will be in a later section, but the transformer characteristics drive the overall design, so it had to be sourced first. The transformers chosen were small transformers explicitly designed by Coilcraft for use in flyback converters, with a maximum primary current of 7A and a 1:4 turns ratio. The specifications given in Equations 2.9 to 2.14 summarize the key parameters of this transformer that will be used in the design. The switching frequency and voltage ranges are also defined.

Transformer Parameters

$$L_P = 4\mu H \tag{2.9}$$

$$L_L = 116nH \tag{2.10}$$

$$L_S = 64\mu H \tag{2.11}$$

$$R_P = 12m\Omega \tag{2.12}$$

$$R_S = 500m\Omega \tag{2.13}$$

$$I_{P,Max} = 6.5A \tag{2.14}$$

$$\tag{2.15}$$

System Constraints

$$F_{SW} = 80kHz \tag{2.16}$$

$$T_{SW} = \frac{1}{F_{SW}} = 12.5\mu s \tag{2.17}$$

$$V_{In,Min} = 2.5V \tag{2.18}$$

$$V_{IN,Max} = 4.2V \tag{2.19}$$

$$V_{Out,Min} = 25V \tag{2.20}$$

$$V_{Out,Max} = 42V \tag{2.21}$$

The primary side of the transformer is nothing more than an ordinary inductor, and when a voltage is applied across an inductor, current ramps until a saturation point is reached, at which point the current is DC. In order to maximize energy transfer while still operating in DCM, it is desired to turn the main switch on just long enough to allow the transformer primary to reach saturation, turn it off just long enough to allow the secondary current to reach zero, and then repeat. This approach allows all of the energy that is put into the primary side from an individual cell to be rectified on the secondary side and placed back into the pack, with the exception of any energy that is stored into the uncoupled leakage inductance which will be discussed later. This energy can be easily solved for, as well as the resulting average primary (balance) current at maximum duty cycle..

$$E_{In} = 0.5 * L_P * I_{P,Max}^2 = 84.5\mu J \tag{2.22}$$

$$I_{SW,Avg} = \frac{E_{In} * F_{SW}}{V_{In,Max}} = 1.61A \tag{2.23}$$

Since we also know that we want to operate in DCM, we know that the beginning of every switching period on the primary side will start at zero current. We also know that we never want to exceed the maximum current rating of the primary side of the transformer. These two constraints allow us to easily identify the maximum on time of the primary switch in order to avoid exceeding the ratings on the primary. It is important to recognize from Equation 2.24 that the maximum on time changes with input voltage. This also implies that, if we are going to maximize the switching frequency of the system in order to minimize ripple current, we must consider every scenario of input voltage to be sure that the transformer is always allowed sufficient time to desaturate. This is how the 80kHz switching frequency was chosen, because 12.5 μ s is the minimum switching period that allows for full saturation and desaturation of the primary side at all input voltages.

$$T_{On,Max} = L_P * \frac{dI_P}{V_{In}} = L_P * \frac{I_{P,Max} - 0}{V_{In}} \tag{2.24}$$

Using the same principle as was applied to determine the input (and subsequent output) energy, we can also calculate the energy stored in the leakage inductance. The leakage inductance is a parameter used to account for the inductance on either the primary or secondary side of the transformer that is not coupled across the transformer. It is best visualized exactly as it is shown in Figure 2.5 (LL), in series with the primary side of the transformer. The energy in this inductance will not be transferred to the secondary side, and the voltage developed across it when the current goes to zero will be apparent on both sides of the transformer, reflected by the turns ratio. To prevent this voltage from becoming the limiting factor in the primary switch and secondary diode part choice, it must be dissipated in a controlled manner. We will discuss how to specify the voltage of this clamping circuit (DZ and D1) later, but first we must determine how much energy will be stored in this inductance so that we can specify a power requirement for the snubber components. This can be done by scaling the total energy in by the ratio of leakage inductance to total primary inductance.

$$E_{LL} = E_{In} * \frac{L_L}{L_P} = 9.8\mu J \tag{2.25}$$

From this energy, we can now find the power requirements of the snubber diode (DZ). Power can be found by integrating the energy over time, which becomes very simple since this is a switching circuit. It is important to note here that this simple calculation considers the scenario where all of the inductive energy in the leakage is snubbed, but in practice we want to snub as little as possible for efficiency sake. As will be shown later, only roughly twenty percent of the leakage inductance energy will be dissipated to protect the MOSFET, so this calculation shows a worst case scenario.

$$P_{Dis,DZ} = E_{LL} * F_{SW} = 1.3W \tag{2.26}$$

Knowing how much energy is being put into the transformer, and how much will not transfer due to uncoupled leakage, we can then solve for how much energy will be transferred to the secondary side. We can use the same technique used to solve for input energy to relate input energy to output energy using the secondary inductance. It is important to realize that the leakage inductance only appears on one side of the transformer, so we do not need to account for it again on the secondary side. We can also use the secondary energy calculation to calculate the peak current of the rectification diode (D2) on the secondary side, which occurs when the pack voltage is at its minimum. This assumption only holds true due to the fact that we will be mapping duty cycle to input voltage such as to keep the average primary current, and therefore input energy, constant. From equations 2.27 and 2.28 we can see we will need a diode with at least 1A of peak current capability to have some factor of safety.

$$E_{Out} = E_{In} - E_{LL} = 88.1\mu J \tag{2.27}$$

$$I_{D2,Peak} = \frac{E_{Out} * F_{SW}}{V_{Out,Min}} = 493mA \tag{2.28}$$

The next step then is to solve for the voltages that will be seen at the primary and secondary side of the transformer during the on and off period of the switch. These calculations will not only allow us to specify the minimum required voltage ratings of the input switch and output diode, but we can also use it to specify a clamp voltage that will minimize the power dissipated in the clamp circuit while ensuring that the MOSFET remains protected. We will start with the voltages during the on period of the switch.

When the switch is on, the voltage across the primary is equal to the input voltage, which is reflected to the secondary side via the turns ratio. This appears as a negative voltage on the secondary side, which is why no current flows on the secondary side while the switch is on. This also means that the rectification diode on the secondary side is responsible for holding off the output voltage as well as the additional voltage developed across the secondary.

$$V_{S,Min} = -1 * V_{In,Max} * \frac{L_s}{L_P} = -15.75V \tag{2.29}$$

$$V_{D2,Max} = V_{Out,Max} - V_{S,Min} = 57.75V \tag{2.30}$$

When the switch is off, just as how the primary voltage was reflected to the secondary side while current was flowing in the primary, secondary voltage will be reflected to the primary side while current is flowing in the output rectification diode. We can simply reflect the peak secondary voltage back to the primary side.

$$V_{P,Max} = \frac{V_{D2,Max}}{\frac{L_P}{L_S}} = 11.2V \tag{2.31}$$

We can see then, that the switch needs to be able to survive a minimum of 11.2 V in order to survive the voltage reflected as a function of the normal flyback operation. However, we also need to account for the voltage developed across the leakage to be able to properly specify a MOSFET as well as a snubber circuit.

$$V_{Leak} = L_L * \frac{I_{P,Max}}{T_{SW} - T_{On,Max}} = 9.1V \tag{2.32}$$

This voltage, added to the input voltage and the reflected secondary voltage, is the peak voltage which the switch should be able to survive. The percentage of the leakage voltage which will be allowed to be present is also what will scale the power dissipation in the snubber circuit.

$$V_{DS,Peak} = V_{P,Max} + V_{Leak} + V_{In,Max} = 24.5V \tag{2.33}$$

From these equations, we can see than that the MOSFET drain-source voltage needs to be rated to no less than 25V. We also know that the MOSFET needs to be able to handle at least 6.5A to carry the peak drain current voltage. The last parameter that needs to be considered for the MOSFET is the gate threshold voltage. Because the MOSFET is on the primary side, and the control circuitry will be isolated completely from the battery, the MOSFET will be driven with an isolated driver circuit. The simplest way to accomplish this is with a complementary output (Push-Pull) optocoupler, which will pull the gate between 0V and cell voltage. This implies then, that the MOSFET must be capable of fully turning on at even the minimum cell voltage of 2.5V. Only a handful of MOSFETs exist that can carry greater than 6.5A, are rated for a drain voltage of 25V or greater, and have the drain resistance full specified at voltages as low as 2.5V. the only manufacturer with a full line of such MOSFETs is Vishay, so a suitable switch was chosen from their lineup. The driver topology will be discussed in full in a later section, but the drain voltage of the chosen switch is 40V, so the snubber should only be chosen such as to clamp the developed primary voltage to less than 40V. The clamp diode (DZ) was chosen to be a 21V TVS diode, that breaks down at approximately 28V and clamps at 38V, and is rated for 1 Watt of power dissipation, which together should be more than enough to protect the switch.

The last flyback parameter to consider is the minimum off time required to keep the converter in DCM. Since the flyback is going to operate in DCM, it is imperative to ensure that adequate time is allowed for the transformer secondary current to ramp to zero during the switch off time, such that all of the energy put into the primary side during the on time is rectified and transferred back into the battery during the off time.

$$T_{Off,Min} = L_S * \frac{I_{D2,Peak}}{V_{Out,Min}} = 1.3\mu s \tag{2.34}$$

This time, in conjunction with the maximum calculated on time, can then be used to determine the total time period required to reach saturation on the primary side and subsequently desaturate fully on the secondary side. This time will determine the maximum switching frequency, shown in Equation 2.36. An important thing to realize here is that the final design should incorporate some additional delay. Many parameters are involved in determining stability of switching nodes in switching circuits, and designers should always expect some oscillations at the switch node and allow adequate time to damp them, which is part of the rational for an increased switching period.

$$T_{SW,Min} = T_{On,Max} + T_{Off,Min} = 8.5\mu s \tag{2.35}$$

$$F_{SW,Max} = \frac{1}{T_{SW,Min}} = 120kHz \tag{2.36}$$

Regarding the design, one final consideration must be entertained. Typically in a switch mode power supply, where a flyback would typically be implemented, the output voltage is defined by the output of the converter. If the converter is off, the output voltage will drop to zero, allowing the transformer to fully desaturate. In this system, however, the output voltage can never fall below the total pack voltage, because the output is tied directly to the battery stack. Thus far, the design has been structured such that when the switch is turned on, the primary is charged, and when the switch is turned off, the secondary is discharged, and the snubber dissipated the excess energy in the leakage. In theory then, all of the transformer's stored energy is accounted for, allowing operation in DCM. However, because the output can only rectified while the secondary voltage is greater than the battery voltage, all of the voltage developed on the secondary cannot be rectified, and the leakage energy is not the only energy remaining in the transformer once the rectification diode stops conducting. From experience, it is expected that the remaining uncoupled energy will begin to oscillate between the primary and secondary, and will continue to do so until it is either dissipated or naturally damped. To prevent this excess energy from creating oscillations, it is necessary to provide a continuous conduction path on either the primary or the secondary. The simple approach is to place a resistor across the secondary, such that minimal energy is dissipated during output diode conduction, but the excess energy is dissipated during the additional off time facilitating complete desaturation of the transformer. This modified flyback topology is shown in Figure 2.6 with the additional dissipation resistor (R1). The actual value of this resistor was tuned in the spice simulation, which is discussed in the next section.

2.3 Component level Spice simulation of flyback stage

Once the necessary flyback component parameters were determined, a component level spice simulation was developed, first for one single cell, and later for a full 10 cell balance system. The parameters used were sourced directly from the mathematical model derived in the previous section, and the components used were either models directly taken from the manufacturers of the components chosen or were modeled to as closely resemble the real components as possible. The simulation was developed in LTSpice, which is a free schematic capture and spice simulation engine from Linear Technologies. The simulation topology for a single cell simulation is shown in Figure 2.7.

To validate the flyback components, the single cell simulation was run for only a few pulses. In the simulation, the primary current peaked at 5.5A, the secondary current peaked at 600mA, and the primary (drain) and secondary voltages were nearly exactly the expected values. Some experimentation was also performed with the output dissipation resistor removed to observe the behavior, and as expected, the trapped energy in the transformer cause extremely high frequency oscillations. Both of these results are exampled in Figures 2.8 and 2.9.

For dexterity and sake of completeness, the simulation was then scaled to a full 10 cell simulation. The goal of this full scale simulation was to identify any point of contention when scaling the system, such as interaction of the converters with one another, effects of added series resistance with fuses, and the effects of the snubber circuitry and the additional output diodes. This topology is shown in Figure 2.10, which also shows the addition of fuse resistance, snubber circuitry, and an additional secondary output rectification diode whose purpose is to prevent high current charging of the output capacitances when the pack is connected to the BMS.

This simulation was configured to run for 1800s, with the duty cycle set to 100 percent. The duty cycle was only applied to half of the cells, every other cell in the stack, so that the effects of the cells with balancing enabled could be compared to those without it. This simulation also was used to verify that the power dissipation within all of the components would not exceed the ratings of the chosen parts over a long period of time. Every component was found to be within the limits of the parts chosen, and energy transfer was clearly observed, as the voltages of the cells with balancing inactive increased and those with balancing active decreased.

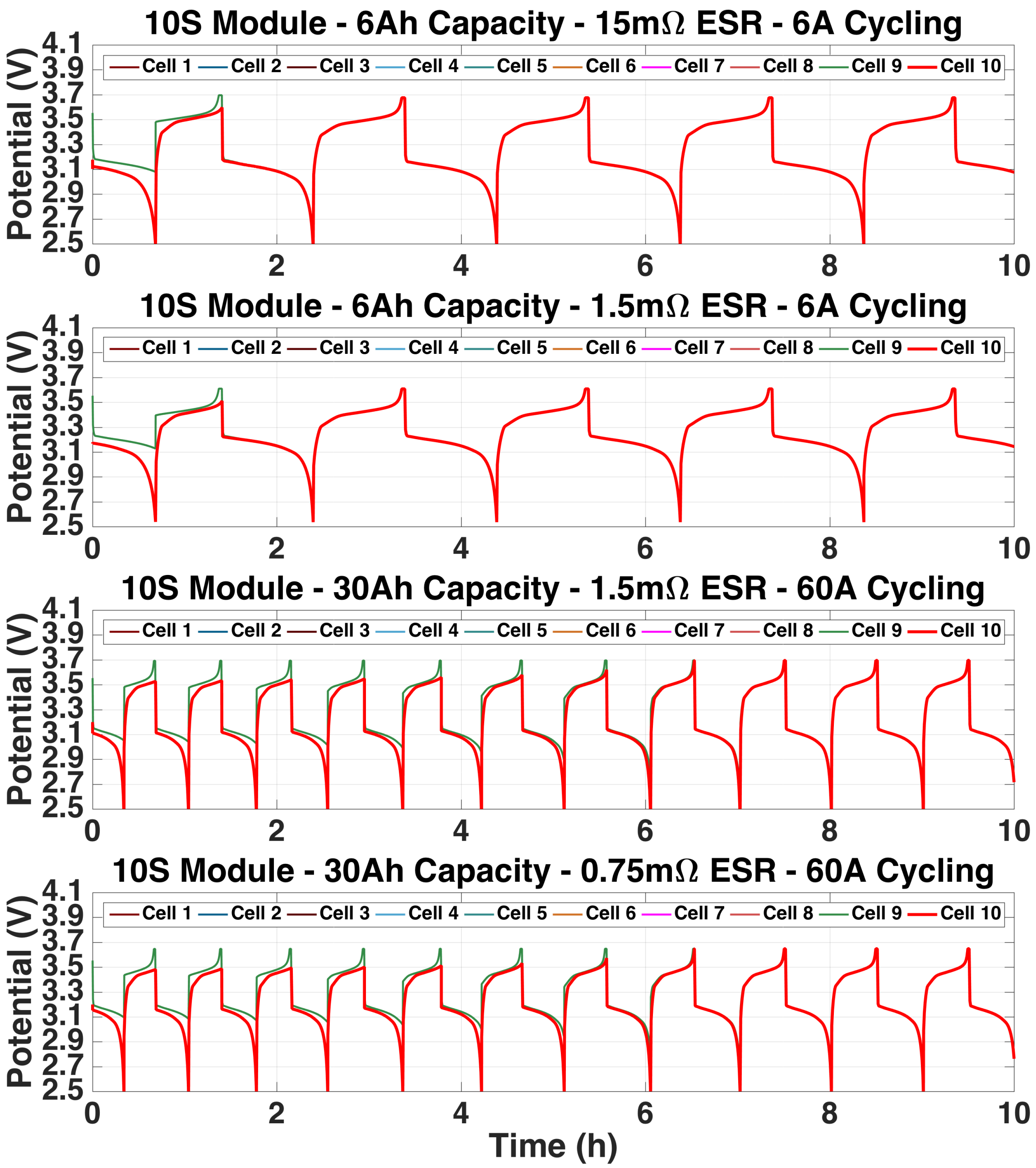


Figure 2.4: MATLAB simulation result showing BMS operation across a sweep of various cell parameters

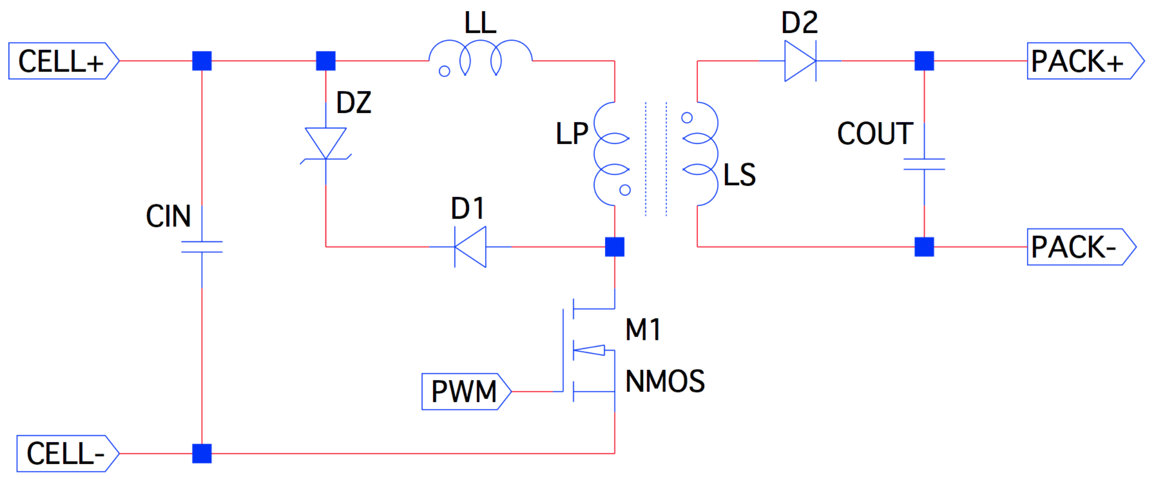


Figure 2.5: Basic flyback converter topology applied in the FlyBMS

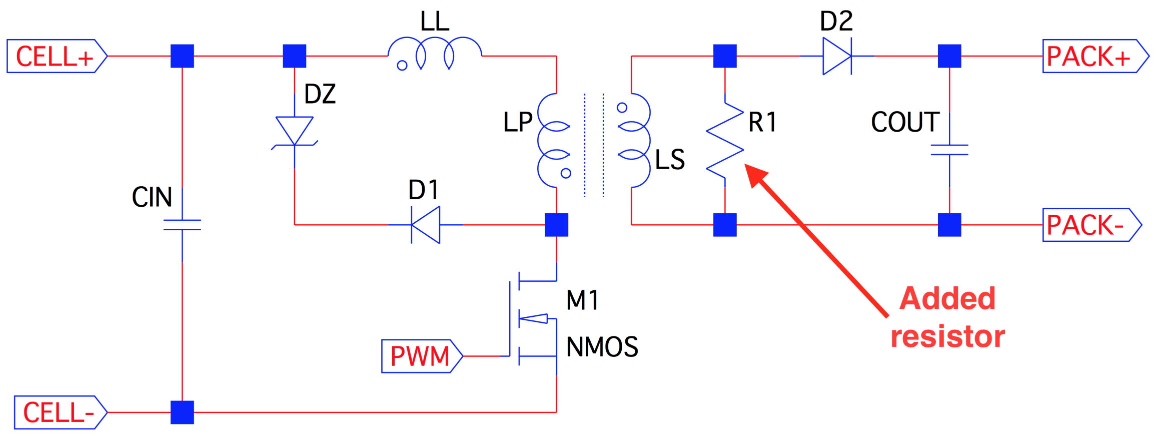


Figure 2.6: Modified flyback converter topology applied in the FlyBMS, with added output dissipation resistor

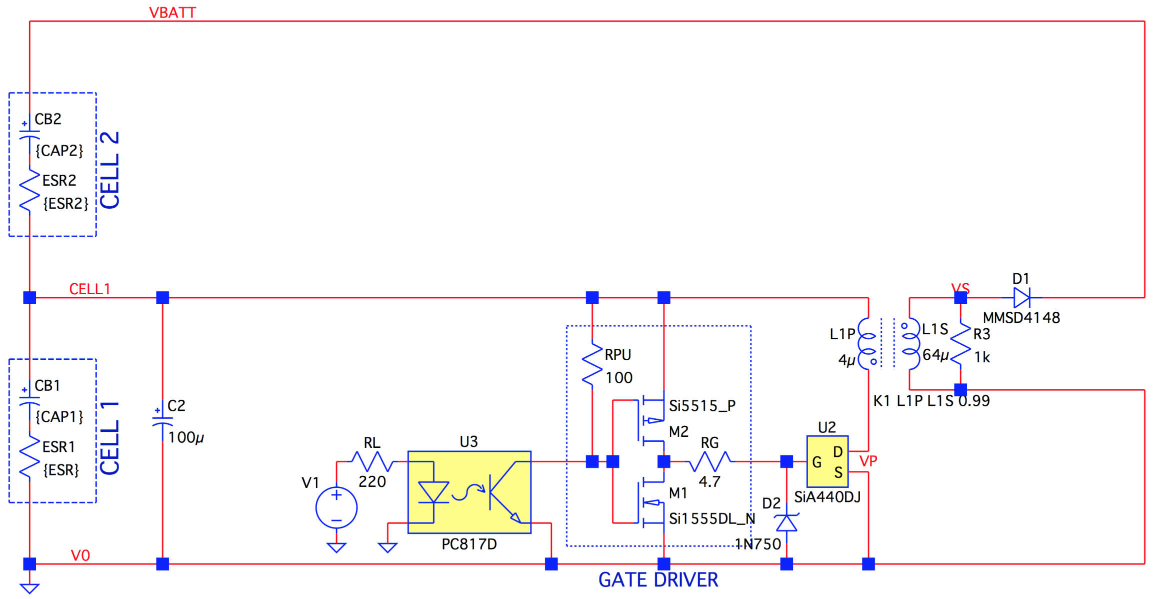


Figure 2.7: LTSpice simulation of proposed flyback balance circuit

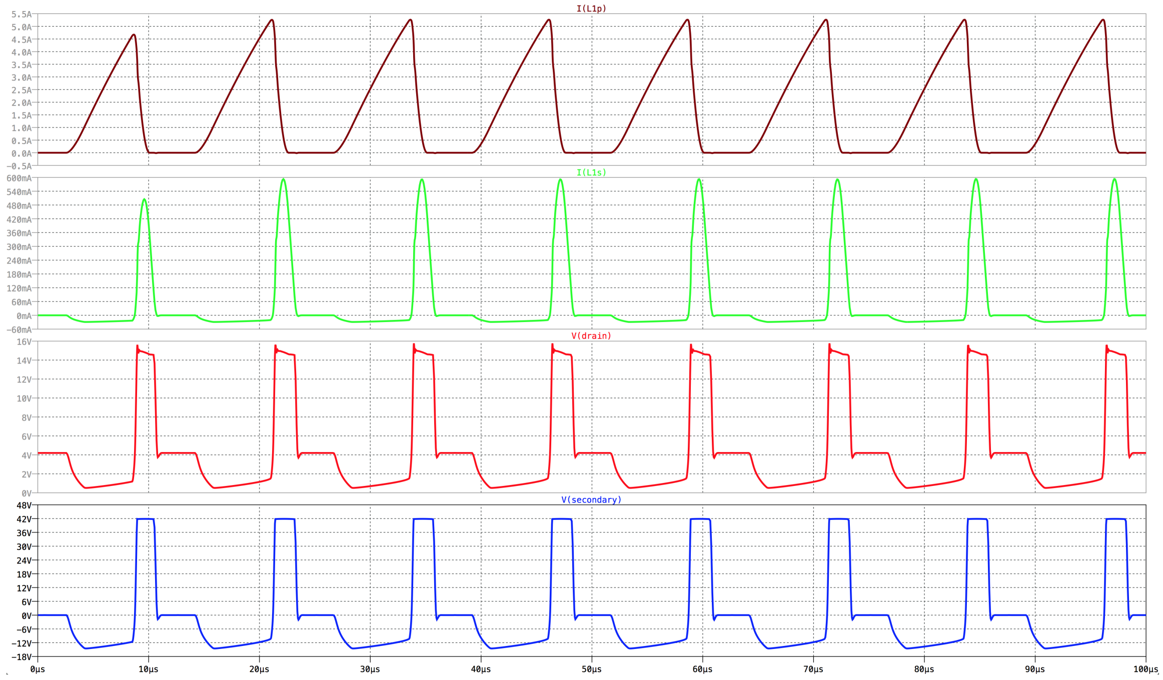


Figure 2.8: LTSpice single cell simulation results with output resistor

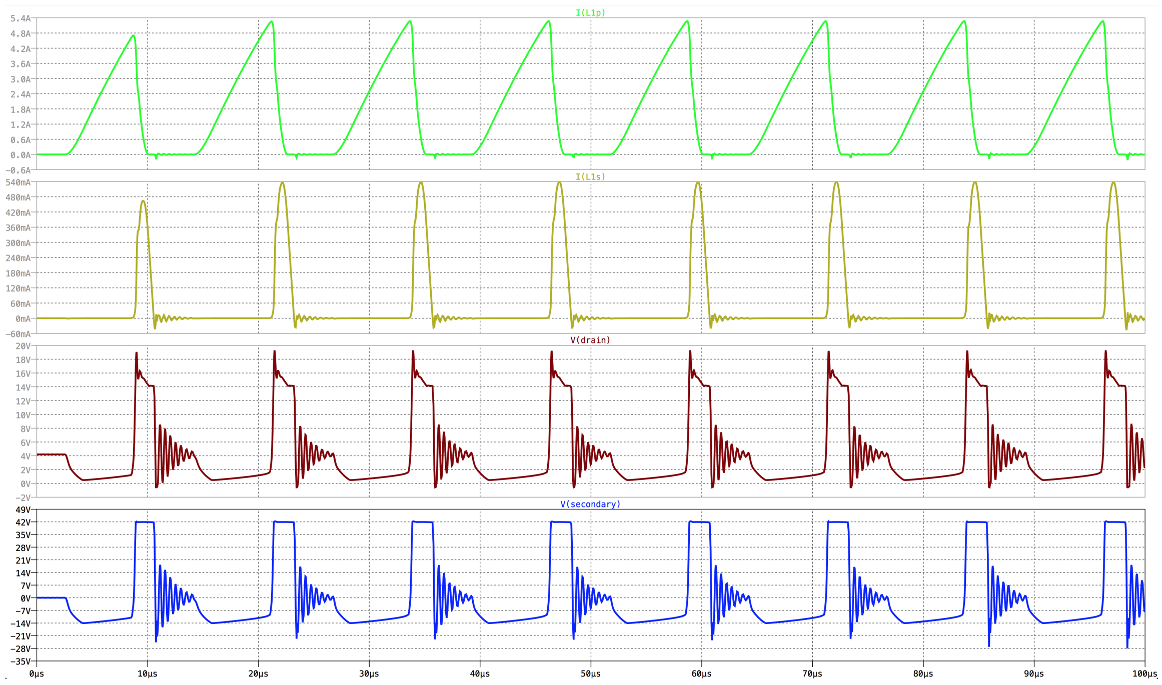
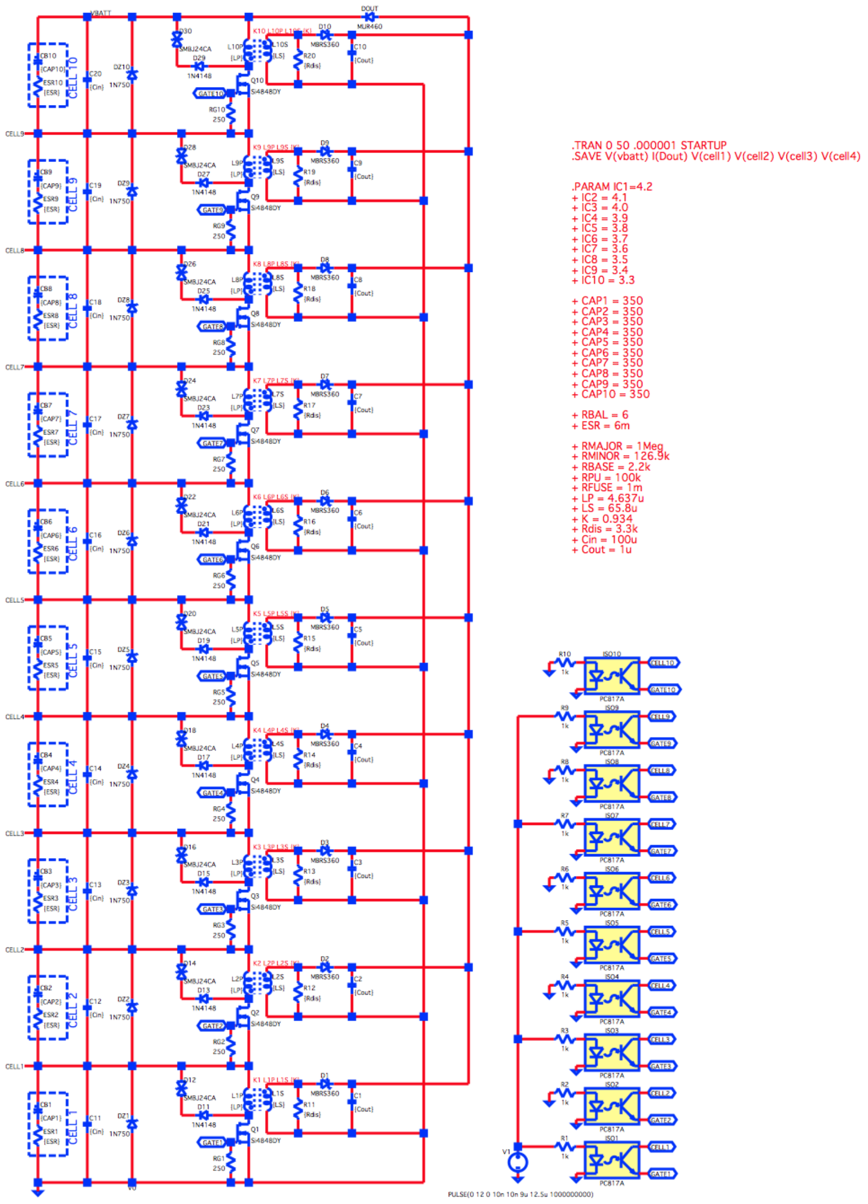


Figure 2.9: LTSpice single cell simulation results without output resistor



```
.TRAN 0 50 .000001 STARTUP
.SAVE V(vbatt) I(Dout) V(cell1) V(cell2) V(cell3) V(cell4)

.PARAM IC1=4.2
+ IC2 = 4.1
+ IC3 = 4.0
+ IC4 = 3.9
+ IC5 = 3.8
+ IC6 = 3.7
+ IC7 = 3.6
+ IC8 = 3.5
+ IC9 = 3.4
+ IC10 = 3.3

+ CAP1 = 350
+ CAP2 = 350
+ CAP3 = 350
+ CAP4 = 350
+ CAP5 = 350
+ CAP6 = 350
+ CAP7 = 350
+ CAP8 = 350
+ CAP9 = 350
+ CAP10 = 350

+ RBAL = 6
+ ESR = 6m

+ RMAJOR = 1Meg
+ RMINOR = 126.9k
+ RBASE = 2.2k
+ RPU = 100k
+ RFUSE = 1m
+ LP = 4.637u
+ LS = 65.8u
+ K = 0.934
+ Rdis = 3.3k
+ Cin = 100u
+ Cout = 1u
```

Figure 2.10: LTSpice simulation of full 10 cell balance system

CHAPTER 3

HARDWARE DESIGN

3.1 Component Selection

Having fully characterized the flyback system as implemented allows for us to fully specify the components in the final design. For each component in the system, a part was sourced that is fully rated to the constraints from chapter 2, was in stock, and was easily workable into an all surface-mount board topology. These component selections are outlined next.

Flyback Transformer: The flyback transformer had to be chosen in advance, such that the rest of the converter could be designed around it. The CoilCraft part chosen met the gain requirements, met the primary and secondary current requirements, met the primary and secondary voltage requirements, had more than adequate isolation ratings between the primary and secondary, and is an 8 pin surface mount part.

Primary Switch: The primary switch requirements are simple. The switch should be able to be driven at 80kHz with minimal switching losses, it should be able to switch a load of up to 10A (for some safety factor) with minimal conduction losses, it should be rated for gate threshold voltages as low as 3.0V, and should be able to survive drain to source voltages in excess of 24.5V. These are very specific requirements for a MOSFET, specifically the low gate threshold and high current capability, and only one manufacturer has a full line of such qualified devices - Vishay.

The MOSFET chosen has a drain to source voltage rating of 40V, has a very low gate charge (6.9 nC) which will facilitate high frequency switching with low gate drive requirements, and the $R_{DS,On}$ is specified at only 35 m Ω at a gate voltage of just 2.5V [13]. The only downside of this particular switch is the package. The only available package is a very small SC-70 surface mount package, that will not only be intolerant to any power dissipation should the switch be inadvertently placed in the linear region during testing, but will also be intolerant to improper soldering, which may be a point of contention during the prototyping phase.

Snubber Circuit: Having the MOSFET narrowed down, we can now choose parts for the snubber circuit. From the math in the previous section, it was determined that the maximum drain to source voltage that will be present on the MOSFET is 24.5V, which is well within the ratings of the MOSFET chosen. To ensure maximum efficiency then, it should only be necessary for the snubber circuit to conduct in the event that a drain to source voltage transient greater than 24.5V occurs. For this reason, a 21V TVS diode was chosen as the snubber diode, which has an initial breakdown voltage at approximately 26.7V, a final breakdown voltage of 38.9V, and a maximum continuous power dissipation of 2W. Since this is a unidirectional part, a second adequately rated Schottky diode was placed in series that would prevent the TVS from becoming forward biased and bypassing the transformer. Both parts are SOD-123 SMD package parts, which should allow for compact packaging.

Output Rectifier: Knowing the peak current, the peak secondary voltage of the transformer, and the peak battery voltage, we can specify the output rectification diode. This diode must be rated to handle the peak pulsed current of the secondary, and must be capable of holding off the the peak voltage developed at the secondary during the switch off period. Math derived previously shows that the peak current in the secondary will be equal to approximately 500mA, but the simulation showed 600mA, so the minimum pulsed current rating of the diode should be no less than 500mA with an adequate factor of safety. The average current will be roughly 35 percent of the peak, so the average forward current should be no less than 350mA. The peak voltage developed at the secondary will be equal to approximately 58V, so it should also have a maximum repetitive reverse voltage of 60V and a maximum non-repetitive reverse voltage of no less than 75V, both with an adequate factor of safety. From these parameters, a Schottky diode was chosen that has a peak non-repetitive reverse voltage of 100V, a peak repetitive reverse voltage of 75V, a peak current capability of 4A, and a continuous forward current rating of 500mA. The package is once again SOD-123 SMD, so it should also allow for simple packaging.

Isolated MOSFET Driver: In order to properly meet the requirement put in place by the rules, for the control circuitry to be completely isolated from the battery, it is necessary to develop an isolated driver stage that is not only able to decouple the control circuitry from the MOSFET, but is also able to drive the MOSFET from the cell voltage with minimal voltage drop in the driver circuit. To accomplish this, a suitable high frequency and low voltage capable MOSFET driver IC was sourced from Skyworks that is specially designed with a CMOS totem pole output capable of driving extremely low voltage MOSFETs [14]. This IC can operate between 2.0V and 5.5V, is capable of driving the MOSFET gate high or low at up to 2A, has an ultra-low logic threshold on the input, and includes a separate enable pin that can be used to locally disable the MOSFET using external detection circuitry for one of many various faults. The only remaining challenge then is isolation, which was simply accomplished using a pull down on the input to the driver accompanied by an open drain optocoupler capable of pulling this enable pin high. This driver stage is depicted in Figure 3.1.

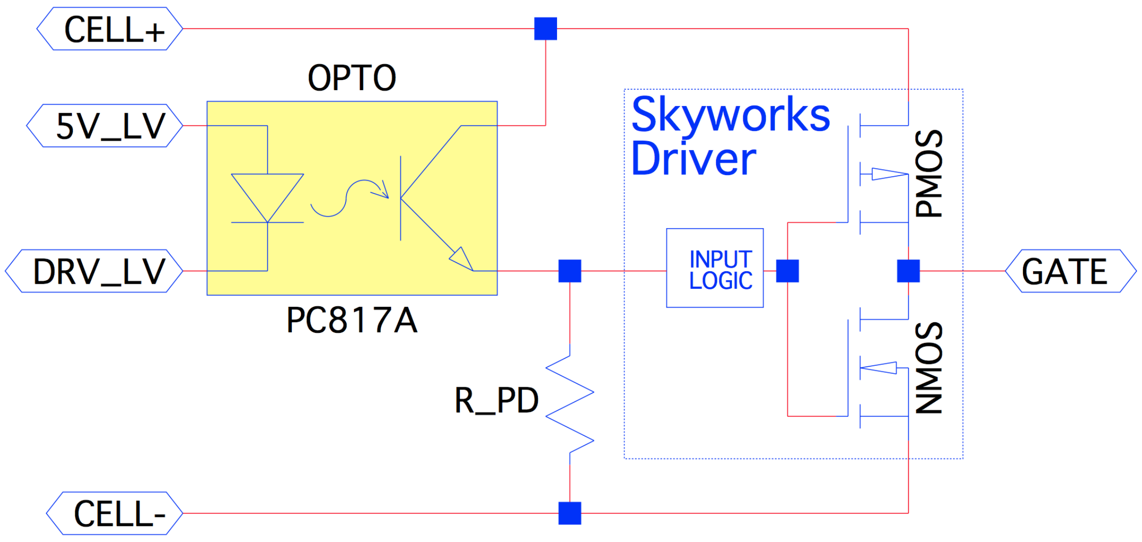


Figure 3.1: Isolated MOSFET driver topology applied in the FlyBMS

Isolated Feedback: In order for the controller, which is isolated from the battery pack, to make informed decisions regarding the state of the cell, there needs to be some form of isolated voltage measurement circuitry. In the final implementation, the controller on each BMS board will talk to each other through an isolated CAN bus, and each board will be able to talk to a host PC through an isolated USB interface, so the safety isolation is already accounted for. The isolation between the pack and the controller is just a second layer of protection, but it is not strictly defined that it is a highly hardened galvanic isolation layer. To simplify the voltage measurements then, very simple differential amplifiers were applied across the input of each cell to take the differential cell voltages and convert them to single ended voltages at the controller for measurement. Rather than implementing these differential amplifiers in discrete components, which would have inherent mismatch and subsequently suffering common mode performance regardless of part tolerances, it was decided to implement purpose built differential amplifier IC. The ICs chosen from Analog Devices have a maximum common mode voltage range of 600V DC, a very low offset voltage for offset error, and have a common mode rejection ratio close to 100 dB. This single IC has all of the differential mode resistance built into the die, and they are laser trimmed for absolute accuracy, so it takes the place of more than 10 discrete components and has twice the expected performance of a discrete solution.

Voltage References: Most micro-controllers only have a 10-bit or 12-bit analog to digital converter built in, so the best resolution we can expect without the use of an external analog to digital converter is just over 1mV, which will then be further reduced by the inherent noise on the analog references of the micro-controller due to the expected noise environment (10 switch mode power supplies nearby). However, the entire lower range of the cells is a useless range to be able to measure. The BMS is never expected to be measuring below 2.0V, and if it is, the only reaction is to disable balancing altogether, making the actual value below 2.0V irrelevant. That being said, we can use the analog references of the analog to digital converter on the micro-controller to effective reduce the range we want to measure, which will increase the resolution within the range measured. The standard internal analog references of the micro-controller are 0V and 3.3V respective, but if we can change the analog range to between 1.5V and 3.3V, that would correspond to the ability to measure battery voltage between 2.0V and 4.4V after a 3:4 voltage divider is applied. This effectively almost doubles the resolution of the measurements, and negates the need for a more costly and more complex external analog to digital converter. Equations

3.1 through 3.4 example these changes and their consequences for a 12-bit analog to digital converter. To accomplish this, two analog reference ICs were sourced from Intersil, one with a 3.3V output and one with a 1.5V output. Both ICs have an accuracy of 0.02 percent, and have internal noise cancellation circuitry to ensure the reference is stable. Both ICs will also be supplied from a 5V source, through a high frequency choke inductors and with multiple local bypass capacitors.

$$V_{Range,Initial} = 4.4V - 0V = 4.4V \tag{3.1}$$

$$V_{Res,Initial} = \frac{V_{Range,Initial}}{12_{bits}} = 1.07mV \tag{3.2}$$

$$V_{Range,New} = 4.4V - 2.0V = 2.4V \tag{3.3}$$

$$V_{Res,New} = \frac{V_{Range,New}}{12_{bits}} = 585\mu V \tag{3.4}$$

Micro-controller: The last major piece in the active BMS puzzle is the controller itself. A micro-controller must be specified that can support up to 12 analog channels (10 cells, one pack voltage, and one supply voltage), has 10 digital output channels capable of at least a simple PWM mode with adequate duty cycle resolution, has 12 additional digital outputs for driving LEDs (10 LEDs indicating cell balancing, 1 for heartbeat, and 1 for overall balance status), supports a UART channel for host PC interfacing, and has an ECAN module for isolated CAN communications. The chosen micro-controller is a flagship product from Microchip. It is a 100 pin TQFP SMD package, with a hardware PWM module, hardware UART, hardware ECAN, and exposed alternative analog reference pins. This MCU has a tried and true architecture, great manufacturer support, is capable of up to 70 million instructions per second (MIPS) operation, and the Microchip line of MCUs are excellently documented which will decrease the time spent in firmware development.

3.2 Printed Circuit Board Design

Once the parts have been specified and sourced, the final step in hardware development is to lay out the printed circuit board. The schematic capture and trace layout package of choice is Altium, which is an industry standard software package. Altium's capabilities far surpass what will be achieved in this design, but the ease of use and built in parts library greatly reduce the development time for the board.

Before the board can be laid out, the schematic needs to be built in Altium with all of the correct parts and packages. To keep the schematic clean, it was captured using a hierarchical multi-channel layout. The low voltage controller and relevant hardware was captured in a single sheet, and then the isolated flyback was was captured in a separate sheet and copied across 10 separate channels. The various modules of the design, including the flyback channels are all presented in depth next.

Power Supply: In order to facilitate a completely isolated design, the input power supply should be of the isolated variety. In order to simplify the creation of a negative 5V rail for biasing the differential amplifiers, an all inclusive IC was sourced that includes two isolated power supplies in one package. One supply generates the +5V for powering the system, and one generates the -5V for biasing. It is a 6W power supply, so both channels can source up to 500mA. The +5V supply then feeds a low dropout 3.3V regulator that power the controller and logic circuitry. The power supply is protected at the input and output from over-voltage, over-current, and reverse polarity, with an emphasis on non-destructive survival, meaning transients just temporarily generate heat, but do not cripple or otherwise shut down the system. The only critical power supply fault that can occur is an over-current at the input, which will open a fuse and subsequently shut down the entire system. The power supply module topology is shown in Figure 3.2.

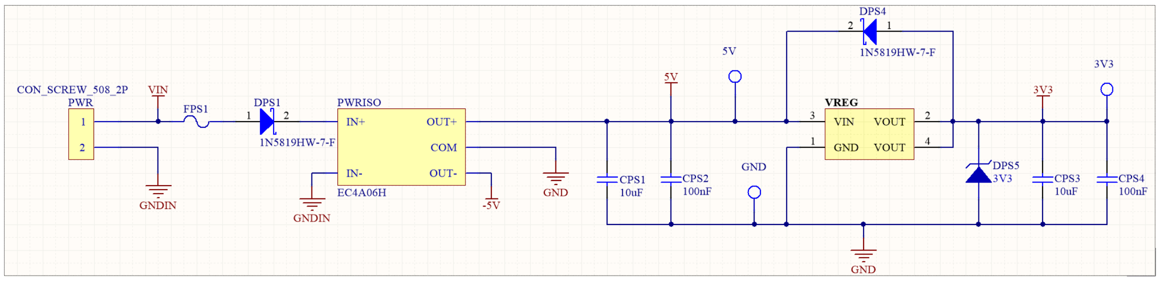


Figure 3.2: FlyBMS embedded system power supply topology

LEDs: The board will have a total of 12 onboard LEDs. Ten of the LEDs represent the balance status of the cells. If balancing is active for a cell, meaning that energy is being actively removed from that cell and put back into the stack, that cell's respective LED will be on. One of the LEDs will represent the status of the balancing state machine. If balancing as a whole is enabled, meaning that the balancing state machine is running, then this particular LED will be on. The last LED is the system heartbeat. This LED blinks at a specific rate to show that the system is alive and running properly. Each LED needs a minimum of 10mA to make is clearly visible, which is a total of 120mA that the MCU would have to source directly if the LEDs were directly driven by the MCU. That is too much. In order to move the main current path away from the LEDs with minimal parts count, digital transistors were used. These are nothing more than common BJTs with biasing resistors on the base such that they produce a specific current - 10mA at 3.3V in this case. Figure 3.3 shows this implementation.

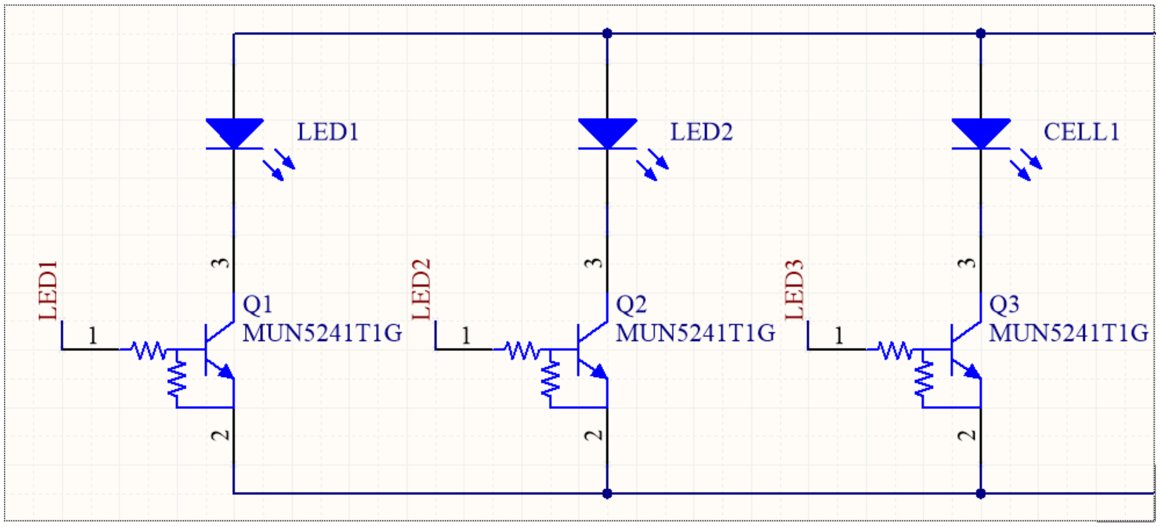


Figure 3.3: FlyBMS embedded system LED driver topology

Micro-controller: Supporting circuitry for the micro-controller is paramount. Improperly laying out your MCU support topology can result in many hard to track down errors. For starters, the data sheet of most MCUs very clearly calls out the required ratings for the various required bypass capacitors. Not following these guidelines and using a 100nF in place of a 10μF or skipping a bypass capacitor altogether can render the IC un-programmable, or even worse, intermittently inoperable. Also improperly loading your oscillator according to the oscillator's data sheet can cause it to not oscillate, or to deviate greatly in frequency, which is extremely difficult to diagnose. In order to make sure that the MCU was set up properly, the data sheet for both the MCU as well as the oscillator were consulted, and the various capacitance and resistance around both were properly specified in order to guarantee proper operation. See Figure 3.4 for the final topology of the MCU supporting circuitry.

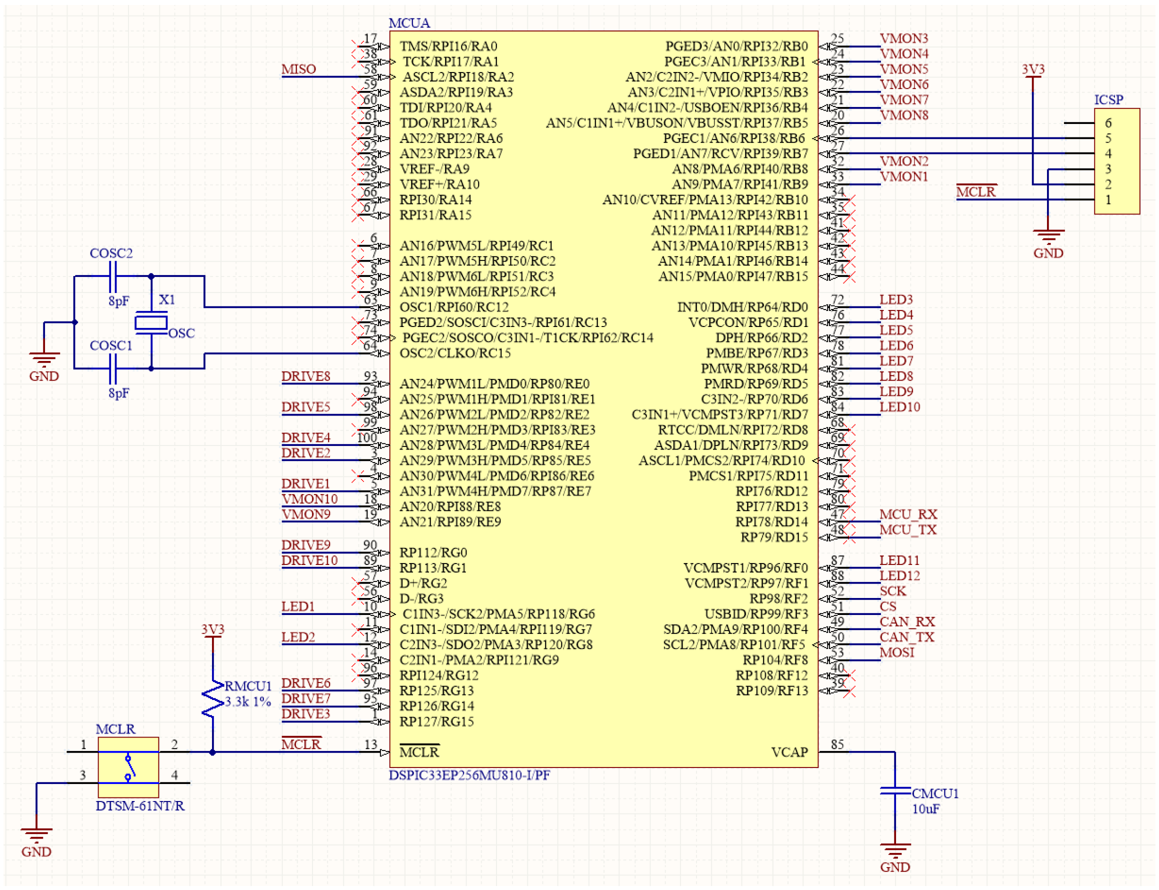


Figure 3.4: FlyBMS embedded system microcontroller topology

USB Host Communication: The BMS itself has no button or other types of input for interaction, so in order to facilitate interfacing with it such that we can change configuration settings live, alternate between automatic balancing mode and manual balancing mode, increase or decrease the duty cycle, or enable and disable duty cycle, we need a way to communicate with it externally via software. This communication interface was implemented using the MCU's Universal Asynchronous Receiver/Transmitter (UART) module, which was interfaced with externally via an FTDI USB to UART bridge IC. This IC and module combination allows the end user to plug a USB cable into the board and interface with the MCU through a virtual COM port and a simple terminal session. In order to facilitate maximum safety and to meet the requirements of complete isolation, the USB bridge was implemented through an isolation boundary via a Texas Instruments digital isolator IC that uses a novel capacitive boundary to achieve full galvanic isolation while supporting data rates up to several megabits. This circuit is shown in Figure 3.5

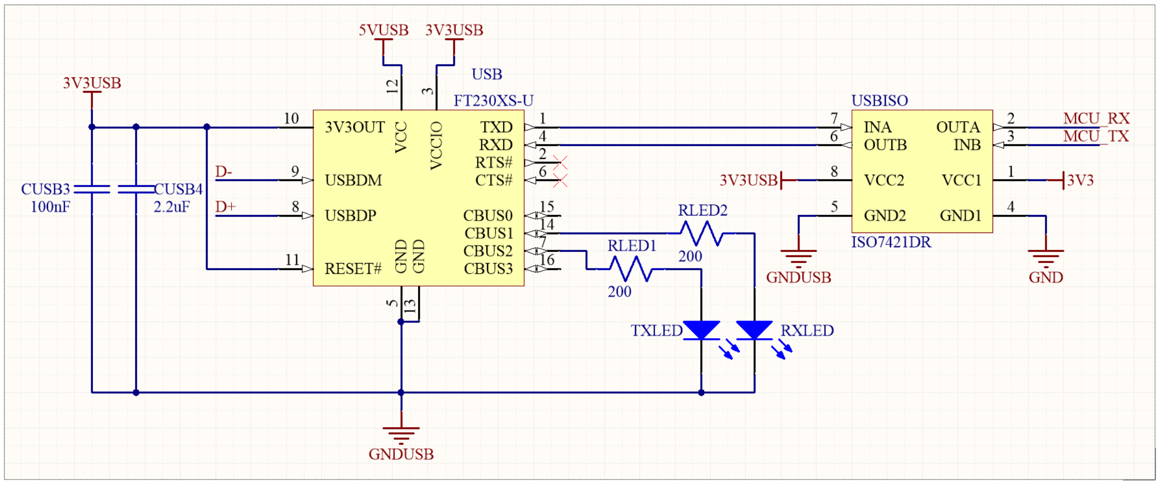


Figure 3.5: FlyBMS embedded system isolated USB host interface topology

Cell Interface: Due to the fact that the cell interfacing circuitry, namely the flyback converter and the voltage measurement circuit, are to be repeated 10 times within the schematic, it was formed as a sub circuit in Altium. All of the flyback components, the isolated driver components, and the cell voltage measurement components were all placed into a separate sheet, and then using the "channel" functionality built into Altium, it was repeated 10 times and built into a bus within the schematic. This multi-channel sub sheet cell interface is shown in Figure 3.6 and Figure 3.7 shows how this sheet's multi-channel bus was integrated into the main schematic.

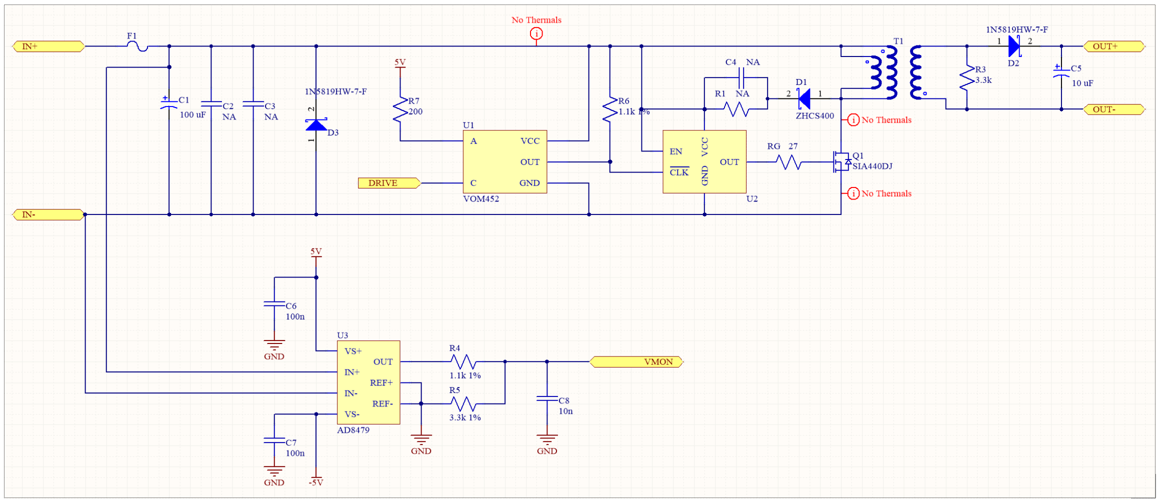


Figure 3.6: FlyBMS embedded system cell interface topology

Having finished the schematic, the next task was PCB layout. Altium had most of the parts available within its vast parts library, but some of the part packages were created manually, such as the transformer, the isolated power supply, the differential amplifiers, and the MOSFET drivers. The philosophy on laying out the first revision of the PCB was simple: keep it as tight as possible while maintaining adequate spacing for the inevitable troubleshooting and part replacement. The cell interfaces

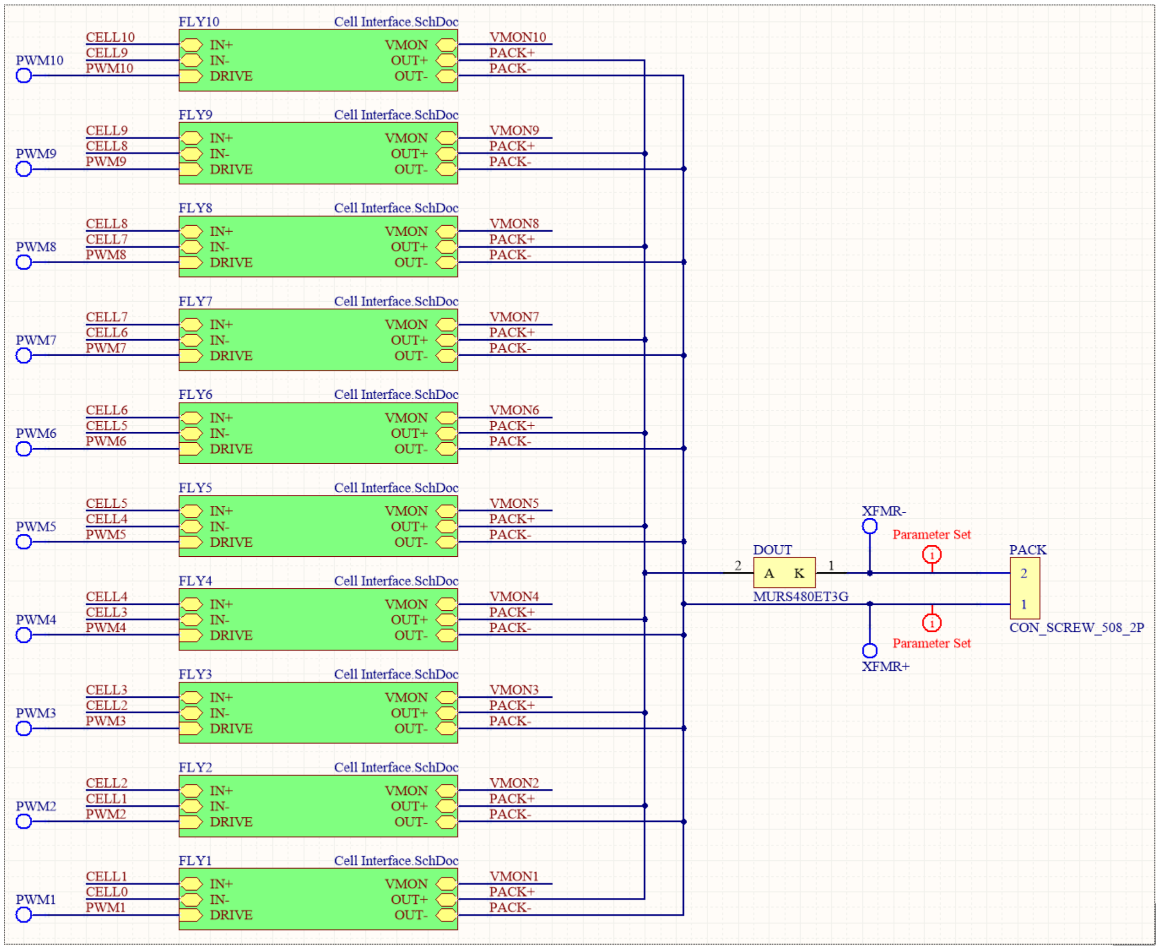


Figure 3.7: FlyBMS embedded system cell interface multi-channel implementation

were all placed in parallel, with a single 11 pole screw terminal connector leading into all of them, and the isolated output to the pack positive and negative terminals was separated such that the BMS could be independently tested with a power supply on a single channel and a battery pack on the output. The isolated power supply was remotely located away from the low voltage control circuitry and analog references to prevent additional unwanted noise from causing issue, though in hindsight that likely would not have been an issue. The external communications were located near each other for ease of access, the LEDs were grouped together by what they represented and were properly labeled, and the low power MOSFETs that drive the optocouplers were all grouped close together for ease of troubleshooting. There were also strategic test points located on the board to give easy access to measuring cell voltages, pack voltage, isolated power supply voltages, MCU power rail voltage, and the PWM signals driving the optocouplers. Figures 3.8 and 3.9 show the 2D and 3D representations of the final PCB design in Altium, and Figure 3.10 is a photo of the final assembled system. It should be noted here that the voltage references that allowed for higher resolution voltage measurements was an addition after the initial design had already been completed, and therefore was added externally to the PCB. The black 3D printed enclosure attached to the board houses this separate circuit and the flying leads are soldered directly to the voltage reference pins of the MCU.

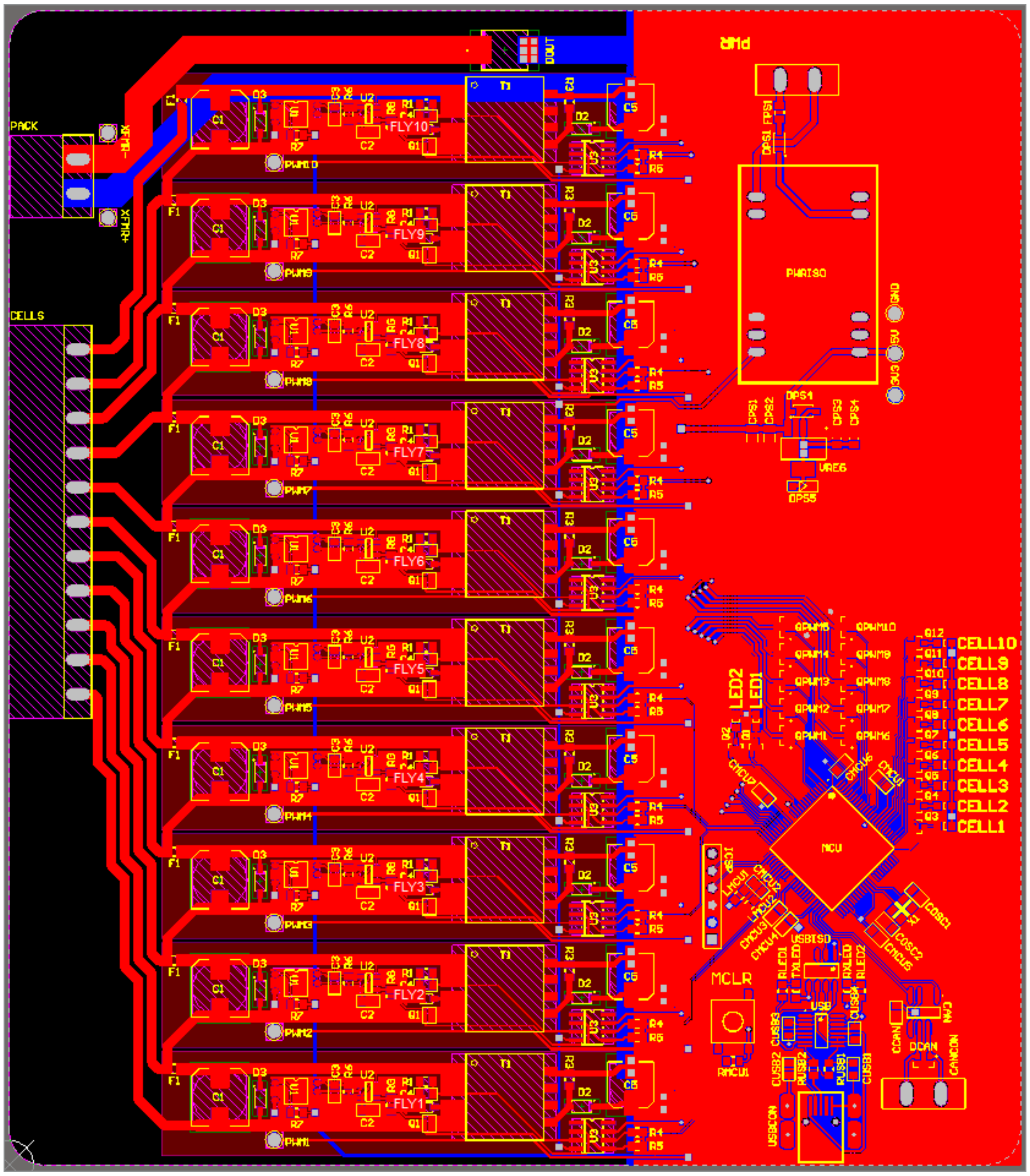


Figure 3.8: FlyBMS embedded system PCB 2D representation

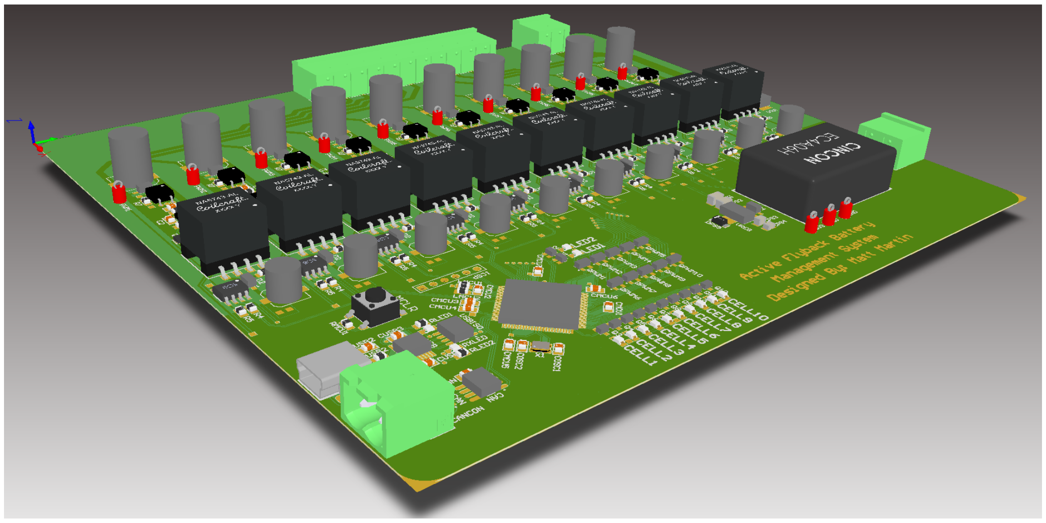


Figure 3.9: FlyBMS embedded system PCB 3D representation

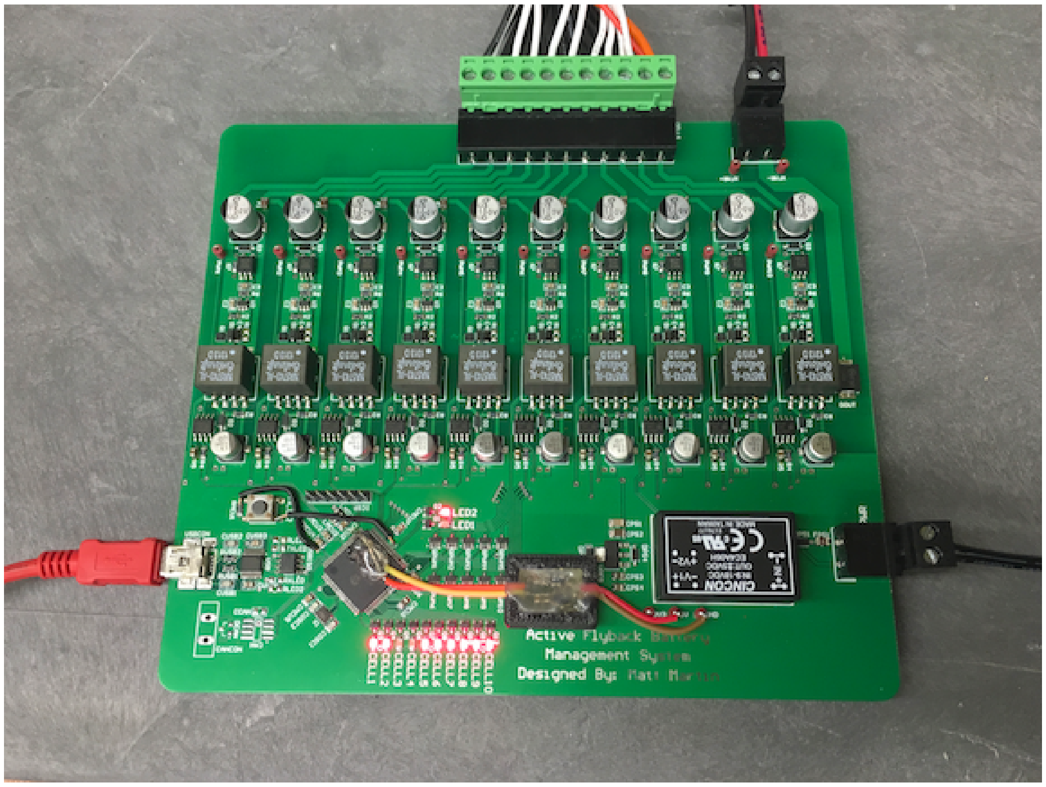


Figure 3.10: FlyBMS embedded system final form

CHAPTER 4

FIRMWARE DESIGN

4.1 SimSTOS - Simple Soft Time Operating System

First and foremost, it should be known that I am not a software engineer, and I have never had any formal training, in the classroom or otherwise, regarding properly designing and constructing frameworks for embedded systems. That being said, this experience was viewed as an opportunity to expand my knowledge when it comes to embedded software, as well as with the C language as a whole. The main goal for SimSTOS was to design a system that is simple to implement and deploy, and that would functionally mimic the behavior of a deterministic real time operating system without the added complexity of context switching and run time task management. After some research, and some insight from those who knew better than I, it was realized that three of the most important concepts to take to heart when designing an embedded system are: flexibility, maintainability, and extensibility. Flexibility refers to the inherent universal applicability of the system. If the system is properly structured, you should easily be able to adapt it from one application to another with only minimal changes. Maintainability of course refers to the overall ease of management of the system during troubleshooting and debugging. Functions, variables, constants, and defines should all follow a meaningful, well structured naming convention, and that convention should be strictly adhered to. Extensibility then refers to the ability to easily expand the system's functionality. Extensibility largely comes as a side-effect of building the system with flexibility and maintainability in mind. If the system is modularly designed such as to ensure flexibility, and is cleanly structured and named, such as to ensure maintainability, it is easy to add modules to the system, making it also extensible. This chapter outlines some of the key points of the embedded firmware architecture and the battery management system specific code.

In order to ensure that the system developed adhered to the strict requirements of extensibility and flexibility, it was important to design it in such a way that it was hardware independent. That is, if I wanted to move it from a Microchip DSPIC33E architecture, to a ST Microelectronics architecture, or to an Arm architecture from one of many manufacturers, it should only require refactoring of the code pertaining to the hardware interfacing functionality of the system. This type of structure, where the application layer sits on top of the system layer, and the system layer sits on top of an abstraction layer, is very common in embedded system design - specially homegrown systems such as this that don't rely on existing platforms or frameworks such as a Real Time Operating System (RTOS) or MikroC's custom compiler/driver platform. This is the approach that was taken in designing "SimSTOS", which is a simple modular embedded system framework with a low level hardware abstraction layer, a system middle layer with a simple run-to-completion (RTS) time sliced non-preemptive scheduler, and a high level application layer where the application specific tasks live. This architecture is shown in Figure 4.1.

Hardware Abstraction Layer: The hardware abstraction layer (HAL) is the layer containing all of the hardware interfacing code for both the internet and external hardware that the system supports, and provides functions for decoupling of the hardware from the system functionality. For SimSTOS on the DSPIC33E, this includes the pulse width modulation module, the analog to digital converter module,

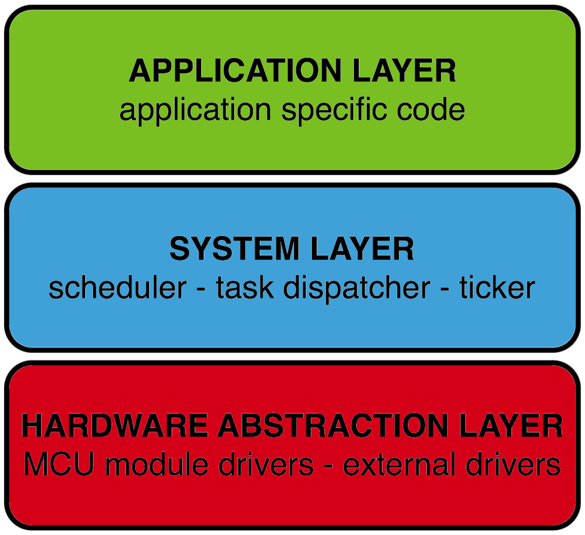

APPLICATION LAYER
application specific code
SYSTEM LAYER
scheduler - task dispatcher - ticker
HARDWARE ABSTRACTION LAYER
MCU module drivers - external drivers

Figure 4.1: SimSTOS architecture layout

the universal asynchronous receiver transmitter module, the enhanced controller area network module, the change notification module, support for the input capture module, support for the output compare module, support for timers with interrupts, and support for external interrupts. For maintainability, all of these support structures are built into their own C-files, with a uniform naming convention, which can be included when support for a specific module is needed. This layer is written much in the form of a large library of functions that address specific hardware registers, with conditional defines in place for the various micro-controllers that the system can be deployed on, which currently is just a host of DSPIC33E devices with only slightly varying hardware modules, but could easily be expanded to other manufacturers/architectures due to this layout. Each module is prefaced with a set of address defines for all of the various registers in the MCU, and the subsequent functions populate the registers according to the function inputs, which are typically received from application layer calls, though some HAL function calls are made within the system layer as well for clock and scheduler configuration and initialization. The HAL also extends OS support to more than just the MCU hardware. It also contains the interface for external hardware, such as sensors, memory, displays, and even software interaction drivers. Anything that the application needs to interact with is supported within the HAL. Currently the HAL's external support functionality only includes support for flashing and communicating with a serial character display, as well as two way communication with a host PC terminal through ASCII character encoding and decoding/parsing, both of which are shown in Figures 4.2 and 4.3 showing the drivers in action on a DC/DC capacitor charger where SimSTOS is also deployed. Note in these figures that the parameters displayed need not be hardcoded for every new application. The drivers expose the ability to create a new item for display, adjust the line on which this item is displayed, change the title of this item, and to populate the displayed value without rewriting the entire line. This driver greatly allows the application designer to display key application specific parameters without the added overhead typically associated with writing large ASCII blocks over a serial port. In the PS terminal example, the system spends less than 1ms populating all of the parameters on the display, which is extremely fast.

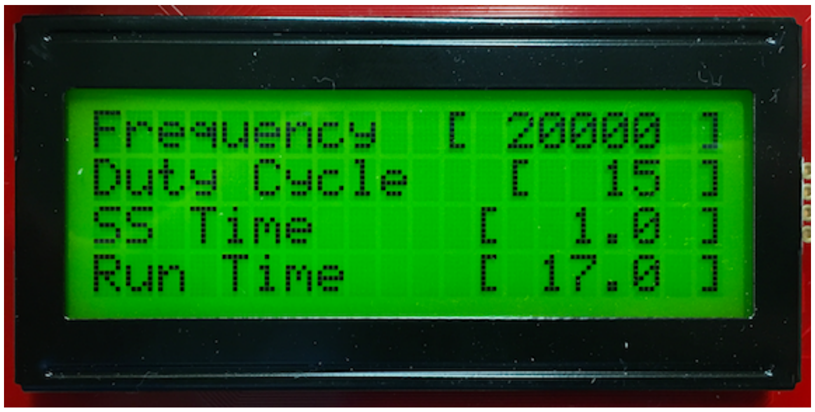
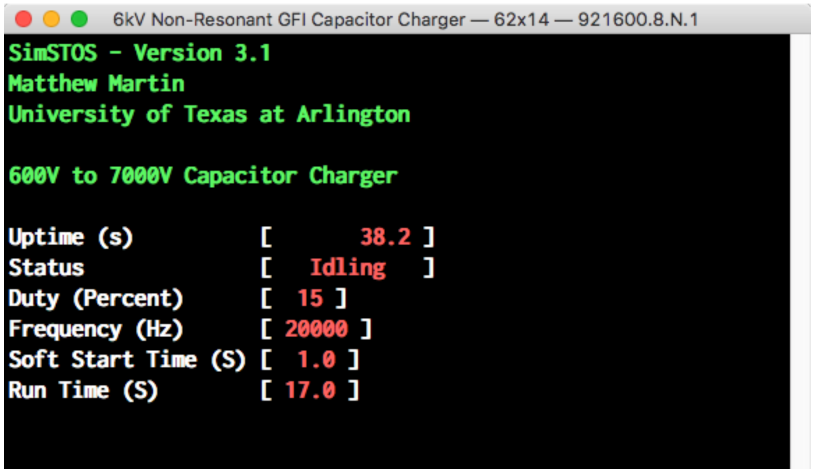

```
Frequency  [ 20000 ]
Duty Cycle   [  15 ]
SS Time     [  1.0 ]
Run Time    [ 17.0 ]
```

Figure 4.2: SimSTOS display driver example usage


6kV Non-Resonant GFI Capacitor Charger — 62x14 — 921600.8.N.1

```
SimSTOS - Version 3.1
Matthew Martin
University of Texas at Arlington

600V to 7000V Capacitor Charger

Uptime (s)          [       38.2 ]
Status              [   Idling   ]
Duty (Percent)      [  15 ]
Frequency (Hz)      [ 20000 ]
Soft Start Time (S) [  1.0 ]
Run Time (S)        [ 17.0 ]
```

Figure 4.3: SimSTOS serial interface driver example usage

System Layer: The system layer is comprised of all of the components related to the operation of the system kernel, ie. the system counter, the task execution stack, the scheduler, and the task dispatch function. The SimSTOS system layer is the middleware that facilitates the execution of the application layer tasks in a semi-deterministic manner, and provides system level functions to the application layer to request task status, view execution stack que depth, and get the system time. The system counter is nothing more than a hardware timer of the micro-controller set up to interrupt at 500 μs intervals, tick the state counter, and flag any current active and running tasks that have expired and should therefore be executed. The OS state counter also triggers a call to a task dispatch function, which looks at the execution state for any tasks that have been flagged for execution and executes which ever task on the stack has the highest priority. The task execution flag that is shared by the state counter and the dispatch function is a 9 level deep counter, which is incremented by the state counter and decremented by the dispatch task, and is protected by the natural program flow. This implementation means that if a task takes a back seat to another task, and misses an entire window of execution, it's execution flag will show that it needs to be ran twice, and it will therefore be called back to back by the task dispatch function, provided that it is the highest priority task on the stack two execution calls in a row. This functionality is extremely useful for systems with fast running time critical loops that rely on measurements from slower tasks that could take longer than their allotted time slice. In this case, the control loop will only experience a single time discrepancy for one iteration, rather than an entire missed iteration. This functionality can also be disabled, such that the highest value in the execution que for any task is 1, which means that missed executions will just be skipped. This kernel then is extremely flexible, has very little CPU overhead for the scheduling functionality, and is inherently non-blocking to the hardware level interrupts since the inherent hardware interrupt priority is leveraged. There are some downfalls of this implementation though, the largest of which is that it relies on a run to completion (RTC) scheduling strategy paired with a time sliced scheduling strategy, which implies that if the application layer designer is not aware of the execution time of a scheduled task, and its execution time is greater than its time slice, the system will hang on executing that task and loop forever. There is a functionality built in that will kill a task if the execution depth gets larger than the allotted number, but it is only in place to call a fail-safe condition that the application layer designer can use to return the system to a safe state and/or trigger alerts.

Application Layer: The application layer is very simple: this is where the code that is specific to the end application lives. In the application layer, the application designer initializes a task with a specific timing, assigns a priority to that task, populates that task's run time code with the application code specific to that loop rate, and then initializes the OS/scheduler. For the BMS, this is where the balance state machine is executed, in a 10ms task with the highest priority. Two other tasks also reside in the application layer for the BMS, a very simple heartbeat task that toggles the heartbeat LED every 250ms, and a host communication task that is responsible for the serial data output for logging and external interaction.

To validate the design of the SimSTOS scheduler, a simple test series was performed that would not only validate the scheduler's timing abilities, but also the scheduler's adeptness at priority arbitration for simultaneously expired tasks. The test was performed using nothing more than 8 LEDs and an oscilloscope. In order to test full scale timing, 8 tasks with varied timing were initialized, each one's job being to toggle one of the 8 LEDs. The LEDs were then probed to verify each of the task's timing, and each was was at least as accurate as the oscilloscope's measurement capabilities. To test the priority arbitration, the same tasks were set up with identical timing, and the priority of each was altered. Two probes were then used to validate that the higher priority tasks were being dispatched first, and they were. Due to the nature of the test, it was required to insert a delay into each task so that there was a delay between executions, because the time resolution of the oscilloscope was not small enough to capture it while simultaneously capturing the entire task's overall timing otherwise. To accomplish this in a non-blocking manner, the final addition to the system layer of SimSTOS was a system level delay function. Rather than entering a finite execution loop full of "no-ops" like a typical embedded delay function is implemented, it was implemented using a simple concept of waiting for a certain amount of time to pass in the system counter. This method ensures that calling a delay at any time will only affect the components in or above the application layer, provided that the sum of the task execution time and the delay time do not exceed the timing of the smallest sliced task that is active in the scheduler.

4.2 Battery Management State Machine

At the heart of the FlyBMS firmware, wrapped in I/O HAL calls and flow control code for the various analog and digital signals, is the balancing state machine. Very similar to the MATLAB simulation balance state machine, this state machine contains many states specific to the operation of the balance algorithm. These various states handle data capture from serial inputs, serial data output structure formatting, analog input capture, pack statistic calculations, PWM duty cycle calculations, PWM duty cycle enable and disable, and cell balance status LED control. The state machine structure is shown in Figure 4.4. Notice in the flow chart that the task is executed every 10ms, but the duty cycle is only applied every 250ms. It was determined in the high level simulations that the optimal control bandwidth for the balance algorithm, which is the portion of the state machine that calculates the duty cycle target as a function of cell voltage discrepancies, was 4Hz. However, to ensure that the transformer primary windings are not saturated when rapid changes in the cell voltage occur, the duty cycle target needs to be scaled by cell voltage much more often that a 4Hz rate.. Instead, the duty cycle target that is calculated every 250ms represents the percentage of time the PWM signal is high out of the maximum duty cycle that can be applied at a given cell voltage, and is applied to a maximum that is calculated every 10ms. In other words, every 250ms the desired balance current is calculated and converted to a duty cycle percentage, but every 10ms the maximum duty cycle for a balance current of 2.5A is recalculated. This approach ensures that even when the battery voltages changes rapidly due to sudden changes in external pack current, the balance current being requested stays constant. This is analogous to constantly changing the angle of a pitcher full of water as it empties in order to keep the rate at which the water is poured out constant. This control topology was a departure from the initial balancing state machine, because it was found in testing that when an external load was removed, the voltage of the cells would rapidly rise, which in turn raised the input voltage to each of the flyback converters and caused saturation of the primary side that resulted in MOSFET failures. This rapid adjustment of maximum duty cycle allows for the balance current to remain nearly constant regardless of the cell voltage.

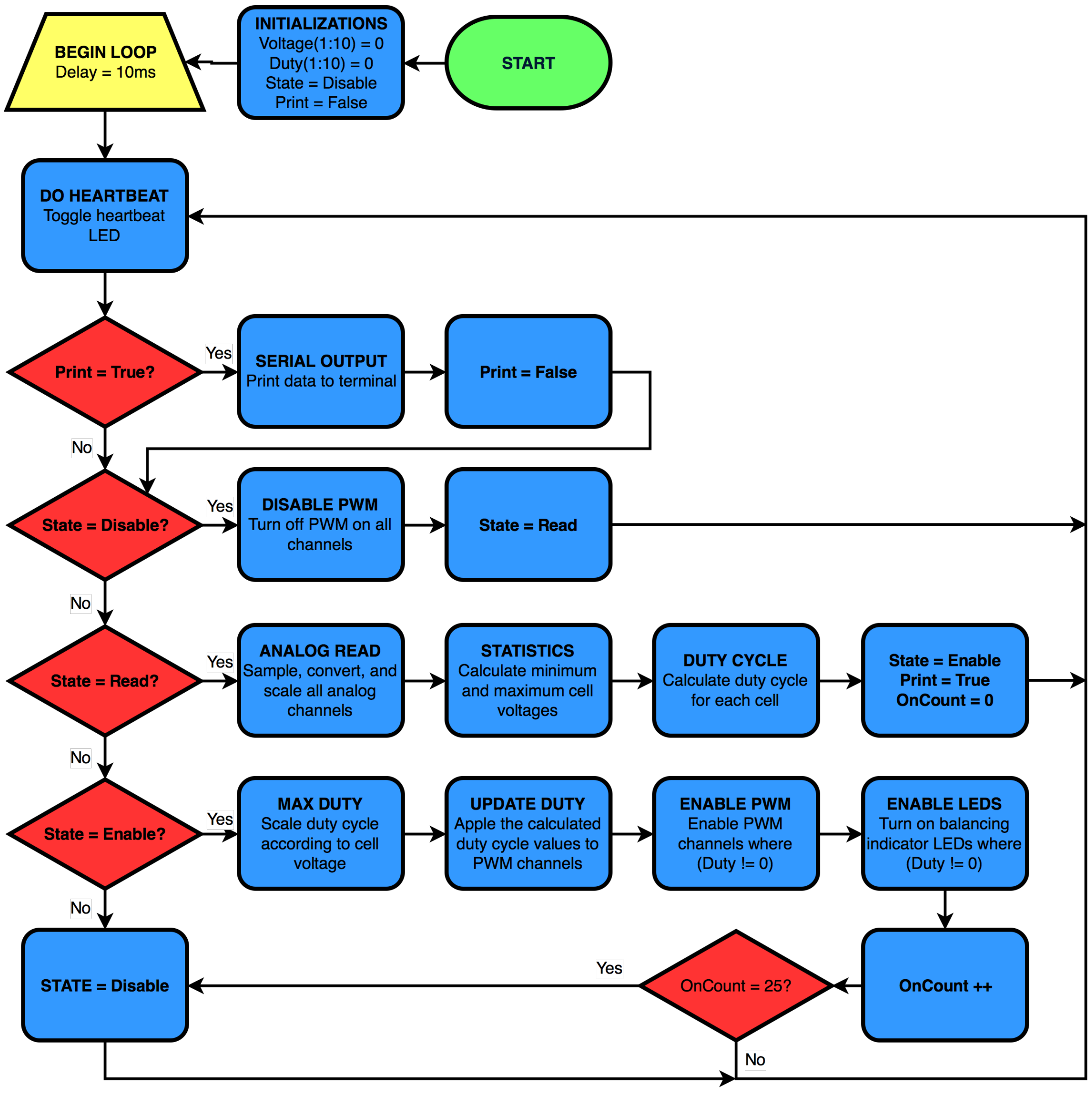


Figure 4.4: Balancing state machine flowchart

CHAPTER 5

DESIGN VERIFICATION

5.1 Measurement Noise

The first step in design verification for any complex system is to test the components individually to validate individual components in an isolated manner. Isolating the testing of each component is critical in troubleshoooting because initial testing always exposes the errors made in design and/or construction. Many tests were performed during the FlyBMS validation. The 5V and 3.3V power rails of the control circuitry were verified to be within specifications. All connections at the MCU were tested, and a small correction to the reset button due to a missed trace was the only modification that was required. Operation of the MCU PWM module, unit testing of the serial data input and output functions, unit testing of the LED controls, and unit testing of the analog input module were all performed on the MCU, and after some minor corrections at the HAL level, proved to be fully functional. Proper PWM signal propagation across the isolation boundary was tested, and after realizing the necessity for and subsequent addition of pull down resistors to the MOSFETs driving the optocoupler LEDs, was found to be fully functional. Each of the 10 flyback converters were indivisually tested by moving energy from a power supply into a bank of super-capacitors, and after some part choice changes involving output diodes and clamping diodes, they were found to be fully functional. The only major point of contention with the system that can be contributed to a design flaw was the analog input stage. The initial design decision for the analog inputs was to sacrifice some resolution of the cell voltage measurements in order to use the MCU's 12-bit ADC in default mode, which involved setting up the micro-controller's ADC module such that the analog inputs were referenced to the 3.3V power supply rails. Based on past experience, it was expected that there would be no more than 5mV of noise on the ADC's internal references, so it was understood that the maximum usable resolution of the ADC was 10mV. Digital filtering would be accomplished by dropping the last two bits of the 12-bit conversion results, effectively operating the ADC in 10-bit mode. However, initial testing of the analog input stage yielded much higher noise levels than this - up to 30mV in some cases.

The initial analog input stage design involved no filtering on the inputs, so it was first assumed that the differential amplifier modules were the source of the noise due to their physical location near the switching components. Some testing with the flybacks on and off proved this to be true to some extent, but it was clear this was not entirely the source of the noise. In an effort to reduce noise, low pass RC filters were implemented as part of the 3:4 voltage dividers on the output of the differential amplifiers, and differential filtering was added to the inputs of the differential amplifiers. The first filter test was performed with a 7kHz low pass filter on the outputs. To validate results, ADC data was collected for 10 minutes from completely stable cells that had been at rest for more than a week. Analysis of this test yielded a standard deviation of approximately 8mV, a significant reduction from initial unfiltered testing. This trend showed promise, so a second test was performed with a further reduced low pass filter of 65Hz, but the results (also in the figure) did not show further improvement. The noise floor was still above the desired threshold, so the remaining noise was clearly within the MCU's ADC module itself. As was discussed earlier in chapter 3, a change in the ADC reference voltage implementation was made. Rather than referencing the inputs to the MCU's internal references, which are derived from the 3.3V rails and apparently extremely noisy, two separate voltage reference ICs were externally added to the board and connected to the MCU's external voltage reference inputs. These references, at 3.3V and 1.5V respectively, not only reduced the noise in the conversion process, but they also increased the effective resolution from 8-bits with noise considered to 12-bits with noise considered. The final noise floor in the analog input system was reduced to a standard deviation of just 1mV. The results of this testing are shown in Figure 5.1. The figures are shown with equal scaling to emphasize the level of improvement.

5.2 Final Results

Once the system components were individually verified and calibrated and the balance state machine was unit tested, full scale testing of the FlyBMS commenced. The first test was was to perform a 1C cycling of a 10-cell battery pack with and without the balancing state machine active. The pack was constructed intentionally with very poor quality lithium-iron phosphate cells that had been in storage for an extremely long period of time. This worst case scenario test facilitates observation of the balance state machine in a host of operating modes, because the cells not only vary in state of charge, but the capacity and ESR of each cell also varied greatly. The pack then, even when balanced, will still have variations in voltage throughout the charge and discharge cycling. To further compound this, the 10-cell pack was actually constructed in a 10S2P configuration, meaning that there were 10 cells in series, and each series node consisted of two cells in parallel. This further causes deviation in the dynamic behavior of the pack, because no two parallel cells were paired with a similar rated cell. Before the pack was constructed, each cell was characterized individually to determine the capacity and ESR of each cell in the system. The parameters for each cell are shown in Table 5.1. The cells were paired in parallel in alphabetical

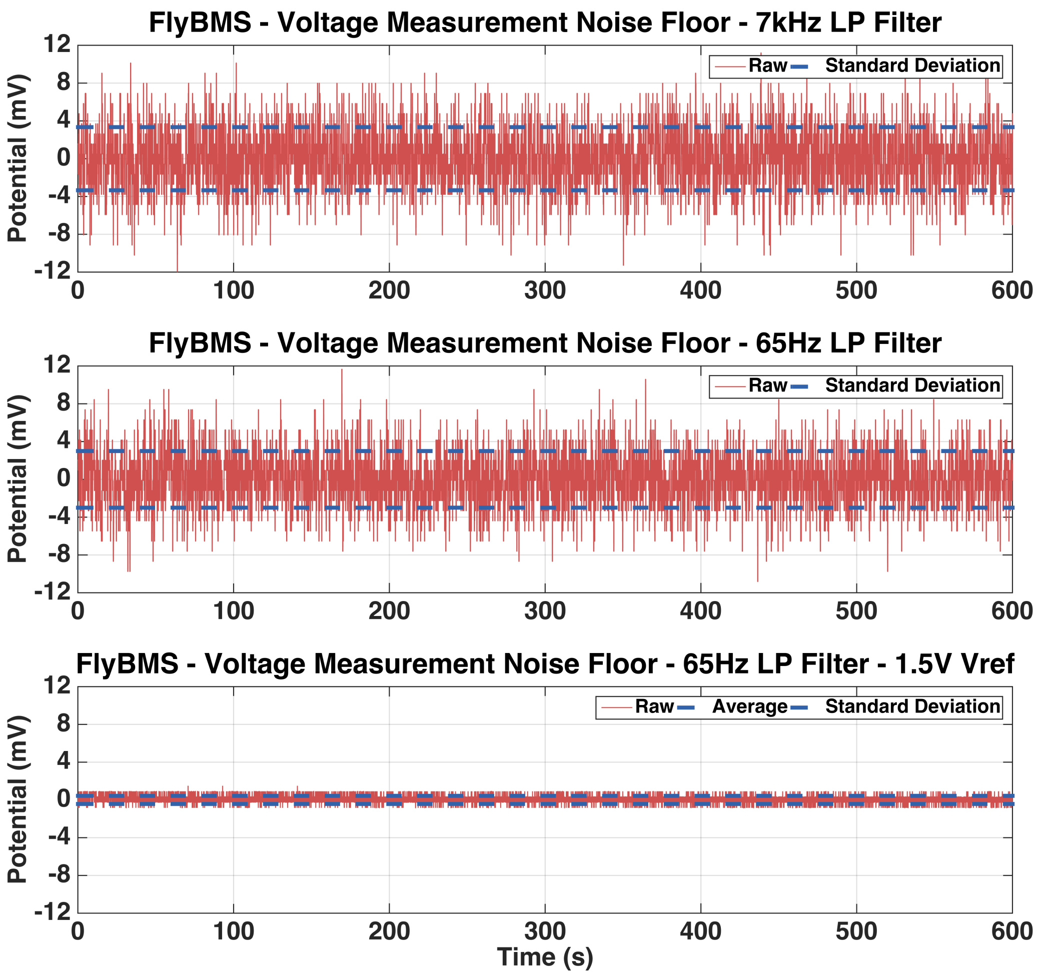


Figure 5.1: ADC noise floor analysis before and after filtering and external references

order, ie. cells AB, CD, EF etc... were placed in parallel. It should be noted too that, per the manufacturer, the original capacity and ESR for each cell listed was 4Ah and 15mΩ respectively, which shows exactly how poor the condition of the cells is. Figure 5.2 show the final constructed battery pack.

For the first full scale test with the constructed pack, in order to ensure that the test was consistently structured in order to facilitate comparisons, the pack was configured with a 1Ah initial discrepancy. Every cell was charged to 100 percent

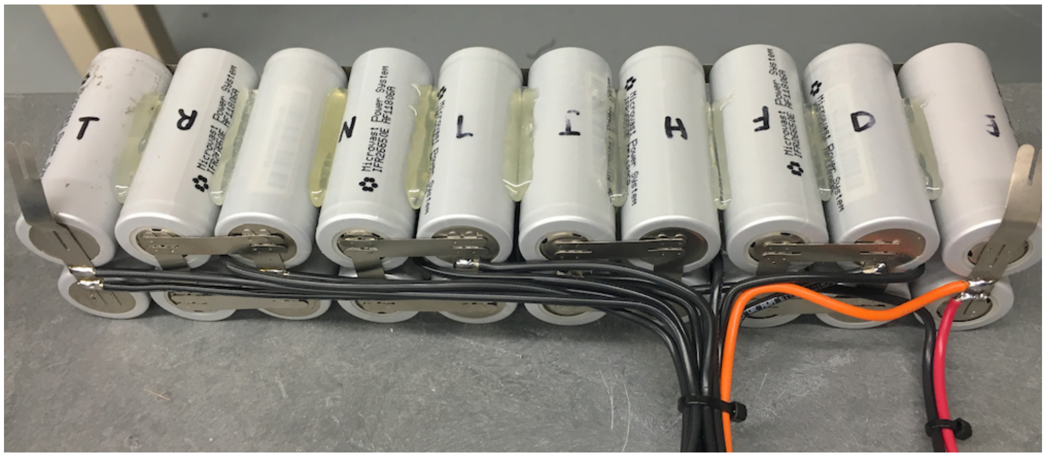


Figure 5.2: Battery module constructed for testing

SOC independently and then the first parallel pair in the module had exactly 1Ah discharged from it. To charge and discharge the battery module, a iCharger 4010 duo combination charger/discharger capable of cycling up to 10-cell battery packs at up to 40A was used. The iCharger was supplied by a 100Ah 12V valve-regulate lead acid battery. The test setup is shown in Figure 5.3. The test was performed at a rate of 4A for charge and discharge, with the charge and discharge cutoff voltages at 3.6V and 2.5V respectively. The test performed was 3 full cycles, both without and with balancing active.

As can be seen from the results in Figure 5.4, the balancing system is extremely effective. A typical passive balancing system would not have even starting correcting the SOC imbalance until nearing the end of charge, and most traditional low power active topologies would not have been able to immediately begin correcting the SOC imbalance upon start of the first discharge. FlyBMS was not only able to start correcting the cell SOC imbalance as soon as the first discharge started, but balance was reached before the first full cycle was completed, and the charger did not have to hold the pack in a constant voltage mode while the BMS corrected cells, which is typically

Table 5.1: Measured cell parameters for battery pack used in testing

Cell Designator	Capacity (Ah)	ESR (mΩ)
A	2.25	26.3
B	2.71	25.9
C	2.70	26.6
D	3.25	25.9
E	2.57	28.1
F	2.75	29.1
G	2.83	27.7
H	2.13	26.0
I	2.96	26.9
J	2.47	28.6
K	3.01	29.4
L	2.66	29.1
M	3.55	27.2
N	2.31	26.9
O	2.82	26.8
P	2.56	26.9
Q	2.29	27.1
R	2.59	25.9
S	3.11	26.1
T	2.69	28.3

an unavoidable delay with most BMS systems. FlyBMS was able to remove a total of 1Ah of equivalent energy from the remaining 9 parallel cell pair's in the module and move it into the first parallel cell pair within 2 hours, and that doesn't even account for the other system dynamics that it accounted for simultaneously. Note also that the test length was exclusively defined by the completion of three full cycles, and with the BMS active, the module took approximately 1.1 hours longer to complete the 3 cycles, indicating that the effective capacity of the battery module was increased by 13 percent by the end of the final cycle. In a real system, this would directly translate to longer run times, less aging due to cycling, and a more distributed aging profile between the cells in the pack.

5.3 Future advancements

Transformer Protection: Many improvements should and will be made to the hardware of the FlyBMS in the second revision. The first of these improvements involves analog over-current protection on the primary side of the transformers. During testing, caution had to be meticulously placed on the system's state during power up and/or programming of the MCU. If the balancing was active when the BMS was reset or when the programmer pulled the reset pin of the MCU low, the PWM module would momentarily hang, which sometimes resulted in random channels locking on. The flyback converters were constructed to maximize the transformer's capabilities, and as such, the transformers are very susceptible to saturation if the on time is stretched out any longer than the calculated maximum. Saturation of the transformer on the primary causes a collapse of the voltage on the primary side, which results in a reduced voltage to drive the gate. High drain-source currents coupled with critically low gate voltages would often result in catastrophic MOSFET failures due to power dissipation in the MOSFET. To remedy this, the plan of action is to drive the enable pin of the MOSFET driver IC with a comparator. This comparator will compare the voltage developed across a series sense resistor with that of a predefined reference (most likely a voltage divider), and will pull the enable pin low if the current is too high. To keep parts count low, the SMD fuse for each channel will be used as the series sense resistor, and the pull up on the gate pin will be used as part of the voltage divider generating the reference voltage, such that the reference voltage is only low when the driver is being commanded on. This nets a total of only two additional parts per channel.

Output Voltage Detection: Testing of the system also yielded another key flaw in the hardware and software. If any of the flyback converters are enabled when the cells are connected, and the output of the board is not connected to something that can accept or dissipate the energy transferred, the energy moved to the secondary side of the transformer is moved into the output filter capacitors of each converter. This intermediate output bus is unclamped, so the energy transferred into it will easily cause the voltage on these capacitors to rise well above their voltage ratings, as well as the ratings of the output rectification diodes, which are usually the point of failure in this event. To prevent this, voltage detection should be added to the pack output port such that an external pack connection can be monitored for, and balance can only be enabled if a pack is connected. This would also allow direct measurement of pack voltage for future added functionality, so that CPU overhead does not need to be wasted on summing the voltages of the other packs.

Isolated CAN: Another addition/correction to the board is an isolated CAN transceiver. The CAN driver in the HAL was never actually tested or optimized because the hardware necessary to develop and test the CAN high-level protocol necessary for CAN communication was not available, so that implementation is on the development timeline. The CAN transceiver also had a minor layout issue in that the Altium part was generated using a package for an older revision of the TI CAN transceiver, and as such, has the power rails swapped. The CAN transceiver also need to have an isolation boundary, which will be implemented using the same digital isolator IC as the FTDI chip isolation implementation.

Charge/Discharge Control: Another key functionality of a BMS system is the ability to not only manage the balance of the cells, but to control the charge and discharge state of the battery pack. For example, if the charging system in an electric vehicle fails, and the pack is being over-charged, the last line of defense is for the BMS to recognize over-charge on the cells and physically disconnect the battery module. This is typically handled by building a pair of isolated relay driver stages into the BMS, one for a charge relay and one for a discharge relay, such that the BMS can physically control the connection of the battery module to a the charger and/or a load.

Extensibility: The last feature to be incorporated is an inter-board communication protocol. This will be implemented via the CAN bus, but the key aspects that need to be developed are in the firmware. Because FlyBMS can only move energy around the pack in an efficient manner, there is no way to intentionally dissipate energy as heat within the BMS. For inter module balancing, this is a fantastic feature, but when one entire module's SOC is greater than another, there is no outlet to increase or decrease the overall SOC of one of the modules so that the two modules can be matched. This feature is still being developed.

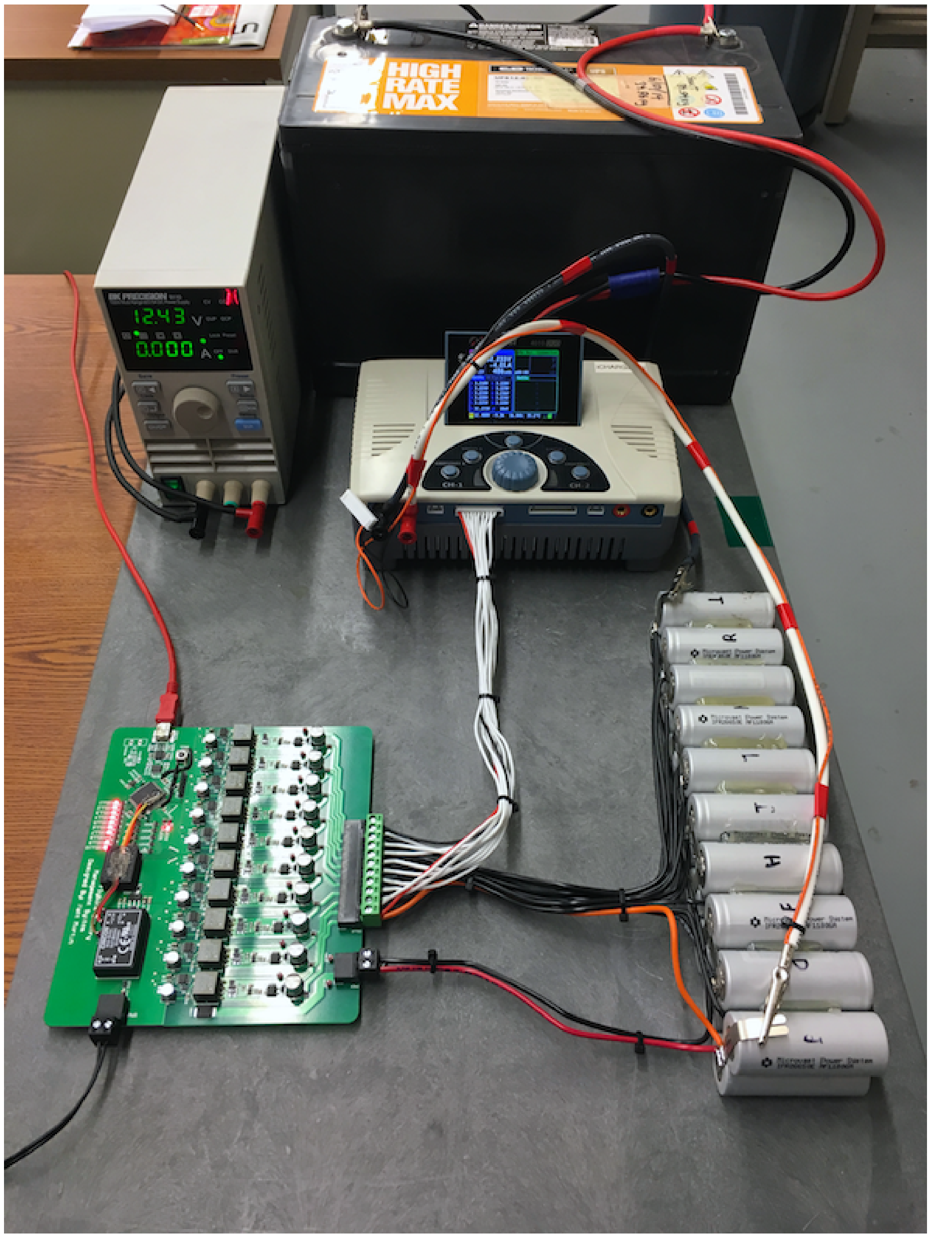
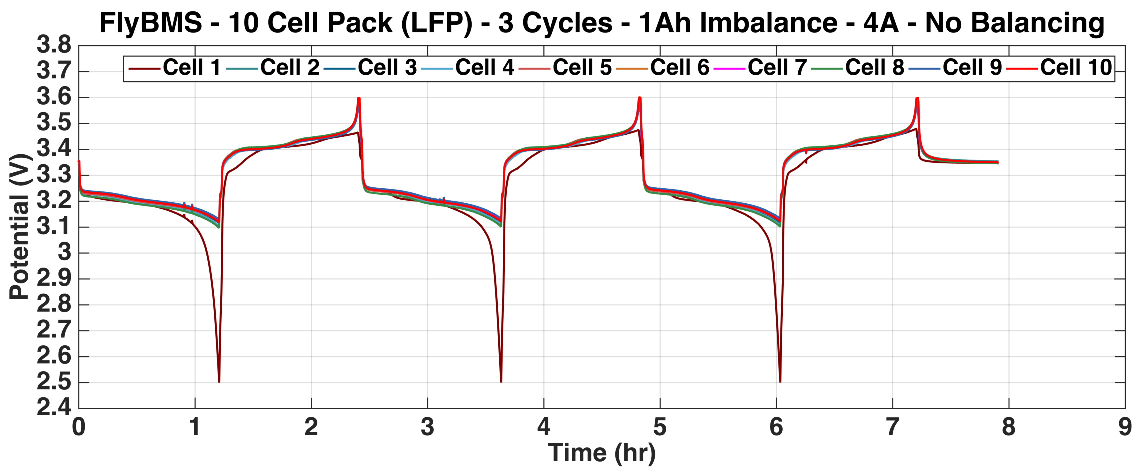


Figure 5.3: Test setup showing FlyBMS, battery module, and iCharger



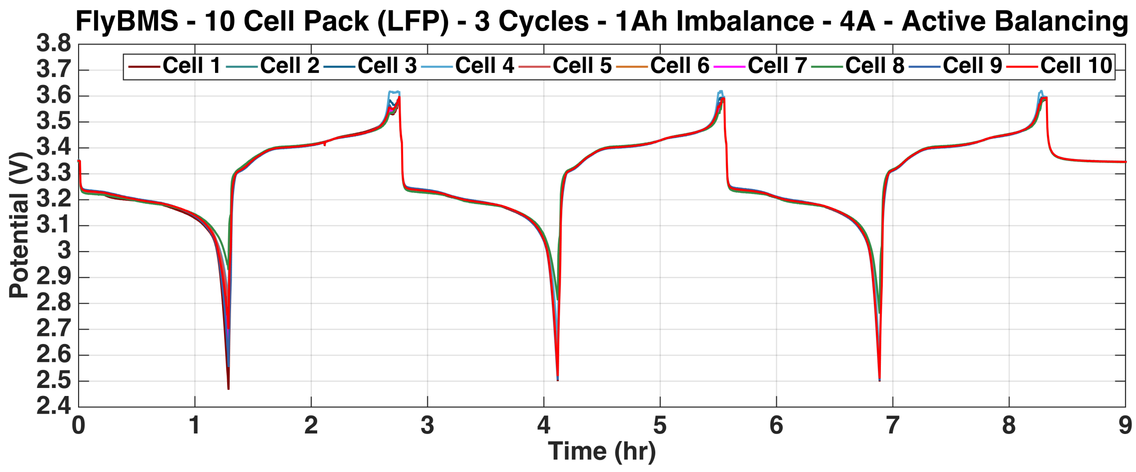


Figure 5.4: FlyBMS initial balance comparison results

APPENDIX A

Firmware: Balance State Machine

```
static int VCell[10] = {0,0,0,0,0,0,0,0,0,0};
static int Duty[10]  = {0,0,0,0,0,0,0,0,0,0};
static int stateBMS = state_BMS_DISABLE;
static int VMin = 0;
static int VMax = 0;
int Enable = 0;
int bManualMode = 0;
int DutyIn = 0;
bool bPrintToTerminal = false;

void osTask10ms(void)
{


    if(bPrintToTerminal)
    {
       // LED1 is the state machine heartbeat
        LED1 = !LED1;

        // LED2 is the status of the balance enable flag
        LED2 = Enable;

        // Print terminal output
        double time  = sysTime();
        #ifndef LOG
```

```
termSetCursor(9,1);
#endif
printf("%8.2f,%8d,%8d,%8d,%8d,%8d,%8d,%8d,%8d,
        %8d,%8d,%8d,%8d,%8d,%8d,%8d,%8d,%8d,
        %8d,%8d,%8d,%8d,%8d,%8d,%8d\n",
        time,
        Enable,
        bManualMode,
        Duty[0],
        Duty[1],
        Duty[2],
        Duty[3],
        Duty[4],
        Duty[5],
        Duty[6],
        Duty[7],
        Duty[8],
        Duty[9],
        VCell[0],
        VCell[1],
        VCell[2],
        VCell[3],
        VCell[4],
        VCell[5],
        VCell[6],
                VCell[7],
                VCell[8],
                VCell[9],
                VMin,
                VMax);
    bPrintToTerminal = false;
}

static int enableCount = 0;
switch(stateBMS)
{
    case state_BMS_DISABLE:
    {
        // Turn balance LEDs off
        LEDCELL1 = LEDCELL2 = LEDCELL3 = LEDCELL4 =
        LEDCELL5 = LEDCELL6 = LEDCELL7 = LEDCELL8 =
        LEDCELL9 = LEDCELL10 = 0;

        if( !bManualMode )
        {
            // Turn all PWM channels off
            disableOCPWM(1);
            disableOCPWM(2);
            disableOCPWM(3);
            disableOCPWM(4);
            disableOCPWM(5);
            disableOCPWM(6);
            disableOCPWM(7);
            disableOCPWM(8);
            disableOCPWM(9);
            disableOCPWM(10);
        }

        // Wait 10 ms and then go read the voltages
        stateBMS = state_BMS_READ;
        break;
    }

    case state_BMS_READ:
    {
        // Read analog inputs every loop regardless
        // of machine state
        analogConv();
        VCell[0] = analogRead(VMON1);
        VCell[1] = analogRead(VMON2);
        VCell[2] = analogRead(VMON3);
        VCell[3] = analogRead(VMON4);
        VCell[4] = analogRead(VMON5);
        VCell[5] = analogRead(VMON6);
        VCell[6] = analogRead(VMON7);
```

```
VCell[7] = analogRead(VMON8);
VCell[8] = analogRead(VMON9);
VCell[9] = analogRead(VMON10);

// Find minimum and maximum
VMax = getMax( VCell[0] , VCell[1] , VCell[2] ,
               VCell[3] , VCell[4] , VCell[5] ,
               VCell[6] , VCell[7] , VCell[8] ,
               VCell[9] );
VMin = getMin( VCell[0] , VCell[1] , VCell[2] ,
               VCell[3] , VCell[4] , VCell[5] ,
               VCell[6] , VCell[7] , VCell[8] ,
               VCell[9] );

// Check if we are in manual mode and calculate
// duty if not
if( !bManualMode )
{
    // Calculate each duty
    Duty[0] = getDuty( VCell[0] , VMin);
    Duty[1] = getDuty( VCell[1] , VMin);
    Duty[2] = getDuty( VCell[2] , VMin);
    Duty[3] = getDuty( VCell[3] , VMin);
    Duty[4] = getDuty( VCell[4] , VMin);
    Duty[5] = getDuty( VCell[5] , VMin);
```

```
        Duty[6] = getDuty( VCell[6] , VMin);
        Duty[7] = getDuty( VCell[7] , VMin);
        Duty[8] = getDuty( VCell[8] , VMin);
        Duty[9] = getDuty( VCell[9] , VMin);
    }

    // Switch states
    stateBMS = state_BMS_ENABLE;
    enableCount = 0;
    bPrintToTerminal = 1;
    break;
}

case state_BMS_ENABLE:
{
    // Check if we are in manual mode and manually
    //assign duty cycle
    if( bManualMode )
    {
        // Manually assign each duty cycle with
        // UART input
        Duty[0] = DutyIn;
        Duty[1] = DutyIn;
        Duty[2] = DutyIn;
        Duty[3] = DutyIn;
    Duty[4] = DutyIn;
    Duty[5] = DutyIn;
    Duty[6] = DutyIn;
    Duty[7] = DutyIn;
    Duty[8] = DutyIn;
    Duty[9] = DutyIn;
}

// Update duty cycle for each channel
// and enable PWM
setDuty(1,Duty[0]);
setDuty(2,Duty[1]);
setDuty(3,Duty[2]);
setDuty(4,Duty[3]);
setDuty(5,Duty[4]);
setDuty(6,Duty[5]);
setDuty(7,Duty[6]);
setDuty(8,Duty[7]);
setDuty(9,Duty[8]);
setDuty(10,Duty[9]);



if(Enable==1)
{
        // Set duty for each channel and update
        // it's respective LED
        if(Duty[0] != 0)
          { LEDCELL1 = 1; enableOCPWM(1); }
        if(Duty[1] != 0)
          { LEDCELL2 = 1; enableOCPWM(2); }
        if(Duty[2] != 0)
          { LEDCELL3 = 1; enableOCPWM(3); }
        if(Duty[3] != 0)
          { LEDCELL4 = 1; enableOCPWM(4); }
        if(Duty[4] != 0)
          { LEDCELL5 = 1; enableOCPWM(5); }
        if(Duty[5] != 0)
          { LEDCELL6 = 1; enableOCPWM(6); }
        if(Duty[6] != 0)
          { LEDCELL7 = 1; enableOCPWM(7); }
        if(Duty[7] != 0)
          { LEDCELL8 = 1; enableOCPWM(8); }
        if(Duty[8] != 0)
          { LEDCELL9 = 1; enableOCPWM(9); }
        if(Duty[9] != 0)
          { LEDCELL10 = 1; enableOCPWM(10); }
    }

    // Do not enable PWM if Enable flag is not set
```

```
            else
            {
                disableOCPWM(1);
                disableOCPWM(2);
                disableOCPWM(3);
                disableOCPWM(4);
                disableOCPWM(5);
                disableOCPWM(6);
                disableOCPWM(7);
                disableOCPWM(8);
                disableOCPWM(9);
                disableOCPWM(10);
            }

            // Tick the state counter and switch state
            // when timer expires
            enableCount++;
            if(enableCount == 48)
              { stateBMS = state_BMS_DISABLE; }
            break;
        }
    }
}
```

REFERENCES

[1] K. Cheng, B. Divakar, H. Wu, K. Ding, and F. H. Ho, “Battery management system (bms) and soc development for electrical vehicles,” *IEEE Transactions on Industrial Electronics*, January 2011.

[2] K. Cheng, “Investigation of the storage energy for classical switched mode power converters,” *IEEE Proceedings - Electric Power Applications*, July 2003.

[3] *2016 Formula SAE Rules*, Formula SAE, May 2015.

[4] W. C. Lee, D. Drury, and P. Mellor, “Comparison of passive cell balancing and active cell balancing for automotive batteries,” September 2011.

[5] M. Gkda and M. Akbaba, “An active battery cell balancing topology without using external energy storage elements,” May 2015.

[6] S. Jeon, J. J. Yun, and S. Bae, “Active cell balancing circuit for series-connected battery cells,” June 2015.

[7] K. M. Lee, Y. C. Chung, C. H. Sung, and B. Kang, “Active cell balancing of li-ion batteries using lc series resonant circuit,” *IEEE Transactions on Industrial Electronics*, Sept 2015.

[8] Y. Ye, K. W. E. Cheng, Y. C. Fong, X. Xue, and J. Lin, “Topology, modeling and design of switched-capacitor-based cell balancing systems and their balancing exploration,” *IEEE Transactions on Power Electronics*, June 2016.

[9] M. Y. Kim, C. H. Kim, J. H. Kim, and G. W. Moon, “A chain structure of switched capacitor for improved cell balancing speed of lithium-ion batteries,” *IEEE Transactions on Industrial Electronics*, August 2013.

[10] I. Aizpuru, U. Iraola, J. Mari, and A. Goikoetxea, “Comparative study and evaluation of passive balancing against single switch active balancing systems for energy storage systems,” May 2016.

[11] W. XueZhe, Z. Xiaopeng, and H. Dai, “The application of flyback dc/dc converter in li-ion batteries active balancing,” September 2009.

[12] P. Horowitz and W. Hill, *The Art of Electronics - Third Edition*. Cambridge: Cambridge University Press, 2015.

[13] *N-Channel 40 V (D-S) TrenchFET Low Gate Threshold MOSFET*, Vishay Siliconix, May 2013, rev. A.

[14] *Buffered Power Half Bridge*, Skyworks, July 2012.

BIOGRAPHICAL STATEMENT

Matthew Martin was born in Fort Worth, Texas in 1988. He recieved his B.S. degree in Electrical Engineering from the University of Texas at Arlington, as well as his M.S. in Electrical Engineering. He spent three years as a graduate research assistant in the Pulsed Power and Energy Lab at the University of Texas at Arlington, developing many systems and performing controlled tests on many different types of batteries. He also served as chief electronics engineer on UTA's Formula SAE team for an all electric race car, designing the entire electronics package for the vehicle.