

# **Sensorless Control of a Bidirectional Boost Converter for a Fuel Cell Energy Management System**

Andy M. McLandrich

Thesis submitted to the Faculty of the Virginia Polytechnic Institute and State University  
in partial fulfillment of the requirements for the degree of

## **MASTERS OF SCIENCE** in Electrical Engineering

Approved:

---

**Jason Lai, Chairman**

---

**Dushan Boroyevich**

---

**Douglas Lindner**

August 13, 2003

Blacksburg, VA

Keywords: bidirectional boost, sensorless, voltage control, average current control, lead-acid batteries, fuel cell, DSP

Copyright 2003, Andy McLandrich

# **Sensorless Control of a Bidirectional Boost Converter for a Fuel Cell Energy Management System**

by  
Andy M. McLandrich

Electrical Engineering

## **Abstract**

Fuel cells have the potential to provide clean power for a variety of uses including stand-alone residential power. But to increase the acceptance of fuel cells for off-grid generation, the cost of the energy management system must be greatly reduced. Of the many ways to accomplish this, this paper looks at reducing cost through topology changes and elimination of current sensors. A dual 2.5kW non-isolated bidirectional boost converter is designed and analyzed. The various bidirectional boost topologies are compared on cost and ability to meet the specifications. A sensorless average current mode is designed, implemented and verified through testing in a low-cost fixed-point DSP. Both boost and buck modes are accurately modeled and voltage and current controllers are designed for good closed-loop response. The accuracy of the sensorless average current measurement is investigated in both modes of operation. A classical dual-loop controller is implemented in boost mode with the sensorless average current and in buck mode, a dual controller operating in either current or voltage mode is implemented. The design is verified through testing in boost and buck mode and it is shown that the results are acceptable.

## Acknowledgements

I would like to thank Dr. Jason Lai for giving me the opportunity to work in his lab the FEC project. Doing research in the area of future energy applications has been one of my goals and I appreciate him giving me the chance to meet that goal.

I also would like to thank my other committee members Dr. Dushan Boroyevich and Dr. Douglas Linder. Though I had only one power electronics class with Dr. Boroyevich, I found it to be one of the best ones of my career. He also was very helpful in convincing me to stick with my work and see it to a successful end. On the control side, I found multi-variable control with Dr. Lindner to be quite interesting due to design process on a real-world system.

Furthermore, I would like to acknowledge the other students on the 2003 Future Energy Competition team for all their hard work. I thank them for putting up with me during those countless long days in the lab. I would especially like to thank Chris Smith for helping me fix my hardware and for all his time helping me to understanding how to program the DSP. Many others in the FEEC lab deserve credit including Elton Pepa, Damian Urciuoli, Mike Gilliom, Amy Johnston, Xudong Huang, Tim Thacker, Mike Schenck and Joel Gouker. Without their hard work my thesis would not have come together as easily as it did.

Finally, a big thanks to my parents for supporting me during these past two years. Without their support in so many ways, this would not have come to a successful end. It has been nice living a bit closer to home and having the ability to spend a bit more time with them these last two years.

# Table of Contents

<b>Abstract</b> .....	II
<b>Acknowledgements</b> .....	III
<b>Table of Contents</b> .....	IV
<b>List of Figures</b> .....	VII
<b>List of Tables</b> .....	IX
Chapter 1 – Introduction.....	1
1.1 Motivation.....	1
1.2 Application.....	1
1.3 Objective and Outline.....	2
1.4 Background Information .....	3
1.4.1 Overview of Fuel Cell Technology.....	3
1.4.2 Harnessing Fuel Cell Power.....	4
1.4.3 Current Sensing.....	5
Chapter 2 - Design Overview .....	7
2.1 Energy Storage Options .....	7
2.1.1 Ultracapacitors .....	7
2.1.2 High-Side Batteries .....	8
2.1.3 Low-Side Batteries.....	9
2.2 Topology .....	11
2.2.1 Isolated vs. Non-Isolated Bidirectional Boost .....	11
2.2.2 Single Phase vs. Interleaved Phases.....	14
2.2.3 Continuous vs. Discontinuous Conduction Mode Operation.....	15
2.3 Topology Selection .....	16
Chapter 3 - Simulation and Design.....	18
3.1 Circuit Simulation and Calculations .....	18
3.1.1 Device Selection.....	18
3.2 Magnetics Design.....	20
3.2.1 Core Selection .....	20
3.2.2 Inductor Design.....	20
3.3 Current Sensing.....	22
3.4 Gate Drives.....	22
3.5 Capacitor Selection .....	22
3.5.1 Bus Capacitance .....	22
3.5.2 Input Capacitance.....	23
Chapter 4 - Control Design and Implementation.....	24
4.1 LEM Current Sensing and Control .....	25
4.2 Sensorless Average Current Mode Control.....	27
4.2.1 Diode On-time Sensor.....	28
4.2.2 Optocoupler Design Verification .....	30
4.2.3 Capture Unit Setup.....	33
4.2.4 Proper Scaling of Converted Times .....	35
4.2.5 Constant Calculation .....	36
4.2.6 Inductor Voltage Calculation .....	36
4.2.7 Switch Voltage Drop Correction.....	37

4.2.8	Calculation of the Actual Current from DSP $I_{avg}$ .....	39
4.3	Controller Design and Implementation .....	39
4.3.1	Derivation of Converter Transfer Functions .....	39
4.3.2	Digital Conversion of RC Filters and Modeling of One-Cycle Delay .....	46
4.4	Implementation of a Digital Compensator .....	46
4.4.1	Proper Scaling of Compensator Coefficients .....	48
4.5	Design of Digital Compensators .....	49
4.5.1	Design of Boost Mode Compensators .....	49
4.5.2	Design of Buck Mode Compensators .....	59
Chapter 5	- Experimental Results .....	67
5.1	Previous Iterations of the Design .....	67
5.2	Experimental Setup .....	68
5.2.1	Sensor Boards .....	69
5.2.2	Optocoupler .....	69
5.2.3	Gate Drive Board .....	69
5.2.4	Loads .....	69
5.2.5	Power Supplies .....	70
5.3	Inductor Tuning and Measurement .....	70
5.4	Boost Mode Open Loop Testing of Power Stage .....	71
5.5	Buck Mode Power Stage Testing .....	73
5.6	Capture and Average Current Testing .....	74
5.6.1	Phase 2 Average Current Measurement .....	77
5.7	Boost Dual-Loop Controller Testing .....	78
5.7.1	Light Load Testing .....	79
5.7.2	Large Load Boost Controller Testing .....	83
5.8	Buck Mode Experimental Results .....	88
5.8.1	Current Mode Control Reference Test .....	89
5.8.2	Buck Battery Charging Test .....	92
Chapter 6	- Conclusion .....	96
6.1	Summary .....	96
6.2	Future Considerations .....	97
Appendix A	- Pspice Average Models .....	98
A.1	Buck Gvod .....	98
A.2	Buck Gild .....	98
A.3	Boost Gvod .....	99
A.4	Boost Gild .....	99
Appendix B	- Boost Matlab Code .....	100
B.1	Boost Voltage Loop Design .....	100
B.2	Boost Current Loop Design .....	103
B.3	Boost Dual-Loop Controller Simulation .....	106
Appendix C	- Buck Matlab Code .....	108
C.1	Buck Current Loop Design .....	108
C.2	Buck Voltage Loop Design .....	111
Appendix D	- Boost Mode C Code .....	114
D.1	Main.c .....	114
D.2	VConst.h .....	120

D.3 VIComp.c .....	121
D.4 Capture.h .....	124
D.5 Capture.c.....	127
D.6 ADC.h.....	128
D.7 ADC.c.....	131
D.8 INIT.h.....	132
D.9 Init.c.....	134
D.10 IO.h.....	136
D.11 IO.c.....	137
D.12 PWM.h.....	139
D.13 PWM.c.....	145
D.14 STD.h.....	146
D.15 STD.c.....	147
Appendix E - Buck Mode C Code .....	148
E.1 Main.c .....	148
E.2 VIconst.h.....	154
E.3 VIComp.c.....	155
E.4 PWM.h.....	158
E.5 PWM.c.....	158
Appendix F – Optocoupler Schematic .....	159
<b>References</b> .....	160
<b>Vita</b> .....	163

## List of Figures

Figure 2.1 Fuel Cell System without Energy Storage .....	7
Figure 2.2 Fuel Cell System with High-Side Batteries .....	8
Figure 2.3 Fuel Cell System with Single 48V Low-Side Battery Pack .....	9
Figure 2.4 Fuel Cell System with Dual 48V Low-Side Batteries .....	10
Figure 2.5 Isolated Full-bridge Bidirectional Boost Converter.....	12
Figure 2.6 Non-isolated Single-Phase Bidirectional Boost.....	13
Figure 2.7 Dual Battery, Single Phase Bidirectional Boost .....	14
Figure 2.8 Dual-Phase Bidirectional Boost.....	15
Figure 2.9 Dual 48V Batteries, Dual Phase Bidirectional Boost .....	17
Figure 4.1 RC Filter on A/D Input .....	24
Figure 4.2 LEM Output Averaging Circuit.....	26
Figure 4.3 LEM and Averaging Circuit Output in Boost Mode .....	26
Figure 4.4 DCM Current Waveform .....	27
Figure 4.5 Boost Mode Optocoupler Test $V_o = 200V$ .....	31
Figure 4.6 Boost Mode Optocoupler Glitching $V_o$ Under 177V.....	31
Figure 4.7 Buck Mode Optocoupler Glitching at $V_o = 58.8V$ .....	33
Figure 4.8 Capture Interrupt on Alternating Switching Cycles .....	34
Figure 4.9 Capture Interrupt During Every Clock Cycle.....	35
Figure 4.10 DSP Code for Conversion from Clock Cycles to Signed Integer.....	36
Figure 4.11 Inductor Voltage Calculation in Buck Mode.....	37
Figure 4.12 C Code to Subtract $V_{CE-On}$ from Inductor Voltage .....	38
Figure 4.13 Boost Mode Control-to-Inductor Current at 160W and 2.5kW.....	43
Figure 4.14 Boost Mode Control-to-Output Voltage at 160 W and 2.5kW.....	44
Figure 4.15 Buck Mode Control-to-Output Voltage at 50 W and 400 W.....	45
Figure 4.16 Buck Mode Control-to-Inductor Current at 50 W and 1000 W.....	45
Figure 4.17 Direct II Form .....	46
Figure 4.18 Block Diagram of Digital Integrator.....	47
Figure 4.19 Block Diagram of the Digital Compensator Implementation.....	48
Figure 4.20 Boost Mode Dual-Loop Controller Block Diagram .....	50
Figure 4.21 Boost Uncompensated Open-loop Current Response.....	51
Figure 4.22 Boost Open Loop Current Response .....	52
Figure 4.23 Boost Open Loop Current Response at Min and Max Power.....	53
Figure 4.24 Boost Uncompensated Open-Loop Voltage Response.....	54
Figure 4.25 Boost Outer Loop Response .....	56
Figure 4.26 Boost Overall Loop Gain at 1500 W .....	57
Figure 4.27 Boost Overall Loop Response at Min and Max Power .....	58
Figure 4.28 Buck Uncompensated Voltage Response .....	60
Figure 4.29 Buck Open-Loop Voltage Response.....	61
Figure 4.30 Buck Open-Loop Voltage Response at Low and High Power .....	62
Figure 4.31 Buck Uncompensated Open-Loop Current Response .....	63
Figure 4.32 Buck Compensated Current Loop.....	64
Figure 4.33 Buck Open-Loop Current Response at Low and High Power.....	65
Figure 5.1 Boost Peak Power Test .....	72
Figure 5.2 Boost Peak Power Average Input and Output Voltage and Current.....	73

Figure 5.3 Buck Mode Power Test.....	74
Figure 5.4 Boost $I_{avg}$ Measurement Waveforms at 160W.....	75
Figure 5.5 Boost $I_{avg}$ Measurement Waveforms at 1500W.....	76
Figure 5.6 Boost Average Inductor and Input Current at 740W.....	77
Figure 5.7 Boost Average Inductor and Input Current at 1.5kW.....	78
Figure 5.8 Boost Transient Response from 160W to 320W.....	80
Figure 5.9 Boost Transient from 160W to 320W Smaller Time Scale.....	81
Figure 5.10 Boost Transient Response from 320W to 160W.....	82
Figure 5.11 Boost Transient from 320W to 160W Smaller Time Scale.....	83
Figure 5.12 Boost Transient Response from 788W to 1.57kW.....	84
Figure 5.13 Zoomed-In Boost Transient Response from 788W to 1.57kW.....	85
Figure 5.14 Boost Transient from 1.57kW to 788W.....	86
Figure 5.15 Boost Transient from 788W to 1.57kW Smaller Time Scale.....	87
Figure 5.16 Input Current and $I_{avg}$ During Boost Transient.....	88
Figure 5.17 Buck Command Step from 100 to 700 of $I_{ref}$ .....	89
Figure 5.18 Buck Current Command Dump from 700 to 100 of $I_{ref}$ .....	90
Figure 5.19 Buck Current Mode Charging at Peak Supply Power.....	92
Figure 5.20 Buck Mode Charging at 383W.....	93
Figure 5.21 Buck Mode Charging, Current to Voltage Mode Transition.....	94
Figure 5.22 Buck Mode Voltage Loop Charging at Low Power.....	95

## List of Tables

Table 1.1 Main FEC Design Requirements .....	2
Table 2.1 Comparison of Half-Bridge and Full-Bridge Isolated Topologies .....	12
Table 2.2 Comparison of Non-isolated Bidirectional Boost Topologies .....	16
Table 3.1 Dual-Phase Bidirectional Boost Circuit Parameters .....	18
Table 3.2 IGBT Device Comparison.....	19
Table 4.1 Bus and Battery Voltage Sensor Gain.....	24
Table 4.2 Bus and Battery A/D Input Filter Values.....	25
Table 4.3 Boost and Buck $V_{CE-On}$ Corrections vs. Collector Current .....	38
Table 4.4 Boost Current Compensator Response Summary .....	53
Table 4.5 Boost Current Compensator Scaled Coefficients and Scaling Factors .....	54
Table 4.6 Boost Voltage Compensator Scaled Coefficients and Scaling Factors.....	56
Table 4.7 Boost Overall Loop Response Summary .....	58
Table 4.8 Buck Voltage Compensator Response Summary .....	62
Table 4.9 Buck Voltage Compensator Scaled Coefficients and Scaling Factors.....	63
Table 4.10 Buck Current Compensator Scaled Coefficients and Scaling Factors .....	65
Table 4.11 Buck Current Compensator Response Summary .....	66
Table 5.1 Phase 1 and Phase 2 Inductor Values as Measured.....	70
Table 5.2 Boost Peak Power Test Inductor Current Results .....	72
Table 5.3 Boost Peak Power Efficiency Results .....	73
Table 5.4 Boost $I_{avg}$ Verification at 160W .....	75
Table 5.5 Boost $I_{avg}$ Verification at 320W, 740W and 1500W.....	76
Table 5.6 Buck Current Mode Reference Tracking Results .....	91
Table 5.7 Buck $I_{avg}$ Verification at 44W .....	95

# Chapter 1 – Introduction

## *1.1 Motivation*

There are many applications for bidirectional DC-DC converters ranging from spacecraft [1] to renewable energy applications such as electric vehicles [2] and fuel cell power generation. Each of these applications requires specific attention to be paid to cost, performance and reliability with fuel cell energy management systems being no different.

To effectively harness the energy of a fuel cell and create usable power for a home, some sort of power processing electronics and energy storage must be employed which are currently one of the main barriers to the acceptance of fuel cell technology.

It is desirable to know the current and thus the power in each stage of the system but to do this with sensors is cost-prohibitive. Thus to reduce the overall system cost while maintaining or increasing the performance and reliability, a DSP-based sensorless average current mode will be designed.

## *1.2 Application*

As mentioned previously, bidirectional DC-DC converters are widely used in a variety of applications and the sensorless average current scheme can be implemented in all of these. One of the more interesting applications is in providing on-site power to residential customers.

To meet the needs of an average household, the fuel cell system must be able to provide 5kW continuous and provide 10kW for times of heavy usage [3]. The fuel cell in this particular system is rated for 5kW peak power which means the other 5kW must be stored either in batteries or capacitors. To interface batteries to the fuel cell system, a bidirectional DC-DC converter must be used. Also the energy storage scheme must be able to provide transient power during load steps that are quicker than the slew rate of the fuel cell.

### 1.3 Objective and Outline

To reduce the cost and speed the adoption of fuel cells for residential applications, it is necessary to reduce the cost to that of current utility generation. The 2003 Future Energy Challenge is designed to advance the inception of Solid Oxide fuel cells (SOFC) for residential off-grid power generation [3]. One of the main costs of the fuel cell system is the fuel cell energy management system that creates AC voltage from the stack voltage. To help reduce the cost and improve performance of the voltage conditioning circuitry, the Department of Energy along with IEEE and the Department of Defense sponsored the 2003 Future Energy Challenge (FEC). The cost target is significantly lower than what is currently available, \$40/kW [3] as compared to \$1000-1500/kW for current SOFC [4].

One of the main design goals is an overall system efficiency of 90% with an emphasis placed on cost reduction and others as listed below from [3].

**Table 1.1 Main FEC Design Requirements**

<b>Design Item</b>	<b>Minimum Target Requirement 10 kW System</b>
1. Manufacturing cost	Less than US\$40/kW for the 10 kW design in high-volume production.
4. Output power capability – nominal	5 kW continuous
Output power capability – overload	Total 10 kW overload for 1 minute 5kW from SOFC, 5kW from batteries
10. Output voltage regulation quality	120V $\pm$ 6% over the full allowed line voltage and temperature range, from no-load to full-load.
13. Battery auxiliary power <sup>1</sup>	48 V dc nom. +10%-20%, with nominal rating of 500 Whr.
28. Lifetime	Sixteen years with routine maintenance when subjected to normal use in a 20°C to 40°C ambient environment.
31. Control power	The power conversion system must draw all its power from either the fuel cell or the battery pack.
32. Galvanic isolation	Galvanic isolation of the system is encouraged. The common neutral point of the ac output phases must be earth grounded.

To achieve these cost and efficiency targets, the hardware must be designed from the bottom up, eliminating expensive components with cost-effective solutions. The fuel cell energy management system has three main parts the front-end DC-DC converter, the AC inverter and energy storage elements. This paper will focus on the energy storage elements and the interfacing them to the inverter system.

To reduce overall system cost it is important to minimize expensive components whenever possible. These come in a variety of forms including off-the-shelf DC-DC converters, device packages and modules, Hall-effect current sensors and digital controllers/DSPs. The objective of this paper is to design and build a low-cost bidirectional converter using discrete components and eliminate current sensors through a DSP-based sensorless average current scheme. In doing this, the overall performance of the system must be maintained and the lifetime and reliability meet the specifications in Table 1.1. Chapter 2 will investigate in-depth the possible energy storage options and then compare different bidirectional boost topologies based on device count, component size, overall cost and circuit operation. Chapter 3 will cover the design of the power stage including the selection of devices and the design of the magnetic components. In Chapter 4 the converter transfer functions will be derived and the sensorless average current mode will be designed and implemented in the DSP. Also in Chapter 4, the dual-loop boost controller will be designed along with both the current and voltage mode buck controllers. These will be converted to the digital domain and also implemented in the DSP. Chapter 5 will cover the experimental results of both boost and buck mode, verifying the sensorless average current design and the closed-loop performance. Chapter 6 will provide a summary of the results and explore future work to improve the sensorless average current and overall system performance.

## ***1.4 Background Information***

### **1.4.1 Overview of Fuel Cell Technology**

There are currently five main types of fuel cells: Proton Exchange Membrane (PEMFC), Phosphoric Acid (PAFC), Molten Carbonate (MCFC) and Solid Oxide (SOFC). All of these types share the same basic theory of operation but are very

different in their construction and applications [4]. Fuel cells require two input chemicals, the fuel and the oxidant. The most attractive fuel is hydrogen as it does not produce any harmful byproducts, only water. This allows oxygen to be used as the oxidant, which is readily available in the atmosphere [5]. Inside a fuel cell, there are three layers, the anode, electrolyte and the cathode. Each type of fuel cell uses different materials for each of these layers, but the basic operating principle remains the same. The fuel is injected into the anode side and is reduced to positive ions and free electrons by the anode. The free electrons flow in the opposite of the positive ions, exit the cell and are used by external circuitry to provide power. The positive ions flow through the electrolyte and when they reach the cathode, they are reduced through another chemical reaction with the oxidant. This reaction recombines the positive ions with the free electrons that return from the external circuitry [4].

Theoretically, one of these cells can produce around 1.18V [6], but when under load, the voltage drops to around 0.7V [5]. At such a low voltage, it is hard to harness the power for residential consumption. To solve this multiple cells are stacked in series, effectively increasing the voltage and the power.

The general efficiency of fuel cell systems is in the range of 40-50% and can be increase up to 85% by using excess heat and steam to drive turbines for cogeneration of electricity [7]. Along with efficient and reliable operation, fuel cells have drastically reduced emissions of greenhouse gases and other harmful hydrocarbons. The fact that they are relatively quiet as compared to a power plant, allows them to be located on-site in residential areas or near businesses [4].

#### **1.4.2 Harnessing Fuel Cell Power**

To efficiently use the electricity created, power electronics are employed to transform the energy into a usable form for a residential customer. Most residential customers require single-phase 120V and 240V 60Hz AC. This requires the low fuel cell voltage, on the order of 23-34V, to be boosted to 400V and then inverted into a 60Hz sinusoid. There are many methods in which to do this which are outside the scope of this paper.

A complicating factor is that the SOFC has a very slow slew rate given in [3] as 200W/minute. The external mechanical pumps and valves that regulate the flow of fuel to the stack determine the slew rate. During a load step, the inverter cannot pull more power from the fuel cell than is currently available so supplemental power must be provided by some sort of energy storage elements. There are many options for these storage elements, which will be discussed further in the following sections.

### **1.4.3 Current Sensing**

To accurately and efficiently control the fuel cell inverter system, the power in each stage must be known to an acceptable level of precision. The load on the fuel cell must remain below the slew rate limit at all times so as not to damage the stack. This requires the bidirectional boost to provide any additional power to the high-voltage bus. To control this power flow, the current in and out of the high-voltage bus must be known. There are many ways to do this including using LEMs to measure the current directly, using a current sense resistor on the high-voltage side or to use an observer method to recreate the current.

LEM's are a good choice if cost is not an issue. They are very accurate and have good noise immunity. They would be placed in series with one inductor in each converter which would allow for the use of a dual-loop control scheme. Since the converter is operating in DCM, the current signal must be averaged before conversion in the DSP.

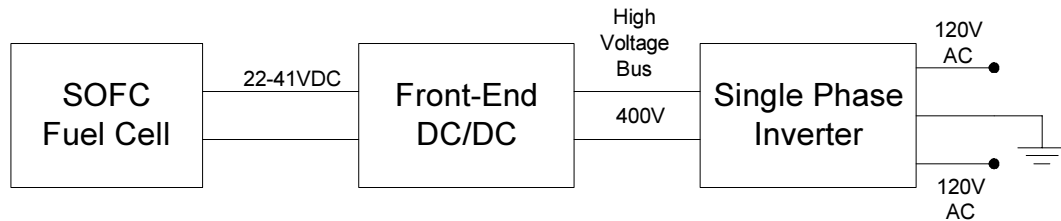
Current sense resistors can be used on the low current side but provide no information about the inductor current. They can be used to measure the power in and out of the bus for management and control, but cannot be used for an inner control loop. To measure the very small signal across the sense resistor requires additional voltage sensors which are very hard to implement due to the large amount of switching noise.

The final option is to use an observer method to recreate the inductor information from measurable states. Sensorless current mode (SCM) control was proposed and implemented in [8] using an observer method. In this control scheme, the voltage across the inductor is observed, integrated and compared against a ramp to create a PWM signal. This method allows for precise control of the inductor current, but does not provide a DC

current value. For the inverter application, it is preferable to know the DC current thus the average current must be computed. Using the method in [8] it is not possible to know the average current so a modification must be made to the observer method which will be explained in Chapter 4.

## Chapter 2 - Design Overview

Figure 2.1 is a block diagram of the basic fuel cell system without energy storage elements for providing peak power.



**Figure 2.1 Fuel Cell System without Energy Storage**

To provide transient and peak power as the fuel cell ramps up some kind of energy storage elements must be included in the inverter system. There are three options available, mainly ultracapacitors, high-side batteries and low-side batteries. Each of these options comes with advantages and disadvantages, which will be discussed in the following section.

### ***2.1 Energy Storage Options***

#### **2.1.1 Ultracapacitors**

Ultracapacitors are a relatively new technology that allows capacitors to maintain voltage over time as a battery does, but does not need any sort of charging scheme. They can be placed on the fuel cell input as done in [9] or placed directly across the high voltage bus. One of the benefits of using ultracapacitors is that they require no maintenance or extra circuitry and have a very long life. The drawbacks of this solution are that ultracaps are very expensive and currently their availability is limited. Since the slew rate of the SOFC fuel cell is very slow and the load step is relatively large, a large number of ultracaps will need to be used to supply the necessary energy. A 5kW load step for 60 seconds requires 300kJ of stored energy. Current ultracap technology can supply around 35kJ/capacitor [9].

### 2.1.2 High-Side Batteries

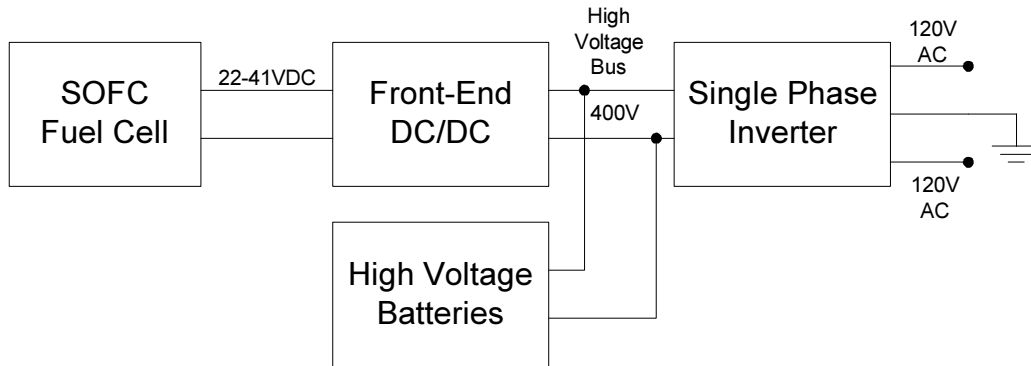


Figure 2.2 Fuel Cell System with High-Side Batteries

The high-side battery option involves placing a large number of 12V lead acid batteries in series across the high voltage bus as shown in Figure 2.2. Depending on the topology of the inverter, the bus may be a dual 200V or a single 400V bus, which requires at least 16 batteries and 32 batteries respectively [10]. As the number of batteries increases, so does the ESR of the series connection. With this large number of cells in series there may be severe unbalancing and loss of performance over time of the individual cells as shown in [11]. With high-side batteries, there cannot be an intelligent management scheme without adding additional cost to the system. During heavy load conditions and large currents, the ESR of the series connection will reduce the available voltage to the inverter. This problem will be exacerbated at times when the batteries are at a low state of charge. Another drawback to this scheme is that to supply auxiliary power for DSPs, gate drives and the fuel cell involves another switching converter.

### 2.1.3 Low-Side Batteries

#### Single 48V Pack

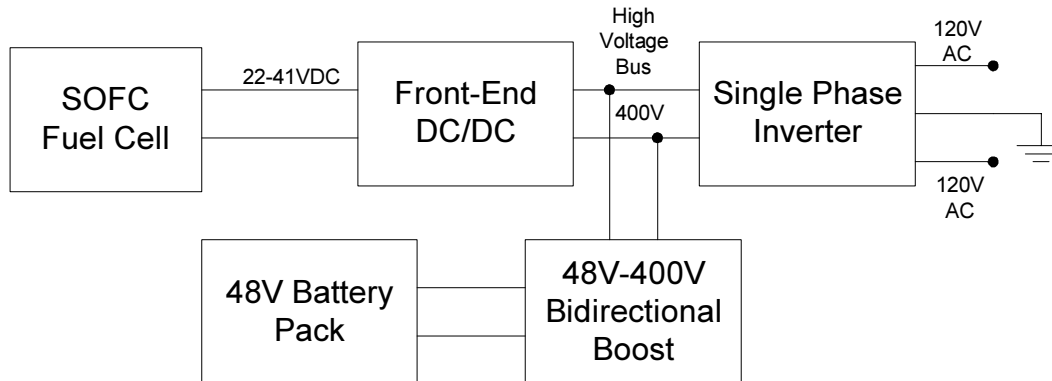
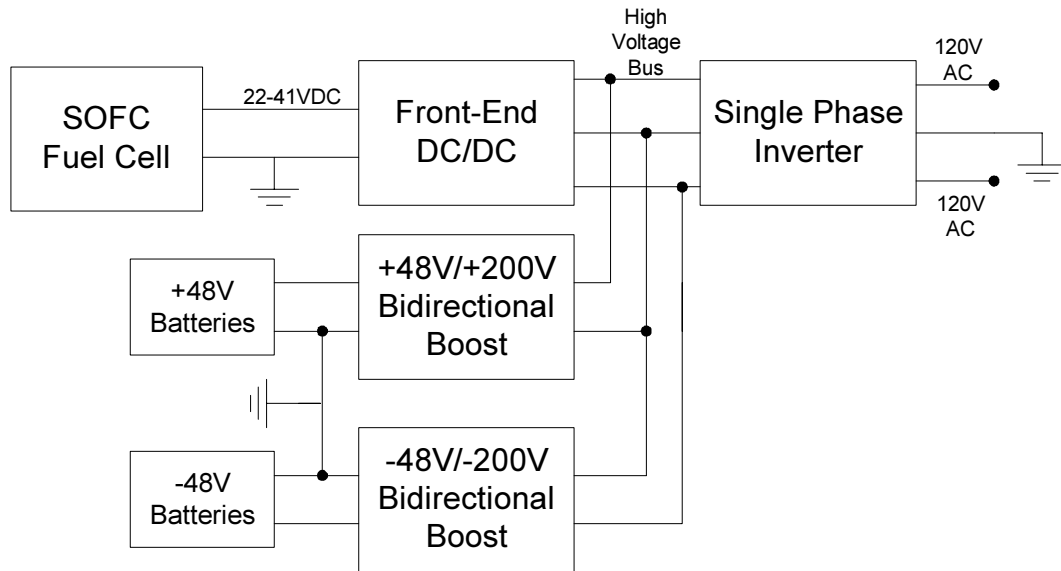


Figure 2.3 Fuel Cell System with Single 48V Low-Side Battery Pack

The remaining option is using lower voltage batteries interfaced to the high voltage bus through a bidirectional converter as shown in Figure 2.3. A bidirectional converter allows power to flow from the batteries to the high voltage bus or from the high voltage bus to charge the batteries. The voltage of this battery pack can be chosen to suit the particular design requirements. The particular advantages to this design are that the batteries can supply auxiliary power to the inverter and fuel cell upon start-up without requiring any additional circuitry. Also depending on the value of the voltage, the number of cells in series is greatly reduced, reducing the series resistance, cost and size. A small number of cells in series greatly reduces the unbalancing effects seen in larger arrays. The bidirectional converter allows for intelligent battery management, which can assure a long and efficient operation of the battery pack.

The drawbacks to this implementation are the added cost of the bidirectional converter and the associated control overhead. Another problem with this scheme is the large conversion ratio necessary to boost 48V to the 400V high voltage bus. This large ratio reduces the efficiency of the converter and requires the use of large devices and magnetic components. With a single HV bus, there can be imbalances in the inverter output stemming from the split-capacitor scheme. To solve these, a third phase leg must be included in the inverter to balance the bus.

## Dual +/- 48V Low-Side Batteries



**Figure 2.4 Fuel Cell System with Dual 48V Low-Side Batteries**

A variation on this design is to use two 48V battery packs providing  $-48\text{V}$  and  $+48\text{V}$  with respect to earth ground as shown in Figure 2.4. This would require two bidirectional converters boosting each battery voltage to  $-200\text{V}$  and  $+200\text{V}$ . This scheme would allow the fuel cell negative to be grounded to the battery midpoint and then tied to the inverter earth ground which prevents any floating of the fuel cell and properly grounds the batteries. By sharing the power in each converter, reduces the switch and magnetic component sizes. Another advantage is independent control of each bus voltage, eliminating balancing problems for the inverter.

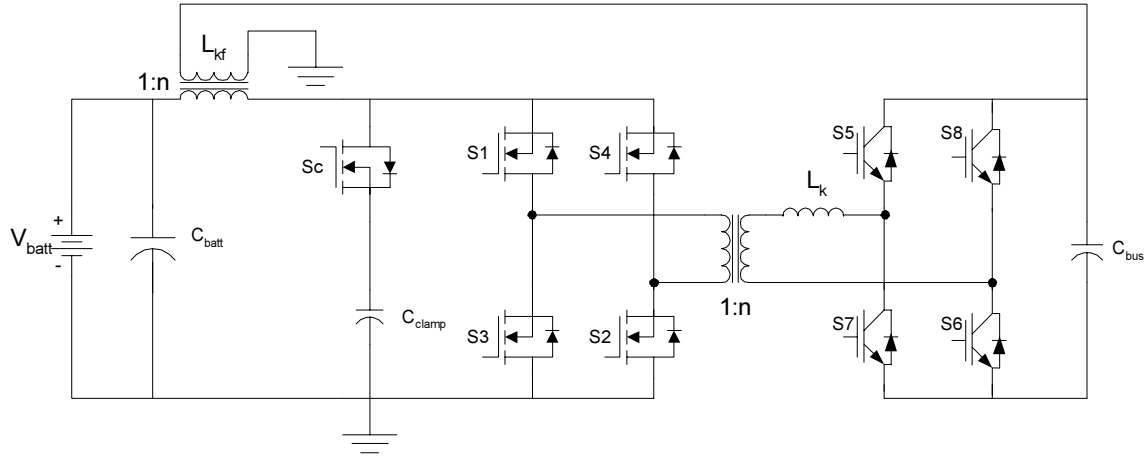
There is the added cost of another set of batteries and another converter, but all the components sizes, including the batteries, can be reduced by half. The control circuitry does increase as the number of gate drives, power supplies and sensors double.

## ***2.2 Topology***

The initial design was based around a single 48V battery pack and a single 400V bus. This was pursued using an interleaved phase structure until updated specs from the FEC were received. They specified that the fuel cell must be grounded to earth ground, which is not possible unless an isolated bidirectional is used. There are two options to solve this problem, one is to use an isolated bidirectional boost or split the HV bus and use two 48V packs as shown in Figure 2.4.

### **2.2.1 Isolated vs. Non-Isolated Bidirectional Boost**

There are three main converters that can be used on either side of the transformer, push-pull, full-bridge or half-bridge and can be either current or voltage fed [12]. Of these, only certain combinations are cost-effective and efficient depending on their placement, either on the low or high side. In [13] a preliminary analysis was done comparing the device ratings, turns ratio, device count and efficiency of a half-bridge, current-fed converter and a full-bridge voltage-fed converter on the low side. It was determined that it is neither cost-effective nor efficient to use a voltage-fed full-bridge on the low side. The two topologies selected for comparison were a half-bridge and a full-bridge, both current-fed on the low side incorporating a voltage-fed full-bridge on the high side of which the current-fed full-bridge is shown in Figure 2.5. From this analysis two prototypes were constructed and tested, measuring the efficiency of each.



**Figure 2.5 Isolated Full-bridge Bidirectional Boost Converter**

As shown in [13], the efficiency of the half-bridge is only 89% during peak discharging while the full-bridge achieves 94% under the same conditions. To achieve this efficiency with the full-bridge circuit, an active clamp was employed along with soft-switching that was not possible on the half-bridge. The device count for these converters is as follows in Table 2.1.

**Table 2.1 Comparison of Half-Bridge and Full-Bridge Isolated Topologies**

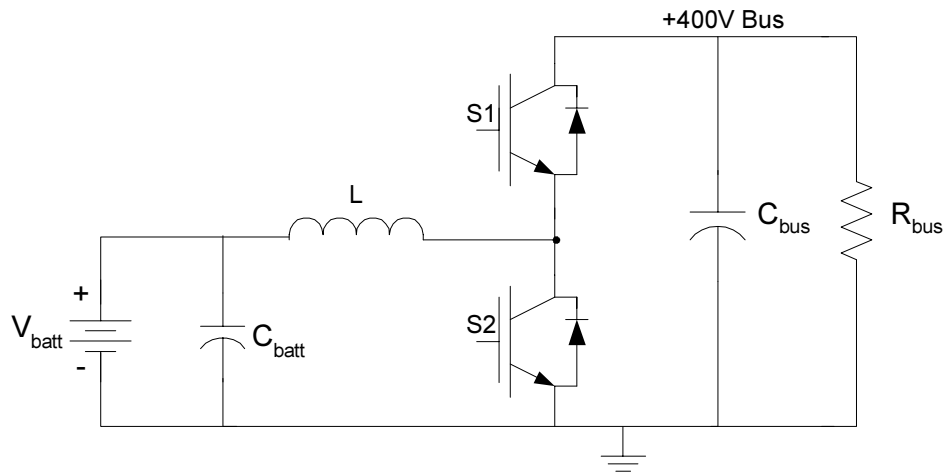
Converter	MOSFETs	IGBTs	Diodes	Inductors	Clamp Capacitors
Half-Bridge	2	4	2	2	1
Full-Bridge	5	4	0	1	1

In [14] a dual full-bridge bidirectional converter was built utilizing an active clamp and a unified soft-switching scheme as shown in Figure 2.5. The current-fed converter was placed on the low side and the voltage-fed converter was placed on the high side. To achieve as high efficiency as possible, the active clamp capacitor chosen must be very well suited to the particular application. It must have a very low ESR, high resonant frequency and due to board layout concerns, be physically small. The types of capacitors that meet these specs are multilayer ceramic (MLC) and metalized polymer capacitors from which MLC were chosen.

The prototype was tested over the full power range of 0-5kW and the efficiency approached 95% in both discharging and charging modes [14].

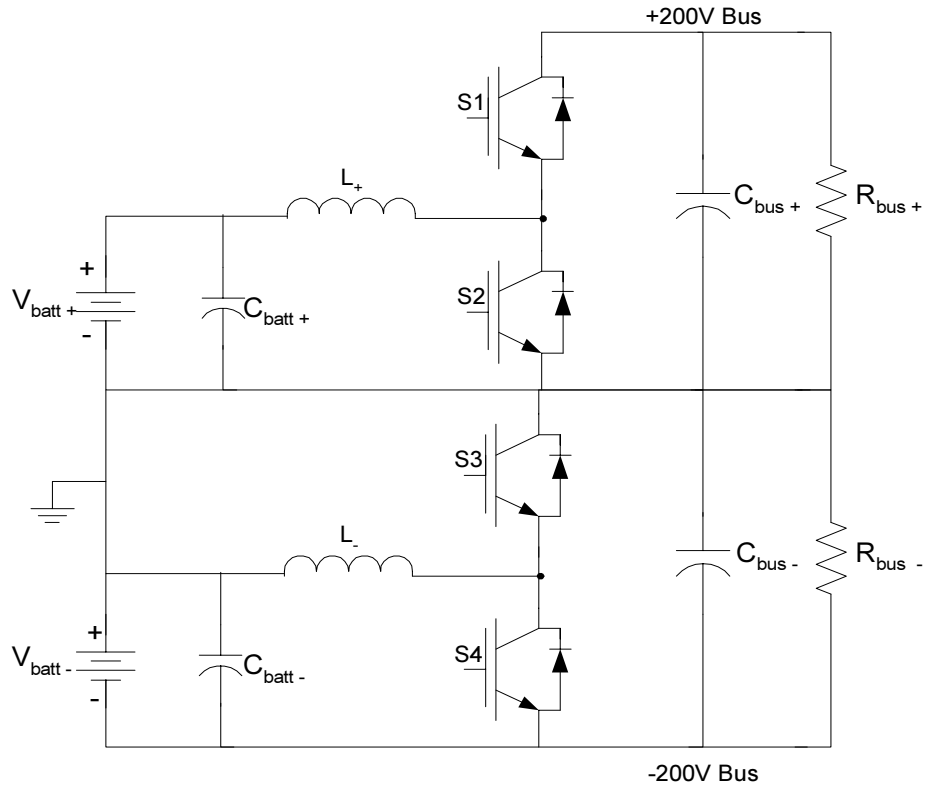
The efficiency is the main advantage of this design and it does allow the batteries to be grounded to the fuel cell negative. There are significant drawbacks to this topology, namely component count and cost. There are a total of nine active components, which adds significant cost in devices, board space and control overhead.

The other option is to use a non-isolated bidirectional boost circuit as shown in Figure 2.6.



**Figure 2.6 Non-isolated Single-Phase Bidirectional Boost**

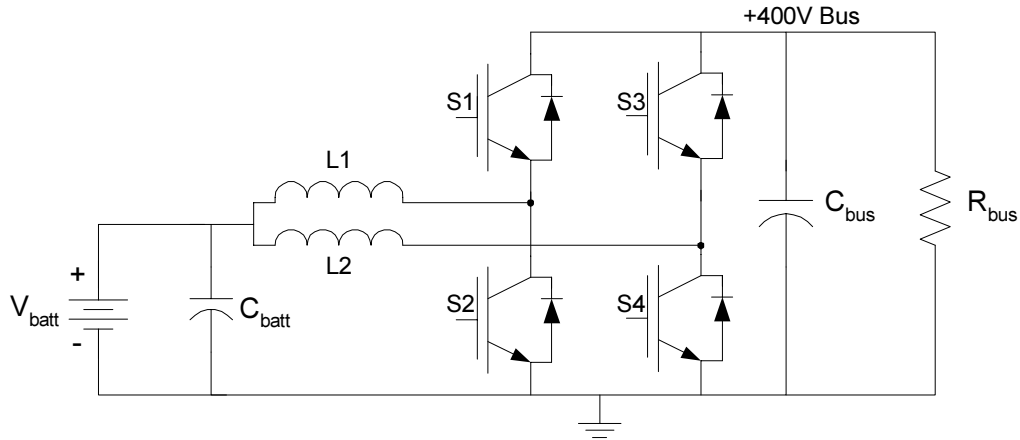
Since there is no isolation between the batteries and the high-voltage bus, to meet the necessary grounding requirements, two sets of 48V batteries must be used in conjunction with two bidirectional boost converters. The schematic of this topology is shown in Figure 2.7.



**Figure 2.7 Dual Battery, Single Phase Bidirectional Boost**

### 2.2.2 Single Phase vs. Interleaved Phases

A single-phase bidirectional boost is the simplest topology and contains two active switches and a single inductor. In boost mode at the peak power level of 2.5kW, the switch must carry the full power and likewise for the inductor. This requires the devices and magnetics to be quite large. To reduce the switch stress and the inductor size, another bidirectional boost circuit can be placed in parallel to share the power as shown in Figure 2.8.



**Figure 2.8 Dual-Phase Bidirectional Boost**

In this topology the switches are driven with the same duty-cycle, but the second leg is  $180^\circ$  out of phase with the first leg. This effectively doubles the frequency of the switching ripple, reducing the size of the filter capacitors. Since the power is shared between the two phases, the current ratings of the switches are reduced by two, as is the size of the inductors. The obvious drawbacks are added component count and the associated control but is partially offset by smaller devices thus less costly.

### **2.2.3 Continuous vs. Discontinuous Conduction Mode Operation**

For any converter there can be two distinct modes of operation, continuous current mode (CCM) and discontinuous current mode (DCM). In CCM the inductor current never falls to zero within a switching cycle but oscillates about a DC value. This has the advantage of reduced device stresses by lower peak currents and results in smaller switching ripple on the output. One drawback is the size of the inductor must be large to maintain the current during the switch off-time and increased switching loss due to the hard turn-on of the current.

In DCM operation, the inductor current falls to zero before the end of the switching cycle. If the converter is designed to operate in DCM, the same energy must be transferred each cycle which requires large peak currents. This puts added stress on the switches but significantly reduces the size and cost of the magnetic components. Also since the current is zero at the beginning of each switching cycle, there are reduced

losses from hard turn-on of the switch and the slow reverse-recovery problem of the body diode is eliminated. In high power DCM converters, there is an inherent problem with parasitic ringing between the switch output capacitance and the inductor, which cannot be eliminated and creates additional losses [15].

DCM will be the mode of operation used in the final implementation due to the reduction in inductor size, increased efficiency and preferable plant dynamics.

### 2.3 Topology Selection

The table below summarizes the different topologies that are under consideration. Using the DCM boost formulas from [16] and assuming a low-line battery under load of 45V with an efficiency of 90%, the following Table 2.2.

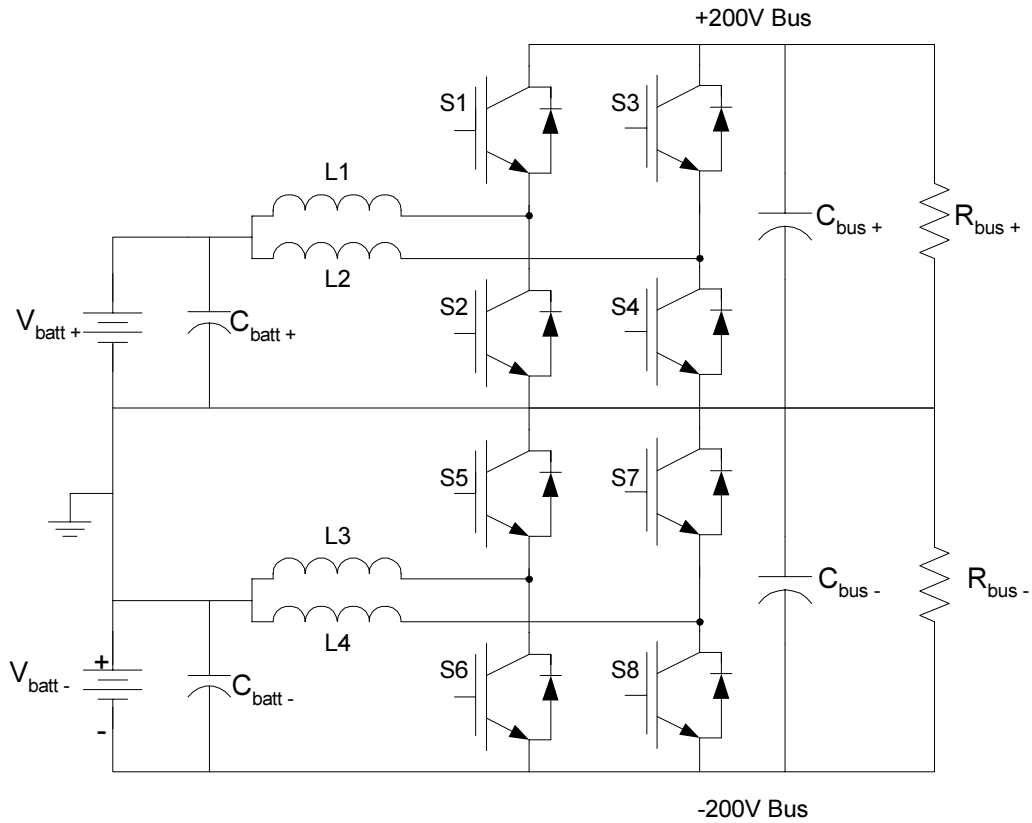
**Table 2.2 Comparison of Non-isolated Bidirectional Boost Topologies**

Device Stress	Single Phase +48 to 400V	Dual Phase +48 to 400V	Dual, Single-Phase Converters, -48V to -200V +48V to +200V	Dual, Two-Phase Converters +48V to +200V -48V to -200V
On-time	0.8783	0.8783	0.7677	0.7677
Switch Voltage Peak	400V	400V	200V	200V
Switch RMS Current	134A	67A	62A	31A
Switch Peak Current	247A	124A	123A	62A
Diode average Current	13.8A	6.9A	13.8A	6.9A
Inductor Value	8uH	16uH	14uH	28uH

As shown in Table 2.2, the single 400V bus topology requires very large devices and an increased on-time. A larger on-time results in a reduction in overall efficiency. The difference between an interleaved 400V design and a dual-supply, dual single phase converter is decreased on-time and decreased voltage stresses on the switches for the latter. The device count for both of these options is the same except the number of batteries increases. The final choice is  $\pm 48V$  batteries with dual interleaved phases as shown in Figure 2.9, which results in the lowest device stresses. Since cost is a main

design constraint, it is preferable to use discrete devices as packages or modules are expensive. Also large inductors are expensive, heavy and hard to mount.

This is the topology chosen for the final design,  $\pm 48\text{V}$  battery packs interfaced to a  $\pm 200\text{V}$  bus through two interleaved phases bidirectional boost circuits.



**Figure 2.9 Dual 48V Batteries, Dual Phase Bidirectional Boost**

## Chapter 3 - Simulation and Design

### 3.1 Circuit Simulation and Calculations

Most simulation of the circuit was performed in Matlab using methods from [16] and [17]. The circuit parameters as calculated are shown below in Table 3.1 using a switching frequency of 20 kHz at an output power of 5kW assuming 90% efficiency.

**Table 3.1 Dual-Phase Bidirectional Boost Circuit Parameters**

Duty Cycle	0.7677
Peak Switch Voltage	200V
RMS Switch Current	31A
Peak Switch Current	62A
Average Diode Current	6.8A
Inductor Value	28uH

#### 3.1.1 Device Selection

From these calculations, the switches can be selected. For switching frequencies around 20kHz, both MOSFETs and IGBTs can be used but due to the reverse recovery speed of the body diode of MOSFETs, IGBTs will be selected. Some IGBTs are manufactured with a high-speed anti-parallel diode, which will provide the passive switching required after the active switch turns off. There are wide selection of devices that can be used so they must be narrowed down according to power capability.

Since the bus is split into two 200V rails, the maximum device voltage should not exceed 300V. There are devices that are rated for 300V and above, but they are much more expensive than those made for 600V and since cost is an issue, the search will be limited to only 600V devices. Another constraint is that only discrete devices are to be used to lower costs.

The switches were chosen based on many design constraints including reverse operation, average current capacity, peak current capacity and allowable power dissipation.

The power dissipation of the device is also of major importance as there are many devices that can handle the current, but are not suited for this application. Some possible candidates are shown in Table 3.2.

**Table 3.2 IGBT Device Comparison**

Part Number	$V_{CE}$	$I_C$ @ 100°C	$I_C$ @ 20kHz	$I_C$ @ 10kHz	Peak Current	$P_{diss}$ @ 100°C
HGTG30N60B3	2.0V	30A	38A	52A	252A	83W
SKW30N60HS	3.15V	30A	42A	42A	112A	100W
IRG4PSC71UD	2.00V	60A	25A	37A	200A	140W
IRGP50B60PD1	2.35V	45A	N/A	N/A	150A	150W

The power dissipation was assumed to be 10% of 2.5kW or 125W/phase. This is the power dissipation each switch must be able to handle. From the datasheets, it can be seen that at 100°C only IRG4PSC71UD [18] and IRGP50B60PD1 can handle the required power dissipation. At the time of the design, IRGP50B60PD1 was not available, as it is a new product, so IRG4PSC71UD was chosen as the only viable option. The main difference is that IRG4PSC71UD is a much slower device, rated for 8-60kHz as IRGP50B60PD1 rated for 60-150kHz. A faster switching speed will further reduce the size of magnetic and filter components but decrease the current capability of the device. From this analysis, it can be seen that at 20kHz, none of these devices will handle the peak RMS current. Looking at the datasheet for IRG4PSC71UD, 15kHz would allow around 34A current capability, which is very close to the calculated value of 31A. To allow enough margin, the switching frequency was reduced by half to 10kHz which provides about 37A.

The unfortunate drawback to this is the doubling of magnetic and filter components but does it allow discrete devices to be used.

## 3.2 *Magnetics Design*

With the switching frequency dropping by half, the inductor size and value must increase by two. This results in an inductor value of approximately 59 $\mu$ H.

### 3.2.1 **Core Selection**

Two possible core types that can be used for this design are EE and torroid which there are advantages and drawbacks to both. The EE core is simple to wind and mount due to its shape but requires gapping to obtain the correct inductance. Torroids come with built-in gaps and are given in terms of inductance/turn, which makes the design much easier but the mounting is not as simple. Since there are a large selection of Magnetics Inc. torroids on hand in the lab, these were used for the design process.

This is not the preferred design method, finding a core and designing around it, but due to time and cost constraints, it is the most viable option. With the switching frequency dropping by half, the size of the core must double to hold the necessary amount of flux.

The core that was chosen was Magnetics Inc. 55111-A2, which has an  $A_L$  of 60 mH/1000 turns when two cores are stacked together. The magnetic properties of MPP cores are almost identical to those of Kool Mu so the corresponding Kool Mu core 77111-A7 can be used to further reduce the overall converter cost. The efficiency of the MPP material does not have much effect at a relatively slow switching frequency such as 10kHz so there should be no change in the overall converter efficiency.

### 3.2.2 **Inductor Design**

Using the procedure in the Kool Mu handbook [19], the following steps were taken to determine the number of turns.

From the datasheet, the mean magnetic path Length of 55111-A2 is  $l_e = 14.30cm$  and has a nominal inductance of

$$A_L = 33 \pm 8\%$$

Using the value of 8% tolerance, the minimum inductance can be found to be as shown in (3.1).

$$A_{L-\min} = 30.36 * 2 = 60.72 \text{ mH/1000 turns} \quad (3.1)$$

With the worst-case  $A_L$  and the desired inductance of 0.059mH, the number of turns is calculated as in (3.2).

$$N = \sqrt{\frac{0.059 \cdot 10^6}{60.72}} = 31 \text{ turns} \quad (3.2)$$

Using (3.3), the flux density at peak power is found to be  $870_e$

$$H = \frac{0.4\pi \cdot 32A \cdot 31}{14.30cm} = 870_e \quad (3.3)$$

Using the Permeability versus DC Bias Curves, the adjusted value of  $A_L$ , corresponding to  $870_e$  is found to be  $A_L = 54.6$  mH/1000 turns. With the new nominal inductance, the number of turns can be recalculated as in (3.4).

$$N = \sqrt{\frac{0.059mH \cdot 10^6}{54.6}} = 33 \text{ turns} \quad (3.4)$$

With the new number of turns, the peak flux density can be calculated as in (3.5) to see if the core will saturate. Based upon this and the datasheet, the core will not saturate at full power.

$$H = \frac{0.4\pi \cdot 62A \cdot 33}{14.30cm} = 1800_e \quad (3.5)$$

Later in the experimental results section, the inductance value will be tuned to the desired design goal of 59 $\mu$ H.

### **3.3 Current Sensing**

To accurately know the power provided to the bus during discharging and to control the power during charging, some current sensing must be employed. The easiest but most expensive method of doing this involves using off-the-shelf LEMs. For the initial design, LTS 25-NP was used on one phase of each converter. This LEM has a 0.5-4.5V output that corresponds to -70A to 70A and runs off a single 5V supply. Later in Chapter 4 a sensorless current control method will be implemented to replace the LEMs.

### **3.4 Gate Drives**

The HP-316J gate drive chip is used to drive the IGBTs since it can supply a dual-rail output voltage and has built-in DESAT over-current protection. The 316J limits the current by sensing  $V_{CE-ON}$  of the IGBT and if that voltage becomes too high the gate drive turns off. This protects the devices within a switching cycle and can also be used to fault the DSP. This trip level is set using the  $R_{DESAT}$  resistor, the value of  $V_{CE-ON}$  at the desired trip current from the device datasheet, and the following equation (3.6).

$$R_{DESAT} = \frac{(7V - 0.5V - V_{CE-ON})}{250\mu A} = \frac{(7V - 0.5V - 2.0V)}{250\mu A} = 18.0k\Omega \quad (3.6)$$

This corresponds to a fault at approximately 90A peak current.

### **3.5 Capacitor Selection**

#### **3.5.1 Bus Capacitance**

The bus capacitance was chosen based upon the acceptable level of voltage ripple on the high voltage bus. The maximum current ripple occurs at the overloaded case of

10kW and can be calculated to be 25A. Assuming an allowable voltage ripple on each bus of 5% of the nominal 200V and using the equation given in [20], the bus capacitance can be calculated as in (3.7).

$$C_{bus} = \frac{\Delta I}{8 \cdot f \cdot \Delta V} = \frac{25A}{8 \cdot 120Hz \cdot 0.05\% \cdot 200V} = 2.6mF \quad (3.7)$$

To leave acceptable margin, the output capacitance will be approximately doubled to 4.4mF. This can be easily implemented with two 2200μF, 250V electrolytic capacitors in parallel on each bus.

### 3.5.2 Input Capacitance

This value was chosen based upon the acceptable voltage ripple that the batteries can handle. The datasheet puts the maximum voltage ripple at  $\pm 2\%$  of the nominal charging voltage of 58.8V [21], resulting in a  $\pm 1.18V$  ripple. To limit the converter to DCM operation, the peak power during charging will be set at 1.8kW. Using this power level and Matlab simulations at 90% efficiency, the ripple current into the capacitor is approximately 17A. Since there are two phase legs  $180^\circ$  apart, the ripple frequency is 20 kHz. Using the formula for capacitor size based upon the allowable voltage ripple in [16], the low-side capacitor can be calculated as in (3.8).

$$C_{batt} = \frac{\Delta i_L T_s}{8\Delta V} = \frac{17A \cdot 50\mu s}{8 \cdot 1.18V} \geq 90\mu F \quad (3.8)$$

At the time of the power stage design, a large electrolytic capacitor was found on the shelf and used on the low side. This 100V, 4700μF capacitor was then later incorporated into the final board layout and used in the testing of the power stage and the controller.

## Chapter 4 - Control Design and Implementation

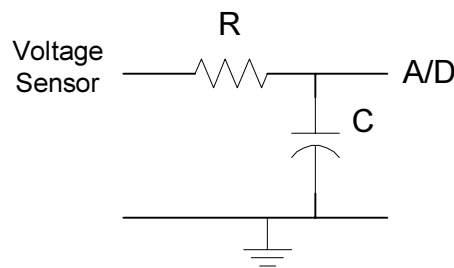
Simple voltage sensing of the input and output voltages will be used to provide closed-loop voltage control. The voltage feedback signal will be sampled by the DSP and the compensator will be implemented digitally. The voltage sensors used were designed and tested in-house for a significant cost savings. The voltage loop can be used in conjunction with an inner current loop for added stability.

The voltage sensors on the low and high side were tested in-circuit and the gain was measured. A range of voltages were measured with each sensor and the gain was averaged to get an accurate value as shown in Table 4.1.

**Table 4.1 Bus and Battery Voltage Sensor Gain**

	Bus Voltage Gain	Battery Voltage Gain
Voltage Gain	12.78mV/V	44.93mV/V
Max Voltage @ 3.3V	258V	73.45V

A simple RC filter was placed on each sensor input to the DSP A/D to filter out any coupled switching noise. This will be included in the loop gain during the compensator design. The RC filter schematic is shown in Figure 4.1 and the component values are shown in Figure 4.2.



**Figure 4.1 RC Filter on A/D Input**

**Table 4.2 Bus and Battery A/D Input Filter Values**

	Battery Sense Filter	Bus Sense Filter
R	1kΩ	1kΩ
C	6.8nF	6.8nF
f <sub>c</sub>	25 kHz	25 kHz

Since this RC filter will increase the impedance seen by the A/D, the prescaler in the DSP was set to an impedance of 1244Ω.

#### **4.1 LEM Current Sensing and Control**

To quickly implement current mode control, a LEM was placed in the path of the inductor current and interfaced to the DSP via an averaging circuit. The LEM chosen was LTS25-NP which can sense ±0-70A of current corresponding to 0.5-4.5V. Since the inductor current is discontinuous, the LEM output signal must be averaged. This is done using a simple analog filter and a voltage follower to interface to the DSP. The analog filter has one pole placed at 1000Hz resulting in the transfer function shown in Equation (4.1).

$$G_{filter} = \frac{3.3/4.5}{\left(s/2\pi 1000 + 1\right)} \quad (4.1)$$

The filter in (4.1) was implemented using two resistors, a capacitor and a single opamp as shown in Figure 4.2.

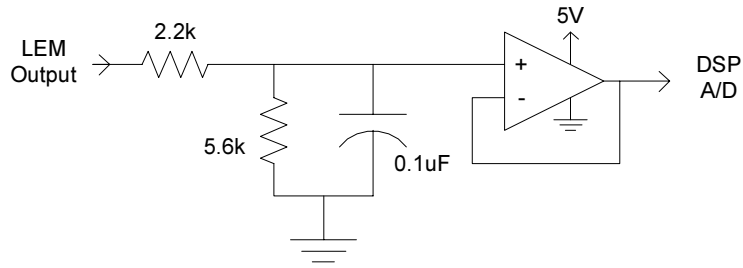


Figure 4.2 LEM Output Averaging Circuit

With this circuit implemented and interfaced to the DSP, the converter was run open loop in boost mode, and it was found that there was significant noise on the output of the LEM and thus on the input to the A/D. This noise was thought to be radiated and coupled onto the signal and power lines and not from the LEM itself. From the scope capture in Figure 4.3 the large amount of noise can be seen on the top two signals, the LEM output and filter output respectively.

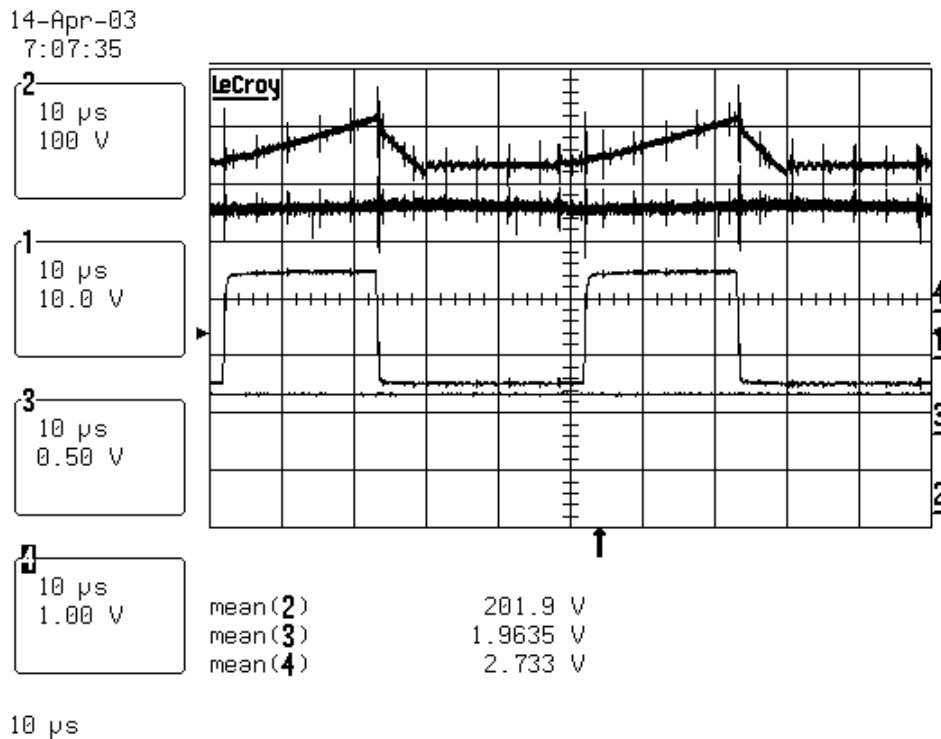


Figure 4.3 LEM and Averaging Circuit Output in Boost Mode

With a simple compensator implemented in the DSP, the loop was closed in current mode and the load was stepped using a contactor on the load bank. With a load step of 200W, the averaged LEM output changed by 6.1mV even though the current changed by 1.5A. This small change is undetectable due to the noise on the signal and the controller does not react at all. After some more testing, it was determined that this is not an acceptable solution. The possible options are to use a LEM with a wider output range and possibly a current output to reduce the effects of noise on the signal. The other option is to implement an observer-based method of current mode control.

## 4.2 Sensorless Average Current Mode Control

Since the system will be controlled using a DSP, this allows for more flexibility in the implementation of the observer. The average current value can be easily computed in the DSP as the area of a triangle  $Area = \frac{1}{2} Base \cdot Height$  since the converter is designed to operate in DCM and the inductors are designed to be linear over the full current range. The height can be easily computed from the inductor value, the switch on-time and the switching period from the waveform shown in Figure 4.4. The peak current value for boost and buck modes are given in Equations (4.2) and (4.3) respectively.

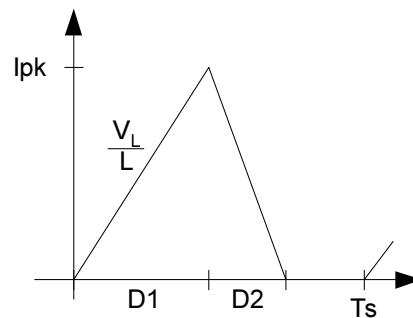


Figure 4.4 DCM Current Waveform

Boost Mode

$$I_{pk} = \frac{V_{Batt}}{L} D_1 T_s \quad (4.2)$$

Buck Mode 
$$I_{pk} = \frac{(V_{Bus} - V_{Batt})}{L} D_1 T_s \quad (4.3)$$

The equations (4.2) and (4.3) neglect the  $V_{CE}$  drop of the switch during conduction and that will be address later in the design. The base of the triangle is simply the sum of the on and off times. The on-time is the duty-cycle and in DCM the off-time is the diode conduction time which must be measured as it depends on the load. This procedure for measuring  $D_2$  will be explained later in this section.

$$Base = D_1 + D_2 \quad (4.4)$$

Using (4.4) and either equation (4.2) or (4.3), the average current through the inductor can be calculated as in (4.5).

$$I_{avg} = \frac{T_s}{2L} D_1 (D_1 + D_2) V_L \quad (4.5)$$

#### 4.2.1 Diode On-time Sensor

To know the base of the triangle, the current fall time must be measured. This can be simply done using an optocoupler that is high when the diode is conducting and low when it is not. This will provide a pulse to the DSP the same width as the time the diode is on than can be measured with the internal capture unit. Since the falling current time at light load is very short, a few microseconds, a very fast optocoupler must be used to provide an accurate measurement. NEC PS-2701 optocoupler was used at first and was found to switch in about  $5\mu s$  which is much too slow. After some testing it was determined that analog devices cannot be used due to their slow speed so digital device was chosen. Agilent Technologies makes a wide range of digital optocouplers and HCPL-060L, a 10MBd device with a switching time of 75ns, was chosen [22]. This

package has two inverting digital logic optocouplers that operate off of 3.3V. LVTTTL voltage levels are preferred to simplify interfacing to the DSP.

### **Boost Mode Bias Resistor Calculation**

The optocoupler diode will be placed anti-parallel to the device diode. This will allow a low output when the diode is off and high when the diode is conducting. When the device diode is off, the voltage across it will be dependant on the difference between the bus and the battery voltage. At the nominal case this voltage can be calculated as in (4.6).

$$\begin{aligned} V_{diode} &= V_{Bus} - V_{Batt} \\ 200V - 48V &= 152V \end{aligned} \quad (4.6)$$

The datasheet recommends 5-15mA of forward current for a high current signal and 0-250 $\mu$ A for a low signal [22]. The forward current was set to 8mA and with the known voltage drop; the necessary series resistance can be calculated as shown in (4.7). To provide more immunity to common-mode noise, the resistance will be split in two halves and placed on both sides of the diode.

$$R = \frac{152V}{8mA} = 19k\Omega \quad (4.7)$$

Dividing this resistance by two and rounding to the next standard value, R becomes 10k $\Omega$ .

### **Buck Mode Bias Resistor Calculation**

For buck mode, the voltage across the diode will be the battery voltage which can range from 45V to 58.8V. With  $V_{Batt}$  is set to 50V and  $I_f = 8mA$  of forward current the resistance can be calculated as in (4.8).

$$R = \frac{50V}{8mA} = 6.25k\Omega \quad (4.8)$$

To use standard values, this value was increased to 6.6k $\Omega$  and divided into 3.3k $\Omega$  halves. The circuit was prototyped with the above values and tested in the following section. See Appendix F for the final schematic.

#### **4.2.2 Optocoupler Design Verification**

The optocoupler circuit was tested on the converter in both buck and boost modes to verify it's correct operation. In boost mode, as the diode turns off and the switch voltage begins to ring it trips the optocoupler, causing glitching on the output. This goes away as the output voltage reaches approximately 177V and is gone at 200V as seen in Figure 4.5. Since the bus voltage will be regulated to 200V, this will not be a problem. In Figure 4.5, the waveforms from top to bottom are as follows,  $V_{CE}$ ,  $I_{L1}$ , optocoupler output and D1.

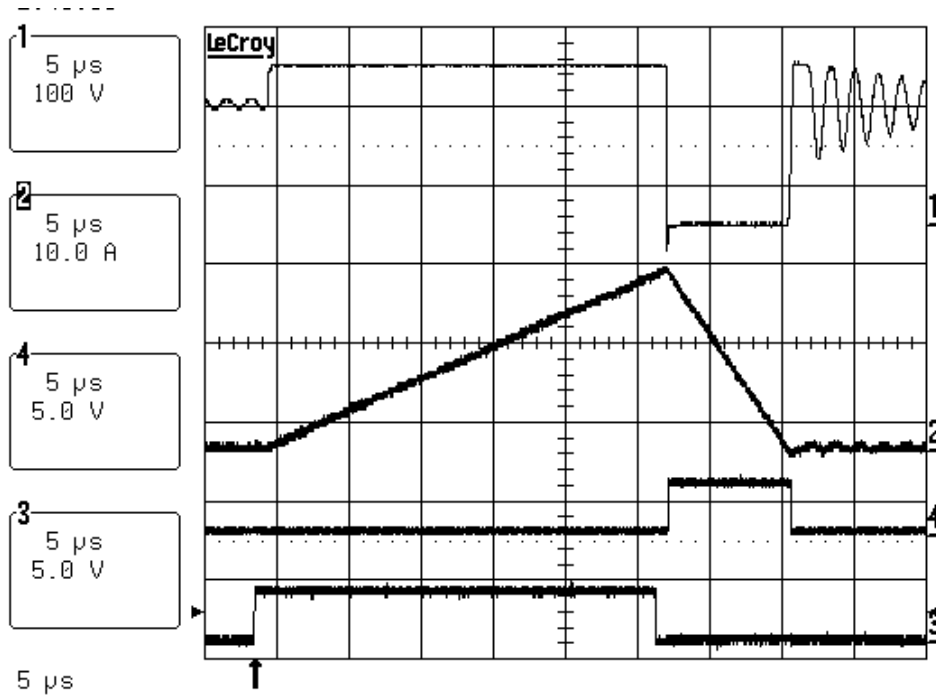


Figure 4.5 Boost Mode Optocoupler Test  $V_o = 200V$

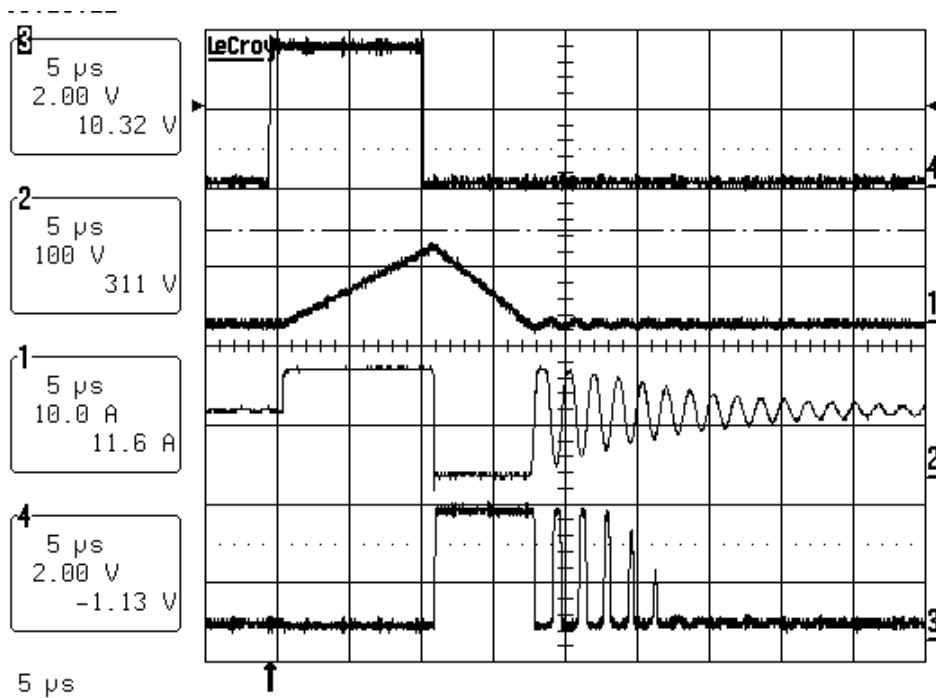


Figure 4.6 Boost Mode Optocoupler Glitching  $V_o$  Under 177V

Figure 4.6 shows the optocoupler ringing when the output voltage is at 136V. The ringing is more severe at light load due to less damping.

In buck mode when the boost diode turns off and the switch voltage begins to ring, the optocoupler also catches this ringing and glitches appear on the output. As the battery voltage is increased, these glitches begin to go away, but they never disappear at 58.8V. To help solve this, the bias resistors were recalculated for the worst case, 58.8V and 5mA of forward current as shown in (4.9).

$$R = \frac{58.8V}{5mA} = 11.7k\Omega \quad (4.9)$$

To use standard values, 11.2k $\Omega$  was chosen and divided evenly into 5.6k $\Omega$ . These new values were tested and resulting in a reduction of glitching at lower voltages.

Unfortunately, there are still glitches on the output at 60V as seen in Figure 4.7 and these will have to be removed in the DSP. In Figure 4.7, the waveforms from top to bottom are: gate drive pulse,  $V_{diode}$ ,  $I_{L1}$  and optocoupler output.

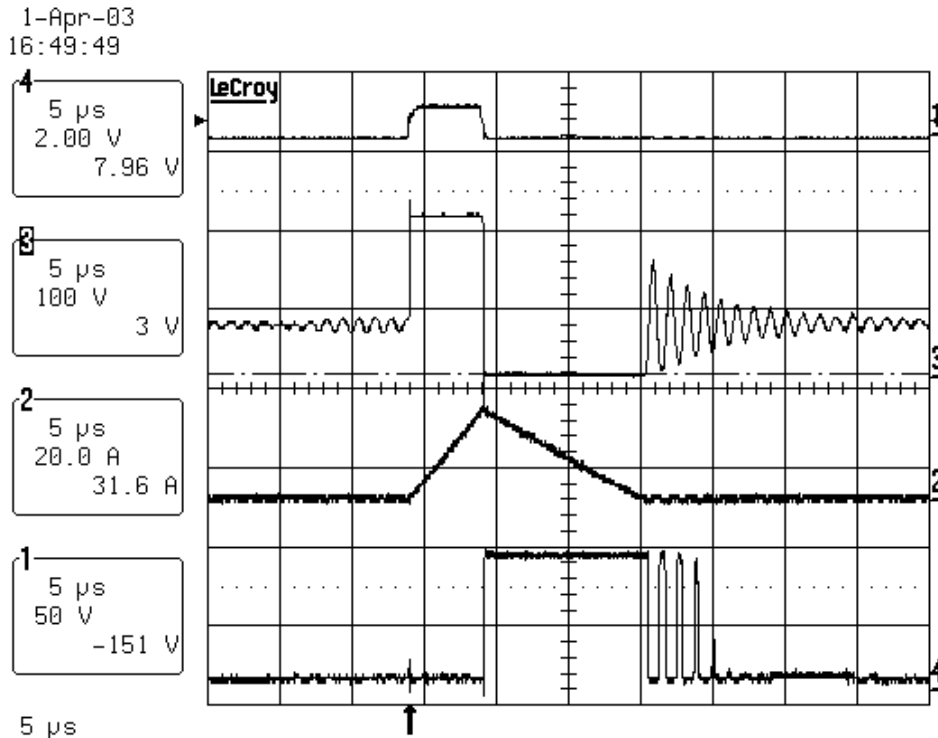


Figure 4.7 Buck Mode Optocoupler Glitching at  $V_o = 58.8V$

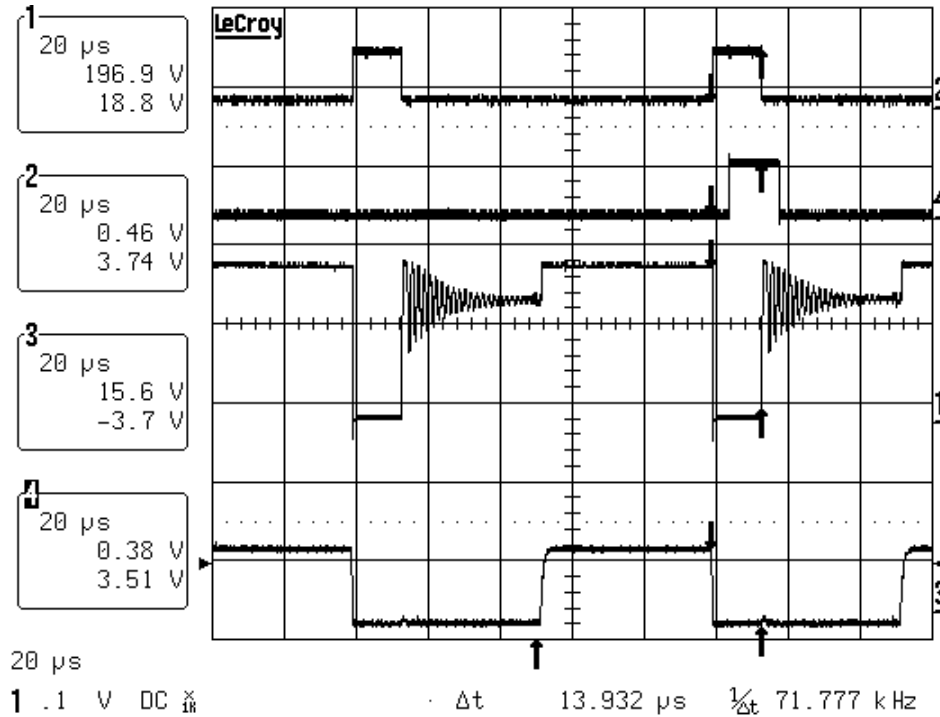
### 4.2.3 Capture Unit Setup

For this particular application, only two optocouplers are needed, one for each device, top and bottom. Since only one optocoupler signal will be relevant in either buck or boost, a simple digital mux is used to multiplex the optocoupler output signal to the capture input.

Capture unit 1 is set to work off of general purpose timer 2 and is zeroed at initialization. An interrupt service routine was created and the interrupt was serviced on both rising and falling edge of the optocoupler output as shown in Appendix D.5. This allowed the counter to be zeroed on the rising edge and then saved on the falling edge. This counter is synchronized with the DSP core frequency of 40MHz for a resolution of 25ns.

The capture routine was implemented and tested using a function generator with a comparable pulse width. The capture unit has a two-level FIFO that is used to store the timer values upon each capture event. If this FIFO becomes full without being read, the

capture unit will not log any more events. In the first iteration of the code, the capture interrupt only occurred every other switching period as shown in Figure 4.8 on channel 4.



**Figure 4.8 Capture Interrupt on Alternating Switching Cycles**

To fix this, the FIFO must be “tricked” into thinking that it has one more spot left and thus it will allow future capture events. This is done by writing a “01” into the CAP1FIFOA register that controls the FIFO operations [23]. This was implemented in the code and the capture operation was again tested and the correct operation is shown in Figure 4.9. To verify the correct measurement of the diode conduction time, the width of the channel 3 pulse can be compared to that of channel 4, which is the optocoupler measurement of the diode conduction time. To create the pulse on channel 3, a bit was set to go high at the first call of the capture routine, a rising edge, and go low when a falling edge is detected.

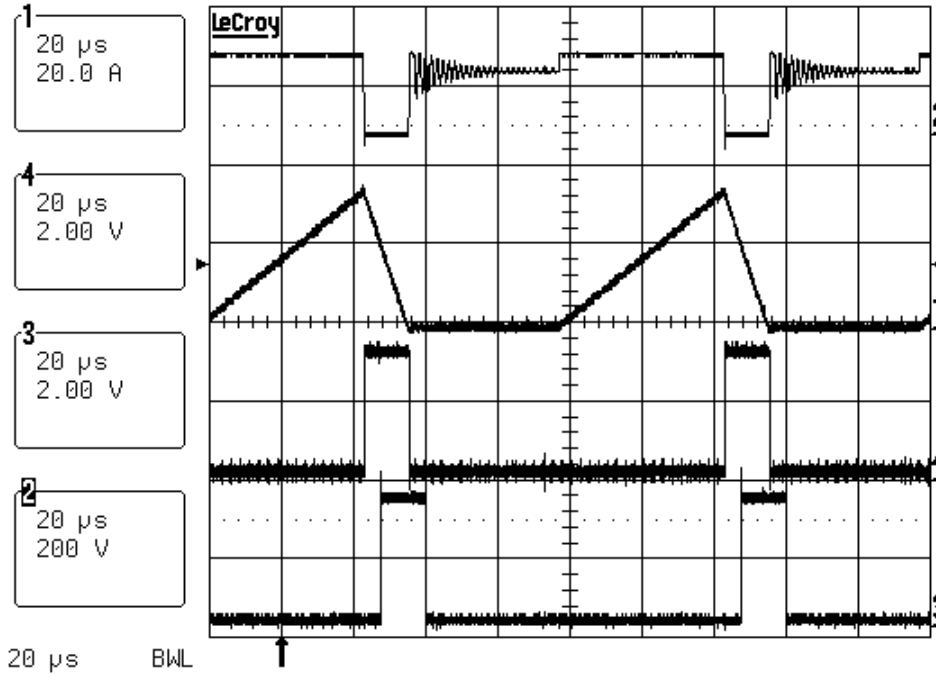


Figure 4.9 Capture Interrupt During Every Clock Cycle

Another way to verify the converted time is to inspect the counter value by placing a breakpoint in the code. Through these two methods it was determined that the capture routine converted the correct time.

The slight delay between channel 4 and channel 3 is the time it takes for the DSP to service the capture interrupt and does not affect the value of the converted time.

#### 4.2.4 Proper Scaling of Converted Times

When the off time is measured using the value of the counter, the number is in terms of core clock cycles at 25ns. Since the switching frequency is 10 kHz, one full cycle takes 100μs which corresponds to signed integer 0x7FFF or 32767. To convert from clock cycles to signed integer notation, the clock cycles must be multiplied by 8.192 which can be derived as shown in (4.10).

$$\frac{25ns}{100\mu s} \cdot 32767 = 8.192 \quad (4.10)$$

Since the DSP is fixed-point device, this multiplication must be done in two parts. First the converted value is shifted to the left by three bits and then added to the result of the off-time multiplied by 0.192. The C code is shown in Figure 4.10.

```
cap.D2 = D2<< 3;
Save = (D2 * 0x1893)>> 15;
cap.D2 = cap.D2 + Save;
```

**Figure 4.10 DSP Code for Conversion from Clock Cycles to Signed Integer**

Cap.D2 is now scaled properly to be used in the average current calculation.

#### **4.2.5 Constant Calculation**

In the average current equation of (4.5)  $D_1$ ,  $D_2$  and  $V_{Batt}$  are known through measurement in the DSP. Since the switching period and the inductor value are assumed to be constant, the term  $\frac{T_s}{2L}$  can be computed as shown in (4.11). The measurement of the inductance will be shown in Chapter 5.

$$G = \frac{100\mu s}{2 \cdot 57.3\mu H} = 0.8726 = 0x6FB1 \quad (4.11)$$

#### **4.2.6 Inductor Voltage Calculation**

The voltage across the inductor in boost mode is simply the battery voltage if the resistive drop of the inductor and the device on voltage are assumed to be zero. To find the voltage across the inductor in buck mode, the difference between the bus voltage and the battery voltage must be calculated. Since the battery and bus voltage are on different scales due to the different gains of the voltage sensing boards, the battery voltage must be converted to the bus voltage scale as shown in (4.12).

$$\frac{G_{Bus}}{G_{Batt}} = \frac{12.78mV/V}{44.93mV/V} = 0.2844 = 0x2467 \quad (4.12)$$

This multiplication and subtraction can now be implemented in code as shown in Figure 4.11.

$$V_s = V_{Batt} * 0x2467;$$

$$V_L = V_{Bus} - V_s;$$

**Figure 4.11 Inductor Voltage Calculation in Buck Mode**

#### 4.2.7 Switch Voltage Drop Correction

In the ideal case, the inductor voltage would be either the battery voltage or the bus minus the battery voltage. But this is not the case due to the  $V_{CE-ON}$  of the IGBT. The voltage drop is a function of current and can be obtained from the device datasheet [18]. Since there is a one cycle delay, the assumption is that the current does not change by much between two cycles and thus the voltage drop will be calculated using the previous value of the current.

There are two different sets of values depending on boost or buck mode. The scale of each current and voltage drop is different depending on what voltage sensor the measurement depends on, as shown previously in Table 4.1. From the datasheet [18], a set of  $V_{CE-On}$  and  $I_C$  values were chosen and converted to the appropriate scale for boost or buck using the equations shown in (4.13) and (4.14). Equation (4.13) relates the actual current to the scaled current as represented in the DSP.  $V_{MAX}$  is the maximum sensed voltage of the sensor being used as shown in Table 4.1.

$$I_{C,hex} = \frac{I_C}{V_{MAX}} \cdot 0x7FC0 \quad (4.13)$$

Equation (4.15) relates the actual on-voltage to the scaled voltage that will be subtracted from the measured inductor voltage.

$$V_{CE,hex} = \frac{V_{CE-On} \cdot V_{GAIN}}{3.3V} \cdot 0x7FC0 \quad (4.14)$$

To create Table 4.3, the battery sense voltage gain is used in boost mode and the bus voltage gain is used for the buck mode calculations.

**Table 4.3 Boost and Buck  $V_{CE-On}$  Corrections vs. Collector Current**

$I_C$	$V_{CE-On}$	$I_C$ - Boost	$V_{CE-On}$ - Boost	$I_C$ - Buck	$V_{CE-On}$ - Buck
0A	0.7V	0	0x0138	0	0x0059
2A	0.75V	0x037E	0x014E	0x00FE	0x005F
6A	1.0V	0x0A79	0x01BE	0x02FA	0x007F
10A	1.07V	0x1174	0x01DD	0x04F6	0x0088
20A	1.25V	0x22E8	0x022D	0x09EC	0x009E
35A	1.46V	0x3D18	0x028B	0x115D	0x00B9

To find the proper value of  $I_C$  and the corresponding value of  $V_{CE-On}$ , a simple look-up table was created in code as shown in Appendix D.1. To find the correct value of current, a while loop increments the value of  $I_C$  until it is larger than the current value of  $I_{avg}$ . When this happens, the proper voltage drop is subtracted from the inductor voltage as shown in the C code in Figure 4.12.

```

while (Iavg >= Ic[count2])          /* Run through Ic until Iavg is larger */
count2++;

VL = VL - Vce[count2];             /* Subtract off VCE from VL */

```

**Figure 4.12 C Code to Subtract  $V_{CE-On}$  from Inductor Voltage**

Since  $I_{avg}$  will always be larger than zero when the converter is running, it was excluded from the  $I_C$  look-up table.

#### 4.2.8 Calculation of the Actual Current from DSP $I_{avg}$

With the design complete of the sensorless current measurement, the final step is to convert the result of  $I_{avg}$  to the actual inductor current. This is simply done for boost and buck as shown respectively in (4.15) and (4.16) by dividing by the maximum DSP result and multiplying by the appropriate maximum sensor output.

$$I_{AVG} = \frac{I_{avg-DSP}}{32736} \cdot 73.5 \quad (4.15)$$

$$I_{AVG} = \frac{I_{avg-DSP}}{32736} \cdot 258 \quad (4.16)$$

### 4.3 *Controller Design and Implementation*

With all of the pertinent plant information converted to the digital domain, the current and voltage compensators can be designed. The boost mode controller will be a dual-loop system with an inner current loop and an outer voltage loop. The buck controller will have two independent loops, a current loop and a voltage loop. The buck converter will run in current mode until the voltage reaches the proper charging voltage and then switch to voltage mode for the remainder of charging. Since the DSP is sampling at 10 kHz, the bandwidth of the system will be limited to 5 kHz and there will be an inherent one-cycle delay that will be modeled accordingly.

#### 4.3.1 Derivation of Converter Transfer Functions

To develop effective compensators for both the voltage and current loops, it is necessary to have accurate small-signal models of the plant. Of the many possible transfer functions, control-to-output voltage  $\frac{v_o}{d}$ , and control-to-inductor current  $\frac{i_L}{d}$  are the most important in designing the closed-loop control. There are many methods in

which to model DCM PWM converters as done in [24] and [25]. For simplicity, the modeling of these transfer functions will use the procedure in [17].

### Boost Small-Signal Analysis

Using the DCM small-signal model from [17] the control-to-output transfer function of the boost converter can be written as shown in (4.17).

$$G_{vod}(s) = \frac{-2V_o(M-1)(1+sR_cC_o)(M^2R_l+sM^2L-R)}{D(a_2s^2+a_1s+a_0)}$$

$$a_2 = M^2LC_o(MR+2MR_c-R_c)$$

$$a_1 = (-C_o+C_oM)R^2 + (-R_cC_o+2MR_cC_o+M^3R_lC_o)R - M^2L - M^2R_lR_cC_o + 2M^3L$$

$$a_0 = (2M-1)(M^2R_l+R)$$

$$M = \frac{V_o}{V_g}$$

(4.17)

This transfer function was checked using PSpice and found to agree with the average model as shown in Appendix A.3.

The derivation of  $\frac{i_L}{d}$ , shown in Equation (4.18) is performed using the same method from [17] and is also verified using the average model as shown in Appendix A.4.

$$Gild(s) = \frac{2V_o M(M-1)(sRC_o + s2R_c C_o + 2)}{D(a_2 s^2 + a_1 s + a_0)}$$

$$a_2 = -M^2 LR_c C_o + 2M^3 LR_c C_o + M^3 LC_o R$$

$$a_1 = -R^2 C_o + R^2 C_o M - M^2 L + 2M^3 L + 2MR_c C_o R - M^2 R_l R_c C_o + 2M^3 R_l R_c C_o - R_c C_o R + M^3 R_l C_o R$$

$$a_0 = -R + 2MR - M^2 R_l + 2M^3 R_l$$

$$M = \frac{V_o}{V_g}$$

(4.18)

### Buck Small-Signal Analysis

The control-to-output transfer function for buck mode, shown in equation (4.19), was derived as in [17] and again check using PSpice. The average model can be found in Appendix A.1.

$$Gvod(s) = \frac{-4V_o RLf_s M (1 + R_c C_o s)}{D(a_2 s^2 + a_1 s + a_0)}$$

$$a_2 = LC_o(R_c + R)(-2Lf_s M - 2D^2 R + 2Lf_s M^2 + D^2 RM)$$

$$a_1 = (-2L^2 M + 2R_x C_o RLM^2 + 2RC_o R_l LM^2 + 2R_c C_o R_l LM^2 - 4R_c C_o RLM + 2L^2 M^2 - 2R_c C_o R_l LM - 2RC_o R_l LM - 2R^2 C_o LM)f_s + R_c C_o R^2 D^2 M - 2D^2 RL + D^2 RLM + R^2 C_o R_l D^2 M + R_c C_o R_l D^2 RM - 2R_c C_o R_l D^2 R - 2R_c C_o R^2 D^2 - 2R^2 C_o R_l D^2$$

$$a_0 = (-4RLM + 2R_l LM^2 - 2R_l LM + 2RLM^2)f_s - 2R_l D^2 R + R^2 D^2 M - 2R^2 D^2 + R_l D^2 RM$$

$$M = \frac{V_o}{V_g}$$

(4.19)

As with the boost converter, the control-to-inductor transfer function was derived and verified using the same method and is shown in (4.20). The PSpice average model is shown in Appendix A.2.

$$G_{ild}(s) = \frac{2V_o(1-M)}{D} \frac{1}{\left( R(1-M) + \frac{R\left(R_c + \frac{1}{sC_o}\right)}{R + R_c + \frac{1}{sC_o}} + R_l + sL \right)} \quad (4.20)$$

$$M = \frac{V_o}{V_g}$$

With these transfer functions derived in terms of converter parameters and duty cycle, the small-signal characteristics for any load or voltage level can be plotted using the Bode() command in Matlab.

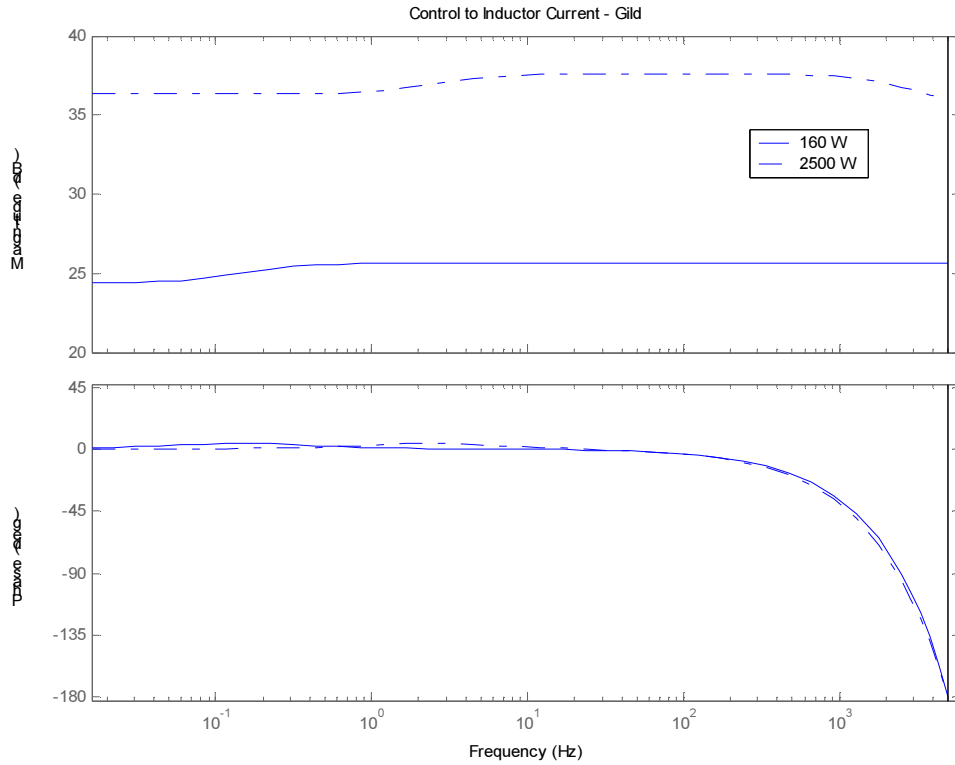
The transfer functions were derived in the analog domain and to easily model the one cycle delay, all of the transfer functions must be converted into the digital domain using the Matlab command c2d(). The continuous to digital command takes the transfer function to be converted, the sampling period and the conversion type, which for this application is zero-order hold.

$$G_{vodz} = c2d(G_{vod}, Ts, 'zoh');$$

### **Boost Mode Transfer Functions**

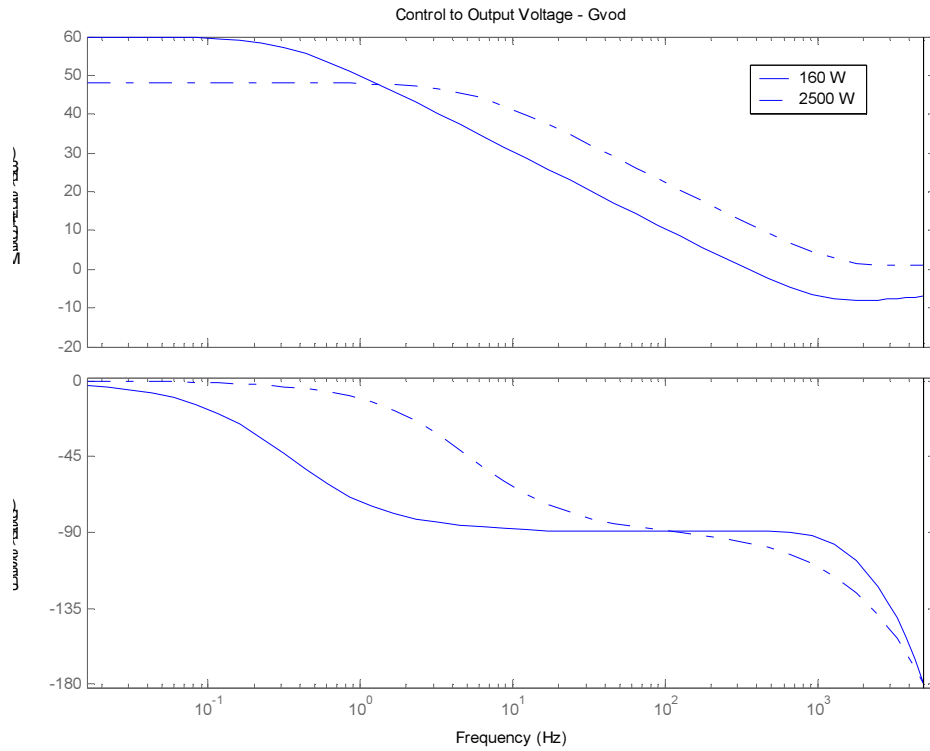
In boost mode, the converter is desired to operate over a load range of 160W to 2.5kW and in buck mode the converter should work down to 50W and up to 1.8kW. To understand the converter operation over this wide range of power levels, the control-to-output voltage and control-to-inductor current will be modeled over the full range.

The boost mode control-to-inductor current is shown in Figure 4.13 at 160W and 2.5kW. For the simulation, the input voltage was set at 50V and the output voltage was set at 200V and can be seen in Appendix B.2.



**Figure 4.13 Boost Mode Control-to-Inductor Current at 160W and 2.5kW**

In Figure 4.14 the boost mode control-to-output voltage transfer function was modeled at the minimum and maximum power levels and the code can be seen in Appendix B.1.

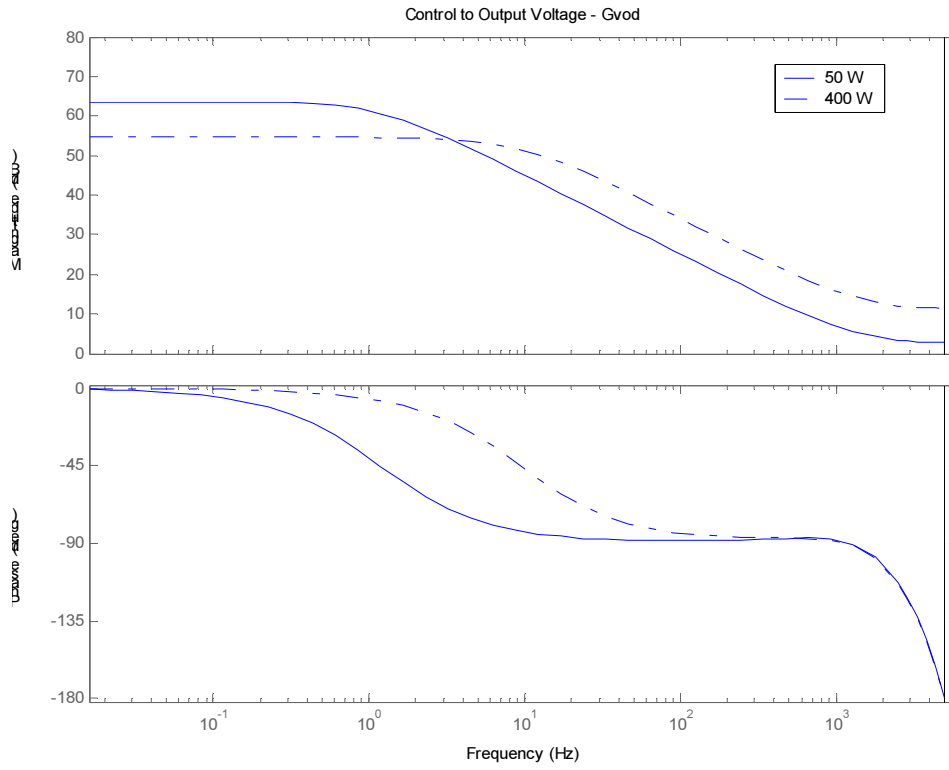


**Figure 4.14 Boost Mode Control-to-Output Voltage at 160 W and 2.5kW**

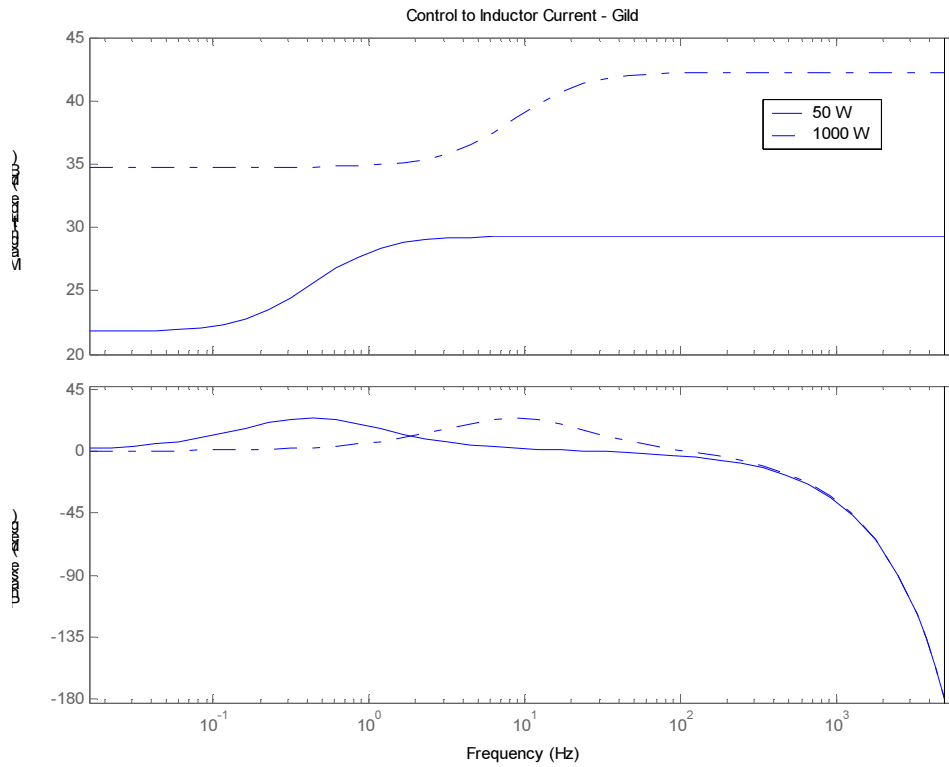
### **Buck Mode Transfer Functions**

The same procedure was followed for buck mode, plotting the converter transfer functions over the full power range. For buck mode control-to-output voltage the voltage was set at the peak battery voltage of 58.8V and for control-to-inductor current, the output voltage was set at 55V.

Figure 4.15 shows the control-to-output voltage and Figure 4.16 shows the control-to-inductor current transfer functions. The code can be seen in Appendixes C.2 and C.1 respectively.



**Figure 4.15 Buck Mode Control-to-Output Voltage at 50 W and 400 W**



**Figure 4.16 Buck Mode Control-to-Inductor Current at 50 W and 1000 W**

### 4.3.2 Digital Conversion of RC Filters and Modeling of One-Cycle Delay

The RC filters are also converted to the digital domain using the `c2d()` command and then the one cycle delay is modeled by simply multiplying by  $z^{-1}$ . Now all the parts of the system have been converted to the digital domain and this information can be loaded into RLTOOL to complete the controller design.

## 4.4 Implementation of a Digital Compensator

There are two main forms of IIR filters, Direct I and Direct II form. The Direct I form requires more memory storage locations, but results in fewer computations. The Direct II form requires only one data buffer but requires more computations. For this application, Direct Form II will be used which is shown in Figure 4.17 with the corresponding difference equation, in (4.21) and transfer function (4.22) [26].

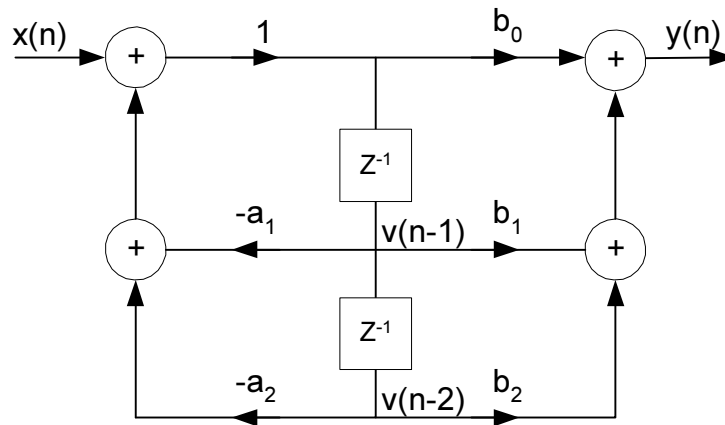
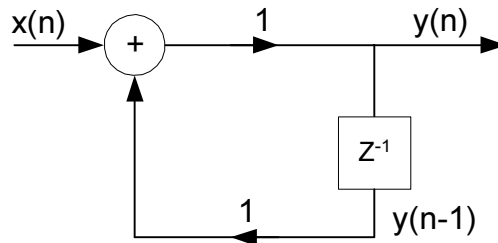


Figure 4.17 Direct II Form

$$\begin{aligned} v(n) &= -a_1 v(n-1) - a_2 v(n-2) + x(n) \\ y(n) &= b_0 v(n) + b_1 v(n-1) + b_2 v(n-2) \end{aligned} \quad (4.21)$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}} \quad (4.22)$$

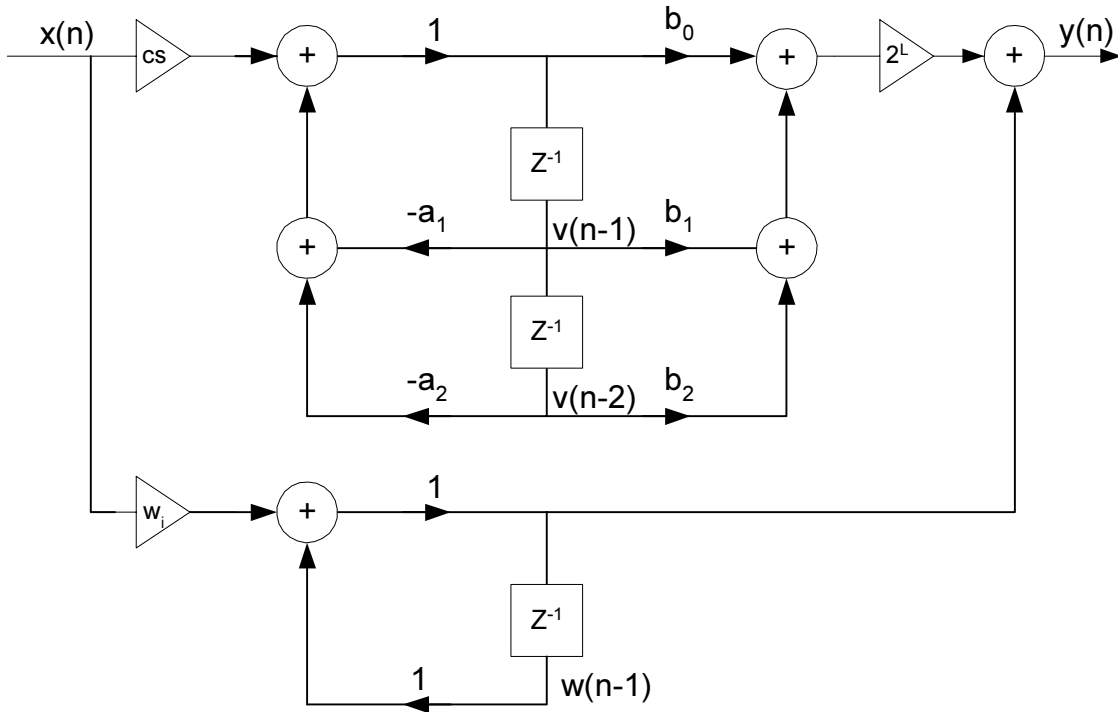
This form only represents the poles and zeros of the compensator. The integrator is implemented independently to allow for an anti-windup scheme. The structure of the integrator is shown in Figure 4.18 and the equation is shown in (4.23).



**Figure 4.18 Block Diagram of Digital Integrator**

$$y(n) = x(n) + y(n-1) \quad (4.23)$$

One of the main concerns during scaling is to not change the value of the feedback coefficients,  $a_1$  and  $a_2$ . These directly affect the response of the filter and cannot be scaled but the output coefficients,  $b_0$ - $b_2$  can be scaled as needed. In an analog system, the gain of the integrator can be placed on either side, but due to the limitations of the digital system, the integrator gain must be placed before the storage element. Combining these different parts yields the full scaled system as it is implemented in the DSP shown in Figure 4.19. The input scaling factor  $cs$  is a value less than one and the output scaling factor  $2^L$  is normally greater than one as is the integrator gain  $w_i$ .



**Figure 4.19 Block Diagram of the Digital Compensator Implementation**

With the full digital system designed, it is now necessary to compute the various coefficients and scaling terms. As mentioned earlier, the DSP has a mode where it limits additions to +1 or -1. This will be used to prevent the integrator and the sum of the integrator with the poles and zeros from overflowing.

#### 4.4.1 Proper Scaling of Compensator Coefficients

Before any compensator can be implemented, it is necessary to scale the coefficients to limit the possibility of overflow in the fixed-point system. There are three main types of scaling, I1 norm, I2 norm and Chebychev norm. Of these types only I1 norm explicitly prohibits overflow and will be used for this controller [27]. To properly use this procedure, the integrator cannot be part of the scaled system and must be removed from the compensator before attempting any scaling.

As shown in the Matlab code for all the compensators, the digital compensator is converted to the analog domain and `zpkdata()` changes the zero/pole/gain transfer function to polynomial form. To split the integrator from the other poles and zeros, the

Matlab command Residue() is used to perform partial fraction expansion on the transfer function. The integrator and its associated gain are recombined and then the remaining poles and zeros are formed into a new compensator. This will be again converted into the digital domain and then back into zero/pole/gain form so that the zp2sos() function can be used. Zp2sos() takes a transfer function in Z,P,K form and converts it into digital second-order section as shown above in Figure 4.19.

The next step is to determine the input scaling term  $cs$ . This term depends on the impulse response of the system to the first storage location  $v(n-1)$ , which is found using the digital impulse command `impz()`. To find the largest possible value, the absolute values of the output of the `impz()` function are summed together. This number is converted to a power of two using the `log2()` command which results in the value of  $cs$ . It is desirable to have all gains in the system represented in powers of two so that the calculations can be performed by bit-shifting.

To determine the output scaling term  $L$ , the largest value of  $b_x$  is found and multiplied by the inverse of  $cs$ . The resulting number is scaled by  $2^L$  so that it is between 0.5 and 1. Now that all the scaling terms have been found, the final coefficient values can be calculated by applying the scaling factors. The integrator is computed the same way using the `zp2sos()` command but the gain is not scaled. The value of  $w_i$  is directly entered into the DSP. This procedure can be seen in Appendices B.1, B.2, C.1 and C.2.

## ***4.5 Design of Digital Compensators***

### **4.5.1 Design of Boost Mode Compensators**

In a two-loop boost system, the current compensator is designed for good loop dynamics and then the voltage compensator is designed for the desired crossover

frequency and phase margin using the closed-loop transfer function  $T_2 = \frac{T_v}{1 + T_i}$ . The

controller is desired to work over a wide range of power levels, from 160W to 2.5kW.

Both the current and voltage loops will be designed using the following operating point:

$$V_{\text{batt}} = 48\text{V}$$

$$V_o = 200V$$

$$P_o = 1.5kW$$

The dual-loop controller for boost mode is shown in Figure 4.20.

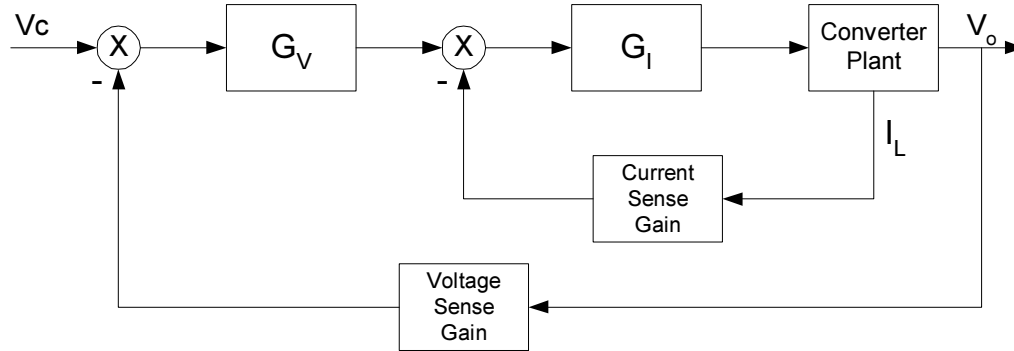


Figure 4.20 Boost Mode Dual-Loop Controller Block Diagram

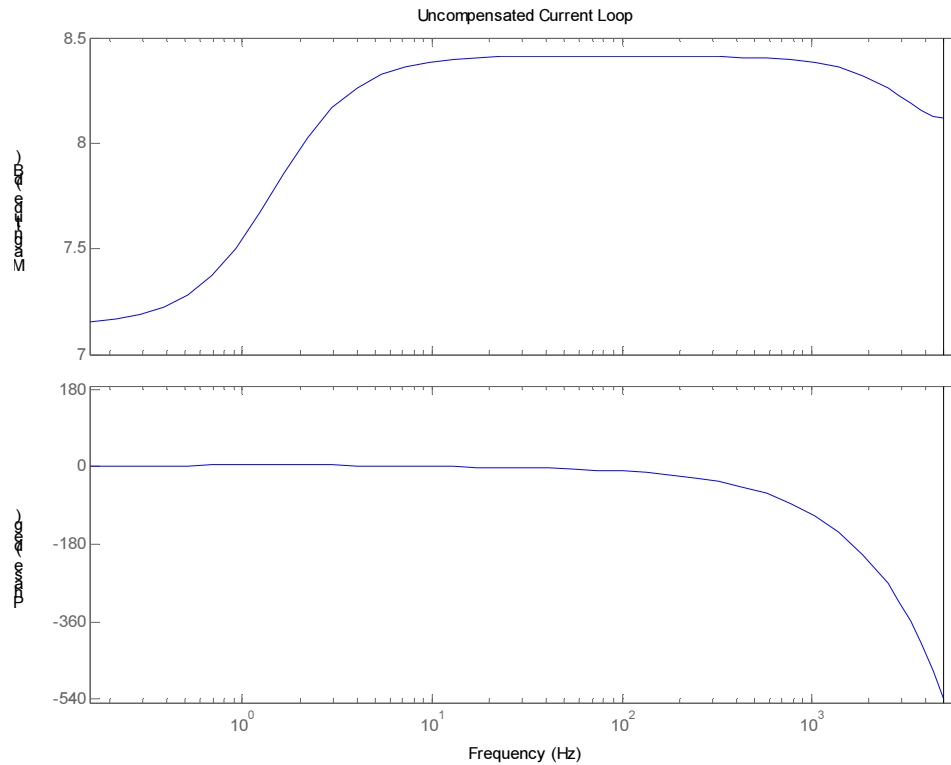
### Design of the Boost Mode Current Compensator

To aid with the compensator design the Matlab tool, RLTOOL will be used. RLTOOL allows compensators to be designed graphically by placing poles and zeros and then adjusting the gain for the desired crossover and phase margin. The program continually computes the open loop response and shows the gain and phase margin and determines stability of the system.

Since only one inductor current is being used for the control, the current control model was changed to only include one phase of the converter. The power was halved and the inductance used was that of a single phase. If this is not done, around 1.5kW a large peak in the gain appears due to the ringing frequency of the inductor and the output capacitance. The frequency of the ringing can be calculated for a single-phase system as shown in (4.24).

$$f_{LC} = \frac{1}{\sqrt{2\pi LC}} = \frac{1}{\sqrt{2\pi(57.3\mu H)(4.4mF)}} = 795Hz \quad (4.24)$$

The digital transfer function,  $G_{d(z)}$ , was loaded into the plant block and the sensor gain was loaded with the battery sense gain including the RC filter. Shown in Figure 4.21 is the uncompensated response of the current loop



**Figure 4.21 Boost Uncompensated Open-loop Current Response**

The current loop is designed to have a large bandwidth and high DC gain. Since the sampling frequency is 10 kHz and there is the inherent one-cycle delay, the phase drops off sharply as it approaches the Nyquist frequency of 5 kHz. This limits the amount of gain that is achievable due to the low phase margin at high frequencies.

Through trial and error it was found that due to scaling concerns only one pair of poles and zeros plus the integrator is able to be simply implemented in the DSP. For this application this provides an acceptable amount of control over the system response.

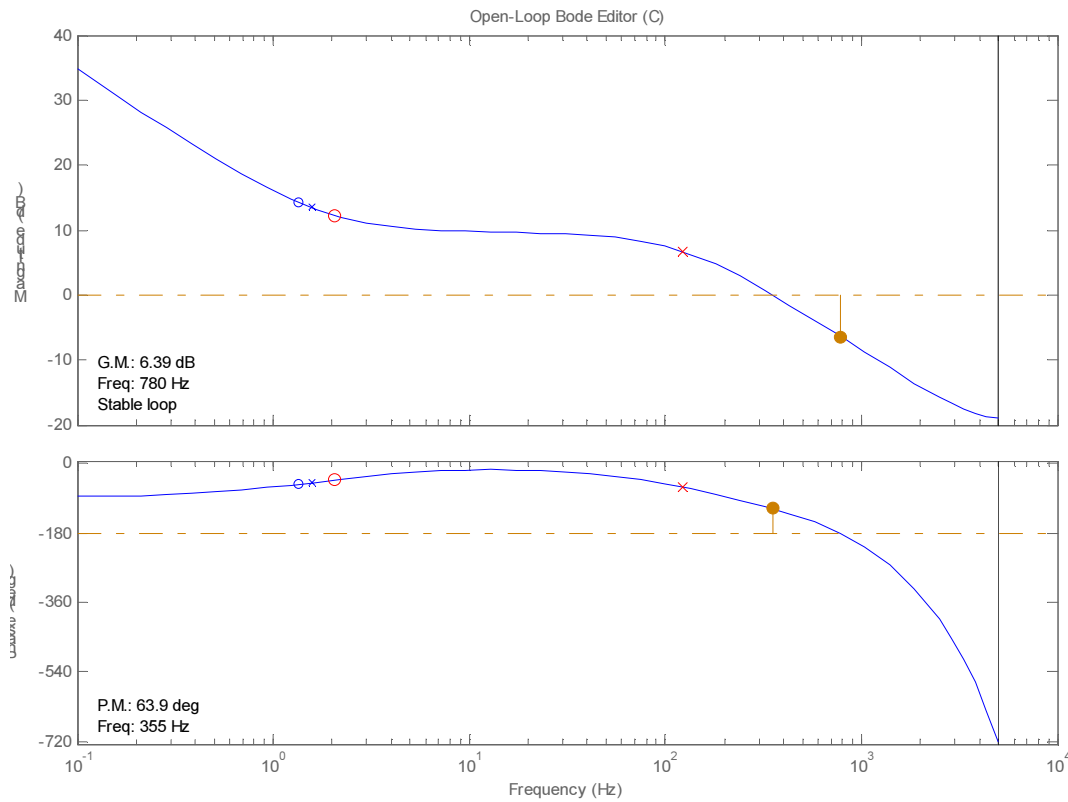
The placement of the poles and zeros not only depends on the desired system response but the achievable gain of the DSP. It was found that due to the resolution of the sampling frequency if the zero was moved to lower frequencies, it would result in

canceling the integrator. Also, the closer the pole and zero are together, the larger the necessary scaling.

After much iteration of poles, zeros and integrator gains, the compensator was designed as shown in (4.25).

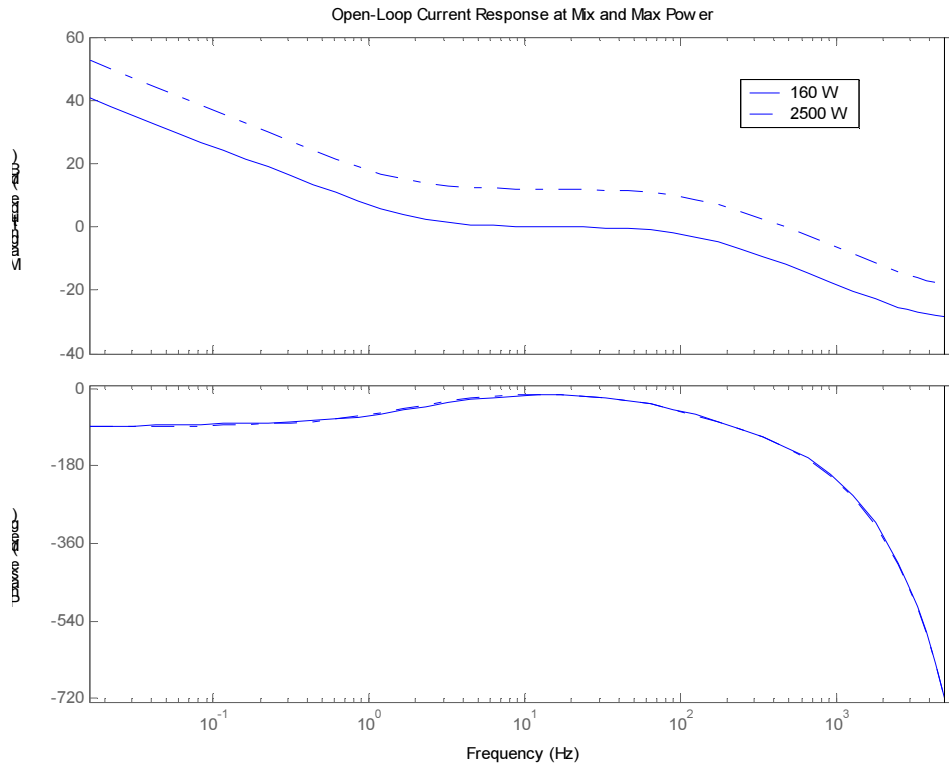
$$C_{i_z} = 0.08614 \frac{(z - 0.9987)}{(z - 1)(z - 0.9254)} \quad (4.25)$$

The compensator has a bandwidth of  $f_c = 355$  Hz and a phase margin of  $\phi = 63.9^\circ$  at 1500W as shown in Figure 4.22.



**Figure 4.22 Boost Open Loop Current Response**

To determine if the loop will be stable over the full power range, the loop response was calculated for 160 W and 2.5kW shown in Figure 4.23 and the loop performance is summarized in Table 4.4.



**Figure 4.23 Boost Open Loop Current Response at Min and Max Power**

Table 4.4 shows that at high power there is a possibility for the loop to become unstable due to the reduced gain and phase margin. On the opposite end of the power spectrum, the loop becomes very stable at low power levels but has a very slow response.

**Table 4.4 Boost Current Compensator Response Summary**

Power Level	Crossover Frequency	Gain Margin	Phase Margin
160 W	12.8 Hz	15.8 dB	164.7°
1500 W	355 Hz	6.39 dB	63.9°
2500 W	478 Hz	3.9 dB	43°

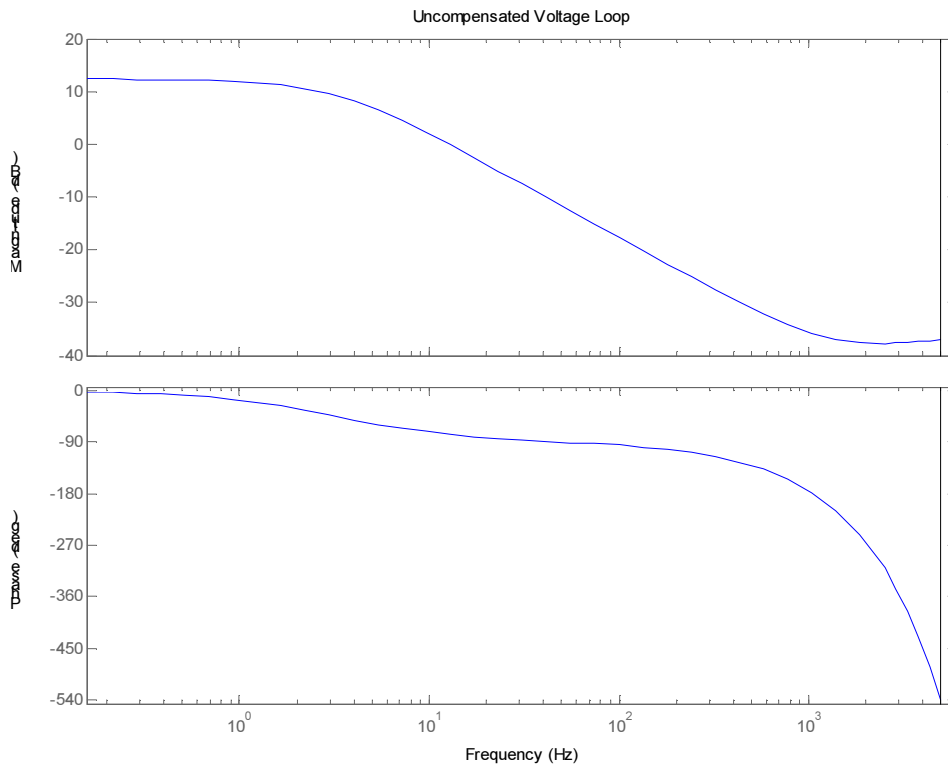
The compensator is put into the Matlab scaling script and the scaled coefficients and gain terms are shown in Table 4.5.

**Table 4.5 Boost Current Compensator Scaled Coefficients and Scaling Factors**

Coefficient	b <sub>0</sub>	b <sub>1</sub>	b <sub>2</sub>	1	a <sub>1</sub>	a <sub>2</sub>	w <sub>i</sub>	Cs	L
Decimal Value	0	0.6771	0	1	-0.9254	0	0.0015	2 <sup>-4</sup>	1
Hex Value	0	0x56AB	0	0x7FFF	-0x7673	0	0x0031		

### Design of the Boost Mode Voltage Compensator

With the current compensator designed, the voltage compensator can be designed by looking at the response of  $T_2 = \frac{T_v}{1 + T_i}$ . The same procedure was used to import the control-to-output transfer function and the associated sensor gain and filter dynamics. The open loop uncompensated response is shown in Figure 4.24. This is the response of the equivalent converter by summing the response of both phases as they both contribute to the output voltage.



**Figure 4.24 Boost Uncompensated Open-Loop Voltage Response**

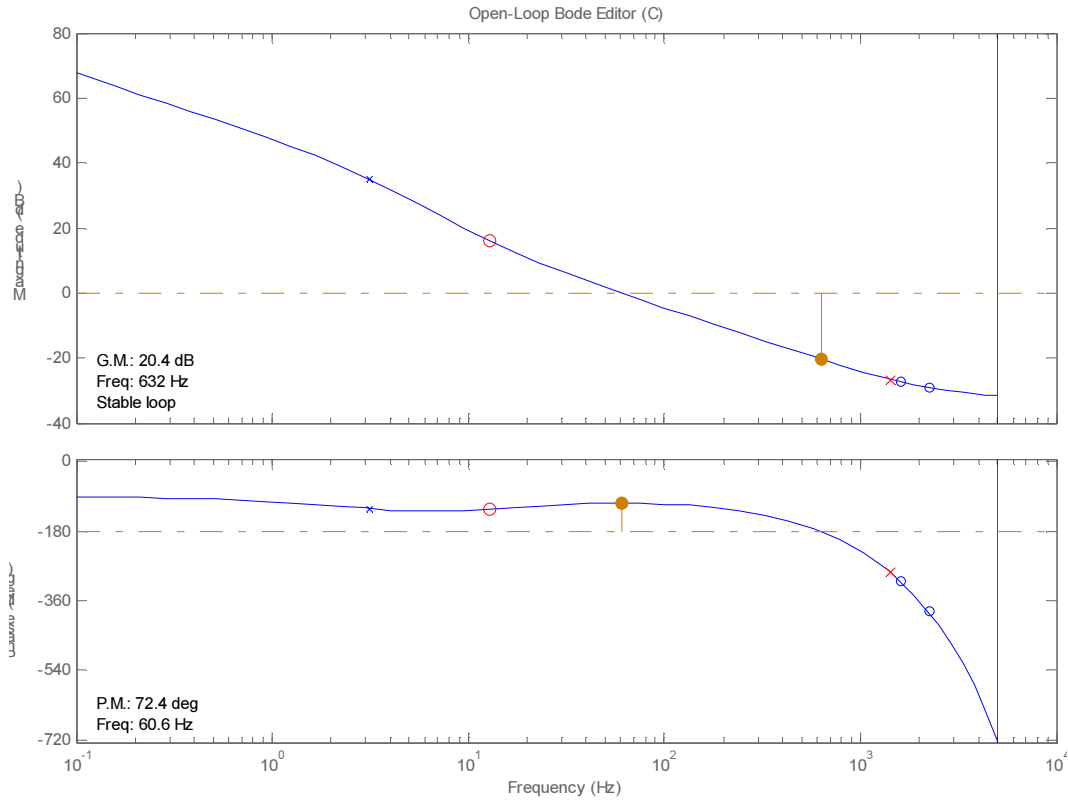
Again the maximum number of poles/zero pairs used was one plus the integrator. The open loop response is meaningless by itself and only the full closed-loop gain should be used to verify the design. Another Matlab script, see Appendix B.3, was written that contained the previously designed current compensator and the voltage compensator being investigated to plot  $T_2$ . As the voltage compensator is changed, it is exported to the Matlab workspace and then the full loop gain is plotted to check the response. This procedure was repeated until an acceptable compensator was designed.

For the voltage compensator it is desirable to have large DC gain and the gain to fall off as quickly as possible. Through many design iterations, it was found that the zero could not be pulled too far in as it would create a large bump in the gain. Again, the pole and zero must be separated so the resulting compensator could be scaled within the limits of the DSP. A zero could also be placed at the LC ringing frequency to further reduce the effect, but this was not simply done with one pole and one zero as the gain factors were unrealizable.

The compensator was designed based upon the ability to implement it in the DSP and the full-loop crossover and phase margin. The resulting compensator in the digital domain is shown in (4.26).

$$Cv_z = 2.7232 \frac{(z - 0.992)}{(z - 1)(z - 0.4053)} \quad (4.26)$$

Using the compensator from (4.26), the open-loop voltage response at 1.5kW can be plotted as shown in Figure 4.25.



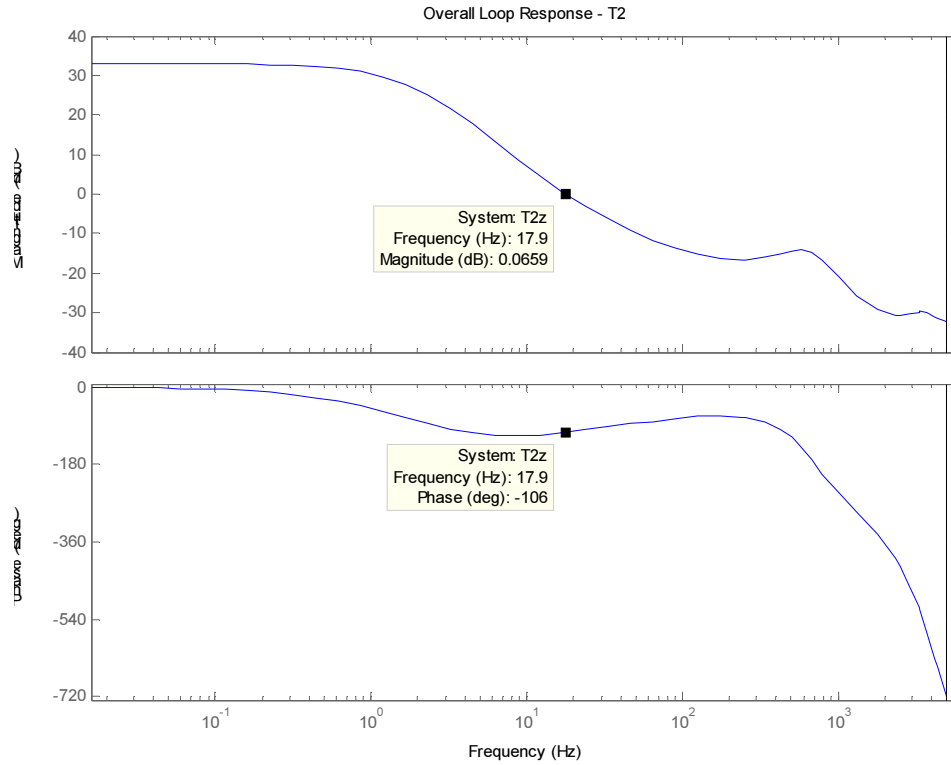
**Figure 4.25 Boost Outer Loop Response**

Since this loop response in Figure 4.25 does not contain any information about the system response, it is not necessary to look at the minimum and maximum power levels. The voltage compensator was scaled using the procedure outlined previously and the resulting coefficients and scaling terms are shown in Table 4.6.

**Table 4.6 Boost Voltage Compensator Scaled Coefficients and Scaling Factors**

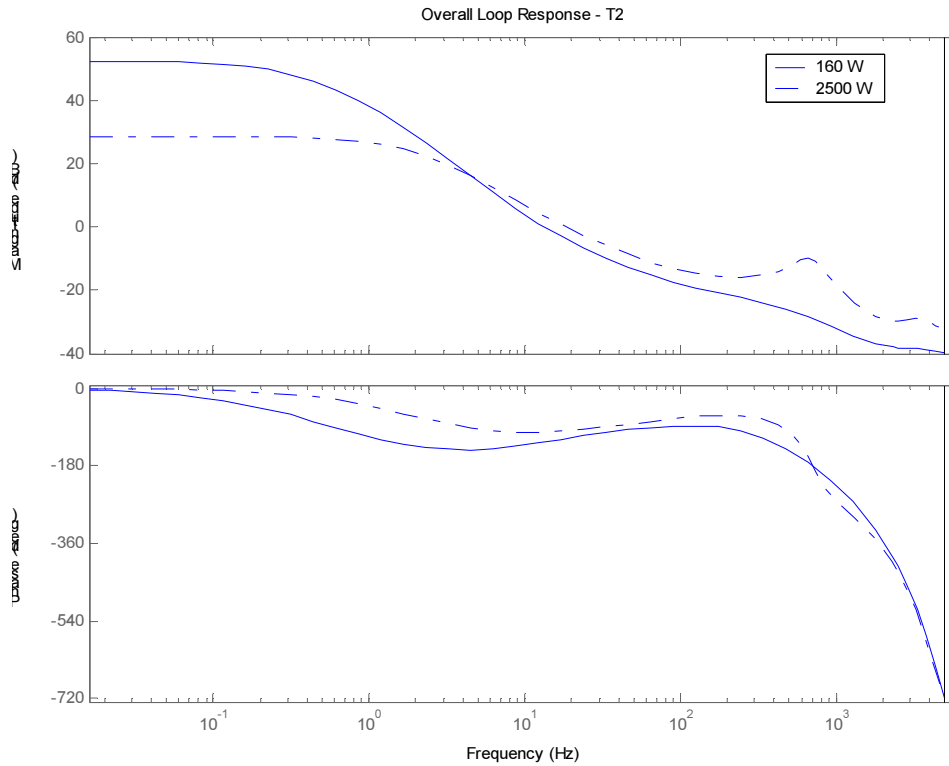
Coefficient	$b_0$	$b_1$	$b_2$	1	$a_1$	$a_2$	$w_i$	Cs	L
Decimal Value	0	0.6716	0	1	-0.4053	0	0.0366	$2^{-1}$	3
Hex Value	0	0x55F8	0	0x7FFF	-0x33E0	0	0x04B0		

With both compensators designed, the full loop gain can be plotted in Matlab and from Figure 4.26, it can be seen the crossover occurs at 17.9 Hz and results in 74° of phase margin.



**Figure 4.26 Boost Overall Loop Gain at 1500 W**

In Figure 4.26 at 586 Hz there is still a slight bump in the gain corresponding to the ringing frequency of the inductor and the output capacitance. This bump does not occur as calculated in (4.24) but at 235 Hz lower around approximately 560Hz. This may be due to the affect of placing two phases in parallel.



**Figure 4.27 Boost Overall Loop Response at Min and Max Power**

Figure 4.27 shows the loop response over the full power range. The effect of the LC ringing is almost gone at low power but still exists at full power. The full-loop gain and phase responses are summarized in Table 4.7.

**Table 4.7 Boost Overall Loop Response Summary**

Power Level	Crossover Frequency	Gain Margin	Phase Margin
160 W	13.5 Hz	28.9 dB	54.5°
1500 W	17.9 Hz	15.3 dB	74°
2500 W	18.4 Hz	10.1 dB	81.2°

From Table 4.7 it can be seen that the controller is stable over all loads. There is the possibility for slight instability at 160W due to the reduced value of the phase margin. This can be corrected by increasing the gain slightly but will reduce the gain margin at all of the loads, especially at 2500W due to the peaking corresponding to the LC ringing. The system response is not greatly affected by changes in the battery voltage.

#### **4.5.2 Design of Buck Mode Compensators**

In buck mode, the converter will operate in two modes during charging, depending on the available power and battery voltage. If the battery voltage is below 58.8V the controller will be in current mode, charging with a constant current. When the battery voltage reaches 58.8V, the controller will switch to voltage mode and maintain a constant charging voltage.

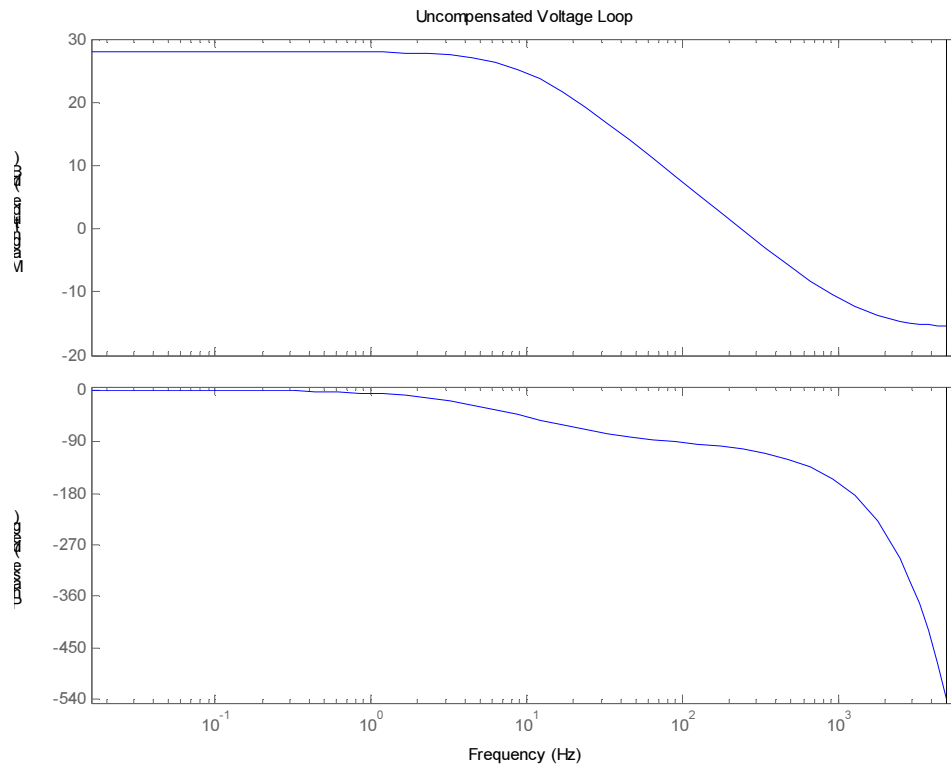
This will require two independent controllers to be designed and implemented in the DSP.

#### **Design of the Buck Mode Voltage Compensator**

As previously mentioned, the buck mode power varies over a wide range so designing a controller to work over that full range may not be the best design choice. Therefore, it would be more appropriate to design the voltage loop for low power since when the batteries are nearly charged, the power will be low. Making this assumption, the voltage loop will be designed at 400 W but should operate well below that.

The batteries can be modeled as a very large capacitor with a series resistance and thus the RC time constant can be calculated to be on the order of seconds. This means that the controller will not have to be very fast as the dynamics of the battery pack are relatively slow. With this in mind, the loop gain was reduced significantly to allow for stability at power levels well below 400W.

The plant model at 400W and the bus sensor gain were imported into RLTOOL and the open-loop uncompensated response was created as shown in Figure 4.28.

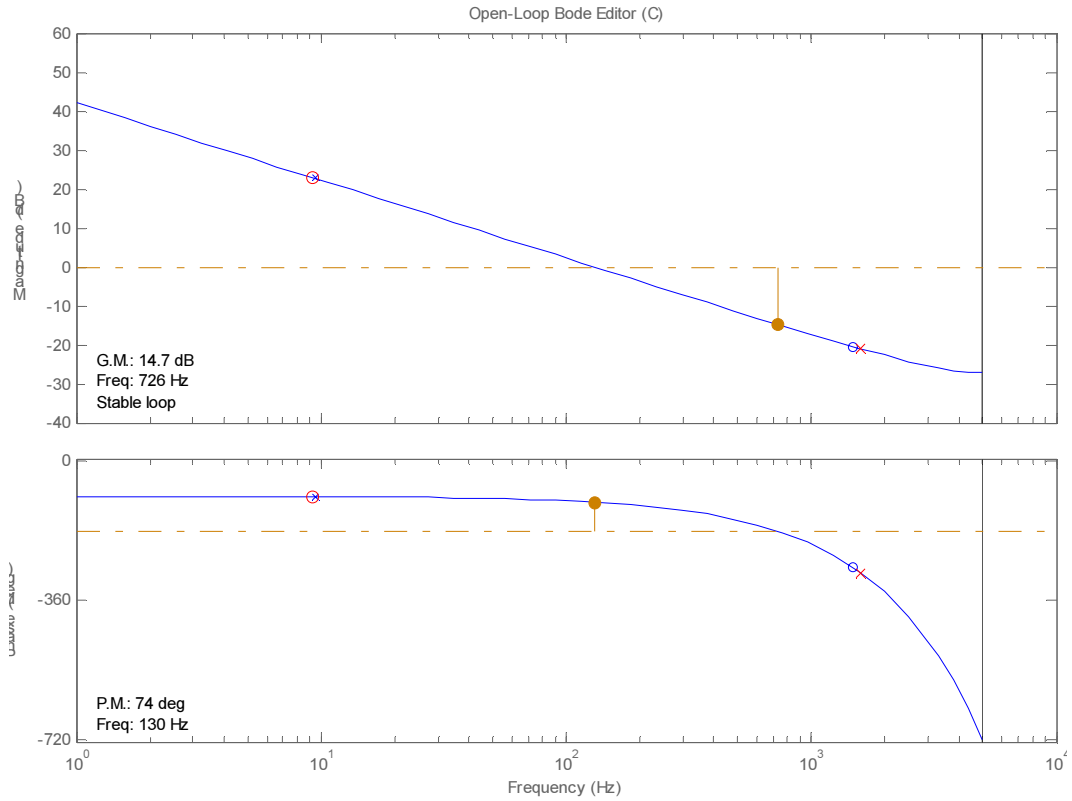


**Figure 4.28 Buck Uncompensated Voltage Response**

The pole was placed at the ESR zero of the output capacitor and the zero was placed to cancel the plant pole, increasing the gain margin significantly. The final compensator is shown in (4.27).

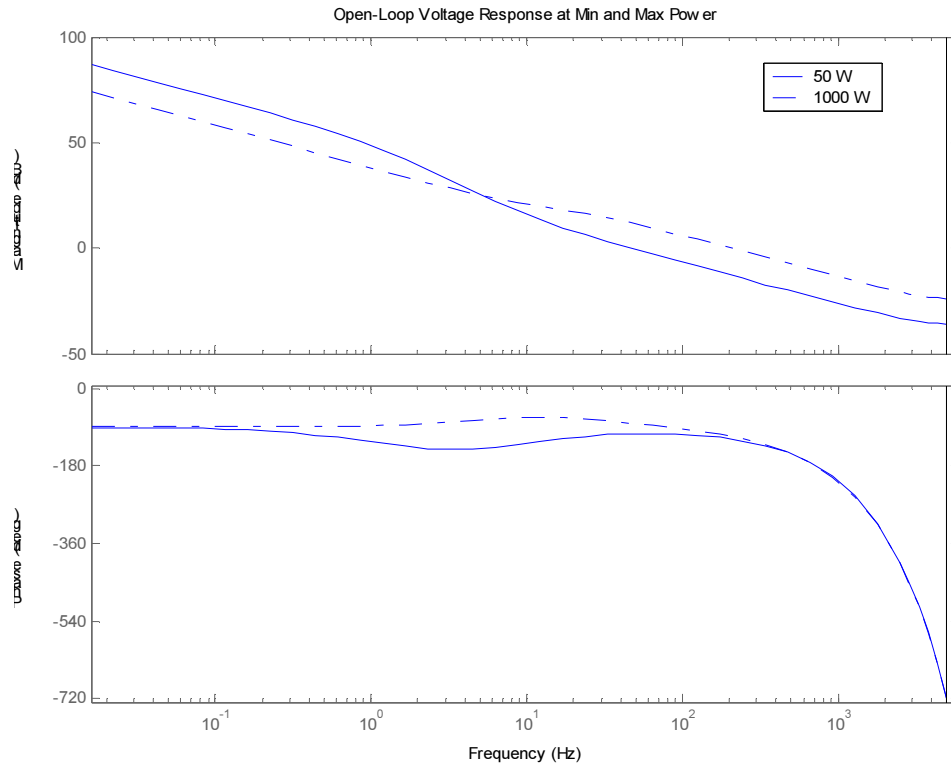
$$Cv_z = 0.3603 \frac{(z - 0.9942)}{(z - 1)(z - 0.3685)} \quad (4.27)$$

The open-loop gain is shown in Figure 4.29.



**Figure 4.29 Buck Open-Loop Voltage Response**

From Figure 4.29 the crossover frequency is 130 Hz with a phase margin of 74° and gain margin of 14.7 dB. As with the boost controller, the response at low and high power is shown in Figure 4.30.



**Figure 4.30 Buck Open-Loop Voltage Response at Low and High Power**

The controller performance at different power levels is summarized in Table 4.8.

**Table 4.8 Buck Voltage Compensator Response Summary**

Power Level	Crossover Frequency	Gain Margin	Phase Margin
50 W	47 Hz	23.5 dB	75°
400W	130 Hz	14.7 dB	74°
1000 W	205 Hz	10.7 dB	68°

The voltage compensator was scaled using the procedure outlined previously and the resulting coefficients and scaling terms are shown in Table 4.9.

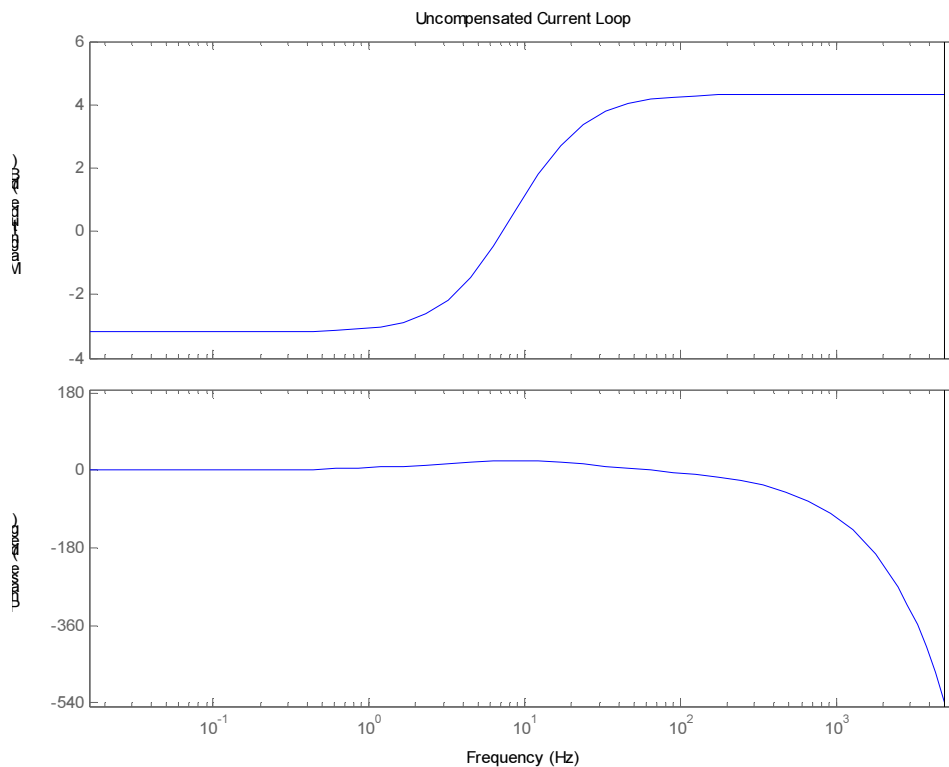
**Table 4.9 Buck Voltage Compensator Scaled Coefficients and Scaling Factors**

Coefficient	$b_0$	$b_1$	$b_2$	1	$a_1$	$a_2$	$w_i$	Cs	L
Decimal Value	0	0.7139	0	1	-0.3685	0	0.0033	$2^{-1}$	0
Hex Value	0	0x5B62	0	0x7FFF	-0x2F2B	0	0x00 6C		

### Design of the Buck Mode Current Compensator

The current mode controller should also be stable over a wide range of charging powers, but will be used more at higher power. Therefore, the compensator will be designed for a charging power of 1000W. The charging voltage is not defined in this mode, but for the creation of the plant model, 55V will be used as the output voltage.

As in previous designs, the plant and sensor were entered into RLTOOL. The uncompensated response is shown in Figure 4.31.



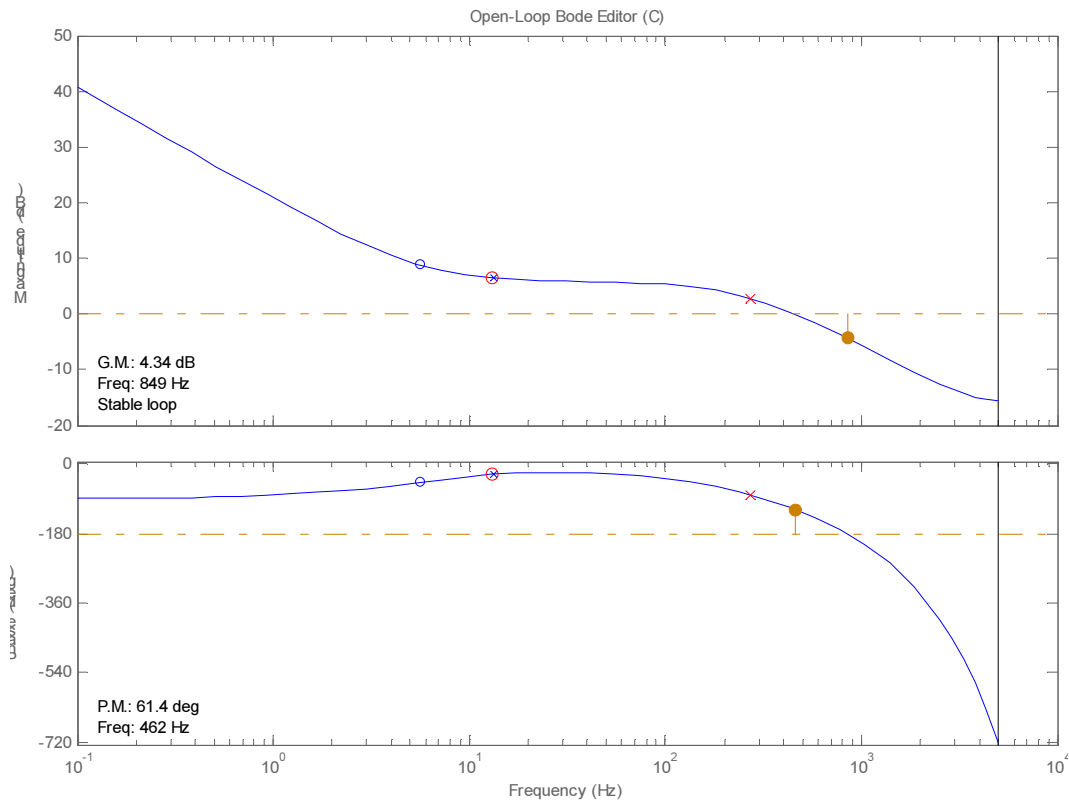
**Figure 4.31 Buck Uncompensated Open-Loop Current Response**

Using RLTOOL, an integrator and a pair of poles/zeros were placed in the loop. The zero was placed to cancel the plant pole at 13.1 Hz and the pole was located to roll off the

gain at higher frequencies. The integrator gain was adjusted to have sufficient phase margin at 1000W but the resulting gain margin is small due to the placement of the pole. It is hard to locate the pole very close to the zero as the scaling becomes quite large, limiting the precision of the controller. The resulting compensator is shown in (4.28).

$$C_{i_z} = 0.1885 \frac{(z - 0.9918)}{(z - 1)(z - 0.8426)} \quad (4.28)$$

The compensated current loop response is shown in Figure 4.32.



**Figure 4.32 Buck Compensated Current Loop**

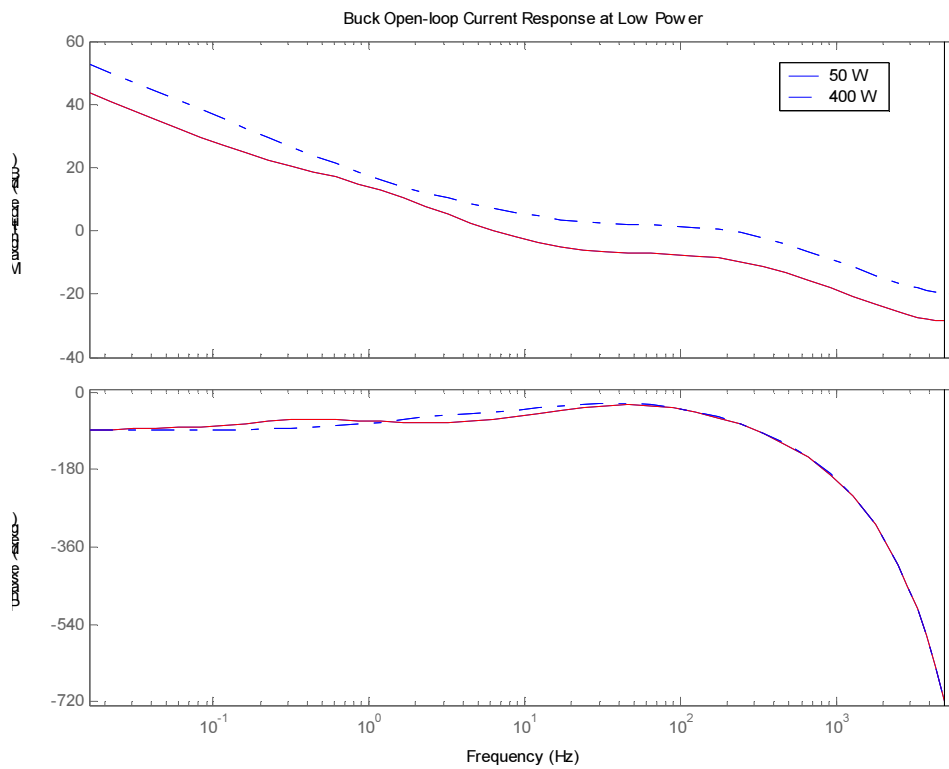
From Figure 4.32 the crossover frequency is  $f_c = 462\text{Hz}$  with a phase and gain margin of  $\phi_m = 61.4^\circ$  and 4.34 dB respectively. The compensator was scaled and converted to a second-order section and the resulting coefficients are shown in

Table 4.10.

**Table 4.10 Buck Current Compensator Scaled Coefficients and Scaling Factors**

Coefficient	$b_0$	$b_1$	$b_2$	1	$a_1$	$a_2$	$w_i$	$C_s$	L
Decimal Value	0	0.7147	0	1	-0.8426	0	0.0098	$2^{-3}$	1
Hex Value	0	0x5B7A	0	0x7FFF	-0x6BD9	0	0x0142		

To verify the controller over the wide range of operating power, the plant was simulated at different power levels and the resulting gain, phase and crossover frequencies are shown in Table 4.11. The plot of the open-loop response at 400 W and 50 W is shown in Figure 4.33.



**Figure 4.33 Buck Open-Loop Current Response at Low and High Power**

**Table 4.11 Buck Current Compensator Response Summary**

Power Level	Crossover Frequency	Gain Margin	Phase Margin
50 W	6.5 Hz	17.2 dB	117°
400W	208 Hz	8.2 dB	114°
1000 W	467 Hz	4.2 dB	60°

At both 1000W and 400W, the gain margin is low which could cause some instability at those power levels. Due to this, the controller design must be verified through testing as done in Chapter 5.

## Chapter 5 - Experimental Results

### 5.1 *Previous Iterations of the Design*

The first iteration of the design was to use a single 48V battery pack and a dual-phase non-isolated bidirectional boost to create a 400V output. The inductors were designed with seven turns of 16mil copper foil on EE70 cores for an inductance of 20 $\mu$ H with an air gap of 1.36mm. To get the converter up and running as soon as possible, Infineon SKW25N120 IGBTs with co-diodes were used. Off-the-shelf VPT dual gate drive boards were interfaced to the DSP and an open-loop control was implemented in code. The converter was tested and worked as designed, but due availability and manufacturability, EE were replaced with torroids.

The second iteration of the converter was changing from EE cores to torroids which simplified the design, eliminating the air gap. They were also sized for a dual battery, two-phase system significantly reducing the size. Choosing among cores that were available in the lab, Magnetics Inc. 77109-A7 was thought to be a good match for the power level. Using adjusted design specifications, the inductors were designed with 21 turns for 34 $\mu$ H. To carry the necessary RMS current and reduce AC losses, 15 strands of #24AWG magnet wire was twisted together and used. The devices were also changed to better match the power requirements. Fairchild Semiconductor HGTG30N60C3 were installed which are rated for 30A at 100°C. After testing the converter open-loop over a range of inductor currents, it was seen that the cores were not large enough and were saturating at high power levels.

The third iteration of the design used Magnetics Inc. 77192-A7 torroids with 17 turns to achieve an inductance of 34 $\mu$ H. With the new inductors installed the output power was increased to 1.96kW which blew a device. At the time it was thought that these devices could handle the power, but after looking at the power dissipation spec on the datasheet, it is obvious that they will not.

The fourth iteration uses the switches that were chosen for the final design, IRG4PSC71UD. These were installed in the circuit with the same inductors and again the peak power condition was tested. The switches failed about 15 seconds into the 60 second 2.5kW test. The theory at the time was that the inductors were saturating and thus

the switches were failing from higher-than-expected peak currents. The solution to this was to stack two cores together to allow for larger flux and thus stay out of saturation.

The core geometry that works for this option was only available in MPP so two 55111-A2 Magnetics Inc. torroids were used together for the next inductor design. The inductance was again designed for  $34\mu\text{H}$  which resulted in 20 turns of 15 strands of #24AWG magnet wire. These cores were put into the circuit with the IR devices and the peak power condition was tested again. A few seconds into the test the devices blew again with approximately  $\sim 35^\circ\text{C}$  on the heatsink. This failure was not from over temperature and the cores were not saturating so the new theory was the switching frequency was too high to allow for the necessary amount of current. As mentioned in Chapter 2, the switching frequency was lowered and the inductors were redesigned and rewound.

## ***5.2 Experimental Setup***

For the FEC inverter competition, the bidirectional boost was laid out on 4-oz PCB and the power stage was built up populating both phases of both converters. The original design had the inductors and caps mounted under the board to allow for the gate drive and sensor boards to be mounted on the topside. Since the inductor cores changed along with the switching frequency, the inductors became too large to fit under the board as they hit the edge of the heatsink. Since the LEM pads were flipped upside down, the LEMs were mounted on the underside as were the screw terminals. This allowed for the low-side capacitors and the inductors to be mounted on the topside.

The high-side capacitors were placed on copper clad and attached to the PCB via mounting lugs and copper foil. This allowed for the gate drive board to sit cleanly on top of the power stage.

The board was designed so the switches could be mounted directly to the heatsink below by bending the leads by  $90^\circ$ . The switches are held in place by a solid metal bar that runs over the top of all the devices and is held to the heatsink at six places for good thermal contact. Non-electrically conductive thermal padding was placed under all the

devices. For most of the testing, only one phase is used and thus only two high-side capacitors are necessary.

### **5.2.1 Sensor Boards**

To sense the bus and battery voltage, the FEC sensor design was used. Due to unresolved layout problems on the multi-channel sensor board, two different boards are connected to the DSP. This requires an additional supply for +15V due to a coupling problem between sensors tied to the same +15V ground. Once this issue is worked out, the voltage sensors function without further problems.

### **5.2.2 Optocoupler**

The optocoupler circuit was designed and laid out on the PCB but was not populated due to time constraints for the competition. For testing and verification of the sensorless control scheme, the circuit was built on a small PCB and connected directly to the device leads. The optocoupler circuit requires only 3.3V which is taken from the DSP eval board and the outputs are connected to the mux through a shielded cable. The mux that selects the correct capture output was implemented on another PCB and connected to the DSP header.

### **5.2.3 Gate Drive Board**

A ribbon cable was run from the PWM outputs to the gate drive board that also contains fault, reset and 5V signals. For most of the testing, only one converter was run so the other cable was disconnected. To assure the gate drives do not float high, a 2.2k $\Omega$  resistor was placed on each input to the 316J gate drive chip to pull it low when it is not being driven.

### **5.2.4 Loads**

The main load for power testing is a converted three-phase heater that has six 9 $\Omega$  coils that are each rated for 5kW at 240V. These are tied in parallel or series depending on the desired load and a contactor is strategically placed so that a load step or dump can be simulated.

The same procedure is used for the light load as two 500 $\Omega$ , 250W resistors are placed in parallel to create 160W at 200V. A contactor is setup to place another 250 $\Omega$  in parallel to create 320W at 200V.

The batteries are used as a dynamic load for buck mode along with the resistive load bank.

### 5.2.5 Power Supplies

The main power supply for boost mode testing are the four 42Ah Pb acid batteries. The larger Ah batteries are used due to their ability to be used longer between charging which is very time consuming.

Buck mode testing will rely on a Lambda 300V, 2.86A supply that can 570W at 200V.

## 5.3 Inductor Tuning and Measurement

To verify and correctly tune the inductor values, a loop of wire was inserted in series with each inductor so a current clamp could be placed around it. The converter was run mainly in boost mode, due to the larger power, and the inductor current and voltage was measured on the scope. The designed number of turns resulted in too large of an inductance value so four turns were taken off and the inductors were again measured. This was repeated until the inductance of each was near the design goal of 59 $\mu$ H which resulted in 29 turns. Once the number of turns was set, a more accurate measurement was done by averaging multiple measurements at various power levels. The final inductor values are shown in Table 5.1.

**Table 5.1 Phase 1 and Phase 2 Inductor Values as Measured**

Inductor 1	Inductor 2
57.3 $\mu$ H	60.5 $\mu$ H

As seen in Table 5.1, the values of the inductors differ by a few microhenries. This will cause a current unbalance in the phases which should not be large enough to cause problems. Since the sensorless average current scheme measures only phase 1 current, the full input or output current will not be double the single phase measured current.

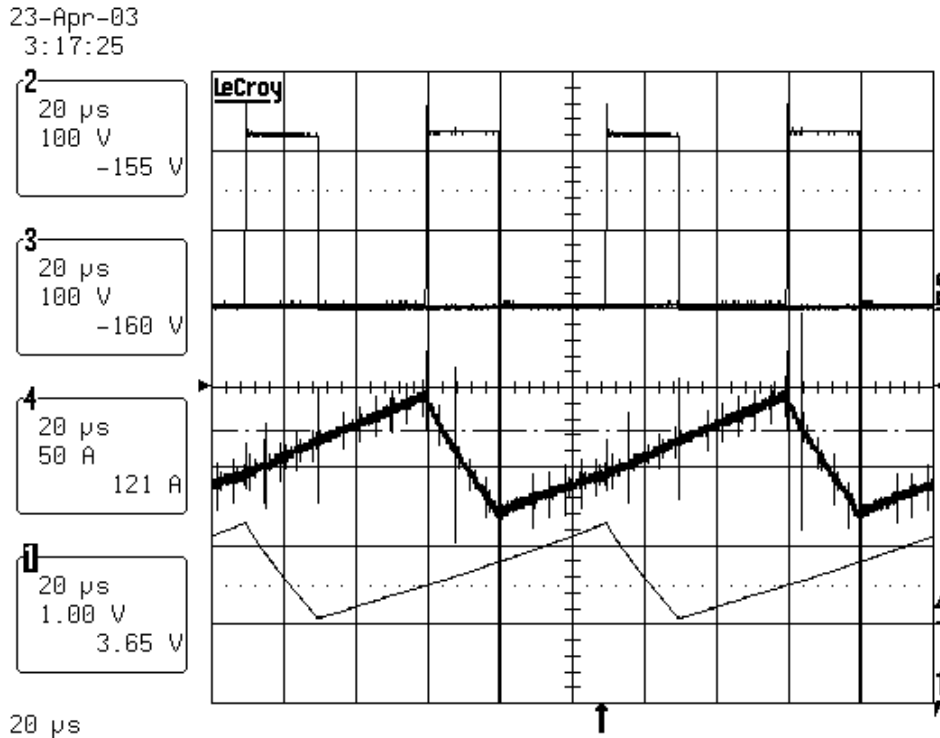
## ***5.4 Boost Mode Open Loop Testing of Power Stage***

As mentioned in the power stage design section, there were many iterations of the power stage design until the desired operation was achieved. This section will only cover testing of the final design.

To allow for the duty-cycle to be increased and decreased while the converter is running, momentary toggle switches are connected to the DSP I/O pins. One was set to increment and the other to decrement the duty-cycle and the third is set to toggle between buck and boost modes.

The main goal of the power stage testing is operate at 2.5kW for one minute. Since both converters are identical, only one will be tested to verify the design. To create 2.5kW at 200V requires 16 $\Omega$  of resistance but due to the multiples of 9 $\Omega$  in the load bank, only 18  $\Omega$  can be used. To get the full power output, the voltage must be increased to 212V.

For this test, the 18  $\Omega$  load was connected directly to the output of the boost converter and the duty-cycle was set to zero. The batteries were connected using a 12V contactor and the output voltage was increased using the up button until the output was 216.8V. A timer was started and the countdown to one minute began. To monitor the device temperature, a thermocouple was placed on the heatsink between the phases of the +V<sub>Batt</sub> converter. This temperature was closely watched during the duration of the test for any signs of overheating. A current clamp was placed on the phase 1 inductor current and the output of the LEM on phase 2 was also watched on the scope. Shown in Figure 5.1 is a screen capture during the 2.5kW duration test showing from top to bottom the phase 1 switch voltage, phase 2 LEM output and phase 1 inductor current.



**Figure 5.1 Boost Peak Power Test**

After the test was completed successfully, the converter operating parameters were calculated and compared to the design.

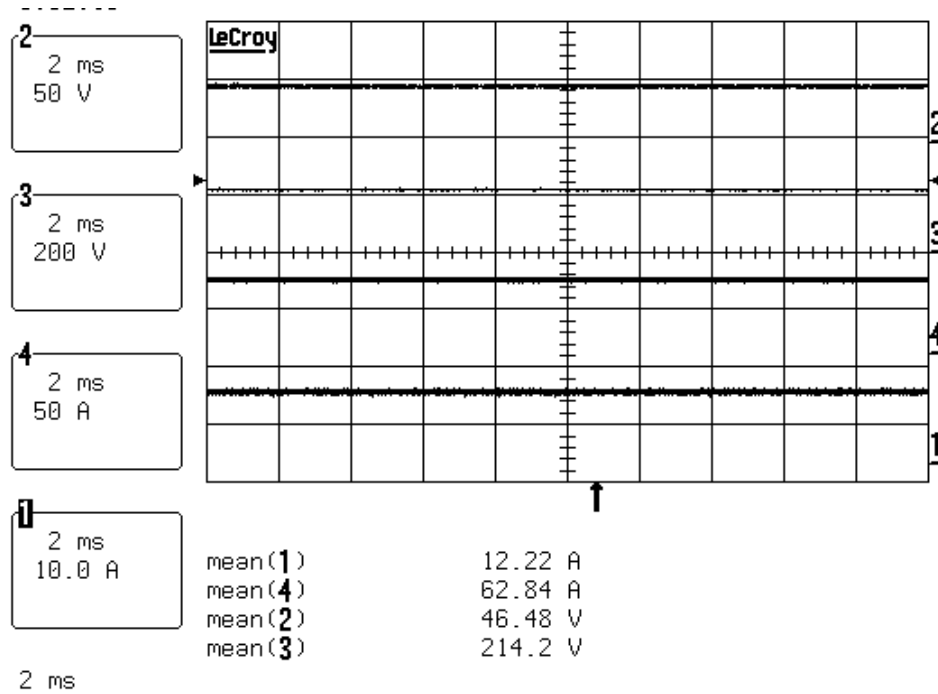
**Table 5.2 Boost Peak Power Test Inductor Current Results**

Parameter	Design	Measurement
$I_{1-PK}$	62A	63.6A
$I_{2-PK}$	62A	62.2A
D	0.7677	0.793

From Figure 5.1 it can be seen that the converter is slightly into CCM at this high power level. This is caused by a larger than expected output power and a very low battery voltage due to the series voltage drop at high current. For the sensorless current control to work, the converter must always remain in DCM. A simple fix would be to remove one turn from each inductor which could be done with little modification to the controller design.

After the first test, another shorter run was done to measure the efficiency of the converter. The screen capture in Figure 5.2 shows the DC values of the converter output;

channel 1 is the output current, channel 2 is the input voltage, channel 3 is the output voltage and channel 4 is the input current.



**Figure 5.2 Boost Peak Power Average Input and Output Voltage and Current**

From this data, the efficiency of the converter can be compared to the design target of 90%.

**Table 5.3 Boost Peak Power Efficiency Results**

Input Power	Output Power	Efficiency
2920.8 W	2617.5 W	89.6%

The experimental results match very well with the design.

### 5.5 Buck Mode Power Stage Testing

Due to the time and power supply constraints, buck mode was never tested at any power above 530W. The only efficiency test that was performed in buck mode was with

inductors designed around Magnetics Inc. torroid 77192-A7 with a 20 kHz switching frequency. To simplify the test, the resistive load bank was used by placing two 9Ω coils in parallel for a 4.5Ω resistive load. Using the 300V Lambda supply for the 200V bus voltage, the converter was run open-loop test for 10 minutes at 48.8V for 529W of output power. Figure 5.3 shows the converter waveforms during the test which are  $V_{CE}$ ,  $I_{L1}$  and  $D_1$  from top to bottom. Using the output current as measured by the power supply, an input power of 592W was calculated yielding an efficiency of roughly as 89%.

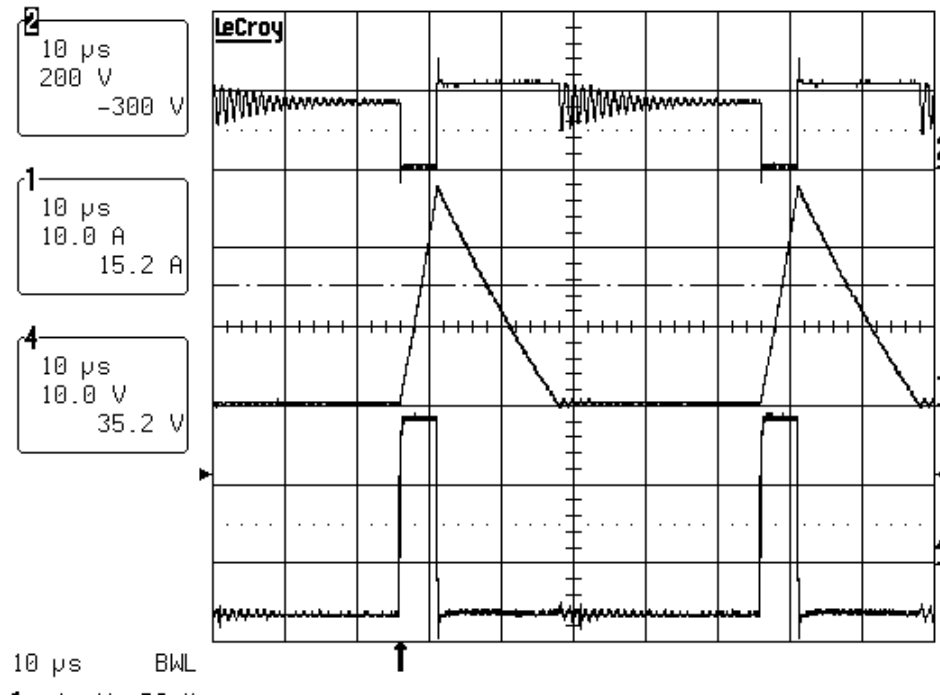


Figure 5.3 Buck Mode Power Test

## 5.6 Capture and Average Current Testing

An important result of the testing is to verify the correct calculation of the average current in phase 1. This is done by looking at the inductor current with the current probe and comparing the average value with that calculated in the DSP. The average current should be correct over the full range of power but will only be tested up to 1.5kW.

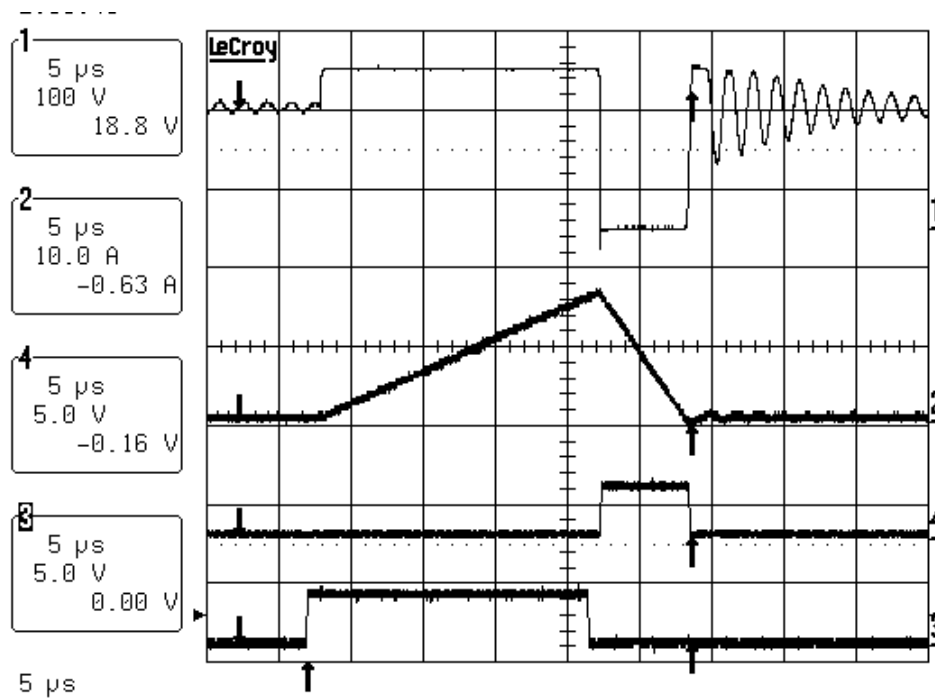
The first test was performed at 160W of resistive load. A breakpoint was set after the average current calculation in the code and the DSP was halted to inspect the

variables. The scope was then stopped and the screen captured.  $I_{pk}$  is calculated in the DSP but is used internally in the  $I_{avg}$  calculation and thus is not available.

**Table 5.4 Boost  $I_{avg}$  Verification at 160W**

Parameter	DSP	Probe
$D_1$	19.507 $\mu$ s	19.314 $\mu$ s
$D_2$	6.372 $\mu$ s	6.110 $\mu$ s
$I_{pk}$		16.1A
$I_{avg}$	2.058A	2.052A
% Error		0.29%

Figure 5.4 shows the scope screen from which the values in Table 5.4 were measured. From top to bottom the waveforms are:  $V_{diode}$ ,  $I_{L1}$ , optocoupler output and  $D_1$ .



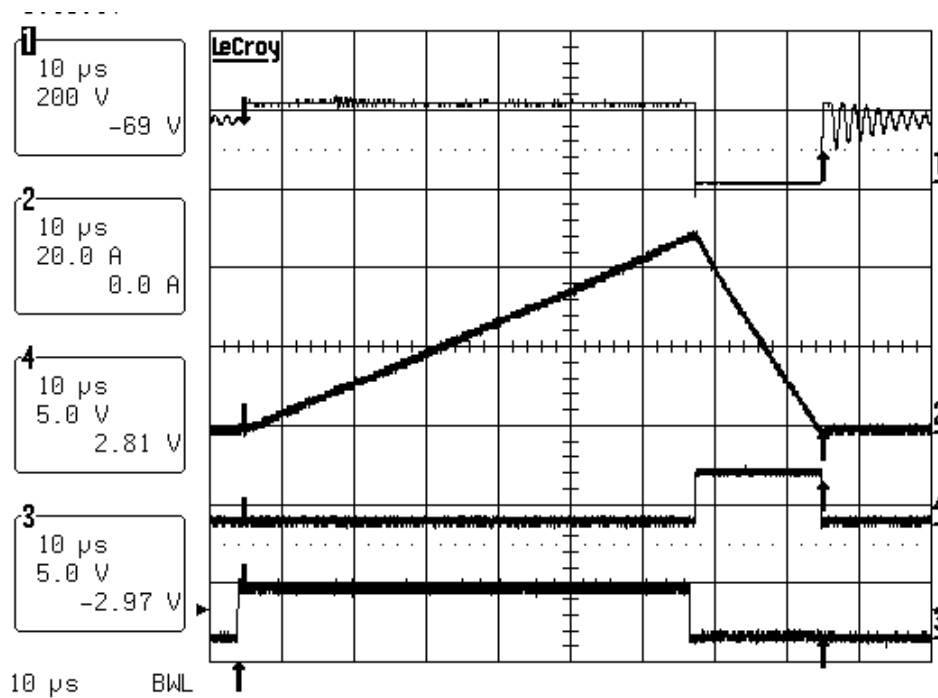
**Figure 5.4 Boost  $I_{avg}$  Measurement Waveforms at 160W**

The same procedure was done for 320W, 740W and 1.5kW and the results are summarized in Table 5.5.

**Table 5.5 Boost  $I_{avg}$  Verification at 320W, 740W and 1500W**

	320 W		740 W		1500 W	
Parameter	DSP	Probe	DSP	Probe	DSP	Probe
$D_1$	27.839 $\mu$ s	27.606 $\mu$ s	42.112 $\mu$ s	41.872 $\mu$ s	62.944 $\mu$ s	61.596 $\mu$ s
$D_2$	8.499 $\mu$ s	8.463 $\mu$ s	12.622 $\mu$ s	12.588 $\mu$ s	17.450 $\mu$ s	16.906 $\mu$ s
$I_{pk}$		22.6A		33.9A		48.8A
$I_{avg}$	3.910A	4.095A	9.339A	9.217A	19.355A	19.25A
% Error	4.52%		1.32%		0.55%	

The scope capture at 1500W is shown in Figure 5.5 and the waveforms from top to bottom are  $V_{diode}$ ,  $I_{L1}$ , optocoupler output and  $D_1$ .



**Figure 5.5 Boost  $I_{avg}$  Measurement Waveforms at 1500W**

From these tests, it is shown that the sensorless average current measurement is very accurate over a wide range of power levels. The error may be attributed to the inductor voltage which is dependant on the voltage drop of the switch and the voltage drop across

the series resistance of the inductor. This is also a snap-shot at a particular instant during which the value can have a large error either way, but on the average be quite precise.

### 5.6.1 Phase 2 Average Current Measurement

As mentioned previously, the inductor values are not the same thus there will be a difference in the average value of the individual phases. To try to quantify this difference, current clamps were placed on each inductor and the input current. The converter was run at 740W and 1500W and the average current values were recorded on the scope.

At 740W the average value of the phase 1 current was 9.708A while the average of phase 2 was 9.064A as shown in Figure 5.6. The value of the average current at this point as calculated by the DSP was 9.603A which is very close to the actual value.

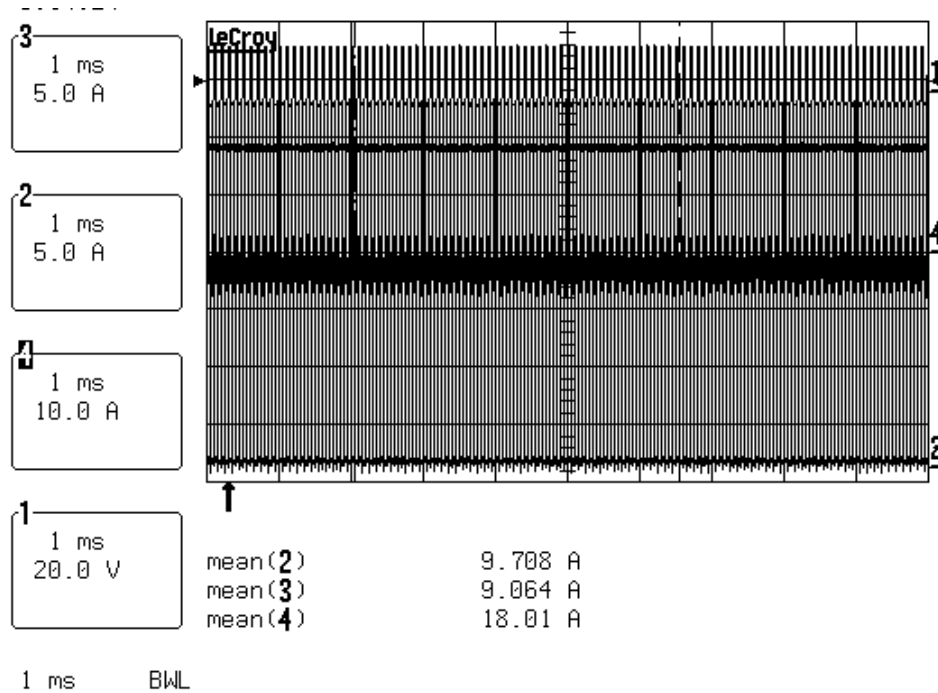
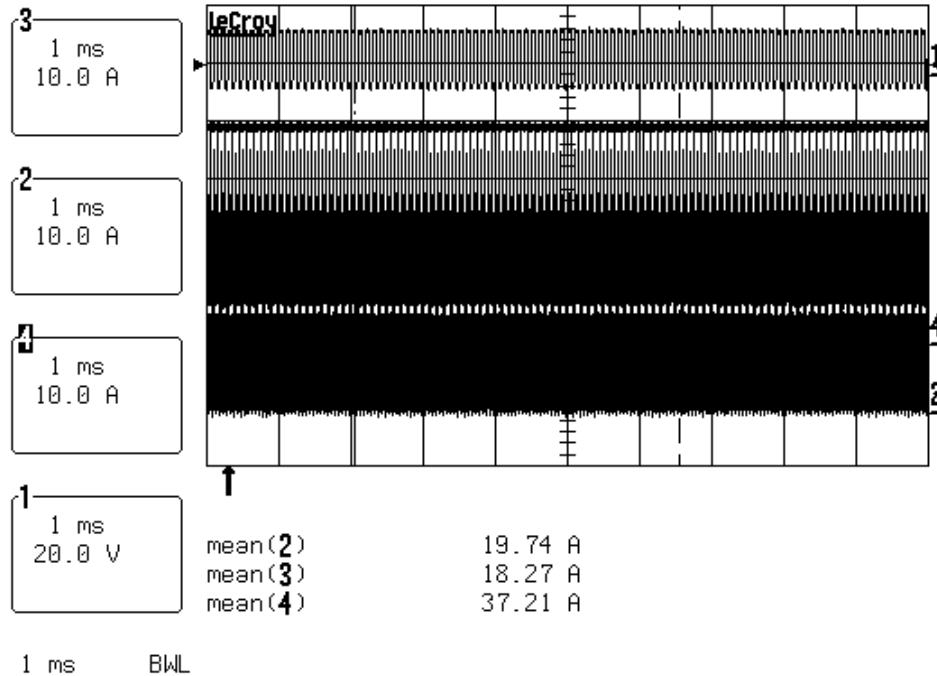


Figure 5.6 Boost Average Inductor and Input Current at 740W

This same test was performed at 1.5kW and the screen capture can be seen in Figure 5.7.



**Figure 5.7 Boost Average Inductor and Input Current at 1.5kW**

Again the average current measurement in the DSP was 19.978A very close to the scope average of 19.74A. The goal of this test is to show the accuracy of the sensorless average current measurement and also determine a factor that can be applied to the average value to predict the total output or input current. Using one value from a buck mode test, the ratio of  $I_1$  to  $I_2$  current can be written as in (5.1).

$$I_2 = \frac{I_1}{1.0811} \quad (5.1)$$

This will be used in buck mode to verify the correct value of the command current.

### 5.7 Boost Dual-Loop Controller Testing

With the hardware and software working as designed, the DSP was setup for closed-loop operation. The toggle switches are setup to control the open and closed loop operation of the converter. The third button is set as the Go button that starts the soft-

start operation. Once the controller reaches 200V on the output, the loop is closed automatically.

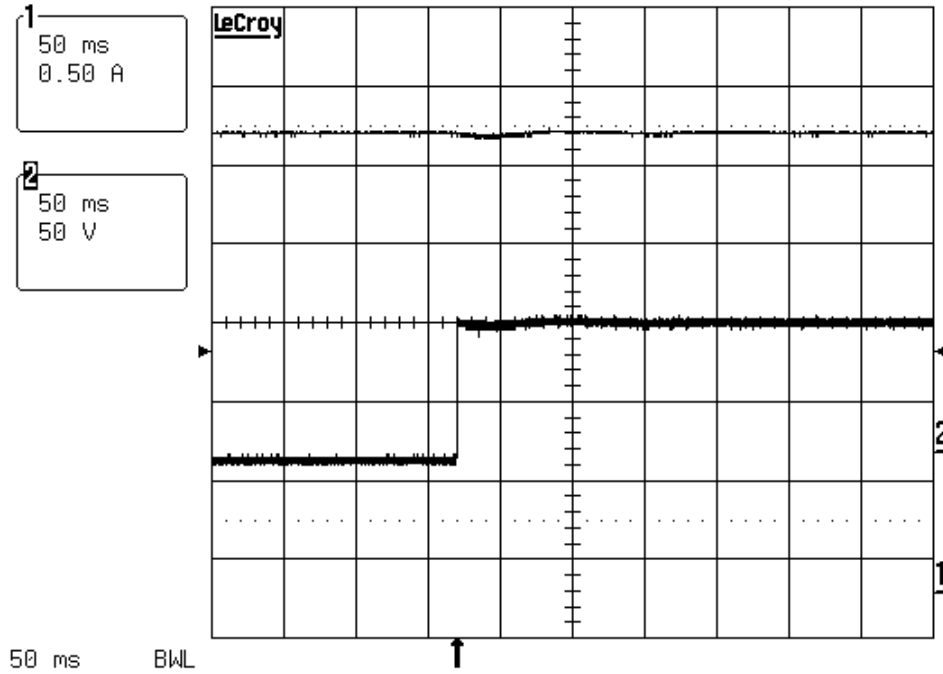
### **5.7.1 Light Load Testing**

The light load was tested first to determine stability of the controller at low power. The 500 $\Omega$  resistors as described earlier were connected to the output and the contactor and additional resistance is placed in parallel with the load for to ability to select either 160W or 320W at 200V.

Since the output capacitors are only rated for 250V, a voltage limit was placed in the DSP code to shut off the switches if the voltage exceeds 215V. For added protection, the converter is run open-loop at 320W and the peak duty-cycle was recorded and a hard limit was placed in the code slightly above this. So if the controller begins to go unstable, the output duty-cycle will be limited as well as the output voltage.

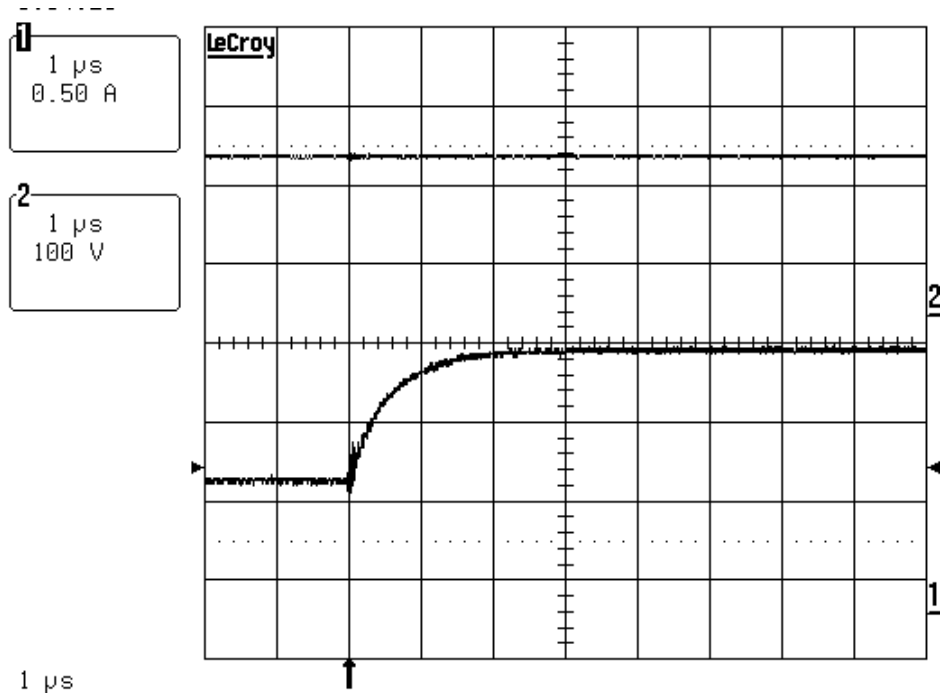
With these protection measures implemented, the controller was reset and the batteries are connected. Due to the operation of the interrupts, there must be a small duty-cycle to trigger the capture interrupt or the controller will cease to function. This causes the output voltage to start at around 70V. From this point, the soft-start button was pressed, the output voltage was ramped up 200V and the loop was closed by the controller.

Using the contactor, the load is stepped from 160W to 320W and the output voltage and current as captured on the scope are shown in Figure 5.8. The top waveform is the output voltage and the lower waveform is the output current. As seen from the screen capture, there is very good regulation during this step as output voltage stays within 5%.



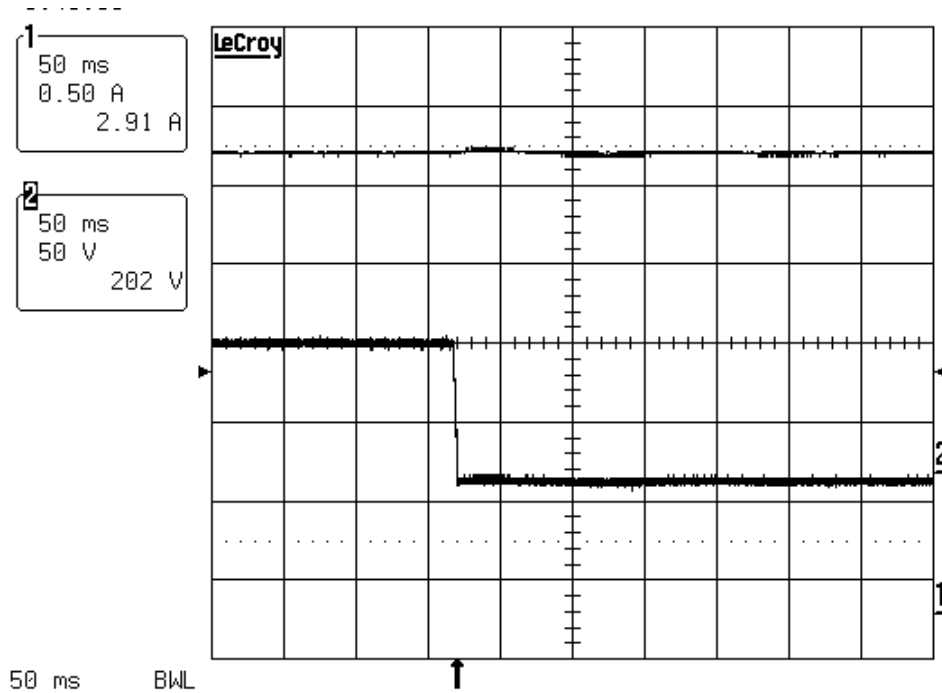
**Figure 5.8 Boost Transient Response from 160W to 320W**

To see in more detail the current step, the time scale was reduced to 1 $\mu$ s and the load step was repeated. As shown in Figure 5.9, there is some ringing at the very beginning of the step. This may be caused by the mechanical operation of the contactor. The output voltage is shown also and there is no noticeable deviation on this small time scale. The current takes about 3 $\mu$ s to rise from 800mA to 1.6A.



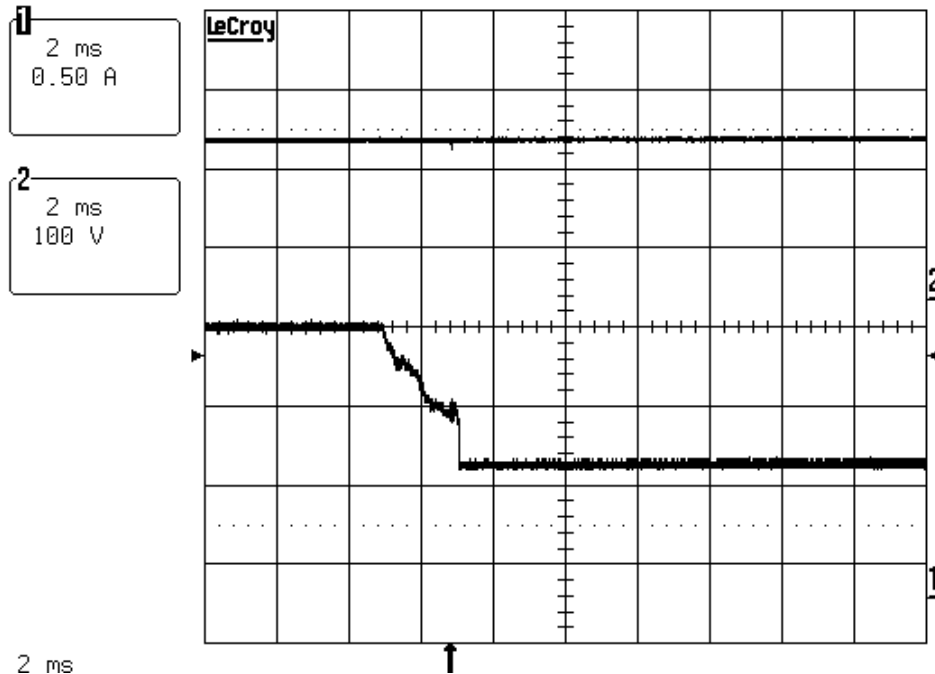
**Figure 5.9 Boost Transient from 160W to 320W Smaller Time Scale**

To test the output regulation during a load dump, with the load at 320W, the contactor is disabled and the load drops back 160W as shown in Figure 5.10.



**Figure 5.10 Boost Transient Response from 320W to 160W**

As expected, there is a bit more overshoot and ringing in the output voltage, but the variation is very small and still remains within 5% of the average value. Again the time scale was reduced to show detail on the transition and the dump repeated. In Figure 5.11 the top waveform is the output voltage and the lower waveform is the output current.



**Figure 5.11 Boost Transient from 320W to 160W Smaller Time Scale**

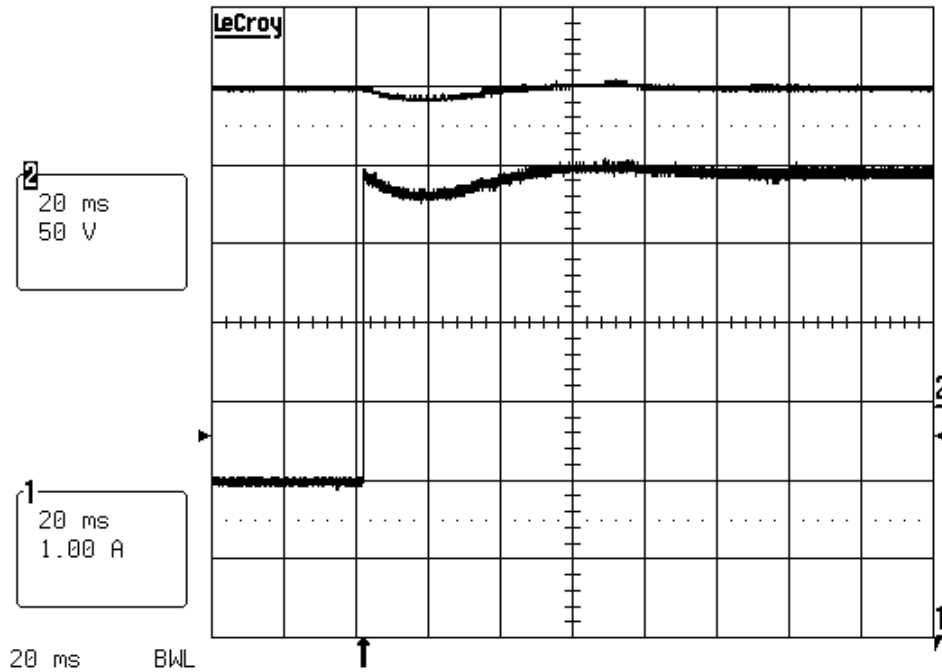
This is one of the cleanest waveforms that was captured. Repeating the dump over and over again, the waveform was never the same twice, which leads to the conclusion that the noise on the transition is caused by the contactor. The variations on the current may be due to the mechanical operation of the contactor or arcing as it breaks the voltage. Again in this time scale, the output voltage does not change very much at all, but there is a slight rise before settling again.

### 5.7.2 Large Load Boost Controller Testing

The final set of tests that need to be performed on the controller are those at high power. Since the load bank only allows for multiples of  $9\Omega$  loads, the two load levels will be 200V into  $54\Omega$  for 740W and 200V into  $27\Omega$  for 1.481kW. Since the converter is limited to DCM operation, the peak power case will not be tested at this time.

The output was reconfigured for the two load levels and a contactor was placed in the loop to select either  $54\Omega$  or  $27\Omega$ . The DSP and duty-cycle were reset and the batteries were connected. As with the light load case, there needs to be a small duty-cycle to allow the proper operation of the interrupts. The initial output voltage was 56V and by pressing the soft-start button, the voltage was ramped up to 200V and the loop

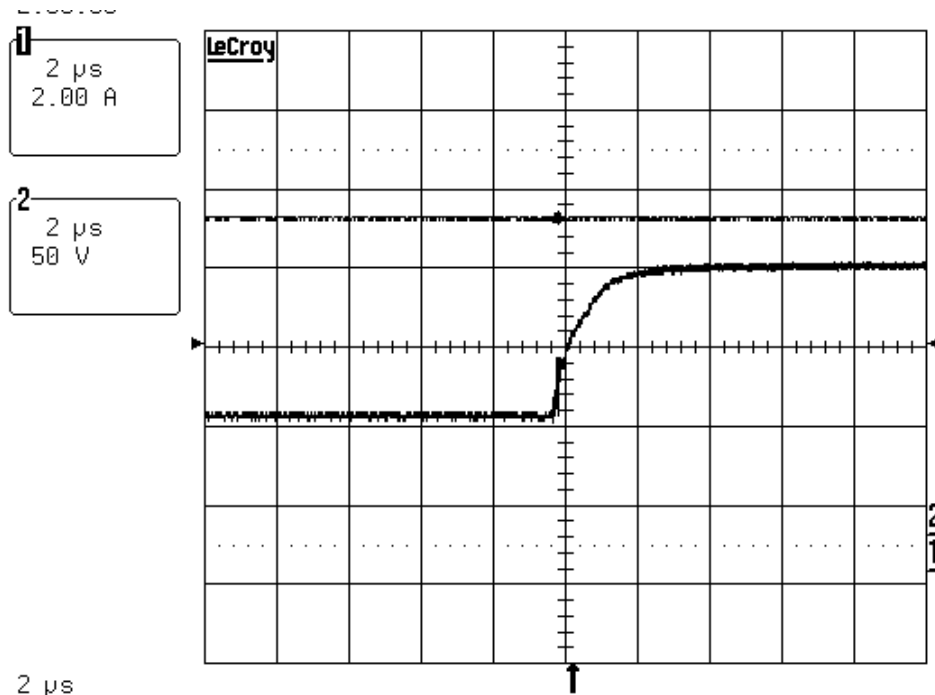
was closed. The value of the output voltage and current are displayed on the scope and the trigger was set to catch the rising edge of the current. The contactor was enabled and the step shown in Figure 5.12.



**Figure 5.12 Boost Transient Response from 788W to 1.57kW**

The controller performs very well, keeping the voltage within 7V of the nominal 200V. The oscillation takes a relatively long time to settle, around 46ms but that is expected due to the controller design and the low bandwidth. The actual power levels can be measured from the scope waveforms by using the cursors. The low load power level was 788W and the high power level was 1.576kW corresponding to 3.92A and 7.82A respectively.

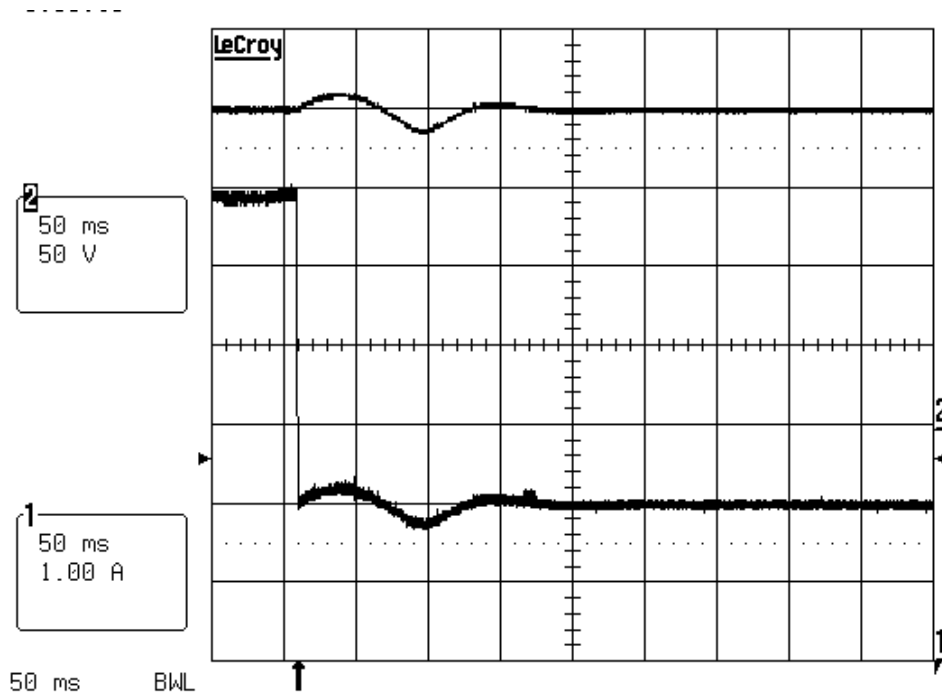
As in the light load case, the scope was zoomed in and set to catch the transition on a smaller time scale to give more information about the slew rate. The load was reduced, the scope was triggered and the load was stepped again. The screen capture of this is shown in Figure 5.13.



**Figure 5.13 Zoomed-In Boost Transient Response from 788W to 1.57kW**

In this case the step is very clean and takes a little over  $2\mu\text{s}$  to reach the new current. The voltage has a slight amount of noise corresponding to the initial step, but overall the output voltage remains constant.

The last test to perform is potentially the hardest to control, the large load dump. The contactor was energized and the scope was zoomed out and set to trigger on the falling current. When the contactor was released, the output voltage and current were captured and are shown in Figure 5.14.

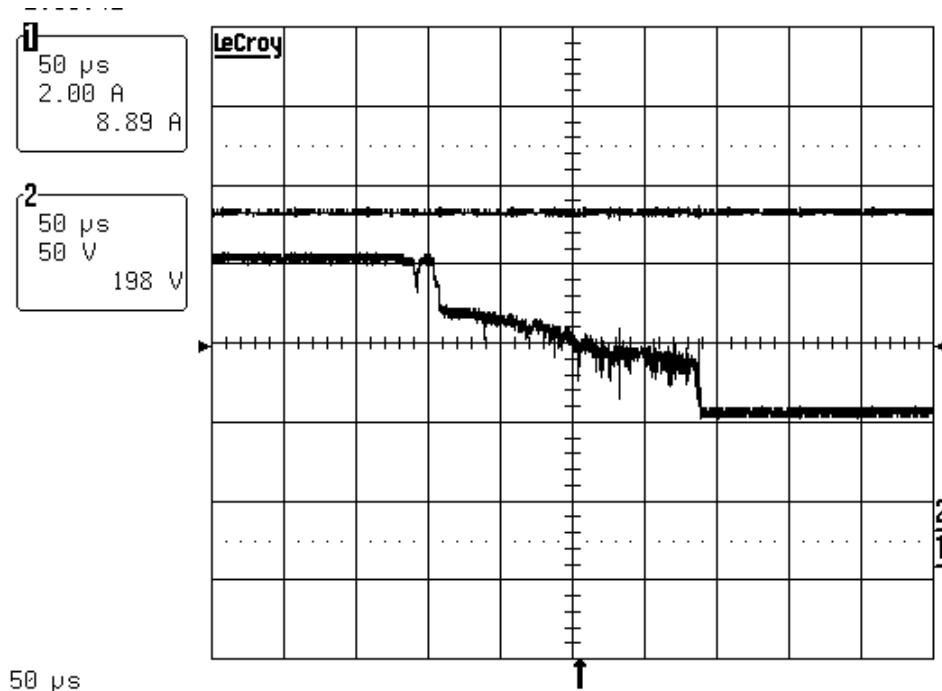


**Figure 5.14 Boost Transient from 1.57kW to 788W**

The response to the dump is much worse than the step but that is expected. The output voltage has a large oscillation that has a peak of 211V and a minimum of 186V. The voltage swing is 12.5% of the nominal which is not good, but based on the controller design, it is expected.

The same ripple is seen in the output current and it varies from 4.22A to 3.63A with about 3.92A of nominal current.

The scope was zoomed in on the current transition and the dump was repeated as shown in Figure 5.15. The noise that was seen in the low power case is also in the dump at high power. It would be expected for the noise to be more pronounced due to the larger current.



**Figure 5.15 Boost Transient from 788W to 1.57kW Smaller Time Scale**

To verify the operation of the sensorless average current sensor, the value of  $I_{avg}$  was written to one of the D/A channels as done in [28]. The current probe was moved to the input and the load step was repeated with both the D/A signal and the current probe on the scope screen. This will allow the scope to capture the real-time behavior of  $I_{avg}$ . The screen capture Figure 5.16 shows from top to bottom the output voltage, input voltage and the internal value of  $I_{avg}$ . It can be seen that  $I_{avg}$  tracks the input current very closely and the noise on the D/A signal is coupled switching noise from the converter and gate drives.

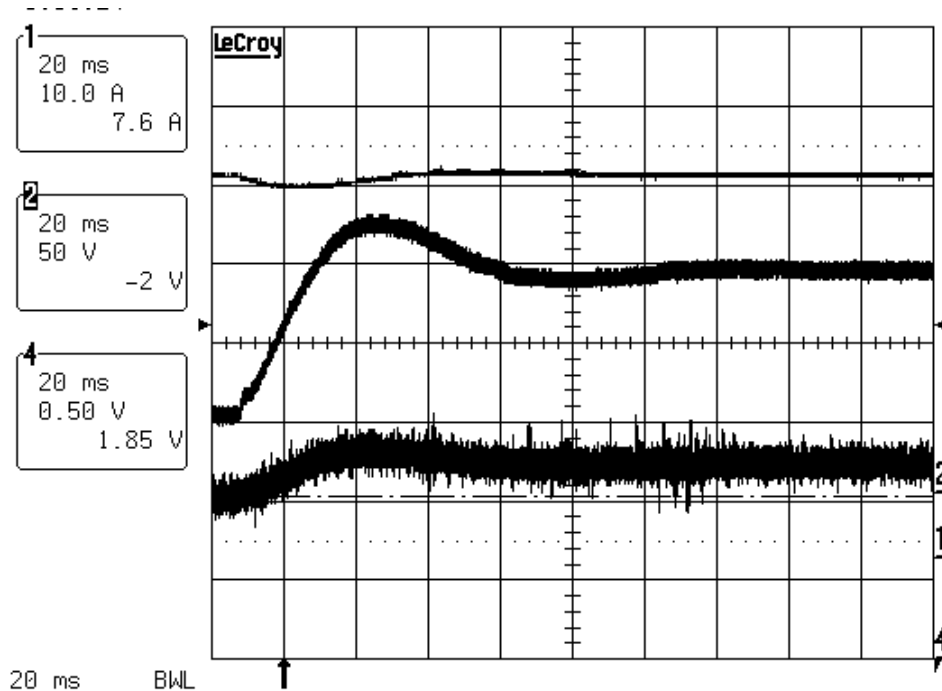


Figure 5.16 Input Current and  $I_{avg}$  During Boost Transient

## 5.8 Buck Mode Experimental Results

To simulate the normal load of the buck converter, batteries were used for most of the testing of the controller. There are two main results that are to be investigated: the ability of the controller to follow a current command and the stability of the current and voltage loops during charging. It is also important to show the smooth transition between current and voltage mode.

As in boost mode, the toggle switches were setup to control the open/closed loop operation. A small duty-cycle must be used upon start-up so the interrupts continue to operate. A soft-start was implemented along with setting the third button to enter closed-loop current mode operation. Once the controller switches to voltage mode, there is currently no mechanism to return to current mode but one could be implemented quite simply. It would be necessary to return to current mode if the available power decreases below the voltage-mode charging power.

### 5.8.1 Current Mode Control Reference Test

To test the ability of the controller to follow a reference, the batteries were connected, the contactor closed and the Lamda supply was set at 200V. By pressing the first button, the soft-start was initialized, ramping up the duty-cycle ever so slightly and then the loop was closed. To step the reference, the variable  $I_{ref}$  in the DSP was changed through the compiler and the charging current, along with the output voltage was displayed on the scope.

The first reference change was a step from 100 to 700 of  $I_{ref}$ . These values correspond to 788mA and 5.52A respectively and only correspond to a single inductor current. The scope screen capture with the output voltage, phase 1 inductor current and output current is shown top to bottom in Figure 5.17.

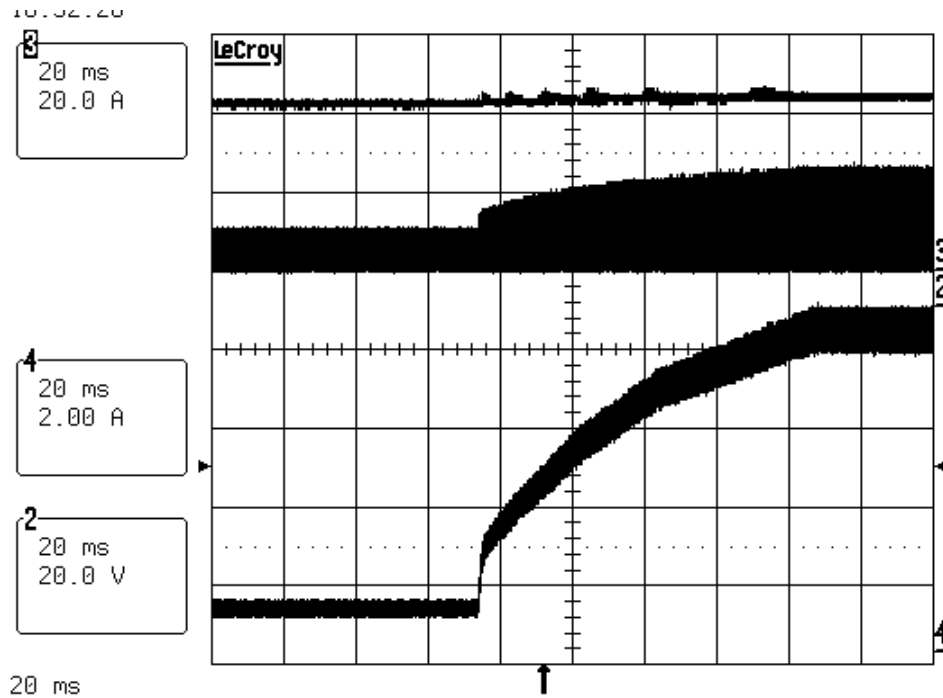


Figure 5.17 Buck Command Step from 100 to 700 of  $I_{ref}$

By using the cursors, the value of the low current can be measured as 1.11A and the stepped current is 8.13 A. The current reference was then decreased from 700 to 100 and the converter response was captured by the scope and shown in Figure 5.18.

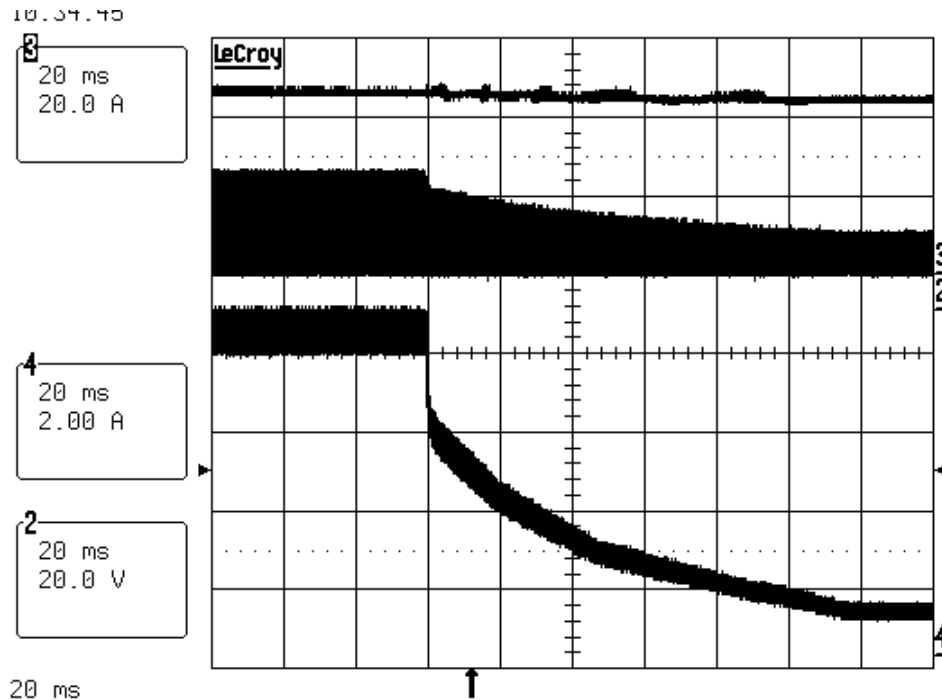


Figure 5.18 Buck Current Command Dump from 700 to 100 of  $I_{ref}$

Again the actual value of the high-level current is 8.13A and the low-level current is 1.11A. By calculating the expected full output current by multiplying the single phase current by two, the expected low output current becomes 1.58 A and the high output current should be 11.04. From the results of this test, this is definitely not the case as there is quite a large error on the high-level current.

To investigate this further, the batteries were discharged again and the converter was run in buck current mode. A current probe was zeroed and placed on both phase 1 inductor current and the total output current. The scope was set to average both of these values and the time scale was zoomed out. The controller was given a reference and a breakpoint was placed to check the corresponding value of the sensed current. The screen was saved to disk and this procedure was repeated for six values of  $I_{ref}$  ranging from 140 to 755. To more accurately compare the results, when the test was completed, the offsets on both current probes were recorded and used to correct the experimental data. The phase 1 inductor current probe had an offset of -220mA and the output current probe was offset by -90mA. The data was put into a Matlab script which calculated the sensed inductor current, percent error and the expected total output current, taking into

account the difference in the inductor values from (5.1). The steady-state error was also calculated and shown with the controller results in Table 5.6.

**Table 5.6 Buck Current Mode Reference Tracking Results**

$I_{ref}$	$I_{avg}$	$I_{error}$	$I_{probe}$ (A)	$I_{DSP}$ (A)	% Error	$I_o$ (A)	Exp. $I_o$ (A)
140	52	88	0.424	0.410	3.3434	0.606	0.789
261	162	99	1.190	1.277	7.2907	2.062	2.458
364	264	100	2.152	2.081	3.3157	3.541	4.005
484	385	99	3.132	3.034	3.1202	5.273	5.841
645	546	99	4.422	4.303	2.6876	7.639	8.284
755	657	98	5.202	5.178	0.4620	9.156	9.968

The steady-state error is very constant over all current levels and this can be attributed to the digital implementation. When the new value of current is calculated and compare to the reference, an error is created which is sent to the compensator. The error is multiplied by both the coefficients in the poles and zeros section and by the integrator gain. The integrator gain in the compensator is 0.009826 which means that the smallest value of steady-state error possible is the inverse of 0.009826 which is almost 102. Since the DSP has a fixed level of precision, any number less than 102 results in zero thus the steady-state error shown column 3 in Table 5.6.

The percent error of the sensorless current vs. the current probe is very good even at extremely low currents but the expected total output current is not as accurate. There is a large difference between the measured and calculated output current for the lower powers corresponding to  $I_{avg}$  of 52, 162 and 264. The error becomes less as the current increases, but is still significant. This could be caused by the other phase taking less current than expected or by simple measurement error of the output current waveform that has a large ripple component about the DC value.

### 5.8.2 Buck Battery Charging Test

The final test that is necessary to verify the correct operation of the controller is the extended battery charging test. The goal of this test is to verify the stability of the current loop as the batteries charge, verify correct transition from current to voltage mode and then verify stability of the voltage loop as the batteries become charged.

After running the high-power boost mode tests, the batteries were significantly discharged and the converter was connected in buck mode. The contactor was energized, connecting the batteries to the converter, and then the supply was set to 200V. The current was increased until the limit of the supply was reached and the screen capture of this point is shown in Figure 5.19.

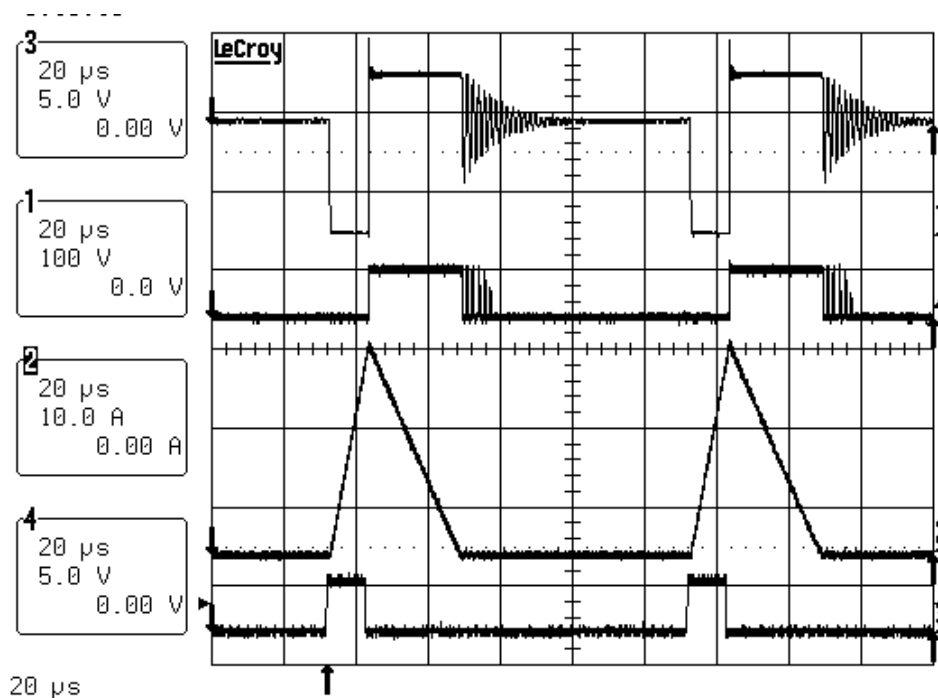


Figure 5.19 Buck Current Mode Charging at Peak Supply Power

In Figure 5.19 the input power is 542W and from top to bottom the waveforms are  $V_{CE}$ , optocoupler output,  $I_{L1}$  and  $D_1$ . To look at the total output power and average current, a probe was placed on each inductor current and another probe was put around the battery output connection. The battery charging voltage was also measured on the scope as shown in Figure 5.20. The current measurement in the DSP at this point was 484 which

corresponds to a phase 1 current of 3.81A as compared to 3.58A. The sum of the two phase currents is expected to equal the total output current, but is shy by 0.23A. Even at 6.4%, this is a very accurate measurement due to the low current being measured. One possible cause is the large ripple on the output current as it may affect the accuracy of the average also the current probe may not be zeroed or degaussed properly.

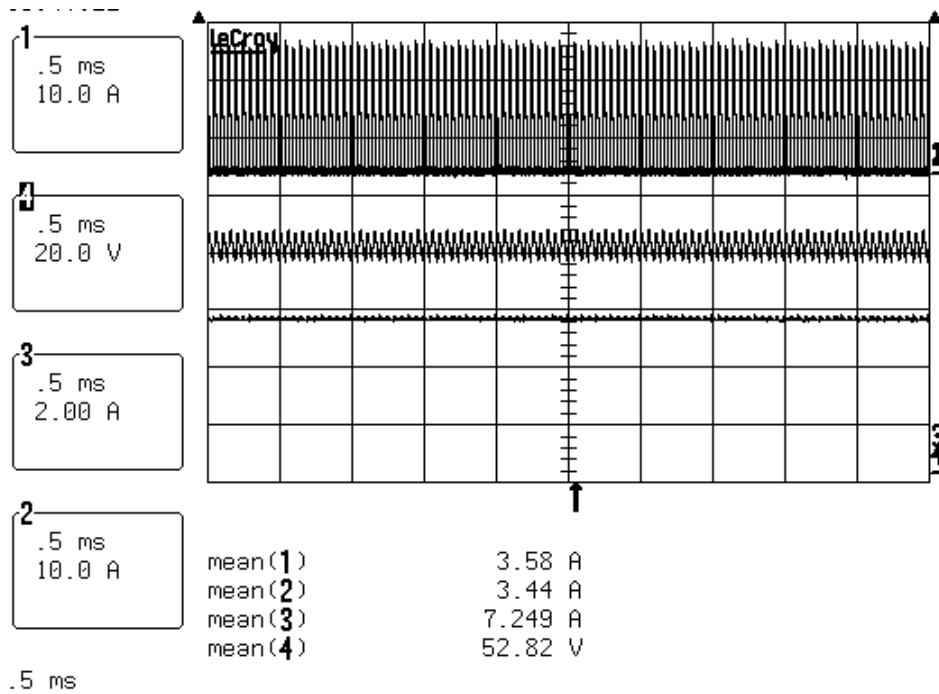


Figure 5.20 Buck Mode Charging at 383W

### Buck Mode Current to Voltage Transition

To verify the stable transition between current and voltage mode, the time scale was changed to 50ms and a bit was set to go high when the controller entered voltage mode. The batteries were slowly charged in current mode and when the charge voltage reached 58.8V, the controller switched to voltage mode charging. The scope was set to trigger on the rising edge of this bit and the screen capture is shown in Figure 5.21. From top to bottom, the waveforms in Figure 5.21 are: mode bit, output voltage, output current and phase 1 inductor current.

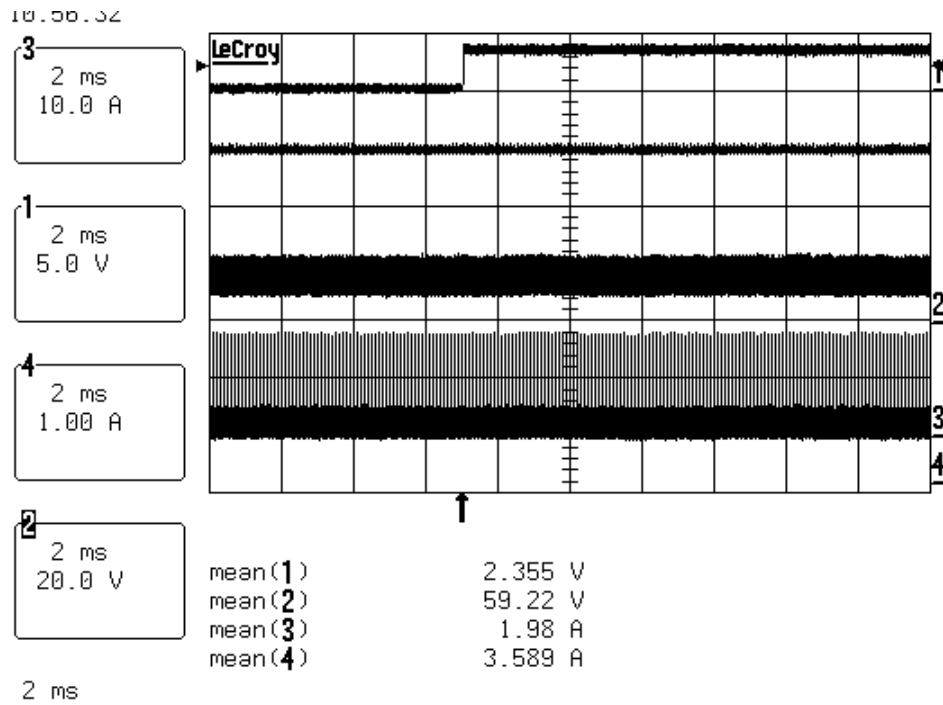


Figure 5.21 Buck Mode Charging, Current to Voltage Mode Transition

During the transition, there is very little deviation from the steady-state and the output voltage regulation is within  $\pm 2\%$  at an output power of 213W.

The final test of the buck voltage loop is the steady-state performance at very low power. Once the converter entered voltage mode, the charging was allowed to continue until the power was below 50W. At an input power of 44W, a breakpoint was used to check the sensorless value of the current and the converter waveforms were captured. In

Figure 5.22 the top waveform is  $D_1$ , the second is the diode voltage followed by phase 1 current and then the optocoupler output.

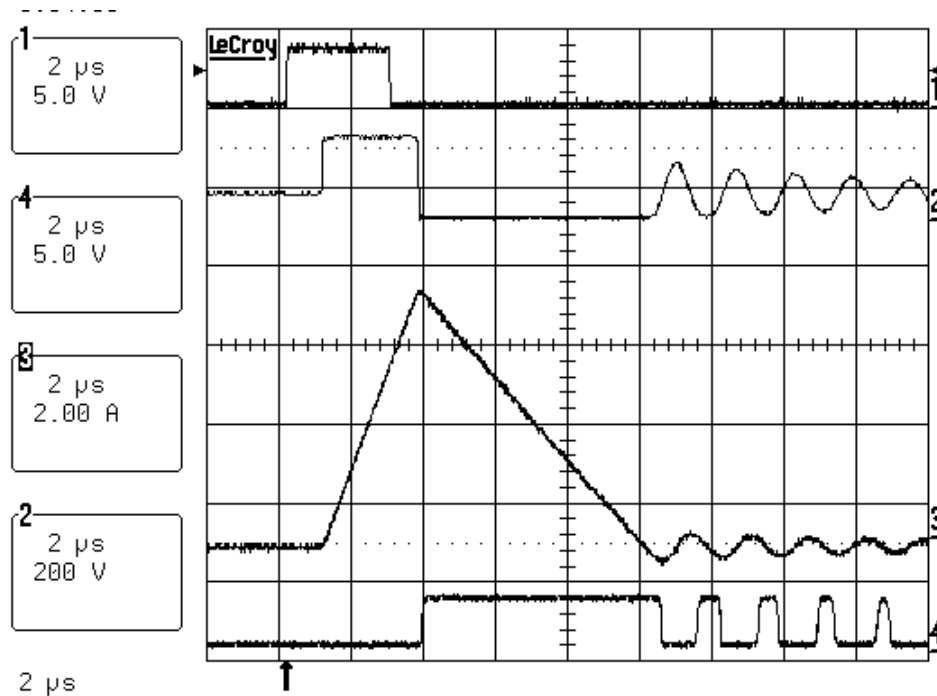


Figure 5.22 Buck Mode Voltage Loop Charging at Low Power

Table 5.7 Buck  $I_{avg}$  Verification at 44W

Parameter	DSP	Probe
$D_1$	2.896 $\mu$ s	2.660 $\mu$ s
$D_2$	6.549 $\mu$ s	6.634 $\mu$ s
$I_{pk}$		6.17A
$I_{avg}$	330.6mA	289.6mAA
% Error		14.2%

As seen in Table 5.7, the percent error is a bit high, but the sensed current is around a third of an amp. Much of the error can be attributed to experimental measurement error of both sets of values and noise on the voltage sensors that calculate the inductor voltage. As seen in Figure 5.22 the optocoupler has many glitches, but the converted fall time is very close to the actual diode response.

## Chapter 6 – Conclusion

### 6.1 *Summary*

This thesis investigated the application of a bidirectional boost converter in supplying peak and transient power in a fuel cell energy management system. The topology was selected based upon cost and performance and the control was implemented in a low-cost DSP using a sensorless average current control method.

To reduce cost while meeting the grounding specifications, the bidirectional boost was split into two parallel converters. The converter was designed to operate in DCM to reduce inductor size and to allow for a simple sensorless average current mode control.

Small signal analysis was performed on both modes to understand the plant dynamics so accurate compensators could be designed. In boost mode a classical dual-loop controller was designed around the sensorless average current mode control. In buck mode, two independent compensators were designed, one for current mode charging and one for voltage mode. Then these compensators were scaled properly and implemented in a fixed-point DSP using a C compiler interface. The sensorless average current mode control was also implemented in the DSP and used a simple digital optocoupler to convert the diode off-time in DCM.

The boost mode was tested at light and heavy load to verify the response and stability of the digital dual-loop controller. The system was stable over all loads tested, 160W to 1.57kW but was not tested at full power. At light loads the performance was very good in steady-state and also during transients. At heavy loads, the regulation was also very good, but during a 740W load dump, there was a large oscillation in the output voltage.

The sensorless average current calculation in the DSP was checked using a current probe over a wide range of loads in both modes. It was found that the sensorless value was very accurate even at light load with phase currents around a third of an amp.

## **6.2 *Future Considerations***

The converter and the associated control did work as expected and the results were satisfactory for the first implementation of the sensorless average current mode control.

The implementation of the digital control can also be greatly improved upon. The digital compensators were implemented with only one set of poles and zeros plus an integrator. For better dynamics a second set of poles and zeros can be included which would allow for better control over phase and gain margin, especially in the overall loop gain in boost mode.

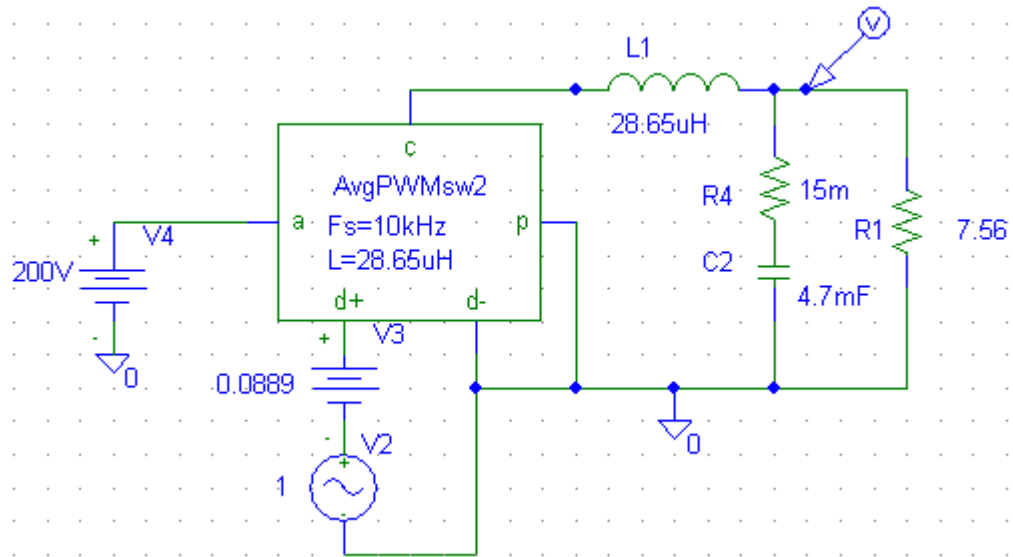
To further verify the dual converter solution, the both converters should be run in the full energy management system and tested with an actual fuel cell. This will require the control be extended to the second converter. To do this it will be necessary to speed-up the code to allow sufficient time for all the calculation to be done in one switching cycle. To increase the speed, it will be necessary to write most if not all of the code in assembly to take advantage of the DSP built-in instruction set. With both converters running closed-loop, it will be possible to verify if the chosen topology can regulate the bus voltages accurately during discharging and especially during charging.

With the integration of the full system an intelligent power-management scheme can be developed around the sensorless average current result. With this information, the power in and out of the bidirectional converter can be known and that information can be fed to the system controller that interfaces with the fuel cell.

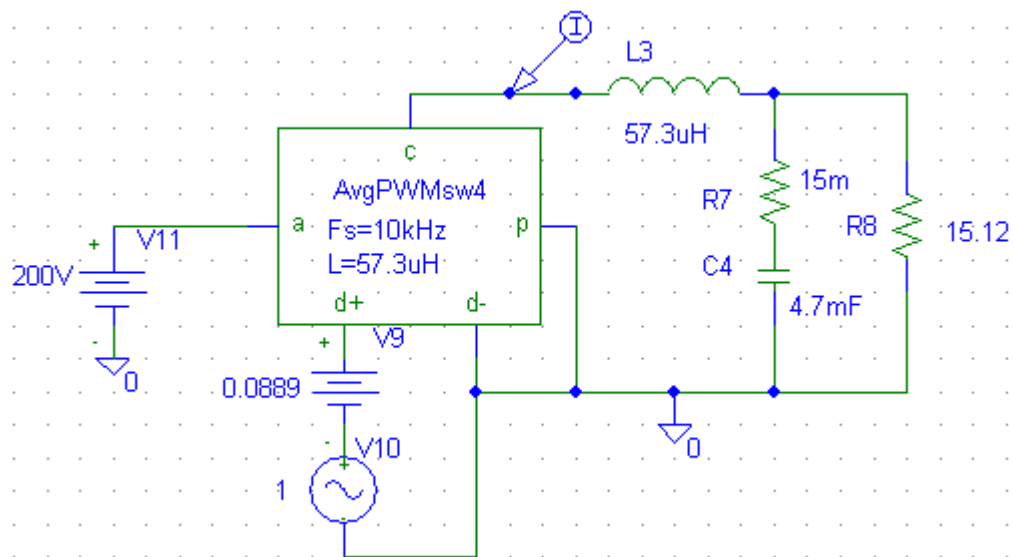
To reduce the inductor size and increase the control bandwidth, new discrete devices can be used that allow for large current and thus an increased switching frequency.

## Appendix A - Pspice Average Models

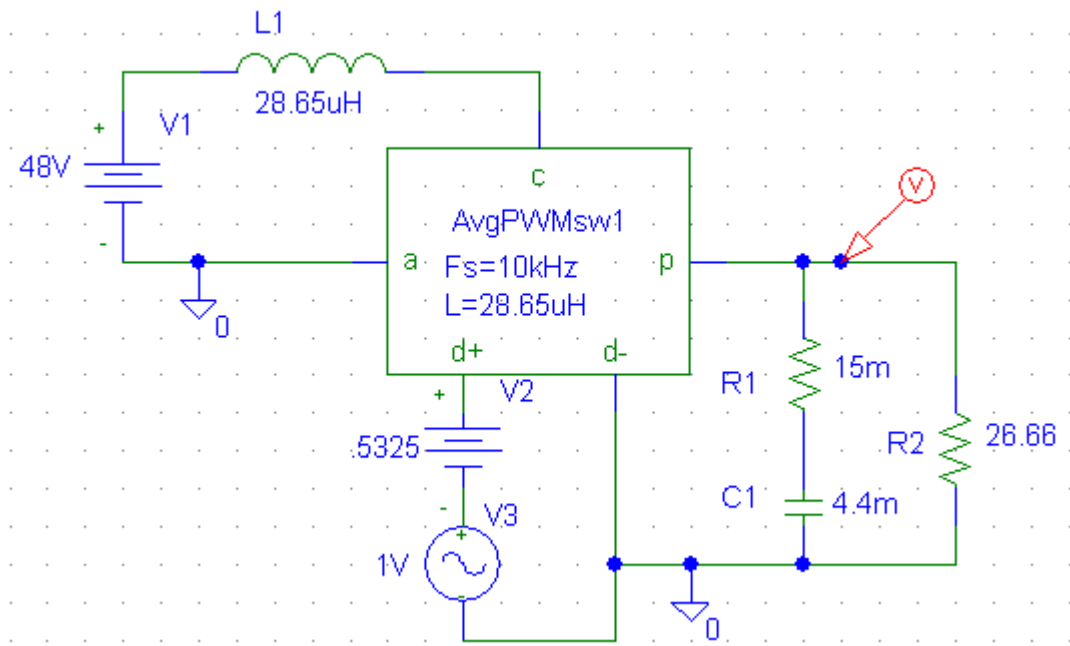
### A.1 Buck Gvod



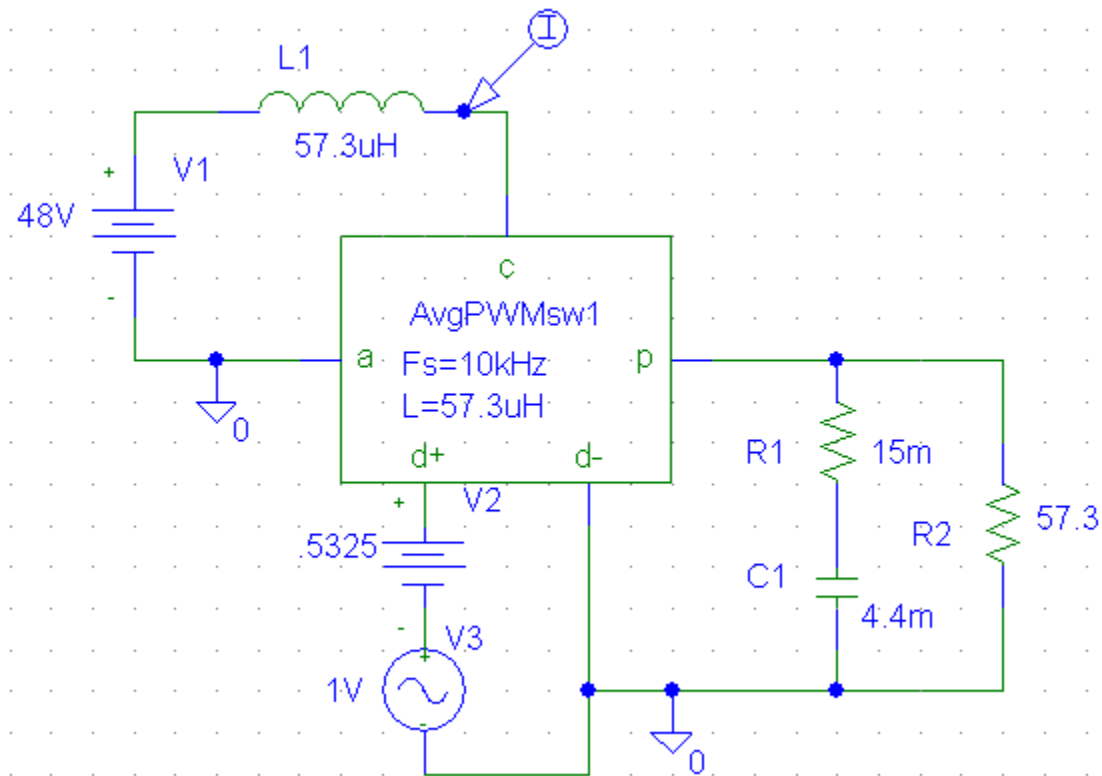
### A.2 Buck Gild



### A.3 Boost Gvod



### A.4 Boost Gild



## Appendix B - Boost Matlab Code

### B.1 Boost Voltage Loop Design

```
% Boost Voltage Loop Design

Vo=200;
Vg=48;
Co=4.4e-3;
L=57.3e-6/2;
P=1.5e3;
R=Vo^2/P;
Rc=15e-3;
Rl=0;
fs=10e3;
Ts=1/fs;

M=Vo/Vg;
s=zpk('s');
z=zpk('z',Ts);

K=2*L/(R*Ts);

syms D1

%Solve for DCM duty cycle
D=max(eval(solve((1+sqrt(1+4*D1^2/K))/2-Vo/Vg)));
D2=(K/D)*(Vo/Vg);

% Small Sigal Parameters from Dr. Lee's notes
Vac = Vg;
Io = Vo/R;
Ip = Io;
Vcp = Vo-Vg;
Ia = (M-1)*Io;
gi = Ia/Vac;
ki = 2*Ia/D;
ko = 2*Ip/D;
go = Ip/Vcp;
gf = 2*Ip/Vac;
Vap = Vo;

Gvod = -(ki*go*Rl+ki*go*s*L-ko*Rl*gi-ko*s*L*gi-ko+gf*ki*Rl+gf*ki*s*L)/ ...
(1+gf*Rl+go*Rl+R*s*Co+Rc*s*Co+s*L*gi+Rl*gi+Rl*gi*R*s*Co+Rl*gi*Rc*s*Co+ ...
s^2*L*gi*R*Co+s^2*L*gi*Rc*Co+go*s*L+gf*s*L+R*go+R*go*Rl*gi+gf*Rl*R*s*Co+ ...
gf*Rl*Rc*s*Co+gf*s^2*L*R*Co+gf*s^2*L*Rc*Co+go*Rl*R*s*Co+go*Rl*Rc*s*Co+ ...
go*s^2*L*R*Co+go*s^2*L*Rc*Co+R*go*s*L*gi+R*go*Rl*gi*Rc*s*Co+ ...
R*go*s^2*L*gi*Rc*Co+R*go*Rc*s*Co)*R*(1+Rc*s*Co);

Gvodz = c2d(Gvod,Ts,'zoh'); %Discrete Plant

% Compensator design iterations

% Cs = 49851.7562*(s+31.93)/s/(s+1.522e004); %60Hz cutoff
```

```

% Cs = 181857.6864*(s+31.93)/s/(s+1.522e004); %206Hz
% Cs = 62297.5545*(s+31.93)/s/(s+1.522e004); %70Hz
% Cs = 17813.4528*(s+31.93)/s/(s+1.522e004); %20Hz
% Cs = 44265.5494*(s+31.93)/s/(s+1.522e004); %50Hz
% Cs = 27861.08548*(s+31.93)/s/(s+1.522e004); %30Hz
% Cs = 9230.0983*(s+31.93)/s/(s+1.522e004); %10Hz Stable at 200W
% Cs = 158667.3925*(s+215.2)/s/(s+1.522e004);
% Cs = 679130.7751*(s+1.632)/s/(s+1.522e004);
% Cs = 797339.3796*(s+32.47)/s/(s+1.522e004);
% Cs = 8553.6163*(s+2906)/s/(s+1.496e004);
% Cs = 449589.5659*(s+1450)/s/(s+1.514e004);
% Cs = 351989.748*(s+899.2)/s/(s+1.514e004);
% Cs = 79266.818*(s+899.2)/s/(s+1.514e004);
% Cs = 18156.1716*(s+899.2)/s/(s+1.514e004); % A - Works at 888W
% Cs = 38789.2888*(s+2515)/s/(s+1.514e004); % Unstable with C GI
% Cvs = 19626.7239*(s+4868)/s/(s+1.514e004);

% Cz = 32.2875*(z-0.9102)/(z-1)/(z-0.409);
% Cz = 3.0199*(z-0.9102)/(z-1)/(z-0.409);
% Cz = 1.0765*(z-0.8753)/(z-1)/(z-0.409);
% Cvz = 5.2536*(z-0.9979)/(z-1)/(z-0.4152); % B Voltage WORKING!
% Cvz = 0.12233*(z-0.992)/(z-1)/(z-0.4053);

% Cvz = 0.3259*(z-0.992)/(z-1)/(z-0.4053); %Unstable
% Cvz = 1.9934*(z-0.992)/(z-1)/(z-0.4053); %Very Bad
% Cvs = 51108.0408*(s+204)/s/(s+1.542e004);
% Cvs = 3905.456*(s+204)/s/(s+1.542e004);
% Cvz = 20.5871*(z-0.998)/(z-1)/(z-0.3599);
% Cvz = 2.6832*(z-0.5833)*(z-0.9911)/(z-1)/(z-0.3688)/(z-0.2428);
% Cvz = 0.05919*(z-0.6093)/(z-1)/(z-0.3688);
Cvz = 2.7232*(z-0.992)/(z-1)/(z-0.4053); % Final Voltage Comp

Cvs = d2c(Cvz,'zoh');

%Bus voltage sense board gain + filter on DSP
Cfilt = 6.8e-9;
Rfilt = 1e3;

Gbus=12.78e-3*inv(s*Cfilt)/(inv(s*Cfilt)+Rfilt);
Gbusz=c2d(Gbus,Ts,'zoh')*z^-1;

Tv = Gvod*Cvs*Gbus; %Voltage Loop Gain
Tvz = Gvodz*Gbusz;

% Begin Scaling of the compensator

[Zs,Ps,Ks]=zpkdata(Cvs,'v'); %Get ZPK data from Cz

[Bs,As]=zp2tf(Zs,Ps,Ks);
[Ra,Pa,Ka]=residue(Bs,As); %Perform partial fraction expansion

s=tf('s');
CsS = 0;
for i = 1:length(Pa)-1

```

```

    CsS = CsS+Ra(i)/(s-Pa(i));           % Create poles/zeros TF
end

CsI = Ra(length(Pa))/(s-Pa(length(Pa))); %Create the integrator TF

CzS = c2d(CsS,Ts,'zoh');               %Convert poles/zeros and integrator
CzI = c2d(CsI,Ts,'zoh');               % to the digital domain

[ZS,PS,KS]=zpkdata(CzS,'v');           % Convert back to TF
coeff = zp2sos(ZS,PS,KS);              %Find the Direct II form

[BS,AS]=zp2tf(ZS,PS,KS);
BS=[0 1];                               % Set TF to the first save location
HS = impz(BS,AS);                       %Find the impulse response of the pole/zero part

Q=ceil(log2(sum(abs(HS))));              %Sum up the response, that is your scaling factor

cs=2^-Q;                                 % define Cs

L=ceil(log2(max(coeff(1:3)*inv(cs))));   %Find largest value of Bx and scale accordingly

disp(['Cs = 2^-',num2str(Q)])
disp(['L = 2^-',num2str(L)])

disp('Final Scaled Coefficients for poles/zeros difference equation')
disp('  b01  b11  b21  1  a11  a21')
coeffS=[coeff(1:3)./2^L*inv(cs) coeff(4:6)] %Final scaled coefficients

[ZI,PI,KI]=zpkdata(CzI,'v');
[BI,AI]=zp2tf(ZI,PI,KI);
disp('Integrator difference equation')
[coeffI] =tf2sos(BI,AI)                  % Calculate integrator coefficients

% Convert decimal numbers to signed integer hex values
disp(['B1 = ',num2str(dec2hex(round(abs(coeffS(1,2))*32767))), ...
      ' A1 = -',num2str(dec2hex(round(abs(coeffS(1,5))*32767)))])

disp(['Int gain = ',num2str(dec2hex(round(coeffI(1,2)*32767)))])

```

## B.2 Boost Current Loop Design

% Boost Current Feedback Loop Design

```
Vo=200;
Vg=55;
Co=4.4e-3;
L=57.3e-6;
P=1.5e3/2;
R=Vo^2/P;
Rc=15e-3;
Rl=0;
fs=10e3;
Ts=1/fs;
s = zpk('s');
```

%Divide by Two for single phase power

```
M=Vo/Vg;
K=2*L/(R*Ts);
```

```
A_D = 32736/32767; %Slight gain due to 10 bit A/D
```

syms D1

%Solve for DCM duty cycle

```
D=max(eval(solve((1+sqrt(1+4*D1^2/K))/2-Vo/Vg)));
```

```
D2=(K/D)*(Vo/Vg);
```

```
Ipk=Vg/L*D*Ts; %Peak Inductor Current
ILavg=Ipk/2*(D+D2); %Inductor Average Current
```

% DC Operating Point and Small Signal Parameters

```
Vac = Vg;
Io = Vo/R;
Ip = Io;
Vcp = Vo-Vg;
Ia = (M-1)*Io;
gi = Ia/Vac;
ki = 2*Ia/D;
ko = 2*Ip/D;
go = Ip/Vcp;
gf = 2*Ip/Vac;
Vap = Vo;
```

% Inductor Current to Duty-Cycle  $i_L/d$

```
Gild = (ki*R*s*Co+ki+ki*Rc*s*Co+ki*R*go*Rc*s*Co+ki*R*go+ko*R*s*Co+ko+ko*Rc*s*Co)/ ...
(1+gf*s*L+R*go+go*s*L+Rl*gi+Rc*s*Co+R*s*Co+s*L*gi+R*go*Rc*s*Co+R*go*Rl*gi+go*Rl+ ...
gf*Rl+go*Rl*R*s*Co+go*Rl*Rc*s*Co+go*s^2*L*R*Co+go*s^2*L*Rc*Co+Rl*gi*R*s*Co+ ...
Rl*gi*Rc*s*Co+s^2*L*gi*R*Co+s^2*L*gi*Rc*Co+gf*Rl*R*s*Co+gf*Rl*Rc*s*Co+ ...
gf*s^2*L*R*Co+gf*s^2*L*Rc*Co+R*go*Rl*gi*Rc*s*Co+R*go*s^2*L*gi*Rc*Co+R*go*s*L*gi);
```

```
z = zpk('z',Ts);
Gildz = c2d(Gild,Ts,'zoh');
```

% Compensator Design Iterations

```
% Ci = 0.0016814*(s+2.505e004)/s; %1kHz
% CI = 44.2659/s; %1kHz
```

```

% CI = 96.61*(s+42.34)/s/(s+1.222e004);           %1kHz
% CI = 121.1738*(s+33.47)/s/(s+1.222e004);       %1.74kHz
% Cis = 103.4017*(s+555.5)/s/(s+1.222e004);      %1.3kHz
% Ci = 0.00715338*(s+555.5)/s;                   % 1.5kHz
% Ci = 0.15441*(s+555.5)/s;                       % 1.1kHz w/ Vbatt gain
% Ci = 0.15715*(s+1609)/s;                         % 1.49kHz
% Ci = 0.16306*(s+33.95)/s;                       % A - Works at 888W
% Ci = 0.53615*(s+4509)/s;                       % 1.13kHz
% Ci = 0.20553*(s+4509)/s;                       % C Not stable w/ A Gv
% Ci = 0.067895*(s+7.019e004)/s;

% Ciz = 0.28547*(z-0.1246)/(z-1);
% Ciz = 0.16469*(z-0.9987)/(z-1)/(z-0.888);
% Ciz = 0.067541*(z-0.9987)/(z-1)/(z-0.9606);
% Ciz = 0.15139*(z-0.9918)/(z-1)/(z-0.9052);     %B WORKING!
% Ciz = 0.13849*(z-0.9987)/(z-1)/(z-0.8502);     % Unstable
% Ciz = 0.0779238*(z-0.9987)/(z-1)/(z-0.8894);
% Ciz = 0.011518/(z-1)/(z-0.613);                 %Very bad
% Ciz = 0.10529*(z-0.9925)/(z-1)/(z-0.9299);
% Ciz = 0.27896*(z-0.9945)/(z-1)/(z-0.9131);

Ciz = 0.086139*(z-0.9987)/(z-1)/(z-0.9254);      %Final Current Comp

Cis = d2c(Ciz,'zoh');                             % Conversion to analog

% Voltage sensor filter on VBattery A/D
Cfilter = 6.8e-9;
Rfilter = 1e3;

% Total Sensor Gain
Gbatt = 44.93e-3*(inv(Cfilter*s)/(Rfilter+inv(Cfilter*s)))*A_D;

Gbattz = c2d(Gbatt,Ts,'zoh')*z^-1;                % Digital Sensor Gain with delay
Tiz = Gildz*Ciz*Gbattz;                          %Digital Open Loop Response

% Begin Scaling of Coefficients

[Zs,Ps,Ks]=zpkdata(Cis,'v');                      %Get ZPK data from Cz
[Bs,As]=zp2tf(Zs,Ps,Ks);                          % Convert to transfer function
[Ra,Pa,Ka]=residue(Bs,As);                        %Perform partial fraction expansion

s=tf('s');
CsS = 0;
for i = 1:length(Pa)-1                             % Recombine poles/zeros
    CsS = CsS+Ra(i)/(s-Pa(i));
end

CsI = Ra(length(Pa))/(s-Pa(length(Pa)));           % Create the integrator

CzS = c2d(CsS,Ts,'zoh');                          %Convert poles/zeros to the digital domain
CzI = c2d(CsI,Ts,'zoh');                          %Convert integrator to digital

[ZS,PS,KS]=zpkdata(CzS,'v');                     % Get Z/P/K data from TF form
coeff = zp2sos(ZS,PS,KS);                         %Find the Direct II form

```

```

[BS,AS] = zp2tf(ZS,PS,KS);           % convert back to TF form
BS=[0 1];                           % find impulse response to first save location
HS = impz(BS,AS);                    %Find the impulse response of the pole/zero part

Q=ceil(log2(sum(abs(HS))))           % Sum up the response, that is your scaling factor

cs=2^-Q;                             % define cs

L=ceil(log2(max(coeff(1:3)*inv(cs)))); %Find largest value of Bx and scale accordingly

disp(['Cs = 2^-',num2str(Q)])
disp(['L = 2^-',num2str(L)])

disp('Final Scaled Coefficients for poles/zeros difference equation')
coeffS=[coeff(1:3)./2^L*inv(cs) coeff(4:6)] %Final scaled coefficients
disp('  b01  b11  b21  1  a11  a21')

[ZI,PI,KI]=zpkdata(CzI,'v');         % Display coefficients for integrator
[B1,A1]=zp2tf(ZI,PI,KI);
disp('Integrator difference equation')
coeffI=tf2sos(B1,A1)

% Convert decimal numbers to signed integer hex values

disp(['B1 = ',num2str(dec2hex(round(abs(coeffS(1,2))*32767))), ...
      ' A1 = ',num2str(dec2hex(round(abs(coeffS(1,5))*32767)))])

disp(['Int gain = ',num2str(dec2hex(round(coeffI(1,2)*32767)))])

```

## B.3 Boost Dual-Loop Controller Simulation

```

% Boost Dual-Loop Controller Simulation

% Converter Operating Parameters
Vo=200;
Vg=48;
Co=4.4e-3;
L=57.3e-6/2; % Divided by Two for Voltage Response
P=1.5e3;
R=Vo^2/P;
Rc=15e-3;
Rl=0;
fs=10e3;
Ts=1/fs;

s=zpk('s');
z=zpk('z',Ts);

flc = inv(sqrt(2*pi*L*Co)); % Expected ringing frequency
A_D = 32736/32767; % Slight gain due to 10 bit A/D
M=Vo/Vg; % Boost Conversion Ratio
K=2*L/(R*Ts);

syms D1
D=max(eval(solve((1+sqrt(1+4*D1^2/K))/2-Vo/Vg))); %Solve for DCM duty cycle
D2=(K/D)*(Vo/Vg);

% Small Sigal Parameters from Dr. Lee's notes
Vac = Vg;
Io = Vo/R;
Ip = Io;
Vcp = Vo-Vg;
Ia = (M-1)*Io;
gi = Ia/Vac;
ki = 2*Ia/D;
ko = 2*Ip/D;
go = Ip/Vcp;
gf = 2*Ip/Vac;
Vap = Vo;

% Control-to-Output Voltage
Gvod = -(ki*go*Rl+ki*go*s*L-ko*Rl*gi-ko*s*L*gi-ko+gf*ki*Rl+gf*ki*s*L)/ ...
(1+gf*Rl+go*Rl+R*s*Co+Rc*s*Co+s*L*gi+Rl*gi+Rl*gi*R*s*Co+Rl*gi*Rc*s*Co+ ...
s^2*L*gi*R*Co+s^2*L*gi*Rc*Co+go*s*L+gf*s*L+R*go+R*go*Rl*gi+gf*Rl*R*s*Co+ ...
gf*Rl*Rc*s*Co+gf*s^2*L*R*Co+gf*s^2*L*Rc*Co+go*Rl*R*s*Co+go*Rl*Rc*s*Co+ ...
go*s^2*L*R*Co+go*s^2*L*Rc*Co+R*go*s*L*gi+R*go*Rl*gi*Rc*s*Co+ ...
R*go*s^2*L*gi*Rc*Co+R*go*Rc*s*Co)*R*(1+Rc*s*Co);

Gvodz = c2d(Gvod,Ts,'zoh'); % Analog to Digital Conversion of Gvod
Cvz = 2.7232*(z-0.992)/(z-1)/(z-0.4053); % Voltage Compensator
Cvs = d2c(Cvz,'zoh'); % Conversion to Analog Domain for Scaling

% Redefine Small-Signal Parameters for One-Phase Current Model
L=57.3e-6;

```

```

PI=P/2;
R=Vo^2/PI;
Vac = Vg;
Io = Vo/R;
Ip = Io;
Vcp = Vo-Vg;
Ia = (M-1)*Io;
gi = Ia/Vac;
ki = 2*Ia/D;
ko = 2*Ip/D;
go = Ip/Vcp;
gf = 2*Ip/Vac;
Vap = Vo;

% Control-to-Inductor Current Transfer Function
Gild = (ki*R*s*Co+ki+ki*Rc*s*Co+ki*R*go*Rc*s*Co+ki*R*go+ko*R*s*Co+ko+ko*Rc*s*Co)/ ...
(1+gf*s*L+R*go+go*s*L+Rl*gi+Rc*s*Co+R*s*Co+s*L*gi+R*go*Rc*s*Co+R*go*Rl*gi+go*Rl+ ...
gf*Rl+go*Rl*R*s*Co+go*Rl*Rc*s*Co+go*s^2*L*R*Co+go*s^2*L*Rc*Co+Rl*gi*R*s*Co+ ...
Rl*gi*Rc*s*Co+s^2*L*gi*R*Co+s^2*L*gi*Rc*Co+gf*Rl*R*s*Co+gf*Rl*Rc*s*Co+ ...
gf*s^2*L*R*Co+gf*s^2*L*Rc*Co+R*go*Rl*gi*Rc*s*Co+R*go*s^2*L*gi*Rc*Co+R*go*s*L*gi);

Gildz = c2d(Gild,Ts,'zoh'); % Convert from Analog to Digital Domain
Ciz = 0.086139*(z-0.9987)/(z-1)/(z-0.9254); % Current Compensator
CI = d2c(Ciz,'zoh'); % Convert to the digital domain

% Voltage sensor filter on VBattery A/D
Cfilter = 6.8e-9;
Rfilter = 1e3;

Gbatt = 44.93e-3/(Rfilter*s*Cfilter+1)*A_D; % Total Battery Sensor Gain
Gbatz = c2d(Gbatt,Ts,'zoh')*z^-1; % Digital Sensor Gain with delay

Ti = Gild*CI*Gbatt; % Analog Current Loop
Tiz = Gildz*Ciz*Gbatz; % Digital Current Loop

% Bus Voltage Sensor Filter
Cfilt = 6.8e-9;
Rfilt = 1e3;

Gbus=12.78e-3*inv(s*Cfilt)/(inv(s*Cfilt)+Rfilt)*A_D;
Gbusz=c2d(Gbus,Ts,'zoh')*z^-1; % Model one-cycle delay

Tv = Gvod*Cvs*Gbus; % Analog Voltage Loop
Tvz = Gvodz*Cvz*Gbusz; % Digital Voltage Loop

T1z = Tvz + Tiz; % T1 Loop Gain in Digital Domain
T2z = Tvz/(1+Tiz); % T2, Overall Loop Gain in Digital Domain
T2 = Tv/(1+Ti); % T2 Loop Gain in Analog Domain

figure(3)
clf
bode(T2z,{10^-1,10^5})
title('Overall Loop Response - T2')

```

## Appendix C - Buck Matlab Code

### C.1 Buck Current Loop Design

```
% Buck Current Mode Control
Vo = 55;
Vg = 200;
Rc = 15e-3;
Rl = 0;
Co = 4700e-6;
fs = 10e3;
Ts = 1/fs;
P = 1000/2;
R = Vo^2/P;
Io = Vo/R;
M = Vo/Vg;
L = 57.3e-6;

A_D = 32736/32767; % Slight Gain of 10 bit A/D

K=2*L/(R*Ts);

s=tf('s');
z = zpk('z',Ts);

syms D1
D=max(eval(solve(2/(1+sqrt(1+4*K/D1^2))-M,D1)))
D2=K/D*M;

%Control-to-Inductor Current, iL/d, F4
kd = 2*Io/D;
r = R*(1-M);
Zt = R*(Rc+inv(s*Co))/(R+Rc+inv(s*Co))+Rl+s*L;
Gild = kd*(r/(r+Zt));

Gildz = c2d(Gild,Ts,'zoh'); % Conversion to Digital Domain

% Iterations of Current Compensator

% Ciz = 0.14408*(z-0.6529)/(z-1);
% Ciz = 0.1658*(z-0.3135)/(z-1);
% Ciz = 0.014743*(z-0.3135)/(z-1);
% Ciz = 0.161068*(z-0.9971)/(z-1)/(z-0.6093); % 744 Hz
% Ciz = 0.11444*(z-0.9971)/(z-1)/(z-0.83); % 594 Hz OK
% Ciz = 0.085978*(z-0.9988)/(z-1)/(z-0.9783);
% Ciz = 0.0064668/(z-1)/(z-0.6712);
% Ciz = 0.23236*(z-0.9971)/(z-1)/(z-0.9064);
% Ciz = 0.1044*(z-0.9971)/(z-1)/(z-0.9529);
% Ciz = 0.42967*(z-0.9971)/(z-1)/(z-0.7393);
% Ciz = 0.30503*(z-0.9988)/(z-1)/(z-0.8917);
% Ciz = 0.43574*(z-0.9969)/(z-1)/(z-0.6614);
% Ciz = 0.30083*(z-0.9931)/(z-1)/(z-0.9338);
% Ciz = 0.29181*(z-0.9951)/(z-1)/(z-0.9338);
% Ciz = 0.325778*(z-0.9982)/(z-1)/(z-0.8932);
% Ciz = 0.279*(z-0.9969)/(z-1)/(z-0.9175);
% Ciz = 0.42884*(z-0.9915)/(z-1)/(z-0.7366);
```

```

% Ciz = 0.71063*(z-0.997)/(z-1)/(z-0.4131);
% Ciz = 0.37677*(z-0.9853)/(z-1)/(z-0.8318); % Good
% Ciz = 0.37598*(z-0.997)/(z-1)/(z-0.8318); % Looks good at 340W
% Ciz = 0.249668*(z-0.9925)/(z-1)/(z-0.8318); % stable at low power 200W
% Ciz = 0.21341*(z-0.9967)/(z-1)/(z-0.8997);
Ciz = 0.188488*(z-0.9918)/(z-1)/(z-0.8426); % Final Compensator

Cis = d2c(Ciz,'zoh'); % Conversion to Analog Domain

%Bus voltage sense board gain + filter on DSP
Cfilt = 6.8e-9;
Rfilt = 1e3;

Gbus=12.78e-3*inv(s*Cfilt)/(inv(s*Cfilt)+Rfilt)*A_D; % Bus Sensor Gain
Gbusz=c2d(Gbus,Ts,'zoh')*z^-1; % Gain w/ one-cycle delay

Ti = Gild*Cis*Gbus; %Analog Open-Loop gain
Tiz = Gildz*Ciz*Gbusz; %Digital Open-loop gain

[Zs,Ps,Ks]=zpkdata(Cis,'v'); % Get ZPK data from Cs
[Bs,As]=zp2tf(Zs,Ps,Ks); % Convert ZPK to TF
[Ra,Pa,Ka]=residue(Bs,As); % Perform partial fraction expansion

CsS = Ra(1)/(s-Pa(1)); %Create the pole transfer function
CsI = Ra(2)/(s-Pa(2)); %Create the integrator TF

CzS = c2d(CsS,Ts,'zoh'); %Convert both to the digital domain
CzI = c2d(CsI,Ts,'zoh');

[ZS,PS,KS]=zpkdata(CzS,'v');
coeff=zp2sos(ZS,PS,KS); % Find the Direct II form

[BS,AS]=zp2tf(ZS,PS,KS);
BS=[0 1]; % set to first save location
HS = impz(BS,AS); %Find the impulse response of the pole/zero part

Q=ceil(log2(sum(abs(HS)))); %Sum up the response, that is your scaling factor
cs=2^-Q;

L=ceil(log2(max(coeff(1:3)*inv(cs)))); %Find largest value of Bx and scale accordingly

disp(['Cs = 2^-',num2str(Q)]) % output scaling factors
disp(['L = 2^',num2str(L)])

disp('Final Scaled Coefficients for poles/zeros difference equation')
coeffS=[coeff(1:3)./2^L*inv(cs) coeff(4:6)] %Final scaled coefficients

[ZI,PI,KI]=zpkdata(CzI,'v');

[Bi,AI]=zp2tf(ZI,PI,KI);
disp('Integrator difference equation')
coeffI=tf2sos(Bi,AI) % convert integrator to SOS

% Convert to HEX for DSP
disp(['B1 = 0x',num2str(dec2hex(round(abs(coeffS(1,2))*32767))), ...

```

```
' A1 = -0x',num2str(dec2hex(round(abs(coeffS(1,5))*32767)))])  
disp(['Int gain = ',num2str(dec2hex(round(coeffI(1,2)*32767)))])
```

## C.2 Buck Voltage Loop Design

```

%Buck Voltage Loop Design
clear all
Vo = 58.8;
Vg = 200;
Rc = 15e-3;
Rl = 0;
Co = 4700e-6;
fs = 10e3;
Ts = 1/fs;
P = 400;
R = Vo^2/P;
Io = Vo/R;
Iin = P/Vg;
M = Vo/Vg;
L = 57.3e-6/2;
A_D = 32736/32767;

s=tf('s');
z = zpk('z',Ts);

K=2*L*fs/R;
M=Vo/Vg;

syms D1
%Ideal Duty Cycle
D=max(eval(solve(2/(1+sqrt(1+4*K/D1^2))-M,D1)));
D2=K/D*M;

% Small-Signal Parameters
u = D^2/K/M;
Ia = Iin;
Vac = Vo/u;
Vcp = Vo;
Ip = Io-Ia;
gi = Ia/Vac;
ki = 2*Ia/D;
ko = 2*Ip/D;
go = Ip/Vcp;
gf = 2*Ip/Vac;

%Control-to-Output, Vo/d, F2
Gvod = inv((1+gi*R+s*L*gi+s*L*go+s*L*gf+Rc*s*Co+R*s*Co+R*go+R*gf+Rc*s*Co*Rl*gf+ ...
    Rc*s^2*Co*L*gi+Rc*s^2*Co*L*go+Rc*s*Co*Rl*go+Rc*s^2*Co*L*gf+Rc*s*Co*gi*R+ ...
    Rc*s*Co*R*go+Rc*s*Co*R*gf+Rc*s*Co*Rl*gi+R*s*Co*Rl*gi+R*s*Co*Rl*go+R*s*Co*Rl*gf+ ...
    R*s^2*Co*L*gi+R*s^2*Co*L*go+R*s^2*Co*L*gf+Rl*go+Rl*gf+Rl*gi)/R/(ki+ko+Rc*s*Co*ki+
    Rc*s*Co*ko));

Gvodz = c2d(Gvod,Ts,'zoh');

% Voltage Compensator Design Iterations

% Ciz = 0.038427*(z-0.9971)/(z-1)/(z-0.9737);

```

```

% Ciz = 0.12408*(z-0.9969)/(z-1)/(z-0.7865);
% Ciz = 0.0510228*(z-0.997)/(z-1)/(z-0.9777);
% Ciz = 0.19948*(z-0.9971)/(z-1)/(z-0.4438);
% Ciz = 0.161068*(z-0.9971)/(z-1)/(z-0.6093);           % OK
% Ciz = 0.11444*(z-0.9971)/(z-1)/(z-0.83);             % OK
% Cz = 69.4786*(z-0.9955)/(z-1)/(z-0.2498);
% Cz = 22.8856*(z-0.9955)/(z-1)/(z-0.2498);
% Cz = 57.5089*(z-0.9925)/(z-1)/(z-0.3841);
% Cz = 10.725*(z-0.9925)/(z-1)/(z-0.3841);
% Cz = 2.4038*(z-0.9928)/(z-1)/(z-0.3876);
% Cz = 1.3932*(z-0.9926)/(z-1)/(z-0.3876);
% Cz = 15.2749*(z-0.9926)/(z-1);
% Cz = 0.50705*(z-0.9928)/(z-1)/(z-0.3844);           %B
% Cz = 0.020774/(z-1)/(z-0.3844);
% Cz = 2.4842*(z-0.9941)/(z-1)/(z-0.395);
% Cz = 0.64039*(z-0.994)/(z-1)/(z-0.395);
% Cz = 0.25873*(z-0.994)/(z-1)/(z-0.395);
% Cz = 0.14399*(z-0.9993)/(z-1)/(z-0.395);
% Cz = 0.71808*(z-0.9993)/(z-1);
% Cz = 0.17306*(z-0.9853)/(z-1)/(z-0.3685);
% Cz = 0.70714*(z-0.9853)/(z-1)/(z-0.3685);
Cz = 0.36028*(z-0.9942)/(z-1)/(z-0.3685);           % Final Voltage Compensator

Cs = d2c(Cz,'zoh');                                   % Conversion to analog domain

% RC Filter values
Cfilt = 6.8e-9;
Rfilt = 1e3;

Gbatt=44.93e-3*inv(s*Cfilt)/(inv(s*Cfilt)+Rfilt)*A_D;   %Voltage sense gain
Gbatzz=c2d(Gbatt,Ts,'zoh')*z^-1;                       % sensor w/one-cycle delay

Tv = Gvod*Cs*Gbatt;                                   %Voltage Open Loop Gain
Tvz = Gvodz*Cz*Gbatzz;                                % Digital Loop gain

% Begin Scaling of Compensator

[Zs,Ps,Ks]=zpkdata(Cs,'v');                             %Get ZPK data from Cz
[Bs,As]=zp2tf(Zs,Ps,Ks);
[Ra,Pa,Ka]=residue(Bs,As);                             %Perform partial fraction expansion

CsS = Ra(1)/(s-Pa(1));                                 %Create the pole transfer function
CsI = Ra(2)/(s-Pa(2));                                 %Create the integrator TF

CzS = c2d(CsS,Ts,'zoh');                               %Convert to the digital domain
CzI = c2d(CsI,Ts,'zoh');

[ZS,PS,KS]=zpkdata(CzS,'v');
coeff=zp2sos(ZS,PS,KS)                                %Find the Direct II form

[BS,AS]=zp2tf(ZS,PS,KS);
BS=[0 1];
HS = impz(BS,AS);                                     % set TF to first save location
%Find the impulse response of the pole/zero part

```

```

Q=ceil(log2(sum(abs(HS))));           %Sum up the response, that is your scaling factor
cs=2^-Q;
L=ceil(log2(max(coeff(1:3)*inv(cs)))); %Find largest value of Bx and scale accordingly

disp(['Cs = 2^-',num2str(Q)])        % output scaling terms
disp(['L = 2^-',num2str(L)])

disp('Final Scaled Coefficients for poles/zeros difference equation')
coeffS=[coeff(1:3)./2^L*inv(cs) coeff(4:6)] %Final scaled coefficients

[ZI,PI,KI]=zpkdata(CzI,'v');

[B1,A1]=zp2tf(ZI,PI,KI);
disp('Integrator difference equation')
coeffI=tf2sos(B1,A1)                 % convert integrator to SOS

disp(['B1 = ',num2str(dec2hex(round(abs(coeffS(1,2))*32767))), ...
      ' A1 = ',num2str(dec2hex(round(abs(coeffS(1,5))*32767)))])
disp(['Int gain = ',num2str(dec2hex(round(coeffI(1,2)*32767)))])

```

## Appendix D - Boost Mode C Code

### D.1 Main.c

```
#include "f2407_c.h"
#include "pwm.h"
#include "adc.h"
#include "io.h"
#include "std.h"
#include "Qmath.h"
#include "capture.h"
#include "VConst.h"

#define MAX_DC          0x6CCC          /* Set to 0.85          */
#define PWM_scale      0x1893          /* 0.192              */
#define G               0x6FBB          /* Gain value of Ts/(2*L) */

/* EVERYTHING DECLARED HERE, BEFORE MAIN, IS GLOBAL */

/* declare the parts of DSP used */
PWMGEN PWM = PWM_DEFAULTS;
ADC_SAMPLER ADC = ADC_DEFAULTS;
CAPTURE cap;

/* bit definitions */
bit_io LED;
bit_io up;
bit_io down;
bit_io toggle;
bit_io reset;
bit_io BBselect;
bit_io in_out;

int GO,count,OV_count;

unsigned int yA = 0x0A00;
signed int output_PWM1 = 0;
signed int output_PWM2 = 0;
signed int output_PWM3 = 0;
signed int output_PWM4 = 0;
signed int temp[15];
int i;
int count =0;

void main()
{
    for(i=0; i < 15; i++) temp[i] = 0;
    i = 0;

    GO=0;
    OV_count=0;

    init(); /* Initialize Boost Mode */

    PWM.mfunc_c1 = 0; /* Set PWM output to zero*/
    PWM.mfunc_c2 = 0xFFFF;
```

```

updatePWM(&PWM);
set(EVAIFRA, BIT1); /* clear GPT1 interrupt */

bit_off(reset); /* Lower reset line to reset gate drives */
for (i=0; i<=50;i++) /* wait for a while */
    {}
bit_on(reset); /* set reset line high, ready to go */
bit_off(BBselect); /* Select boost mode optocoupler */

while (1) /* Main loop, all interrupt driven */
{
}
}

void interrupt TenkHz() /* 10 kHz interrupt */
{
    static unsigned int y, Vlow, Vhigh;
    static unsigned int Soft_Start = 0;
    int count2 = 0;
    static int LCDcount = 0;
    char LCD_Buffer[4][4];

    unsigned long int Ans, Atemp, Btemp;

    signed int VBatt, VBus, D1, D2, Base, Iavg, Save, VL;
    signed int Iavg1, Iavg2, Iavg3, Ierror1;

    static int Ic[5] = {0x037E, 0x0A79, 0x1174, 0x22E8, 0x3D18};
    static signed int Vce[6] = {0x0138, 0x014E, 0x01BE, 0x01DD, 0x022D, 0x028B};

    static signed int Verror, Vn, Vcomp, Vref;
    static signed int Vc = 0x630C;
    static signed int Ierror, In, Icomp, Iref, yn;

    asm ( " SETC OVM "); /* set overflow mode to ON */
    asm ( " clrc INTM "); /* Re Enable all interrupts for nesting*/
    /* of the capture interrupt */

    set(EVAIFRC, CAP1INT); /* clear capture 1 interrupt to allow for */
    /* first rising edge interrupt */

    bit_on(in_out);

    /* Voltage Reference Control and OL/CL select */

    if(!bit_test(toggle))
        Soft_Start=1;

    if(!bit_test(up))
        if((count++)%20 == 0)
        {
            if (!GO)
                yA++;
            else
                Vc++;
        }
}

```

```

if(!bit_test(down))
    if((count++)%20 == 0)
        if (!GO)
            yA--;
        else
            Vc--;

/* Get Current A/D value */

Vlow = *RESULT0;
Vhigh = *RESULT1;

set(ADCTRL2,ADC_SOC_SEQ1); /* This should start a new conversion */
set(ADCTRL2,ADC_SOC_SEQ1); /* but three times makes it better */
set(ADCTRL2,ADC_SOC_SEQ1);

VBatt = Vlow >> 1; /* Shift A/D unsigned result into signed int */
VBus = Vhigh >> 1; /* result will always be positive */

/* Compute Correct Voltage across the inductor */

while (Iavg >= Ic[count2] & count2<6) /* Set through Ic until Iavg is greater*/
    count2++;

VL = VBatt - Vce[count2]; /* subtract off the corresponding value of VCE*/

/* COMPUTE AVERAGE CURRENT */

if (Soft_Start & !GO) /* for OL, use current value of PWM for D1 */
    D1 = yA >> 1;
else
    D1 = yn; /* for CL use last value of DC for D1 */

Atemp = (long) D1 <<16; /* Add the current rise and fall times to get */
Btemp = (long) cap.D2 <<16; /* the base of the triangle */
Ans = Atemp + Btemp; /* this will limit to +1 at high power */
Base = Ans>>16;

Iavg1 = ((long) Base * (long) D1) >>15; /* Compute the average current value*/
Iavg2 = ((long) Iavg1 * (long) G) >>15; /* Iavg = Ts(.5)/L*(D1+D2)*D1*VL */
Iavg3 = ((long) Iavg2 * (long) VL) >>15;

Iavg = Iavg3; /* For some reason the DSP will not function properly*/
/* if the value of current is sent to the PWM D/A outputs*/

/* Implement the Dual-Loop Controller */

Atemp = (long) Vc <<16;
Btemp = (long) VBus <<16;
Ans = Atemp - Btemp;
Verror = Ans>>16; /* Verror = Vc - VBus; */

output_PWM3 = Verror;

Vcomp = V_comp(Verror, Iavg); /* Call Voltage compensator procedure */

```

```

/* Pass voltage error and Iavg */
/* for OL seeding of the integrator*/

Atemp = (long) Vcomp <<16;
Btemp = (long) Iavg <<16;
Ans = Atemp - Btemp; /* Ierror = Vcomp - Iavg; */
Ierror1 = Ans>>16;

Ierror = Ierror1; /* As before, anything with the current cannot be used*/

output_PWM4 = Ierror1;

Icomp = I_comp(Ierror); /* Send current error to current compensator procedure*/

yn = Icomp; /* send current compensator output to PWM output*/

output_PWM1 = Iavg3;
output_PWM2 = yn;

if (yn>=MAX_DC) /* Limit maximum duty cycle */
    yn=MAX_DC;

if (yn&(0x8000)) /* If duty-cycles is negative, set to zero */
    yn=0x0000;

y = yn << 1; /* Convert yn from to unsigned integer */

/* Limit Max output voltage and turn off PWM */
if (VBus >= 0x6A93 | OV_count>=4) /* If Vbus voltage is at or above 215V for more than*/
    if ((OV_count++)>= 4) /* four cycles, turn everything off */
    {
        OV_count = 4;
        GO=0;
        yA = 0;
        y = 0;
    }

if (!GO) /* GO button to turn on CL compensator PWM outputs */
    y = 0;

if (Soft_Start & !GO) /* soft-start set for 200V into 54 ohms */
    if (VBus <= Vc)
        yA++;
    else
        GO=1;

if (!Soft_Start)
    y = 0;

if (GO)
{
    PWM.mfunc_c1 = y;
    PWM.mfunc_c2 = 0xFFFF - y;
}

```

```

else
    }
    {
        PWM.mfunc_c1 = yA;
        PWM.mfunc_c2 = 0xFFFF - yA;
    }

/* PWM D/A Output scaling -1 to 1 to 0 to 1
if (output_PWM1 <=0x7FFF & output_PWM1 >= 0)
    PWM.mfunc_c3 = output_PWM1 + 0x7FFF;
else
    PWM.mfunc_c3 = output_PWM1 - 0x8000;

if (output_PWM2 <=0x7FFF & output_PWM2 >= 0)
    PWM.mfunc_c4 = output_PWM2 + 0x7FFF;
else
    PWM.mfunc_c4 = output_PWM2 - 0x8000;

if (output_PWM3 <=0x7FFF & output_PWM3 >= 0)
    PWM.mfunc_c5 = output_PWM3 + 0x7FFF;
else
    PWM.mfunc_c5 = output_PWM3 - 0x8000;

if (output_PWM4 <=0x7FFF & output_PWM4 >= 0)
    PWM.mfunc_c6 = output_PWM4 + 0x7FFF;
else
    PWM.mfunc_c6 = output_PWM4 - 0x8000;

updatePWM(&PWM);          /* Update PWM with current PWM values */

bit_off(in_out);
}

void interrupt capture()
{
    static int state = 1; /* waiting on fall = 2, waiting on rise = 1 */
    signed int D2, Save;
    long int Atemp, Btemp, Ans;

    if(state == 1)          /* waiting on rise */
    {
        *T2CNT = 0;          /* zero timer to begin conversion of diode off-time */

        clear(CAPCONA, CAP1_EDGE_RISE);
        /* clear rising edge detection and set capture to */
        set(CAPCONA, CAP1_EDGE_FALL); /* detect a falling edge */
        state = 2;
        bit_on(LED);
        set(EVAIFRC, CAP1INT); /* clear capture 1 interrupt to allow for next capture event */
    }
    else                    /* waiting on fall */
    {
        D2 = *T2CNT;          /* Save timer value which is now D2 in clock cycles*/

```

```

clear(CAPCONA, CAP1_EDGE_FALL);    /* clear falling edge detection and set*/
set(CAPCONA, CAP1_EDGE_RISE);      /* rising edge detection for next cycle/
state = 1;

bit_off(LED);

/* Convert clock cycles to signed percentage of period      */

cap.D2 = D2 << 3;                    /* multiply clock cycles by 8*/

/* and then multiply clock cycles by 0.192                  */
Save = ((long) D2 * PWM_scale) >> 15;

cap.D2 = cap.D2 + Save;              /* add both numbers together to get D2*8.192*/

set(EVAIFRA, BIT1);                 /* clear GPT1 interrupt to allow for next switching cycle*/

set(CAPFIFOA,CAP1_FIFO_ONE);        /* tell FIFO that there is one spot open */
/* so next capture won't be lost      */

    }
}

void interrupt fault()
{
    bit_on(LED);                      /* turn off the led */

    /* stop here indefinitely */
}

```

## D.2 VIConst.h

```
#define A0V          0x7FFF
#define B0V          0x0000

#define A0I          0x7FFF
#define B0I          0x0000

/*      Boost Mode Voltage and Current Compensator Constants

/*      Voltage Compensator a GOOD ONE!          */
#define A1V          -0x33E0          /*-0.2139      */
#define B1V          0x55F8          /* 0.6427      */

#define B0iV          0x04B0          /* 0.0676      */

// cs = 2^-1      Scaling Terms
// L = 2^3

/* Poles and Zeros Coefficients of Current Compensator          */
#define A1I          -0x7673          /*-0.7284      */
#define B1I          0x56AB          /*0.6639*/

#define B0iI          0x0036          /* 0.0257      */

//cs = 2^-4
//L = 2^1
```

### D.3 VIComp.c

```
#include "f2407_c.h"
#include "VICConst.h"
extern int GO,yA;
extern signed int output_PWM1;
extern signed int output_PWM2;
extern signed int output_PWM3;
extern signed int output_PWM4;

signed int I_comp (signed int In)
{
    signed int xn, yn, h;
    signed int vS, yS, yI;
    static signed int vS_1 = 0;          /* Initialize save variables */
    static signed int yI_1 = 0;

    unsigned long int Atemp, Btemp, Ans, R0;

    asm ( " SETC OVM ");          /* Set OVM again */

    /* Perform input Ierror scaling */

    xn = In >> 4;                /* Divide by 16 for cs scaling */

    /* Implement the Poles and Zeros of the Compensator */

    h = ( (long) A1I * (long) vS_1 ) >>15; /* A1*vS(n-1) */

    Atemp = (long) xn <<16;
    Btemp = (long) h <<16;
    Ans = Atemp - Btemp;
    vS = Ans >>16;                /* vS = xn - A1*vS(n-1); */

    /* Output of the poles/zeros of the compensator */

    yS = ( (long) B1I * (long) vS_1 ) >>15; /* B1*vS(n-1) */

    /* Mutlply output by 2^L but check first for possible overflow */

    if (!(yS & (0x8000)))
    /* Check for positive... */
    {
        if (((unsigned int)yS <= 0x3FFF) & ((unsigned int)yS >= 0))
            yS = yS <<1;          /* if less than 0.4999 then */
        else                      /* multiply by 2 */
            yS = 0x7FFF;          /* if too large then limit to +1 */
    }
    else                          /* if negative... */
        if (((unsigned int)yS >= 0xC000) & ((unsigned int)yS <= 0xFFFF))
```

```

        yS = yS<<1;          /* if less than -0.5 multiply by 2 */
    else
        yS = 0x8000;        /* if too large set to -1 */

    vS_1=vS;                /* Save last value of vS */

/* Implementation of the Integrator */

h = ( (long) B0iI * (long) In ) >>15; /* use Ierror without scaling */

Atemp = (long) h<<16;      /* OVM, the integrator will limit */
Btemp = (long) yI_1 <<16; /* at -1 or +1 */
Ans = Atemp + Btemp;
yI = Ans>>16;             /* yI = h + yI_1 */

if (!GO)                  /* if OL, then use current duty-cycle */
    yI = yA >> 1;

yI_1 = yI;                /* Save last value of yI */

/* Add the output of the poles/zeros and integrator */

Atemp = (long) yI <<16;
Btemp = (long) yS <<16;
Ans = Atemp + Btemp;      /* yn = yS + yI */
yn = Ans>>16;            /* OVM will limit the sum to +1 */

return(yn);               /* return new value of duty-cycle */
}

signed int V_comp (signed int Vn, signed int Iavg)
{
    signed int xn, yn, h;
    signed int vS, yS, yI;
    static signed int vS_1 = 0; /* Initialize save variables */
    static signed int yI_1 = 0;

    unsigned long int Atemp, Btemp, Ans, R0;

    asm ( " SETC OVM ");    /* Set OVM mode */

    xn = Vn >> 1;         /* Perform input scaling of Voltage error */
                        /* Divide by two for cs scaling */

/* Implement the Poles and Zeros of the Compensator */

h = ( (long) A1V * (long) vS_1 ) >>15; /* A1*vS(n-1) */

Atemp = (long) xn <<16;
Btemp = (long) h <<16;
Ans = Atemp - Btemp;
vS = Ans>>16;           /* vS = xn - A1*vS(n-1); */

/* Output of the poles/zeros of the compensator */

```

```

yS = ( (long) B1V * (long) vS_1) >>15;          /* B1*vS(n-1) */

/* Multiply output by 2^L but first check for overflow */

if (!(yS&(0x8000)))          /* if ys is positive... */
{
    if (((unsigned int)yS<=0x0FFF) & ((unsigned int)yS>=0))
        yS = yS<<3;          /* if yS <= 0.12497 */
    else
        yS = 0x7FFF;          /* multiply by 8 */
                                /* else set to +1 */
}
else          /* if yS negative...*/
    if (((unsigned int)yS>=0xF000) & ((unsigned int)yS<=0xFFFF))
        yS = yS<<3;          /* if yS<=-.12497*/
    else
        yS = 0x8000;          /*multiply by 8 */
                                /* if too large set to -1*/

vS_1=vS;          /* Save last value of vS */

/* Implementation of the Integrator */
h = ( (long) B0iV * (long) Vn) >>15;          /* use unscaled value of Verror*/

Atemp = (long) h<<16;          /* OVM mode will limit sum to -1 to +1 */
Btemp = (long) yI_1 <<16;
Ans = Atemp + Btemp;
yI = Ans>>16;          /* yI = h + yI_1 */

if (!GO)          /* if OL, set integrator to avg. current value */
    yI = Iavg;

yI_1 = yI;          /* Save last value of yI */

/* Add the output of the poles/zeros and integrator */

Atemp = (long) yI <<16;
Btemp = (long) yS <<16;
Ans = Atemp + Btemp;          /* yn = yS + yI */
yn = Ans>>16;          /* sum will saturate to +1 */

return(yn);          /* return new value of duty-cycle */
}

```

## D.4 Capture.h

```
/* constants for registers */

/*-----*/
F2407 Control Register TxCON, shared with PWM
/*-----*/

#define FREE_RUN_FLAG      0x8000
#define SOFT_STOP_FLAG    0x4000

#define TIMER_STOP        0x0000
#define TIMER_CONT_UPDN   0x0800
#define TIMER_CONT_UP     0x1000
#define TIMER_DIR_UPDN    0x1800

#define TIMER_CLK_PRESCALE_X_1  0x0000
#define TIMER_CLK_PRESCALE_X_2  0x0100
#define TIMER_CLK_PRESCALE_X_4  0x0200
#define TIMER_CLK_PRESCALE_X_8  0x0300
#define TIMER_CLK_PRESCALE_X_16 0x0400
#define TIMER_CLK_PRESCALE_X_32 0x0500
#define TIMER_CLK_PRESCALE_X_64 0x0600
#define TIMER_CLK_PRESCALE_X_128 0x0700

#define TIMER_ENABLE_BY_OWN  0x0000
#define TIMER_ENABLE_BY_T1   0x0080

#define TIMER_ENABLE         0x0040
#define TIMER_DISABLE        0x0000

#define TIMER_CLOCK_SRC_INTERNAL  0x0000
#define TIMER_CLOCK_SRC_EXTERNAL  0x0010
#define TIMER_CLOCK_SRC_QEP       0x0030

#define TIMER_COMPARE_LD_ON_ZERO  0x0000
#define TIMER_COMPARE_LD_ON_ZERO_OR_PRD 0x0004
#define TIMER_COMPARE_LD_IMMEDIATE 0x0008

#define TIMER_ENABLE_COMPARE  0x0002
#define TIMER_SELECT_T1_PERIOD 0x0001

/*-----*/
F2407 Control Register CAPCONx
/*-----*/

#define CAPTURE_RESET      0x4000

#define DIS_CAP_1AND2      0x0000
#define EN_CAP_1AND2       0x2000

#define DIS_CAP3           0x0000
#define EN_CAP3            0x1000

#define CAP3_GPT2         0x0000
#define CAP3_GPT1         0x0400
```

```

#define CAP12_GPT2          0x0000
#define CAP12_GPT1          0x0200

#define CAP3_START_ADC      0x0100

#define CAP1_EDGE_NO        0x0000
#define CAP1_EDGE_RISE      0x0040
#define CAP1_EDGE_FALL      0x0080
#define CAP1_EDGE_BOTH      0x00C0
#define CAP2_EDGE_NO        0x0000
#define CAP2_EDGE_RISE      0x0020
#define CAP2_EDGE_FALL      0x0010
#define CAP2_EDGE_BOTH      0x0030

#define CAP3_EDGE_NO        0x0000
#define CAP3_EDGE_RISE      0x0008
#define CAP3_EDGE_FALL      0x0004
#define CAP3_EDGE_BOTH      0x000C

/*-----
F2407 Status Register CAPFIFOx
-----*/
#define CAP3_FIFO_EMPTY      0x0000
#define CAP3_FIFO_ONE        0x1000
#define CAP3_FIFO_TWO        0x2000
#define CAP3_FIFO_TWO_LOST   0x3000

#define CAP2_FIFO_EMPTY      0x0000
#define CAP2_FIFO_ONE        0x0400
#define CAP2_FIFO_TWO        0x0800
#define CAP2_FIFO_TWO_LOST   0x0C00

#define CAP1_FIFO_EMPTY      0x0000
#define CAP1_FIFO_ONE        0x0100
#define CAP1_FIFO_TWO        0x0200
#define CAP1_FIFO_TWO_LOST   0x0300

/* EVxIMRC constants */

#define CAP3INT              0x4
#define CAP2INT              0x2
#define CAP1INT              0x1

/*-----
** EVA
-----
/*-----
Initialization constant for the F2407 TxCON register for PWM Generation.
Sets up the timer frequency and modes.
-----*/
#define T2CON_INIT_STATE_A ( SOFT_STOP_FLAG + \
                             TIMER_CONT_UP + \
                             TIMER_CLK_PRESCALE_X_1 + \
                             TIMER_ENABLE_BY_OWN + \
                             TIMER_CLOCK_SRC_INTERNAL + \
                             TIMER_ENABLE)

```

```

/*-----
Initialization constant for the F2407 CAPCONA register for Capture config.
Sets up edge detection mode.
-----*/
#define CAPCON_INIT_STATE_A ( EN_CAP_1AND2 + \
                             CAP12_GPT2 + \
                             CAP1_EDGE_RISE)

/*-----
Initialization for the EVAIMRC interrupt enable register
-----*/
#define EVAIMRC_INIT_STATE ( CAP1INT )

/*-- ** EVB-----
/*-----
Initialization constant for the F2407 TxCON register for PWM Generation.
Sets up the timer frequency and modes.
-----*/
#define T2CON_INIT_STATE_B ( FREE_RUN_FLAG + \
                             TIMER_CONT_UPDN + \
                             TIMER_CLK_PRESCALE_X_1 + \
                             TIMER_ENABLE_BY_OWN + \
                             TIMER_ENABLE + \
                             TIMER_CLOCK_SRC_INTERNAL + \
                             TIMER_COMPARE_LD_ON_ZERO + \
                             TIMER_ENABLE_COMPARE )

/*-----
Initialization constant for the F2407 CAPCONA register for Capture config.
Sets up edge detection mode.
-----*/
#define CAPCON_INIT_STATE_B ()

/*-----
Initialization for the EVAIMRC interrupt enable register
-----*/
#define EVBIMRC_INIT_STATE ()

/*-----
Define the structure of the capture Object
-----*/
typedef struct {
    int (*initCapture)(); /* Pointer to the init function */
    int D2; /* save location for counter value */
} CAPTURE ;

/*-----
Define a PWMGEN_handle
-----*/
typedef CAPTURE *CAPTURE_handle;

/*-----
Prototypes for functions
-----*/
int initCapture(CAPTURE *);

```

## D.5 Capture.c

```
#include "capture.h"
#include "f2407_c.h"
#include "std.h"

int initCapture(CAPTURE *cap)
{
    cap->D2 = 0;

    /* In PWM A block setup timer 2 for capture interrupt */
    *T2PR = 0xFFFF; /* set period to max so as not to reset */
    *T2CNT = 0; /* zero the counter */
    *T2CON = T2CON_INIT_STATE_A;
    *CAPCONA = CAPCON_INIT_STATE_A;
    *EVAIMRC = EVAIMRC_INIT_STATE;
    set(IMR, BIT3); /* enable global system interrupt for capture */
}
/* This is the original code as created by Chris Smith */
/* Minor modification were made to work in this application */
/* this type of code needs to go in the same file as main function
-----

ALSO MAKE SURE TO MAKE INT4 POINT TO "_capture" in cvectors.asm
*/
void interrupt capture()
{
    static int state = 1; /* waiting on fall = 2, waiting on rise = 1 */

    /* if(state == 1) /* waiting on rise */
    /* {
        clear(CAPCONA, CAP1_EDGE_RISE);
        set(CAPCONA, CAP1_EDGE_FALL);
        state = 2;

        *T2CNT = 0;
    }
    else /* waiting on fall */
    /* {
        clear(CAPCONA, CAP1_EDGE_FALL);
        set(CAPCONA, CAP1_EDGE_RISE);
        state = 1;

        cap.time = *T2CNT;
    }

    set(EVAIFRC, CAP1INT); /* clear capture 1 interrupt */
} */
```

## D.6 ADC.h

```
/*-----  
F2407 Register ADCTRL1  
-----*/  
  
#define ADC_RESET_FLAG      0x4000  
#define ADC_SOFT_STOP_FLAG  0x2000  
#define ADC_FREE_RUN_FLAG   0X1000  
  
#define ADC_ACQ_PS_1        0x0000  
#define ADC_ACQ_PS_2        0x0100  
#define ADC_ACQ_PS_3        0x0200  
#define ADC_ACQ_PS_4        0x0300  
#define ADC_ACQ_PS_5        0x0400  
#define ADC_ACQ_PS_6        0x0500  
#define ADC_ACQ_PS_7        0x0600  
#define ADC_ACQ_PS_8        0x0700  
#define ADC_ACQ_PS_9        0x0800  
#define ADC_ACQ_PS_10       0x0900  
#define ADC_ACQ_PS_11       0x0a00  
#define ADC_ACQ_PS_12       0x0b00  
#define ADC_ACQ_PS_13       0x0c00  
#define ADC_ACQ_PS_14       0x0d00  
#define ADC_ACQ_PS_15       0x0e00  
#define ADC_ACQ_PS_16       0x0f00  
  
#define ADC_CPS_1           0x0000  
#define ADC_CPS_2           0x0080  
#define ADC_CONT_RUN        0x0040  
#define ADC_INT_PRI         0x0020  
#define ADC_SEQ_CASC        0x0010  
  
#define ADC_CAL_ENA         0x0008  
#define ADC_BRG_ENA         0x0004  
#define ADC_HI_LO           0x0002  
#define ADC_STEST_ENA       0x0001  
  
/*-----  
F2407 Register ADCTRL2  
-----*/  
  
#define ADC_EVB_SOC         0x8000  
#define ADC_RST_SEQ1        0x4000  
#define ADC_SOC_SEQ1        0x2000  
#define ADC_SEQ1_BSY        0x1000  
  
#define ADC_INT_ENA_SEQ1_MODE1 0x0400  
#define ADC_INT_ENA_SEQ1_MODE2 0X0800  
  
#define ADC_INT_FLAG_SEQ1    0x0200  
#define ADC_EVA_SOC_SEQ1    0x0100  
  
#define ADC_EXT_SOC_SEQ1    0x0080  
#define ADC_RST_SEQ2        0x0040
```

```

#define ADC_SOC_SEQ2      0x0020
#define ADC_SEQ2_BSY     0x0010

#define ADC_INT_ENA_SEQ2_MODE1 0x0004
#define ADC_INT_ENA_SEQ2_MODE2 0x0008

#define ADC_INT_FLAG_SEQ2  0x0002
#define ADC_EVB_SOC_SEQ2   0x0001

/*-----
F2407 Register MAX_CONV
-----*/
#define ADC_MAX_CONV_SEQ2_1  0x0000
#define ADC_MAX_CONV_SEQ2_2  0x0010
#define ADC_MAX_CONV_SEQ2_3  0x0020
#define ADC_MAX_CONV_SEQ2_4  0x0030
#define ADC_MAX_CONV_SEQ2_5  0x0040
#define ADC_MAX_CONV_SEQ2_6  0x0050
#define ADC_MAX_CONV_SEQ2_7  0x0060
#define ADC_MAX_CONV_SEQ2_8  0x0070

#define ADC_MAX_CONV_SEQ1_1  0x0000
#define ADC_MAX_CONV_SEQ1_2  0x0001
#define ADC_MAX_CONV_SEQ1_3  0x0002
#define ADC_MAX_CONV_SEQ1_4  0x0003
#define ADC_MAX_CONV_SEQ1_5  0x0004
#define ADC_MAX_CONV_SEQ1_6  0x0005
#define ADC_MAX_CONV_SEQ1_7  0x0006
#define ADC_MAX_CONV_SEQ1_8  0x0007
#define ADC_MAX_CONV_SEQ1_9  0x0008
#define ADC_MAX_CONV_SEQ1_10 0x0009
#define ADC_MAX_CONV_SEQ1_11 0x000a
#define ADC_MAX_CONV_SEQ1_12 0x000b
#define ADC_MAX_CONV_SEQ1_13 0x000c
#define ADC_MAX_CONV_SEQ1_14 0x000d
#define ADC_MAX_CONV_SEQ1_15 0x000e
#define ADC_MAX_CONV_SEQ1_16 0x000f

/*-----
Define the structure of the ADCGEN
-----*/

typedef struct {
    int (*initADC());          /* Initialization func pointer */
    int (*updateADC());       /* Update function */
} ADC_SAMPLER;

/*-----
Define a ADCVALS_handle
-----*/

typedef ADC_SAMPLER *ADC_handle;

```

```

/*-----
Default Initializers for the F2407 ADCVALS Object
-----*/

#define ADC_DEFAULTS { (int (*)(int))initADC, \
                      (int (*)(int))updateADC \
                      }

#define CALIBRATION_CONSTANT 0

#define ADCTRL1_INIT_STATE ADC_SOFT_STOP_FLAG + ADC_ACQ_PS_3 + \
                          ADC_SEQ_CASC + ADC_CPS_1

#define ADCTRL2_INIT_STATE ADC_SOC_SEQ1

#define MAX_CONV_INIT_STATE ADC_MAX_CONV_SEQ1_3

/*-----
Function prototypes
-----*/

int initADC(void);
int updateADC(void);

```

## D.7 ADC.c

```
#include "f2407_c.h"
#include "adc.h"
```

```
int initADC(void)
{
    *ADCTRL1 = 0x4000; /* reset ADC module */

    *ADCTRL1 = ADCTRL1_INIT_STATE;
    *ADCTRL2 = ADCTRL2_INIT_STATE;
    *MAX_CONV = MAX_CONV_INIT_STATE;

    *CHSELSEQ1 = 0x0210; /* for simplicity we will do them all in order */
    *CHSELSEQ2 = 0x0000;
    *CHSELSEQ3 = 0x0000;
    *CHSELSEQ4 = 0x0000;

    return 1;
}

int updateADC(void)
/* returns 1 if the conversion was ready, 0 if not */
{
    if((*ADCTRL2 & ADC_SOC_SEQ1) == 0) /* conversion done */
        *ADCTRL2 = ADC_SOC_SEQ1; /* start a new conversion */
    else return 0; /* not ready yet */

    return 1;
}
```

## D.8 INIT.h

```
#include "f2407_c.h
#define CLOCK_4000000 /* 40 MHz */
/***** SETUP for the MCRA - Register *****/
#define MCRA15      0 /* 0 : IOPB7   1 : TCLKIN   */
#define MCRA14      0 /* 0 : IOPB6   1 : TDIR     */
#define MCRA13      1 /* 0 : IOPB5   1 : T2PWM    */
#define MCRA12      0 /* 0 : IOPB4   1 : T1PWM    */
#define MCRA11      1 /* 0 : IOPB3   1 : PWM6     */
#define MCRA10      1 /* 0 : IOPB2   1 : PWM5     */
#define MCRA9       1 /* 0 : IOPB1   1 : PWM4     */
#define MCRA8       1 /* 0 : IOPB0   1 : PWM3     */
#define MCRA7       1 /* 0 : IOPA7   1 : PWM2     */
#define MCRA6       1 /* 0 : IOPA6   1 : PWM1     */
#define MCRA5       0 /* 0 : IOPA5   1 : CAP3     */
#define MCRA4       1 /* 0 : IOPA4   1 : CAP2/QEP2  */
#define MCRA3       1 /* 0 : IOPA3   1 : CAP1/QEP1  */
#define MCRA2       0 /* 0 : IOPA2   1 : XINT1    */
#define MCRA1       1 /* 0 : IOPA1   1 : SCIRXD   */
#define MCRA0       1 /* 0 : IOPA0   1 : SCITXD   */
/*****
/***** SETUP for the MCRB - Register *****/
#define MCRB9       0 /* 0 : IOPD1   1 : XINT2/EXTSOC */
#define MCRB8       1 /* 0 : CKLKOUT 1 : IOPD0     */
#define MCRB7       0 /* 0 : IOPC7   1 : CANRX    */
#define MCRB6       0 /* 0 : IOPC6   1 : CANTX    */
#define MCRB5       0 /* 0 : IOPC5   1 : SPISTE    */
#define MCRB4       0 /* 0 : IOPC4   1 : SPICLK    */
#define MCRB3       0 /* 0 : IOPC3   1 : SPISOMI   */
#define MCRB2       0 /* 0 : IOPC2   1 : SPISIMO    */
#define MCRB1       0 /* 0 : IOPC1   1 : BIO       */
#define MCRB0       0 /* 0 : IOPC0   1 : XF       */
/*****
/***** SETUP for the MCRC - Register *****/
#define MCRC13      0 /* 0 : IOPF5   1 : TCLKIN2  */
#define MCRC12      0 /* 0 : IOPF4   1 : TDIR2    */
#define MCRC11      0 /* 0 : IOPF3   1 : T4PWM/T4CMP  */
#define MCRC10      0 /* 0 : IOPF2   1 : T3PWM/T3CMP  */
#define MCRC9       0 /* 0 : IOPF1   1 : CAP6     */
#define MCRC8       0 /* 0 : IOPF0   1 : CAP5/QEP3  */
#define MCRC7       0 /* 0 : IOPE7   1 : CAP4/QEP2  */
#define MCRC6       1 /* 0 : IOPE6   1 : PWM12    */
#define MCRC5       1 /* 0 : IOPE5   1 : PWM11    */
#define MCRC4       1 /* 0 : IOPE4   1 : PWM10    */
#define MCRC3       1 /* 0 : IOPE3   1 : PWM9     */
#define MCRC2       1 /* 0 : IOPE2   1 : PWM8     */
#define MCRC1       1 /* 0 : IOPE1   1 : PWM7     */
#define MCRC0       1 /* 0 : IOPE0   1 : CLKOUT   */
/*****
/***** SETUP for the WDCR - Register *****/
#define WDDIS       1 /* 0 : Watchdog enabled 1: disabled */
#define WDCHK2      1 /* 0 : System reset 1: Normal OP */
#define WDCHK1      0 /* 0 : Normal Oper. 1: sys reset */
#define WDCHK0      1 /* 0 : System reset 1: Normal OP */
```

```

#define WDSP                0        /* Watchdog prescaler 7 : div 64 */
/*****
/***** SETUP for the SCSR1 - Register *****/
#define CLKSRC              0        /* 0 : intern(20MHz) */
#define LPM                 0        /* 0 : Low power mode 0 if idle */
#define CLK_PS              0        /* 000 : PLL multiply by 4 */
#define ADC_CLKEN          1        /* 0 : ADC-service in this test */
#define SCI_CLKEN          1        /* 0 : No SCI-service in this test */
#define SPI_CLKEN          0        /* 0 : No SPI-service in this test */
#define CAN_CLKEN          0        /* 0 : No CAN-service in this test */
#define EVB_CLKEN          1        /* 0 : No EVB-Service in this test */
#define EVA_CLKEN          1        /* 1 : EVA-Service in this test */
#define ILLADR              1        /* 1 : Clear ILLADR during startup */
/*****
/***** SETUP for the WSGR - Register *****/
#define BVIS                0        /* 10-9 : 00 Bus visibility OFF */
#define ISWS                0        /* 8 -6 : 000 0 Waitstates for IO */
#define DSWS                0        /* 5 -3 : 000 0 Waitstates data */
#define PSWS                0        /* 2 -0 : 000 0 Waitstaes code */
/*****

void init(); /* initialize all settings above */

```

## D.9 Init.c

```
#include "init.h"
#include "pwm.h"
#include "adc.h"
#include "sgen.h"
#include "lcd.h"
#include "capture.h"

extern PWMGEN PWM;
extern ADC_SAMPLER ADC;
extern CAPTURE cap;

void init()
{
    asm (" setc INTM");          /* Disable all interrupts
    */
    asm (" clrc SXM");          /* Clear Sign Extension Mode bit
    */
    asm (" clrc OVM");          /* Reset Overflow Mode bit
    */
    asm (" clrc CNF");          /* Configure block B0 to data mem.
    */

    WSGR=((BVIS<<9)+(ISWS<<6)+(DSWS<<3)+PSWS);
    /* set the external waitstates      WSGR    */

    *WDCR=((WDDIS<<6)+(WDCHK2<<5)+(WDCHK1<<4)+(WDCHK0<<3)+WDSP);
    /* Initialize Watchdog-timer
    */

    *SCSR1=((CLKSRC<<14)+(LPM<<12)+(CLK_PS<<9)+(ADC_CLKEN<<7)+
            (SCI_CLKEN<<6)+(SPI_CLKEN<<5)+(CAN_CLKEN<<4)+
            (EVB_CLKEN<<3)+(EVA_CLKEN<<2)+ILLADR);
    /* Initialize SCSR1
    */

    *SCSR2 = (*SCSR2 | 0x004B) & 0x000F;
    /*
    bit 15-7  0's: reserved
    bit 6      1: input qualifier
    bit 5      0: do NOT clear the WD OVERRIDE bit
    bit 4      0: XMIF_HI-Z, 0=normal mode, 1=Hi-Z'd
    bit 3      1: disable the boot ROM, enable the FLASH
    bit 2      no change MP/MC* bit reflects state of MP/MC* pin
    bit 1-0    11: 11 = SARAM mapped to prog and data
    */

    *MCRC = ((MCRC13<<13)+(MCRC12<<12)+(MCRC11<<11)+(MCRC10<<10)+
            +(MCRC9<<9)+(MCRC8<<8)+(MCRC7<<7)+(MCRC6<<6)+
            +(MCRC5<<5)+(MCRC4<<4)+(MCRC3<<3)+(MCRC2<<2)+
            +(MCRC1<<1)+MCRC0);
    /* Initialize master control register C
    */

    *MCRB = ((MCRB9<<9)+(MCRB8<<8)+
            (MCRB7<<7)+(MCRB6<<6)+(MCRB5<<5)+(MCRB4<<4)+
```

```

(MCRB3<<3)+(MCRB2<<2)+(MCRB1<<1)+MCRB0);
/* Initialize master control register B */

*MCRA = ((MCRA15<<15)+(MCRA14<<14)+(MCRA13<<13)+(MCRA12<<12)+
(MCRA11<<11)+(MCRA10<<10)+(MCRA9<<9)+(MCRA8<<8)+
(MCRA7<<7)+(MCRA6<<6)+(MCRA5<<5)+(MCRA4<<4)+
(MCRA3<<3)+(MCRA2<<2)+(MCRA1<<1)+MCRA0);
/* Initialize master control register A */

/** Setup the core interrupts */
*IMR = 0x0000; /* clear the IMR register */
*IFR = 0x003F; /* clear any pending core interrupts */
*IMR = 0x0000; /* enable desired core interrupts */

/** Setup the event manager interrupts */
*EVAIFRA = 0xFFFF; /* clear all EVA group A interrupts */
*EVAIFRB = 0xFFFF; /* clear all EVA group B interrupts */
*EVAIFRC = 0xFFFF; /* clear all EVA group C interrupts */
*EVAIMRA = 0; /* disable all EVA interrupts */
*EVAIMRB = 0;
*EVAIMRC = 0;

*EVBIFRA = 0xFFFF; /* clear all EVB group A interrupts */
*EVBIFRB = 0xFFFF; /* clear all EVB group B interrupts */
*EVBIFRC = 0xFFFF; /* clear all EVB group C interrupts */
*EVBIMRA = 0; /* disable all EVA interrupts */
*EVBIMRB = 0;
*EVBIMRC = 0;

initPWM(&PWM); /* setup PWM stuff */
initADC(); /* setup analog to digital stuff */
initPorts(); /* setup output ports */
initCapture(&cap); /* setup capture timer and control */

asm (" clrc INTM"); /* enable all interrupts */

return;
}

```

## D.10 IO.h

```
char* var2hex(unsigned int var, char* buffer);
    /* var2hex -- convert integer to ascii hex
       var is an integer to convert
       buffer is a 4 character buffer to write to
       function returns a pointer to buffer */

char* var2percent(int var, char* buffer);
    /* var2percent -- convert integer to ascii percent
       var is an integer to convert
       buffer is a 3 character buffer to write to
       function returns a pointer to buffer */

typedef struct {
    volatile unsigned int* port;
    char bit;
} bit_io;

/* turn bits on and off without affecting rest of word */
void bit_on(bit_io bit_info);
void bit_off(bit_io bit_info);
char bit_test(bit_io bit_info);
```

## D.11 IO.c

```
#include "io.h"
#include "f2407_c.h"

char* var2hex(unsigned int var, char* buffer)
{
    char table[16] = {'0','1','2','3','4','5','6','7','8','9','A','B','C','D','E','F'};
    buffer[3] = table[(var & 0xF)];
    (buffer[2]) = table[(var & 0xF0)>>4];
    (buffer[1]) = table[(var & 0xF00)>>8];
    (buffer[0]) = table[(var & 0xF000)>>12];
    return buffer;
}

char* var2percent(int var, char* buffer)
{
    return buffer;
}

/* these next two functions could be made faster by
   not shifting the bit and using a predefined
   set of constants */
void bit_off(bit_io bit_info)
{
    *(bit_info.port) = *(bit_info.port) & ~(0x1 << bit_info.bit);
}

void bit_on(bit_io bit_info)
{
    *(bit_info.port) = *(bit_info.port) | (0x1 << bit_info.bit);
}

char bit_test(bit_io bit_info)
{
    return !((*(bit_info.port) & (0x1 << bit_info.bit))==0);
}

/* bit deffinitions */
extern bit_io LED;
extern bit_io up;
extern bit_io down;
extern bit_io toggle;
extern bit_io reset;
extern bit_io BBselect;
extern bit_io in_out;

void initPorts()
{
    /* 0 means output */
    /* 1 means input */
    /* set port directions */
    *PCDATDIR = *PCDATDIR | 0x0100; /* set to all inputs except LED*/
}
```

```
*PBDATDIR = *PBDATDIR | 0xFF00; /* set to all outputs */

LED.port = PCDATDIR;
LED.bit = 0;

up.port = PCDATDIR;
up.bit = 2; /* iopc2 */

down.port = PCDATDIR;
down.bit = 3; /* iopc3 */

toggle.port=PCDATDIR;
toggle.bit=4; /* iopc4 */

reset.port = PBDATDIR;
reset.bit = 7;

BBselect.port = PBDATDIR;
BBselect.bit = 6;

in_out.port = PBDATDIR;
in_out.bit = 4;
}
```

## D.12 PWM.h

```
/* constants for registers */

/*-----*/
F2407 Register TxCON
/*-----*/

#define FREE_RUN_FLAG      0x8000
#define SOFT_STOP_FLAG    0x4000

#define TIMER_STOP        0x0000
#define TIMER_CONT_UPDN   0x0800
#define TIMER_CONT_UP     0x1000
#define TIMER_DIR_UPDN    0x1800

#define TIMER_CLK_PRESCALE_X_1  0x0000
#define TIMER_CLK_PRESCALE_X_2  0x0100
#define TIMER_CLK_PRESCALE_X_4  0x0200
#define TIMER_CLK_PRESCALE_X_8  0x0300
#define TIMER_CLK_PRESCALE_X_16 0x0400
#define TIMER_CLK_PRESCALE_X_32 0x0500
#define TIMER_CLK_PRESCALE_X_64 0x0600
#define TIMER_CLK_PRESCALE_X_128 0x0700

#define TIMER_ENABLE_BY_OWN  0x0000
#define TIMER_ENABLE_BY_T1   0x0080

#define TIMER_ENABLE        0x0040
#define TIMER_DISABLE       0x0000

#define TIMER_CLOCK_SRC_INTERNAL 0x0000
#define TIMER_CLOCK_SRC_EXTERNAL 0x0010
#define TIMER_CLOCK_SRC_QEP      0x0030

#define TIMER_COMPARE_LD_ON_ZERO 0x0000
#define TIMER_COMPARE_LD_ON_ZERO_OR_PRD 0x0004
#define TIMER_COMPARE_LD_IMMEDIATE 0x0008

#define TIMER_ENABLE_COMPARE 0x0002
#define TIMER_SELECT_T1_PERIOD 0x0001

/*-----*/
F2407 Register ACTR 0x7413 BIT FIELD MASKS
/*-----*/

/*-----*/
Space Vector Direction Commands
/*-----*/

#define SV_DIRECTION_CW      0x8000
#define SV_DIRECTION_CCW    0x0000

/*-----*/
Space Vector Generation Vectors
/*-----*/
```

```

/*-----*/
#define SPACE_VECTOR_0          0x0000
#define SPACE_VECTOR_1          0x1000
#define SPACE_VECTOR_2          0x2000
#define SPACE_VECTOR_3          0x3000
#define SPACE_VECTOR_4          0x4000
#define SPACE_VECTOR_5          0x5000
#define SPACE_VECTOR_6          0x6000
#define SPACE_VECTOR_7          0x7000

```

```

/*-----*/
Compare action definitions

```

```

/*-----*/
#define COMPARE6_FL             0x0000
#define COMPARE6_AL             0x0400
#define COMPARE6_AH             0x0800
#define COMPARE6_FH             0x0C00

```

```

/*-----*/
#define COMPARE5_FL             0x0000
#define COMPARE5_AL             0x0100
#define COMPARE5_AH             0x0200
#define COMPARE5_FH             0x0300

```

```

/*-----*/
#define COMPARE4_FL             0x0000
#define COMPARE4_AL             0x0040
#define COMPARE4_AH             0x0080
#define COMPARE4_FH             0x00C0

```

```

/*-----*/
#define COMPARE3_FL             0x0000
#define COMPARE3_AL             0x0010
#define COMPARE3_AH             0x0020
#define COMPARE3_FH             0x0030

```

```

/*-----*/
#define COMPARE2_FL             0x0000
#define COMPARE2_AL             0x0004
#define COMPARE2_AH             0x0008
#define COMPARE2_FH             0x000C

```

```

/*-----*/
#define COMPARE1_FL             0x0000
#define COMPARE1_AL             0x0001
#define COMPARE1_AH             0x0002
#define COMPARE1_FH             0x0003

```

```

/*-----*/
F2407 Register COMCONA/COMCONB

```

```

/*-----*/
#define CMPR_ENABLE             0x8000
#define CMPR_LD_ON_ZERO         0x0000
#define CMPR_LD_ON_ZERO_OR_PRD 0x2000
#define CMPR_LD_IMMEDIATE       0x4000
#define SVENABLE                 0x1000
#define SVDISABLE                0x0000
#define ACTR_LD_ON_ZERO          0x0000
#define ACTR_LD_ON_ZERO_OR_PRD  0x0400

```

```
#define ACTR_LD_IMMEDIATE      0x0800
#define FCOMPOE                0x0200
```

```
/*-----
F2407 Register DBTCON
-----*/
```

```
#define DBT_VAL_0      0x0000
#define DBT_VAL_1      0x0100
#define DBT_VAL_2      0x0200
#define DBT_VAL_3      0x0300
#define DBT_VAL_4      0x0400
#define DBT_VAL_5      0x0500
#define DBT_VAL_6      0x0600
#define DBT_VAL_7      0x0700
#define DBT_VAL_8      0x0800
#define DBT_VAL_9      0x0900
#define DBT_VAL_10     0x0a00
#define DBT_VAL_11     0x0b00
#define DBT_VAL_12     0x0c00
#define DBT_VAL_13     0x0d00
#define DBT_VAL_14     0x0e00
#define DBT_VAL_15     0x0f00
```

```
#define EDBT3_DIS      0x0000
#define EDBT3_EN       0x0080
#define EDBT2_DIS      0x0000
#define EDBT2_EN       0x0040
#define EDBT1_DIS      0x0000
#define EDBT1_EN       0x0020
```

```
#define DBTPS_X32      0x0014
#define DBTPS_X16      0x0010
#define DBTPS_X8       0x000C
#define DBTPS_X4       0x0008
#define DBTPS_X2       0x0004
#define DBTPS_X1       0x0000
```

```
/* EVAIMRA constants */
```

```
#define TIOFINT        0x400
#define TIUFINT        0x200
#define TICINT         0x100
#define T1PINT         0x80
#define CMP3INT        0x8
#define CMP2INT        0x4
#define CMP1INT        0x2
#define PDPINTA        0x1
```

```
/* EVAIMRB constants */
```

```
#define PDPINTB        0x1
```

```
/*-----
** PWM A
-----
```

Initialization constant for the F2407 ACTRx register for PWM Generation.

```

Sets up PWM polarities.
-----*/
/* Boost Mode */
#define BOOST_STATE_A ( COMPARE1_FL + \
    COMPARE2_AL + \
    COMPARE3_FL + \
    COMPARE4_AH + \
    COMPARE5_AL + \
    COMPARE6_AL )

#define OFF_STATE ( COMPARE1_FL + \
    COMPARE2_FL + \
    COMPARE3_FL + \
    COMPARE4_FL + \
    COMPARE5_FL + \
    COMPARE6_FL )

/*-----
Initialization constant for the F2407 DBTCONx register for PWM Generation.
Sets up the dead band for PWM and sets up dead band values.
-----*/
#define DBTCON_INIT_STATE_A ( 0 )

/*-----
Initialization constant for the F2407 COMCONx register for PWM Generation.
Sets up the compare control methods.
-----*/
#define COMCON_INIT_STATE_A ( CMPR_ENABLE + \
                                                                    FCOMPOE )

/*-----
Initialization constant for the F2407 TxCON register for PWM Generation.
Sets up the timer frequency and modes.
-----*/
#define TCON_INIT_STATE_A ( FREE_RUN_FLAG + \
                                                                    TIMER_CONT_UPDN + \
                                                                    TIMER_CLK_PRESCALE_X_1 + \
                                                                    TIMER_ENABLE_BY_OWN + \
                                                                    TIMER_ENABLE + \
                                                                    TIMER_CLOCK_SRC_INTERNAL + \
                                                                    TIMER_COMPARE_LD_ON_ZERO + \
                                                                    TIMER_ENABLE_COMPARE )

/*-----
Initialization for the EVAIMRA interrupt enable register
-----*/
#define EVAIMRA_INIT_STATE ( PDPINTA + \
                                                                    CMP1INT )

/*-----
** PWM B
-----
Initialization constant for the F2407 ACTRx register for PWM Generation.
Sets up PWM polarities.
-----*/
/* Boost State */

```

```

#define PWM_D_A_STATE_A ( COMPARE1_AL + \
    COMPARE2_AL + \
    COMPARE3_AH + \
    COMPARE4_AL + \
    COMPARE5_AL + \
    COMPARE6_AL )

/*-----
Initialization constant for the F2407 DBTCONx register for PWM Generation.
Sets up the dead band for PWM and sets up dead band values.
-----*/
#define DBTCON_INIT_STATE_B ( 0 )

/*-----
Initialization constant for the F2407 COMCONx register for PWM Generation.
Sets up the compare control methods.
-----*/
#define COMCON_INIT_STATE_B ( CMPR_ENABLE + \
    FCOMPOE )

/*-----
Initialization constant for the F2407 TxCON register for PWM Generation.
Sets up the timer frequency and modes.
-----*/
#define TCON_INIT_STATE_B ( FREE_RUN_FLAG + \
    TIMER_CONT_UPDN + \
    TIMER_CLK_PRESCALE_X_1 + \
    TIMER_ENABLE_BY_OWN + \
    TIMER_ENABLE + \
    TIMER_CLOCK_SRC_INTERNAL + \
    TIMER_COMPARE_LD_ON_ZERO + \
    TIMER_ENABLE_COMPARE )

/*-----
Initialization for the EVBIMRA interrupt enable register
-----*/
#define EVBIMRA_INIT_STATE ( PDPINTA )

/*-----
Define the structure of the PWM Driver Object
-----*/
typedef struct {
    unsigned int periodA_max;    /* PWMA Period in CPU clock cycles. Q0-Input */
    unsigned int periodB_max;    /* PWMB Period in CPU clock cycles. Q0-Input */
    unsigned int mfunc_c1;      /* PWM 1&2 Duty cycle ratio. Q15, Input */
    unsigned int mfunc_c2;      /* PWM 3&4 Duty cycle ratio. Q15, Input */
    unsigned int mfunc_c3;      /* PWM 5&6 Duty cycle ratio. Q15, Input */
    unsigned int mfunc_c4;      /* PWM 1&2 Duty cycle ratio. Q15, Input */
    unsigned int mfunc_c5;      /* PWM 3&4 Duty cycle ratio. Q15, Input */
    unsigned int mfunc_c6;      /* PWM 5&6 Duty cycle ratio. Q15, Input */
    int (*initPWM)();           /* Pointer to the init function */
    int (*updatePWM)();        /* Pointer to the update function */
} PWMGEN ;

/*-----

```

```

Define a PWMGEN_handle
-----*/
typedef PWMGEN *PWMGEN_handle;

/*-----
Default Initializers for the F2407 PWMGEN Object
-----*/
/* For 10kHz Switching Frequency */

#define PWM_DEFAULTS
    {2000, \
     2000, \
     0, 0, 0, 0, 0, 0, \
     (int (*)(int))initPWM, \
     (int (*)(int))updatePWM \
    }

/*-----
Prototypes for functions
-----*/
int initPWM(PWMGEN *);
int updatePWM(PWMGEN *);

```

## D.13 PWM.c

```
#include "pwm.h"
#include "f2407_c.h"
#include "std.h"

/* This code is to be used for just the complimentary outputs of EVA and EVB. Timer 2 and 4 stuff needs
to be separate. */
int initPWM(PWMGEN *pwm)
{
    /* PWM A */
    *ACTRA = BOOST_STATE_A;           /* setup PWM polarities */
    *DBTCONA = DBTCON_INIT_STATE_A;  /* setup dead times */
    *COMCONA = COMCON_INIT_STATE_A;  /* setup compare control register */
    *T1PR = pwm->periodA_max;         /* setup period */
    *T1CNT = 0;                       /* zero the counter */
    *T1CON = TCON_INIT_STATE_A;       /* setup timer mode */
    *EVAIMRA = EVAIMRA_INIT_STATE;    /* enable desired EVA group A interrupts */
    set(IMR, BIT0);                   /* enable interrupt for fault protection */
    set(IMR, BIT1);                   /* enable interrupt on GPTimer 1 match */

    /* PWM B */
    *ACTRB = PWM_D_A_STATE_A;         /* setup PWM polarities for D/A outputs*/
    *DBTCONB = DBTCON_INIT_STATE_A;  /* setup dead times */
    *COMCONB = COMCON_INIT_STATE_A;  /* setup compare control register */
    *T3PR = pwm->periodA_max;         /* setup period */
    *T3CNT = 0;                       /* zero the counter */
    *T3CON = TCON_INIT_STATE_A;       /* setup timer mode */
    *EVBIMRA = EVBIMRA_INIT_STATE;    /* enable desired EVB group A interrupts */
    return 1;
}

int updatePWM(PWMGEN *pwm)
/* calculate pwm counts based on 1.15 duty cycle */
/* any compare value not used can be commented out for efficiency */
{
    extern signed int buffer[100];
    extern long flag;
    extern int count;

    *CMPR1 = (((unsigned long)pwm->mfunc_c1)*((unsigned long)pwm->periodA_max)) >> 16;
    *CMPR2 = (((unsigned long)pwm->mfunc_c2)*((unsigned long)pwm->periodA_max)) >> 16;
    *CMPR3 = (((unsigned long)pwm->mfunc_c3)*((unsigned long)pwm->periodA_max)) >> 16;

    *CMPR4 = (((unsigned long)pwm->mfunc_c4)*((unsigned long)pwm->periodB_max)) >> 16;
    *CMPR5 = (((unsigned long)pwm->mfunc_c5)*((unsigned long)pwm->periodB_max)) >> 16;
    *CMPR6 = (((unsigned long)pwm->mfunc_c6)*((unsigned long)pwm->periodB_max)) >> 16;

    return 1;
}
```

## D.14 STD.h

```
#define BIT0    0x1
#define BIT1    0x2
#define BIT2    0x4
#define BIT3    0x8
#define BIT4    0x10
#define BIT5    0x20
#define BIT6    0x40
#define BIT7    0x80
#define BIT8    0x100
#define BIT9    0x200
#define BIT10   0x400
#define BIT11   0x800
#define BIT12   0x1000
#define BIT13   0x2000
#define BIT14   0x4000
#define BIT15   0x8000
```

```
void set(volatile unsigned int* address, char mask);
void clear(volatile unsigned int* address, char mask);
char test(volatile unsigned int* address, char mask);
```

## D.15 STD.c

```
#include "std.h"

void set(volatile unsigned int* address, char mask)
/* mask: 1 in bits you want to set */
{
    *address |= mask;
}

void clear(volatile unsigned int* address, char mask)
/* mask: 1 in bits you want to clear */
{
    *address &= ~mask;
}

char test(volatile unsigned int* address, char mask)
{
    return !((*address & mask)==0);
}
```

## Appendix E - Buck Mode C Code

### E.1 Main.c

```
#include "f2407_c.h"
#include "pwm.h"
#include "adc.h"
#include "io.h"
#include "std.h"
#include "Qmath.h"
#include "capture.h"
#include "VConst.h"

#define MAX_DC          0x4000          /* Set to 0.9          */
#define V_Batt_Max     0x66D2          /* 58.8V = 0x66D2     */
#define PWM_scale      0x1893          /* 0.192              */
#define Vscale         0x2467          /* Scaling VBatt to VBus */
#define G              0x6FBB          /* Gain value of Ts/(2*L) */

/* EVERYTHING DECLARED HERE, BEFORE MAIN, IS GLOBAL */

/* declare the parts of DSP used */
PWMGEN PWM = PWM_DEFAULTS;
ADC_SAMPLER ADC = ADC_DEFAULTS;
CAPTURE cap;

/* bit definitions */
bit_io LED;
bit_io up;
bit_io down;
bit_io toggle;
bit_io reset;
bit_io BBselect;
bit_io in_out;

int GO,count,OV_count;

unsigned int yA = 0x0A00;

signed int Vref = 0;
signed int Iref = 0;

int i;
int count =0;
signed int output_PWM1 = 0;
signed int output_PWM2 = 0;
signed int output_PWM3 = 0;
signed int output_PWM4 = 0;

void main()
{
    O=0;
    V_count=0;
```

```

init();                /*Initilaze all parts of DSP      */
bit_off(reset);        /* reset gate drives with AL signal */
for (i=0; i<=50;i++)  /* Wait awhile for proper reset    */
    {}
bit_on(reset);         /* set reset bit high, Gate drives ready */
bit_on(BBselect);     /* Select Buck mode, multiplexes correct capture */
                       /*channel into the DSP              */

bit_off(in_out);
PWM.mfunc_c1 = 0x0A00; /* Set PWM to a small duty-cycle to begin interrupts */
PWM.mfunc_c2 = 0xFFFF - 0x0A00;
updatePWM(&PWM);

et(EVAIFRA, BIT1);    /* clear GPT1 interrupt */

while (1)
{
    /* main loop, all interrupt driven */
}
}

void interrupt TenkHz() /* 10 kHz interrupt */
{

    signed int Vn, In, Verror, Vcomp, Ierror, Icomp, ynI, ynV, Iavg3;
    signed int VBatt, VBus, D1, Base, Iavg, VL, Save;
    static int Ic[5] = {0x00FE, 0x02FA, 0x04F6, 0x09EC, 0x115D};
    static signed int Vce[6] = {0x0059, 0x005F, 0x007F, 0x0088, 0x009E, 0x00B9};

    int count2 = 0;
    static int V_limit = 0;
    static signed int yn = 0;
    static unsigned int Soft_Start = 0;
    unsigned int Vlow, Vhigh, y;
    unsigned long int Ans, Atemp, Btemp;

    asm ( " SETC OVM "); /* turn overflow mode ON */
    asm ( " clrc INTM"); /* Enable all interrupts so capture int will */
                       /* will nest within 10 kHz */
    set(EVAIFRC, CAP1INT); /* clear capture 1 interrupt */

    /* Control of the charging current and Open/Closed loop operation */

    if(!bit_test(toggle))
        GO=1;

    if(!bit_test(up))
        if((count++)%50 == 0)
        {
            Soft_Start = 1;
            if (!GO)
                yA++;
            else
                Iref++;
        }
}

```

```

if(!bit_test(down))
    if((count++)%50 == 0)
    {
        if (!GO)
            yA--;
        else
            Iref--;
    }

if((!bit_test(down)) && (!bit_test(up)))
    Iref = 0x0000;

/* Get current A/D value and reset A/D for next conversion */

Vlow = *RESULT0;
Vhigh = *RESULT1;

set(ADCTRL2,ADC_SOC_SEQ1);          /* start a new conversion */
set(ADCTRL2,ADC_SOC_SEQ1);          /* this is hack that appears to make */
set(ADCTRL2,ADC_SOC_SEQ1);          /* the A/D convert the value properly */

VBatt = Vlow >> 1;                   /*Shift the 16-bit A/D value to signed notation */
VBus = Vhigh >> 1;

if (VBatt > V_Batt_Max)
    {
        bit_on(in_out);               /* Check to see if the batteries are at */
        V_limit = 1;                 /* perfered charging voltage */
    }

/* Compute the voltage across the inductor */

VL = ((long) VBatt * (long) Vscale) >> 15; /* Scale Battery voltage to bus voltage */
/* scale */
Atemp = (long) VBus <<16;
Btemp = (long) VL <<16;
Ans = Atemp - Btemp;
VL = Ans>>16;                         /* subtract, VL = VBus - VBatt */

/* Correct VL for Device VCE drop */

while (Iavg >= Ic[count2])           /* Run through Ic until Iavg is larger */
    count2++;

VL = VL - Vce[count2];               /* subtract off VCE from VL */

/* COMPUTE AVERAGE CURRENT */

```

```

D1 = yn; /* Use last output duty-cycle as D1 */

Atemp = (long) D1 <<16; /* Add the rising and falling current */
Btemp = (long) cap.D2 <<16; /* times to get the base of the triangle */
Ans = Atemp + Btemp;
Base = Ans>>16;

Iavg = ((long) Base * (long) D1) >>15; /* MULTIPLY ALL VALUES TOGETHER */
Iavg = ((long) Iavg * (long) G) >>15; /* Scale by gain factor */
Iavg3 = ((long) Iavg * (long) VL) >>15; /* multiply by voltage across inductor*/

Iavg = Iavg3;

output_PWM1 = Iavg3;

/* Implement Controller */

if (!GO) /* Keeps the controller stable when running */
    Iref = Iavg; /* OL */

if (!V_limit) /* If current mode, set Vref to Vbatt */
    Vref = VBatt; /* keep voltage integrator up-to-date */
else
    Vref = V_Batt_Max; /* If at charging voltage, set reference to Vmax */

In = Iref-Iavg; /* Calculate current error */
Vn = Vref - VBatt; /* Calculate voltage error */

ynI = I_comp(In,V_limit); /* Call current compensator procedure */
/* Pass error and if in voltage or current */
/* mode for possible return to current mode */

ynV = V_comp(Vn,yn, V_limit); /* Call voltage comp procedure, pass Verror */
/* current duty-cycle and mode */
/* if in current mode, voltage integrator */
/* is set to the current duty-cycle from I_comp */

putput_PWM2 = VBatt; /* Set output duty-cycle according to the mode */

if (!V_limit)
    yn = ynI;
else
    yn = ynV;

if (yn>=MAX_DC) /* Set hard limit on duty cycle */
    yn=MAX_DC;

if (yn&(0x8000)) /* prohibit negative duty-cycles, set to 0 */
    yn=0x0000;

y = yn << 1; /* Convert yn to unsigned integer */

/* Limit Max output voltage and turn off PWM */

if ( VBatt >= 0x6A34 | OV_count>=4) /* If the voltage goes above 61V for more */

```

```

        if((OV_count++) >= 4)          /* than four cycles, turn everything off */
        {
            OV_count = 4;
            GO=0;
        }

/* soft-start procedure */

if (Soft_Start & !GO)
{
    if( yA <=3000)
        if((count++)%15 == 0)
            yA++;
            PWM.mfunc_c1 = yA;
            PWM.mfunc_c2 = 0xFFFF - yA;
}
else
    if (GO)
    {
        PWM.mfunc_c1 = y;
        PWM.mfunc_c2 = 0xFFFF - y;
    }

if (output_PWM1 >= 0)
    PWM.mfunc_c3 = output_PWM1 + 0x7FFF;
else
    PWM.mfunc_c3 = output_PWM1 - 0x8000;

/* if (output_PWM2 >= 0)
    PWM.mfunc_c4 = output_PWM2 + 0x7FFF;
else
    PWM.mfunc_c4 = output_PWM2 - 0x8000;

/* if (output_PWM3 >= 0)
    PWM.mfunc_c5 = output_PWM3 + 0x7FFF;
else
    PWM.mfunc_c5 = output_PWM3 - 0x8000;

if (output_PWM4 >= 0)
    PWM.mfunc_c6 = output_PWM4 + 0x7FFF;
else
    PWM.mfunc_c6 = output_PWM4 - 0x8000;
*/

updatePWM(&PWM);
}

void interrupt capture()
{
    static int state = 1; /* waiting on fall = 2, waiting on rise = 1 */
    signed int D2, Save;
    long int Atemp, Btemp, Ans;

```

```

if(state == 1) /* waiting on rise */
{
    *T2CNT = 0;          /* zero counter upon entry to start new conversion time */

    bit_on(LED);        /* turn on LED to check capture time */

    clear(CAPCONA, CAP1_EDGE_RISE); /* Clear rising edge detection */
    set(CAPCONA, CAP1_EDGE_FALL);  /* set falling edge detection for */
                                    /* diode turn-off */

    state = 2;

    set(EVAIFRC, CAP1INT); /* clear capture 1 interrupt to allow for falling */
                            /* edge detection */

}
else /* waiting on fall */
{

    D2 = *T2CNT;          /* Save timer which is now D2 */
    bit_off(LED);        /* Turn on LED to check end of conversion */
    clear(CAPCONA, CAP1_EDGE_FALL); /* clear falling edge mode */
    set(CAPCONA, CAP1_EDGE_RISE);  /* set rising edge trigger for next */
                                    /* cycle */

    state = 1;          /* alternates rising and falling edge detection*/
    cap.D2 = D2 << 3;  /* multiply by 8 */

    Save = ((long) D2 * PWM_scale) >> 15; /* multiply counter value by 0.192 */

    cap.D2 = cap.D2 + Save; /* add together for D2*8.192 */
                            /* This is the conversion from clock cycles */
                            /* to signed integer duty-cycle percentage */

    set(CAPFIFOA, CAP1_FIFO_ONE); /* Set FIFO to One entry */
    set(EVAIFRA, BIT1); /* clear GPT1 interrupt to allow for 10kHz */
                            /* interrupt on next cycle */

}
}

void interrupt fault()
{
    bit_on(LED); /* turn off the led */

    /* do nothing until reset */
}

```

## E.2 VIConst.h

```
#define A0I          0x7FFF      /* These are constant for all compensators */
#define B0I          0x0000      /* They are hardwired into the code */
#define A0V          0x7FFF
#define B0V          0x0000

/*      Buck Mode Voltage and Current Compensator Constants      */

/* Voltage Compensator */
#define A1V          -0x2F2B     /*-0.3685 */
#define B1V          0x5B62     /*0.7139*/

#define B0iV         0x006C     /* 0.0033 */
//cs = 2^-1
//L = 2^0

/* Current Compensator */
#define B1I          0x5B7A     /*0.7147*/
#define A1I          -0x6BD9     /* -0.8426 */

/* Integrator Gain */

#define B0iI         0x0142     /*0.098 */

// Cs = 2^-3
// L = 2^1
```

## E.3 VIComp.c

```
#include "f2407_c.h"
#include "VICConst.h"
extern int GO,yA;
extern signed int output_PWM1;
extern signed int output_PWM2;
extern signed int output_PWM3;
extern signed int output_PWM4;

signed int I_comp (signed int In, int V_limit)
{
    static signed int yI_1 = 0;
    signed int h, yS, yn, yI;
    unsigned long int Atemp, Btemp, Ans;
    signed int xn;
    signed int vS, ySa;
    static signed int vS_1 = 0;

    asm ( " SETC OVM ");          /* Reset overflow mode to ON      */
    xn = In>>3;                  /* shift by 3 to divide by cs for scaling */

    /* Implement the Poles and Zeros of the Compensator      */

    h = ( (long) A1I * (long) vS_1 ) >>15;    /* A1*vS(n-1) */

    Atemp = (long) xn <<16;
    Btemp = (long) h <<16;
    Ans = Atemp - Btemp;

    vS = Ans>>16;                /* vS = xn - h; */

    /* Output of the poles/zeros of the compensator      */

    yS = ( (long) B1I * (long) vS_1 ) >>15;    /* B1*vS(n-1) */

    /* Before multiplying by 2^L, must check for maximum value so as not to rollover      */

    if (!(yS&(0x8000)))          /* Check to see if yS is positive */
    {
        if (((unsigned int)yS<=0x3FFF) & ((unsigned int)yS>=0))
            yS = yS<<1;        /* if less than 0.2499 then multiply */
        else
            yS = 0x7FFF;        /* Else limit multiplication at 1 */
    }
    else
        if (((unsigned int)yS>=0xC000) & ((unsigned int)yS<=0xFFFF))
            yS = yS<<1;        /* Check to see if negative */
        else
            yS = 0x8000;        /* multiply if less than -0.2499 */
                                /* Limit to -1 if too large */

    vS_1=vS;                    /* Save last value of vS */
}
```

```

/* Implementation of the Integrator */

h = ( (long) B0iI * (long) In ) >>15;          /* Use unscaled error          */

Atemp = (long) h<<16;
Btemp = (long) yI_1 <<16;          /* this will limit the output of the */
Ans = Atemp + Btemp;              /* integrator to -1 or 1 due to OVM */
yI = Ans>>16;                      /* yI = h + yI_1 */

if (!GO)                            /* Check to see if CL, if not set integrator to */
    yI = yA >> 1;                    /* current value of duty-cycle in OL          */

yI_1 = yI;                          /* Save last value of yI */

/* Add the output of the poles/zeros and integrator */

Atemp = (long) yI <<16;
Btemp = (long) yS <<16;
Ans = Atemp + Btemp;                /* yn = yS + yI */
yn = Ans>>16;                       /* Will limit the output to 1 in hardware */

if (V_limit)                        /* Seed integrator in voltage mode for */
    yn = yI;                         /* possible return to current mode    */

/* Not sure if this really works */

return(yn);                          /* Return new value of duty-cycle */
}

signed int V_comp (signed int Vn, signed int yin, signed int V_limit)
{
    signed int xn, yn, R1, h;
    signed int vS, yS, yI, ySa;
    static signed int vS_1 = 0;
    static signed int yI_1 = 0;

    unsigned long int Atemp, Btemp, Ans, R0;

    asm ( " SETC OVM ");                /* Set overflow mode again          */
    xn = Vn>>1;                        /* Divide by two for cs scaling */

    /* Implement the Poles and Zeros of the Compensator */

    h = ( (long) A1V * (long) vS_1 ) >>15; /* A1*vS(n-1) */

    Atemp = (long) xn <<16;
    Btemp = (long) h <<16;
    Ans = Atemp - Btemp;
    vS = Ans>>16;                      /* vS = xn - h; */

    /* Output of the poles/zeros of the compensator */

    yS = ( (long) B1V * (long) vS_1 ) >>15; /* B1*vS(n-1) */

```

```

vS_1=vS;                /* Save last value of vS */

/* Implementation of the Integrator */

h = ( (long) B0iV * (long) Vn) >>15;    /* Use unscaled value of Verror */

Atemp = (long) h<<16;    /* this will limit the value of the */
Btemp = (long) yI_1 <<16; /* integrator to -1 or 1 due to OVM */
Ans = Atemp + Btemp;
yI = Ans>>16;           /* yI = h + yI_1 */

if (!V_limit)          /* if not in voltage mode, seed integrator */
    yI = yin;          /* with current duty-cycle value */

yI_1 = yI;             /* Save last value of yI */

/* Add the output of the poles/zeros and integrator */

Atemp = (long) yI <<16; /* this sum will be limited to -1 or 1 */
Btemp = (long) yS <<16; /* due to OVM */
Ans = Atemp + Btemp;   /* yn = yS + yI */
yn = Ans>>16;

return(yn);            /* return new value of duty-cycle */
}

```

## E.4 PWM.h

```
/* Buck Mode */
#define BUCK_STATE_A ( COMPARE1_AL + \
    COMPARE2_FL + \
    COMPARE3_AH + \
    COMPARE4_FL + \
    COMPARE5_AL + \
    COMPARE6_AL )
```

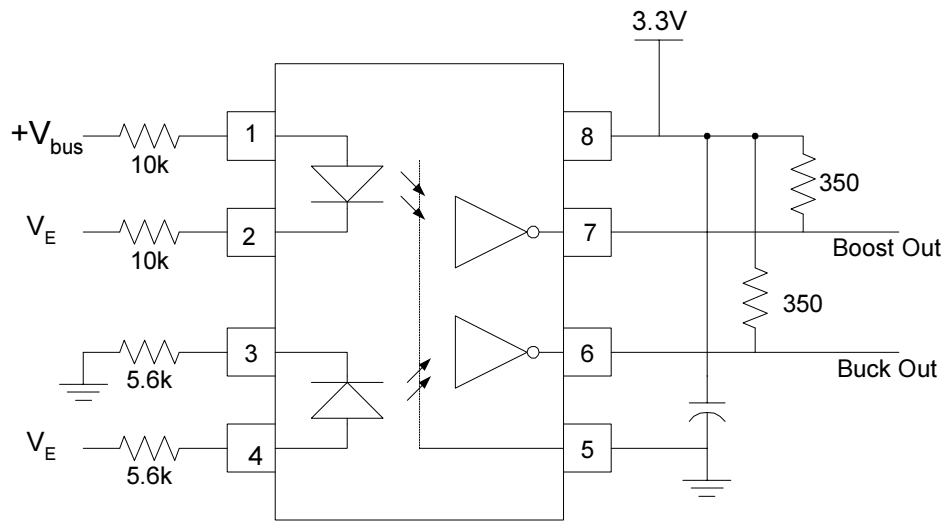
The rest of PWM.h in buck mode is the same as in boost mode.

## E.5 PWM.c

```
int initPWM(PWMGEN *pwm)
{
    /* PWM A */
    *ACTRA = BUCK_STATE_A;
...}
```

This is the only change that needs to be made to the boost code to switch to buck mode.

## Appendix F – Optocoupler Schematic



## References

- [1] D. M. Sable, F. C. Lee and B. O. Cho, "A Zero-Voltage-Switching Bidirectional Battery Charger / Discharger for the NASA EOS Satellite," in Proc. *Applied Power Electronics Conference and Exposition*, 1992, pp. 614 -621.
- [2] H. Li, F.Z Peng, J.S. Lawler, "A Natural ZVS High-power Bidirectional DC-DC Converter with Minimum Number of Devices," in Proc. *IEEE Industry Applications Conference*, 2001, pp. 1874 -1881.
- [3] Future Energy Competition Updated Specifications, 3, March 2003.
- [4] M. W. Ellis, M. R. Von Spakovsky, and D. J. Nelson, "Fuel cell systems: efficient, flexible energy conversion for the 21<sup>st</sup> century," Proceedings *IEEE*, Vol. 89, No. 12, Dec. 2001, pp. 1808-1818.
- [5] S. Thomas and M. Zalbowitz, "Fuel Cells-Green Power," Los Alamos National Laboratory (LA-UR-99-3231), 1999.
- [6] M. A. Laughton, "Fuel Cells," *Power Engineering Journal*, Feb 2002 pp. 37-47.
- [7] DOE Energy SOFC Webpage,  
[http://www.fe.doe.gov/coal\\_power/fuelcells/fuelcells\\_sofc.shtml](http://www.fe.doe.gov/coal_power/fuelcells/fuelcells_sofc.shtml)
- [8] P. Midya, M. Greuel, P. T. Krein, "Sensorless current mode control-an observer-based technique for DC-DC converters," in Proc. *Power Electronics Specialists Conference*, 1997, Vol.1, pp. 197-202.
- [9] T. A. Nergaard, J. F. Ferrell, L. G. Leslie, and J. S. Lai, "Design considerations for a 48 V fuel cell to split single phase inverter system with ultracapacitor energy storage," in Proc. *IEEE Power Electronics Specialist Conference*, 2002.
- [10] R. Gopinath, S. Kim, J. H. Hahn, M. Webster, J. Burghardt, S. Campbell, D. Becker, P. Enjeti, M. Yeary, J. Howze, "Development of a Low Cost Fuel Cell Inverter System with DSP Control," in Proc. *33rd Power Electronics Specialists Conference*, 2002.
- [11] S. West, P. T. Krein, "Equalization of valve-regulated lead-acid batteries: issues and life test results," in Proc. *IEEE Int'l Telecommunications Energy Conference*, 2000, pp. 439-446.
- [12] K. Wang, F. C. Lee, and J. Lai, "Operational Principles of Bidirectional Full-bridge DC/DC Converter with Unified Soft-switching Scheme and Soft-starting Capability," in Proc. *IEEE APEC*, 2000, Vol. 1, pp. 111-118.

- [13] K. Wang, C. Y. Lin, L. Zhu, D. Qu, F. C. Lee, J. S. Lai, "Bidirectional DC to DC Converter for Fuel Cell Systems," in Proc. *IEEE APEC*, 1998, pp. 47-51.
- [14] K. Wang, L. Zhu, D. Qu, H. Odendaal, J. Lai, and F. C. Lee, "Design, Implementation and Experimental Results of Bidirectional DC/DC Converter with Unified Soft-switching Scheme and Soft-starting Capability," in Proc. *VPEC Seminar*, 1998, pp. 150-156.
- [15] X. Huang, X. Wang, J. Ferrell, T. Nergaard, J. S. Lai, X. Xu, L. Zhu, "Parasitic ringing and design issues of high power interleaved boost converters," *IEEE Power Electronics Specialists Conference, 2002*, vol.1, pp. 30-35.
- [16] R. W. Erickson and D. Maksimovic, Fundamentals of Power Electronics, Second Edition, Kluwer Academic Publishers, 2001.
- [17] Dr. Fred. C. Lee, ECE 5254 Class Notes
- [18] IRG4PSC71UD Datasheet, International Rectifier, May 1999.
- [19] Kool Mu Handbook, [www.mag-inc.com](http://www.mag-inc.com)
- [20] Dr. Jason Lai, ECE5334 Class Notes
- [21] Charging pure-tin Lead batteries: A guide for Cyclon and Genesis products, First Edition," [www.hepi.com](http://www.hepi.com), March 1999.
- [22] HCPL-060L Datasheet, Agilent Technologies, October 2002.
- [23] TI Literature SPRU357B, "TMS320LF/LC240xA DSP Controllers Reference Guide - System and Peripherals," December 2001.
- [24] M. Veerachary, T. Senjyu and K. Uezato, "Modeling and analysis of interleaved dual boost converter," in Proc. *IEEE International Symposium Industrial Electronics*, 2001, vol. 2, pp. 718 -722.
- [25] J. Sun, D. M. Mitchell, M. F. Greuel, P. T. Krein, and R. M. Bass, "Averaged Modeling of PWM Converters Operating in Discontinuous Conduction Mode," *IEEE Transactions on Power Electronics*, 2001, vol. 16, no. 4, pp. 482-492.
- [26] John G. Proakis and Dimitris G. Manolakis, Digital Signal Processing – Principles, Algorithms, and Applications, Third Edition, Prentice Hall, 1996.
- [27] A. K. Aboagye, TI Application Report SPRA509, "Overflow Avoidance Techniques in Cascaded IIR Filter Implementations on the TMS320 DSP's," May 1999.

- [28] David M. Alter, TI Application Report SPRA490, "Using PWM Output as a Digital-to-Analog Converter on a TMS320C240 DSP," November 1998.

## **Vita**

Andy M. McLandrich was born in Cincinnati, Ohio in February of 1978. After graduation from Wyoming High School in June of 1996, he moved to Boulder, Colorado to pursue a degree in Electrical Engineering at the University of Colorado. Upon graduation in May of 2000, he took a job at local telecom firm Carrier Access and remained there until February of 2001 when he was laid off and decided to go back to graduate school. He entered school at Virginia Polytechnic and State University in fall of 2001 to pursue a masters in electrical engineering with a concentration in power electronics and control. After graduation, he hopes to move back to Boulder to ride trials and ski.