Enabling Miniaturized Grid-Interface Power Conversion

by

Alex J. Hanson

B.E., Electrical Engineering
Thayer School of Engineering at Dartmouth (2014)
S.M., Electrical Engineering and Computer Science
Massachusetts Institute of Technology (2016)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Signature redacted

Author ....
Department of Electrical Engineering and Computer Science
May 23, 2019

Signature redacted

Certified by ....
David J. Perreault,
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Signature redacted

Accepted by ....
Leslie A. Kolodziejski
Professor of Electrical Engineering and Computer Science
Chair, Department Committee on Graduate Students

Enabling Miniaturized Grid-Interface Power Conversion

by

Alex J. Hanson

Submitted to the Department of Electrical Engineering
and Computer Science
on May 23, 2019, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy in Electrical Engineering and Computer Science

Abstract

Many of the most critical challenges of the twenty-first century revolve around energy and its management. Improved performance (efficiency, density) in electrical energy management systems require advancements in a number of areas – semiconductor devices, passive energy storage components, and a variety of circuit- and system-level concerns.

The sections of this thesis are somewhat distinct and may find application in a great variety of circumstances. Nevertheless, they can be understood as contributions to a single application system: a grid-interface power converter. These kinds of converters have several unique aspects that make them good targets for research, including a heavy reliance on magnetic components, relatively high voltages for application of emerging GaN transistors, wide range of operating voltages and powers, and a twice-line-frequency energy storage component that is difficult to miniaturize.

This thesis will present a high-frequency inductor structure with greatly improved density, an exploration of the limits of magnetic-based current sensing, a method for characterizing GaN losses with large-signal excitations, a control approach for miniaturizing grid-interface energy buffers, and a grid-interface circuit with several advantages over the state of the art.

Thesis Supervisor: David J. Perreault
Title: Professor

Acknowledgments

This work was supported by funds from the National Science Foundation under Grants 1609240 and 1610719, Fairchild Semiconductor, the MIT Undergraduate Research Opportunities (UROP) Program, and the MASDAR MUSES program.

Much of the work in this thesis was accomplished by capable and creative undergraduate students under my direction. The magnetics innovations in Chapter 2 were largely carried out by Rachel Yang. The current sensing exploration in Chapter 3 was carried out by Noah Moroze. Andreea Martin accomplished the majority of the work in Chapter 4 on harmonic injection. Bryson Galapon similarly led the sensing of GaN transistor losses in Chapter 5. It has been my pleasure and privilege to oversee a small part of their potent trajectories.

Dave Perreault provided a great deal of guidance for the work in this thesis. The investigations into magnetic structures and harmonic injection were begun at his suggestion. The power factor correction circuit in Chapter 6 was originally devised by him and Seungbum Lim. His guidance has enhanced this work in countless ways in all of the projects described here. I wish to express special gratitude not only for his intellectual contributions to this work, but for his humanity and generous stewardship in his role as a mentor and ally of myself and all of his students.

I likewise thank my fellow students in LEES, from those who long since have graduated to those who have only recently arrived at MIT, for creating a healthy and supportive environment in which the work of this thesis could grow. The fruits of their ideas, intellectual refinements, and hunger to discuss and listen are scattered through many pages of this work.

For all of this support, I am grateful.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Many of the most critical challenges of the twenty-first century revolve around energy and its management: in urbanization, climate and ecology, economic development, and information technology. Evolving technology takes the form of application buzzwords like electric vehicles, renewable energy, microgrids, data centers, and internet of things. Enumerating these applications lends concreteness to the argument, but risks limiting its scope. The challenges of managing energy stem from something more enduring than a given moment in technological development – they stem from energy's central role in physics and nature's tendency to diffuse rather than concentrate, to dissipate rather than conserve. We do not often invoke entropy directly when discussing the technology of energy, but in every difficulty we encounter it haunts us.

It is tempting to turn a blind eye to these challenges when nature puts up such a strong defense. Are there not more exciting things to do when information is the currency of interest, rather than energy? Especially at a time when information is so abundant and energy is both scarce and uncooperative?

Yet, as the challenges in energy systems are not easily circumvented, neither are the needs. Cities will continue to become more polluted, the planet hotter, developing nations more energy-intensive, and ubiquitous computing more demanding.

Against the backdrop of these grandiose challenges and goals, the primary accomplishments of this thesis may appear as lifeless technological concerns – enabling power converters to operate at higher switching frequencies. Nevertheless, by oper-

ating at higher switching frequencies, power converters can be made smaller, faster, and more efficient. This improves their operation, enables increased proliferation of power electronics in existing applications, and enables new applications. Thus, before we dig into the technical weeds, we recognize that the accomplishments of this thesis are a (small) contribution to more efficiently and intelligently managing energy on a broader scale, with real impacts on important problems.

1.1 Background

The sections of this thesis are somewhat distinct and may find application in a variety of circumstances. Nevertheless, they can be understood as contributions to a single application system: a grid-interface power converter. These kinds of converters have several unique aspects that make them good targets for research, including a heavy reliance on magnetic components, relatively high voltages for application of GaN transistors, wide range of operating voltages and powers, and a twice-line-frequency energy storage component that is difficult to miniaturize.

Below, I briefly review the contents of each of the thesis chapters. Detailed relevant background can be found in the chapters themselves.

1.2 An HF Inductor Geometry

Prior work has revealed magnetic materials with low loss at HF ([1–3]. With much higher quality core materials, and little hope of finding a better winding material than copper, our attention turns to the geometric structure of magnetic components to improve their performance.

It is well known that high-frequency magnetic fields impinging on conductors induce lossy eddy currents. This concept is sometimes divided in the jargon into "skin effect" and "proximity effect," but it is the same physical phenomenon at work. For loss-limited components, these effects are the dominant concerns.

The first solution to these issues is to reduce the relevant conductor dimension

to less than a skin depth. For higher powers, multiple strands of thin wire are often braided together to form "litz" wire. As frequencies increase, thinner and thinner wire is necessary, but electromagnetically this strategy can continue indefinitely. The difficulty is mechanical, and AWG 48 wire is approximately the limit of what can be manufactured and braided economically, with high prices strands even larger than those used here. The diameter of AWG 48 wire is 31 µm, which is equal to one skin depth in copper at 4.5 MHz at 25 °C. Printed circuit boards are commonly available with copper as thin as 0.5 oz/ft^2 (17.5 µm) which corresponds to the skin depth at 14 MHz. The challenge in this technology is likewise mechanical. In this case, the difficulty is not in manufacturing thin copper, but rather in stacking many layers with complex interconnects. Electromagnetically, it is also not necessarily clear that edge-wound magnetic structures (as with PCB windings) are optimal. The overall strategy of using ultra-thin conductors thus has a physical limit of sorts. This limit is blurry, but rests roughly in the high-frequency (HF) regime (3–30 MHz). As switching frequencies approach and surpass this range, other techniques become necessary.

This chapter will examine an inductor structure that controls magnetic fields with several techniques instead of relying on exceptionally thin conductors. Prototype inductors achieved quality factors of 700–1000 1 in a 1 in^3 form factor. The prototype is a significant departure from typical magnetic structures and has opened additional lines of investigation.

Parts of this work have been published in [4, 5].

1.3 Current Sensing

Because of magnetic fields' origins in moving electric charges, current sensing often involves some magnetic components, like current transformers and Rogowski coils. It is known that such components can achieve bandwidths well into the MHz range and beyond, with high-end commercial examples having bandwidths of 200–1000 MHz [6].

Still, these examples are often not suitable for embedding as part of a control system, and their highly-engineered nature makes them prohibitively expensive ($1000+)

except as test equipment.

Here we explore an open-ended question – how well can custom current transformers and Rogowski coils perform at a size, cost, and design effort that would be reasonable to embed into an application? This chapter demonstrates examples of both kinds of sensors with bandwidths above 100 MHz, which is sufficient to embed in, for example, radio-frequency (rf) applications operating at the 13.56 MHz ISM band. The examples of each type are reasonably small, inexpensive, and straightforward to design. In other words, they achieve a sufficient cross-section of technical and business specifications to proliferate to a wide variety of application spaces.

1.4 Characterizing GaN Switches

High-speed gallium-nitride (GaN) and silicon-carbide (SiC) transistor switches have been the backbone of advanced power electronics research for a decade, with the first enhancement-mode GaN devices reaching the market in 2009 [7]. Because of their wide band gap, GaN and SiC switches can operate at higher voltages with lower resistances and capacitances than their silicon counterparts. SiC has dominated at high powers and voltages (roughly greater than 1 kV); GaN operates in the lower voltage territory.

Since most grid-interface power conversion today must be able to accomodate any grid voltage ("universal input," 90–240 Vac), GaN transistors have typically not had access to this space. Circuit designers in academia have turned to stacked topologies [8–10] to accommodate higher voltages, but commercial adoption remained weak. The introduction of 600 V GaN transistors in 2013 [11] opened the application of GaN to grid-interface power conversion directly.

It remains important to understand this new technology to maximize its impact, and while GaN may appear to mimic silicon power MOSFETs (just with better performance), it suffers from several potential difficulties.

The first is termed dynamic on resistance or dynamic $R_{ds,on}$. Dynamic $R_{ds,on}$ is an observation that GaN transistors exhibit higher resistance during their on-times

in a switching application than they do when conducting dc.

The second is due to energy losses in charging and discharging the parasitic output capacitance C_{oss}. This is not to be confused with switching loss (where the C_{oss} energy is simply lost when the switch shorts the capacitor); rather, this loss may be thought of as a resistance in series with C_{oss}.[1] Thus, even a soft-switched application may experience C_{oss} loss; for example, the transistor may turn off "softly," but the charging current into C_{oss} must still pass through R_{oss}.

Both dynamic $R_{ds,on}$ and C_{oss} loss are complex physical phenomena and typically appear as non-linear loss components. To date, there is no straightforward way to model their dependence on circuit characteristics like voltage, current, frequency, duty cycle, etc. Indeed, mere characterization of such components in authentic scenarios (e.g. hard switched vs soft switched) is lacking and, since losses may depend on the test conditions, testing in authentic scenarios is necessary.

Chapter 5 presents an approach for measuring both dynamic $R_{ds,on}$ and C_{oss} loss in 600 V GaN transistors. The measurement conditions are soft switched at high frequency with the ability to independently vary temperature. The waveforms used to evaluate loss closely mimic those of a number of high-frequency converters, especially in rf power amplifiers. This approach is thus a strong candidate for characterizing GaN transistors in authentic settings across several parameters (off-state voltage, on-state current, frequency, and temperature). This work is reported in [13].

1.5 Harmonic Injection

Another peculiar challenge in grid-interface power conversion is the presence of a large energy buffer whose sizing is necessary to buffer the pulsating energy from the low-frequency line (50–60 Hz). This capacitor can take up 25 % of system volume, a percentage that would increase as higher switching frequencies reduce the size of most other components.

[1] Modeling the C_{oss} loss as a resistor is conceptually useful, but virtually useless as a model, even with a non-linear resistance. Like core loss, it is currently best characterized empirically. Hysteresis modeling [12] may have more utility.

Chapter 4 investigates a technique known as "harmonic injection" to reduce the size of this buffer. The term derives from the perceived and semi-enforced need to draw current only at the grid's fundamental frequency; the injection of harmonic current has been shown to reduce the amount of energy that the large buffer needs to store. While the term "harmonic injection" is not incorrect, it potentially misplaces the readers' attention. This technique works by drawing more constant current from the grid (thus requiring less buffering); doing so effectively requires injection of harmonics, but not all combinations will work. Buffer reduction always requires harmonics; not all harmonics reduce the buffer.

This chapter explores the limits of harmonic inclusion – how much can the buffer be reduced while still obeying the IEC/EN 61000-3-2 standards for various device classes? This far-reaching exploration has been lacking in the literature, and has great importance for expanding this technique's application beyond lighting (its typical application) to especially higher power devices. The work in this chapter is reported in [14, 15].

1.6 Power Factor Correction Circuit

Having considered magnetic components, switching devices, and control, we finally turn to a grid-interface power converter circuit. In particular, Chapter 6 explores a power factor correction (PFC) circuit, usually the first in a two-stage architecture for interfacing the single-phase grid to an isolated low-voltage load. The PFC stage takes as its input any ac grid voltage and typically outputs an approximately dc voltage. The most frequent solution is the boost converter so that it can operate near the grid voltage zero-crossings; this choice requires that the output voltage be above the maximum peak-of-line with margin ($1.1\times$ 240 Vac = 375 Vac).

This solution has two limitations. The first is the obviously paradoxical approach of first boosting the input voltage when the ultimate goal is to step it down. This severely limits miniaturization of the second stage. The second, more subtle limitation is that the boost converter can only achieve soft switching when $V_{in} < V_{out}$, which is

violated for significant portions of the cycle for high ac line voltages. This precludes operation of such PFC stages with frequencies above a few hundred kHz. This limits miniaturization of the PFC stage itself.

This chapter presents another circuit that can achieve soft switching for nearly any combination of input and output voltage. It can also both buck and boost voltages, which means it can operate near the zero crossings of the line without requiring an output voltage of nearly 400 V. Thus the PFC can operate at greatly elevated frequencies with a low voltage output, miniaturizing both stages of the overall converter. This work has been reported in [3, 16, 17].

1.7 Impact

Most of the work in this thesis has been presented in peer-reviewed conference presentations and journal publications [4, 5, 13–15, 17, 18]. Continued industry support, especially with respect to magnetics, likewise points to the potential impact of this work. The motivations and potential impacts of each chapter individually are discussed within the chapters. Their overlap is considered in the conclusion.

Part I

High-Frequency Magnetics

Chapter 2

High Frequency Magnetic Components

Miniaturization of power electronics is often limited by the magnetic components due to high losses [19]. Although miniaturization of these components is still available with increased frequencies into the HF (3–30 MHz) range [2], significant design challenges remain. Skin and proximity effects play large roles at HF, where conventional litz wire solutions become less practical due to manufacturing difficulties for strands thinner than a skin depth [20]. Therefore, other approaches for reducing proximity effect, such as single-layer windings or multi-layer foil windings, have been investigated [20–23]. Fringing fields from gaps in the core also significantly increase winding loss, and various winding configurations and materials have been explored to deal with these effects [20,24,25]. In particular, distributed or quasi-distributed gaps have successfully mitigated fringing field effects [26] and are beginning to be implemented in cores on the market [27].

To better understand the design challenges for magnetic components at HF, much research has focused on modeling. Analytical models of conductor loss [28–33] and core loss [34, 35] have been developed, with some work targeting the HF range [36]. While modeling can provide valuable analysis tools, it leaves unclear how to effectively design HF magnetic components.

We propose an inductor structure suitable for high-frequency operation with large

ac currents (such that the magnetic field is constrained by loss and not saturation), along with analytic design guidelines to maximize its quality factor. The proposed structure achieves high Q through double-sided conduction in the winding and through quasi-distributed gaps. Section 2.1 provides an overview of the proposed inductor geometry. The design guidelines are discussed in Section 2.2, and automation of the design process is outlined in Section 2.3. In Section 2.4, an example design is provided for a 16.6 µH inductor designed for 2 A (peak) of ac current at 3 MHz. The example achieves a quality factor of 700 in simulation, and simulation results verify that the design guidelines achieve the desired low-loss features. In Section 2.5, we present a hardware prototype that achieves an experimental Q of 720, agreeing with simulations. In addition, we demonstrate the prototype improving the efficiency and thermal performance of a high-frequency, high-current-swing power converter (1–3 MHz). In Section 2.6, we discuss using litz wire in the proposed structure to reduce loss, present additional design guidelines for litz, and demonstrate improved performance of the prototype inductor with litz wire ($Q = 980$). We conclude that the proposed structure can achieve high Q and that the analytic design guidelines are effective in designing high-Q inductors operating at high frequency with large ac current components.

For details on the simulations performed here, refer to Appendix I. For even more detail and an exploration of how this technique extends across application spaces, refer to [37].

2.1 Geometry Overview

The proposed core geometry resembles a pot core, but has a specific geometry with a single-layer winding and quasi-distributed gaps in the center post and outer shell (Fig. 2-1). To implement the quasi-distributed gaps, the core is composed of thin magnetically permeable discs and outer shell sections separated by small gaps. The center post and outer shell are bridged by magnetic end caps at the top and bottom of the structure. A single-layer winding is centered in the window, with evenly spaced

turns.

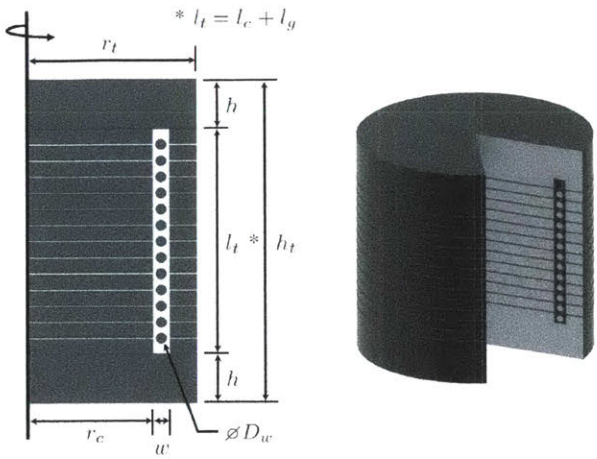


Figure 2-1: Radial cross-sectional view (left) of the proposed inductor, with a center post, outer shell, and end caps encasing a single-layer winding. Parameters defining the geometry are labelled on this view as reference for Sections 2.2 and 2.3. Revolving the cross-section about the axis of rotation produces the 3D model of the inductor on the right (a piece is cut out for clarity).

This structure uses a single-layer winding to reduce proximity-effect losses and has a permeable return path to contain the flux, increase inductance, and improve the predictability of the inductance. The quasi-distributed gaps help reduce fringing field losses while still allowing the use of a high-permeability core material. Properly designed, the structure can also conduct current through a large fraction of the winding cross-sectional area, as explained in Section 2.2.2.

2.2 Design Guidelines

The design guidelines below optimize the Q of the proposed structure for a given volume and inductance. Most of the guidelines can be mathematically defined so that initial designs can be largely automated. A few of the parameters, however, must be manually tuned using the guidelines, as would be done in a non-analytic design process.

2.2.1 Use quasi-distributed gaps to reduce gap fringing loss

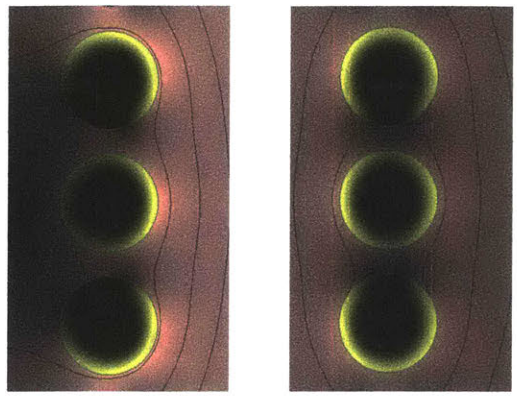
Gapping cores in high-current-swing applications is important for keeping B fields low to reduce core loss, which scales as B^β ($\beta \approx 2$–3), per the Steinmetz equation $P_v = k_c f^\alpha B^\beta$. As frequency increases, even lower B fields are needed to keep core loss low, leading to larger gaps. The impact of fringing fields from gaps on copper losses can thereby become more severe at higher frequencies. To reduce the fringing loss, the proposed inductor uses quasi-distributed gaps [26], as opposed to a conventional single lumped gap. Instead of dropping the entire MMF across one gap, the quasi-distributed gap has a smaller MMF across each of multiple gaps, causing less total loss in the winding. As shown in [26], the ratio of the pitch between the gaps (p) to the spacing between the gaps and the conductor (s) is an important parameter for fringing loss; [26] recommends $p < 4s$.[1] For the proposed structure, we set the number of gaps equal to the number of turns ($N_g = N$); Appendix A discusses how this selection, in tandem with the guidelines in Sections 2.2.4 and 2.2.5, generally meets the $p < 4s$ criterion of [26].

2.2.2 Balance H fields to achieve multi-sided conduction

For a single-layer winding, copper loss at high frequencies is primarily due to skin effect, which reduces the effective area of current flow. In most cases, only a single side of the wire has a skin depth of conduction (not the entire circumference, as is commonly shown in textbooks for a wire in isolation). This single-sided conduction occurs in typical inductor geometries because the H fields near each turn are imbalanced, causing uneven current distribution (Fig. 2-2a). To reduce copper loss, the geometry should instead be designed to balance the H fields near each turn. If the H fields on either side of a turn are balanced, double-sided conduction can be achieved (Fig. 2-2b).

The proposed structure implements double-sided conduction to achieve low copper loss. To balance the H fields in this structure, the center post and the return path need

[1] While increasing the number of gaps at lower pitch reduces fringing loss, it does so with diminishing returns and also makes construction increasingly difficult.



(a) Imbalanced H fields (b) Balanced H fields

Figure 2-2: When H fields (red) are balanced, the effective conduction area in the winding (yellow) is increased. A winding with a lower H field on one side (dark red) than the other side (light red) has only single-sided conduction (2-2a), while a winding with comparable H fields on either side has double-sided conduction (2-2b). The field imbalance/balance can also be seen in the plotted B field lines.

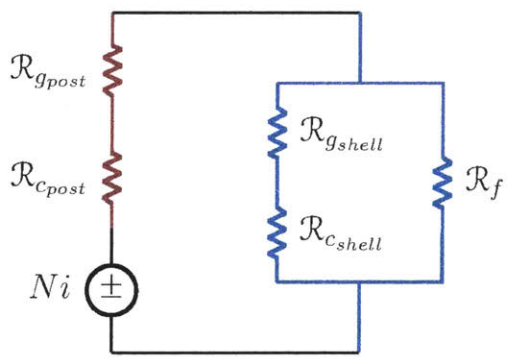


Figure 2-3: Magnetic circuit model used to balance the H fields in the proposed structure by making the reluctances of the center post (red) and the return path (blue) equal. This model includes the overall fringing field outside the structure but not the gap fringing fields. The discs of core material and the quasi-distributed gaps in the center post and the outer shell are treated as lumped reluctances. The end cap reluctances are assumed to be negligible.

to have equal reluctances (Fig. 2-3). Doing so makes the MMF drop ($\mathcal{F}$) across each region the same. Since both regions also have the same effective length (l), having equal $\mathcal{F}$ results in balanced H fields ($\mathcal{F} = Hl$).

To accurately design for equal reluctances, we include the overall fringing field outside the structure in the return path. Mathematically, we need

$$\mathcal{R}_{c_{post}} + \mathcal{R}_{g_{post}} = (\mathcal{R}_{c_{shell}} + \mathcal{R}_{g_{shell}}) \parallel \mathcal{R}_f \tag{2.1}$$

where $\mathcal{R}_{c_{post}}$ and $\mathcal{R}_{c_{shell}}$ are, respectively, the lumped reluctance of the discs of core material in the center post and in the outer shell, $\mathcal{R}_{g_{post}}$ and $\mathcal{R}_{g_{shell}}$ are, respectively, the lumped reluctance of the quasi-distributed gaps in the center post and in the outer shell, and $\mathcal{R}_f$ is the reluctance of the overall fringing path outside of the structure.

Neglecting local gap fringing, $\mathcal{R}_{c_{post}}$, $\mathcal{R}_{c_{shell}}$, $\mathcal{R}_{g_{post}}$, and $\mathcal{R}_{g_{shell}}$ can be calculated directly from the geometry (Fig. 2-1):

$$\mathcal{R}_{c_{post}} = \frac{l_c}{\mu_c \pi {r_c}^2} \tag{2.2}$$

$$\mathcal{R}_{c_{shell}} = \frac{l_c}{\mu_c \pi ({r_t}^2 - (r_c + w)^2)} \tag{2.3}$$

$$\mathcal{R}_{g_{post}} = \frac{l_g}{\mu_0 \pi {r_c}^2} \tag{2.4}$$

$$\mathcal{R}_{g_{shell}} = \frac{l_g}{\mu_0 \pi ({r_t}^2 - (r_c + w)^2)} \tag{2.5}$$

where l_c is the combined height of the core material discs, l_g is the overall length of the gap, and μ_c is the permeability of the core material.

$\mathcal{R}_f$, however, is more difficult to calculate from first principles; instead, we estimate it using a solenoid model. Since the proposed inductor and a solenoid of the same size have similar overall fringing B fields (Fig. 2-4), their fringing field reluctances are about the same. So, to estimate $\mathcal{R}_f$ of the proposed inductor, we can back out the fringing field reluctance from any appropriate solenoid inductance model. In general, for a solenoid,

$$L = \frac{N^2}{\mathcal{R}_{inside} + \mathcal{R}_f} \tag{2.6}$$

where $\mathcal{R}_{inside} = h_t/(\mu_0 \pi {r_t}^2)$ is the reluctance of the path through the center of the solenoid. By substituting a solenoid inductance model of our choosing into (2.6), we can then derive an expression for $\mathcal{R}_f$. For example, for structures where $h_t > \frac{2}{3} r_t$, the following air-core solenoid model [38] can be used:

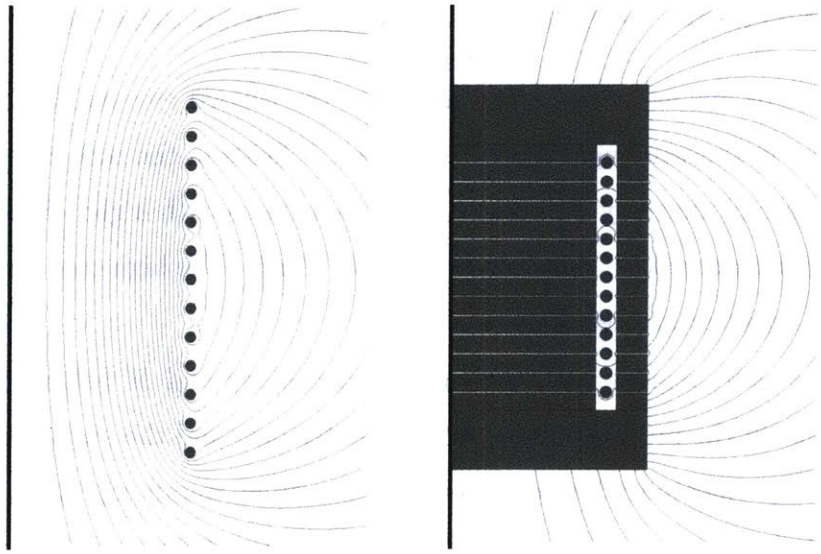


Figure 2-4: A solenoid (left) and the proposed inductor (right) have similar fringing fields, so the fringing field reluctances can be modeled as approximately equal. This approximation is then used in calculations for balancing the H fields in the proposed inductor. B field lines are shown here, though the fields outside the structure are of interest here, where B and H are always aligned.

$$L \approx \frac{\mu_0 N^2 \pi {r_t}^2}{h_t + 0.9 r_t} \tag{2.7}$$

We can then back out

$$\mathcal{R}_f \approx \frac{0.9}{\mu_0 \pi r_t} \tag{2.8}$$

For more general cases, the short solenoid model [39] may be more appropriate:

$$\widetilde{L} \approx 2FN^2 \widetilde{r_t} \tag{2.9}$$

where $\widetilde{L}$ is the inductance in µH, $\widetilde{r_t}$ is the radius of the solenoid in inches, and F is an experimentally derived quantity defined in [39]. With this model,

$$\mathcal{R}_f \approx \frac{2.54 \times 10^4}{2 r_t F} - \frac{h_t}{\mu_0 \pi {r_t}^2} \tag{2.10}$$

Using $\mathcal{R}_f$, we can then design the center post and the return path to have equal reluctances, and thus balance the H fields to achieve double-sided conduction.

2.2.3 Distribute B fields to reduce overall core loss

While H field balancing helps better distribute the carried current and reduce conduction losses in the winding, evenly distributed B fields in the core can reduce core loss. In the case of unevenly distributed B fields, regions with higher B fields experience much greater core loss, since core loss scales as B^{β}. The high core losses in these regions then result in greater total core loss.

Since $B = \mu H$, regions with the same permeability and H fields will have the same B fields. In the proposed inductor, the center post and the outer shell have the same effective permeability because they have the same overall gap and core lengths. Therefore, designing for balanced H fields in the proposed structure will also achieve evenly distributed B fields in these core regions. For cases in which the center post and the outer shell do not have the same effective permeability, the structure cannot achieve both balanced H fields and evenly distributed B fields. Instead, to minimize overall loss, the designer would need to find the optimal balance with partial double-sided conduction and a slight imbalance in the B field distribution.

For the end caps, the B field distribution, and thus core loss, is affected by their thickness. Thicker end caps allow the B field to distribute more in these regions for lower core loss, but with diminishing returns for added volume. The designer can use simulation to determine an end cap thickness that reduces loss without excessive volume.

2.2.4 Select a wire size that optimizes effective conduction area

Since the structure is designed to achieve double-sided conduction in the winding, larger diameter wire reduces copper loss by providing more circumferential conduction area. As the wire diameter increases, however, proximity effect losses between the turns play a larger role.

One metric for selecting a wire diameter (D_w) is the vertical window fill (F_v), defined as the fraction of the window height (l_t) that is occupied by conductive material,

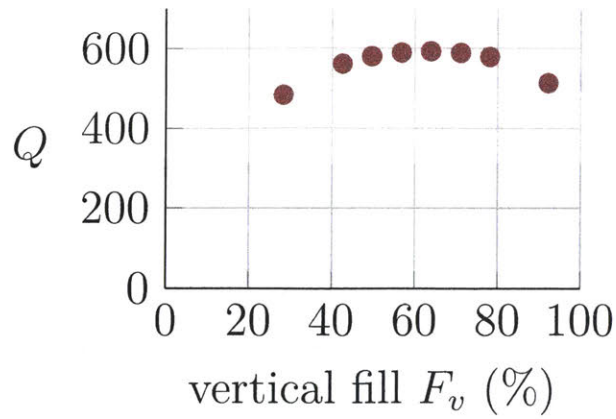


Figure 2-5: For a given window height, a wire diameter that yields a 50–80 % vertical window fill optimizes the total effective conduction area to reduce copper loss. To find this optimum, inductors with the same inductance and core geometry but different winding diameters were simulated. To make the gap fringing loss on the winding negligible, the inductors had a large window width that was 2.25 times the maximum wire diameter at $F_v = 100\%$.

i.e.

$$F_v = \frac{ND_w}{h_t - 2h} \tag{2.11}$$

using the geometry in Fig. 2-1. Finite element analysis (FEA) simulations[2] show that a wire diameter yielding a vertical window fill between 50–80 % optimizes the total effective conduction area for these two competing effects (Fig. 2-5). For a given window height, the copper loss is largely insensitive to deviations in the wire diameter near the optimum.

2.2.5 Select a window size that balances gap fringing field loss and core loss in end caps to reduce overall loss

To minimize gap fringing field loss, the structure would ideally have a large window to increase the horizontal distance between the gaps and the winding. However, since flux crowding around the ends of the window leads to higher B fields in and near

[2]All FEA simulations were run in ANSYS Maxwell, except for those in Section 2.6 which were run in Finite Element Method Magnetics (FEMM).

the end caps (Fig. 2-6), a larger window would increase core loss by increasing the volume of these high-B-field regions.

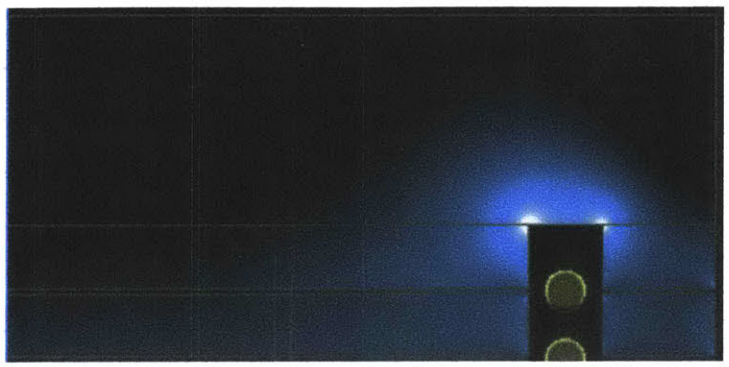


Figure 2-6: Flux crowding at the end of the window leads to higher B fields (white and light blue) and thus greater core loss.

One metric for selecting a window width (w) is the horizontal window fill (F_h), defined as the fraction of the window width that is occupied by conductive material, i.e.

$$F_h = \frac{D_w}{w} \tag{2.12}$$

using the geometry in Fig. 2-1. FEA simulations show that to balance the fringing loss and the end cap core loss, the horizontal window fill of the winding should be between 40–60 % (Fig. 2-7). So, for a given wire diameter D_w, the optimal window size is approximately $2D_w$, but the overall loss is largely insensitive to changes in the window size near the optimum.

2.2.6 Use a square aspect ratio to minimize overall loss

A "square" aspect ratio (diameter $\approx$ height) is the preferred overall geometry for this structure. FEA simulations of otherwise optimized inductors show that structures that are much wider than they are tall, or vice-versa, achieve lower Q (Fig. 2-8).

Conceptually, we can explain the disadvantages of unbalanced geometries by considering the end caps separately from the rest of the structure (everything within l_t).

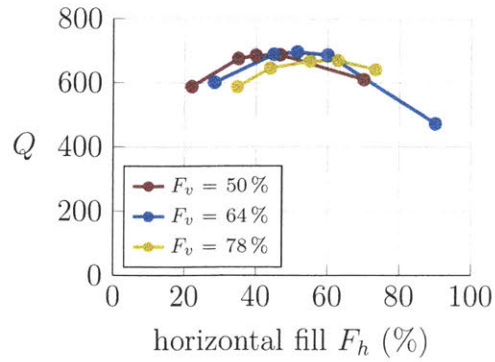


Figure 2-7: For a given wire diameter, a window size with a 40–60 % horizontal fill for the winding balances the gap fringing loss and end cap core loss. To find this balance, inductors with the same inductance and volume but different window widths were simulated. As shown in the graph, the optimal range of horizontal fill holds across the optimal vertical fill (F_v) range.

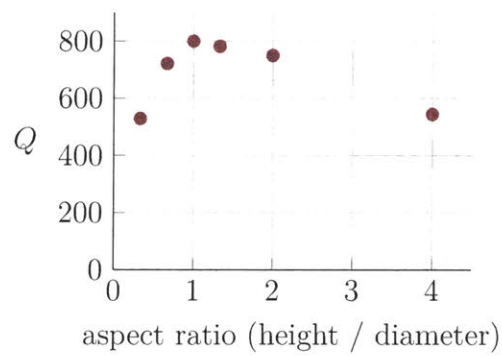


Figure 2-8: Structures with a "square" aspect ratio achieve the optimum Q. To find this optimum, inductors with different aspect ratios but the same inductance and volume were simulated, and each design was optimized using the guidelines discussed in Sections 2.2.1 to 2.2.7.

The section within l_t may be thought of as the "active" section where flux links the winding and substantial reluctance is provided, while the end caps may be thought of as overhead required to complete the magnetic path. These two sections have opposite loss dependencies on diameter: increasing diameter increases loss in the end caps by adding volume (for a fixed end cap height) but decreases loss in the active section[3]. This competing tendency explains why intermediate aspect ratios provide the best performance.

2.2.7 Approximately balance copper and core loss to reduce overall loss

As in conventional inductor designs, for a given core material, the number of turns and overall gap length in the proposed structure can be used to tune the copper and core losses. The overall loss is usually minimized at a point where core loss is close to, but slightly less than winding loss [40]. To achieve this, the designer can model the losses with exact core loss parameters or hand-tune the design in simulation.

2.3 Automating initial designs of the proposed inductor structure

Using the design guidelines discussed in Section 2.2, we can mathematically define the proposed inductor geometry. The design process can then be largely automated to generate high-Q inductor designs for a desired volume and inductance at a given frequency and current (Fig. E-2). The end cap height and the number of turns, however, must still be manually tuned. An example python script for automating the design process can be found in the related MIT M.Eng thesis [37].

[3]For a first-order derivation showing that loss in the active section decreases as diameter increases, see Appendix B.

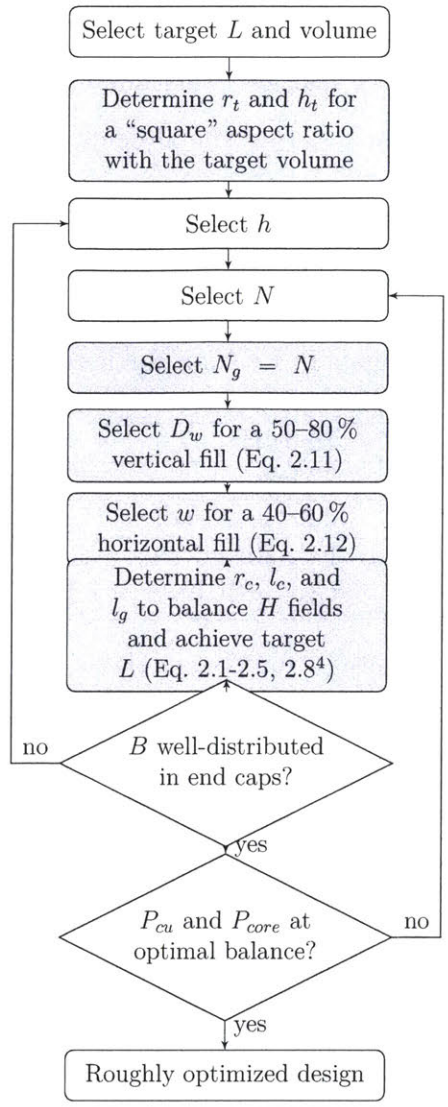


Figure 2-9: Flowchart of the design process for the proposed inductor structure using the guidelines presented in Section 2.2. The parameters used in the flowchart are labelled on the cross-sectional view in Fig. 2-1. Grey fill denotes steps that can be automated.

2.4 An Example 16.6 µH Design: Simulations

Using the guidelines in Section 2.2, we designed an example 16.6 µH inductor that achieved a Q of 700 at 3 MHz and 2 A (peak) of ac current in FEA simulation (Table 2.1). To design the example inductor, a script was used. The target inductance

[4]Eq. 2.8 may be replaced with Eq. 2.10 or any other appropriate fringing field reluctance model.

and volume as well as a selected h and N were entered into the script, which generated dimensions for the geometry that were then simulated. Afterwards, the height of the end caps was manually tuned so that the B fields were well-distributed, and the script was re-run with the optimized h. Next, designs with varying number of turns were generated using the script to find the optimum core and copper loss balance. At this point, the example design was roughly optimized. We then chose to continue with additional minor adjustments in FEA for further optimization (Table 2.2).

Inductance	16.6 µH
Frequency	3 MHz
Current	2 A (peak, ac)
Core Material	Fair-Rite 67, $\mu_r = 40$ $C_m = 34040$, $\alpha = 1.18$, $\beta = 2.24$[5]

Table 2.1: Specifications for the simulated example inductor

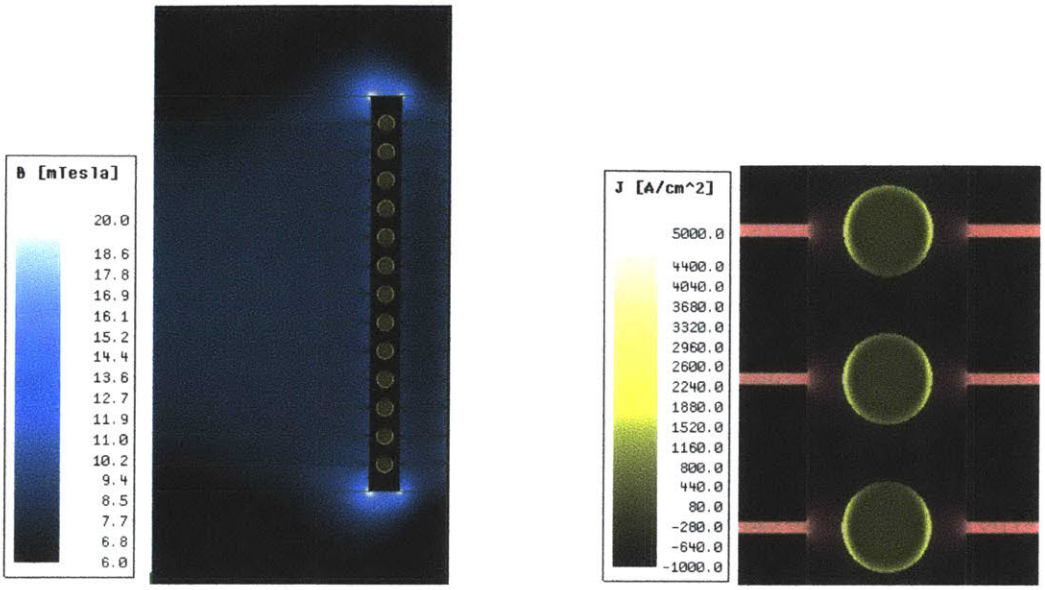
Total Diameter ($2r_t$)	26.9 mm
Centerpost Radius (r_c)	9.9 mm
Window Width (w)	1.4 mm
Total Height (h_t)	26.0 mm
End Cap Height (h)	4.0 mm
Total Core Length (l_c)	16.5 mm
Total Gap Length (l_g)	1.5 mm
Number of Turns (N)	13
Number of Gaps (N_g)	13
Wire Gauge (D_w)	20 AWG

Table 2.2: Geometry of the simulated example inductor (see Fig. 2-1)

The simulation results verified that by following the design guidelines, the example design achieved all of the desired low-loss features, and thus a roughly optimized Q. The B fields in the center post and the shell were roughly equal for low core loss (Fig. 2-10a), and most turns had balanced H fields and associated double-sided conduction for low copper loss (Fig. 2-10b). It was verified that additional thickness to the end caps would have minimal effect on loss, and that larger or smaller window

[5]Steinmetz parameters for power loss in mW/cm^3, derived in ANSYS Maxwell using core loss data for Fair-Rite 67 from [2].

sizes would increase total loss. The core and copper loss were also verified to be well balanced.



(a) Roughly even distribution of B fields (b) Turns with double-sided conduction

Figure 2-10: B field (blue), H field (red), and current distribution (yellow) simulations of the example 16.6 µH inductor verifying that it achieves the desired low-loss features by following the design guidelines in Section 2.2. These simulations are of the "worst-case" distributions for a helical winding, with each turn next to a gap. Other cross sections of the inductor would have turns in between the gaps and thus lower loss.

2.5 An Example 16.6 µH Design: Experimental Results

We constructed a prototype (Fig. 2-11) of the example inductor presented in Section 2.4.[6] The prototype inductor achieved a large-signal quality factor measurement[7] of $Q = 720$ at 3 MHz and 2 A (peak) of ac current (Table 2.3), which agrees with simulations. In addition, the prototype continued to have high Q outside of its optimized designed operating point. In this section, we demonstrate the performance of the inductor across drive level and at higher frequencies. We also show the proto-

[6]For fabrication details of the prototype inductor, see Appendix C.

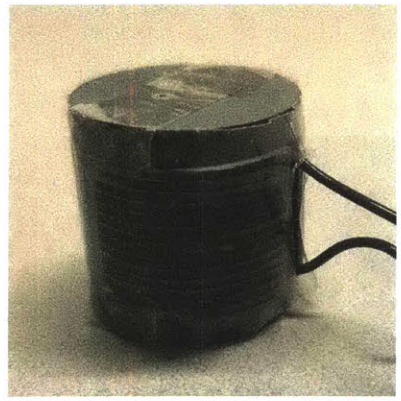


Figure 2-11: Prototype inductor of the example design (Section 2.4) having a measured Q of 720. Vertical windows in the outer shell were added to impede the circumferential component of flux and to allow the winding terminations to leave the structure.

type improving the efficiency and thermal performance of a high-current-swing power converter.

	Simulated	Prototype
Inductance	16.6 µH	13.4 µH
Q at 3 MHz, 2 A (peak, ac)	700	720

Table 2.3: The simulated example inductor and the prototype with 20 AWG wire

2.5.1 Experimental Q measurements of the prototype inductor verified simulations

The Q of the prototype inductor was measured across drive levels (0.5–3.5 A), and the experimental measurements closely matched the simulated quality factors (Fig. 2-12). This agreement experimentally verified the simulations, and the experimental Q measurements also verified that the guidelines in Section 2.2 achieve a high Q inductor.[8]

[7]For details on the large-signal Q measurement approach, see Appendix D.

[8]In some MnZn ferrite quasi-distributed designs, increased surface losses from multiple gaps have been observed [41]. For the prototype inductor, however, the agreement between the experimental and simulated quality factors indicates that any surface loss effects are minimal.

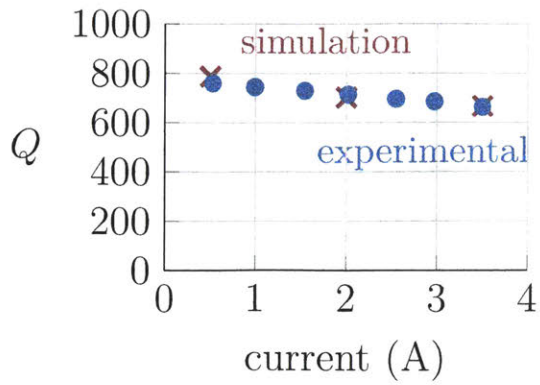


Figure 2-12: The experimental Q measurements of the prototype inductor (Fig. 2-11) closely matched the simulated quality factors, thereby verifying the simulations and demonstrating that the guidelines in Section 2.2 can achieve a high Q inductor.

2.5.2 Prototype inductor can achieve high Q at higher frequencies

The features that allow the prototype inductor to achieve high Q at 3 MHz, namely double-sided conduction and quasi-distributed gaps, continue to be beneficial at higher frequencies. In simulations at 4.5 MHz and 5.5 MHz[9], the example inductor achieved high quality factors ($Q = \sim 700$) at 2 A (peak) of ac current (Fig. 2-13). The prototype inductor also had measured quality factors of $Q = \sim 700$ at these two frequencies, demonstrating the structure's potential to achieve high Q at higher frequencies.

2.5.3 Prototype inductor improved efficiency of a high-current-swing power converter

In addition to achieving a high Q under controlled conditions, the example inductor was used in a power factor correction converter operating at dynamically varying frequencies of 1–3 MHz and with large ac current components in the inductor [17]. The inductor improved converter performance significantly (Fig. 2-14) over a more

[9]For 4.5 MHz and 5.5 MHz, the Steinmetz parameters were $k_c = 0.00163$, $\alpha = 1.37$, and $\beta = 2.21$ (for P_v in mW/cm^3, f in MHz, $\hat{B}$ in mT). The parameters were derived using core loss data for Fair-Rite 67 from [2].

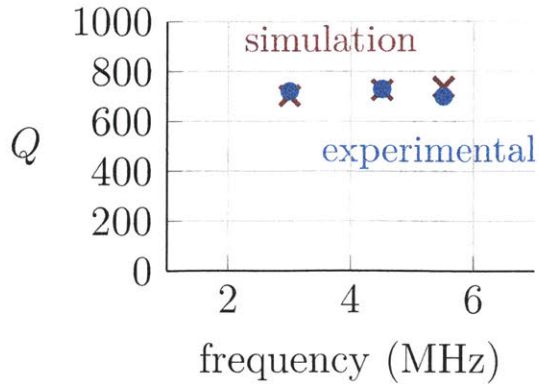


Figure 2-13: The prototype inductor (Fig. 2-11) continued to have high quality factors at frequencies higher than its designed frequency of 3 MHz. Simulations and measurements were taken at 2 A (peak) of ac current.

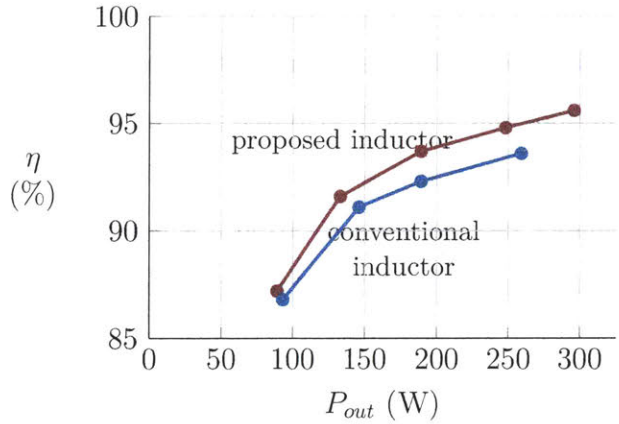
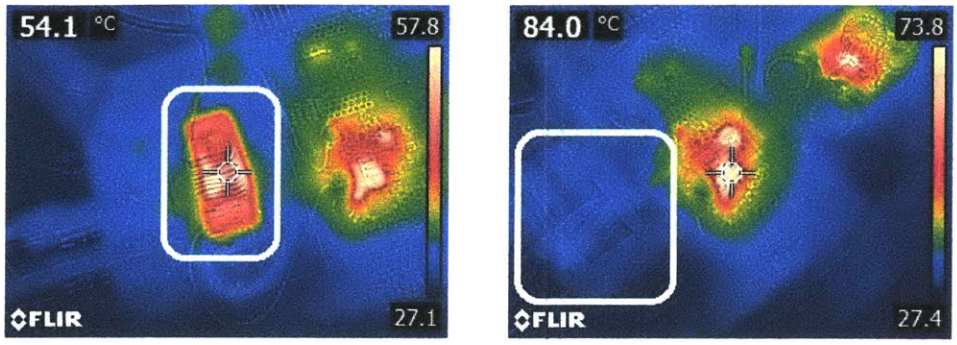


Figure 2-14: The proposed inductor improved the efficiency of a power converter operating at 1–3 MHz at different output powers, compared to a conventional inductor.

conventional open-magnetic-circuit inductor (a half toroid core with litz wire), despite having similar effective volume. This improvement can also be seen in thermal measurements: at a 93 W operating point, the conventional inductor saw a ~30 °C temperature rise, while at a much higher power (296 W), the proposed inductor only saw a ~3 °C rise (Fig. 2-15).



(a) conventional inductor
($\Delta T = \sim 30\,°\mathrm{C}$ at $P_{out} = 93\,\mathrm{W}$)

(b) proposed inductor
($\Delta T = \sim 3\,°\mathrm{C}$ at $P_{out} = 296\,\mathrm{W}$)

Figure 2-15: Thermal images showing the proposed inductor (2-15b, white box) having a much smaller temperature rise for a higher converter output power than a more conventional open-magnetic-circuit inductor (2-15a, white box).

2.6 Litz Wire in the Proposed Structure

While the example inductor in Section 2.4 can achieve low winding loss through double-sided conduction, a large fraction of the solid-core winding cross-sectional area still remains unused. In some cases, litz wire can have greater effective conduction area for improved performance in the proposed structure. For example, a litz wire version of the prototype inductor (Fig. 2-11) achieved a higher Q of 980 at the same frequency and drive level (3 MHz, 2 A (peak) of ac current). In this section, we describe design guidelines for optimizing litz wire and discuss the improved simulation and experimental results of the example inductor with litz wire.

2.6.1 Design guidelines for optimizing litz wire

As a starting point, the simple design procedure for economical litz wire presented in [42] can be used to optimize litz wire. For a given winding window, the procedure optimizes the number of strands and strand diameter for loss and cost. To estimate power loss, the ac resistance factor (F_R) is used and can be calculated by

$$F_R = \frac{R_{ac}}{R_{dc}} = 1 + \frac{(\pi n N_s)^2 d_s^6}{192 \cdot \delta^4 b^2} \tag{2.13}$$

where δ is the skin depth, b is the breadth of the winding window, N_s is the number of turns, n is the number of strands, and d_s is the strand diameter. When the strand diameter is close to or greater than the skin depth, however, (2.13) may not be accurate. Instead, the semi-empirical approach from [43] can be used to better estimate power loss.

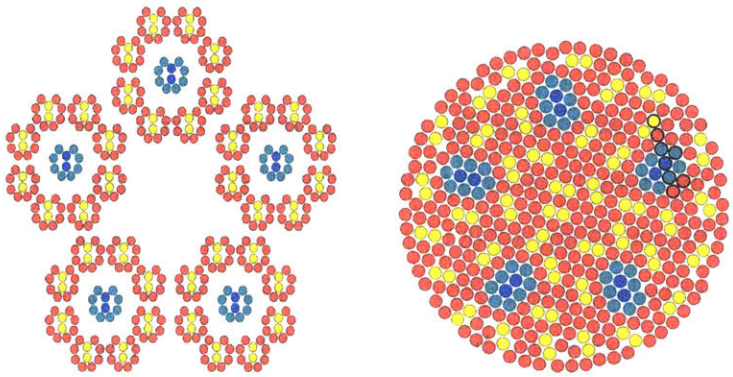
The simple litz design procedure is useful, but it is agnostic to the construction of the litz wire, which can affect performance when d_s is not much less than δ, as may frequently be the case in high-frequency designs. Litz wire is constructed from strands of individually insulated wire that are twisted together into bundles; multiple bundles may be twisted together to form a larger effective wire, and such second-level bundles may also be twisted together to increase the effective wire size further. Thus, there are many ways to construct litz wire for a given number of strands and strand diameter. Since each level of bundling may experience skin and proximity effects similar to those experienced by solid core wire [44], the choice of construction can be important. To mitigate bundle-level skin effect, [42] recommends that the number of strands in the first twisting operation should be less than

$$n_{1,max} = 4\frac{\delta^2}{d_s^2}. \tag{2.14}$$

Subsequent twisting operations should combine no more than five bundles. If for some reason these guidelines cannot be followed (e.g., using a standard litz wire design to reduce cost), bundle-level skin effect losses are no longer negligible and should be included when estimating power loss [43, 45].

In addition, when the strand diameter is close to or larger than the skin depth, the way the strands are twisted together can be important and should be included when estimating power loss. Bundles may be "bunched" together (indicated by the "/" symbol), meaning that the bundles are twisted in the same direction as the prior level bundles/strands. Alternatively, bundles may be "cabled" together (indicated by the "×" symbol), meaning that the bundles are twisted in the opposite direction. For example, the $5 \times 9 \times 10/48$ configuration in Fig. 2-16a is 10 strands of 48 AWG wire

bunched together, then 9 of those bundles cabled together, and finally 5 of those bundles cabled together. The 5/9/10/48 configuration in Fig. 2-16b has the same number of strands and bundles as $5 \times 9 \times 10/48$, but is bunched in each twisting operation rather than cabled. In this example, bunching achieves higher packing factor than cabling.



(a) $5 \times 9 \times 10/48$ (b) 5/9/10/48

Figure 2-16: Idealized cross-sections of litz wires with 450 strands using cabling (2-16a) and bunching (2-16b) twisting operations. The different colors of strands correspond to different circuit "shells" used to simulate bundle-level skin effect [43].

2.6.2 Simulations showed litz wire improving Q of prototype inductor at 3 MHz

Using the guidelines in Section 2.6.1, we investigated the effect of different litz wire designs on the performance of the example inductor (Section 2.4) at 3 MHz. For these designs, we chose strands of 48 AWG since they are a good trade-off between cost and power loss at this frequency.[10]

First, we used the simple litz wire design procedure [42] to estimate the optimal number of strands. Since the strand diameter is close to the skin depth at 3 MHz, we then used the approach from [43] to more accurately find an approximately optimal number of strands (275) and construction ($5 \times 5 \times 11/48$). We also used this

[10]Power loss could be further reduced with finer strands; however, the costs of magnet wire manufacturing and litz construction increase rapidly for strands with wire gauge greater than 44 AWG.

approach to simulate a configuration that was readily available for experimental verification (450 strands, constructed as 5/9/10/48) (Fig. 2-16b). Since the 5/9/10/48 configuration is more susceptible to bundle-level skin effect, it was simulated with bundle-level skin effect (worst case) and without it (best case). Because of random perturbations in the positions of the strands in real litz wire, some bundle-level skin effect may be mitigated, and it is expected that experimental results will fall between the worst and best cases.

Simulation results show that litz wire can provide significant improvement over solid wire for the example inductor used throughout this paper (Fig. 2-17). The approximately optimal configuration ($5 \times 5 \times 11/48$) performs slightly better[11] than the simple litz model prediction by 7.9%, due to the self shielding effect that occurs when the strand diameter is close to the skin depth [43]. The readily available 5/9/10/48 configuration under-performs the simple litz model by 6.6% when bundle-level skin effect is included.

2.6.3 Experimental Q measurements of litz wire prototype verified simulations

Using the same core geometry as the example inductor presented in Section 2.4, we constructed a prototype inductor with the readily available 5/9/10/48 litz wire. At 3 MHz and 2 A (peak) of ac current, the litz wire prototype achieved an experimental quality factor of $Q = 980$, agreeing with simulations (Table 2.4). For this operating point, litz wire provided a 36 % improvement in Q over solid-core wire. This improvement demonstrates the potential of litz wire to improve performance of the proposed structure for certain operating points.

[11] While the number of strands in the first twisting operation is higher than the recommendation from (2.14) ($n_{1,max} = 5$ at 3 MHz), it does not result in significant bundle-level skin effect in this case (a difference of 0.96% in Q).

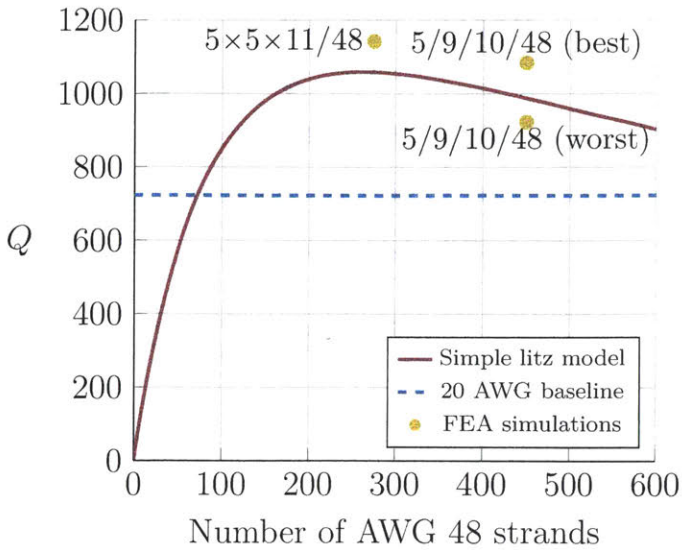


Figure 2-17: Simulated inductor Q versus number of AWG 48 litz wire strands using a simple design method (red line) and FEA simulations of specific litz configurations (yellow points) at 3 MHz and 2 A (peak) of ac current. At this operating point, the example inductor can achieve higher Q with litz wire than with 20 AWG solid wire (blue dashed line).

	Simulated (average case)	Prototype
Inductance	16.6 µH	12.6 µH
Q at 3 MHz, 2 A (peak, ac)	1000	980

Table 2.4: The simulated example inductor and the experimental prototype with 5/9/10/48 litz wire

2.6.4 Litz wire prototype can achieve high Q at high frequencies

At higher frequencies (up to 5.5 MHz), the litz wire prototype continued to achieve high Q at 2 A (peak) of ac current. However, since the litz wire in the prototype inductor was designed for 3 MHz, the performance using this particular construction (5/9/10/48) over 20 AWG wire declined at higher frequencies (Fig. 2-18). Other litz wire configurations, e.g. with fewer number of strands, could have lower high-frequency copper loss and litz wire may still be beneficial at higher frequencies [43].

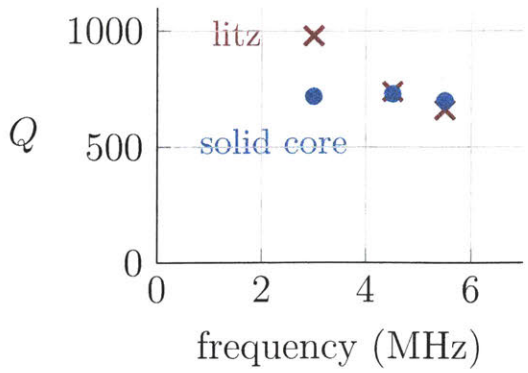


Figure 2-18: Experimental results: Using 5/9/10/48 litz wire instead of 20 AWG solid-core wire in the prototype inductor improved its Q at 3 MHz, with diminishing returns at higher frequencies.

2.7 Conclusion

Design of highly efficient, miniaturized inductors in the HF range is a significant challenge. The proposed inductor structure and design approach provide a solution for low-loss high-frequency power inductors. Using a set of analytic design guidelines, designers can achieve a roughly optimized inductor for a desired inductance and volume and then choose to further refine the design in FEA using the general design rules. This geometry and its guidelines for achieving high Q were confirmed experimentally through an example inductor with a manufactured Q of 720. In some cases, using litz wire with this geometry can also improve its performance, and a Q of 980 was demonstrated with suitable litz wire.

Chapter 3

High-Frequency Current Sensing

Sensing voltages and currents is an important element in controlling power electronic systems. At higher frequencies and powers, this can become more difficult. For example, sensing voltage often requires a voltage divider. At elevated voltages, very high impedances may be required to reduce power dissipation or impact on circuit operation. High-impedance circuits may be susceptible to noise and interference, especially if large components are used to block high voltages. Thus, it is the combination of power and frequency that presents great difficulty. This fundamental tradeoff partly explains the somewhat paradoxical observation that low-power circuits can operate at tens and even hundreds of GHz, while power electronic circuits sometimes encounter difficulties even at hundreds of kHz.

While voltage sensing does present challenges, current sensing is much more difficult. At elevated frequencies, sometimes voltage sensing alone may be sufficient, and circuits may be designed to avoid the need for high-frequency current sensing. Nevertheless, current sensing is necessary in some applications and would enable enhanced control in others.

Conventional current sensing in reality involves voltage sensing across a series impedance[1]. At higher powers, exceptionally low impedances are required (the dual of the voltage sensing problem). If we assume that acceptable measurement voltages

[1]often confusingly called a "shunt" impedance as it appears in parallel with the measurement circuit

are constant, then the power dissipated in the series resistor must scale linearly with current. When this becomes untenable, other solutions like traditional current transformers and Rogowski coils permit sensing with exceptionally low circuit impedance with acceptable sensing voltages.

These approaches work at high power. Whether they hold up at high power and high frequency is a practical and somewhat underspecified question, but one of importance for a variety of applications as suggested above. The state of the art is somewhat murky. Current sensing solutions exist. e.g. for oscilloscope probes that can sense at tens of amps and a couple hundred MHz. Other commercial products exist to measure high RF powers. Still, these are expensive, bulky, heavily engineered products – typically inappropriate to embed in a system and utilize in a control loop.

In this chapter, we explore whether current transformers and Rogowski coils can be used with sufficiently low size, complexity, and expense to be embedded for use in control. We find that, without extravagance, current transformers and Rogowski coils can be designed to sense ac currents at tens of amps with approximately 200 MHz of bandwidth.

3.1 Overview

Current transformers and Rogowski coils are conventional transformers with particular design criteria. In both cases, the design goal is not power transfer, but a combination of precision in measurement and minimal interference in the measured circuit.

Both transformers can be modeled as in Fig 3-1. In the current transformer, the magnitude of the magnetizing impedance $Z_{mag} = \omega L_{mag}$ is designed to be much larger than the load. The load or "burden" R_{sense} (typically resistive) is the dominant impedance in the path, and the sense voltage is proportional to the current. In the Rogowski coil, by contrast, Z_{mag} is designed to be much smaller than the load impedance. Indeed, the load impedance may be purely parasitic, e.g. the high input impedance of an operational amplifier.

In both models, the secondary leakage inductance affects the precision of the

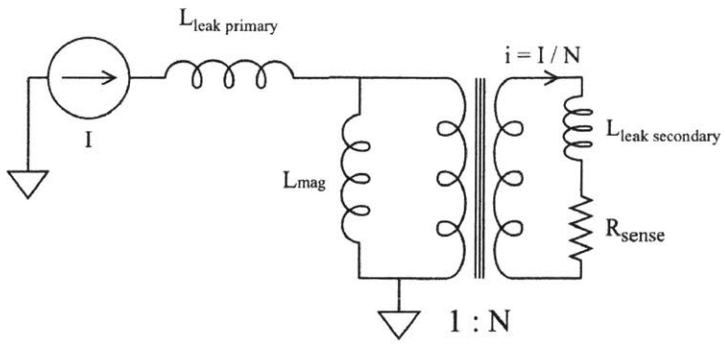


Figure 3-1: Transformer model with a resistive load. In a current transformer, $Z_{mag} \gg Z_{sense}$, the net impedance is dominated by the load and v_{sense} is proportional to I. In a Rogowski coil, $Z_{sense} \gg Z_{mag}$, the net impedance is dominated by the magnetizing inductance, and v_{sense} is proportional to the dI/dt.

results by causing the load to deviate from its expected impedance. The primary leakage appears in series with the sense current and does not affect precision, but it does increase the overall impedance inserted into the sensed circuit. As the primary is almost always a simple wire threaded through the sense structure, its leakage can be made very small and can be ignored in many cases.

3.2 Current Transformers

We designed a series of current transformers by affixing toroidal magnetic cores around a section of coaxial cable. Circuit current is passed through the center conductor of the coaxial cable; the secondary is wound around the core. The insulated coaxial shield is grounded and separates the primary and secondary windings as a Faraday shield.

We evaluated the current transformers by loading the circuit driving it with one port of a network analyzer, and measuring with the second port on the secondary. The primary circuit was terminated with 50 Ω to ensure optimal operation of the network analyzer. The secondary was sometimes loaded by the 50 Ω input imepdance of the network anlayzer; at other times a separate shunt load was used (the parallel combination constituted the net load impedance). A flat power ratio indicates proper operation; for the purposes of evaluation, we set a 0.5 dB gain variation as

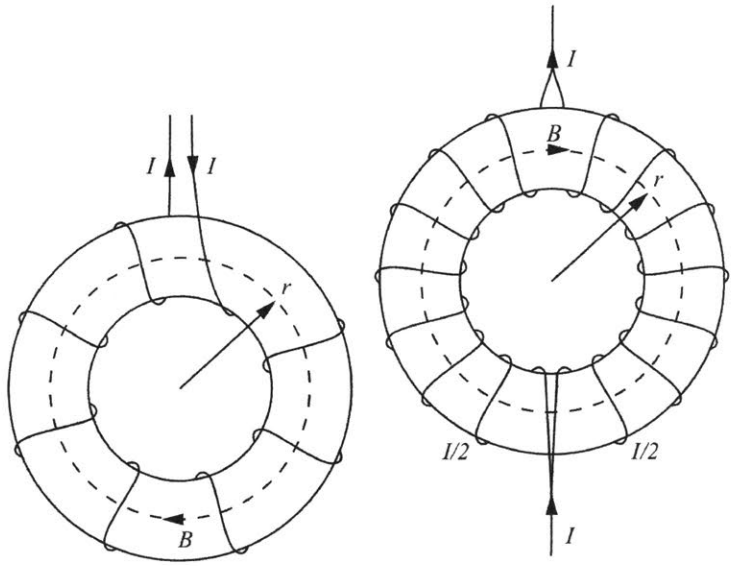


Figure 3-2: Conventional winding and "split" winding on a magnetic structure. The split winding cancels the single-turn inductance and separates the leads to reduce parasitic capacitance (image from [24]).

the "bandwidth" of the current transformer, also noting the phase.

We also experimented with both traditional winding styles and a "split" winding, as in Fig 3-2. The split winding, in principle, cancels the single-turn leakage inductance caused by circumferential components of the current. The leads are also separated, which helps reduce parasitic capacitance.

The core material selected was Fair-Rite 61 for a combination of reasonable initial permeability ($\mu_r = 125$) and high rolloff frequency.[2]

The results of a series of experiments are shown in Table 3.1. Overall, these experiments show that, without resorting to extraordinary means, current transformers can be designed with bandwidths of ~100 MHz. This bandwidth is sufficient for a great many switching applications operating below ~10 MHz and for sinusoidal rf applications at 13.56 or 27.12 MHz while still accurately accounting for a sufficient number of harmonics.

[2]Tests with Fair-Rite 67, a material with a higher rolloff frequency, exhibited worse high-frequency bandwidths. Modeling the impact of the core was not further explored, but ought to be to best understand design.

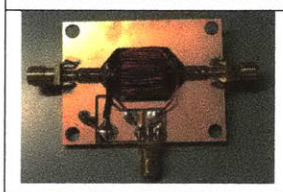
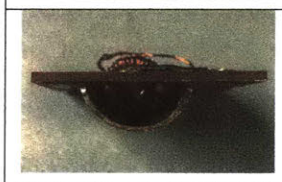
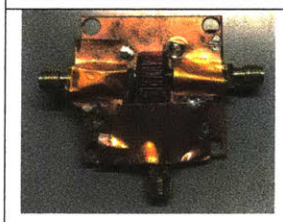
Image	Features	N_s	f_{low}	f_{high} (MHz)	f at $1\,\Omega$ impedance (MHz)
	Initial, split, 61	25	0.3	40	
	w/board, split, 61	25	0.2	88	65
	added shield, split, 61	25	0.2	88	88
	single, 31, shield	25	0.027	31	
	single, 46, shield	25	0.026	33	
	single, 67, shield	25	.736	37	82 (7.5°)
	single, 61, shield	35 w/4.3 Ω resistor		75	35
	shortened board, split, 61, extensively shielded	35		80 or 138 (12.6°)	

Table 3.1: Parameters for CTs. The core PN 2661801902 was used for the larger experiments with 61 material and the same size for the other materials. The smaller core is PN 5961001101.

3.3 Rogowski Coils

We also designed a series of "planar" PCB Rogowski coils, with an eye toward manufacturability (Fig. 3-3). These boards (layouts appear in Appendix J) were placed within an aluminum housing that was designed as a $50\,\Omega$ transmission line segment, with rf connectors on the ends (Fig. 3-4). This setup was intended to permit insertion in a typical rf system with minimal interference to the sensed signal, while providing easy-to-use connections for the main circuit and for sensing.

The initial set of board designs and results are shown in Table 3.2, in which we experimented with wide or slim traces, the density of vias to connect layers, inclusion (or not) of a return path that passes through the interior of the coil to cancel the single-turn inductance, and the position of the return loop. Each of these elements has trade-offs associated with it (esp. capacitance vs inductance trade-offs) and the ideal design is empirically determined.

Bandwidth was evaluated by driving one end of the main circuit with a network analyzer, with the other end terminated in $50\,\Omega$. The "secondary" was measured using a high-impedance probe (Agilent 41800A) for the network analyzer (Agilent 4395A). The output/input ratio is expected to be linear with frequency (following the magnetizing impedance). The upper limit of bandwidth is set by lowest-frequency parasitic resonance (unlike the Z_{mag} vs Z_{sense} rolloff in the case of the current transformer). Note that the Rogowski coils may not be able to operate near the resonant point "bandwidth" due to distortion in the output vs frequency curve, depending on the required accuracy. Nevertheless, the resonant frequency is an easily quantifiable upper bound.

From Table 3.2, we infer that wide traces and no return loop offer the best performance by reducing secondary leakage and secondary capacitance.[3] A dense set of vias is also interpreted to help by reducing secondary leakage.

Pursuing these design choices, we designed a second set of coils with additional features (Table 3.3). Here we focused on wide traces, densely packed vias, and no

[3]Capacitances between turns, with reduced voltage drops, are much less important that capacitances directly from the signal output to the signal ground.

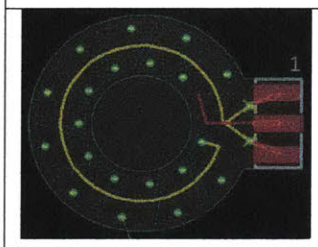
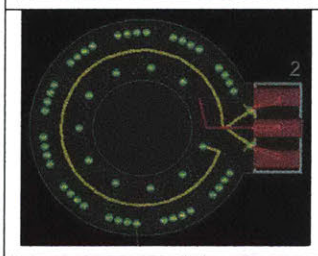
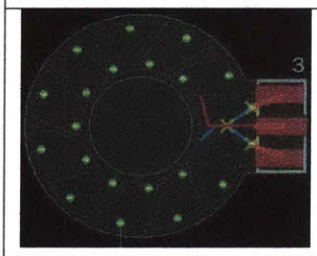
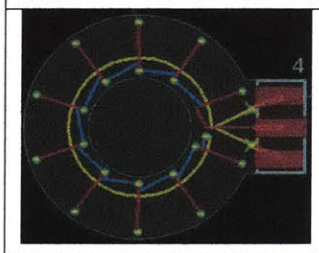
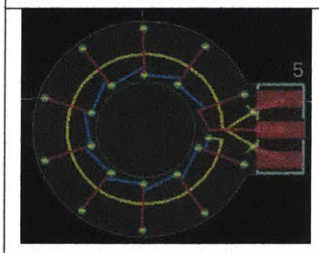
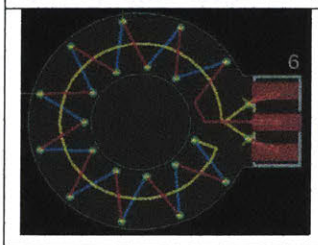
Image	Traces	Return Loop	Vias	f_{high} (MHz)
	Thick	Center	Single	178
	Thick	Center	Dense	183
	Thick	(none)	Single	223
	Thin, radial	Tight	Single	131
	Thin, radial	Center	Single	124
	Thin, sawtooth	Loose	Single	136

Table 3.2: Parameters for first-round Rogowski coils

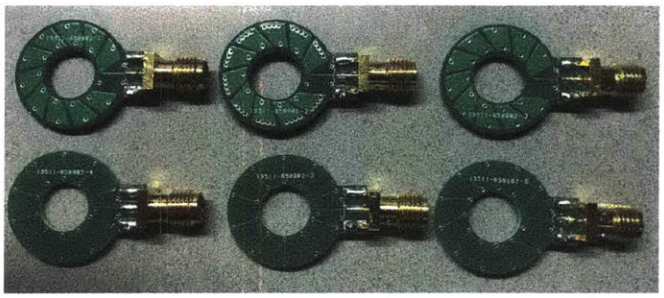


Figure 3-3: Rogowski coils realized with PCB windings.

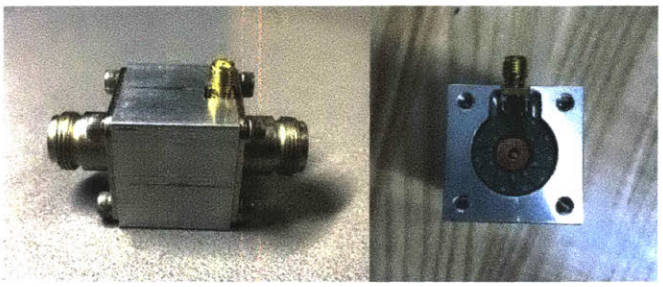


Figure 3-4: Rogowski coil housing with convenient secondary access and maintained $50\,\Omega$ characteristic impedance.

return loop, while investigating how a primary-to-secondary shield and the coil design near the connector influenced results. Several boards had shields, several did not. Some boards avoided overlap as much as possible between signal and signal ground near the connector; others permit the signal (top layer) and signal ground (bottom layer) to maximally approach each other without actually overlapping; still others filled the available surface area without regard to overlap. These choices address the tradeoff between secondary leakage and secondary capacitance.

The results from these experiments firstly confirm our previous results. "Maximizing" the prior rules resulted in a set of boards all with higher bandwidth than the previous set. In addition, we infer that the shield is helpful and that the interface with the connector is of minimal importance. Both dependencies are quite weak.

3.4 Conclusion

Many power electronics applications operate with frequencies into the MHz range. Accurately sensing such waveforms (including harmonic content) requires measure-

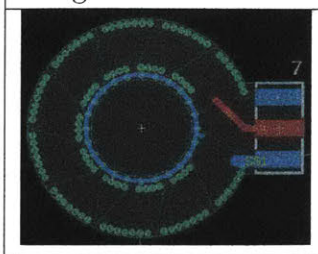
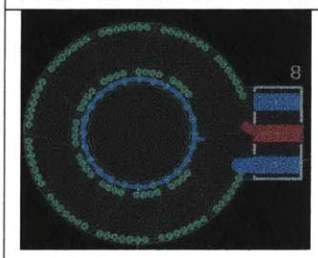
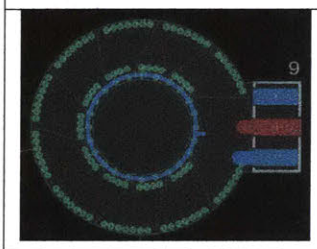
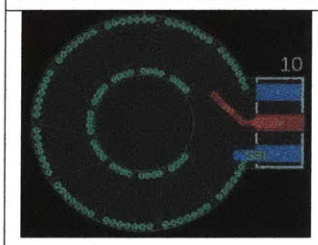
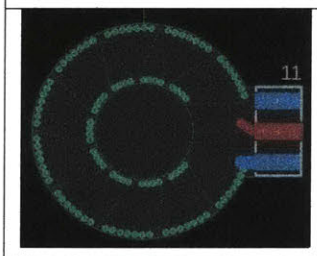
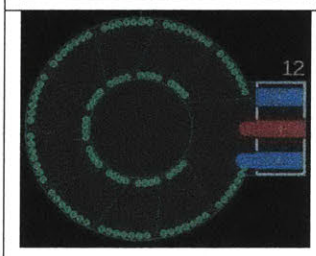
Image	Shield	Termination	f_{high} (MHz)
	Yes	Min Capacitance	247
	Yes	Near Overlap	246
	Yes	Full Overlap	240
	No	Min Capacitance	238
	No	Near Overlap	239
	No	Full Overlap	232

Table 3.3: Parameters for second-round Rogowski coils

ment bandwidths of several tens of MHz. Some high power rf applications operate at 6.78, 13.56, or 27.12 MHz, often with nearly sinusoidal operation. Accurately sensing these current waveforms, with some harmonic content, requires a similar bandwidth.

The (by no means exhaustive) experiments in this chapter show that such bandwidths are quite feasible in current transformers and Rogowski coils without exotic constructions. While neither of these solutions can sense dc currents, there are many applications where the currents are purely ac (e.g. in transformers) or where one wishes to exclude the dc component (e.g. ripple sensing).

The Rogowski coil solution in particular offers very high bandwidths in a PCB construction that may be suitable for embedding in rf transmission lines (as explored here) or for PCB integration in a switching converter. The main vulnerability of this approach lies in post-processing of the sensed voltage, which must be integrated accurately which may present difficulties both at high frequency (through parasitics and undesired amplifier behavior) and at low frequency (through accumulated offsets). Added circuitry may also present economic barriers in low-cost applications. For an initial approach to these issues, see [46, 47].

Part II

Grid-Interface Power Conversion

Chapter 4

Harmonic Injection

4.1 Introduction

Grid-interface converters (illustrated schematically in Fig. 4-1) are often required to provide power factor correction (PFC) [48, 49] wherein they draw current having strictly limited harmonic content. The ideal unity power factor case (PF = 1), which draws only sinusoidal current in phase with the grid voltage, leads to a pulsating power waveform ($\propto \sin^2$) with very large instantaneous differences from the (typically) constant load power, as illustrated in Fig. 4-2. The converter must buffer this twice-line-frequency power pulsation, and the resulting low-frequency energy storage $E_{store} = P_o/\omega_{grid}$ is necessarily very large. In this equation, E_{store} is the energy that must be buffered by the converter during a line cycle, P_o is the average (constant) output power, and ω_{grid} is the angular frequency of the ac grid (typically 2π times 50 or 60 Hz. This storage is typically achieved with electrolytic capacitors, which have low lifetime and can occupy over 50 % of PFC converter volume [50] (20–30 % of overall system volume).

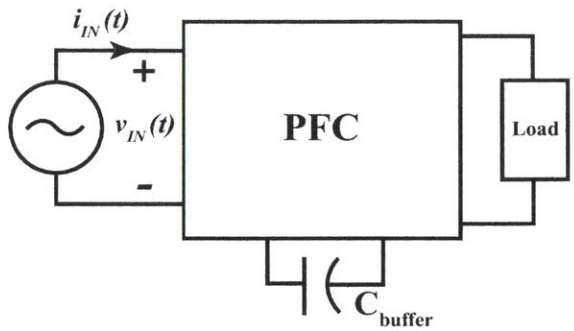


Figure 4-1: Three-terminal representation for power converter with PFC, including input from grid source, dc output to load, and ac buffer capacitor.

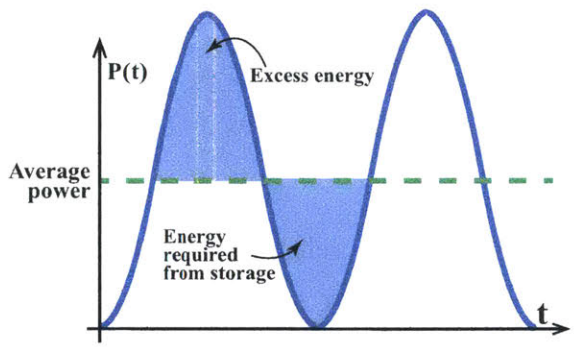


Figure 4-2: When PF = 1, power oscillates $\propto \sin^2$; integrating the difference between the input power and the output power gives the energy storage requirement over a line cycle, shown as shaded.

Energy buffer capacitors are stubbornly immune to typical miniaturization approaches when PF = 1 because the energy storage requirement is fixed by factors outside of the circuit designer's control – the power rating of the converter and the frequency of the grid. In other words, the energy storage requirement for unity power factor is not a function of efficiency, topology, architecture, or switching frequency [51].

Some research (i.e. into active buffers [52]) has observed that the usable energy storage of such a capacitor depends on both the buffer capacitance and its voltage swing:

$$E_{store} = \frac{1}{2}CV_{peak}^2 - \frac{1}{2}CV_{trough}^2 = CV_{mid}\Delta V \tag{4.1}$$

where V_{peak} is the maximum capacitor voltage, V_{trough} is the minimum capacitor voltage, V_{mid} is the arithmetic average of V_{peak} and V_{trough}, and ΔV is the arithmetic

difference between V_{peak} and V_{trough}. These works have used high voltage swings ΔV to permit lower capacitance C. Entire converters (sometimes called active buffers) have been designed to emulate a large capacitor while taking advantage of this observation [52–55]. These approaches have largely been successful at miniaturizing the energy buffer, but suffer primarily from added component counts while still buffering the same amount of energy.[1]

Here we investigate an alternative approach which fundamentally reduces the amount of twice-line-frequency energy that needs to be stored, which sets the buffer size for many applications.[2] It accomplishes this by purposefully drawing harmonic current, resulting in a more constant input power and therefore less required energy storage. While operating within line harmonic current regulations, we show that this method can substantially reduce energy storage requirements – and consequently energy buffer size – for every IEC/EN 61000-3-2 regulation class (A-D). This approach usually requires no additional hardware and can be applied to many existing PFC converters solely by a change in control.

Energy storage reduction has been explored before, mainly in the context of LED drivers which fall under Class C regulations [60–67], but this method has not thoroughly been explored in other classes which are sometimes thought to have substantially stricter regulations [68].[3] Here, in addition to extending the analysis of Class C, we show that this approach maintains substantial benefit for devices operated under Class D and even Classes A and B well into the kilowatt range.

In addition, the side effects introduced by this approach (e.g. loss, frequency vari-

[1]For a good review of techniques with reduced component count, see [56]; some techniques require no additional switching devices, but may still add energy storage components [57].

[2]Some uninterruptible applications (e.g. servers and aerospace applications [58]) impose an additional hold-up time requirement wherein the converter must maintain its output power for some duration (e.g. one line cycle) in the event of a voltage interruption. This requirement may dwarf the twice-line-frequency energy buffering requirement and such converters may be unaffected by the proposed technique. Nevertheless, the proposed approach has broad applicability in charger, adapter, appliance, and motor drive applications which have no hold-up time requirement. Note that active power decoupling techniques may have utility in fully utilizing stored energy in hold-up circumstances [59]; nevertheless, since all energy delivered to the load must come from stored energy, the fundamental requirement on stored energy cannot be affected.

[3]Exceptions include [69, 70] which only consider Class D and [71] which considers all classes but with limited harmonic inclusion and a highly specific control implementation.

ation, etc.) have not been thoroughly explored previously but are investigated here. In particular, we investigate a valley-switched boost PFC converter both theoretically and with a hardware prototype. For this implementation, we find negligible changes in loss by introducing harmonic input current. We also find a beneficial compression in the operating frequency range (from 4:1 to 1.4:1 for a given average power), which alleviates some of the challenges with using high-efficiency, variable-frequency converters like the valley-switched boost, resonant converters, etc. in PFC applications.

4.2 The ideal case: No buffer

If we first imagine our goal is to eliminate the need for an energy buffer entirely, in the absence of regulations or notions of power factor, then we would need to draw constant power from the grid, implying that the input line current must be:

$$i_{in,C=0}(t) = \frac{P_{out}}{V_{in}} \frac{1}{\sin(\omega t)} \tag{4.2}$$

where P_{out} is the dc output power of the PFC stage and V_{in} is the ac line voltage amplitude.

When drawing such a current, since there would be no instantaneous mismatch in power, the energy buffer size could be reduced by 100 % (i.e. no buffer). Undoubtedly, this is not a feasible current to draw, as it clearly violates harmonic limits (Table 4.1) and requires infinite current at zero-crossings of the grid voltage, as illustrated in Fig. 4-3. Nevertheless, we can take inspiration from this approach and analyze the harmonic content of $i_{in,C=0}$ which is composed of an infinite, equally weighted sum of all odd harmonics of the fundamental line frequency.

$$\frac{1}{\sin(\omega t)} = 2 \sum_{n \text{ odd}} \sin(n \times \omega t) \tag{4.3}$$

One interpretation of (4.3) is that intentionally drawing harmonic currents can be used to reduce the energy buffer size. While we may not achieve the full 100 % reduction in energy buffer size, we can draw a subset of current harmonics, with

weights limited by regulations, and achieve some (indeed much) of the same benefit.

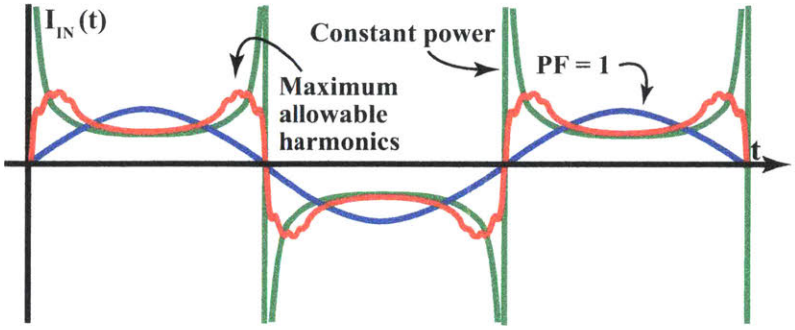


Figure 4-3: Input current waveforms for PF = 1 (blue), zero-buffer solution (green), and maximum Class D harmonic current (red). The maximum harmonic current waveform closely approximates the zero-buffer current for a large portion of line cycle.

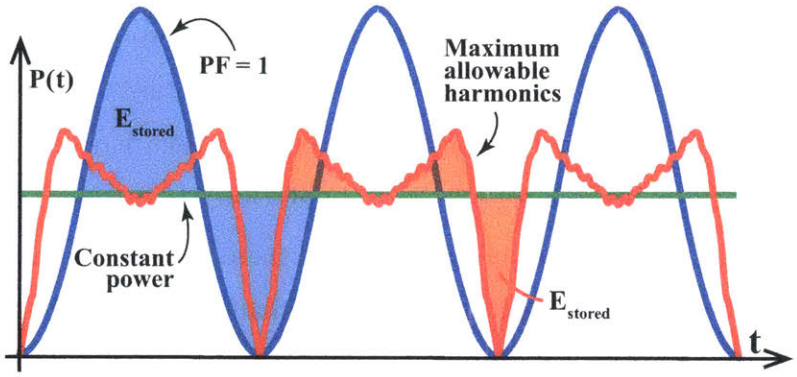


Figure 4-4: The energy storage requirement when using maximum allowable Class D harmonics (shaded area,red) is significantly decreased from the energy storage required at PF = 1 (shaded area,blue).

4.3 Operating at Regulation Limits

To appreciate the limits that regulations impose on this approach, consider the IEC/EN 61000-3-2 Class D requirements [48], which apply to devices in the 75–600 W power range, governing all odd harmonics to the 39th. These current limits are expressed in terms of device power (mA_{rms}/W), with decreasing amplitudes for higher order harmonics (Table 4.1). Beyond 600 W, most devices fall under the Class A regulation, which imposes constant limits on all odd harmonic components, independent

n-th Harmonic	Class D Limit (mA/W)	Class A Absolute Maximum (A)
3	3.4	2.30
5	1.9	1.14
7	1.0	0.77
9	0.5	0.40
11	0.35	0.33
13	$3.85/n$	0.21
$15 \leqslant n \leqslant 39$	$3.85/n$	$0.15 * 15/n$

Table 4.1: IEC/EN 61000-3-2 Class D & Class A Limits on Odd Harmonics

of device power.[4] In all of the work here, we utilize the latest 2018 edition of the EN61000-3-2 harmonic standard [48].

There are infinitely many ways to incorporate harmonic current across a many-dimensional space. To constrain the problem, we choose two approaches: first, by introducing all governed harmonics together in equal percentages p of their individual maximum allowable values; and second, by introducing each harmonic individually to its maximum before introducing the next. In all cases, harmonic content is introduced with zero phase, under the assumption that there is no benefit to be gained by introducing asymmetry to the input current/power half-waveform. These approaches are investigated numerically (see Appendix K for code).

The former method allows us to observe what happens in the most extreme case of utilizing the maximum of every regulated harmonic within the IEC/EN 61000-3-2 regulations. Let the input current be

$$i_{in}(t) = \sum_{n=1}^{39\ (\text{odds})} I_n \sin(n \times \omega t) \tag{4.4}$$

where, in Class D, each harmonic coefficient is proportional to the regulated limit $I_{reg,n}$ (mA/W) and to the output power:

$$I_n = \sqrt{2}(I_{reg,n} \times p)P_{out}. \tag{4.5}$$

[4]Class A also governs even harmonics, but systems with power electronic front ends typically have half-wave-symmetric input currents which have no even harmonics. Even harmonics are also not useful for twice-line-frequency energy storage reduction, and are not considered further.

By increasing the percentage p of all harmonics, the energy storage requirement monotonically decreases (Fig. 4-5), yielding up to a 62 % decrease in the energy storage requirement at $p = 1$. This can be seen geometrically in Fig. 4-3 where the current approximates (4.2) and also in Fig. 4-4 where the shaded energy storage area is clearly reduced.

While using the maximum allowable amount of of each harmonic current yields the largest drop in storage, it is an undeniably difficult function to generate reliably without violating regulations. Fortunately, as described below, it is still possible to benefit from the majority of these storage savings by only incorporating third and fifth harmonic terms.

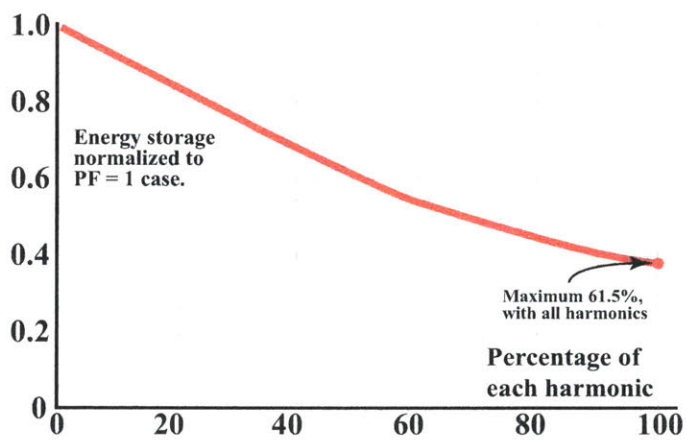


Figure 4-5: Energy storage requirement as all harmonics are included at the same percentage p of their individual allowed maxima under Class D. By including all every available harmonic, the energy storage requirement can be reduced by nearly 62 %

4.4 Incorporating Harmonics Sequentially

Instead of drawing all harmonics in equal proportion to their individual maxima, we can instead include one harmonic at a time. Let us start by drawing only third harmonic current,

$$i_{in}(t) = I_1 \sin(\omega t) + I_3 \sin(3\omega t). \tag{4.6}$$

as shown in Fig. 4-6 where I_3 is varied from 0–100 % of its allowed maximum value in class D.

With the inclusion of I_3, we see that the resulting input power begins to approximate the input power of Fig. 4-4, with reduced peak power and more constant power overall.[5] We also observe a significant impact on energy storage (Fig. 4-7), even when operating well within the allowable Class D harmonic limits. Introducing the third harmonic component alone can yield up to a 44 % improvement in the storage requirement compared to the unity power factor case, which is approximately two thirds of the maximum possible reduction under Class D.

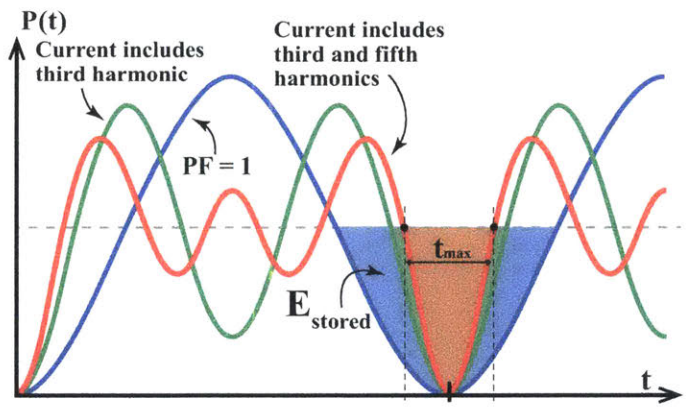


Figure 4-6: Introducing the maximum allowed third harmonic reduces the central peak (blue) and divides it into smaller peaks (green); introducing fifth harmonic further corrects the extremities (red). Shaded regions correspond to time of maximum capacitor depletion (e.g. t_{max} of line cycle using fifth harmonic), and correspond to required energy storage.

Once we have included 100 % of $I_{reg,3}$, we can further improve the result by incorporating incremental amounts of a new fifth harmonic term

$$i_{in}(t) = I_1 \sin(\omega t) + I_{3,max} \sin(3\omega t) + I_5 \sin(5\omega t). \tag{4.7}$$

The energy storage requirement continues to decrease (Fig. 4-7) by introducing increasing amounts of allowed fifth harmonic, although the additional energy savings

[5] As we increase I_3 beyond 65 % of its maximum allowable value, the input power at high voltage falls below the constant desired output. This area should not be included in the integral to calculate energy storage requirements, as the minor ΔV associated with this time does not affect the overall peak-to-peak ripple voltage on the energy buffer capacitor.

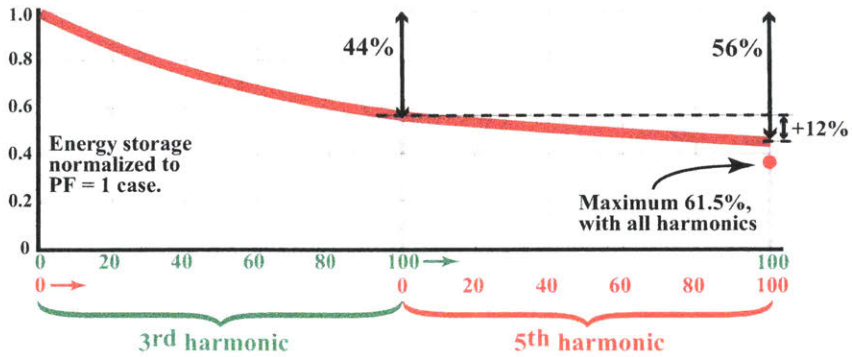


Figure 4-7: Reduction in energy storage requirement by incorporating third harmonic current up to its regulation limit, then adding fifth harmonic up to its limit. These two harmonics contribute substantially towards the maximum achievable energy storage reduction.

are much less substantial. Maximizing the fifth harmonic contributes an additional 12 % reduction to the storage requirement, significantly less than the third harmonic. The same logic applies to each successive harmonic, each having less impact on overall energy storage due to the tighter limits on higher-order harmonic currents (e.g. introducing the maximum seventh harmonic contributes an additional 4 % reduction to the storage requirement).

4.5 Impact Across Device Classes

The previous discussion was based on the Class D requirements of IEC/EN 61000-3-2, which apply to power supplies for personal computers and similar devices up to 600 W. Devices in other classes (A,B,C) must meet other requirements.

4.5.1 Class A

Devices not belonging to any other class belong to Class A. This includes a variety of device types, as well as devices rated for more than 600 W that would otherwise be considered Class D. Class A regulations define maximum permissible harmonic current values independent of power (Table 4.1) As power is increased, the allowed harmonics become smaller relative to the fundamental and we observe (Fig. 4-8) that

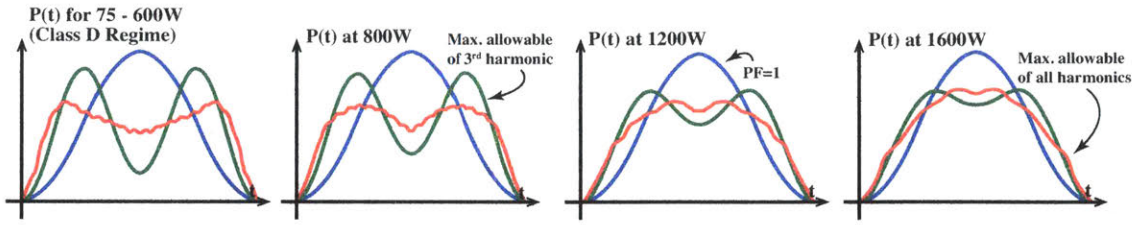


Figure 4-8: Power waveforms when including all available harmonic currents are identical across the 75 W-600 W Class D range. Beyond 600 W (in Class A), the benefit of using harmonic currents diminishes as their weight relative to the fundamental decreases. Still, this method yields up to a 35 % reduction in energy storage at 1600 W.

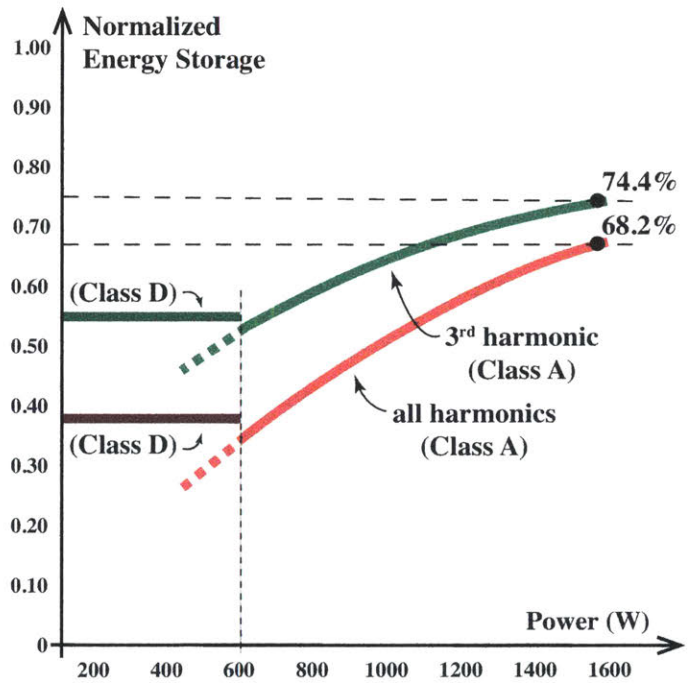


Figure 4-9: Available energy storage reduction decreases with power in Class A (with fixed harmonic maxima) as opposed to Class D (with harmonic maxima that scale with power); still, significant energy storage reduction is available even when approaching the power limits of single-phase equipment. Class D regulation limits do not seamlessly transition into their Class A maxima at 600 W, hence the discontinuity in achievable energy storage at this boundary.

the power waveform with maximum harmonic content begins to recede toward the PF = 1 shape. This is a significant departure from Class D; because Class D harmonic limits scale with power, the results are largely the same across the entire power range.[6]

[6]The results are identical for devices operating at or below 584 W. At 584 W, the higher-order 15th-39th harmonics reach the Class D absolute limits on maximum permissible harmonic current. This has negligible impact on the available energy storage savings, as high-order harmonics are

This trend obviously decreases the available benefit from harmonic inclusion at higher powers, but the benefit is still substantial well into the kilowatt range (Fig. 4-9). Indeed, at 1600 W, a roughly 35 % energy storage reduction from harmonic inclusion is still available.

4.5.2 Class B

Portable tools and some arc welding equipment belong to Class B (regardless of power), which has the same requirements as Class A with the harmonic limits multiplied by 1.5. The normalized energy storage by using the maximum[7] available harmonic content is shown in Fig. 4-10. Using the third harmonic alone, a maximum of 50 % energy storage reduction is possible at about 750 W. Below this power, the third harmonic limit is higher than the fundamental, and using a higher magnitude would only increase the required energy storage again. (We again remind the reader that the code for these calculations is found in Appendix K).

When all harmonics are used, the energy storage reduction continues to scale as power is decreased. As power approaches zero, the fundamental becomes less than every harmonic limit and it becomes possible (in theory) to replicate (4.3) and achieve complete elimination of the energy storage requirement. Nevertheless, reducing the normalized energy storage requirement below ~20 % requires a very large number of harmonics, making it practically unfeasible. Nevertheless, for portable tools of moderate power (400–800 W), it is both feasible and permissible to reduce the energy storage requirement by roughly two thirds from the PF = 1 case.

While IEC/EN 61000-3-2 is usually the relevant regulation, not power factor, it may still be valuable to examine the power factor when harmonic injection is used. The results for Class A and Class B (under the same cases as Figs. 4-9,4-10) are shown in Fig. 4-11. For those applications requiring power factor above a certain value, refer to Section 4.6.

already tightly regulated.

[7]Due to the constant limits in both Class A and Class B, at low power some harmonic limits may exceed the fundamental current. In these cases, the magnitude of those harmonics are set equal to the fundamental to minimize the energy storage requirement.

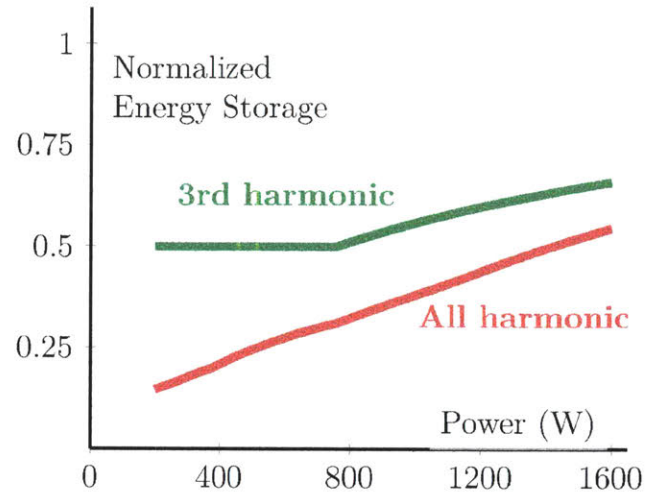


Figure 4-10: Energy storage normalized to PF = 1 conditions when using the maximum allowed third harmonic and the maximum allowed of all harmonics for Class B. For a given power, if an allowed harmonic limit is more than the fundamental, that harmonic current is set equal to the fundamental.

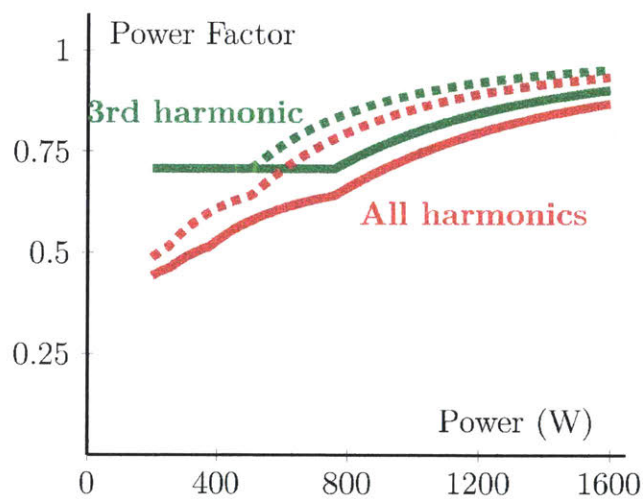


Figure 4-11: Power factor for Class A (dotted) and Class B (solid) when the maximum allowable harmonic content is used across power levels. When a harmonic is allowed to be greater than the fundamental, the magnitude of that harmonic is set equal to the fundamental for this calculation.

4.5.3 Class C

Lighting equipment exclusively falls under Class C. Most of the past research on using harmonics for reduced energy storage requirements has targeted this class in an effort to eliminate electrolytic capacitors from LED drivers to extend their lifetime.

Nevertheless, much of this research has examined specific designs and control strategies or uses incomplete or outdated limits to investigate the available energy storage reduction; therefore, we investigate the general limits of this technique on Class C here.

Class C is divided into a higher power (> 25 W) regime and a lower power ($\leqslant 25$ W) regime. In the higher power regime, harmonic limits are set as a percentage of the fundamental current (Table 4.2). Therefore, as in Class D, the available energy storage reduction is not a function of power in this regime. In addition, the allowed third harmonic is a function of the circuit power factor. To investigate the limit of the available energy storage reduction, we consider the fifth and seventh harmonics as percentages of their individual allowed maxima. For a given combination of fifth and seventh harmonic content, we calculate the maximum third harmonic content based on the power factor constraint:

$$\mathrm{PF} = \sqrt{\frac{I_{1,rms}^2}{\sum_{n=1}^{\infty} I_{n,rms}^2}} = \sqrt{\frac{1}{1 + p_3^2 + \ldots}} \tag{4.8}$$

where p_n is the nth harmonic content as a percentage of the fundamental (expressed as a decimal). With third, fifth, and seventh harmonics included and setting $PF = p_3/0.3$ as the specification requires, solving for p_3 yields

$$p_3^2 = \frac{-\left(1 + p_5^2 + p_7^2\right) + \sqrt{\left(1 + p_5^2 + p_7^2\right)^2 + 4 \times 0.3^2}}{2} \tag{4.9}$$

For a given combination of fifth and seventh harmonic, the maximum allowed third harmonic was calculated and the energy storage requirement was calculated with these three harmonics included (Fig. 4-12). Due to the tight limits on the fifth and seventh harmonics, they have relatively little impact on the result; higher order harmonics would have a smaller impact still. In addition, because the available third harmonic content is a function of the power factor, including larger quantities of higher order harmonics is not always desirable; in Fig. 4-12, the energy storage requirement

n-th Harmonic	$> 25\,\mathrm{W}$ limits (% of fundamental)
3	$30 \times \mathrm{PF}$
5	10
7	7
9	5
$11 \leqslant n \leqslant 39$	3

Table 4.2: IEC/EN 61000-3-2 Class C (>25 W) Limits on Odd Harmonics

improves and then worsens as seventh harmonic is increased. Overall, the energy storage requirement can be reduced by approximately 25 % (from the PF = 1 case) in this higher power regime of Class C.

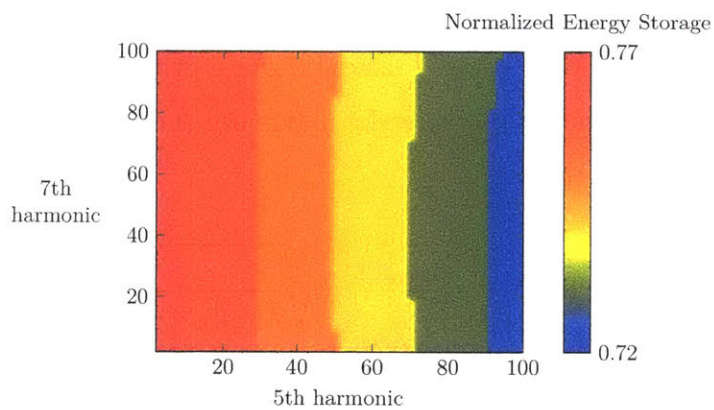


Figure 4-12: Energy storage for the high power class C case using max 5th and 7th harmonics (as percentages of their fixed maxima) and allocating the maximum allowable 3rd harmonic that fits the PF spec. Power factor is approximately 0.95 and, since very little 5th and 7th harmonics are allowed, does not vary much across these variables and is not plotted.

In the lower power regime ($\leqslant 25\,\mathrm{W}$), there are three separate options to satisfy the IEC/EN 61000-3-2 regulation:

1. Harmonics may meet Class D requirements;

2. Harmonics may meet $p_3 < 0.86$, $p_5 < 0.61$, as long as the current rises before a line angle of 60°, peaks before 65°, and returns to zero after 90°;

3. Harmonics may meet $p_3 < 0.35$, $p_5 < 0.25$, $p_7 < 0.3$, $p_9 < 0.2$, $p_{11} < 0.2$, and $p_2 < 0.05$, as long as the total harmonic distortion remains below 70 %. This

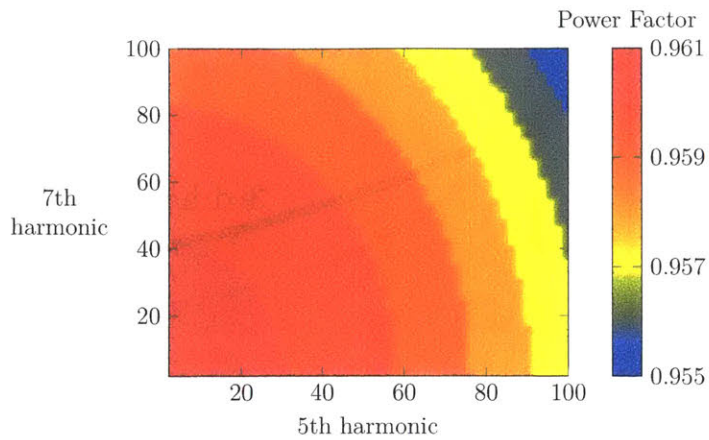


Figure 4-13: Power factor for the high power class C case using given amounts of fifth and seventh harmonics (as percentages of their individually allowed maxima) and using the maximum allowable third harmonic consistent with the IEC/EN 61000-3-2 power factor specification for this class.

option is new in the fifth edition (2018) of the IEC 61000-3-2 requirements [48].

We have already covered the first option (Class D), which permits a 62 % reduction in energy storage requirements. We also need not consider the third option, as it is less permissive than the second for controlled waveforms like the ones considered here. Therefore, we need only consider the second option.

To consider the limit of energy storage reduction in this case, we first use the maximum of the third and fifth harmonics (Fig. 4-14); adding more third and/or fifth harmonic at no point raises the energy storage requirement, so we proceed considering the maximum usage. We then include seventh and ninth harmonics, each up to a maximum of 100 % of the fundamental (Fig. 4-16). As in the higher power regime, we see that additional harmonic content does not always reduce the energy storage requirement; however, in this case, appropriately adding higher order harmonic content can reduce the energy storage requirement substantially (from ~37 % of the PF = 1 case with only third and fifth to ~26 % with seventh and ninth also included). Thus, in this regime, the second regulation option offers even more energy storage reduction than the first option (i.e. Class D limits).

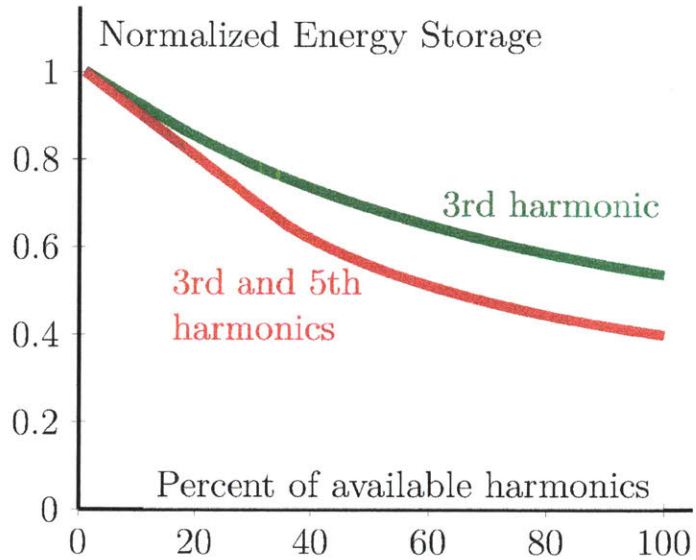


Figure 4-14: Energy storage for the low power class C case (second option) using only the third harmonic and using the third and fifth harmonics. As including more harmonic content never raises the energy storage requirement, one can maximize the third and fifth and consider further harmonics (Fig. 4-16).

4.6 Limited Power Factor and Energy Star

In some cases, designers may be constrained to operate above a certain power factor limitation. Although we know of no case where this is required by regulation for the classes of devices considered here, it is required for voluntary Energy Star compliance in the United States and may be an effective industrial standard in other sectors.

As an example, we may consider Energy Star compliance, which corresponds to a power factor of 0.9 for most kinds of equipment [60]. We investigate the energy storage requirement as third harmonic is added until power factor is reduced to 0.9. Higher order harmonics are not included as they impact power factor at the same rate as the third harmonic but provide less energy storage reduction. Power factor can be expressed as

$$\mathrm{PF} = \frac{I_{1,rms}}{\sqrt{I_{1,rms}^2 + I_{3,rms}^2}} \tag{4.10}$$

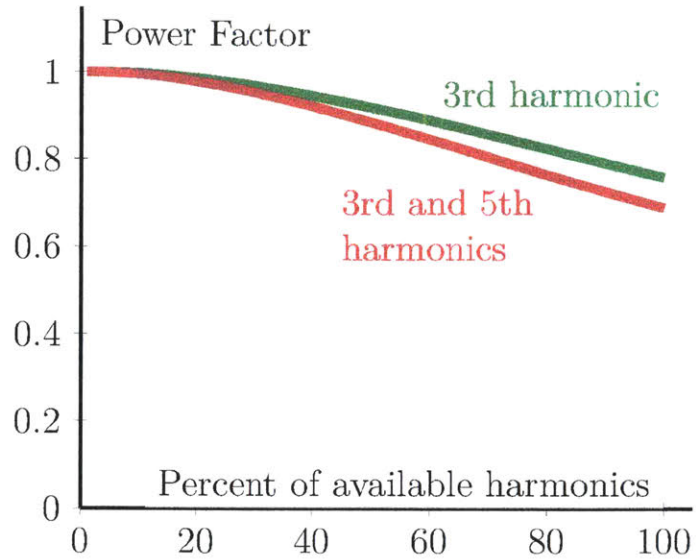


Figure 4-15: Power factor reduction when considering third and fifth harmonic inclusion. If a power factor above a certain quantity is desired, compare with Fig. 4-14 to determine the corresponding energy storage reduction.

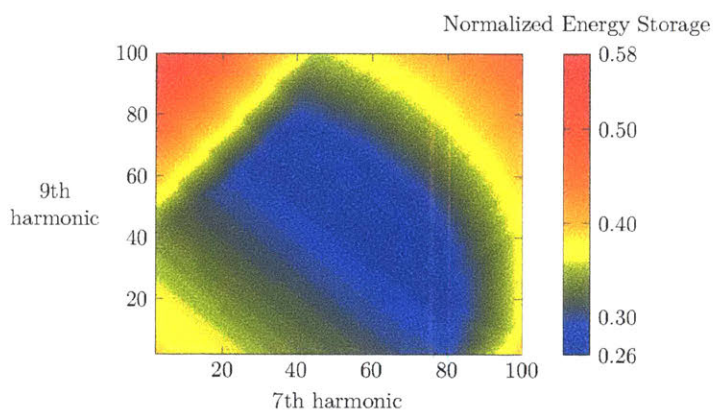


Figure 4-16: Energy storage for the low power class C case (second option) using the maximum allowable third and fifth harmonics and allocating seventh and ninth harmonic as percentages of the fundamental (seventh and ninth harmonics are not regulated directly in this option).

which may be solved for $p_3 = I_{3,rms}/I_{1,rms}$

$$p_3 = \frac{I_{3,rms}}{I_{1,rms}} = \sqrt{\left(\frac{1}{\mathrm{PF}^2} - 1\right)^2} = 0.484 \tag{4.11}$$

which corresponds to an energy storage of 65.7 % compared to the PF = 1 case, or

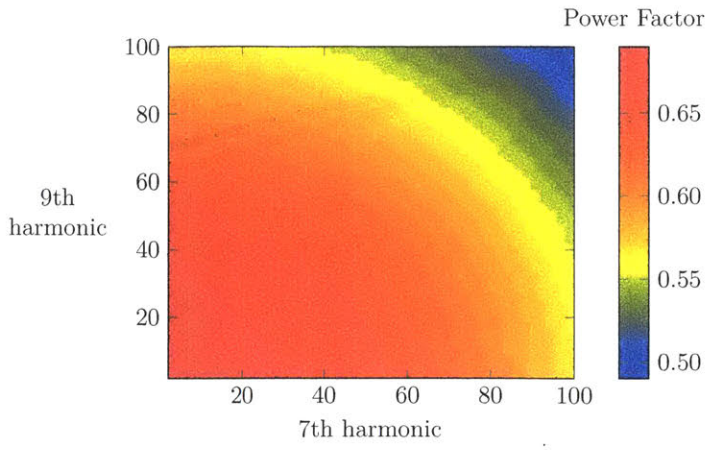


Figure 4-17: Power factor for the low power class C case (second option) using the maximum allowable third and fifth harmonics and allocating seventh and ninth harmonics as percentages of the fundamental.

approximately 35 % savings on the energy buffer size.

By comparing with Figs. 4-11,4-13,4-15,4-17, it can be seen that this particular choice of harmonic content is already allowed by IEC/EN 61000-3-2 for Class A, Class B, Class C (low power) and Class D. The only exception is Class C above 25 W, which has stringent enough standards within IEC/EN 61000-3-2 that PF $>$ 0.9 is already guaranteed, and a maximum of approximately 25 % energy storage reduction is available.

4.7 Impact on Losses

Although reducing energy buffer size can be an important gain for power density, the increased current drawn is not necessarily free (e.g. in terms of loss) and the side effects of using harmonic current have not been thoroughly explored in the literature. Since this approach can be applied independent of the converter topology, one cannot quantify the exact impacts on system loss without considering detailed design, but we can attempt to model which converter components or stages will be affected and how.

Adding harmonic content increases the rms and average rectified current at the input, when compared to the PF = 1 case. Resistive losses will grow $\propto i_{rms}^2$, while

diode losses are approximately proportional to their average currents. Adding harmonics will increase both of these metrics without increasing output power, lowering efficiency.

Nevertheless, not all components are affected equally, or at all, and loss reductions may also accrue in some cases.[8] As an example, consider a two-stage architecture with an input diode bridge, dc-side EMI filter, boost PFC stage, energy buffer capacitor holding approximately constant voltage, and a subsequent isolated dc/dc step-down stage, as in Fig. 4-18.

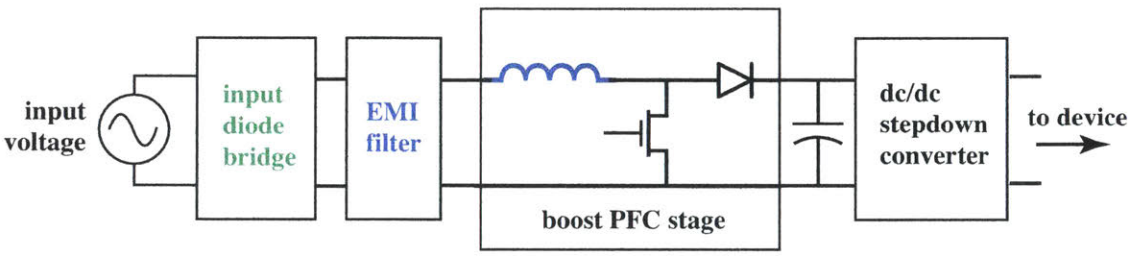


Figure 4-18: The two-stage converter with boost PFC is a very popular grid-interface architecture. For this example, incorporating input current harmonics may negatively impact losses in the diode bridge, EMI filter, and boost inductor, should not affect the conduction losses in the boost diode or dc/dc step-down converter, and may improve losses in the buffer capacitor and boost switch.

By drawing additional harmonic current at the input, the diode bridge and EMI filter will see increased average and rms currents, increasing their loss. These losses extend to the boost inductor of the PFC, but not to all PFC stage components. Since the PFC output voltage is approximately constant in this example, the PCF output current tracks the power waveform in Fig. 4-4 which has the same average value regardless of harmonic content. Since $i_{D,ave} = i_{out,ave}$, it can be reasonably argued that the boost diode conduction losses should be largely unaffected by drawing harmonic input current. Additionally, the output current actually has a lower rms value when the input harmonics are included and the boost switch conduction losses may even improve (although they remain also functions of duty cycle). The energy buffer capacitor sees reduced rms currents and therefore reduced esr losses. Even

[8]For example, switching frequency range compression may be achieved which can be used to reduce skin/proximity effect losses, core losses, and frequency-dependent semiconductor losses like dynamic R_{on} and losses in C_{oss} capacitance [12, 72, 73].

if capacitance is reduced to maintain the same voltage ripple (and therefore esr is increased), the loss $P_{esr} = I_{C,rms}^2 R_{esr}$ is still reduced. Finally, downstream elements (in this example, the dc/dc step-down stage) should be entirely unaffected by the inclusion of input harmonics. Thus, only "input facing" components see additional losses by introducing input harmonic content.

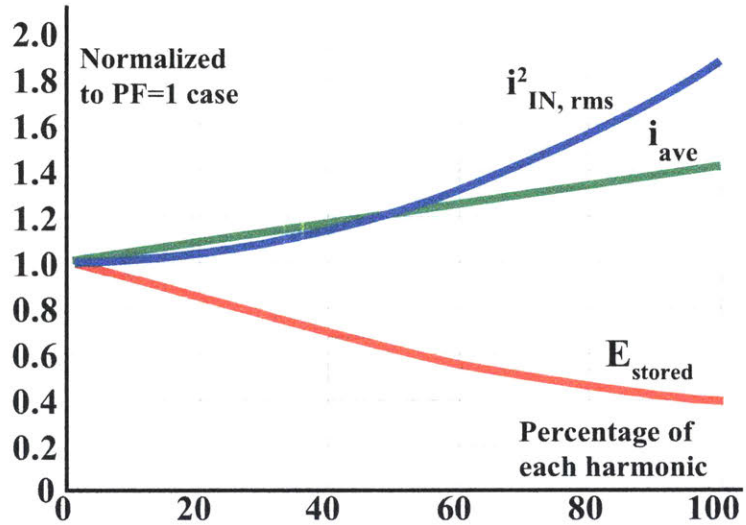


Figure 4-19: Increases in $i_{in,ave}$ and $i_{in,rms}$ for a given amount of harmonic currents, each at equal percentages of their Class D limits.

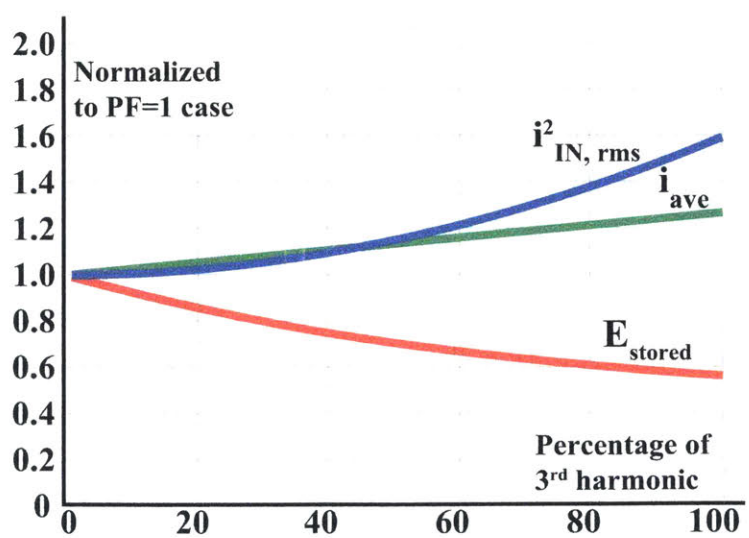


Figure 4-20: Increases in $i_{in,ave}$ and $i_{in,rms}$ for a given amount of third harmonic current (as a percentage of its Class D limit).

We can begin to model the increased losses in affected components by examining the mean-square and average rectified input currents when utilizing all harmonic currents together (Fig. 4-19), subject to Class D regulations. Logically, the largest mean-square and average rectified input currents correspond to the largest harmonic currents. The same pattern is observed when only the third harmonic is included (Fig. 4-20). While currents and associated losses do increase, they may be a small fraction of overall loss. In addition, because losses and energy storage do not vary linearly, effective compromises are available. For example, incorporating 40 % of the third harmonic alone grants a nearly 30 % decrease in energy storage (in Class D) with a very small impact on the rms and average rectified input current metrics.

4.8 Hardware Validation

Many PFC implementations can draw input currents with specified harmonics. Indeed, one benefit of this approach is its versatility across topologies without requiring additional hardware. Nevertheless, as a concrete example, we implemented a valley-switched boost PFC (Table 4.3) which serves to demonstrate experimentally the claimed performance benefits of using harmonic injection and investigate other practical effects (see Appendices L, M, and N for schematics, layout, and microcontroller code). While implementation techniques for harmonic injection are not the focus of this work, for completeness we do include a brief overview of the control used here in Appendix H which is taken from [74].

The converter was operated at constant power and adjustable harmonic content, with third and fifth harmonics included up to the same percentage p of their individual allowed Class D maxima. Fig. 4-21 shows a series of oscilloscope captures for the specifications in Table 4.3 where p is increased and the peak-to-peak amplitude of the output voltage ripple decreases (recall from (4.1) that, for constant average bus voltage, energy storage is directly proportional to voltage ripple ΔV). The measured output voltage ripples are plotted Fig. 4-22, normalized to the ripple expected in PF = 1 conditions. The calculated reduction in energy storage is also plotted, and

$V_{in,rms}$	220 V
$V_{out,ave}$	400 V
Power	250 W
Efficiency	96 % (see Fig. 4-25)
Boost Inductance	116 µH[a]
Buffer Capacitors	10 µF × 10
Buffer Capacitor PN	Nichion UCY2H100MHD1TO
Boost Diode PN	C3D1P7060Q (SiC)
Boost FET PN	GS66506T (GaN)

Table 4.3: Prototype Details for All Experiments

[a]The inductor was a PQ26/20 core of 3C95 material with 0.05 inches of gap on every leg and wound with 22 turns of 450/46 litz wire.

matches to within measurement precision.

The capacitor size is limited by the allowed output voltage ripple, so any decrease in voltage ripple for a specific power can also be interpreted as an available reduction in bus capacitance. Therefore, with modest amounts of third and fifth harmonics alone, the bus capacitor can be reduced by upwards of 50 %. This is verified in Figs. 4-23-4-24, where the converter is operated with output capacitance $C = 100\,\mu\text{F}$ and low harmonic content, and also with $C/2 = 50\,\mu\text{F}$ output capacitance and high harmonic content. It can be seen that the reduced voltage ripple from Fig. 4-22 can be translated into a capacitance reduction instead and that the impact on system volume is substantial (in this example, about a 1/3 reduction in PFC volume).

We also measured system losses for varying amounts of harmonic currents,[9] plotted in Fig. 4-25. When introducing up to 70% of maximum allowable amounts of third and fifth harmonic currents, entire system losses across the input diode bridge, EMI filter, and PFC stage remained well within 10 % of the losses otherwise incurred by operating at perfect power factor. This is likely due to the converter being heavily dominated by conduction losses in the boost diode which is not expected to change with harmonic inclusion. This is verified thermally in Fig. 4-26.

Additionally, incorporating harmonic content introduces new benefits to the converter's switching frequency. Fig. 4-28 shows the measured converter switching fre-

[9]When measuring efficiency with input harmonics, it is important to remember that the real power into the system with no phase shift is $I_{1,rms} \times V_{rms}$, not $I_{rms} \times V_{rms}$.

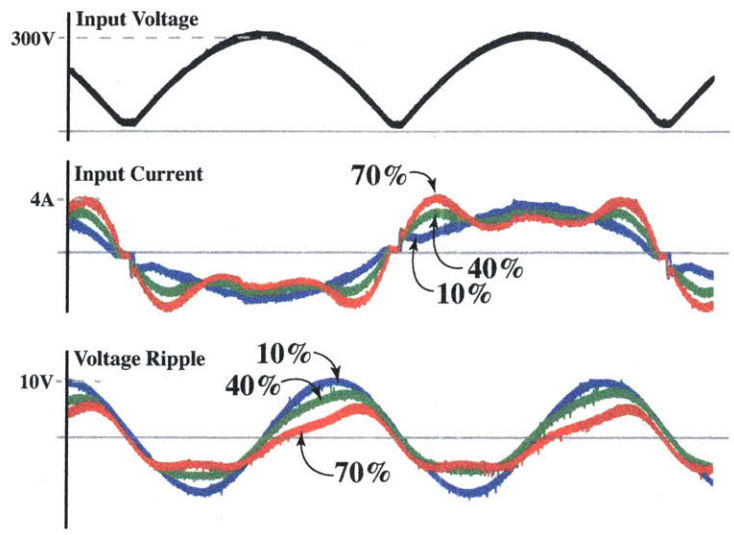


Figure 4-21: Experimental input voltage, input current, and output voltage ripple for 10 % (blue), 40 % (green), and 70 % (red) of the allowed 3rd and 5th harmonic. The output voltage ripple decreases for fixed capacitance, as expected; the original voltage ripple magnitude could be restored with less capacitance and improved power density. Power supplied from a Agilent 6813B set to 220 V_{rms} input, with a BK 8522 electronic load set to 200 Ω resistance.

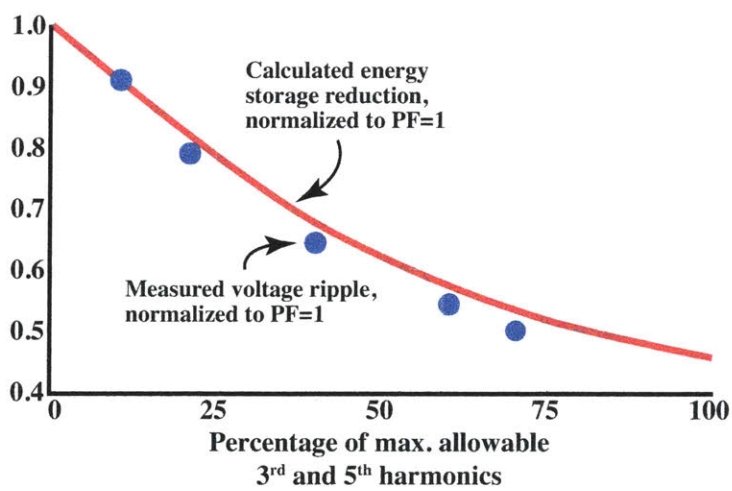


Figure 4-22: Experimental output voltage ripple, normalized to the PF = 1 case, showing a close match to theory.

quency, across the rising half of each line half-cycle for different amounts of harmonic input current. In sinusoidal current (PF ≈ 1) operation, the switching frequency of the example boost PFC varies from 200 kHz near the peak of the line to almost 800 kHz at low voltages. When harmonics are introduced, more current is drawn at

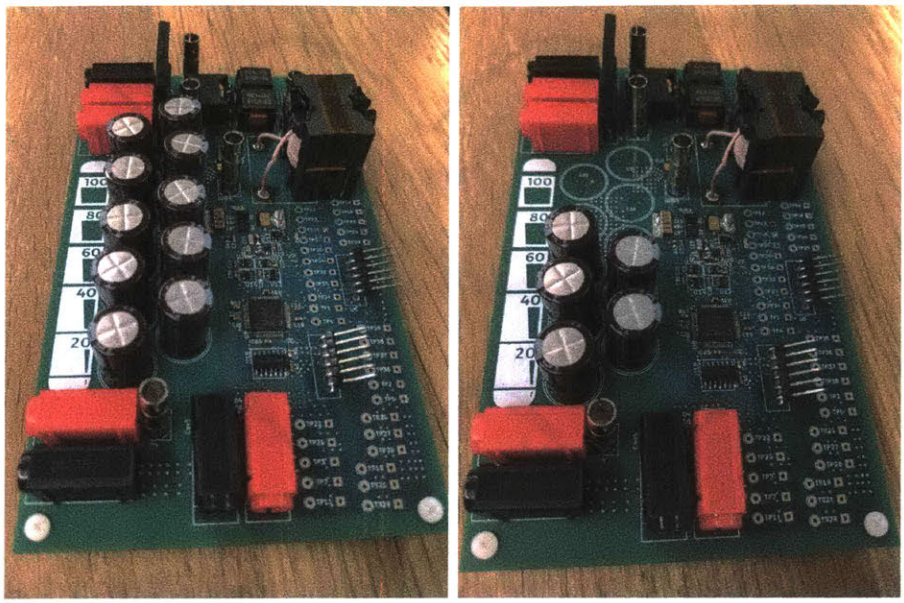


Figure 4-23: Photograph of prototype PFC showing the available buffer size reduction when introducing 70 % of third and fifth harmonic Class D limits with constant output ripple. The capacitor reduction matches theory and is a major improvement to the system power density.

low line which reduces the switching frequency (this will generally hold for most variable-frequency converters). Indeed, when the example converter operates with approximately 50 % of the third and fifth harmonics allowed in Class D, the switching frequency range is reduced to 250–300 kHz, or a ratio of 1.4:1. This compression has a variety of benefits, including for EMI filter and magnetic component design and for avoiding dynamic R_{on} and C_{oss} loss penalties. Indeed, by suppressing the highest operating frequencies, the inclusion of harmonics may improve the loss in the boost inductor, which may contribute to the flat loss characteristic in Fig. 4-25.

Overall, the prototype demonstrates many of the benefits (and costs) of purposefully drawing higher order harmonic currents discussed earlier. While drawing many harmonics offers the greatest volume reduction, by using only third and fifth harmonics one can achieve a substantial amount of that reduction while still operating well within harmonic limits. Variable frequencies may beneficially have their ranges compressed, and additional losses may be reasonable and/or compensated by loss reductions and operating benefits.

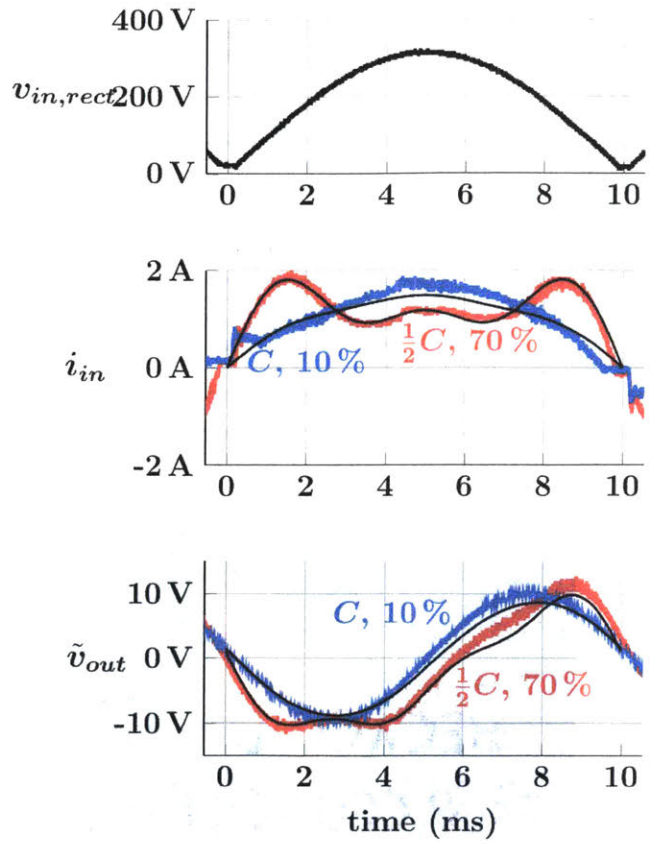


Figure 4-24: Comparison of output voltage ripple when harmonics are included (10 % vs 70 % of the allowable third and fifth harmonics) and capacitance is reduced. The reduced voltage ripple of about 50 % in Fig. 4-22 is traded for 50 % less capacitance. Calculated waveforms are shown in black for comparison.

4.9 Conclusion

As increased efficiency and switching frequency improve the size of other components of ac/dc converters, energy buffers become more of a bottleneck to miniaturization. By intentionally drawing currents at harmonics of the grid voltage, designers can greatly reduce the energy that must be stored each cycle, and therefore significantly reduce the size of energy buffer capacitors. We show this for every regulation class, with energy storage reductions between 25–75 % available depending on the class and power level. In most cases, this technique is available with a change of controls only, which is an important advantage over other techniques for cost-constrained applications. We presented a prototype which validates the results without incurring

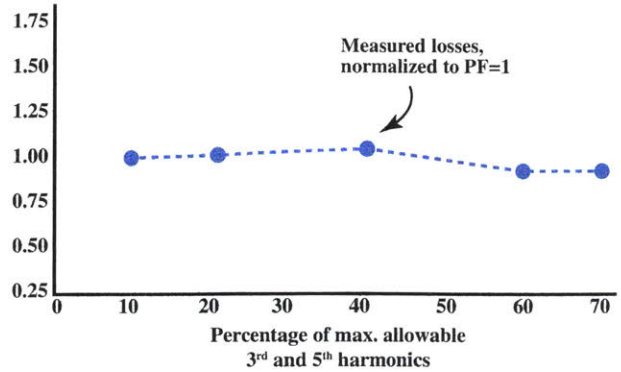


Figure 4-25: Measured converter losses, normalized to the low-harmonic case (10 % harmonic usage, 96 % efficiency). In this prototype, which is dominated by diode losses, including significant harmonic content has negligible effect on efficiency.

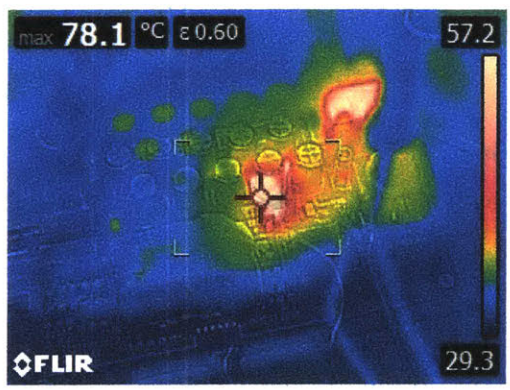


Figure 4-26: Thermal capture of the converter operating with 70 % of allowable harmonics, suggesting that diode losses (which are harmonic-independent) dominate in this prototype. The hot spot in the center is the boost diode, and the hot spot in the upper right is the diode bridge.

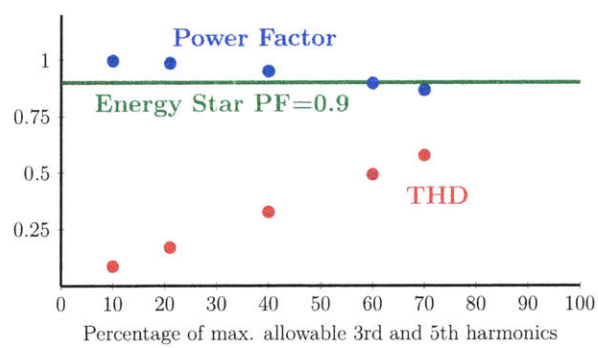


Figure 4-27: Experimental power factor and THD for the prototype converter for different levels of harmonic inclusion showing that significant energy storage reduction can be achieved even with reasonable power factor constraints (compare Fig. 4-22).

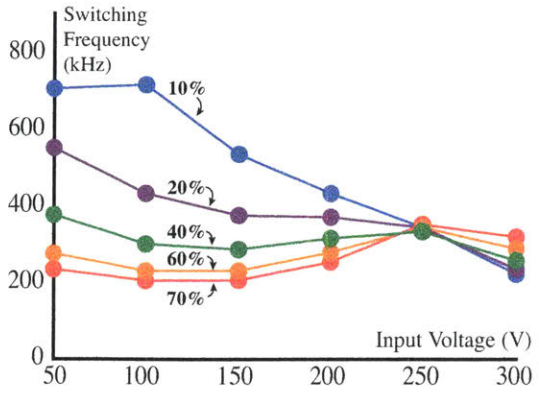


Figure 4-28: Local operating frequency of the valley-switched boost PFC across the first half of the rectified input voltage half-cycle. The variable frequency introduced by the valley-switched boost is greatly mitigated with the inclusion of input harmonics by drawing more current at low voltage. For each curve, third and fifth harmonics are each included at the listed percentage of their individually allowed maxima.

a significant efficiency penalty and demonstrates frequency compression which may be valuable for implementing high-efficiency variable-frequency PFC stages.

Looking forward, we note that there is nothing fundamentally incompatible between this approach and others that aim for high voltage ripple or use "active buffers" to reduce the buffer size (e.g. [52–55]). The benefits available from each approach are compoundable, such that a 50 % energy buffer reduction from each approach should reduce the buffer to 25 % of its original volume.

Chapter 5

Losses in GaN Transistors

Gallium Nitride (GaN) transistors, and in particular lateral GaN-on-silicon HEMTs, have attracted great attention due to their ability to operate well at higher frequencies than silicon devices. Nevertheless, GaN transistors experience increased effective on-state resistance when switched at high voltage and frequency due to charge trapping in the device [75–77]. This increased resistance (often much more than 2x with respect to dc) is variously known as current collapse, dynamic on-state resistance, or dynamic R_{on}. Dynamic R_{on} is not commonly reported in datasheets, and how it varies with operating parameters is not well characterized. In addition, characterization techniques for dynamic R_{on} have not been sufficiently developed or agreed upon. It is also notable that other high-frequency loss mechanisms exist in GaN devices that are not well described in datasheets. Important among these is hysteretic "off-state" loss P_{oss} that occurs in charging and discharging the device output capacitance [72] (this loss is often thought of as similar to an effective series resistance Ross with the device output capacitance, though how the loss varies with operating characteristics doesn't necessarily match the effect of even a nonlinear resistor). Therefore, better measurements of GaN device loss under realistic operating conditions, including dynamic R_{on}, will greatly benefit the design of power electronics incorporating GaN devices and aid in the improvement of the devices.

Some previous efforts have measured dynamic on-state resistance [73, 78–83]. The most common approach is to measure both switch current and voltage waveforms

during the on state. Since on-state voltage is small and the off-state voltage is large, a voltage clamp must be placed on the switch node to allow for accurate low-voltage measurements. However, some setups suffer from the clamp's RC time constant which limits operation speed [83].

Although this approach can be useful to determine the instantaneous dynamic R_{on} during a switching event and may provide useful insights into GaN device physics, it is difficult to convert the data into a lumped R_{on} parameter for modeling and simulation. Furthermore, this method is often measured under hard-switching conditions at low frequencies - an inauthentic environment for GaN switches in high-frequency power converters.

Another approach uses thermal data from a soft-switched high-frequency converter to determine dynamic R_{on} [12, 72]. While this approach provides an authentic high-frequency environment, the power converter may limit reconfigurability for different operating conditions and the reliance on thermal measurements is slow and subject to errors which may be difficult to assess. Moreover, as this technique relies on temperature rise as the measurement signal, it is difficult to determine the temperature dependence of both dynamic R_{on} and the off-state losses P_{oss}.

We propose an approach that is able to measure GaN device loss, including the effects of dynamic R_{on} and P_{oss}, at megahertz frequencies, using commonly-encountered HF voltage and current waveforms, while varying frequency, off-state voltage, and temperature. This is done by accurately measuring dc power into an unloaded resonant switching circuit operating under zero-voltage switching (ZVS). The measured input power is entirely attributable to losses, which can be designed to be dominated by conduction loss in the transistor. By subtracting the (small) estimated extraneous losses, one can compute an equivalent lumped dynamic R_{on} for a given operating condition. Moreover, as will be shown, the method can be extended to disambiguate the loss component P_{oss} when resonantly charging/discharging the output capacitance.

The remaining sections of the paper include a detailed description of the proposed technique (5.1), estimation of non-conduction losses (5.2), and validation of the technique (5.3). We then present experimental dynamic R_{on} data for 3 MHz operation

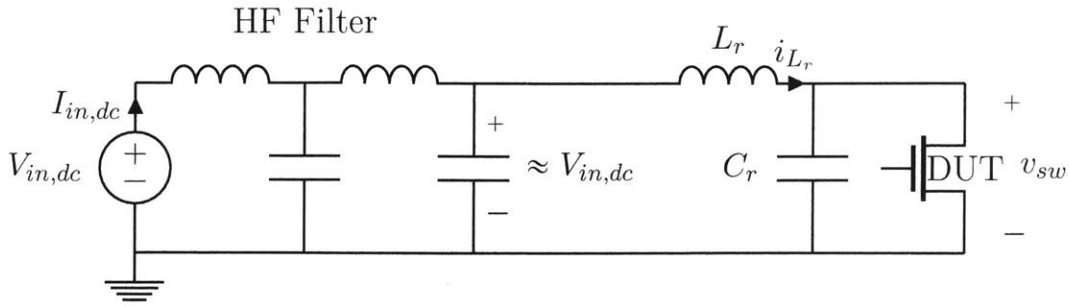


Figure 5-1: Simplified circuit diagram for the proposed measurement technique. The device under test (DUT) is operated under zero-voltage switching as indicated in Fig. 5-2. A nearly dc voltage is provided at the output of the filter, and the input power to the filter can be accurately measured with dc multimeters.

across temperature and voltage for GaN-on-silicon power devices manufactured by GaN Systems, Navitas, and Panasonic (5.4).

5.1 Measurement Technique

A simplified schematic of the proposed measurement circuit is shown in Fig. 5-1, with operating waveforms in Fig. 5-2. When the switch is on, the inductor current ramps up linearly. When the switch turns off, L_r resonates with the capacitor C_r (in parallel with the device capacitance, C_{oss}), delivering a half-sine pulse of voltage v_{sw} across the device. As the switch voltage returns to zero, the switch is turned back on with ZVS. The magnitude and frequency of the currents and voltage pulses can be designed by the choice of L_r, C_r, the dc input voltage, and the on-time of the FET. It is noteworthy that the on- and off-state waveforms are closely matched to those of some high-frequency converters (e.g. [84–88]) and reasonably match a wide variety of soft-switching circuits. Since the variation of dynamic R_{on} across operating conditions is largely unknown, testing in an authentic environment may provide more useful results for circuit designers than other available methods.

Overall Measurement Strategy:

1. Adjust t_{on} and V_{in} to impose off-state voltage pulses at the desired voltage and frequency.

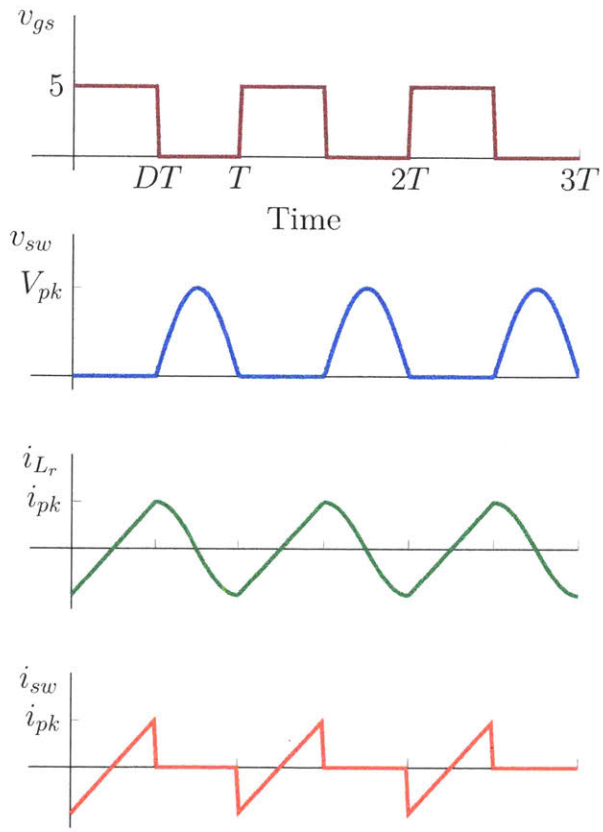


Figure 5-2: Circuit operating waveforms showing half-sinusoidal voltage pulses, quasi-sinusoidal inductor currents, and soft-switching conditions that are commonly found in HF converters.

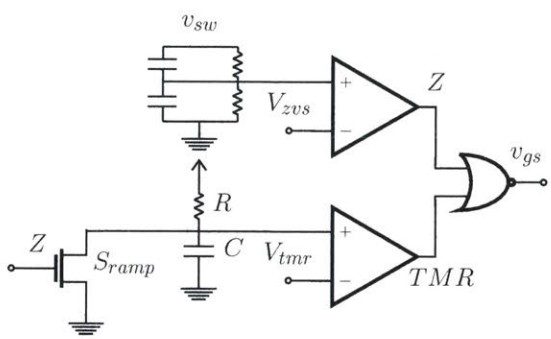


Figure 5-3: ZVS detector and timer circuit. FET on-time is set by the RC time constant and the dc V_{tmr} value. In this prototype, ADCMP601 were used for the comparators, with SN74LVC1G06 for the reset FET and 74LVC1G27 and 74LVC1G32 for the NOR and OR gates.

2. Measure the dc power P_{in} at the input port of the circuit. This can be done accurately with multimeters.

3. Estimate losses not attributable to conduction in the transistor P_{other} and subtract them from P_{in}.

4. Measure or infer the switch current $I_{sw,rms}$, and solve for dynamic $R_{on} = (P_{in} - P_{loss,other}) / I_{sw,rms}^2$.

Though these tasks can be summarized briefly, the challenge lies in making the measurements and estimations in steps 2-4 as accurately as possible.

Switching Control: The circuit is controlled by detecting ZVS each cycle to turn the FET on and using a ramp timer to determine the on-time and hence control turn-off (Fig. 5-3). This is similar to ZVS switching controls used in other applications [8, 16, 89]. The switch node voltage v_{sw} is divided and compared to a threshold voltage V_{zvs} producing a digital signal Z. When v_{sw} gets close enough to zero, Z transitions to low, which turns the DUT on. At the same time, when Z goes low, the switch S_{ramp} turns off and thus activates an RC-timed voltage ramp v_{ramp}. When v_{ramp} exceeds the threshold V_{tmr}, the digital signal TMR goes high and the circuit turns the DUT off. Turning the device off causes v_{sw} to pulse again, and the process repeats.

By manually adjusting V_{tmr}, the on-time can be varied. By varying t_{on} and V_{in}, the desired pulse voltage V_{pk} and overall frequency can be achieved. For any operating condition, V_{zvs} can be adjusted to achieve ZVS.

Hardware Setup: To facilitate testing of multiple devices, we divide the test setup into two components. First, a main board contains the filters[1], input power measurement ports, and a microcontroller and DACs that set signals V_{tmr} and V_{zvs}. (See Appendices O, P, and S for schematics, layout, and microcontroller code). Second, the transistor board contains the DUT and the HF control circuit (Fig. 5-3). (See Appendices Q and R for schematics and layout). The main board (Fig. 5-4) connects to the transistor board (Fig. 5-5) through a cable. This division allows the transistor board to be easily exchanged to test different devices, and it facilitates mounting the

[1]Since input current I_{in} demand is low but the input voltage V_{in} is high ($\sim$100 V), some voltage sources may operate in a light-load mode in which bursts of current are supplied at low frequency ($\sim$10 kHz). A lower frequency filter may be required between the source and the measurement point to prevent source noise from affecting the measurement of P_{in}.

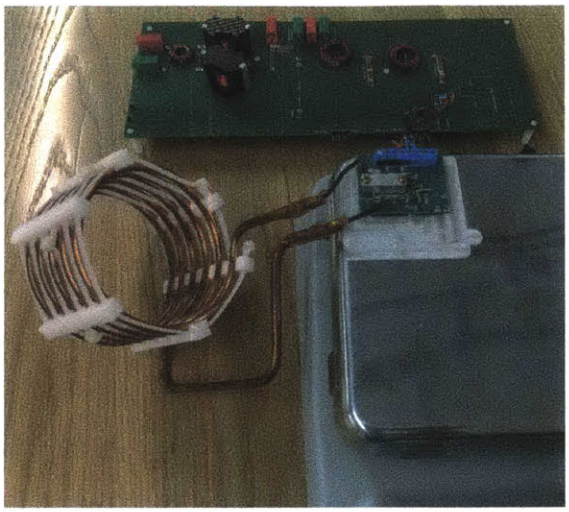


Figure 5-4: Overall setup, showing the main board with filters, input power measurement and dc signal generation, as well as a transistor board mounted to the hot plate (Talboys 984TA7AHPEUA 7) and the resonant inductor L_r.

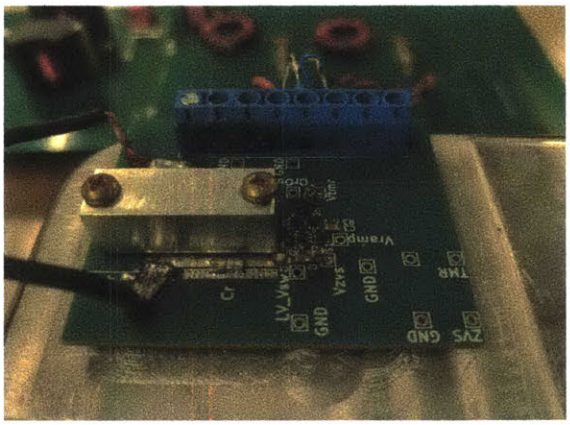


Figure 5-5: Transistor board mounted to the hot plate, containing the DUT under a mechanical clamp (Fig. 5-6) as well as the high-frequency ZVS detection and timing circuitry.

transistor board to a hot plate (Talboys 984TA7AHPEUA 7) for temperature control. Only dc signals are sent from the main board to the transistor board, including input voltage, logic supply V_{cc}, and control reference voltages. Analog voltages are filtered at the comparator inputs to prevent interference. All high frequency signals (e.g. v_{gs} or divided switch voltage $v_{sw,step}$) are contained on the transistor board and are run through short traces.

Temperature Control: We control device junction temperature T_j in order to

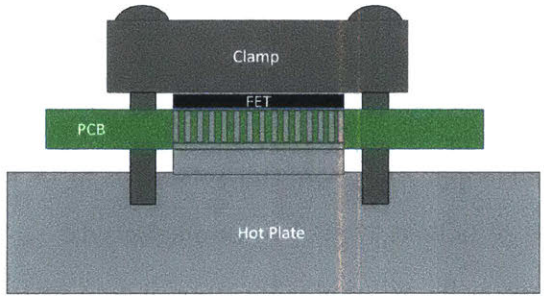


Figure 5-6: Side view of thermal control setup. The FET's source pad is via-farmed to the bottom layer of the PCB which makes thermal contact with the hot plate through an aluminum pedestal (machined flat, with thermal grease applied at each junction). A clamp is also screwed on top of the FET, through the PCB, and into the hot plate to provide even pressure.

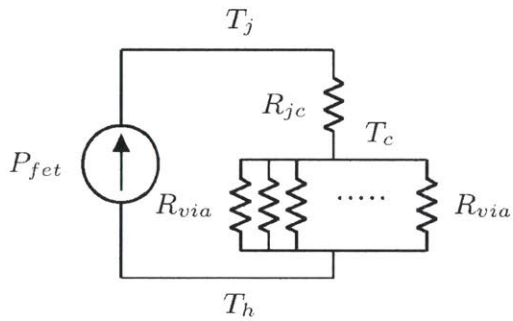


Figure 5-7: Thermal model of the thermal control setup. The hot plate temperature T_h is controlled, and heavy via-farming ensures a low thermal resistance connection from the hot plate to the device case (T_c) and junction (T_j). In the prototype system, the thermal resistance from device case to hotplate is kept below ~1 °C/W.

	R_{jc}	P_{fet}	ΔT_{jc}	% Error ($\frac{\Delta T_{jc}}{80\,°C}$)
PGA26E19BA	1.9 °C/W	2.35 W	4.46 °C	5.56 %
NV6131	2.2 °C/W	2.21 W	4.86 °C	6.08 %
GS66504B	1.0 °C/W	2.20 W	2.20 °C	2.75 %

Table 5.1: Breakdown of expected errors between junction temperature and case temperature T_c, taken from measurements at $T_c = 80\,°C$ and $V_{pk} = 400\,V$. It can be seen that there are small discrepancies between T_j and T_c; case temperature control is effectively junction temperature control for purposes of testing across a wide range of temperatures.

determine the effects of temperature on dynamic R_{on}. This is done by heat-sinking the device to a hot plate through a low thermal resistance path as shown in Figures 5-6-5-7. Using an aluminum hot plate with high thermal conductivity, and with typical dissipation values in our system, we can assume that the case temperature of the

device is very close to that of the hot plate. The temperature at the device junction is then the hot plate temperature T_h plus a small ΔT_{jc} owing to the non-zero thermal resistance from the junction to the case (Table 5.1). With an appropriately-designed via farm to an exposed copper pad on the reverse side of the PCB, this thermal resistance can be made sufficiently small to be neglected.

To validate this approach, we model the junction-to-case thermal resistance according to the datasheet values, which are about 1–2 °C/W. We model a single via has having $R_{via} = 37.4\,°\mathrm{C/W}$ [90]. Assuming 100 vias in parallel, we obtain an effective case-to-hot plate resistance of $R_{ch} = 0.37\,°\mathrm{C/W}$, for a total junction-to-plate thermal resistance of approximately 2.5 °C/W. With 2 W of loss in the FET (for example), the expected temperature difference from the junction to the hot plate is a negligible 5 °C.

5.2 Other Loss Attribution

Given the simplicity of the circuit, the total losses P_{in} can be ideally attributed to only a few sources, namely conduction loss in the FET P_{cond} and other losses including losses in the resonant inductor P_{lr}, in the FET output capacitance's P_{oss} [72] [12], in the filter inductors P_{filter}, and in the ZVS detector voltage divider P_{zvs}. By design, the non-conduction losses P_{other} should be as small as possible; nevertheless, their actual losses can be estimated and subtracted from P_{in} before computing dynamic R_{on}. Most losses are amenable to accurate estimation; we expound on the major contributors of the loss below, in order of decreasing significance. In doing so, we note that a limitation of the proposed technique is that all unidentified loss sources (or under-represented losses) are construed as additional conduction loss owing to dynamic Rds-on. Thus, it is important to account for other loss sources as accurately as possible. In particular, the design or selection of the passive components in the system, especially L_r, is crucial to obtaining precise measurements of dynamic R_{on}. For more details on the design and selection of passive components, refer to Appendix E.

C_{oss} Losses (P_{oss}): The loss associated with the output capacitor of the DUT

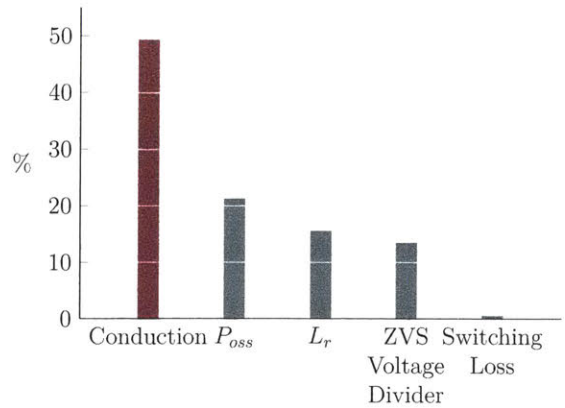


Figure 5-8: Typical breakdown of estimated losses' contribution to total loss (%) based on measurements under 400V V_pk at 3 MHz on a Panasonic FET (PGA26E19BA). It is important that P_{cond} represent a large fraction of the overall loss to prevent errors in estimating P_{other} from impacting the results.

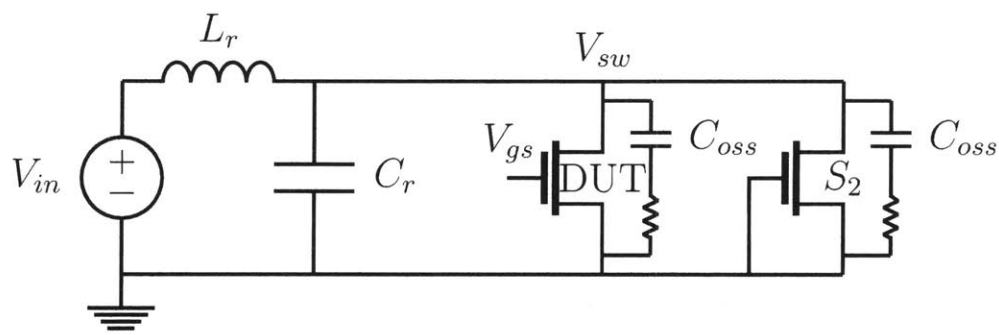


Figure 5-9: Method for extracting P_{oss} loss by adding a second always-off transistor (filters not shown). For the same experiment with this setup, the additional loss is wholly attributable to P_{oss}.

(P_{oss}) can be high since the transistor terminals experience high-voltage pulses at high frequency; this loss mechanism is known to be significant at high frequencies in GaN devices and some Si devices. In some cases, these losses are ohmic in nature, while in others the losses are hysteretic or have even more complex relationships [12,72]. This loss can be difficult to disambiguate from conduction loss since both occur within the FET. Nevertheless, one means to disambiguate P_{oss} from conduction loss is by connecting a second always-off transistor S_2 in parallel to the DUT and using the additional loss imposed by the second device as an estimate of P_{oss}, similar to [12,72].

Because S_2 is always off, the operating waveforms shown in Figure 5-2 are not significantly changed (recall that C_r dominates the switch node capacitance by de-

sign, so the added C_{oss} of the second device does not significantly affect the operating waveforms; if it does, then the resonant capacitance can be slightly adjusted to compensate for this). P_{oss} can be estimated by performing power output measurements twice: once *without* S_2 in place, and once *with* S_2. The difference in FET loss in the two cases is attributed to P_{oss}. Note that this estimation technique assumes that P_{oss} of an always-off device is the same as that of a device that switches at high frequency.

Resonant Inductor Loss: The resonant inductor L_r accrues loss P_{lr} as current flows through its equivalent ac resistance R_{lr}. P_{lr} is estimated from the measured rms inductor current $I_{lr,rms}$ and the equivalent resistance of the inductor at the operating frequency R_{lr}.[2] The inductor current is captured on an oscilloscope and the measured inductor current is used to obtain $I_{lr,rms}$. The resonant inductor loss can then be calculated as $P_{lr} = I_{lr,rms}^2 R_{lr}$.

It is important to estimate P_{lr} as accurately as possible since P_{lr} is expected to be one of the most significant contributors to P_{other}. We use a large air-core structure to achieve a high-Q inductor with linear resistance, which keeps P_{lr} both low and predictable. For more details on the design of inductor L_r used in the experimental system, see Appendix E. Note that it is important to keep this large inductor away from nearby conductors or magnetic materials, including incidental metals in or under the test bench.

Voltage-Divider Loss: The switch voltage is monitored in order to obtain ZVS (see Fig. 5-3). This is done by stepping down the voltage through a voltage divider, which accrues loss. This loss is calculated straightforwardly as $P_{zvs} = V_{sw,rms}^2/R_{step}$, where R_{step} is the total resistance of the divider. $V_{sw,rms}$ may be calculated from the switch voltage as measured on an oscilloscope.

Turn-on and Turn-off (Transition) Loss: The loop containing the resonant capacitor and the FET contains some parasitic inductance L_p. During every turn-on/turn-off period, L_p is magnetized/demagnetized and dissipates loss in the process. The turn-on and turn-off losses are calculated equivalently as $P_{ton} = P_{toff} =$

[2]To avoid accidentally including probe losses, which can be significant at high frequency, measurement of input power should only be taken with no voltage/current probes attached to the system. Probes are then attached to measure e.g. I_{lr} and v_{sw}.

$f\frac{1}{2}L_pI_{tran}^2$, where I_{tran} is the instantaneous current through the device during a transition event. We infer I_{tran} from the measurement of the resonant inductor current. Assuming $L_p = 1$ nH, calculations at 3 MHz show that transition losses account for a small portion (0.47 %) of total losses (Figure 5-8). In many cases, it could be ignored.

We need not consider any turn-on overlap or capacitive discharge loss since the circuit achieves zero-voltage switching (ZVS). Overlap turn-off loss can also be ignored due to the strong snubbing effect of C_{oss} and C_r.

5.3 Validation

To validate the proposed approach, we performed FET loss measurements across various voltages and verified them as plausible against thermal measurements. For example, we performed thermal resistance measurements on a Panasonic device (PGA26E19BA) by setting v_{gs} to zero and passing dc current from the source to the drain *without* heat-sinking the transistor board. Measurements of I_{sd}, V_{sd}, and ΔT showed a measured thermal resistance of 35 °C/W (case ΔT for given dissipation).

We then performed a dynamic R_{on} test at (3 MHz and 400 V_{pk}), again without heat-sinking the device. The experiment produced a calculated FET loss (P_{cond} + P_{oss}) of 1.6 W during operation. We simultaneously recorded the temperature of the FET case with a thermal camera, which was $\approx$ 86 °C. Letting ambient temperature $T_A = 25$ °C, we expect the temperature of the FET to be: $T_A + 35\,°C/W \cdot 1.6W = 81\,°C$ which is reasonably close to the measured value of 86 °C. We performed several of these tests across various voltages and achieved very similar results. This verifies the accuracy of our loss modeling and adds confidence to our FET loss calculations.

5.4 Results

In this section, we report experimental results of dynamic on-state resistance of commercial GaN-on-silicon FETs from GaN Systems, Navitas, and Panasonic (all in the 600–650 V rating range). For each device, data was recorded at 3 MHz over-

all frequency, for two different peak off-state voltages (V_{pk} = 200 V and 400 V), at several temperatures (room temperature, 80 °C, and 120 °C) as explained in Sec. 5.1. As measurements were performed at the same frequency for similar devices, the same resonant inductor was used while the resonant capacitor value was adjusted slightly.

Results from each device are shown in Figs. 5-10–5-12, with dynamic R_{on} normalized to the datasheet value at 25 °C. The most striking feature of the results is the high overall value for dynamic R_{on}, roughly 4-6 times the room-temperature static R_{on} value on the datasheets. This result is consistent with findings by other research groups [72] that reported FET conduction loss multipliers ranging from 2-3 over their datasheet values at elevated temperatures.[3] The switching environment between this work and that of [72] are somewhat similar (soft-switched at MHz frequencies) and any discrepancy between the findings may be attributable to differences in v_{sw} voltage and i_{sw} current waveforms. Nevertheless, the results agree that the dynamic R_{on} values are substantially higher than static R_{on}, even when temperature is taken into account.

In general, dynamic R_{on} increases with peak switch voltage as seen in Figs. 5-10–5-12. This result is consistent with previous findings [79], [80], [91]. This is most likely due to high electric fields in the channel forcing electrons into trap states, causing dynamic R_{on} to increase.

The results in Figs. 5-10–5-12 also allow us to see how dynamic R_{on} varies with operating temperature. In particular, the results suggest that dynamic R_{on} is not a strong function of temperature for these devices. This may be the result of two conflicting phenomena: 1) as temperature increases, electrons in trap states are energized and more readily escape the traps, decreasing dynamic R_{on}, and 2) as temperature increases, ohmic resistance increases which increases the apparent R_{on}. The relative strength between these two competing effects may vary across devices, voltages, and temperature ranges.

[3] [72] does not report operating temperatures, but reasonable elevated temperatures can cause dc resistance to increase by 1.5-2 times.

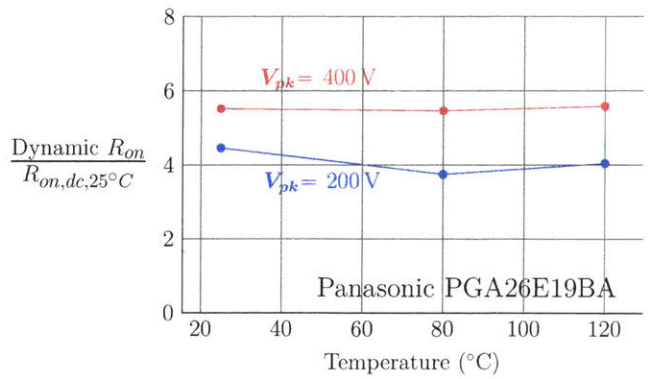


Figure 5-10: Thermal sweep across 20, 80, and 120 Celsius show that dynamic R_{on} isn't a strong function of temperature. Measurements were done at 3 MHz on a Panasonic FET with static $R_{on} = 140m\Omega$ at 200V and 400V V_{pk}.

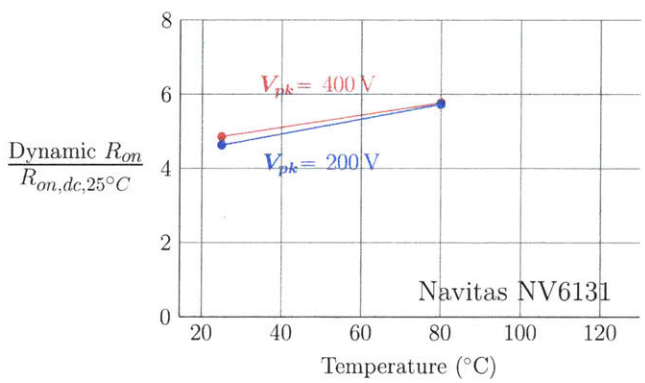


Figure 5-11: Thermal sweep across 20 and 80 Celsius show that dynamic R_{on} isn't a strong function of temperature. Attempts at measuring at 120 Celsius caused the FET to fail. Measurements were done at 3 MHz on a Navitas FET with static $R_{on} = 135m\Omega$ at 200V and 400V V_{pk}.

5.5 Conclusion

We proposed a technique for measuring losses in GaN transistors at high frequency. This approach is capable of disambiguating dynamic R_{on} losses and P_{oss} losses, and can be performed at a variety of frequencies, off-state voltages, and temperatures.[4] Finally, the voltage and current waveforms imposed upon the DUT authentically resemble those of many high-frequency converters.

We also contribute some loss data for extant commercial GaN transistors. In

[4]We have tested devices using this approach at 3 MHz and 200–400 V peak, consistent with the application of such devices explored later in this thesis.

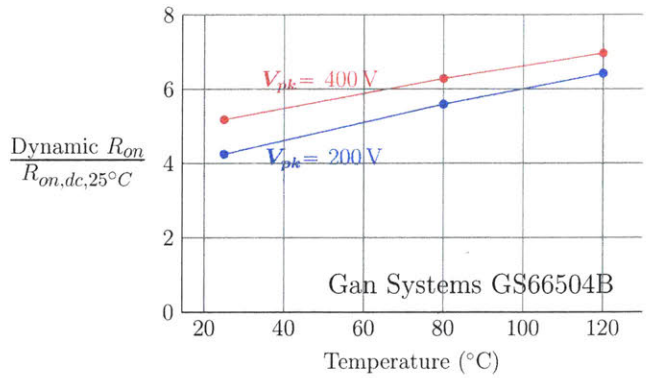


Figure 5-12: Thermal sweep across 20, 80, and 120 Celsius show that dynamic R_{on} increases with temperature with decreasing margin. Measurements were done at 3 MHz on a GaN Systems FET with static $R_{on} = 100m\Omega$ at 200V and 400V V_{pk}.

general, the losses in the GaN FETs presented are significantly higher than expected from the listed R_{on}, even when accounting for temperature rise and other losses like P_{oss}. The tested devices show an increasing dependence of dynamic R_{on} with off-state voltage, as is expected. The dependence on temperature is less severe than would be expected from the static R_{on} temperature dependence.

Given the importance of accurate loss predictions for design, simulation, and modeling - along with the difficulty of measuring dynamic R_{on}, the proposed technique offers a solution to characterizing these significant discrepancies.

Chapter 6

An Improved PFC Circuit

Electrical loads process real power by drawing current at the same frequency as (and in phase with) the source voltage. Other frequency components of the input current result in reactive power and deliver no net energy to the load; these currents are nevertheless physically real and may dissipate energy in any source impedance (e.g. resistance in mains distribution lines and transformers). For devices connected to the single-phase ac grid, currents at harmonic frequencies of the grid voltage are therefore regulated according to international standards, e.g. IEC/EN 61000-3-2. Power conversion stages which draw compliant currents by design are called Power Factor Correction (PFC) converters [92–94].

Power factor correction stages often make up a significant fraction of the overall volume of grid-interfaced power converters. Miniaturization using MHz switching frequencies is attractive, but fCV^2 switching losses become intolerable at grid voltages without "zero voltage (soft) switching" (ZVS). Most soft switching techniques are only suitable for narrow operating voltages and/or powers and therefore have not been widely used in PFC stages [95], limiting PFC stages to low frequency (LF, 30–300 kHz) operation[1] with large passive components for both power conversion and EMI filtering.

[1]Some PFC converters exceed this frequency range by achieving soft switching at the expense of additional lossy circuitry or by only partly achieving soft switching [96, 97]. Other approaches can achieve ZVS using complex switching networks [98, 99], stacked architectures [100, 101], or wide frequency ranges [102–104]. See [95] for further discussion.

To illustrate the problem, consider the boost PFC converter as part of a two-stage architecture – arguably the most common combination in use. The PFC stage boosts from rectified universal input (85–265 Vrms) to a dc bus around 400 V. Operated near boundary conduction mode, the boost converter may allow a resonant transition to reduce its switch node voltage prior to turn-on. This process results in true ZVS for $V_{in} < V_{out}/2$, and "valley-switching" at $v_{min} = 2V_{in} - V_{out}$ otherwise [105]. For much of the line cycle, the switch still turns on with hundreds of volts across it, making high frequency operation untenable.

Here we present a PFC converter which achieves ZVS for any step-up voltage conversion ratio. It can therefore act as a soft-switched replacement for popular boost PFC stages without any modifications to the rest of the system architecture. In addition, the converter uses a blended feedforward/feedback control scheme which eliminates the need for current sensing (both high-frequency inductor current and low-frequency input current). These features enable switching frequencies in the MHz regime and the opportunity for greatly reduced inductor and EMI filter sizes [106].

The proposed converter is based on a dc-dc converter with wide operating range previously published as part of this research by the author [95]; this chapter focuses on new aspects required for its adaptation to ac-dc PFC applications. In Section 6.1, the basic converter operation is cursorily reviewed, but readers are referred to [95] for more thorough background. Section 6.2 proposes a blended feedforward/feedback control technique to achieve power factor correction without the need to sense input current or inductor current. Section 6.3 presents a prototype and discusses details which are important for practical implementation. Section 6.4 presents experimental results showing that the converter reliably achieves ZVS at MHZ frequencies across the line cycle with power factors around 0.998 (THD $\sim$ 6%). We conclude that the converter has potential for high power density in a wide array of existing applications, including those with stringent power quality requirements.

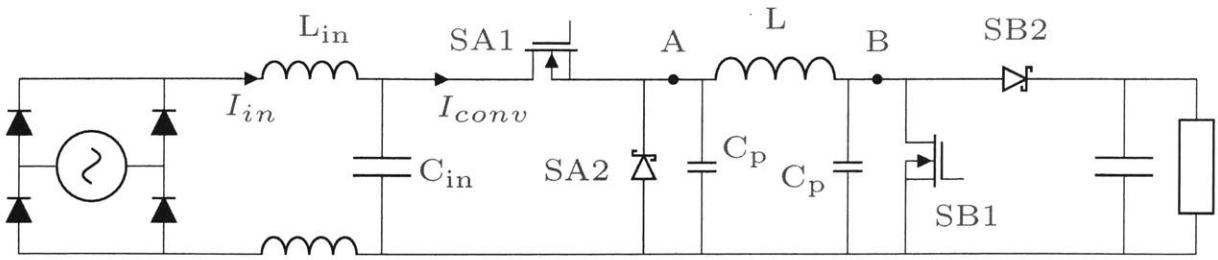


Figure 6-1: The proposed power factor correction topology, which is identical to the four-switch buck-boost converter (the advantages of the proposed converter arise through control). Lumped parasitic capacitances C_p are drawn explicitly, and rectification bridge and an example emi filter are shown for completeness.

6.1 Abridged Operation Overview

The proposed converter (Fig. 6-1) has a power stage that is topologically identical to the four-switch buck-boost converter, but is controlled to achieve zero-voltage soft switching (unlike pure boost converters) with low rms current (unlike pure buck-boost converters) across the line cycle. The proposed converter can thus operate at much higher frequencies without incurring high loss penalties. The converter has two distinct modes of operation.

Low Voltage / Boost Mode: The converter may be operated as a conventional boost converter by turning switch SA1 on for the entire switching cycle. This mode (see Fig. 6-2) has an energy storage phase (SB1 on), a direct delivery phase (SB2 on), and a resonant phase to achieve ZVS (SB1 and SB2 off). The LC resonant phase begins with zero initial current, a dc offset voltage V_{in}, and an initial capacitor voltage V_{out}; as such, v_B will ring down to zero as long as $V_{in} < V_{out}/2$. Switch SB1 is turned on in response to the zero voltage condition (see Sec. 6.2 and Fig. 6-4) which may occur before the inductor current returns to zero; as such, the current at turn-on i_0 may be somewhat negative.

High Voltage / Buck Mode: This mode is nearly identical to the boost mode in operation, with SA1 and SA2 active instead of SB1 and SB2. During the resonant period of this mode, the switching node A may ring up as high as $2V_{out}$. Therefore, for any $V_{out} < V_{in} < 2V_{out}$, the buck mode may be employed with ZVS and hence

high-frequency operation.

Medium Voltage Mode: The boost mode and buck modes leave a substantial voltage range $V_{out}/2 < V_{in} < V_{out}$ with no ability to achieve ZVS and hence high-frequency operation. When the input voltage approaches the output, the relevant switch may "miss" ZVS by as much as V_{out}, or hundreds of volts. Therefore, a new mode is proposed in [95] which achieves ZVS for any $V_{in} < V_{out}$. The progression of switching states includes an energy storage phase (SA1 and SB1 on), a direct delivery phase (SA1 and SB2 on), an indirect delivery phase (SA2 and SB2 on), and a resonant phase (all switches off). During the resonant phase, switch SA1 turns on when node A reaches V_{in}, while switch SB1 turns on Δt later when node B reaches zero; this does not significantly affect the understanding of the switching states, but must be accounted for in control.

The advantage of this mode lies in its "CLC" resonant phase, unlike the "LC" resonant phase in the boost mode. When the resonant phase begins, node A starts at zero volts, node B starts at V_{out}, and there is no voltage offset (i.e. from the input voltage source). This scenario is guaranteed to return node B to zero and node A to (at least) V_{in} for *any* $V_{in} < V_{out}$.

The progression of switching states may be understood as that of a boost converter with an indirect delivery phase added to the end, thereby creating the necessary conditions in the resonant period to achieve ZVS. (The resonant phase distinguishes this approach from previous uses of this topology in CCM [107] or DCM [108, 109], thus supplying the important advantage of soft switching.) The progression of states may alternatively be understood as that of the conventional four-switch buck-boost converter (i.e. with a triangular inductor current waveform) with a direct delivery phase added in the middle, thereby reducing the rms current required for the same power.

There is a minimum value for the inductor current at the end of the direct delivery phase, denoted i_2 in Fig. 6-3. In order to properly commute from SA1 turning off to SA2 turning on, the inductor energy $\frac{1}{2}Li_2^2$ must be sufficient to discharge the parasitic capacitance C_p at node A. To the extent that this condition is violated, node B will

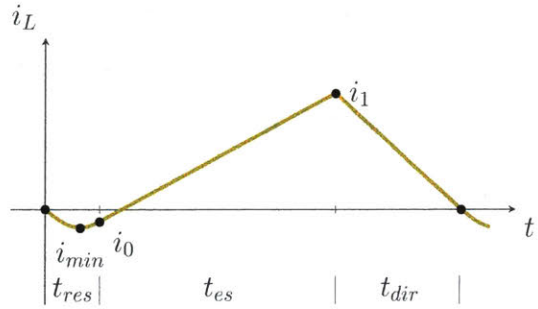


Figure 6-2: Low Voltage (Boost) Mode inductor current waveform. This mode achieves ZVS when $V_{in} < V_{out}/2$. The switching cycle is divided into a resonant phase (SB1, SB2 off), an energy storage phase (SB1 on), and a direct delivery phase (SB2 on).

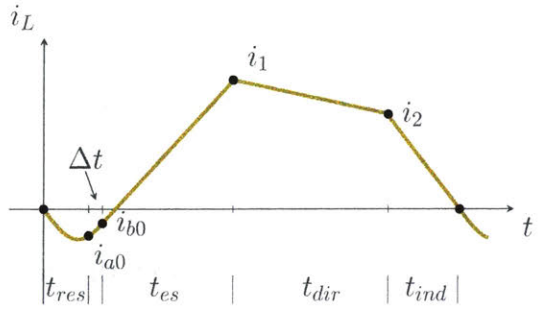


Figure 6-3: High Voltage Mode inductor current waveform. This mode achieves ZVS when $V_{in} < V_{out}$, but is most useful where the Boost mode loses ZVS, i.e. $V_{in} > V_{out}/2$. The switching cycle differs from the Boost mode primarily with the indirect delivery phase (SA2, SB2 on).

not ring all the way down to zero volts in the final resonant transition. Maintaining i_2 above its minimum value will be an important constraint on control.

It is important to note that the inductor current may be inferred from on-times without measuring switching-frequency current. Indeed, in the high voltage mode, t_{res}, Δt, i_{a0}, and i_{b0} may be computed from voltage measurements and circuit parameters alone. The switch on-times are directly related to t_{es}, i_1, t_{dir}, i_2, and t_{ind}. A similar logic applies to the typical boost and buck modes. This observation means that any desired features of the inductor current waveform (esp. the average input current and i_2) may simply be computed and executed without current measurement or feedback. This is an important advantage where complex control is required (as in PFCs) while maintaining ZVS at high frequency. The only additional requirement is

a control circuit that can (1) respond to ZVS detection by turning a switch on with sufficiently low delay and (2) hold the switch on for a programmable duration. Such a circuit is described in Sec. 6.2 and Fig. 6-4.

6.2 Control

The control of the proposed converter is different from many converters, though simply understood. A dedicated high-speed control circuit, like that shown in Fig. 6-4, is required for each controlled switch.[2] A given switch (e.g. SA1, SB1) is turned on when its corresponding ZVS Detector senses low voltage across the switch. The switch is then kept on for a certain on-time by way of its corresponding Ramp Timer, whose time-out is dictated by a dc control voltage. After the switch turns off, the cycle repeats as long as ZVS is eventually achieved again.

The switch turn-on and turn-off actions are thus, in a sense, passive. No input from a microcontroller is required, except to select the dc or slowly varying ZVS trigger voltage $\mathrm{REF}_{\mathrm{ZVS}}$ and on-time control voltage $\mathrm{REF}_{\mathrm{TMR}}$. The turn-on and turn-off events for a switch are asynchronous from the microcontroller clock, and indeed even from the events of the other switches. Therefore, the proposed converter control should not be understood as pulse-width-modulation or frequency control, though pulse-widths and frequencies will both vary. The most apt description would be "on-time control," though this phrase risks confusion with different methods having unfortunately similar names (e.g. constant on-time control, adaptive on-time control, etc.). In this case, "on-time control" simply means that the on-times are the only control variables, and the off-times and the timings of the turn-on/turn-off events are not directly commanded.

With the concept of on-time control understood, we turn to the proposed converter in particular. A possible control approach for this converter using constant on-times was suggested in [95]; despite its commendable simplicity, this prior approach was

[2]The control circuit in Fig. 6-4 is improved over the one in [95] by using a current mirror (instead of a lone resistor) in the Ramp Timer. This solution is easily executed either on- or off-chip and results in a much more linear ramp.

underspecified, had limited control of input current, and forced the converter into inefficient operation.

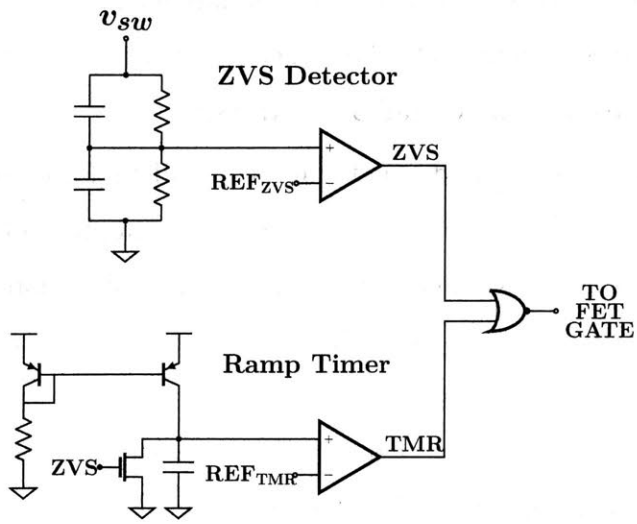


Figure 6-4: Auxiliary comparator-based control circuit used for each active switch in the proposed topology, allowing for turn-on in response to zero-voltage conditions and a programmable on-time. Minor variations to this circuit may be required as discussed in Section 6.3. ADCMP601 were used for the comparators, with SN74LVC1G06 for the reset FET and 74LVC1G27 and 74LVC1G32 for the NOR and OR gates.

Here we propose an approach which achieves higher efficiency and grants the designer arbitrary control of the input current waveform (including high power factor/low THD waveforms) at the expense of only superficial complexity. The approach modulates on-times across the line cycle to control the input current shape; this is done in a feedforward manner using only input/output voltage measurements and pre-programmed circuit parameters. Thus, the designer need not measure input current nor design a feedback loop for this purpose. Though feedforward control in general is rightly avoided for its inaccuracy under uncertain parameters, inaccurate measurements, and incomplete models, we will show that feedforward control may be sufficiently reliable for purposes of power factor correction.[3]

Let us consider the medium-voltage mode. With two control variables ($t_{a,on}$, $t_{b,on}$),

[3]In addition to the feedforwad "inner loop" controlling the input current *shape* over the line cycle, a traditional feedback "outer loop" (slower than the line cycle) controls the output voltage by way of the *magnitude* of the input current waveform. There is nothing unique about this feedback loop, and it is mentioned only for completeness.

we may select two features of the inductor current waveform to target. We choose to target the average (over a switching cycle) input current I_{in} and the corner current i_2. We wish to maintain the corner current i_2 at its minimum allowable value, minimizing rms currents while maintaining ZVS. We target the average input current, of course, for power factor correction and overall power transfer.

To meet these targets, we need a mathematical model relating switch on-times to the average input current I_{in} and to i_2. The analysis is simply explained, though the actual computations are messy and left to Appendix F. We quote only the driving logic and the final results here. To maintain precision, we use capital symbols to denote values that are constant or averages across a switching cycle; we use lower-case symbols to denote values that change within a switching cycle or that only have meaning within a switching cycle. We also introduce the notation $X = V_{in}/V_{out}$, as this ratio appears frequently. Finally, because we use V_{in} for the local input voltage (averaged over a switching cycle), we instead use V_{rms} to refer to the rms input voltage (taken over the line cycle). The peak of the input voltage waveform is then $\sqrt{2}V_{rms}$.

The logic for input current control is as follows. We are able to use a feedforward approach to because the converter always returns to the same state each cycle (at $t = 0$ in Figs. 6-2, 6-3). In principle, the entire behavior of the switching cycle can be predicted from circuit parameters, measured input/output voltages, and the commanded on-times. Starting at $t = 0$, we can determine when v_A will reach zero (t_{res}), what the inductor current will be (i_{a0}), then at what point will v_B reach zero (Δt later), etc. With selected on-times, the required variables are all known and the entire switching cycle is predictable. We can then compute I_{conv} and i_2.

In practice, the above analysis is reversed. One begins with a desired average input current I_{in} (determined by the position in the line cycle and also the slow output voltage feedback controller) and then accounts for the current into C_{in} to obtain a required average converter input current I_{conv}:

$$I_{conv} = I_{in} - C_{in}\omega_{line}\sqrt{2V_{rms}^2 - V_{in}^2} \tag{6.1}$$

With a required I_{conv} and a desired i_2, the required i_1 may be computed:

$$i_1 = I_{conv} + \sqrt{I_{conv}^2 + i_2^2 X - 2I_{conv} i_2 X^2 + 2I_{conv} i_2 D X(1-X)} \tag{6.2}$$

where $X = V_{in}/V_{out}$, $D = t_{res2} V_{out}/L i_2$ and $t_{res2} = \frac{1}{2}\frac{2\pi}{\omega_2} = \pi\sqrt{LC_p/2}$. From there, the required times $t_{a,on} = \Delta t + t_{es} + t_{dir}$ and $t_{b,on} = t_{es}$ can be backed out and then commanded:

$$t_{b,on} = L\frac{i_1}{V_{in}} + L\frac{V_{out}\sqrt{\frac{C_p}{L}}(1-X)}{V_{in}} \tag{6.3}$$

$$t_{a,on} = t_{b,on} + L\frac{i_1 - i_2}{V_{out} - V_{in}} + \frac{2\sqrt{2LC}(1-X)}{\sqrt{X - X^2} + \sqrt{2}(1-X)} \tag{6.4}$$

While the equations for this feedforward approach appear complicated on paper, the approach is actually simple to implement in hardware. A microcontroller or ASIC measures the input/output voltages and has pre-programmed values for circuit parameters. It then simply performs a few calculations and commands the switch on-times by way of the control voltages $\mathrm{REF}_{\mathrm{TMRa,b}}$. The actual turn-on and turn-off events are executed with the dedicated high-speed circuitry and need not be "controlled" *per se*. Finally, we emphasize once again that no current measurement is required, neither high-frequency inductor current nor low-frequency input current. Microcontroller code can be found in Appendix V.

Although the discussion above treats the proposed high voltage operating mode, similar logic applies to the low voltage boost mode. The calculations for the boost mode and buck mode also appear in Appendix F.

The waveform quality available with the above approach can be seen in Fig. 6-5 showing experimental waveforms from the prototype in Sec. 6.3 for 220 V rms input, 400 V output, and 300 W power. Note in particular that the mode transition, often

troublesome in multi-mode converters, is indiscernible. The converter achieved THD below 10 % over the entire power range, with the majority of THD attributable to zero-crossing distortion. Nothing prevents further refinements to the model (in particular the assumption that the C_p charging/discharing times after t_{es},t_{dir} are negligible) from reducing THD further if desired.

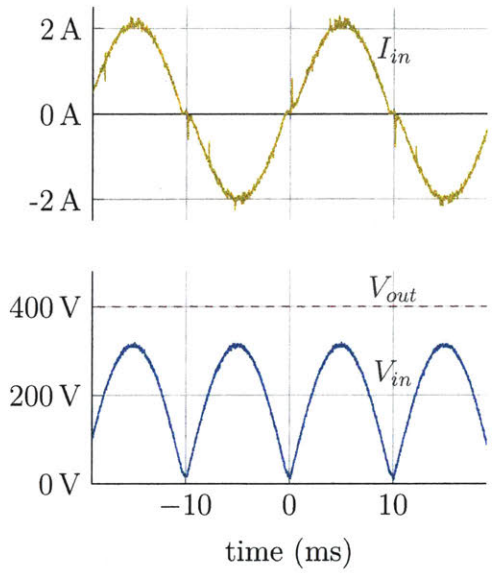


Figure 6-5: Experimental operating waveforms for the proposed converter. The filtered input current has high power quality (THD < 10 %), with no discernible distortion from mode transitions. Zero-crossing distortion is the major source of THD, and can be eliminated with more detailed modeling. Measurements at 220 V rms input, 400 V output, and 300 W power

6.3 Implementation Details

The PFC converter was implemented in a hardware prototype utilizing GaN FETs, SiC diodes and advanced high frequency magnetics [110] (Table 6.1). The design operates with dynamic frequency variation in the 2–4 MHz range, approximately 10x that of conventional PFC systems, with commensurate reductions in passive component values.

In addition to the power stage components, the high speed control circuit components used are also listed in Table 6.1. The control circuit for SB1 is straightforward

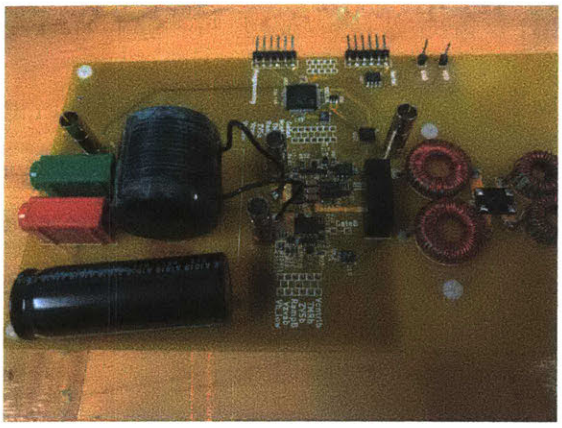


Figure 6-6: Photograph of the prototype converter, including twice-line-frequency energy buffer capacitor, power stage inductor, switching and control elements, and input filter and rectifier.

Component	Part/Value
FET SA1,SB1	Navitas 6131
Diode SA2,SB2	Wolfspeed/Cree C3D1P7060Q (2 each)
Capacitor C_{out}	220 µF 450QXW220MEFC18X50
Inductor L	13.5 µH
Core Material	Fair-Rite 67
Inductor Design	$Q = 620$ at 3 MHz ([110], Ch. 2)
Diode Bridge	Z4DGP406L-HF
Comparators	ADCMP601
Current Mirror BJTs	2SA1873
Ramp Reset FET	SN74LVC1G06
Gate Drive	"TinyLogic" NC7WZ16
Microcontroller	PIC32MZ0512EFE064

Table 6.1: Part selection for the prototype. See Appendix T for more details.

of powers and at switching frequencies of 2–4 MHz, as predicted in Section 6.1. By contrast, the conventional boost PFC would lose ZVS with hundreds of volts at turn-on (for universal input). As such, the conventional solution would not be feasible at these frequencies with available semiconductor devices and loss allowances; the conventional boost solution is thus limited to lower frequencies and larger passive values.

The switching frequency was also measured for the rising portion of the line cycle (Fig. 6-8), showing that frequency variation is low for a particular power and reasonable across powers. Experimental frequency measurements validate the model used

to implement, since SB1 is ground referenced. For SA1, there are two options:

1. Use the circuit in Fig. 6-4 and reference the control circuit to the source (node A) and v_{sw} to the drain V_{in}. The $\mathrm{REF_{ZVS}}$ and $\mathrm{REF_{TMR}}$ signals must be obtained in an isolated way, but no high frequency signals have an isolator in their path. This implementation is theoretically faster, but susceptible to noise as the signal "ground" is referenced to a switching node. This approach is difficult but feasible, and was used in [95].

2. Reference the control circuit to ground and connect v_{sw} to node A. In this case the ZVS Detector is not watching for v_{sw} to ring down, but rather to ring up – as such, the polarity of the ZVS comparator should be reversed. Additionally, the ZVS trigger signal $\mathrm{REF_{ZVS}}$ must be modulated as V_{in} changes (as opposed to the ZVS trigger for SB1 which need not change). This implementation requires isolation to bring the gate drive signal to the SA1 source voltage domain, thus placing a delay in the high-frequency path. The advantage of the ground-referenced controls is substantial, however, and we use this approach in this prototype. For isolation purposes, we used the SI8610 digital isolator, with $sim10$ ns delay.

The inductor was implemented with a high-frequency structure (see [110] for details, as well as Chapter 2 in this thesis) using Fair-Rite 67, a magnetic material appropriate to the frequency range [106]. The prototype thus served as a platform to explore both the proposed topology and the magnetic structure in [110].

6.4 Experimental Results

Measured[4] efficiencies and input THD under varying load conditions show that the prototype achieves a combination of high performance, high frequency and high power quality (Fig. 6-7). The converter achieved ZVS across the line cycle for the full range

[4]Experiments were performed with Agilent 6813B as the ac source set to 220 $\mathrm{V_{rms}}$, with BK8522 as a resistive load. Efficiencies were obtained with two Instek GDM-8341 dual-measurement multimeters in this section, and with Yokogawa WT1800 for the four-switch prototype in the next section.

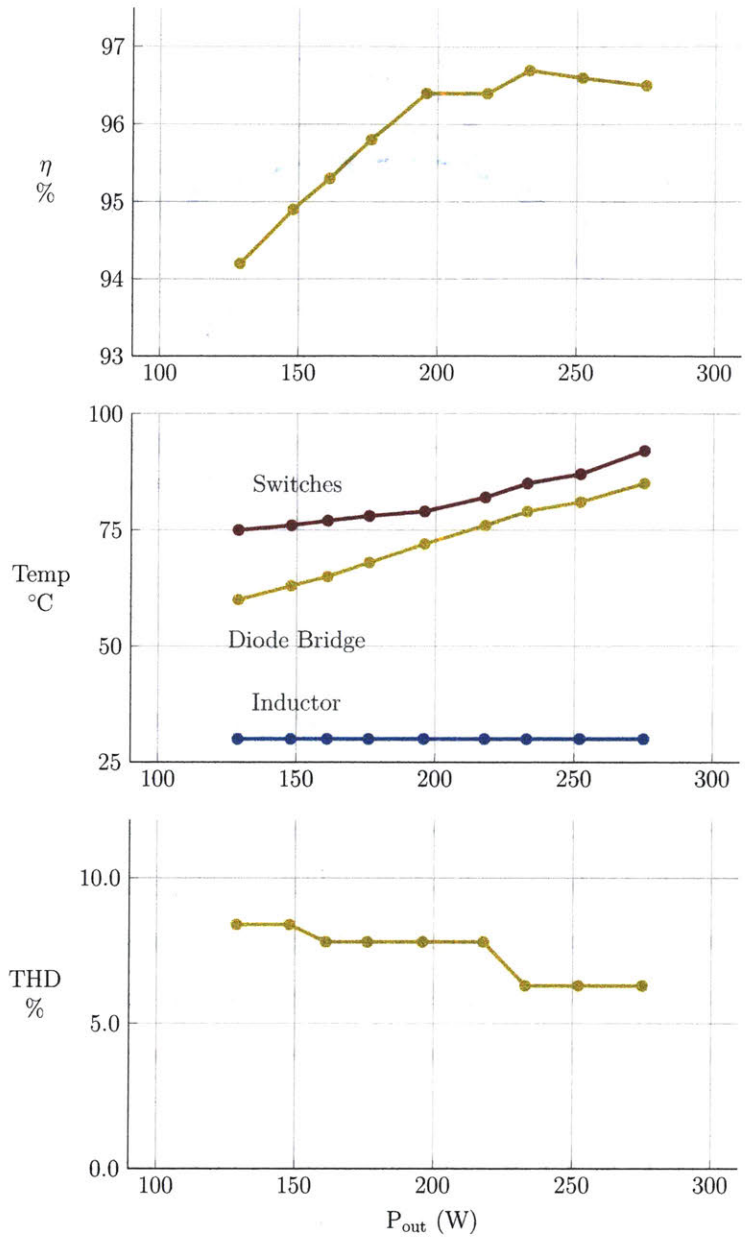


Figure 6-7: Measured prototype performance, showing high efficiency and high power quality. Experiments were performed at 220 Vrms input and 400 Vdc output maintained by low-bandwidth digital closed-loop control for a resistive load. Modest forced convection was applied to the switches, though the devices used are primarily bottom-side cooled.

in Appendix F, with the largest deviations at low power where model idealizations break down. These idealizations can be corrected with a more detailed model, if necessary.

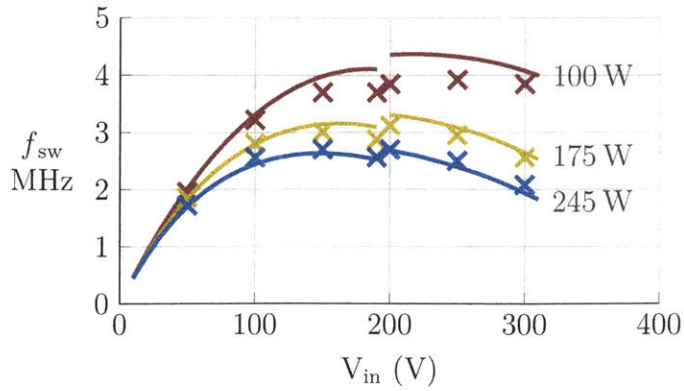


Figure 6-8: Instantaneous frequencies across the rising half of the line cycle, showing reasonable frequency variation across voltage/power. Measured values agree well with curves calculated from the model in Appendix F.

The prototype was not optimized for volume. The inductor size in particular was driven by concerns related to the experimental nature of its structure and hence difficulty/expense in prototyping. Nevertheless, thermal measurements (Fig. 6-7) indicate that the inductor has very low loss and temperature rise ($\Delta T < 5\,°\mathrm{C}$); we infer that the inductor volume could be greatly reduced without impacting thermal limits or efficiency. The high frequencies, small passive component values (e.g. $L = 13.5\,\mu\mathrm{H}$), and high efficiencies make it clear that the converter has potential for high density.

We also infer from Fig. 6-7 that the input diode bridge is a significant source of loss. This may appear to be a disadvantage compared to now-popular "bridgeless" PFC topologies; however, this loss can be largely mitigated by using active rectification [9]. Any apparent disadvantage should also be weighed judiciously against other factors; for example, converters with a front-end bridge may take advantage of smaller emi filter components on the rectified side (this becomes increasingly important with reductions in the power stage volume). Finally, the added control flexibilty and variety of accessible modes of the proposed converter allows the designer to meet a variety of demands, including high frequency with ZVS, tolerable frequency range, variable output voltage, etc.

6.5 Prototype with Low Output Voltage

A second prototype was developed with SA2 and SB2 implemented as active FETS instead of diodes. This prototype is capable of operating in the buck mode (with the ability to leave SB2 on). SA2 could still be implemented as a diode (to save on cost) but we implement it here as a FET to enable synchronous rectification and for resonant symmetry.

This prototype is similar to the previous one, with some differences. The FETs were implemented as Panasonic PGA26E07BA GaN devices with 56 mΩ of on-resistance. The input bridge was replaced with active FETs (STL36N55M5), which may be controlled as diodes or active devices for improved efficiency. See Appendices T, U, and V for schematics, layout, and microcontroller code.

The converter was operated with $V_{out} = 200\,\mathrm{V}$ instead of $V_{out} = 400\,\mathrm{V}$, which provides a number of advantages at the system level:

1. Any output switches of the PFC and any input switches of the second stage may take advantage of better-FOM, low-voltage devices

2. The volt-seconds applied to the inductor, for much of the cycle, will be substantially lower

3. The second stage transformation ratio will be greatly reduced, which is a great advantage since physical transformers and power converters exhibit reduced efficiency with large conversion ratios

4. Greater flexibilty is available regarding the voltage swing on the bus capacitor.

For this mode of operation, the converter passes through the following stages/-modes:

1. $V_{in} < V_{out}/2$: Boost mode

2. $V_{out}/2 < V_{in} < V_{out}$: Modified boost mode

3. $V_{in} \approx V_{out}$: Transition mode. This is nothing more than the modified boost mode where, in computation, we assume $V_{in} = V_{out}$ and compute the on-times accordingly. Calculations with $V_{in} > V_{out}$ in this mode otherwise yield complex results. This mode may be carried out for $V_{in} > V_{out}$ to a limited extent (i.e. for margin before the buck mode is engaged).

4. $V_{out} < V_{in} < 2V_{out}$: Buck mode

This prototype was also operated with a degree of synchronous rectification, as shown in Fig. 6-9 with efficiencies, power factors, and THDs as shown in Fig. 6-12, Fig. 6-13, and Fig. 6-14 respectively. The high-load efficiency of the converter is improved compared to the previous prototype, largely owing to the lower-resistance switches and synchronous rectification. The additional capacitance of more/larger FETs (as opposed to smaller FETs and diodes) does cause a longer resonant period each cycle, which tends to decrease light-load efficiency. Efficiencies for the lower voltage inputs (e.g. those used in the US and Japan) are forthcoming in future publications.

6.6 Conclusion

The proposed converter has been shown to be capable of achieving ZVS for any step-up voltage conversion ratio and for a large step-down conversion range with effective high frequency controls which require no current sensing, making it suitable for developing PFC converters operating at MHz frequencies. We validated the design approach and controls in a hardware prototype and demonstrated that the converter can maintain high efficiency ($\sim 98\,\%$) MHz switching frequencies, allowing small-valued passive components. We also show that a feedforward control approach can be used to meet IEC/EN 61000-3-2 input harmonic requirements, and even more stringent requirements for low THD.

In addition, the prototype highlights the potential offered by advanced magnetic materials [106] and design [110] when operated at high frequency. Converter perfor-

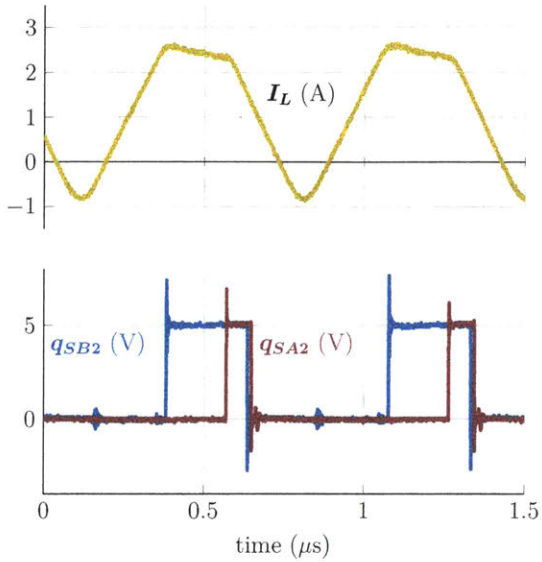


Figure 6-9: The technique of calculating on-times for the FETs is used here to achieve synchronous rectification. While margin is given here to ensure that the FET behaves as a diode, there is no danger of shoot-through even if a rectifying device stays on with some negative current as the main device will not turn on until its own ZVS condition is again achieved.

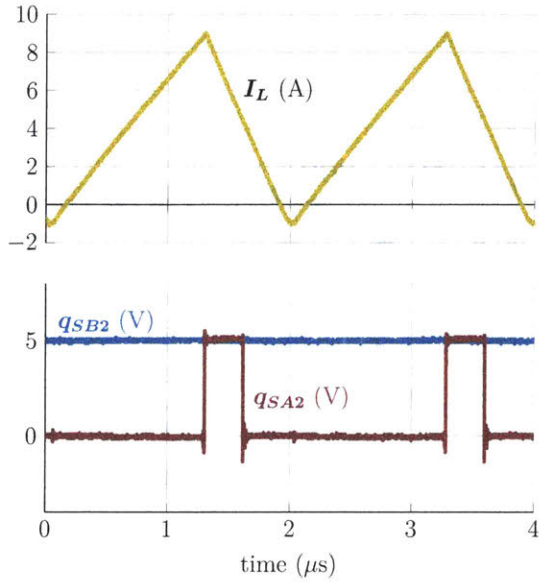


Figure 6-10: Experimental waveform for the buck mode demonstrating the inductor current and synchronous rectification.

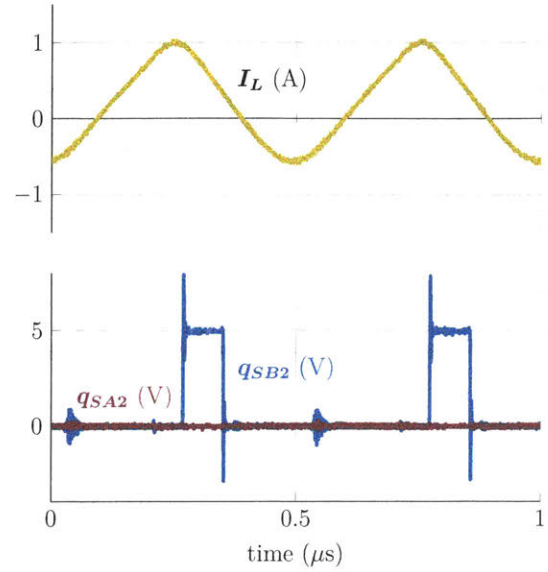


Figure 6-11: Experimental waveform for the boost mode demonstrating the inductor current and synchronous rectification.

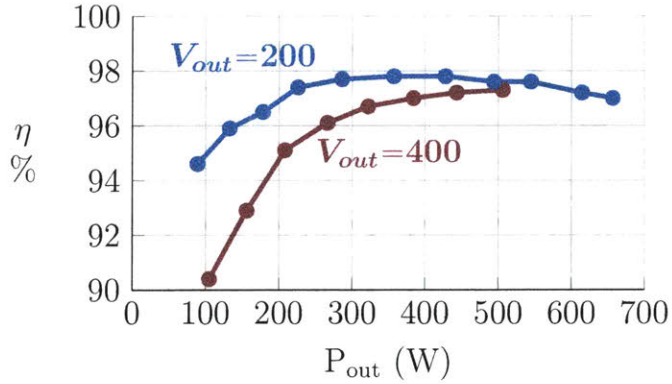


Figure 6-12: Efficiency for the full four-active-switch implementation of the proposed converter, operated with two different output voltages.

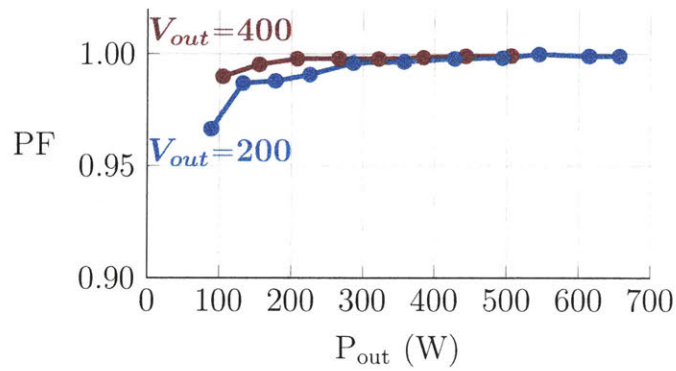


Figure 6-13: Power factor for the full four-active-switch implementation of the proposed converter, operated with two different output voltages.

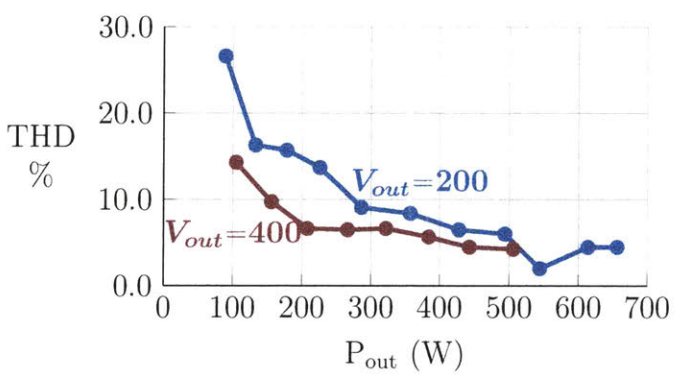


Figure 6-14: THD for the full four-active-switch implementation of the proposed converter, operated with two different output voltages.

mance may be improved further with refinements to wide-bandgap switch technology, which limits both the operating frequency and efficiency through C_{oss} and $R_{DS,on}$. Overall, we expect the opportunities enabled by this converter to improve the power density of PFC stages and EMI filters for grid-interface power converters.

Chapter 7

Conclusion

This thesis has presented a variety of contributions to high-frequency power conversion. While all may be understood in the context of grid-interface power conversion, their impacts extend beyond that context. High-frequency magnetics and sensing of the type described may benefit designs from the sub-watt to the kW range. Likewise, GaN transistors cover a similar space and may suffer from dynamic $R_{ds,on}$ and C_{oss} loss across the entire range. The operation of the PFC may be profitably applied to other converters dc-dc converters with wide operating ranges. The harmonic injection technique, while seemingly particular to the grid-interface context, is a useful way of thinking about buffering ac currents. Although implementation may need to be different, high-frequency rectification (as opposed to grid rectification) may benefit from similar thinking (e.g. in very high current applications where size and even R_{esr} may be concerns).

Appendix A

Low-loss inductor: Designing the Distributed Gap Geometry to Minimize Gap Fringing Loss

In this appendix, we show that setting the number of gaps N_g equal to the number of turns N aligns closely with the recommendation for minimizing gap fringing loss from [26], where the pitch between gaps (p) should be less than four times the spacing between the gap and the conductor (s), or $p < 4s$. We assume a large N so that the center-to-center spacing between each turn (p_w) can be approximated as $p_w = l_t/(N+1) \approx l_t/N$, where l_t is the window height. Setting $N_g = N$ also sets $p = p_w$, so the vertical and horizontal window fill are then

$$F_v = \frac{ND_w}{l_t} \approx \frac{D_w}{p} \qquad \text{(A.1)} \qquad\qquad F_h = \frac{D_w}{w} \qquad \text{(A.2)}$$

where D_w is the wire diameter and w is the window width (Fig. 2-1) Based on the geometry, the spacing between the gap and the winding is

$$s = \frac{w - D_w}{2} \qquad \text{(A.3)}$$

By combining (A.1)−(A.3), we get

$$\frac{p}{s} = \frac{2F_h}{F_v(1 - F_h)} \tag{A.4}$$

Most combinations of F_v and F_h within the recommended ranges (Sections 2.2.4 and 2.2.5) satisfy the design criteria from [26], $p < 4s$. For example, for values in the center of these ranges, $F_v = 0.65$ and $F_h = 0.50$, $p/s = 3.1 < 4$. At the edge of these ranges where F_v is small and F_h is large, the p/s ratio surpasses the recommendation of $p/s < 4$, with the worst case at $p/s = 6$, when $F_v = 0.50$ and $F_h = 0.60$. These edge cases, however, still achieve roughly optimal designs. Therefore, setting $N_g = N$ yields designs for the proposed inductor that meet (or nearly meet) the design criterion of [26] and thus, achieve roughly optimum Q.

Appendix B

Low-loss inductor: First-Order Derivation for Loss of the Active Section

In this appendix, we quantitatively show to first-order that the loss of the active section of the structure (everything within l_t) decreases as the diameter increases. To do this, we consider the equivalent resistance of the active section.

The winding resistance is

$$R_w = \rho N \frac{2\pi r_c}{A_1} = \rho k_w N^2 \frac{2\pi r_c}{l_t} \tag{B.1}$$

where it is assumed that the available conduction area of a turn A_1 is proportional to the area of the window with the proportionality constant $1/k_w$, i.e. $A_1 = l_t/(k_w N)$, and the radius of the winding path is approximated as the center post radius r_c.

For the equivalent resistance of the core loss in the active section, we consider only the center post core loss for simplicity, since the outer shell core loss is on the same order. The core loss in the center post is given by

$$P_{core} = k_c(l_t \pi {r_c}^2) \times \hat{B}^\beta = k_c(l_t \pi {r_c}^2) \times \left(\frac{LI_{pk}}{N\pi {r_c}^2}\right)^\beta \approx k_c(l_t \pi {r_c}^2) \times \left(\frac{LI_{pk}}{N\pi {r_c}^2}\right)^2 \tag{B.2}$$

where we approximate $\beta = 2$ (not uncommon for low frequencies; usually an underestimate at high frequencies). We may then express core loss through an equivalent resistance,

$$R_{core} = P_{core}/I_{rms}^2 = 2k_c l_t \frac{L^2}{N^2 \pi {r_c}^2} \tag{B.3}$$

In an optimized design, core loss is approximately equal to winding loss. Equating the resistances yields

$$N_{opt} = \left(\frac{k_c l_t^2 L^2}{\rho k_w \pi^2 {r_c}^3} \right)^{1/4} \tag{B.4}$$

$$R_w = R_{core} = \rho k_w {N_{opt}}^2 \frac{2\pi r_c}{l_t} = 2L\sqrt{\frac{\rho k_w k_c}{r_c}} \propto \sqrt{\frac{1}{r_c}} \tag{B.5}$$

so that the total equivalent resistance of the active section is proportional to $\sqrt{1/r_c}$. Thus, the loss in this section decreases as diameter increases.

Appendix C

Low-loss inductor: Prototype Construction

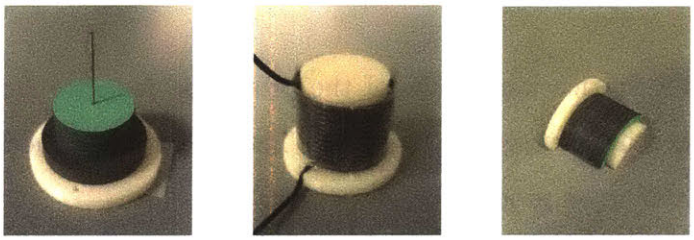
Below, we provide fabrication details of the prototype inductor from Section 2.5.1 for those interested in prototyping processes. The construction method below is not intended as a viable mass production process.

The prototype inductor was constructed modularly with the aid of custom 3D-printed fixtures. The center post was constructed first (Fig. C-1a) with one of the end caps. To control the quasi-distributed gaps, we stacked laser cut pieces of polyester plastic shimstock with the appropriate thickness (0.114 mm) in between each layer of core material. To center all of the layers of the centerpost, a 1 mm diameter hole was drilled in the center of the discs and the center post shimstock pieces so they could be assembled on a rod. Since the drilled holes were relatively small, we expect minimal effect on the fields.

For the winding, 20 AWG solid core wire with Teflon fluorinated ethylene propylene (FEP) insulation was wound around a 3D-printed fixture of the same diameter as the center post (Fig. C-1b). The wire was chosen to have the appropriate insulation thickness (0.229 mm from the conductor diameter to the outer diameter) to center it in the window. Then, the winding was wrapped in a single layer of 0.079 mm thick polypropylene tape (package sealing tape) to maintain its shape, removed from the fixture and put on the center post.

The outer shell, composed of three sections to allow for vertical windows (with approximate widths of 1.5 mm), was constructed one section at a time. Each section was stacked on a 3D-printed fixture, alternating between layers of core material and laser cut shimstock (Fig. C-1c). The outer surface of each section was taped to hold all the pieces together. Then, the sections were added to the center post structure so that the two winding terminations could leave the structure through one of the vertical windows in the shell.

Afterwards, the second end cap was added, and the rod was removed from the centerpost. Finally, the entire circumference of the inductor was wrapped with a single layer of package sealing tape to apply radial pressure, and a strip of package sealing tape was wrapped vertically around the inductor to apply vertical pressure.



(a) Center post (b) Winding (c) Outer shell section

Figure C-1: Construction of the prototype inductor using custom 3D-printed fixtures (white).

Appendix D

Low-loss inductor: Measuring High Q (large-signal)

To measure the large-signal Q of the prototype inductors, we used the same resonant measurement approach from [2] and [111] and added some modifications for measuring high Q. The original approach operates a series LC circuit at resonance so that the ratio of the peak capacitor voltage to the peak input voltage can be approximated as the Q of the inductor. When measuring a high Q, though, several assumptions in this approach no longer hold, leading to two modifications. Below, we discuss these modifications and other considerations for high-Q measurements at high frequency. We also show the validation of this modified measurement approach with an air-core inductor.

D.1 Use a capacitor divider to minimize probe loss and loading

When measuring a high-Q inductor, we expect a high resonant capacitor voltage. The probe loss and loading at this high-frequency, high-voltage node, however, can significantly affect results. To get a more accurate measurement of the resonant capacitor voltage, we replaced the capacitor in the original approach with a capacitor

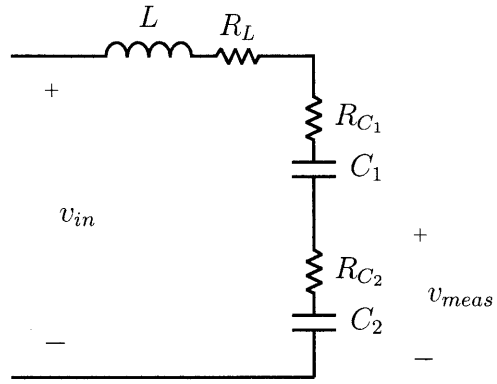


Figure D-1: Circuit for the resonant measurement approach to measure high Q, modified to include a capacitor divider to step down the measured output voltage. The capacitor ESRs are also included.

divider having the same net impedance. The stepped-down voltage can then be measured with minimal probe loss and loading (Fig. D-1).

D.2 Include capacitor ESR to accurately measure high Q

For measuring high Q, the approximation made in [2] and [111] that the equivalent series resistances (ESRs) of the capacitors (R_{C_1}, R_{C_2}) are small compared to the equivalent series resistance of the inductor (R_L) no longer holds, even with NP0, porcelain, or mica capacitors. For example, to measure an inductor with $Q = 1000$, using mica capacitors with $Q = 4000$ would still introduce a 25% loss error in the measurement.

Since the capacitor ESRs are no longer negligible, we include them in deriving an expression for the quality factor of the inductor (Q_L), using the measured input voltage v_{in} and stepped-down voltage v_{meas}. From Fig. D-1, we can see that at resonance, since the impedances of the inductor and capacitors cancel,

$$\frac{V_{meas,pk}}{V_{in,pk}} = \left| \frac{R_{C_2} + \frac{1}{j\omega_0 C_2}}{R_{C_1} + R_{C_2} + R_L} \right| \tag{D.1}$$

We also know that at resonance,

$$Q_L = \frac{\omega_0 L}{R_L} \tag{D.2}$$

From (D.1) and (D.2), the quality factor of the inductor as a function of $V_{in,pk}$ and $V_{meas,pk}$ is

$$Q_L = \frac{\omega_0 L}{\frac{V_{in,pk}}{V_{meas,pk}}\sqrt{{R_{C_2}}^2 + (\frac{1}{\omega_0 C_2})^2} - R_{C_1} - R_{C_2}} \tag{D.3}$$

where R_{C_1} and R_{C_2} are the ESR values found on the datasheet for the capacitors.[1]

The non-negligible capacitor ESR loss was validated thermally. At around 3 MHz, the highlighted mica capacitor in Fig. D-2 has an ESR of $\sim 0.07\,\Omega$, as extrapolated from the datasheet. With 2.0 A of current and a thermal resistance of 95 °C/W for the closest standard package size (2010) [112], we expect the capacitor to have a $\sim 13\,°C$ temperature rise, which agrees with the thermal image. This agreement confirms that the capacitor ESR loss can be predicted, and thus, corrected for when measuring the Q of the inductor.

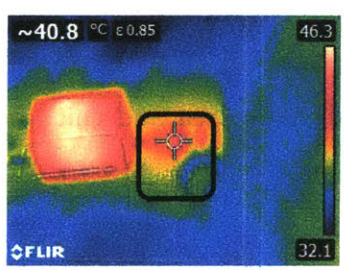


Figure D-2: Thermal image showing a mica capacitor (black box) with a $\sim 13\,°C$ temperature rise due to its ESR loss, in accordance with calculations.

[1] In cases where the capacitors are physically composed of multiple capacitors in parallel, the ESRs R_{C_1} and R_{C_2} can each be approximated as the equivalent parallel resistance for the corresponding ESRs. C_1 and C_2 can also be approximated as the equivalent parallel capacitance of the capacitors comprising it.

D.3 Minimize dielectric loss through careful board layout

For measuring a high-Q inductor, the node between the inductor and the capacitor divider sees a high voltage at HF. Therefore, in a layout where the return path runs directly under this high voltage node, the resulting parasitic capacitor can have non-negligible dielectric loss. We can mathematically show this by modeling the dielectric loss as the ESR loss of the parasitic capacitor. The dielectric loss is

$$P_{loss} = \frac{1}{2} V^2 \omega C \tan\delta \tag{D.4}$$

where V is the peak voltage at the node, ω is the measurement frequency, C is the parasitic capacitance, and $\tan\delta$ is the dielectric loss tangent of the board material.

As seen from (D.4), nodes with high voltages ($\sim 1000\,\mathrm{V}$) can have substantial dielectric loss, especially when their parasitic capacitance and the dielectric loss tangent of the board are relatively large. For example, for measuring the Q of the example inductor (Fig. 2-11), the high voltage node expects $\sim 600\,\mathrm{V}$ at $3\,\mathrm{MHz}$. If the node has an area of $1\,\mathrm{cm}^2$ on a standard 1.6 mm-thick FR-4 board ($\tan\delta = 0.02$), the dielectric loss at this node is then 160 mW, which is about 20 % of the inductor loss.

To minimize the dielectric loss, the capacitance at the high voltage node should be minimized by using a small node area and thick board. Board material with a lower dielectric loss tangent than FR-4, e.g. Rogers 4350B, can also be considered.

D.4 Resonant measurement approach validated using an air-core inductor

We validated the large-signal measurement approach described above with small-signal Q measurements of an air-core inductor. Since an air-core inductor has no nonlinear core loss, its small-signal and large-signal quality factors are the same. Using the equivalent series resistance of an air-core inductor measured at 3 MHz (with an

Agilent 4395A Impedance Analyzer and a custom resonant fixture), the small-signal quality factor of the inductor was calculated to be $Q = 540$ at this frequency. Using the large-signal resonant measurement approach, the same air-core inductor had a measured quality factor of $Q = 500$ at 3 MHz and 2 A (peak) of ac current, which validates this approach for measuring large-signal Q in this range. For even higher Q (>1000), sources of error have a greater impact on measurements, which makes it more difficult to accurately measure Q. The validation of the air-core inductor at $Q = 500$, however, indicates that it is possible to accurately measure even higher Q using this measurement approach.

Appendix E

GaN Measurement Implementation Details

The final component design and selection for the GaN characterization system are summarized in Table E.1. The choice of resonant components is particular to 3 MHz operation. The filter and IC component selection is not strongly a function of operating conditions; these components are found on the "main board" which is used in common across multiple devices and operating points.

Design of Resonant Components: In this approach, passive component design and selection are crucial to obtain high precision dynamic R_{on} measurements.

The resonant components L_r and C_r in Fig. 5-1 form a resonant tank. The component values must be chosen to satisfy a variety of constraints, including providing

Resonant Inductor		**Resonant Capacitor**	
L_r	2.5 µH	C_r	500 pF
Turn Diam.	0.25 in	C_r Type	C0G 1 kV
Turn Spacing	0.30 in		
Turns	5		
L_r Diam.	4.53 in	**Important ICs**	
L_r Length	2.45 in	Comparator	ADCMP601
Q	680	DAC	LTC2602

Table E.1: Implementation details for the experimental setup for 3 MHz operation

red pulse voltage peak and duration (or, equivalently, resonant frequency). They must also be chosen or designed such that the required current to generate the voltage pulse does not violate switch ratings but provides sufficient conduction loss to make P_{sw} the majority of the loss. An additional constraint is ensuring that C_r is large enough to be the dominant capacitance at the switch node (i.e. $C_r \gg C_{oss}$).

While the resonant component values influence the above metrics, the inductor quality factor Q (i.e. the specifics of its design) matters as well.[1] Inductor loss as a fraction of total loss depends both on the required current (determined by the L_r, C_r values) and the inductor's equivalent resistance (determined by its Q). The achievable quality factor is itself a function (albeit not analytically defined) of the inductor value, and this coupling makes optimization difficult.

To aid in design, we wrote a program to check various L_r and C_r pairs (for given inductor quality factor) and to highlight those combinations that satisfy all the constraints. The program output is a color map in the L_r, C_r plane, with acceptable and unacceptable regions (e.g. Fig. E-1). A generalized control-flow diagram of the program is provided in Fig. E-2.

For a given L_r and C_r, we calculate the required on-time and input voltage to achieve the desired pulse voltage and overall frequency. To achieve the correct overall period, the on-time is chosen to be

$$t_{on} = \frac{1}{2}\left(T - \frac{\pi}{\omega_r}\right) + \sqrt{\left(\frac{\pi}{\omega_r} - T\right)^2 - \frac{16}{\omega_r}} \tag{E.1}$$

With t_{on} determined, the dc input voltage required to generate the desired V_{pk} is

$$V_{in} = \frac{V_{pk}}{1 + \frac{t_{on}\omega_r}{2}} \tag{E.2}$$

We then analytically compute the inductor and switch rms currents. The rms

[1]The resonant capacitor quality factor Q_C may be important in principle; nevertheless, low-loss NP0/C0G ceramic, porcelain, and mica capacitors in the 500–5000 pF range are readily available with quality factors above 1000.

inductor current is found to be

$$I_{l,rms} = \sqrt{\frac{V_{in}^2 t_{on}^2}{12TL_r} + \frac{V_{in}^2 t_{on}^2}{16L_r^2}} \tag{E.3}$$

The rms switch current is also found to be

$$I_{sw,rms} = \sqrt{\frac{V_{in}^2 t_{on}^3}{12TL_r^2}} \tag{E.4}$$

We then estimate the losses in the FET based on a hypothesis of its dynamic R_{on} and the losses in the inductor based on a hypothesis of its Q; these hypotheses are subject to error, so the results of the program are understood to be estimates only. Nevertheless, although R_{on} is not known exactly and the achievable Q may not be known *a priori*, multiple plots can be generated for several values of R_{on} and Q to explore the design space.

The color regions are mapped as:

- Green → Acceptable - Passes all constraints
- Red → Device is expected to overheat
- Blue → switch loss P_{sw} does not represent the majority of total loss
- Purple → Intersection of Red and Blue case (i.e. device overheats *and* P_{sw} not majority of loss)

For a given device and frequency, an (L_r, C_r) point is chosen from the acceptable region, sufficiently far from any failure region as the region boundaries are calculated with a hypothesis for dynamic R_{on}. Nevertheless, L_r and C_r can be tuned afterwards to account for assumption errors.

After the L_r and C_r values are chosen for a given assumption about the inductor quality factor, it is important that L_r be implemented to achieve at least that quality factor. Since the inductor losses are likely to be substantial, it is also important that its quality factor be well characterized so its losses can be accurately accounted for.

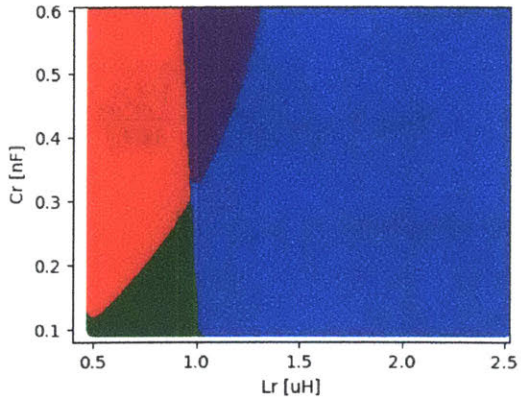


Figure E-1: Example program output categorizing L_r, C_r space into regions of varying acceptability. Green areas correspond to L_r and C_r pairs that satisfy all constraints. Other regions are color-coded by failure type. Simulation was done assuming an inductor Q of 600 and an operating frequency of 1 MHz.

To achieve these features, The resonant inductor is implemented as a large air-core solenoid (Fig. 5-4) for two reasons. First, air-core solenoids for the 1–10 MHz frequency range can achieve quality factors well above 500. Second, air-core inductors have linear resistance and accurate resistance measurements with an impedance analyzer can be appropriately extrapolated to large-signal conditions.

The design of high-Q air-core solenoids is well understood [113, 114]. A design obtained analytically (e.g. in [114]) can be verified with FEA tools or with available calculators which take into account a number of empirically-determined relationships for air-core solenoids (e.g. [115]). Using this approach, we obtained an inductor with $Q = 680$ at 3 MHz.

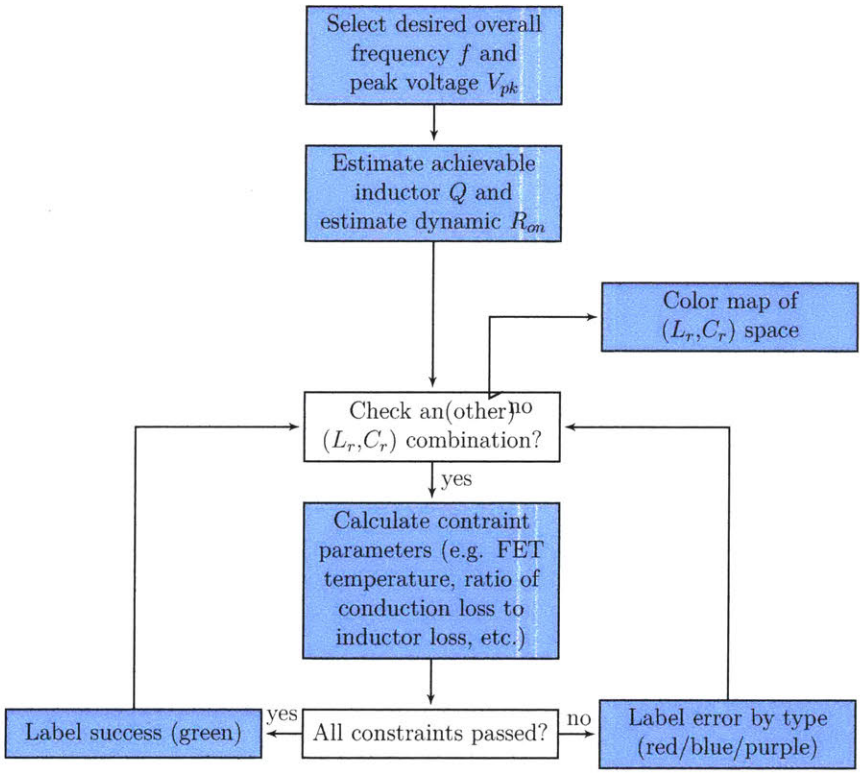


Figure E-2: Flowchart of the program for obtaining a rough estimate of L_r,C_r values. Several inductor Q can be considered, giving the designer a sense of how feasible the inductor implementation stage will be.

Appendix F

PFC Converter Analysis for Feedforward Control

One critical feature of the proposed converter is that the inductor current always returns to zero. From this point, based on constant circuit parameters and the instantaneous input and output voltages, the entire circuit behavior is predictable. The switch on-times can be computed to produce a desired inductor current waveform without actually sensing the inductor current. Indeed,the circuit need not even sense the converter input current for power factor correction, as even the average converter input current I_{conv} is calculable from an appropriately accurate model. This totally feed-forward approach avoids the need for current sensing altogether.

The remainder of this section outlines how to model the converter for this purpose. The end goal of the model is to compute the required switch on-times as functions of *only* constant circuit parameters and measureables the controller will already have. The results will be $t_{a,on}$ and $t_{b,on}$ as functions of

- the values of the inductor L, parasitic capacitors C_p, and input capacitor C_{in},
- the measured input voltage V_{in}, output voltage V_{out}, line frequency ω_{line} and line rms voltage V_{rms}, and
- the desired values of i_2 and I_{in} (I_{in} is varied over the line cycle for input current shaping, and varied in magnitude more slowly as part of the output voltage

feedback loop).

We remind the reader that capital symbols denote constants or averages across a switching cycle and lower-case symbols denote truly instantaneous values or those that only have meaning within a switching cycle. For compact notation, we also use $X = V_{in}/V_{out}$, $\omega_1 = 1/\sqrt{LC_p}$ for the LC resonant frequency and $\omega_2 = 1/\sqrt{LC_p/2}$ for the CLC resonant frequency.

F.1 Low Voltage Mode

We analyze the low-voltage mode in accordance with Fig. 6-2, assuming that i_1 is sufficient to charge C_p in negligible time after SB1 turns off. We wish to derive a model to relate the average input current I_{in} to the on-time of SB1. To do this, we first recognize that the input current I_{in} is not equal to the converter input current I_{conv}, but rather is the sum of I_{conv} and current into the input capacitance I_C. For simplicity, we model the input capacitance as a lumped linear sum of any EMI filter capacitor values. Using this simplified model, we compute the required I_{conv} in terms of the desired I_{in} and the I_C drawn by the input capacitance.

$$\begin{aligned} I_{conv} &= I_{in} - I_C \\ &= I_{in} - C_{in}\frac{dV_c}{dt} \\ &= I_{in} - C_{in}\omega_{line}\sqrt{2}V_{rms}\cos(\omega_{line}t) \end{aligned} \tag{F.1}$$

We can replace $\omega_{line}t = \sin^{-1}\left(\frac{V_{in}}{\sqrt{2}V_{rms}}\right)$ to make the above equation a function of instantaneous measurables instead,

$$\begin{aligned} I_{conv} &= I_{in} - C_{in}\omega_{line}\sqrt{2}V_{rms}\sqrt{1-\left(\frac{V_{in}}{\sqrt{2}V_{rms}}\right)^2} \\ &= I_{in} - C_{in}\omega_{line}\sqrt{2V_{rms}^2 - V_{in}^2} \end{aligned} \tag{F.2}$$

Next, we must express I_{conv} as a function of the control input $t_{b,on}$. To do this, we first compute the current at turn-on i_0 and the most negative inductor current

during resonance i_{min}. During resonance:

$$v_B = (V_{out} - V_{in})\cos(\omega_1 t) + V_{in} \tag{F.3}$$

$$i_L = i_{C_p} = C_p \frac{dv_B}{dt} = -C_p(V_{out} - V_{in})\omega_1 \sin(\omega_1 t) \tag{F.4}$$

The minimum inductor current is easily computed,

$$i_{min} = -C_p\omega_1(V_{out} - V_{in}) \tag{F.5}$$

The current at turn-on is related to the time t_0 from the beginning of resonance until the time that node B reaches zero volts and SB1 turns on, $t_0 = \frac{1}{\omega_1}\cos^{-1}\left(\frac{-V_{in}}{V_{out}-V_{in}}\right)$.

$$\begin{aligned} i_0 &= -C_p\omega_1(V_{out} - V_{in})\sin\left(\cos^{-1}\left(\frac{-V_{in}}{V_{out} - V_{in}}\right)\right) \\ &= -C_p\omega_1 V_{out}\sqrt{1-2X} \end{aligned} \tag{F.6}$$

By making a piecewise-linear approximation for the inductor current, we compute the average converter input current as:

$$I_{conv} \approx \frac{i_1 + i_{min}}{2} = \frac{1}{2}\left(i_0 + \frac{V_{in}t_{b,on}}{L} + i_{min}\right) \tag{F.7}$$

Plugging in for I_{conv}, i_0, and i_{min} and rearranging, we get

$$\begin{aligned} t_{b,on} = \quad & 2\frac{L}{V_{in}}I_{in} \\ & + \sqrt{LC_p}\frac{1}{X}\left(1 - X + \sqrt{1-2X}\right) \\ & \mp 2LC_{in}\omega_{line}\sqrt{2\left(\frac{V_{rms}}{V_{in}}\right)^2 - 1} \end{aligned} \tag{F.8}$$

where the $\mp$ depends on whether the input voltage is rising $(-)$ or falling $(+)$. Note

that this expression is easily interpreted: the first term is equivalent to the popular constant-on-time control used in true Boundary Conduction Mode; the second term corrects for the resonant transition time (significant at high frequency); the third term accounts for the effect of input capacitance.

F.2 Medium Voltage (Modified Boost) Mode

The model for the high voltage mode proceeds in a similar fashion. We take the input capacitance into account in the same way. We need only compute the relationship between the switch on-times and our design targets, I_{conv} and i_2. The average input current during the HV mode is given by:

$$\begin{aligned} I_{conv} &= \frac{1}{T}\left[\frac{1}{2}i_1 t_{es} + \frac{1}{2}(i_1 + i_2)t_{dir}\right] \\ &= \frac{1}{2}\frac{\frac{i_1^2}{V_{in}} - \frac{i_2^2}{V_{out}-V_{in}} + \frac{i_1^2}{V_{out}-V_{in}}}{\frac{i_1}{V_{in}} + \frac{i_1-i_2}{V_{out}-V_{in}} + \frac{i_2}{V_{out}} + \frac{t_{res2}}{L}} \\ &= \frac{i_2}{2}\frac{(i_1/i_2)^2 - X}{i_1/i_2 - (X)^2 + DX(1-X)} \end{aligned} \tag{F.9}$$

where $D = t_{res2}V_{out}/Li_2$ and $t_{res2} = \frac{1}{2}\frac{2\pi}{\omega_2} = \pi\sqrt{LC_p/2}$, recalling that we use $C_p/2$ because there are two such capacitors in series in the CLC case. The above equation reduces to a quadratic in i_1/i_2 which is easily solved,

$$\begin{aligned} i_1 = I_{conv} + \\ \sqrt{I_{conv}^2 + i_2^2 X - 2I_{conv}i_2X^2 + 2I_{conv}i_2DX(1-X)} \end{aligned} \tag{F.10}$$

Having set i_2 to maximize efficiency and i_1 to achieve the correct average converter input current, we only need to specify the on-times of the switches. To do this, we must understand when the switches turn on (equivalently, when the switch voltages reach zero). Switch SA1 will turn on first, with the inductor current equal to i_{a0} after

a time t_{res}

$$t_{res} = \frac{1}{\omega_2} \cos^{-1}(1 - 2X) \tag{F.11}$$

$$\begin{aligned} i_{a0} &= -\frac{1}{2} C V_{out} \omega_2 \sin\left(\cos^{-1}(1 - 2X)\right) \\ &= -C_p \omega_2 \sqrt{V_{out} V_{in} - V_{in}^2} \end{aligned} \tag{F.12}$$

Once SA1 turns on, the equivalent circuit changes from an undriven CLC resonant circuit to an LC resonant circuit with a low impedance input V_{in}. Taking t_{res} as an initial condition, we may use an energy argument to calculate i_{b0}:

$$\frac{1}{2} C_p \left[V_{in} - (V_{out} - V_{in})\right]^2 + \frac{1}{2} L i_{a0}^2 = \frac{1}{2} C_p V_{in}^2 + \frac{1}{2} L i_{b0}^2 \tag{F.13}$$

$$\begin{aligned} i_{b0} &= \sqrt{\frac{C_p}{L}(V_{in} - V_{out} + V_{in})^2 - \frac{C_p}{L} V_{in}^2 + i_{a0}^2} \\ &= \sqrt{\frac{C_p}{L}} V_{out} (1 - X) \end{aligned} \tag{F.14}$$

Finally, we may estimate $\Delta t = t_{b0} - t_{a0}$ by assuming the parasitic capacitance is discharged by an average current $\frac{1}{2}(i_{a0} + i_{b0})$ from its initial voltage $V_{out} - V_{in}$ to zero.

$$\begin{aligned} \Delta t &= (V_{out} - V_{in}) C_p / \left(\frac{i_{a0} + i_{b0}}{2}\right) \\ &= \frac{2\sqrt{2LC_p}\,(1 - X)}{\sqrt{X - (X)^2} + \sqrt{2}\,(1 - X)} \end{aligned} \tag{F.15}$$

Now, the on-time for SB1 is simply based on a linear inductor current ramp from i_{b0} to i_1,

$$t_{b,on} = L \frac{i_1}{V_{in}} + L \frac{V_{out}\sqrt{\frac{C_p}{L}}\,(1 - X)}{V_{in}} \tag{F.16}$$

And finally, the on-time for SA1 is simply equal to the on-time for SB1, plus the

direct delivery time, plus Δt.

$$t_{a,on} = t_{b,on} + L\frac{i_1 - i_2}{V_{out} - V_{in}} + \frac{2\sqrt{2LC_p}\,(1-X)}{\sqrt{X-(X)^2} + \sqrt{2}\,(1-X)} \tag{F.17}$$

F.3 Buck Mode

The analysis for the buck mode proceeds in identical fashion. The time t_0 from the moment SB2 turns off until the time ZVS is achieved on SA1, and the inductor current at that moment i_0, are given by:

$$t_0 = \frac{1}{\omega}\cos^{-1}\left(1 - \frac{V_{in}}{V_{out}}\right) \tag{F.18}$$

$$i_0 = -CV_{out}\omega\sin\left(\cos^{-1}\left(1 - \frac{V_{in}}{V_{out}}\right)\right) = -CV_{out}\omega\sqrt{X\,(2-X)} \tag{F.19}$$

The input current to the converter is then given by

$$I_{conv} = \frac{1}{T}\left[\frac{1}{2}\,(i_1 + i_0)\,t_{buck}\right] \tag{F.20}$$

which, when worked, yields a formula for the desired peak current i_1:

$$i_1 = I_{conv}X + \sqrt{I_{conv}^2X^2 - 2I_{conv}i_0 + i_0^2 + \frac{2I_{conv}t_0V_{out}(X-1)}{L}} \tag{F.21}$$

This equation at yields the necessary on-time for switch SA1:

$$t_a = \frac{(i_1 - i_0)L}{V_{in} - V_{out}} \tag{F.22}$$

If synchronous rectification is desired, the on-time for SA2 is also easily calculated. Since there is substantial current at the moment SA1 turns off, we assume that ZVS

is detected on SA2 after negligible time. The time SA2 must remain on is therefore:

$$t_{a2} = \frac{i_1 L}{V_{out}} \tag{F.23}$$

Since switches do not turn on until ZVS is achieved, there is no risk of shoot-through even if SA2 remains on too long due (e.g. due to nonlinearity in parameters or other mis-calculations). At worst, mis-commands of this sort slightly alter the average input current.

Appendix G

Harmonic injection: calculation method

To calculate the energy storage associated with any particular harmonic combination, we calculate how the stored energy changes over a cycle. The maximum minus the minimum energy gives the required energy storage in a cycle, $E_{store} = E_{peak} - E_{trough}$.

This computation is performed numerically in the following manner:

1. Specify the conditions of the test, including input voltage $V_{in,rms}$, net power P (which is equivalent to selecting $I_{1,rms}$), and the values of $I_{n,rms}$ for harmonics $n > 1$.

2. Compute half-cycle input current waveform through $i_{in}(t) = \sum_{n=1}^{N} I_n \sin(\omega t)$ where N is the highest order harmonic one wishes to consider.

3. Compute half-cycle input power waveform through $p(t) = i_{in}(t) \times v_{in}(t)$

4. Integrate the difference between the input power and the output power $E(t) = \int_0^{T/2} [p(t) - P_{out}]\, dt$, where $P_{out} = I_{1,rms} \times V_{in,rms} = \langle P_{in} \rangle$ is a constant and the integration may be performed numerically using e.g. cumtrapz in MATLAB/Octave. The energy storage requirement is the difference max(E)−min(E) and is normalized against the PF = 1 case.

5. Power factor is computed for each case with PF = $I_{1,rms}/\sqrt{\sum_{n=1}^{N} I_{n,rms}^2}$

This procedure is repeated by varying the test conditions (power level, harmonic content) within the specifications of the product class in question. For example, in Fig. 4-14, the third harmonic maximum is hard-coded based on Class C specifications, a vector of values for $I_{3,rms}$ is generated up to the hard-coded maximum, and the procedure above is followed for each values of the $I_{3,rms}$ vector. In most cases, harmonic content is swept. In some cases like Fig. 4-9, power is swept and the maximum allowable harmonic content must be computed in Step 1 for each power point. The above procedure is easily replicated across multiple variables to produce plots like Fig. 4-12.

Appendix H

Harmonic injection: control overview

There are a variety of control techniques that may be used to achieve the types of input currents discussed in Chapter 4, several of which are discussed in the references. The technique used here is a blended approach which uses feedforward to shape the input current waveform over the line cycle with a slower outer voltage feedback loop to set the output voltage (i.e. to control power). The feedforward line current shaping takes as inputs the desired harmonic rms currents as fractions of the fundamental, which itself is set by the slow outer voltage feedback loop. This approach is discussed in full in [74] and is not the main emphasis of this work. Nevertheless, we briefly outline the approach below.

We begin by observing that the average (over a switching cycle) line input current I_{in} is not equal to the converter input current I_{conv}, but rather is the sum of I_{conv} and current into any input capacitance I_C (e.g. in the EMI filter). Mathematically,

$$\begin{aligned} I_{conv} &= I_{in} - I_C \\ &= I_{in} - C_{in}\omega_{line}\sqrt{2}V_{rms}\cos(\omega_{line}t) \\ &= I_{in} - C_{in}\omega_{line}\sqrt{2}V_{rms}\sqrt{1-\left(\frac{V_{in}}{\sqrt{2}V_{rms}}\right)^2} \\ &= I_{in} - C_{in}\omega_{line}\sqrt{2V_{rms}^2 - V_{in}^2} \end{aligned} \tag{H.1}$$

where we have replaced $\omega_{line}t = \sin^{-1}\left(\frac{V_{in}}{\sqrt{2}V_{rms}}\right)$ to make the above equation a function of instantaneous measurables. If we can further express the converter input current I_{conv} in terms of control inputs, then we can implement feedforward control based on directly measurable quantities. I_{in} can be set to be any desired waveform, e.g. a fundamental sinusoid with harmonics.

The converter input current $I_{conv}(t_{on})$ is derived in [74] and solved to yield the control variable t_{on} as a function of the desired net input current I_{in}:

$$\begin{aligned} t_{on} = \quad & 2\frac{L}{V_{in}}I_{in} \\ & + \sqrt{LC_p}\frac{1}{X}\left(1 - X + \sqrt{1-2X}\right) \\ & \mp 2LC_{in}\omega_{line}\sqrt{2\left(\frac{V_{rms}}{V_{in}}\right)^2 - 1} \end{aligned} \tag{H.2}$$

where the $\mp$ depends on whether the input voltage is rising $(-)$ or falling $(+)$, C_p represents the parasitic capacitance at the switching node, and $X = V_{in}/V_{out}$. We interpret the first term as a constant on-time (for PF = 1, i.e. $I_{in} \propto V_{in}$) which is often used in Boundary Conduction Mode boost converters; the second term corrects for the resonant transition time (significant at high frequency); the third term accounts for the effect of input capacitance.

In this instance, we may set $I_{in} = \sqrt{2}I_{1,rms}\sin(\omega t) + \sqrt{2}I_{3,rms}\sin(3\omega t) + \ldots$ and replace $\omega_{line}t = \sin^{-1}\left(\frac{V_{in}}{\sqrt{2}V_{rms}}\right)$ as before, noting that $\sin(n \times \sin^{-1}(x))$ can be expressed as polynomials of x, e.g. $\sin(3 \times \sin^{-1}(x)) = 3x - 4x^3$ (such expressions are exact, not series approximations).

Thus, the control input t_{on} is continuously updated based on instantaneous measurements of input voltage and output voltage alone and can be made to conform with any desired waveform of the type discussed here.

Appendix I

Low-loss inductor: Simulation Details

This section includes complete simulation details for relevant tables and plots from Chapter 2. All simulations were done in ANSYS Maxwell 2D Design, Versions 18.2 and 19.2. The windings for all simulations used the default ANSYS model for copper ($\epsilon_r = 1$, $\mu_r = 0.999991$, $\sigma = 5.8 \times 10^{-7}$ S/m). For the simulations discussed in this appendix, all inductors were simulated at 3 MHz and 2 A (peak, ac).

For simulations that used Fair-Rite 67 material ($\mu_r = 40$), the following Steinmetz parameters were used to model core loss at 3 MHz: $k_c = 0.034$, $\alpha = 1.18$, and $\beta = 2.24$ (P_v in mW/cm^3, f in MHz, $\hat{B}$ in mT).

I.1 Simulation and Geometry Details for Fig. 2-2

The example simulations for single-sided and double-sided conduction in Fig 2-2 were close-up views of the middle three turns of a winding on a rod-core inductor. Both simulations used the same inductor geometry with the winding vertically centered on the rod core (Table I.1) but with different core permeabilities. The single-sided conduction simulation had a high permeability of $\mu_r = 1000$ for a lower H field on the inner side of the winding compared to the outer side. The double-sided conduction simulation had a low permeability of $\mu_r = 10$ for balanced H fields on the inner and

outer sides of the winding.

Rod Core Radius	1.5 mm
Rod Core Height	7.35 mm
Number of Turns	13
Winding Radius (to center of conductor)	1.84 mm
Wire Diameter	0.32 mm
Turn Spacing (center-to-center)	0.4 mm

Table I.1: Geometry of the simulated rod-core inductors in Fig. 2-2

I.2 Simulation and Geometry Details for Fig. 2-4

The simulated proposed inductor in Fig. 2-4 is the same as the example 16.6 µH design in Chapter 2; the geometry of this inductor is listed in Table 2.2. The simulated solenoid had roughly the same total height and radius as the simulated proposed inductor (Table I.2).

Solenoid Radius (to center of conductor)	13.45 mm
Solenoid Height (conductor edge-to-edge)	26 mm
Number of Turns	13
Wire Diameter	0.8 mm
Turn Spacing (center-to-center)	2.1 mm

Table I.2: Geometry of the simulated solenoid in Fig. 2-4

I.3 Simulation and Geometry Details for Fig. 2-5

In Fig. 2-5, a range of vertical window fills F_v was simulated for three different core geometries with the same volume ($14.4\,\mathrm{cm}^3$) but different window heights $l_t = \{14.3\,\mathrm{mm}, 18.3\,\mathrm{mm}, 22.3\,\mathrm{mm}\}$. Table I.3 lists dimensions for the three core geometries, and Table I.4 lists the winding diameters used to achieve the desired range of vertical window fills for each window height. The inductors all had the same inductance of 16.1 µH and used Fair-Rite 67 for the core material.

Window Height (l_t)	14.3 mm	18.3 mm	22.3 mm
Total Diameter ($2r_t$)	26.298 mm	26.298 mm	26.298 mm
Centerpost Radius (r_c)	6.106 mm	6.106 mm	6.106 mm
Window Width (w)	6.0 mm	6.0 mm	6.0 mm
Total Height (h_t)	26.3 mm	26.303 mm	26.311 mm
End Cap Height (h)	6.0 mm	4.0 mm	2.0 mm
Total Core Length (l_c)	13.832 mm	18.004 mm	22.246 mm
Total Gap Length (l_g)	0.468 mm	0.299 mm	0.065 mm
Number of Turns (N)	13	13	13
Number of Gaps (N_g)	13	13	13

Table I.3: Core geometry for the simulations in Fig. 2-5 at different window heights l_t

Vertical Fill (F_v)	Wire Diameter (D_w) for l_t = 14.3 mm	Wire Diameter (D_w) for l_t = 18.3 mm	Wire Diameter (D_w) for l_t = 22.3 mm
0.3	0.33 mm	0.40 mm	0.53 mm
0.4	0.44 mm	0.55 mm	0.70 mm
0.5	0.55 mm	0.70 mm	0.87 mm
0.6	0.66 mm	0.85 mm	1.04 mm
0.7	0.77 mm	1.00 mm	1.21 mm
0.8	0.88 mm	1.15 mm	1.38 mm
0.9	0.99 mm	1.30 mm	1.55 mm

Table I.4: Wire diameters for a range of vertical window fills at different window heights

I.4 Simulation and Geometry Details for Fig. 2-7

In Fig. 2-7, a range of horizontal window fills F_h was simulated for five different core geometries with the same volume (14.4 cm^3) but different window widths $w = \{1.00\,\text{mm}, 1.50\,\text{mm}, 1.75\,\text{mm}, 2.00\,\text{mm}, 3.17\,\text{mm}\}$. Each core geometry had the same number of turns ($N = 13$) but different winding diameters $D_w = \{0.7\,\text{mm}, 0.9\,\text{mm}, 1.1\,\text{mm}\}$ to achieve different vertical window fills $F_v = \{50\%, 64\%, 78\%\}$, respectively. The inductors all had the same inductance of 16.5 µH and used Fair-Rite 67 for the core material. Table I.5 lists dimensions for the five core geometries, and Table I.6 provides the achieved F_h for each combination of winding diameter and window width.

Window Width (w)	1.00 mm	1.50 mm	1.75 mm	2.00 mm	3.17 mm
Total Diameter ($2r_t$)	26.3 mm	26.3 mm	26.3 mm	26.3 mm	26.3 mm
Centerpost Radius (r_c)	9.5 mm	9.19 mm	9.03 mm	8.875 mm	8.11 mm
Total Height (h_t)	26.311 mm	26.307 mm	26.312 mm	26.303 mm	26.31 mm
End Cap Height (h)	4.0 mm	4.0 mm	4.0 mm	4.0 mm	4.0 mm
Total Core Length (l_c)	16.842 mm	16.968 mm	17.038 mm	17.094 mm	17.374 mm
Total Gap Length (l_g)	1.469 mm	1.339 mm	1.274 mm	1.209 mm	0.936 mm
Number of Turns (N)	13	13	13	13	13
Number of Gaps (N_g)	13	13	13	13	13

Table I.5: Core geometry for the simulations in Fig. 2-7 at different window widths

	$D_w = 0.7$ mm ($F_v = 50\%$)	$D_w = 0.9$ mm ($F_v = 64\%$)	$D_w = 1.1$ mm ($F_v = 78\%$)
$w = 1.00$ mm	$F_h = 22\%$	$F_h = 28\%$	$F_h = 35\%$
$w = 1.50$ mm	$F_h = 35\%$	$F_h = 45\%$	$F_h = 44\%$
$w = 1.75$ mm	$F_h = 40\%$	$F_h = 51\%$	$F_h = 55\%$
$w = 2.00$ mm	$F_h = 47\%$	$F_h = 60\%$	$F_h = 63\%$
$w = 3.17$ mm	$F_h = 70\%$	$F_h = 90\%$	$F_h = 73\%$

Table I.6: Combinations of wire diameters D_w and window widths w to achieve a range of horizontal window fills F_h at different vertical window fills F_v

I.5 Simulation and Geometry Details for Fig. 2-8

In Fig. 2-8, inductors with a range of different height-to-diameter aspect ratios (h/D) were simulated at three different volumes ($7\,\text{cm}^3, 14\,\text{cm}^3, 28\,\text{cm}^3$). For each inductor, all geometric parameters, except for the aspect ratio, were designed using the guidelines from Section 2.2. The inductors had the same inductance (16.6 µH) and used Fair-Rite 56 for the core material. Tables I.7, I.8, and I.9 list geometry details for the simulated inductors with different aspect ratios at volumes of $7\,\text{cm}^3$, $14\,\text{cm}^3$, and $28\,\text{cm}^3$, respectively.

Aspect Ratio (h/D)	0.33	0.67	1.0
Total Diameter ($2r_t$)	29.902 mm	23.734 mm	20.734 mm
Centerpost Radius (r_c)	10.658 mm	8.401 mm	7.303 mm
Window Width (w)	0.663 mm	0.882 mm	1.098 mm
Total Height (h_t)	9.989 mm	15.821 mm	20.73 mm
End Cap Height (h)	2.5 mm	3.5 mm	3.5 mm
Total Core Length (l_c)	4.17 mm	7.813 mm	12.56 mm
Total Gap Length (l_g)	0.819 mm	1.008 mm	1.17 mm
Number of Turns (N)	9	12	15
Number of Gaps (N_g)	9	12	15
Wire Diameter (D_w)	0.331 mm	0.442 mm	0.55 mm

Aspect Ratio (h/D)	1.33	2.0	4.0
Total Diameter ($2r_t$)	18.838 mm	16.456 mm	13.061 mm
Centerpost Radius (r_c)	6.675 mm	5.787 mm	4.508 mm
Window Width (w)	1.208 mm	1.413 mm	1.752 mm
Total Height (h_t)	25.096 mm	32.885 mm	52.211 mm
End Cap Height (h)	3.5 mm	3.5 mm	3.5 mm
Total Core Length (l_c)	16.53 mm	24.081 mm	43.072 mm
Total Gap Length (l_g)	1.566 mm	1.804 mm	2.139 mm
Number of Turns (N)	18	22	31
Number of Gaps (N_g)	18	22	31
Wire Diameter (D_w)	0.604 mm	0.706 mm	0.876 mm

Table I.7: Geometries for simulated inductors with different aspect ratios at a volume of 7 cm^3 from Fig. 2-8

Aspect Ratio (h/D)	0.33	0.67	1.0
Total Diameter ($2r_t$)	37.942 mm	30.114 mm	26.3 mm
Centerpost Radius (r_c)	13.626 mm	10.815 mm	9.475 mm
Window Width (w)	0.697 mm	1.207 mm	1.465 mm
Total Height (h_t)	12.671 mm	20.081 mm	26.303 mm
End Cap Height (h)	4.0 mm	4.0 mm	4.0 mm
Total Core Length (l_c)	3.447 mm	10.257 mm	16.128 mm
Total Gap Length (l_g)	1.224 mm	1.824 mm	2.175 mm
Number of Turns (N)	8	12	15
Number of Gaps (N_g)	8	12	15
Wire Diameter (D_w)	0.348 mm	0.604 mm	0.732 mm

Aspect Ratio (h/D)	1.33	2.0	4.0
Total Diameter ($2r_t$)	23.902 mm	20.88 mm	16.572 mm
Centerpost Radius (r_c)	8.536 mm	7.473 mm	5.784 mm
Window Width (w)	1.685 mm	1.929 mm	2.412 mm
Total Height (h_t)	31.855 mm	41.719 mm	66.238 mm
End Cap Height (h)	4.0 mm	4.0 mm	4.0 mm
Total Core Length (l_c)	21.492 mm	30.8 mm	54.99 mm
Total Gap Length (l_g)	2.363 mm	2.919 mm	3.248 mm
Number of Turns (N)	17	21	29
Number of Gaps (N_g)	17	21	29
Wire Diameter (D_w)	0.842 mm	0.964 mm	1.206 mm

Table I.8: Geometries for simulated inductors with different aspect ratios at a volume of 14 cm^3 from Fig. 2-8

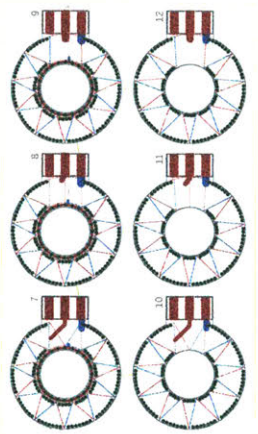
Aspect Ratio (h/D)	0.33	0.67	1.0
Total Diameter ($2r_t$)	47.468 mm	37.674 mm	32.912 mm
Centerpost Radius (r_c)	16.946 mm	13.392 mm	11.778 mm
Window Width (w)	1.323 mm	1.758 mm	2.05 mm
Total Height (h_t)	15.84 mm	25.1 mm	32.882 mm
End Cap Height (h)	3.5 mm	4.5 mm	4.5 mm
Total Core Length (l_c)	6.984 mm	13.68 mm	20.76 mm
Total Gap Length (l_g)	1.856 mm	2.42 mm	3.122 mm
Number of Turns (N)	8	11	14
Number of Gaps (N_g)	8	11	14
Wire Diameter (D_w)	0.662 mm	0.879 mm	1.024 mm

Aspect Ratio (h/D)	1.33	2.0	4.0
Total Diameter ($2r_t$)	29.902 mm	26.122 mm	20.734 mm
Centerpost Radius (r_c)	10.342 mm	9.081 mm	6.789 mm
Window Width (w)	2.47 mm	2.731 mm	3.412 mm
Total Height (h_t)	39.835 mm	52.207 mm	82.874 mm
End Cap Height (h)	4.5 mm	4.5 mm	4.5 mm
Total Core Length (l_c)	28.24 mm	39.92 mm	70.416 mm
Total Gap Length (l_g)	2.595 mm	3.287 mm	2.458 mm
Number of Turns (N)	15	19	26
Number of Gaps (N_g)	15	19	26
Wire Diameter (D_w)	1.234 mm	1.366 mm	1.706 mm

Table I.9: Geometries for simulated inductors with different aspect ratios at a volume of 28 cm^3 from Fig. 2-8

Appendix J

Rogowski Coil Layout


7
8
9
10
11
12

Appendix K

Harmonic Injection Analysis Code

```
close all;
clear all;
clc;

modnum = 2;  %Roughly, the size of data points to be included in output CSV.  Larger number yields coarser data

P = 20;   %Output power; convenient to assign a value, but irrelevant as all results are ratiometric
Vinrms = 220;  %Input voltage, expressed in rms
f = 50; %Input frequency, Hz
w = 2*pi*f; %Input frequency, rad/sec
T = 1/f;  %Line period, sec




%Class C low power case

I1rms = P/Vinrms;  %Fundamental current, in Arms

limit3rms = 0.86 * I1rms;  %Limit to 3rd harmonic in EN61000-3-2 for Class C sub 25 watt
limit5rms = 0.61 * I1rms;  %Limit to 5th harmonic
limit7rms = I1rms;  %No limit for 7th harmonic; code will only consider values up to I7rms = I1rms
limit9rms = I1rms;  %Same for 9th
limit11rms =I1rms;  %Same for 11th


dt = 10^-5;  %Time step for time-domain calculations
t = 0:dt:T/2;  %Time vector based on time step.  Symmetry ensures that only up to T/2 must be considered
percent = [1:1:100]/100;   %Code will cycle through this vector to compute various outputs for a given percentage of included harmonic current
i35 = zeros(length(percent),length(t));  %Matrix to contain time-domain current with 3rd and 5th harmonics included at various percentages
p35 = zeros(length(percent),length(t));  %Matrix of time-domain powers
buffer35 = zeros(1,length(percent));     %Vector of energy buffering required for certain percentages of 3rd and 5th harmonics
pf35 = zeros(1,length(percent));         %Vector of power factors

i3  = zeros(length(percent),length(t));  %Matrix of time-domain currents with only 3rd harmonic included at various percentages
p3  = zeros(length(percent),length(t));  %Matrix of powers
pf3 = zeros(1,length(percent));          %Vector of power factors


inobuffer = P./(sqrt(2)*Vinrms*sin(w*t));  %Current that would result in no buffer, for comparison
pnobuffer = P * ones(1,length(t));         %Power that would result in no buffer, for comparison



%For any percentage harmonic inclusion, figure out the energy buffering, power factor
for n = 1:length(percent)
I3rms = percent(n) * limit3rms;
I5rms = percent(n) * limit5rms;
I7rms = percent(n) * limit7rms;
I9rms = percent(n) * limit9rms;


%Construct the time-domain current, and then compute instantaneous power
i35(n,:) = sqrt(2)*I1rms*sin(w*t) + sqrt(2)*I3rms*sin(3*w*t) + sqrt(2)*I5rms*sin(5*w*t);
p35(n,:) = i35(n,:) .* sqrt(2) .* Vinrms .* sin(w*t);

%Compute energy storage.  Note that it is important to do the full integral in q35,
%   as minor deviations up and down in stored energy may not contribute to the
%   peak-to-peak energy storage requirement
q35 = cumtrapz(pnobuffer-p35(n,:));  %Integral of power
buffer35(n) = max(q35) - min(q35);  %Energy storage
pf35(n) = I1rms / sqrt(I1rms^2 + I3rms^2 + I5rms^2);
[a b] = max(i35(n,[1:500]));  %Checking for max and min rules
fivepercent = 0.05*a;
if (  (b*dt)>(65/360)*T  ) && (n!=1)
```

```
  buffer35(n) = NaN;    %Violating this condition doesn't count in EN61000-3-2
endif


%Do the same if only 3rd harmonic is included
i3(n,:) = sqrt(2)*I1rms*sin(w*t) + sqrt(2)*I3rms*sin(3*w*t);
p3(n,:) = i3(n,:) .* sqrt(2) .* Vinrms .* sin(w*t);
%pcrossing = find( (pnobuffer-p3(n,:)) < 0 , 1);
%buffer3(n) = 2*sum(pnobuffer(1:pcrossing) - p3(n,1:pcrossing)) * dt;
q3 = cumtrapz(pnobuffer-p3(n,:));
buffer3(n) = max(q3) - min(q3);
pf3(n) = I1rms / sqrt(I1rms^2 + I3rms^2);
[a b] = max(i3(n,[1:500]));  %Checking for max and min rules
fivepercent = 0.05*a;
if (  (b*dt)>(65/360)*T  ) && (n!=1)
  buffer3(n) = NaN;
endif

endfor

%Plot the energy buffered, normalized
figure
hold on
plot(percent, buffer3./buffer3(1));
plot(percent, buffer35./buffer35(1));

%Plot the power factors
figure
hold on
plot(percent, pf3);
plot(percent, pf35);
title('Power Factor')

%Print the data.  Not ultimately used in the paper
fid = fopen('c_low2_35.csv','w');
fprintf(fid,'%s, %s, %s, %s, %s\n','percent','buffer3','buffer35','pf3','pf35');
for i = 1:length(percent)
 fprintf(fid,'%f, %f, %f, %f, %f\n',100*percent(i),buffer3(i)./buffer35(1),buffer35(i)./
 buffer35(1),pf3(i),pf35(i));
 endfor
fclose(fid);



%Since adding 3rd and 5th harmonic to Class C low power only helps,
%Keep going and add 7th and 9th and see what happens

%Values for when "all" harmonics are included (3rd through 9th)
iall  = zeros(length(percent),length(t));
pall  = zeros(length(percent),length(t));
bufferall = zeros(length(percent),length(percent));
pfall = zeros(length(percent),length(percent));



for n = 1:length(percent)  %Cycle 7th harmonic inclusion
  for m = 1:length(percent)  %Cycle 9th harmonic inclusion
      I7rms = percent(n) * limit7rms;
      I9rms = percent(m) * limit9rms;
      iall = sqrt(2)*I1rms*sin(w*t) + sqrt(2)*I3rms*sin(3*w*t) + sqrt(2)*I5rms*sin(5*w*t) +
      sqrt(2)*I7rms*sin(7*w*t) + sqrt(2)*I9rms*sin(9*w*t);
      pall = iall .* sqrt(2) .* Vinrms .* sin(w*t);
      qall = cumtrapz(pnobuffer-pall);
      bufferall(n,m) = max(qall) - min(qall);
      pfall(n,m) = I1rms/sqrt(I1rms^2 + I3rms^2 + I5rms^2 + I7rms^2 + I9rms^2);
      [a b] = max(iall([1:500]));  %Checking for max and min rules
      [c d] = max(iall([100:500])); %Checking for max and min rules

      fivepercent = 0.05*a;
      if (  (b*dt)>(65/360)*T  ) || ( c < fivepercent)
```

```
        bufferall(n,m) = NaN
      endif


  endfor
endfor


%Plot energy storage
figure
colormap(flipud(rainbow));
imagesc(bufferall./buffer35(1));
colorbar;
xlabel('9th');
ylabel('7th');
%colormap(default);

%Plot power factor
figure
colormap(flipud(rainbow));
imagesc(pfall);
colorbar;
xlabel('9th');
ylabel('7th');
title('Power Factor, Low Power Class C')
%colormap(default);


%Print data for publication
fid = fopen('c_low2_79.csv','w');
%fprintf(fid,'%s, %s, %s\n','seventh','ninth','buffer');
for i = 1:length(percent)
  for j = 1:length(percent)
    if (mod(i,modnum) == 0) && (mod(j,modnum) == 0)
      fprintf(fid,'%.2f, %.2f, %.2f\n',i,j,bufferall(i,j)./buffer35(1));
    endif
  endfor
  fprintf(fid,'\n');
endfor
fclose(fid);

%Print power factor data for publication
fid = fopen('c_low2_PF79.csv','w');
%fprintf(fid,'%s, %s, %s\n','seventh','ninth','buffer');
for i = 1:length(percent)
  for j = 1:length(percent)
    if (mod(i,modnum) == 0) && (mod(j,modnum) == 0)
      fprintf(fid,'%.2f, %.2f, %.2f\n',i,j,pfall(i,j));
    endif
  endfor
  fprintf(fid,'\n');
endfor
fclose(fid);




%  Now for Class C above 25 watts

P = 20;
Vinrms = 220;
f = 50;
w = 2*pi*f;
T = 1/f;


I1rms = P/Vinrms;


```

```
limit5rms = 0.10 * I1rms;
limit7rms = 0.07 * I1rms;
limit9rms = 0.05 * I1rms;


inobuffer = P./(sqrt(2)*Vinrms*sin(w*t));
pnobuffer = P * ones(1,length(t));


for n = 1:length(percent)
  for m = 1:length(percent)
      I5rms = percent(n) * limit5rms;
      I7rms = percent(m) * limit7rms;
      p5 = I5rms/I1rms;
      p7 = I7rms/I1rms;

      %The maximum allowable 3rd harmonic is a function of power factor
      %Therefore, first choose a value of 5th and 7th harmonic to investigate
      %    then figure out how much 3rd can be included within EN61000-3-2 specs
      p3 = sqrt( 0.5 * ( -(p5^2+p7^2+1)+sqrt((p5^2+p7^2+1)^2+4*.3^2)));
      I3rms = p3*I1rms;

      iall = sqrt(2)*I1rms*sin(w*t) + sqrt(2)*I3rms*sin(3*w*t) + sqrt(2)*I5rms*sin(5*w*t) +
      sqrt(2)*I7rms*sin(7*w*t);
      pall = iall .* sqrt(2) .* Vinrms .* sin(w*t);
      qall = cumtrapz(pnobuffer-pall);
      bufferall(n,m) = max(qall) - min(qall);
      pfall(n,m) = I1rms / sqrt(I1rms^2 + I3rms^2 + I5rms^2 + I7rms^2);

  endfor
endfor


figure
colormap(flipud(rainbow));
imagesc(bufferall./buffer35(1));
colorbar;
xlabel('7th');
ylabel('5th');
%colormap(default);


figure
colormap(flipud(rainbow));
imagesc(pfall);
colorbar;
xlabel('7th');
ylabel('5th');
title('power factor Class C high power')
%colormap(default);


fid = fopen('c_high_57.csv','w');
%fprintf(fid,'%s, %s, %s\n','fifth','seventh','buffer');
for i = 1:length(percent)
  for j = 1:length(percent)
    if (mod(i,modnum) == 0) && (mod(j,modnum) == 0)
      fprintf(fid,'%.2f, %.2f, %.2f\n',i,j,bufferall(i,j)./buffer35(1));
    endif
  endfor
  fprintf(fid,'\n');
endfor
fclose(fid);

fid = fopen('c_high_PF57.csv','w');
%fprintf(fid,'%s, %s, %s\n','fifth','seventh','buffer');
for i = 1:length(percent)
  for j = 1:length(percent)
    if (mod(i,modnum) == 0) && (mod(j,modnum) == 0)
      fprintf(fid,'%.3f, %.3f, %.3f\n',i,j,pfall(i,j) );
```

```
    endif
  endfor
  fprintf(fid,'\n');
endfor
fclose(fid);



%Class B


P = [200:10:1600];
Vrms = 220;
I1rms = P./Vrms;

%Class B limits are 1.5 times class A limits, not a function of power
limit3rms = 2.3*1.5;
limit5rms = 1.14*1.5;
limit7rms = 0.77*1.5;
limit9rms = 0.40*1.5;
limit11rms= 0.33*1.5;
limit13rms= 0.21*1.5;
limit15rms= 0.15*1.5;
limit17rms= 0.15*15/17*1.5;
limit19rms= 0.15*15/19*1.5;
limit21rms= 0.15*15/21*1.5;
limit23rms= 0.15*15/23*1.5;
limit25rms= 0.15*15/25*1.5;


%i3 = zeros(length(P),length(t));
%iall = zeros(length(P),length(t));
buffer3 = zeros(1,length(P));
bufferall = zeros(1,length(P));
pf3 = zeros(1,length(P));
pfall = zeros(1,length(P));
Ezero = P/w;

%Cycle through power to obtain limits for Class B
for n = 1:length(P);
  pnobuffer = P(n) * ones(1,length(t));
  I1rms = P(n)/Vrms;

  %Limit every harmonic to be less than the fundamental
  if I1rms < limit3rms
    I3rms = I1rms;
  else
    I3rms = limit3rms;
  endif

  if I1rms < limit5rms
    I5rms = I1rms;
  else
    I5rms = limit5rms;
  endif

  if I1rms < limit7rms
    I7rms = I1rms;
  else
    I7rms = limit7rms;
  endif

  if I1rms < limit9rms
    I9rms = I1rms;
  else
    I9rms = limit9rms;
  endif

  if I1rms < limit11rms
```

```
    I11rms = I1rms;
  else
    I11rms = limit11rms;
  endif

  if I1rms < limit13rms
    I13rms = I1rms;
  else
    I13rms = limit13rms;
  endif

  if I1rms < limit15rms
    I15rms = I1rms;
  else
    I15rms = limit15rms;
  endif

  if I1rms < limit17rms
    I17rms = I1rms;
  else
    I17rms = limit17rms;
  endif

  if I1rms < limit19rms
    I19rms = I1rms;
  else
    I19rms = limit19rms;
  endif

  if I1rms < limit21rms
    I21rms = I1rms;
  else
    I21rms = limit21rms;
  endif

  if I1rms < limit23rms
    I23rms = I1rms;
  else
    I23rms = limit23rms;
  endif
  if I1rms < limit25rms
    I25rms = I1rms;
  else
    I25rms = limit25rms;
  endif

  %Consider effects with just 3rd harmonic, or with all harmonics
  i3 = I1rms*sqrt(2)*sin(w*t) + I3rms*sqrt(2)*sin(3*w*t);
  p3 = i3.*sqrt(2).*Vrms.*sin(w*t);
  q3 = cumtrapz(pnobuffer-p3)*dt;
  buffer3(n) = max(q3) - min(q3);
  pf3(n) = I1rms/sqrt(I1rms^2 + I3rms^2);

  iall = I1rms*sqrt(2)*sin(w*t) + I3rms*sqrt(2)*sin(3*w*t)+ I5rms*sqrt(2)*sin(5*w*t)+
  I7rms*sqrt(2)*sin(7*w*t)+ I9rms*sqrt(2)*sin(9*w*t)+ I11rms*sqrt(2)*sin(11*w*t)+
  I13rms*sqrt(2)*sin(13*w*t)+ I15rms*sqrt(2)*sin(15*w*t)+ I17rms*sqrt(2)*sin(17*w*t)+
  I19rms*sqrt(2)*sin(19*w*t)+ I21rms*sqrt(2)*sin(21*w*t)+ I23rms*sqrt(2)*sin(23*w*t)+
  I25rms*sqrt(2)*sin(25*w*t);
  pall = iall.*sqrt(2).*Vrms.*sin(w*t);
  qall = cumtrapz(pnobuffer-pall)*dt;
  bufferall(n) = max(qall) - min(qall);
  pfall(n) = I1rms/sqrt(I1rms^2 + I3rms^2 + I5rms^2 + I7rms^2 + I9rms^2 + I11rms^2 + I13rms^2 + I15rms^2
  + I17rms^2 + I19rms^2 + I21rms^2 + I23rms^2 + I25rms^2);

  if (P(n) == 200)
    P(n)
    Vrms
    I1rms
    I3rms
    I5rms
```

```
    I7rms
    I11rms
    I13rms
    I15rms
    I17rms
    I19rms
    I21rms
    I23rms
    bufferall(n)

    figure; hold on;
    plot(t,pall);
    plot(t,pnobuffer);
    %axis([0 0.011 0 3]);
  endif


endfor


figure
hold on
plot(P, buffer3./Ezero);
plot(P, bufferall./Ezero);
plot(P, pf3);
plot(P, pfall);
title('Class b');
legend('Buffer 3','Buffer All','PF 3','PF all');

fid = fopen('classB.csv','w');
fprintf(fid,'%s, %s, %s, %s, %s\n','power','buffer3','bufferall','pf3','pfall');
for i = 1:length(P)
 fprintf(fid,'%f, %f, %f, %f, %f\n',P(i),buffer3(i)./Ezero(i),bufferall(i)/Ezero(i),pf3(i),pfall(i) );
 endfor
fclose(fid);




%Class A


P = [200:10:1600];
Vrms = 220;
I1rms = P./Vrms;
limit3rms = 2.3;
limit5rms = 1.14;
limit7rms = 0.77;
limit9rms = 0.40;
limit11rms= 0.33;
limit13rms= 0.21;
limit15rms= 0.15;
limit17rms= 0.15*15/17;
limit19rms= 0.15*15/19;
limit21rms= 0.15*15/21;
limit23rms= 0.15*15/23;
limit25rms= 0.15*15/25;


%i3 = zeros(length(P),length(t));
%iall = zeros(length(P),length(t));
buffer3 = zeros(1,length(P));
bufferall = zeros(1,length(P));
pf3 = zeros(1,length(P));
pfall = zeros(1,length(P));
Ezero = P/w;

for n = 1:length(P);
  pnobuffer = P(n) * ones(1,length(t));
```

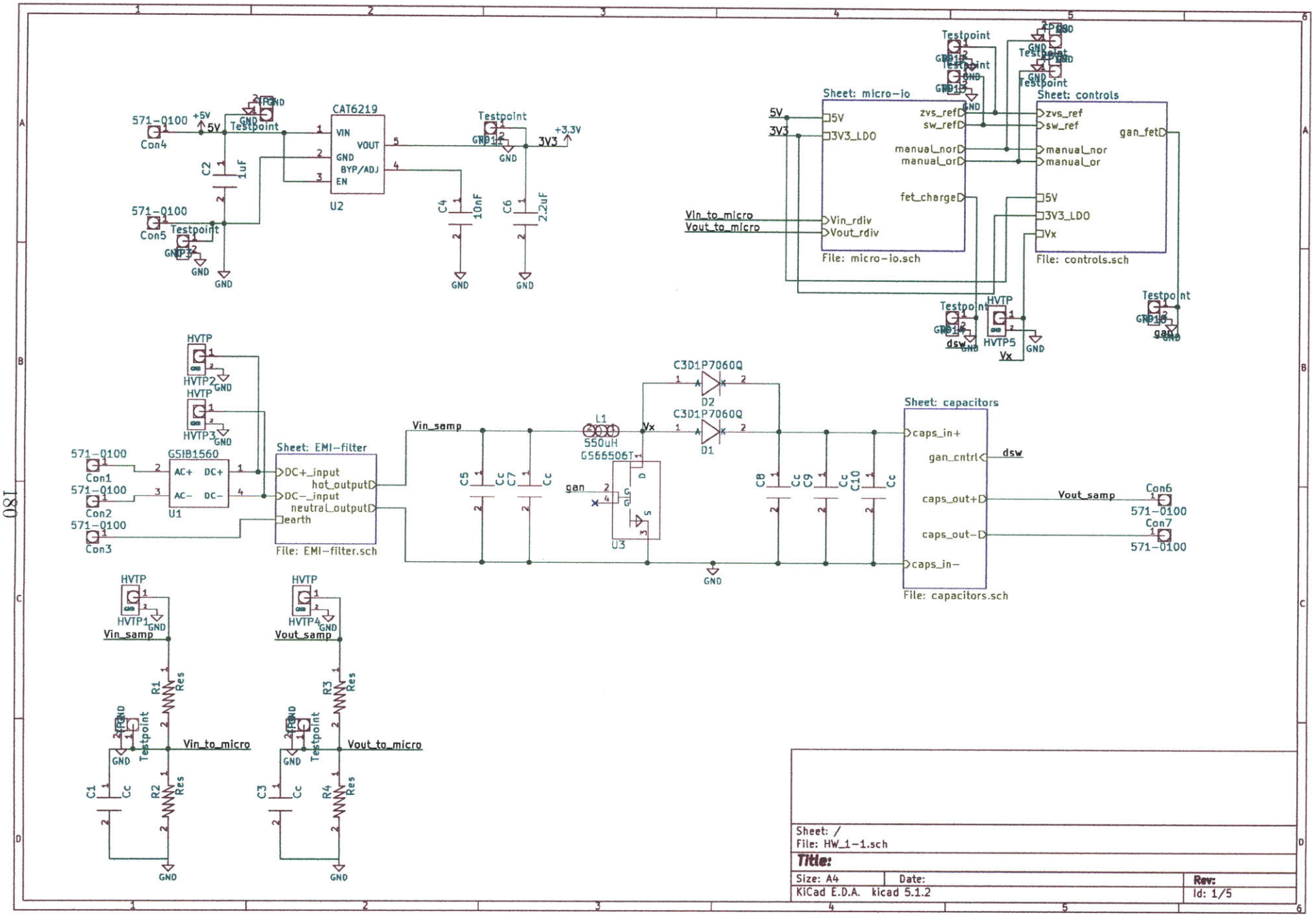
```
I1rms = P(n)/Vrms;
if I1rms < limit3rms
  I3rms = I1rms;
else
  I3rms = limit3rms;
endif

if I1rms < limit5rms
  I5rms = I1rms;
else
  I5rms = limit5rms;
endif

if I1rms < limit7rms
  I7rms = I1rms;
else
  I7rms = limit7rms;
endif

if I1rms < limit9rms
  I9rms = I1rms;
else
  I9rms = limit9rms;
endif

if I1rms < limit11rms
  I11rms = I1rms;
else
  I11rms = limit11rms;
endif

if I1rms < limit13rms
  I13rms = I1rms;
else
  I13rms = limit13rms;
endif

if I1rms < limit15rms
  I15rms = I1rms;
else
  I15rms = limit15rms;
endif

if I1rms < limit17rms
  I17rms = I1rms;
else
  I17rms = limit17rms;
endif

if I1rms < limit19rms
  I19rms = I1rms;
else
  I19rms = limit19rms;
endif

if I1rms < limit21rms
  I21rms = I1rms;
else
  I21rms = limit21rms;
endif

if I1rms < limit23rms
  I23rms = I1rms;
else
  I23rms = limit23rms;
endif
if I1rms < limit25rms
  I25rms = I1rms;
else
  I25rms = limit25rms;
```

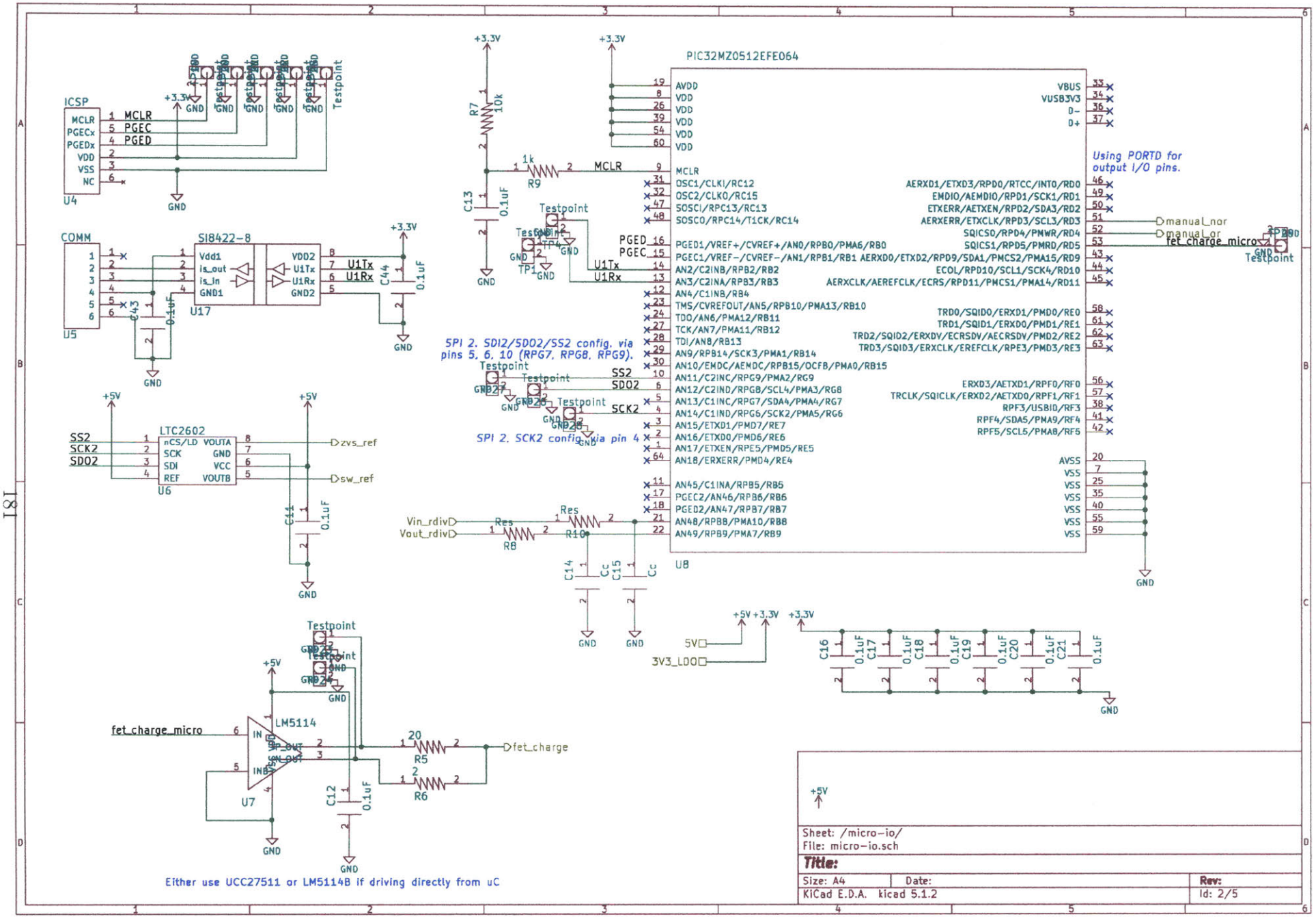
```
  endif


  i3 = I1rms*sqrt(2)*sin(w*t) + I3rms*sqrt(2)*sin(3*w*t);
  p3 = i3.*sqrt(2).*Vrms.*sin(w*t);
  q3 = cumtrapz(pnobuffer-p3)*dt;
  buffer3(n) = max(q3) - min(q3);
  pf3(n) = I1rms/sqrt(I1rms^2 + I3rms^2);

  iall = I1rms*sqrt(2)*sin(w*t) + I3rms*sqrt(2)*sin(3*w*t)+ I5rms*sqrt(2)*sin(5*w*t)+
  I7rms*sqrt(2)*sin(7*w*t)+ I9rms*sqrt(2)*sin(9*w*t)+ I11rms*sqrt(2)*sin(11*w*t)+
  I13rms*sqrt(2)*sin(13*w*t)+ I15rms*sqrt(2)*sin(15*w*t)+ I17rms*sqrt(2)*sin(17*w*t)+
  I19rms*sqrt(2)*sin(19*w*t)+ I21rms*sqrt(2)*sin(21*w*t)+ I23rms*sqrt(2)*sin(23*w*t)+
  I25rms*sqrt(2)*sin(25*w*t);
  pall = iall.*sqrt(2).*Vrms.*sin(w*t);
  qall = cumtrapz(pnobuffer-pall)*dt;
  bufferall(n) = max(qall) - min(qall);
  pfall(n) = I1rms/sqrt(I1rms^2 + I3rms^2 + I5rms^2 + I7rms^2 + I9rms^2 + I11rms^2 + I13rms^2 + I15rms^2
  + I17rms^2 + I19rms^2 + I21rms^2 + I23rms^2 + I25rms^2);


  if (P(n) == 200)
    P(n)
    Vrms
    I1rms
    I3rms
    I5rms
    I7rms
    I11rms
    I13rms
    I15rms
    I17rms
    I19rms
    I21rms
    I23rms
    bufferall(n)

    figure; hold on;
    plot(t,pall);
    plot(t,pnobuffer);
    %axis([0 0.011 0 3]);
  endif


endfor


figure
hold on
plot(P, buffer3./Ezero);
plot(P, bufferall./Ezero);
plot(P, pf3);
plot(P, pfall);
title('Class a');
legend('Buffer 3','Buffer All','PF 3','PF all');


fid = fopen('classA.csv','w');
fprintf(fid,'%s, %s, %s, %s, %s\n','power','buffer3','bufferall','pf3','pfall');
for i = 1:length(P)
 fprintf(fid,'%f, %f, %f, %f, %f\n',P(i),buffer3(i)./Ezero(i),bufferall(i)/Ezero(i),pf3(i),pfall(i) );
 endfor
fclose(fid);



Vout = 406;
Vinrms = 220;
P = 239;
Cin = 2.8*10^-6;
```

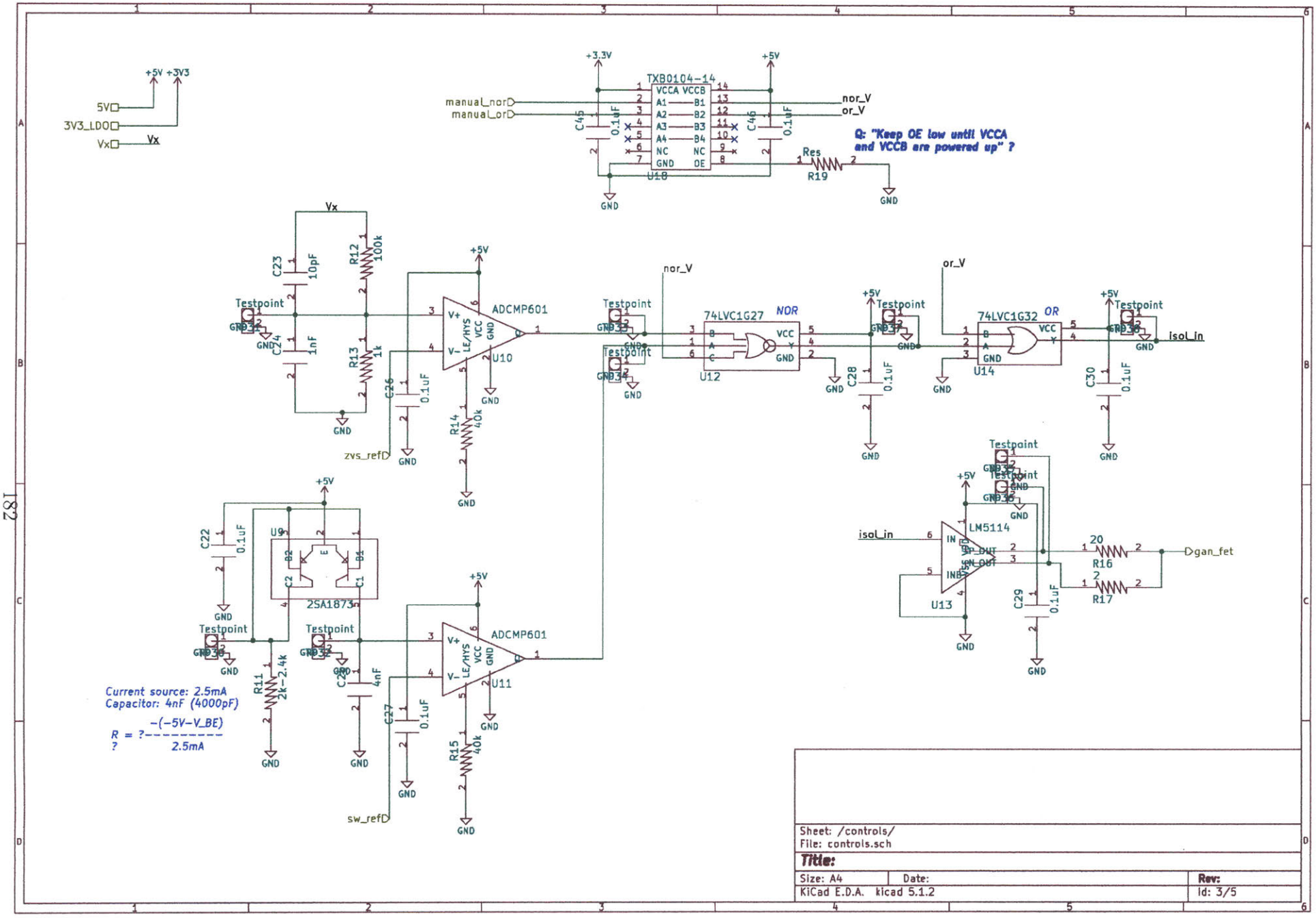
```
I1rms = P/Vinrms;
limit3rms = 0.0034 * P;
limit5rms = 0.0019 * P;
dt = 1*10^-5;  %Time step for time-domain calculations
t = [0:dt:10/1000];  %Time vector based on time step.  Symmetry ensures that only up to T/2 must be considered


Vin = Vinrms *sqrt(2) * sin(w * t);
icin = Cin * diff(Vin) / dt;
icin = [icin icin(end)];

percent = 0.1;
C = 100*10^-6;
I3rms = percent * limit3rms;
I5rms = percent * limit5rms;
ilow = sqrt(2)*(I1rms*sin(w*t) + I3rms*sin(3*w*t) + I5rms*sin(5*w*t)) - icin;
iclow = Vin.*ilow/Vout-P/Vout;
vclow = 1/C * cumtrapz(iclow) * dt + 1.5;

percent = 0.7;
C = 50*10^-6;
I3rms = percent * limit3rms;
I5rms = percent * limit5rms;
ihigh = sqrt(2)*(I1rms*sin(w*t) + I3rms*sin(3*w*t) + I5rms*sin(5*w*t)) - icin;
ichigh = Vin.*ihigh/Vout-P/Vout;
vchigh = 1/C * cumtrapz(ichigh) * dt +1;

figure
hold on
plot(t,ilow+icin);
plot(t,ihigh+icin);

figure
hold on
plot(t,vclow);
plot(t,vchigh);

fid = fopen('ExperimentalPrediction.csv','w');
fprintf(fid,'%s, %s, %s, %s, %s\n','time','iinlow','iinhigh','voutlow','vouthigh');
length(t)
for i = 1:length(t)
 t(i)*1000
 fprintf(fid,'%f, %f, %f, %f, %f\n',(t(i)*1000),(ilow(i)+icin(i)),(ihigh(i)+icin(i)),vclow(i),vchigh(i));
endfor
fclose(fid);
```

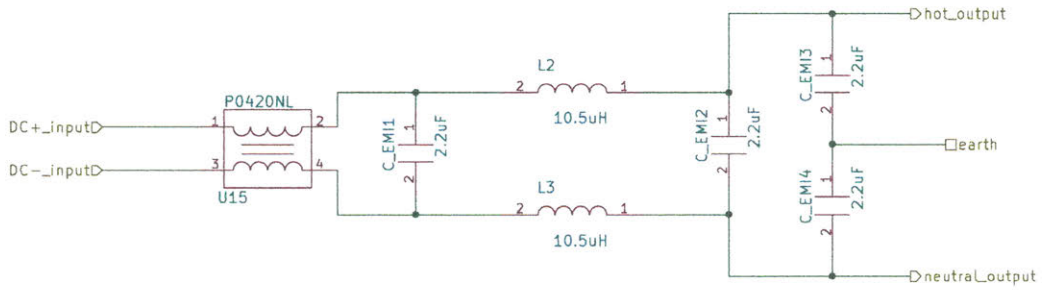
Appendix L

Harmonic Injection Schematic


Sheet: micro-io
File: micro-io.sch
Sheet: controls
File: controls.sch
Sheet: EMI-filter
File: EMI-filter.sch
Sheet: capacitors
File: capacitors.sch
CAT6219
GSIB1560
G566506T
C3D1P7060Q
550uH
Vin_samp
Vout_samp
Vin_to_micro
Vout_to_micro
Sheet: /
File: HW_1-1.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 1/5


ICSP
MCLR
PGECx
PGEDx
VDD
VSS
NC
U4
PGEC
PGED
Testpoint
GND
+3.3V
COMM
U5
SI8422-8
Vdd1
is_out
is_in
GND1
VDD2
U1Tx
U1Rx
GND2
U17
C43
0.1uF
C44
LTC2602
nCS/LD
SCK
SDI
REF
VOUTA
GND
VCC
VOUTB
U6
SS2
SCK2
SDO2
zvs_ref
sw_ref
+5V
C41
LM5114
IN
HB
HO
HS
LO
VDD
LN
U7
fet_charge_micro
fet_charge
20
R5
2
R6
C12
Either use UCC27511 or LM5114B if driving directly from uC
R7
10k
1k
R9
MCLR
C13
TP1
TP4
PIC32MZ0512EFE064
AVDD
VDD
VSS
VBUS
VUSB3V3
D-
D+
OSC1/CLKI/RC12
OSC2/CLKO/RC15
SOSCI/RPC13/RC13
SOSCO/RPC14/T1CK/RC14
PGED1/VREF+/CVREF+/AN0/RPB0/PMA6/RB0
PGEC1/VREF-/CVREF-/AN1/RPB1/RB1
AN2/C2INB/RPB2/RB2
AN3/C2INA/RPB3/RB3
AN4/C1INB/RB4
TMS/CVREFOUT/AN5/RPB10/PMA13/RB10
TDO/AN6/PMA12/RB11
TCK/AN7/PMA11/RB12
TDI/AN8/RB13
AN9/RPB14/SCK3/PMA1/RB14
AN10/EMDC/AEMDC/RPB15/OCFB/PMA0/RB15
AN11/C2INC/RPG9/PMA2/RG9
AN12/C2IND/RPG8/SCL4/PMA3/RG8
AN13/C1INC/RPG7/SDA4/PMA4/RG7
AN14/C1IND/RPG6/SCK2/PMA5/RG6
AN15/ETXD1/PMD7/RE7
AN16/ETXD0/PMD6/RE6
AN17/ETXEN/RPE5/PMD5/RE5
AN18/ERXERR/PMD4/RE4
AN45/C1INA/RPB5/RB5
PGEC2/AN46/RPB6/RB6
PGED2/AN47/RPB7/RB7
AN48/RPB8/PMA10/RB8
AN49/RPB9/PMA7/RB9
AERXD1/ETXD3/RPD0/RTCC/INT0/RD0
EMDIO/AEMDIO/RPD1/SCK1/RD1
ETXERR/AETXEN/RPD2/SDA3/RD2
AERXERR/ETXCLK/RPD3/SCL3/RD3
SQICS0/RPD4/PMWR/RD4
SQICS1/RPD5/PMRD/RD5
AERXD0/ETXD2/RPD9/SDA1/PMCS2/PMA15/RD9
ECOL/RPD10/SCL1/SCK4/RD10
AERXCLK/AEREFCLK/ECRS/RPD11/PMCS1/PMA14/RD11
TRD0/SQID0/ERXD1/PMD0/RE0
TRD1/SQID1/ERXD0/PMD1/RE1
TRD2/SQID2/ERXDV/ECRSDV/AECRSDV/PMD2/RE2
TRD3/SQID3/ERXCLK/EREFCLK/RPE3/PMD3/RE3
ERXD3/AETXD1/RPF0/RF0
TRCLK/SQICLK/ERXD2/AETXD0/RPF1/RF1
RPF3/USBID/RF3
RPF4/SDA5/PMA9/RF4
RPF5/SCL5/PMA8/RF5
U8
Using PORTD for output I/O pins.
manual_nor
manual_or
fet_charge_micro
SPI 2. SDI2/SDO2/SS2 config. via pins 5, 6, 10 (RPG7, RPG8, RPG9).
SPI 2. SCK2 config. via pin 4
Vin_rdiv
Vout_rdiv
Res
R8
R10
C14
C15
Cc
5V
3V3_LDO
C16
C17
C18
C19
C20
C21
0.1uF
Sheet: /micro-io/
File: micro-io.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 2/5

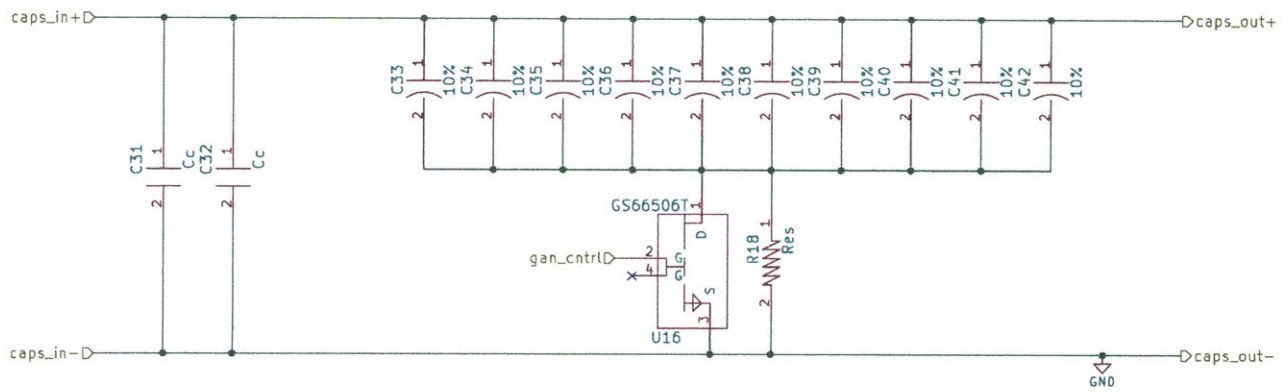

+5V +3V3
5V
3V3_LDO
Vx
Vx
+3.3V
+5V
TXB0104-14
VCCA VCCB
A1 B1
A2 B2
A3 B3
A4 B4
NC NC
GND OE
U18
manual_nor
manual_or
nor_V
or_V
C45 0.1uF
C46 0.1uF
Res
R19
GND
Q: "Keep OE low until VCCA and VCCB are powered up" ?
C23 10pF
R12 100k
C24 1nF
R13 1k
Testpoint
ADCMP601
V+ V- LE/HYS VCC GND
U10
C26 0.1uF
R14 40k
zvs_refD
nor_V
74LVC1G27 NOR
U12
C28 0.1uF
or_V
74LVC1G32 OR
U14
C30 0.1uF
isol_in
C22 0.1uF
U9
2SA1873
R11 2k-2.4k
C25 4nF
ADCMP601
U11
C27 0.1uF
R15 40k
sw_refD
LM5114
IN
SP_OUT
SN_OUT
INB
U13
C29 0.1uF
20
R16
2
R17
gan_fet
Current source: 2.5mA
Capacitor: 4nF (4000pF)
R = ? -(-5V-V_BE) / 2.5mA
Sheet: /controls/
File: controls.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 3/5



Sheet: /EMI-filter/
File: EMI-filter.sch

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 4/5



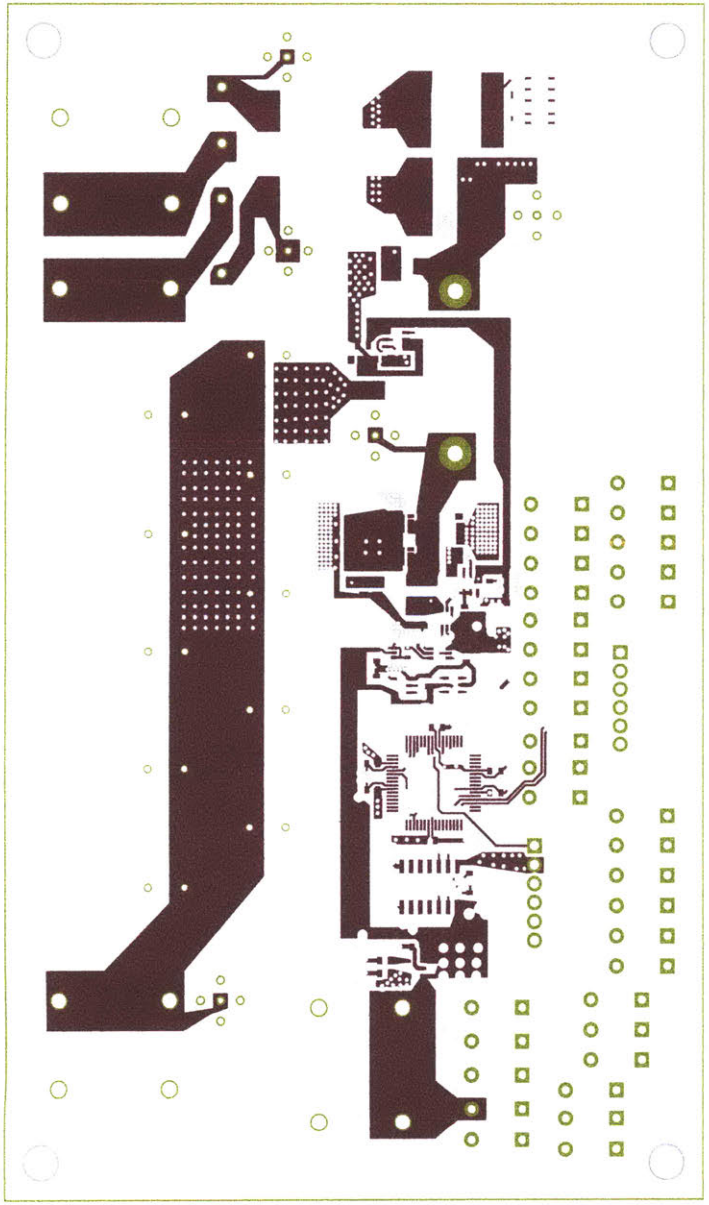
Sheet: /capacitors/
File: capacitors.sch

Title:

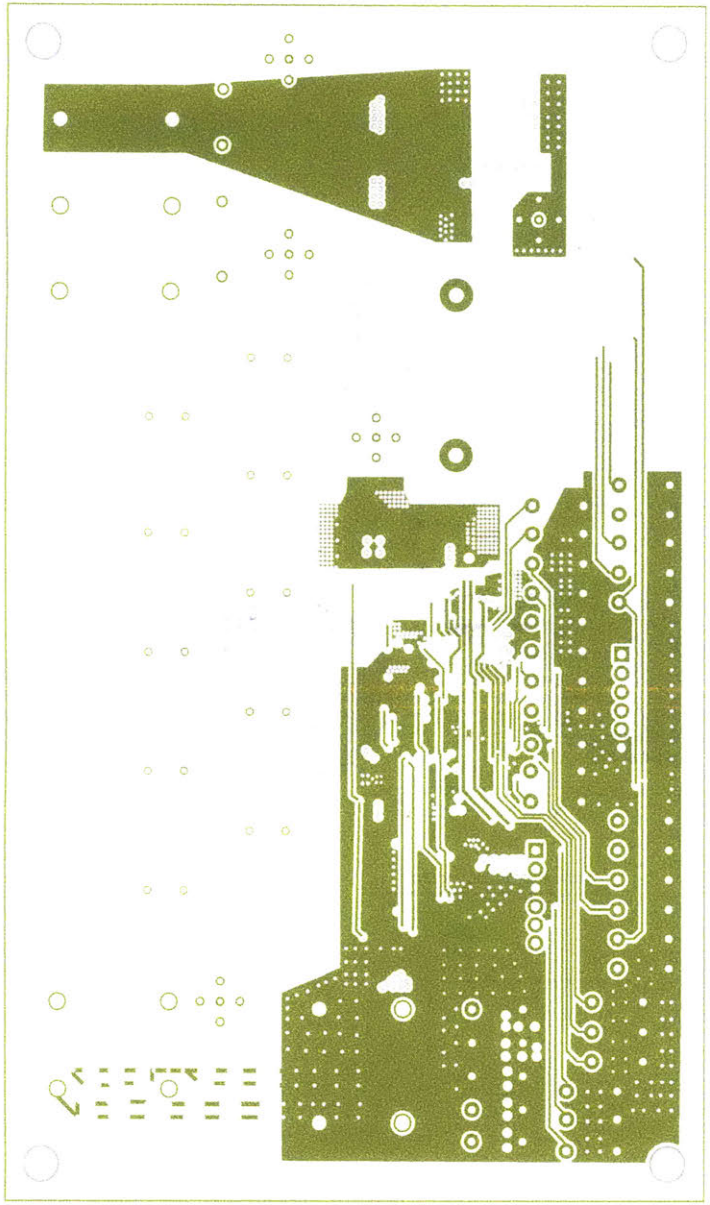
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 5/5

Appendix M

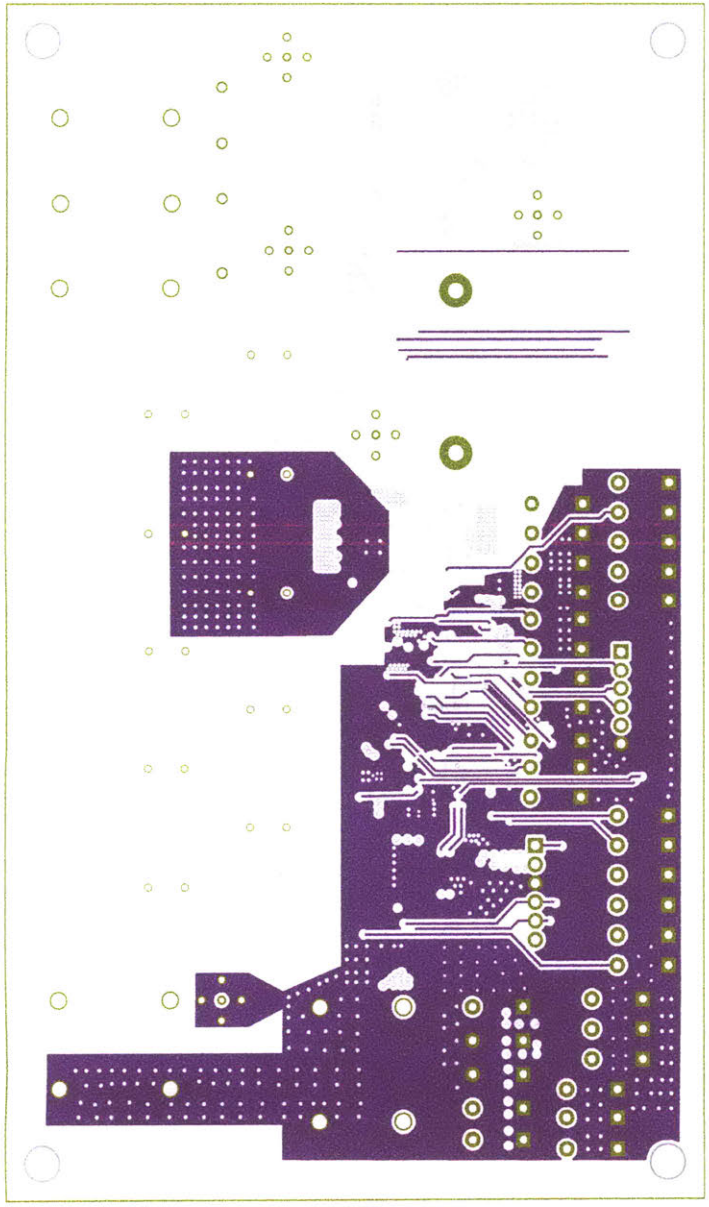
Harmonic Injection Layout



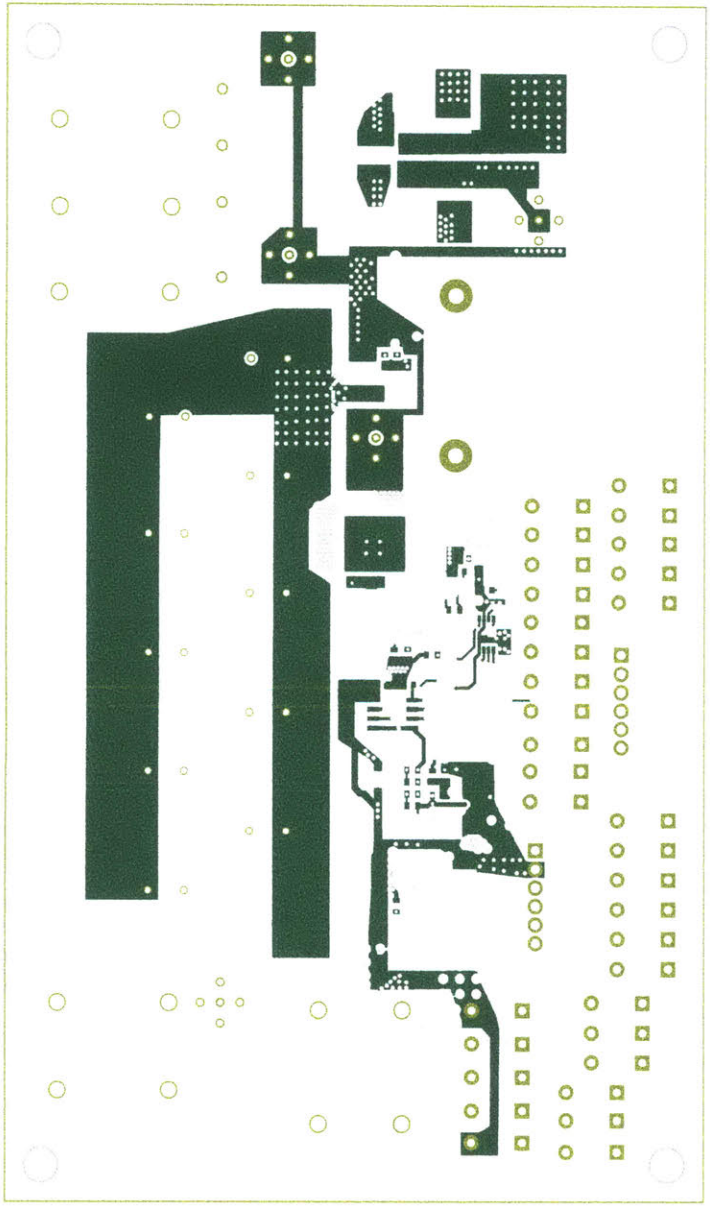
Sheet:		
File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



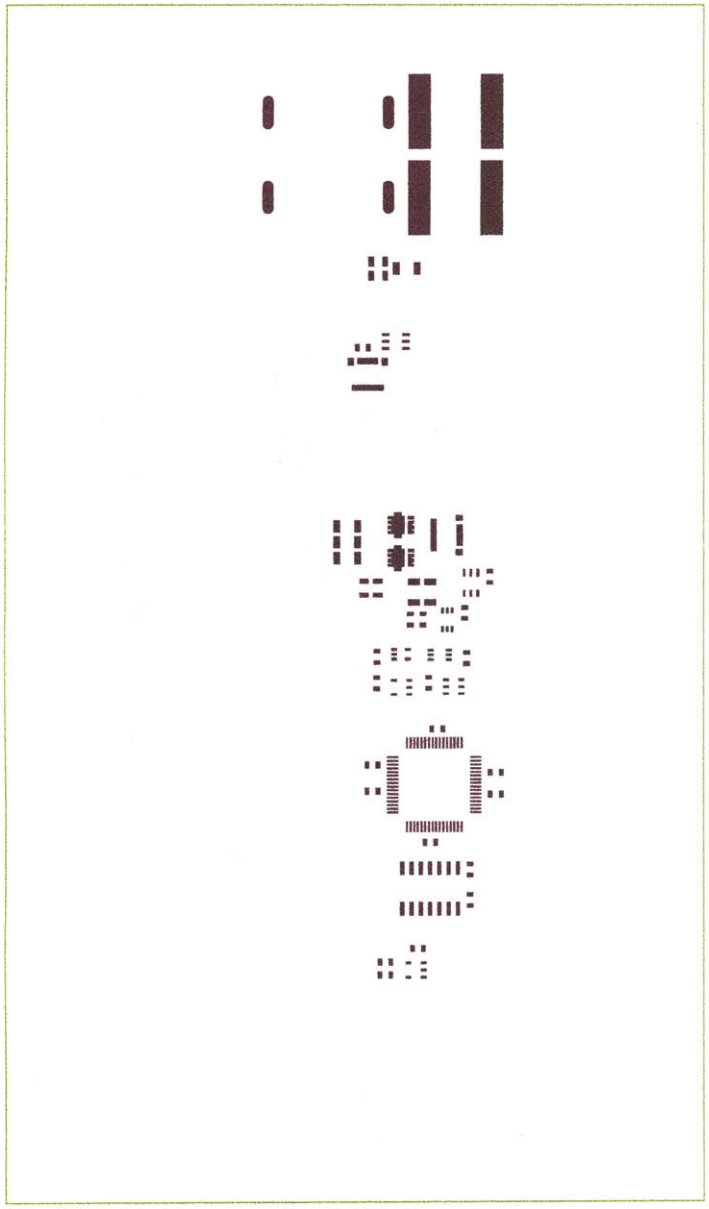
Sheet: File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



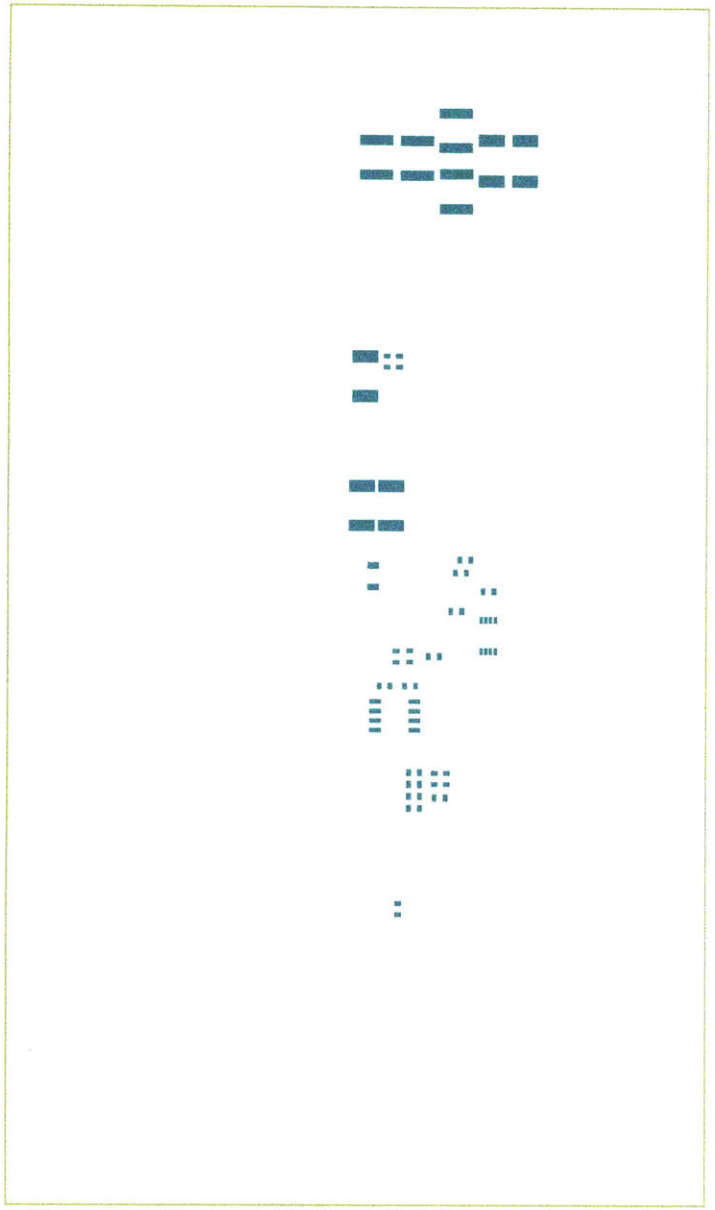
Sheet: File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



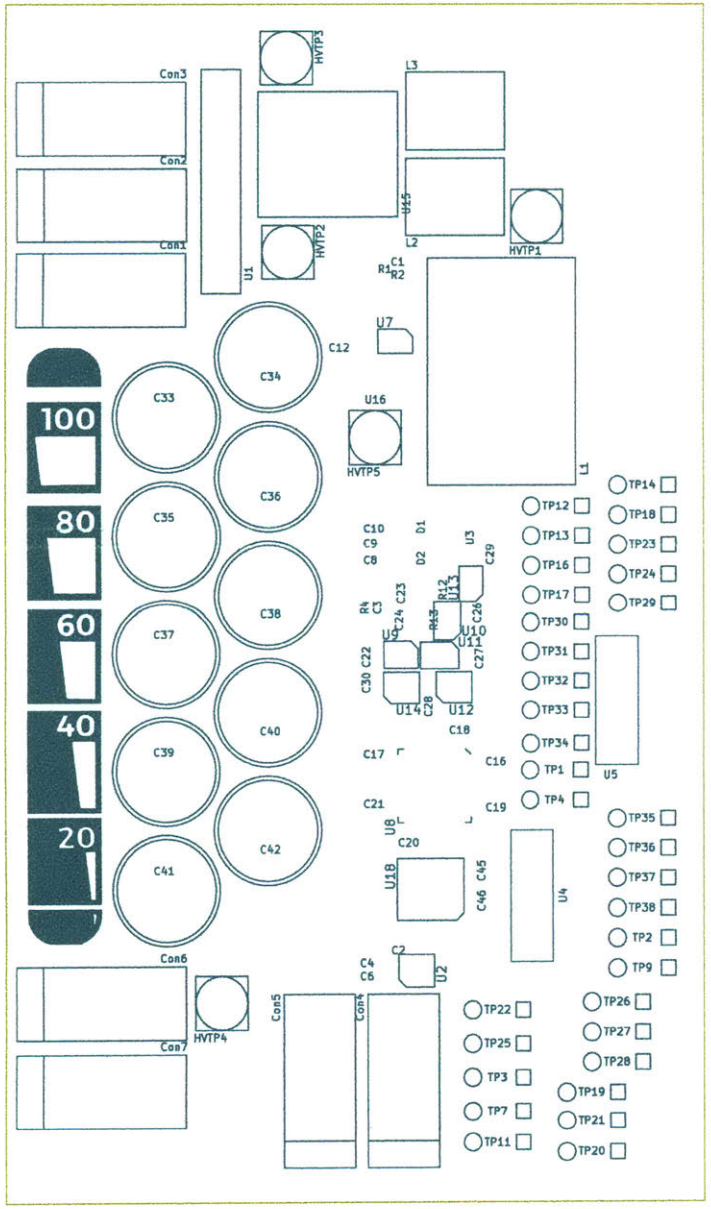
Sheet: File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



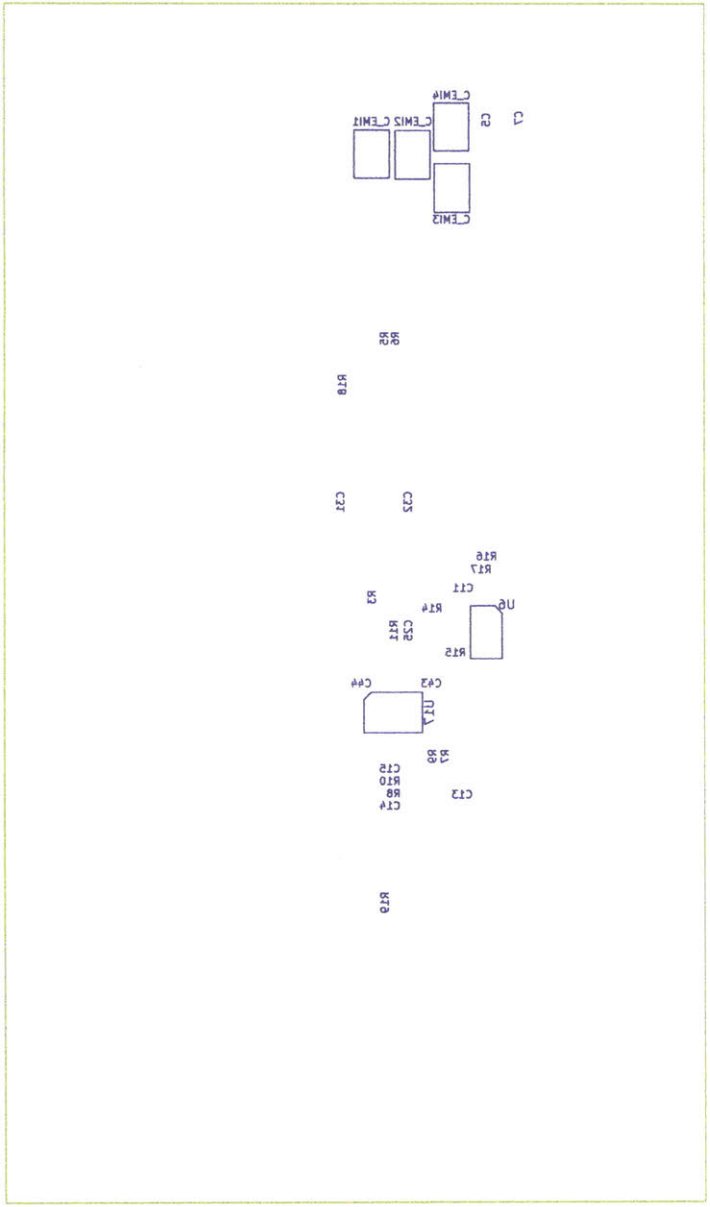
Sheet: File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet: File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet:		
File: HW_1-1.kicad_pcb		
Title:		
Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet:
File: HW_1-1.kicad_pcb

Title:

Size: A4	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1

Appendix N

Harmonic Injection Code

/home/alex/Dropbox (MIT)/Harmonic Input Project/HarmonicInputShare/micro_code/harmonic_input2.c

```c
/*
 * File:   modboost32_refactored.c
 * Author: Alex Hanson
 *
 * Created on May 2, 2017, 11:36 AM
 */



// <editor-fold defaultstate="collapsed" desc="Configuration Bits">

// DEVCFG3
// USERID = No Setting


#pragma config FMIIEN = OFF          // Ethernet RMII/MII Enable (RMII Enabled)
#pragma config FETHIO = ON           // Ethernet I/O Pin Select (Default Ethernet I/O)
#pragma config PGL1WAY = OFF         // Permission Group Lock One Way Configuration (Allow multiple reconfigurations)
#pragma config PMDL1WAY = OFF        // Peripheral Module Disable Configuration (Allow multiple reconfigurations)
#pragma config IOL1WAY = OFF         // Peripheral Pin Select Configuration (Allow multiple reconfigurations)
#pragma config FUSBIDIO = OFF        // USB USBID Selection (Controlled by Port Function)

// DEVCFG2
#pragma config FPLLICLK = PLL_FRC     // System PLL Input Clock Selection (FRC is input to the System PLL)
#pragma config FPLLIDIV = DIV_1       // System PLL Input Divider (1x Divider)
#pragma config FPLLRNG = RANGE_5_10_MHZ // System PLL Input Range (5-10 MHz Input)
#pragma config FPLLMULT = MUL_64      // System PLL Multiplier (PLL Multiply by 4)
#pragma config FPLLODIV = DIV_8       // System PLL Output Clock Divider (2x Divider)
#pragma config UPLLFSEL = FREQ_24MHZ   // USB PLL Input Frequency Selection (USB PLL input is 24 MHz)

// DEVCFG1
#pragma config FNOSC = SPLL          // Oscillator Selection Bits (Fast RC Osc w/Div-by-N (FRCDIV))
#pragma config DMTINTV = WIN_127_128   // DMT Count Window Interval (Window/Interval value is 127/128 counter value)
#pragma config FSOSCEN = OFF         // Secondary Oscillator Enable (Disable SOSC)
#pragma config IESO = OFF            // Internal/External Switch Over (Disabled)
#pragma config POSCMOD = OFF         // Primary Oscillator Configuration (Primary osc disabled)
#pragma config OSCIOFNC = OFF        // CLKO Output Signal Active on the OSCO Pin (Disabled)
#pragma config FCKSM = CSDCMD        // Clock Switching and Monitor Selection (Clock Switch Disabled, FSCM Disabled)
#pragma config WDTPS = PS1048576     // Watchdog Timer Postscaler (1:1048576)
#pragma config WDTSPGM = STOP        // Watchdog Timer Stop During Flash Programming (WDT stops during Flash programming)
#pragma config WINDIS = NORMAL       // Watchdog Timer Window Mode (Watchdog Timer is in non-Window mode)
#pragma config FWDTEN = OFF          // Watchdog Timer Enable (WDT Disabled)
#pragma config FWDTWINSZ = WINSZ_25   // Watchdog Timer Window Size (Window size is 25%)
#pragma config DMTCNT = DMT31        // Deadman Timer Count Selection (2^31 (2147483648))
#pragma config FDMTEN = OFF          // Deadman Timer Enable (Deadman Timer is disabled)

// DEVCFG0
#pragma config DEBUG = OFF           // Background Debugger Enable (Debugger is disabled)
#pragma config JTAGEN = OFF          // JTAG Enable (JTAG Disabled)
#pragma config ICESEL = ICS_PGx1     // ICE/ICD Comm Channel Select (Communicate on PGEC1/PGED1)
#pragma config TRCEN = ON            // Trace Enable (Trace features in the CPU are disabled)
#pragma config BOOTISA = MIPS32      // Boot ISA Selection (Boot code and Exception code is MIPS32)
#pragma config FECCCON = OFF_UNLOCKED  // Dynamic Flash ECC Configuration (ECC and Dynamic ECC are disabled (ECCCON bits are writable))
#pragma config FSLEEP = OFF          // Flash Sleep Mode (Flash is powered down when the device is in Sleep mode)
#pragma config DBGPER = PG_ALL       // Debug Mode CPU Access Permission (Allow CPU access to all permission regions)
#pragma config SMCLR = MCLR_NORM     // Soft Master Clear Enable bit (MCLR pin generates a normal system Reset)
#pragma config SOSCGAIN = GAIN_2X    // Secondary Oscillator Gain Control bits (2x gain setting)
#pragma config SOSCBOOST = ON        // Secondary Oscillator Boost Kick Start Enable bit (Boost the kick start of the oscillator)
#pragma config POSCGAIN = GAIN_2X    // Primary Oscillator Gain Control bits (2x gain setting)
#pragma config POSCBOOST = ON        // Primary Oscillator Boost Kick Start Enable bit (Boost the kick start of the oscillator)
#pragma config EJTAGBEN = NORMAL     // EJTAG Boot (Normal EJTAG functionality)

// DEVCP0
#pragma config CP = OFF              // Code Protect (Protection Disabled)


// </editor-fold>

//Windows
#include "./proc/p32mz0512efe064.h"
#include "xc.h"
#include <sys/attribs.h>
#include <stdlib.h>
#include <stdio.h>
#include <math.h>

/////Linux
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/proc/p32mz0512efe064.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/xc.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/stdio.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/math.h"
//#include <sys/attribs.h>


// <editor-fold defaultstate="collapsed" desc="Pin #defines">

//#define DIO1_set LATESET = 1<<5
//#define DIO2_set LATESET = 1<<6
//#define DIO3_set LATESET = 1<<7
//#define DIO4_set LATGSET = 1<<6
//#define DIO5_set LATGSET = 1<<7

//#define DIO1_clr LATECLR = 1<<5
//#define DIO2_clr LATECLR = 1<<6
//#define DIO3_clr LATECLR = 1<<7
//#define DIO4_clr LATGCLR = 1<<6
//#define DIO5_clr LATGCLR = 1<<7


//#define ZVS_select_set LATPSET= 1<<3
//#define CSa_set       LATDSET= 1<<0
#define CSb_set       LATGSET= 1<<9  //#define CSb_set       LATDSET= 1<<3
//#define ENABLEa_set    LATESET= 1<<1
#define ENABLEb_set    LATDSET= 1<<3  //#define ENABLEb_set    LATESET= 1<<4
//#define MANUALa_set    LATESET= 1<<2
#define MANUALb_set    LATDSET= 1<<4  //#define MANUALb_set    LATESET= 1<<3

//#define ZVS_select_clr LATPCLR= 1<<3
//#define CSa_clr       LATDCLR= 1<<0
#define CSb_clr       LATGCLR= 1<<9
//#define ENABLEa_clr    LATECLR= 1<<1
#define ENABLEb_clr    LATDCLR= 1<<3
//#define MANUALa_clr    LATECLR= 1<<2
#define MANUALb_clr    LATDCLR= 1<<4

#define FETcharge_set   LATDSET= 1<<5
#define FETcharge_clr   LATDCLR= 1<<5

#define Trig_set       LATBSET= 1<<1
#define Trig_clr       LATBCLR= 1<<1

//Define ADCdata registers
//Will take care of Uart and SPI in functions rather than #defines
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Function Prototypes">
//double Compute11(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double i2, double limitpercent, double i3, double i5);
//void SetupSPIa(void);
void CheckReset(void);
void SetupSPIb(void);
void SetupADC(void);
void SetupUART2(void);
void SetupUART3(void);
void SetupTimer(void);
void StartupBoost(void);
//void StartupCycle(void);
void TriggerPulse(void);
void WriteFlags(unsigned int running, unsigned int mode);
void WriteKickCount(unsigned int kickcount);
void WriteMenu(void);
void WriteScreen(char s[50]);
//void WriteSPIa(unsigned int* dacinput, char select);
void WriteSPIb(unsigned int* dacinput, char select);
void WriteVin(double vin);
void WriteVout(double vout);

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Static Volatile Initializations/Declarations">

typedef struct {
    unsigned Timer :1;
    unsigned ADC1 :1;
    unsigned U3RX :1;
    unsigned Rising :1;
    unsigned WriteTimes :1;
    unsigned Running :1;
    unsigned Mode :1;
    unsigned Control :1;
} flagStruct;
//Mode and Running form a state machine with states Running,Mode
//State 0,0 = Everything is off, converter hasn't started
//State 0,1 = SA1 is on for initial charging of output capacitor
//State 1,1 = Operation in mod boost mode
```

```
//State 1.0 = Operation in regular boost mode
//See code for state transitions
static volatile FlagStruct Flags;

static volatile double adcount;
static volatile double vinaccum;
static volatile double voutaccum;
static volatile int protect = 0;
static volatile char selection;
// </editor-fold>


int main(int argc, char** argv) {

// <editor-fold defaultstate="collapsed" desc="Global Interrupt Enable">
  //Enable uC to receive interrupts
  //  macros come from this page:  http://microchipdeveloper.com/32bit:mz-arch-exceptions-usage
  INTCONSET = _INTCON_MVEC_MASK; //Set interrupt controller to Multi Vector Mode through macro
  __builtin_enable_interrupts(); // Globally enable interrupts through CP0 StatusIE bit
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Oscillator Setup">


  REFO1CONbits.OE = 0;
  OSCCONbits.PBCDIV = 000;  //Same as default, divide by 1

  //REFCLK1 can be used for SPI (or PBCLK2) or as an output
  //REFCLK2 can be used for output or SQI (unused)
  //REFCLK3 can be used for ADC (or FRC or SYSCLK) or as an output

  REFO1CONbits.RODIV = 0;
  REFO1CONbits.RSLP = 1; //Run during sleep
  if(REFO1CONbits.ACTIVE == REFO1CONbits.ON) { //Shouldn't write if active != ON
    REFO1CONbits.ROSEL = 0b0000;  //Use SYSCLK output as source for reference
    REFO1CONbits.RODIV = 0;  //No divider for reference clock
    REFO1TRIM = 0;
    REFO1CONbits.ON = 0;   //Turn on to activate
  }
  REFO1CONbits.OE = 0; //Turn on to drive the REFCLKO1 pin (defined by PPS)
  //Note that SPLLCON should be completely defined by config registers on reset




  // <editor-fold defaultstate="collapsed" desc="Unlock Sequence for Modifying Oscillator">
  volatile unsigned int int_status;
  volatile unsigned int dma_suspend;

  // Disable global interrupt
  int_status = __builtin_get_isr_state();
  __builtin_disable_interrupts();

  // Suspend DMA
  dma_suspend = DMACONbits.SUSPEND;
  if (dma_suspend == 0)
  {
    DMACONSET = _DMACON_SUSPEND_MASK;
    while (DMACONbits.DMABUSY == 1);
  }

  /* Unlock */
  SYSKEY = 0x00000000;
  SYSKEY = 0xAA996655;
  SYSKEY = 0x556699AA;
  // </editor-fold>


  PB3DIVbits.PBDIV = 0;  //PBCLK3 = SYSCLK  (no scaling)  runs TMR1 which triggers ADC
  PB3DIVbits.ON = 1;  //Activate PBCLK3

  PB2DIVbits.PBDIV = 0;  //Divide by 2; UART
  PB2DIVbits.ON = 1;

  PB1DIVbits.PBDIV = 0;
  //PB1DIVbits.ON = 1;


  // <editor-fold defaultstate="collapsed" desc="Re-Lock Sequence After Modifying Oscillator">
  SYSKEY = 0x33333333;

  if (dma_suspend == 0)
  {
    DMACONCLR = _DMACON_SUSPEND_MASK;
  }

  __builtin_set_isr_state(int_status);
  // </editor-fold>


// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Analog Pin Assignment">
  //Note that pins with analog option default to analog input unless changed manually
  //Analog pins only appear on B, E, G ports
  ANSELB = 0;
  ANSELE = 0;
  ANSELG = 0;
  ANSELBbits.ANSB8 = 1;   //Vin_low
  ANSELBbits.ANSB9 = 1;   //Vout_low
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Pin State Initialization">
  TRISEbits.TRISE5 = 0; LATEbits.LATE5 = 0;   //Pin 1 output low (unused)
  TRISEbits.TRISE6 = 0; LATEbits.LATE6 = 0;   //Pin 2 output low (unused)
  TRISEbits.TRISE7 = 0; LATEbits.LATE7 = 0;   //Pin 3 output low (unused)
  TRISGbits.TRISG6 = 0; LATGbits.LATG6 = 0;   //Pin 4 output low (SCK2)
  TRISGbits.TRISG7 = 0; LATGbits.LATG7 = 0;   //Pin 5 output low (unused)
  TRISGbits.TRISG8 = 0; LATGbits.LATG8 = 0;   //Pin 6 output low (unused)
                        //Pin 7 VSS
                        //Pin 8 VDD
                        //Pin 9 MCLR
  TRISGbits.TRISG9 = 0; LATGbits.LATG9 = 0;   //Pin 10 output low (CSb)
  TRISBbits.TRISB5 = 0; LATBbits.LATB5 = 0;   //Pin 11 output low (unused)
  TRISBbits.TRISB4 = 0; LATBbits.LATB4 = 0;   //Pin 12 output low (unused)
  TRISBbits.TRISB3 = 1;                       //Pin 13 input (U1RX)
  TRISBbits.TRISB2 = 0; LATBbits.LATB2 = 0;   //Pin 14 output low (unused)
  TRISBbits.TRISB1 = 0; LATBbits.LATB1 = 0;   //Pin 15 PGEC1, also used as debugger pin
                        //Pin 16 PGED1
  TRISBbits.TRISB6 = 1; LATBbits.LATB6 = 0;   //Pin 17 output low (unused)
  TRISBbits.TRISB7 = 0; LATBbits.LATB7 = 0;   //Pin 18 output low (unused)
                        //Pin 19 AVDD
                        //Pin 20 AVSS
  TRISBbits.TRISB8 = 1;                       //Pin 21 input (Vin_low)
  TRISBbits.TRISB9 = 1;                       //Pin 22 input (Vout_low)
  TRISBbits.TRISB10 = 0; LATBbits.LATB10 = 0; //Pin 23 output low (unused)
  TRISBbits.TRISB11 = 0; LATBbits.LATB11 = 0; //Pin 24 output low (unused)
                        //Pin 25 VSS
                        //Pin 26 VDD
  TRISBbits.TRISB12 = 0; LATBbits.LATB12 = 0; //Pin 27 output low (unused)
  TRISBbits.TRISB13 = 0; LATBbits.LATB13 = 0; //Pin 28 output low (unused)
  TRISBbits.TRISB14 = 0; LATBbits.LATB14 = 0; //Pin 29 output low (unused)
  TRISBbits.TRISB15 = 0; LATBbits.LATB15 = 0; //Pin 30 output low (unused)
  TRISCbits.TRISC12 = 0; LATCbits.LATC12 = 0; //Pin 31 output low (unused)
  TRISCbits.TRISC15 = 0; LATCbits.LATC15 = 0; //Pin 32 output low (unused)
                        //Pin 33 VBUS (unused)
                        //Pin 34 VUSB3V3 (grounded)
                        //Pin 35 VSS
                        //Pin 36 D- (unused)
                        //Pin 37 D+ (unused)
  TRISFbits.TRISF3 = 0; LATFbits.LATF3 = 0;   //Pin 38 output low (unused)
                        //Pin 39 VDD
                        //Pin 40 VSS
  TRISFbits.TRISF4 = 0; LATFbits.LATF4 = 0;   //Pin 41 output low (unused)
  TRISFbits.TRISF5 = 0; LATFbits.LATF5 = 0;   //Pin 42 output low (unused)
  TRISDbits.TRISD9 = 0; LATDbits.LATD9 = 0;   //Pin 43 output low (unused)
  TRISDbits.TRISD10 = 0; LATDbits.LATD10 = 0; //Pin 44 output low (unused)
  TRISDbits.TRISD11 = 0; LATDbits.LATD11 = 0; //Pin 45 output low (unused)
  TRISDbits.TRISD0 = 0; LATDbits.LATD0 = 0;   //Pin 46 output low (unused)
  TRISCbits.TRISC13 = 0; LATCbits.LATC13 = 0; //Pin 47 output low (unused)
  TRISCbits.TRISC14 = 0; LATCbits.LATC14 = 0; //Pin 48 output low (unused)
  TRISDbits.TRISD1 = 0; LATDbits.LATD1 = 0;   //Pin 49 output low (SCKb)
  TRISDbits.TRISD2 = 0; LATDbits.LATD2 = 0;   //Pin 50 output low (SDOb)
  TRISDbits.TRISD3 = 0; LATDbits.LATD3 = 0;   //Pin 51 output low (manual_nor/enable)
  TRISDbits.TRISD4 = 0; LATDbits.LATD4 = 0;   //Pin 52 output low (manual_or)
  TRISDbits.TRISD5 = 0; LATDbits.LATD5 = 0;   //Pin 53 output low (fet_charge_micro)
                        //Pin 54 VDD
                        //Pin 55 VSS
  TRISFbits.TRISF0 = 0; LATFbits.LATF0 = 0;   //Pin 56 output low (unused)
  TRISFbits.TRISF1 = 0; LATFbits.LATF1 = 0;   //Pin 57 output low (unused)
  TRISEbits.TRISE0 = 0; LATEbits.LATE0 = 0;   //Pin 58 output low (unused)
                        //Pin 59 VSS
```

```
                                    //Pin 60 VDD
TRISEbits.TRISE1 = 0; LATEbits.LATE1 = 0;   //Pin 61 output low (unused)
TRISEbits.TRISE2 = 0; LATEbits.LATE2 = 0;   //Pin 62 output low (unused)
TRISEbits.TRISE3 = 0; LATEbits.LATE3 = 0;   //Pin 63 output low (unused)
TRISEbits.TRISE4 = 0; LATEbits.LATE4 = 0;   //Pin 64 output low (unused)
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Peripheral Pin Select Assignments">
U3RXR = 0b1000;     //Set U3RX to be pin RB3

RPB2R = 0b0010;     //Set RPB2 to be U2TX

RPG6R = 0b0110;    //Set RPG8 to be SDO2
//RPD2R = 0b0101;     //Set RPD2 to be SDO1
//RPE5R = 0b1111;     //Set RPE5 (pin 1) to be REFCLKO1
                //REFCLKO1 can also go to RPG7 (pin 5)
                //REFCLKO3 can go to RPG6 (pin 4)

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Variable Definition and Initialization">
Flags.Timer = 0;
Flags.ADC1 = 0;
Flags.U3RX = 0;
Flags.WriteTimes = 0;
Flags.Running = 0;
Flags.Mode = 0; //
Flags.Rising = 1;
Flags.Control = 0;

ENABLEb_set;
MANUALb_clr;

//Note that I can start up with R = 1100 kicking in at 350 V
//Cb starts at 8000*65536; Ca starts at 4850*65536
//Expect slightly higher than 400 at first; very close after engaging control
//Or same Cb, Ca and R = 850 for a 240 V in put



//(variable)_16 corresponds to 16 bit codes to send to 5V DAC
//They are converter from "normal" variables by (variable in volts) * 2^16/5
unsigned int vcntrlb_16 = 14000;
//unsigned int vzvsb_16 = 5500;  This worked forever before removing linear caps
unsigned int vzvsb_16 = 7500;
unsigned int vcntrla_16 = 23000;
unsigned int vzvsa_16 = 40000;
unsigned int timeb_pause_16;
unsigned int vout;
unsigned int i = 0;
unsigned int ControlCount = 0;
unsigned int KickCount = 0;
unsigned int VrmsCount = 0;
unsigned int AvgCount = 0;

//double zvsamult = 0.8;  //This worked forever before removing linear caps
double zvsamult = 0.75;
double temp;
double timeb;
double timea;
//double Cb = 8000*65536;//7000*65536;  = L*i1, where L approx "4000"

//double L = 16600; //inductance, units ns*V/A (= nH) was 16600
double L = 116600;  //Rachel says inductor is actually like 13.9 uH
double i1 = 1.0;  //Current, units A  (2.8 worked for a long time)
double i2 = 1.0;  //Current, units A
//double Cb = 40000; //Uni      ts ns*V; 32000 corresponds to 16 uH * 2.5 A
//double Ca = 5150*65536;//5000*65536;  = L*i2
//double Ca = 28800; //Units ns*V; 28800 corresponds to 16 uH * 1.8 A
double i1Startup = i1;
double integral = i1;
double derivative;
double error;
double preverror;
//double correctiona = 0;  //ns
//double correctionb = 0;  //ns
//double correctionb_prefactor = 0.133; //  ns/V
//double correctiona_prefactor = 0.216; //  ns/V
double Vout;
double Vin;
//double rampslope = 0.00375;  //3 Volt per 800 ns, verified for 5.6 kohm, 220 pF
//double rampslope = 0.00191; //1.53 Volt per 800 ns, assumed for 11 kohm, 220 pF
double rampslope = 0.00025; //0.42 volt per 800 ns, assumed for 2.2kohm, 4nF
//currently 2 V over 7 us as of 4/29/18 --

double R = 80;  //R=250 gives 40 W at 120 Vrms   //Worked very well at R =125, gives about 220 W at 200 Vrms
double I3 = 0.0034; //A/W  (spec given in mA/W)
double I5 = 0.0019;
double I7 = 0.0010;
double I9 = 0.0005;
double I11 = 0.00035;
double limitpercent = 0;

double tnormal;
double tforres;
double tforcap;
double ttotal;
//double C_in = 2000 ;  //1.7 uF = 250 ns *A/V  //I think Cin is measured in nF
double C_in = 7000;
double w_line = 0.0000003142; // per ns  (50 hz)
double Vrms = 110; //volts
double VinMax = 0;
//double Cres = 0.12;  //100 pF = 0.1 ns * A/V
//double Cres = 0.15; //0.12 worked well for a long time; needed higher for zero crossing, stitching
double Cres = 0.2; //Used 0.15 before removing linear caps
double Peff = 200;
double Iin = 0;
double iconv = 0;
double x = 0;
double D = 0;
double test = 1;

double Vin_prev = 0;
double cap_sign = 1;
double Vin_prevSign = 0;
int signstate = 1;
unsigned int peakwaitcount = 0;

unsigned int SignCount = 0;

char buf[16];

// </editor-fold>


SetupSPIb();
SetupADC();
SetupTimer();
SetupUART2();
SetupUART3();

PRECONbits.PREFEN = 0b11;  //Allow Predictive Prefetch of CPU instructions and data
PRECONbits.PFMWS = 0b000;  //Zero wait states for PFM access time (see Table 37-13)


WriteSPIb(&vzvsb_16, '2');
WriteSPIb(&vcntrlb_16, '3');

FETcharge_set;  //start with the charging FET just on all the time

for(i=0;i<10000;i++){Nop();} //Letting modules get started up, clean WriteMenu
WriteScreen("\n\r");
WriteMenu();




CheckReset();




while(1) {


  if(Flags.U3RX == 1) {
    // <editor-fold defaultstate="collapsed" desc="User Interface Update">
    WriteScreen("\n\r");
    //selection = U4RXREG; U4TXREG = selection;  //Echo selection to the screen
```

```
503
504         switch (selection) {
505           case 'x':
506             limitpercent = limitpercent + 0.1;
507             //sprintf(buf, "%.1f", limitpercent);
508             //WriteScreen(buf);
509             break;
510           case 'z':
511             limitpercent = limitpercent - 0.1; if(limitpercent<0){limitpercent=0;}
512             //%sprintf(buf, "%.1f", limitpercent);
513             //WriteScreen(buf);
514             break;
515           case 'c':
516             zvsamult = zvsamult + 0.05; break;
517           case 'v':
518             zvsamult = zvsamult - 0.05; break;
519           case 'g':
520             Flags.Control = 1; break;
521           case 'f':
522             WriteVout(Vout); break;
523           case 'r':
524             WriteVin(Vrms); break;
525           case 'q':
526             R = R + 20; break;
527           case 'w':
528             R = R - 20; break;
529           case 'b':
530             WriteFlags(Flags.Running, Flags.Mode);break;
531           case 'n':
532             WriteKickCount(KickCount); break; //WriteSPIa(&vzvsb_16, 'c'); break;
533           case 'm':
534             WriteMenu(); break;
535           default:
536             WriteScreen("Invalid Selection");
537         } //End of switch statement
538
539         Flags.U3RX = 0;
540         // <editor-fold>
541       } // End of If(Flags.U3RX)
542
543
544
545       if(Flags.Timer == 1) {
546         // <editor-fold defaultstate="collapsed" desc="State Update">
547
548
549         Flags.Timer = 0;
550         KickCount = KickCount + 1;
551
552         ////////////////  Voltage Update  ////////////////////////
553        //Taking ADC data and convert to 16-bit (double) where number is in volts
554         Vout = (double) (ADCDATA0>>16); //.0067575 = 3.3 * 134.3 / 2^16
555         Vout = Vout*(3.3 * 126.8 / 65536);  //Was 134.3; 115.5 tuned on 270 V out
556         Vin_prev = Vin;
557         Vin = (double) (ADCDATA3>>16);
558         Vin = Vin*(3.3 * 133 / 65536);  //Was 134.3; 117 tuned on 25 Vin
559
560         ////////////////  Vrms Detection ////////////////
561         if(Vin > Vrms*1.414) {Vrms = Vin * 0.7071;}
562         if(Vin > VinMax) {VinMax = Vin;}
563         VrmsCount = VrmsCount + 1;
564         if(VrmsCount > 1000){    //Timer goes off every 32 us, or 259 times per half-period
565                     //So waiting for every 4 half-cycles is about 1000 counts
566
567           //if(VinMax > 350) { Vrms = 240; }  //Don't want it to accidentally think the voltage is crazy high
568           //else { Vrms = VinMax * 0.7071 + 5; }
569           if(VinMax < Vrms*1.414){Vrms = VinMax*0.7071;}
570
571           VinMax = 0;
572           VrmsCount = 0;
573         }
574
575
576         ////////////////  dV/dt sign Detection ////////////////
577         if(SignCount == 2) {  //32 us * 8 = 256 us  WAS 8
578
579 //          //If the last time you looked, voltage was rising
580 //          //Add hysteresis to make it harder to change states
581 //          if(signstate == 1) {
582 //            if(Vin-Vin_prevSign > 0)
583 //              {Flags.Rising = 1; cap_sign = -1;}
584 //            else
585 //              {Flags.Rising = 0; cap_sign = 1;}
586 //          }
587 //          else {   //If the last time you looked, voltage was falling
588 //            if(Vin-Vin_prevSign > 6)
589 //              {Flags.Rising = 1; cap_sign = -1;}
590 //            else
591 //              {Flags.Rising = 0; cap_sign = 1;}
592 //          }
593           switch(signstate) {
594
595             case 1: //voltage rising
596               cap_sign = -1;
597               if(Vin > Vrms*1.41*0.96) {  //WAS 0.85
598                 signstate = 2; peakwaitcount = 0;
599                 if(Flags.Control ==1) {
600                   R = R + (Vout-400)*0.04;  //0.1 worked fine at low power
601                 }
602                 //StartupBoost();
603               }//WriteScreen("2");}  //Moving to top state, #2
604               break;
605             case 2: //voltage near peak
606               cap_sign = 0;
607               if(peakwaitcount > 8) {signstate = 3; peakwaitcount = 0;  }// USED TO WAIT FOR 2 peakwaitcounts
608               else {peakwaitcount = peakwaitcount + 1;}
609               break;
610             case 3: //Voltage falling
611               cap_sign = 1;
612               if(Vin < Vrms*1.41*0.12) {signstate = 4; }//WriteScreen("4");}
613               break;
614             case 4:
615               cap_sign = -1;
616               if(Vin > Vin_prevSign-3){
617                 signstate = 1;
618                 StartupBoost();
619               }//WriteScreen("1");}
620               break;
621             default:
622               cap_sign = -1;
623               signstate = 1;
624           } //close switch
625
626           SignCount = 0;
627           Vin_prevSign = Vin;
628
629         }
630         else {
631           SignCount = SignCount + 1;
632         }
633
634
635         ////////////////// Sanitize Vin to avoid sqrt problems ////////////
636         //If Vin is even close to Vrms*sqrt(2), make it just smaller than Vrms*sqrt(2)
637         //Because Vrms updates every cycle, this should only engage at the very edges
638         if(Vin > Vrms*1.414) {Vin = 1.414*Vrms-1;}  //
639
640
641
642         ////////////////  Set ZVS threshold ////////////////
643         if(   Vin > (138)   ){
644           vzvsb_16 = (unsigned int) (  (0.014*Vin-1.85) *65536/5);
645           //WriteScreen("M");
646         }
647         else{
648           vzvsb_16 = (unsigned int) (0.080*65536/5);  //worked great on 0.075 up to 175 V
649         }
650         //vzvsb_16 = (unsigned int) (0.1*65536/5);
651         WriteSPIb(&vzvsb_16, 'z');
652         //for(i=10000;i>0;i--){Nop();}
653
654
655         if(Flags.Running==0 && Flags.Mode==0){
656           // <editor-fold defaultstate="collapsed" desc="Charging State">
657
658           if(Vout > 350) { //350 //27525 -- above this Vout regular boost safe  //Standard was 375
659             WriteScreen("H");
660             Flags.Running = 1;
661             Flags.Mode = 0;
662
663             StartupBoost();
664           } else {
665
666
667             //timeb = L*I1Startup/(Vin);
668
669             //WriteScreen("S");
670             //vcntrlb_16 = (unsigned int) (timeb * rampslope * 65536/5);  // ns * V/ns * code/V
```

```
            vcntrlb_16 = (unsigned int) (2.5 * 65536/5);  //Worked with 2 for a long time; need more juice
            WriteSPIb(&vcntrlb_16,CS);

            if(KickCount > 350){
              for(i=0;i<3;i++) {Nop();}
              StartupBoost();
              KickCount = 0;
            }

          }
          // </editor-fold>
        }

      //Boost State
      if(Flags.Running==1 && Flags.Mode==0){
        Trig_set;
        // <editor-fold defaultstate="collapsed" desc="Boost State">

        if(Vout<150){  //10000 => 76 V
          // <editor-fold defaultstate="collapsed" desc="Boost -> Charge">
          WriteScreen("C");
          Flags.Running = 0;
          Flags.Mode = 0;

          ENABLEb_set; //Enable B should be on
          //ENABLEa_set; //EnableA should be on
          MANUALb_clr; //Set Manual B to off
          //MANUAla_clr; //Set Manual A to off
          // </editor-fold>
        }
        else {//If no change required, then give another kick to keep going
          // <editor-fold defaultstate="collapsed" desc="Keep Boost">


          //WriteScreen("D");
          ttotal = 2*L/R * (1 + limitpercent*(3*I3+5*I5)*Vrms - limitpercent*(4*I3+20*I5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*I5)*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  ) + (Vout/Vin)*sqrt(L*Cres)*(sqrt(1 - 2*Vin/Vout)+1-Vin/Vout) + cap_sign * 2*L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
          //When I ran this before I didn't have the 2 before the C_in term, but it should be there from my notes
          //I'm trying with the factor of 2 but reducing C_in as a variable
          //This will clean up the HV mode (which clearly is acting like C_in is too big) and also matches what I think the actual C_in is better


          if (  (ttotal*rampslope) > 4.8 ) {
            ttotal = 4.8/rampslope;
          }
          else if( (ttotal*rampslope) < 0.21 && Vin < 100) {
            ttotal = 0.21/rampslope;
          }


          vcntrlb_16 = (unsigned int) (ttotal * rampslope * 65536/5);
          WriteSPIb(&vcntrlb_16,CS);

          if(KickCount > 1000){
            for(i=0;i<3;i++) {Nop();}
            StartupBoost();
            KickCount = 0;
          }
          // </editor-fold>
        }
        Trig_clr;
        // </editor-fold>
      }
      // </editor-fold>

    }  //End of timer interrupt



  } //End of infinite while

  return (EXIT_SUCCESS);
}



void __ISR_AT_VECTOR(_UART3_RX_VECTOR, IPL2AUTO) IntUart3Handler(void) {

  Flags.U3RX = 1;
  selection = U3RXREG; U3TXREG = selection;  //Echo selection to the screen
  //WriteScreen("hi");
  IFS4bits.U3RXIF = 0;

}

void __ISR_AT_VECTOR(_TIMER_1_VECTOR, IPL2AUTO) IntTimer1Handler(void) {

  IFS0bits.T1IF = 0;

}

void __ISR_AT_VECTOR(_TIMER_2_VECTOR, IPL2AUTO) IntTimer2Handler(void) {

  Flags.Timer = 1;
  IFS0bits.T2IF = 0;

}


void CheckReset(void) {

    if(RCON & 0x0003){WriteScreen("POR");}
  else if(RCON & 0x0002){WriteScreen("BOR");}
  else if(RCON & 0x0080){WriteScreen("MCLR");}
  else if(RCON & 0x0040){WriteScreen("SWR");}
  else if(RCON & 0x0200){WriteScreen("CONFIG");}
  else if(RCON & 0x0010){WriteScreen("WDT");}
  else if(RCON & 0x0020){WriteScreen("DMT");}
    RCON = 0;
}

void SetupADC(void){


  ADC3CFG = DEVADC3;
  ADC4CFG = DEVADC4;
  //Vout_low is on B9, Pin 22, AN49, (ADC4)
  //Vin_low is on B8, Pin 21, AN48, (ADC3)
  //Page 474 "When an alternate input is used as the input source for a
  //  dedicated ADC module, the data output is still read from the primary
  //  input data output register"


  //First thing, load calibration data into the CFG registers
  //DEVADC is stored in flash memory
  //The *only* thing the FRM says is that the user must copy the data
  //Examples in the FRM have only these exact commands



  //Check section 22.4.3 of FRM "Selecting the Format of ADC result
  //Fractional is left justified, while integer is right-justified
  ADCCON1bits.FRACT = 1;  //Fractional output (left justified, 16-bit number with zeros for LSB)
  ADC1MCON1bits.SIGN1 = 0;
  ADC1MCON1bits.SIGN4 = 0;

  ADCCON1bits.SIDL = 0;  //Keep running in idle mode
  ADCCON1bits.ARDYPMPEN = 0; //Charge pump disabled (for Vdd > 2.5, see P427)
  CFGCON8bits.IOANCPEN = 0;  //Charge pump disabled (for Vdd > 2.5, see P427, 596)
  ADCCON1bits.FSSCLKEN = 1; //Fast System Clock to ADC Control Clock Enable (see FRM 22 p.77)
  ADCCON1bits.FSPBCLKEN = 1; //Peripheral clock to ADC enable (specific conditions FRM22 p.77)

  ADCCON3bits.ADCSEL = 0b11; //Use SYSclk as input (can use FRC, PBCLK3, REFCLK3)
  ADCCON3bits.CONCLKDIV = 1; //SYSclk/2 = 32 MHz = TQ
  ADC3TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC3TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
  ADC4TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC4TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
  //Recall that 12-bit conversion takes 13 Tad
  //(8+13)/(16 MHz) = 1.3 us
  //With Timer1 running at 64 MHz, need to count to at least 84

  ADCCON3bits.VREFSEL = 0;  //Use AVdd and AVss as pos/neg references

  ADCTRGMODEbits.SH4ALT = 0b01;  //Use AN49 on ADC4 (vout_low)
  ADCTRGMODEbits.SH3ALT = 0b01;  //Use AN48 on ADC3 (vin_low)

  ADC1MCON1 = 0;  //All inputs use single-ended, unsigned data
  ADC1MCON2 = 0;
  ADC1MCON3 = 0;

  ADCGIRQEN1 = 0;  //No interrupts from any ADC
  ADCGIRQEN2 = 0;
```

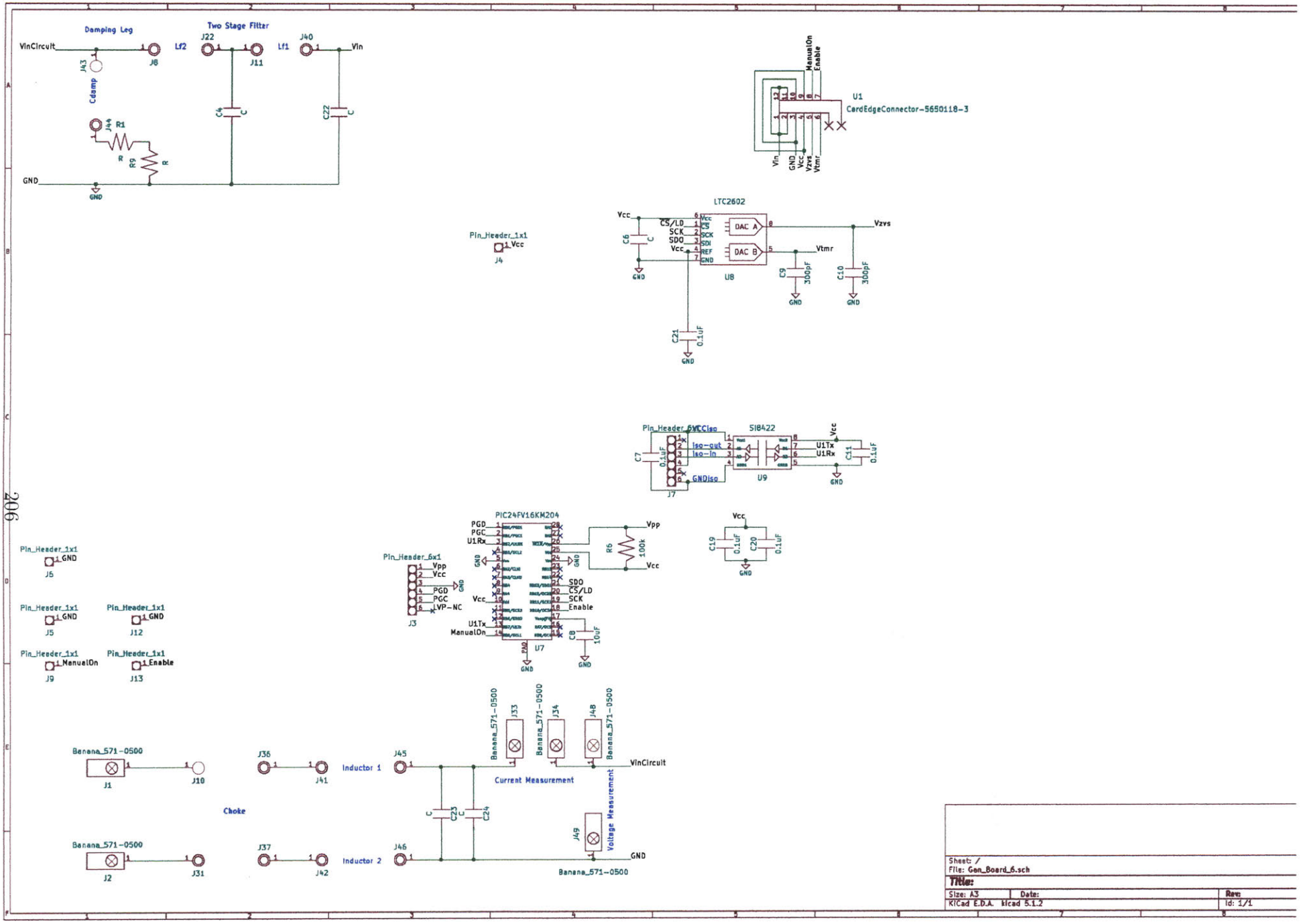
```
ADC3TIMEbits.SELRES = 0b11; //12 bit resolution
ADC4TIMEbits.SELRES = 0b11; //12 bit resolution

//From Section 22 of FRM: "Each Class 1 input has a unique trigger and
//  upon arrival of the trigger, ends sampling and starts conversion.
//  Upon completion of conversion, the ADC module reverts back to
//  sampling mode. When a Class 1 input is enabled and is not being
//  converted, it is always sampled."
//  Exception: SAMC is a minimum sample time. If it is not met when
//  a trigger arrives, the ADC will wait until it is met.
//  See FRM Sec 22.4.4.2.1-2

//ADC1CFG and ADC4CFG need to be written with DEVADC1, DEVADC4 prior to turn on?
//ADCFLTRx? May use in averaging or oversampling mode
//Could you use the AFRDY (or the standard RDY signal) to trigger a new conversion?
//  and effectively create a continual measurement scheme?
//  (might need to go through a CPU interrupt)
//Use ADCDATA1 and ADCDATA4 to access 46 and 49 (merely alternates)

ADCTRG1bits.TRGSRC3 = 0b00101; //AN3(=46?) using TMR1 match as trigger source
ADCTRG2bits.TRGSRC4 = 0b00101; //AN4(=49?) using TMR1 match as trigger source

ADCFSTAT = 0;  //FIFO and all associated properties are disabled


/*********************************************************/
T1CONbits.SIDL = 0;  //Run T1 even in idle mode
T1CONbits.TCS = 0;  //Run on peripheral clock PBCLK 3 (instead of external clock)
T1CONbits.TCKPS = 0b00; //1:1 prescaling
T1CONbits.TGATE = 0;
T1CONbits.TSYNC = 0;
PR1 = 100;  //Set this value to the number of counts.
        //Note that the ADC takes (SAMC+24)*Tad for sample/convert
        //Tad = 2*TQ based on ADCXTIME.ADCDIV
        //TQ = TCLK based on ADCCON3.CONCLKDIV
        //And lastly TCLK = FRC or System Clock based on ADCCON3.ADCSEL
        //So need roughly 30 Tad counts, or 60 TQ = 60 TCLK
        //So maybe let PBCLK3 count to 100

T1CONbits.ON = 1;  //Turn timer on

//IPC1bits.T1IP = 2;
//IEC0bits.T1IE = 1;


/*********************************************************/


//Digital and analog can be disabled to conserve power
//Digital starts up quickly and is easily enabled
//Further power saving by shutting down analog biasing, but takes time to start up again
//I assume the default is off, but better set them that way anyway
ADCANCONbits.ANEN0 = 0;
ADCANCONbits.ANEN1 = 0;
ADCANCONbits.ANEN2 = 0;
ADCANCONbits.ANEN3 = 1;  //ADC3 analog and bias circuitry enabled
ADCANCONbits.ANEN4 = 1;  //ADC4 analog and bias circuitry enabled
ADCANCONbits.ANEN7 = 0;

ADCCON3bits.DIGEN0 = 0;
ADCCON3bits.DIGEN1 = 0;
ADCCON3bits.DIGEN2 = 0;
ADCCON3bits.DIGEN3 = 1;  //ADC3 is digitally enabled (for vin_low)
ADCCON3bits.DIGEN4 = 1;  //ADC4 is digitally enabled (for vout_low)
ADCCON3bits.DIGEN7 = 0;


ADCCON1bits.ON = 1;  //Turn on last thing


}


void SetupSPIb(void) {

CSb set;   //idle High

SPI2CONbits.MSSEN = 0;  //Manually do slave select
SPI2CONbits.MCLKSEL = 0; //Use PBLK2 as CLK (as opposed to REFCLKO1)
SPI2CONbits.SIDL = 0; //Continue in idle mode

SPI2CONbits.MODE32 = 0; //These three bits define 8-bit communication
SPI2CONbits.MODE16 = 0;
SPI2CON2bits.AUDEN = 0; //AUDEN is for audio codecs

SPI2CONbits.CKE = 1;  //Change data on active->idle transition
SPI2CONbits.CKP = 0;  //Clock idle is low

SPI2CONbits.SSEN = 0; //SSx pin is unused (will manually do chip select)
SPI2CONbits.MSTEN = 1; //Master mode
SPI2CONbits.DISSDI = 1; //SDI pin is unused (never expecting to receive)

SPI2BRG = 0;  //Baud rate divisor (Fsck = Fpb/(2*(BRG+1))
        //Fastest possible is Fpb/2 (=32 MHz in this case)

SPI2CONbits.ON = 1;  //Turn on unit last thing


}

void SetupTimer(void) {
//This is the main timer which sets the time between DAC updates
//Not the same as Timer1 which is used to trigger the ADC (much faster)

T2CONbits.SIDL = 0;  //Run T1 even in idle mode
T2CONbits.TCS = 0;  //Run on peripheral clock PBCLK 3 (instead of extrnal clock)
T2CONbits.TCKPS = 0; //1:1 prescaling
T2CONbits.T32 = 0;  //Each timer is separate 16 bit timer, not combo 32 bit
//PR2 = 2048;  //Set this value to the number of counts.
PR2 = 2048;
//This is nominally the same as from the PIC24 code which used
//0b0000100000000000
//For PBCLK3 = SYSCLK = 64 MHz, 2048/64 MHz = 32 us
//Which is anywhere between 32 and 96 switching periods worth


T2CONbits.ON = 1;  //Turn timer on

IPC2bits.T2IP = 2;
IEC0bits.T2IE = 1;


}

void SetupUART2(void) {  //Using this UART for transmission on RB2

U2MODEbits.STSEL = 0;      //1 stop bit
U2MODEbits.PDSEL = 0b00;   //8-bit, no parity
U2MODEbits.BRGH = 0;       //High speed mode
U2MODEbits.UEN = 0b00;      //Use TX and RX but not RTS/CTS/BCLK
U2MODEbits.RXINV = 0;      //Idle state is high


U2BRG = 72;  //Based on PBCLK2 = SYSCLK (= 64 MHz)
        //See FRM 21.3 for tables


U2MODEbits.ON = 1;
U2STAbits.UTXEN = 1;
U2STAbits.URXEN = 1;
U2STAbits.URXISEL = 0;  //Interrupt asserted while buffer has any characters


IFS4bits.U2RXIF = 0;
IFS4bits.U2TXIF = 0;
IEC4bits.U2RXIE = 1;
IEC4bits.U2TXIE = 0;
IPC36bits.U2RXIP = 0b010;


}


void SetupUART3(void) {  //Using this UART for transmission on RB2

U3MODEbits.STSEL = 0;      //1 stop bit
U3MODEbits.PDSEL = 0b00;   //8-bit, no parity
U3MODEbits.BRGH = 0;       //High speed mode
U3MODEbits.UEN = 0b00;      //Use TX and RX but not RTS/CTS/BCLK
U3MODEbits.RXINV = 0;      //Idle state is high


U3BRG = 72;  //Based on PBCLK2 = SYSCLK (= 64 MHz)
        //See FRM 21.3 for tables

```

```c


    U3MODEbits.ON = 1;
    U3STAbits.UTXEN = 1;
    U3STAbits.URXEN = 1;
    U3STAbits.URXISEL = 0;  //Interrupt asserted while buffer has any characters



    IFS4bits.U3RXIF = 0;
    IFS4bits.U3TXIF = 0;
    IEC4bits.U3RXIE = 1;
    IEC4bits.U3TXIE = 0;
    IPC39bits.U3RXIP = 0b010;


}


void StartupBoost(void){
    int i;

    //MANUALa_clr; //Make sure ManualON A is off
    MANUALb_clr;  //Make sure ManualON B is off


    ENABLEb_set;
    //ENABLEa_set;


    MANUALb_set;  //Turn on B
    //MANUALa_set;  //Turn on A -- this opens a path through the inductor
    for(i=0; i<7; i++) {Nop();}

    ENABLEb_clr;  //Quickly disable B before reset action
    MANUALb_clr;  //Remove Manual B


}

void TriggerPulse(void) {
    Trig_set;
    Trig_clr;
}

void WriteFlags(unsigned int running, unsigned int mode ) {

    //unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    //float ADCValF;
    //ADCVal = ADCDATA1;
    //ADCValF = (double) (ADCDATA4>>16);  //Convert 32 bit to 16 bit
    //ADCValF = ADCValF / 1.127;


    //ADCValF = 3.3 * ADCValF / (0x100000000);
    char buf[16];
    sprintf(buf, "%d", running);
    WriteScreen("\n\rRunning  = ");
    WriteScreen(buf);

    sprintf(buf, "%d", mode);
    WriteScreen("\n\rMode  = ");
    WriteScreen(buf);

}


void WriteKickCount(unsigned int kickcount) {

    //unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    //float ADCValF;
    //ADCVal = ADCDATA1;
    //ADCValF = (double) (ADCDATA4>>16);  //Convert 32 bit to 16 bit
    //ADCValF = ADCValF / 1.127;


    //ADCValF = 3.3 * ADCValF / (0x100000000);
    char buf[16];
    sprintf(buf, "%d", kickcount);
    WriteScreen("\n\rRunning  = ");
    WriteScreen(buf);

}


void WriteMenu(void) {

    WriteScreen("\n\n\n\n\r ***Modified Boost Menu***");
    WriteScreen("\n\rq:InvLimit%d;InvLimit%d");
    WriteScreen("\n\rs:InvSysAdd;InvSysSub");

    WriteScreen("\n\rp:ControlOld");
    WriteScreen("\n\rd:WriteOutDac;WriteAtmo");
    WriteScreen("\n\rg:IncSPIbDac");
    WriteScreen("\n\rh:WriteFlags\n\rInc12");

    WriteScreen("\n\rit");


}


void WriteScreen(char s[50]) { //Using UART2 on RB2 for transmitting

    char *p;
    p = s;
    while (*p) {
        while (!(U2STAbits.TRMT)) {
        }
        U2TXREG = *p++;
    }

}


void WriteSPIb(unsigned int* dacinput, char select) {

    int i;
    int mode;
    unsigned int temp;

    switch (select) {
        case 'c':
            mode = 0b0000000000110001; //Control word to select right DAC for VcntrlB
            break;
        case 'v':
            mode = 0b0000000000110000; //Control word to select right DAC for VzvsB
            break;
        default:
            ;
    }


    CSb_clr;  //Chip select B active low

    //Transmit 8-bit mode word

    SPI2BUF = mode; while(SPI2STATbits.SPITBE == 0){}

    temp = *dacinput;
    temp = temp >> 8;

    //Transmit 8-bit data word

    SPI2BUF = temp; while(SPI2STATbits.SPITBE == 0){}


    temp = (*dacinput) & 0x0011;

    //Transmit 8-bit data word

    SPI2BUF = temp; while(SPI2STATbits.SPITBE == 0){}
    for (i=1; i<10; i++) {Nop();}  //Necessary delay for LTC2602 timing

    CSb_set;


}

void WriteVin(double vin) {


    //unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    //float ADCValF;
    //ADCVal = ADCDATA4;
    //ADCValF = (double) (ADCDATA3>>16) ;
    //ADCValF = ADCValF / 1.127;
```

```
  //ADC Values are saved as
  // dddd dddd dddd 0000 0000 0000 0000 0000
  //So divide by 2^32 to get fraction of full voltage which is 3.3


  char buf[16];
  sprintf(buf, "%f", vin);
  WriteScreen("\n\r\tVin = ");
  WriteScreen(buf);

}

void WriteVout(double vout) {

  //unsigned int ADCVal;   //int is 32 bit in XC32 compilor;  short's are 16 bit
  //float ADCValF;
  //ADCVal = ADCDATA1;
  //ADCValF = (double) (ADCDATA4>>16);  //Convert 32 bit to 16 bit
  //ADCValF = ADCValF / 1.127;


  //ADCValF = 3.3 * ADCValF / (0x100000000);
  char buf[16];
  sprintf(buf, "%f", vout);
  WriteScreen("\n\r\tvout = ");
  WriteScreen(buf);

}
```

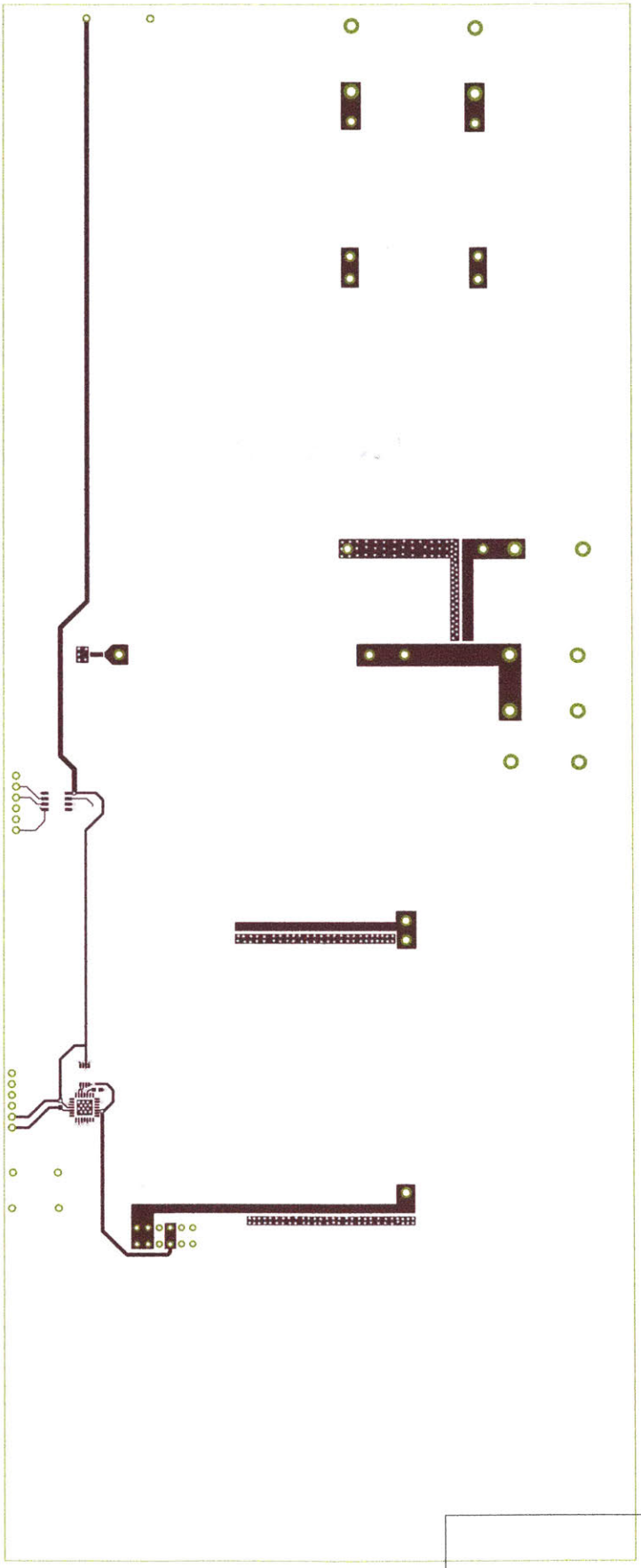
Appendix O

GaN Measurement Filter Board Schematic

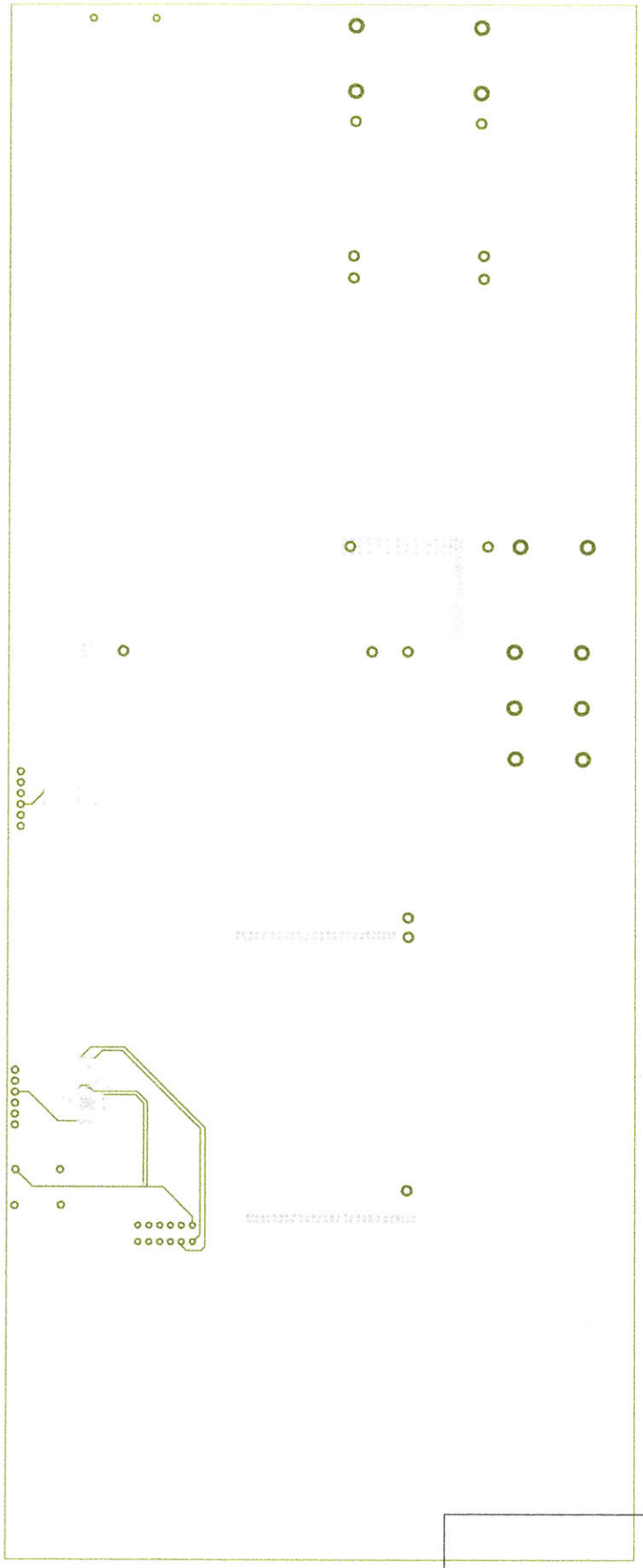

Damping Leg
Two Stage Filter
VinCircuit
Lf2
J22
Lf1
J40
Vin
J43
J8
J11
Cdamp
C4
C
C22
J44
R1
R
R9
R
GND
U1
CardEdgeConnector-5650118-3
ManualOn
Enable
Vin
GND
Vcc
Vzvs
Vtmr
Pin_Header_1x1
J4
LTC2602
Vcc
CS/LD
SCK
SDO
REF
DAC A
DAC B
Vzvs
Vtmr
C6
C9
300pF
C10
300pF
U8
C21
0.1uF
Pin_Header_6x1
J7
VCCiso
Iso-out
Iso-in
GNDiso
C7
0.1uF
SI8422
U9
U1Tx
U1Rx
C11
0.1uF
PIC24FV16KM204
PGD
PGC
U1Rx
Vcc
U1Tx
ManualOn
SDO
CS/LD
SCK
Enable
Vpp
R6
100k
C8
10uF
U7
Pin_Header_6x1
J3
Vpp
Vcc
PGD
PGC
LVP-NC
C19
0.1uF
C20
0.1uF
Pin_Header_1x1
J6
GND
Pin_Header_1x1
J5
GND
Pin_Header_1x1
J12
GND
Pin_Header_1x1
J9
ManualOn
Pin_Header_1x1
J13
Enable
Banana_571-0500
J1
J10
Banana_571-0500
J2
J31
Choke
J36
J41
J37
J42
Inductor 1
Inductor 2
J45
J46
C
C25
C24
Banana_571-0500
J33
Banana_571-0500
J34
J48
Banana_571-0500
Current Measurement
VinCircuit
Voltage Measurement
J49
GND
Banana_571-0500
Sheet: /
File: Gen_Board_6.sch
Title:
Size: A3
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 1/1

Appendix P

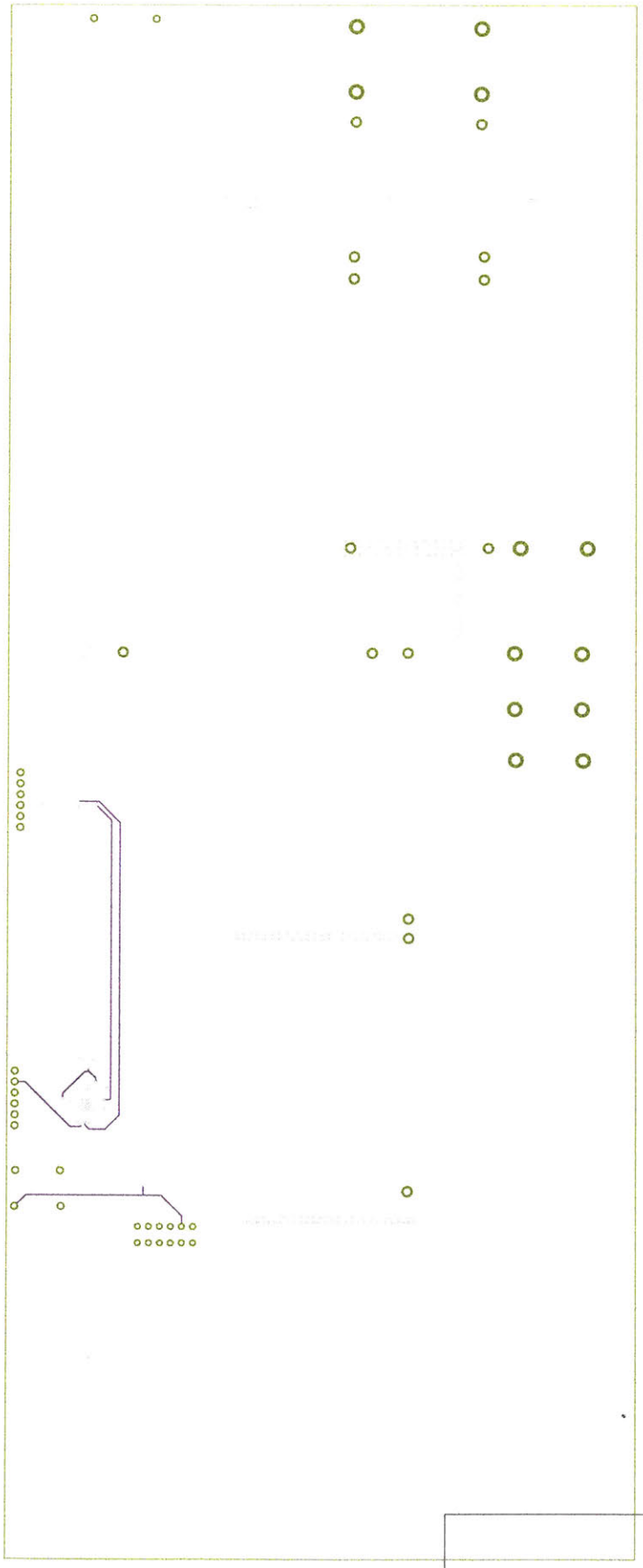
GaN Measurement Filter Board Layout



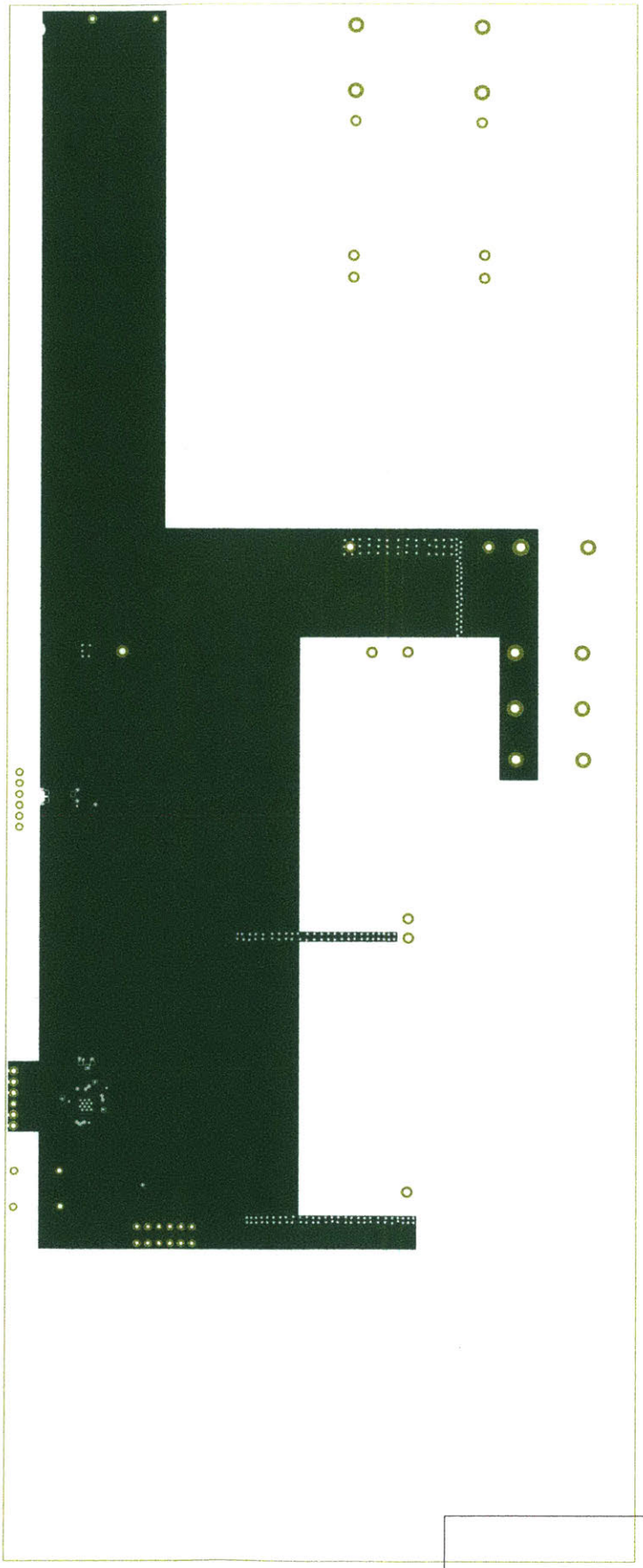
Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3 | Date: | Rev:
KiCad E.D.A. kicad 5.1.2 | Id: 1/1



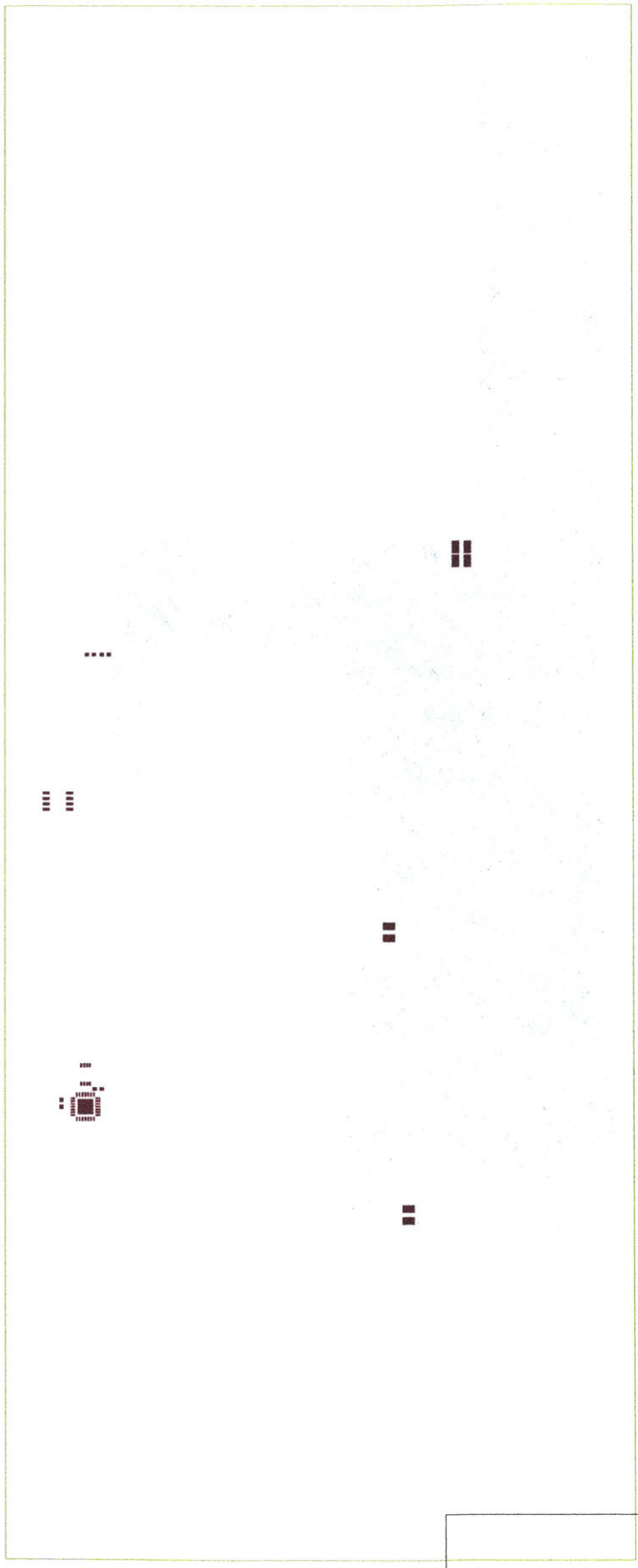
Sheet:	
File: Gan_Board_6.kicad_pcb	
Title:	
Size: A3 Date:	Rev:
KICad E.D.A. kicad 5.1.2	Id: 1/1



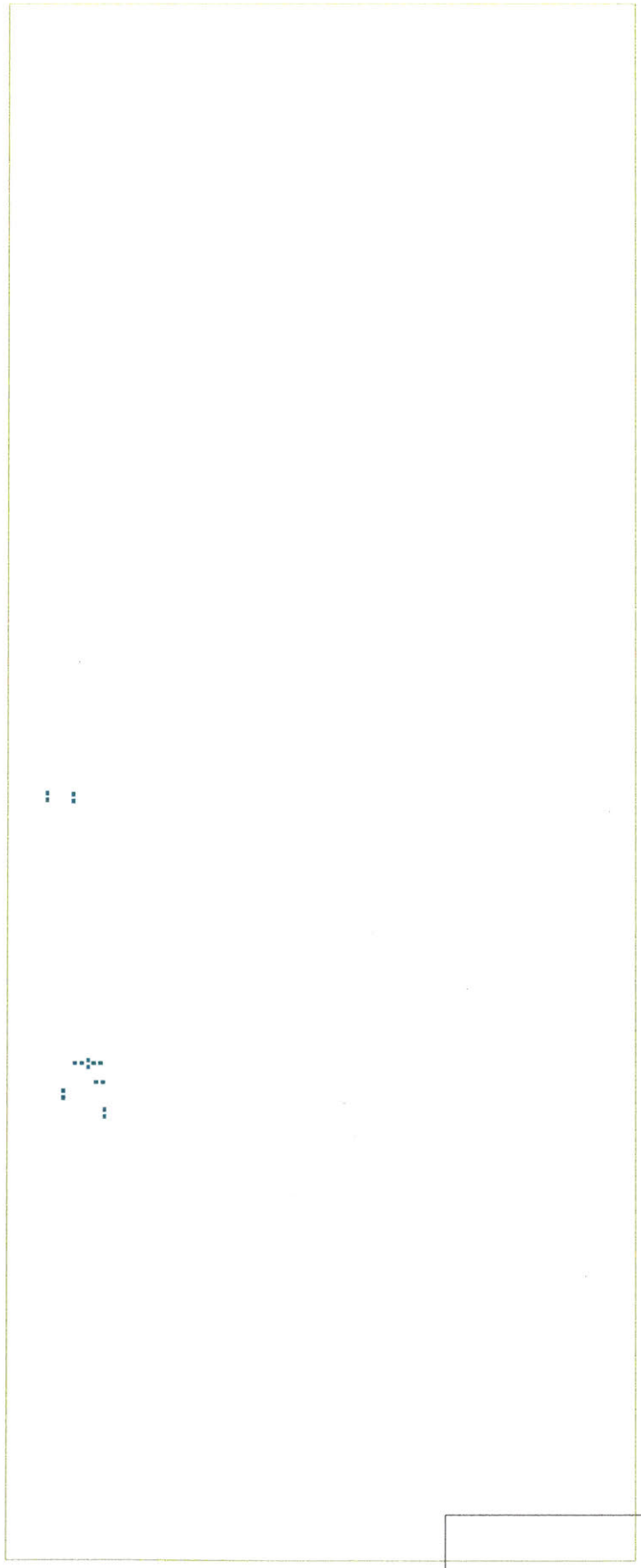
Sheet:
File: Gen_Board_6.kicad_pcb
Title:
Size: A3 | Date: | Rev:
KiCad E.D.A. kicad 5.1.2 | Id: 1/1



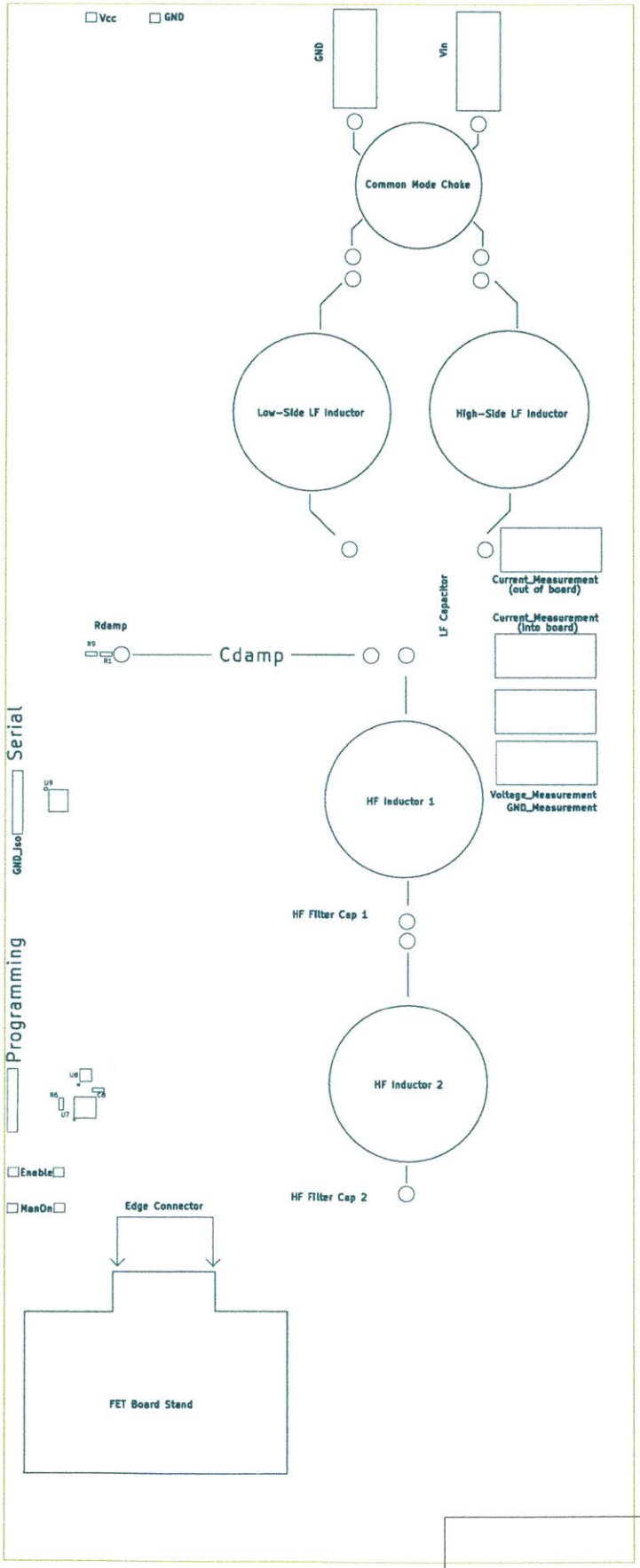
Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3 Date: Rev:
KiCad E.D.A. kicad 5.1.2 Id: 1/1



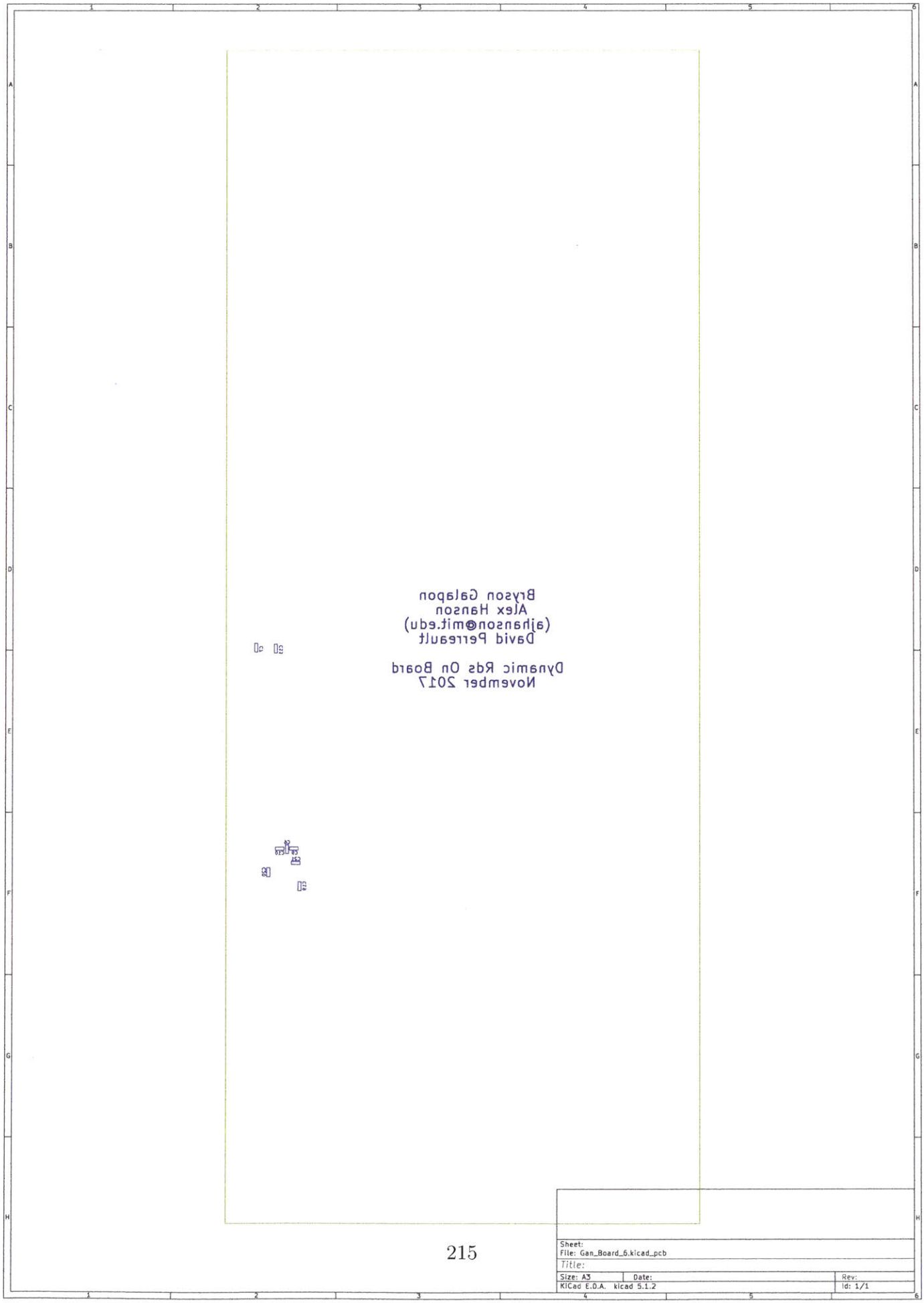
Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3 | Date: | Rev:
KiCad E.D.A. kicad 5.1.2 | Id: 1/1



Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3 Date: Rev:
KiCad E.D.A. kicad 5.1.2 Id: 1/1

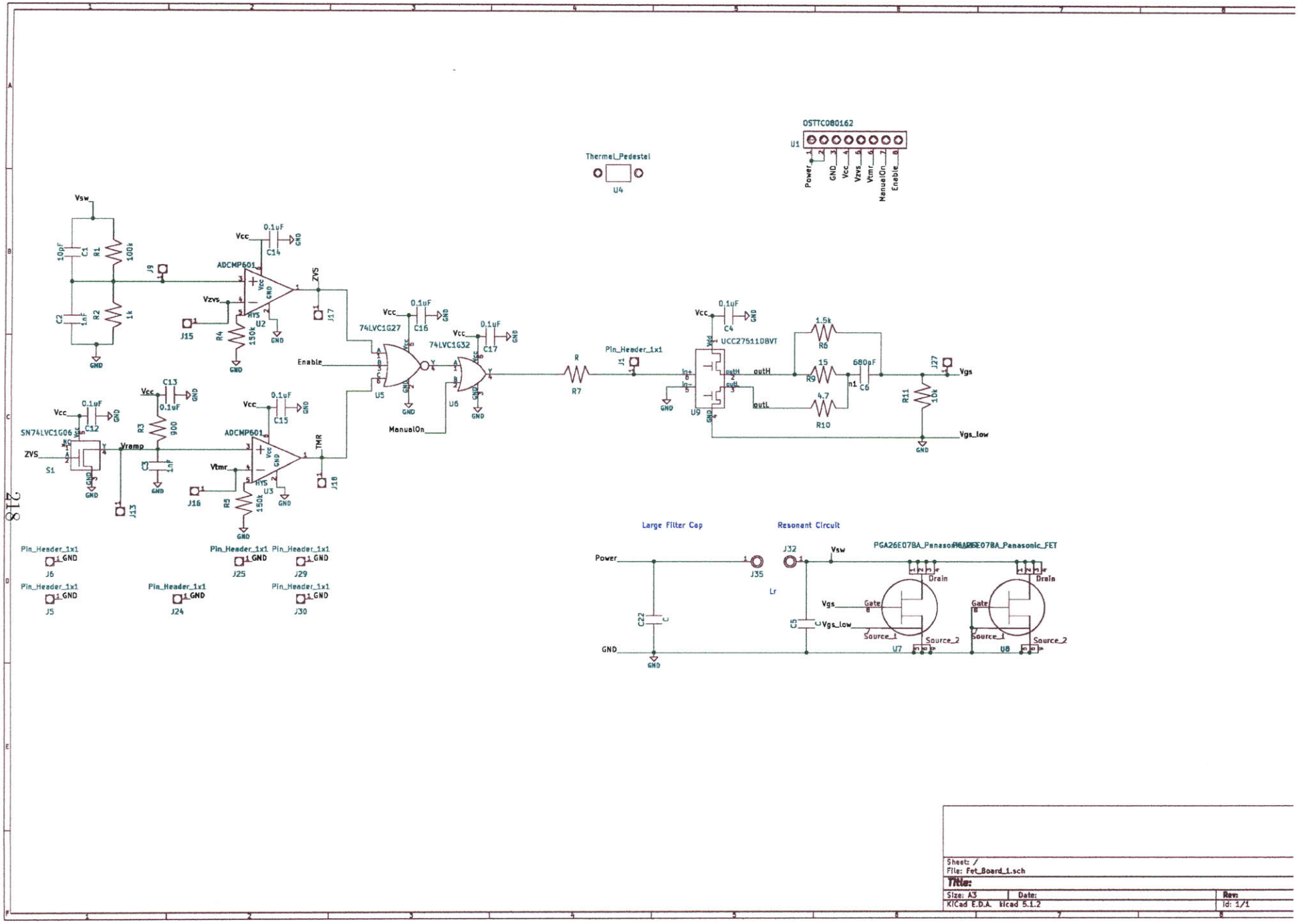


Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3 | Date: | Rev:
KiCad E.D.A. kicad 5.1.2 | Id: 1/1


Bryson Galapon
Alex Hanson
(ajhanson@mit.edu)
David Perreault
Dynamic Rds On Board
November 2017
Sheet:
File: Gan_Board_6.kicad_pcb
Title:
Size: A3
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 1/1

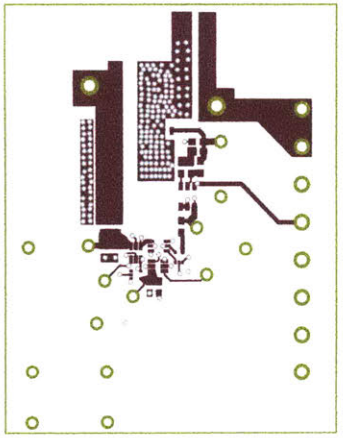
Appendix Q

GaN Measurement FET Board Schematic

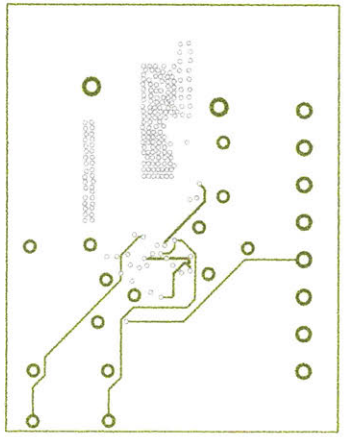

OSTTC080162
U1
Power
GND
Vcc
Vzvs
Vtmr
ManualOn
Enable
Thermal_Pedestal
U4
Vsw
10pF
C1
R1
100k
C2
1nF
R2
1k
J9
GND
Vcc
0.1uF
C14
ADCMP601
U2
Vzvs
J15
R4
150k
ZVS
J17
Vcc
0.1uF
C16
74LVC1G27
U5
Enable
ManualOn
Vcc
0.1uF
C17
74LVC1G32
U6
R
R7
Pin_Header_1x1
J1
Vcc
0.1uF
C4
UCC27511DBVT
U9
outH
outL
1.5k
R6
15
R9
680pF
C6
4.7
R10
J27
Vgs
R11
10k
Vgs_low
Vcc
C13
0.1uF
R3
900
C3
1nF
Vcc
0.1uF
C12
SN74LVC1G06
S1
ZVS
Vramp
J13
Vcc
0.1uF
C15
ADCMP601
U3
Vtmr
J16
R5
150k
TMR
J18
Pin_Header_1x1
J6
J5
J25
J29
J24
J30
Large Filter Cap
Resonant Circuit
Power
J35
C22
GND
J32
Vsw
Lr
C5
Vgs
Vgs_low
PGA26E07BA_Panasonic_FET
Drain
Gate
Source_1
Source_2
U7
U8
Sheet: /
File: Fet_Board_1.sch
Title:
Size: A3
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 1/1

Appendix R

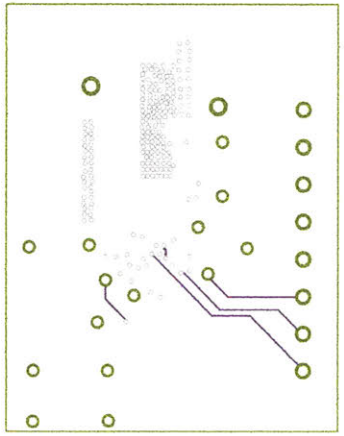
GaN Measurement FET Board Layout



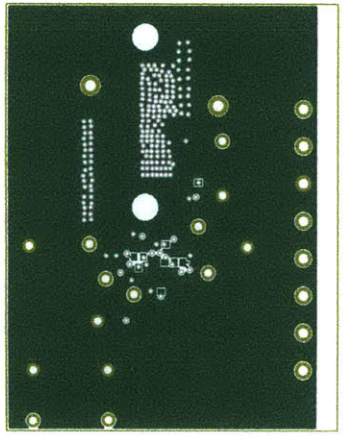
Sheet: File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



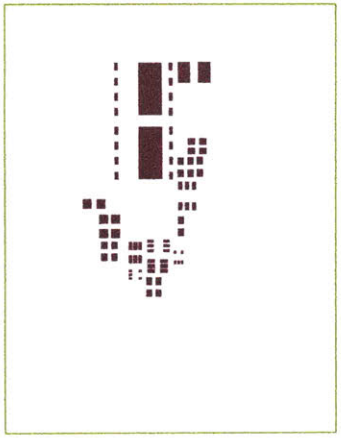
Sheet:		
File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



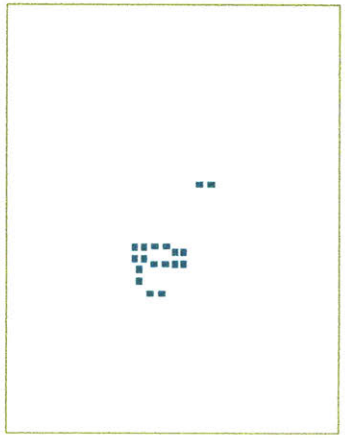
Sheet: File: Fet_Board_1.kicad_pcb	
Title:	
Size: USLetter Date:	Rev:
KiCad E.D.A. kicad 5.1.2	Id: 1/1



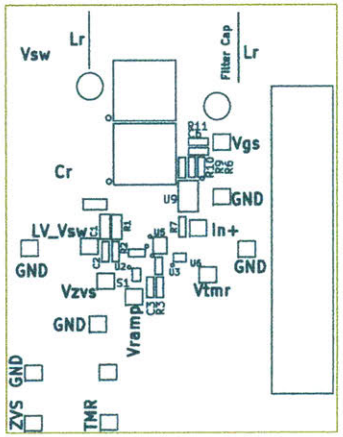
Sheet: File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



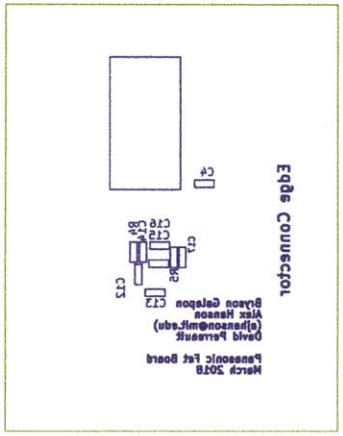
Sheet: File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet: File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet:		
File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet: File: Fet_Board_1.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1

Appendix S

GaN Measurement Code

/home/alex/Dropbox (MIT)/Dynamic Ron/Dynamic Ron Share/uC Code 3 - interrupt/modboostinterrupt.c

```
/*
 * File:   modifiedboost.c
 * Author: Alex
 *
 *
 * For Bryson, GaN code
 *
 * Created on December 9, 2015, 10:56 AM
 */

#include "p24FV16KM202.h"
#include "stdio.h"
#include "string.h"


// <editor-fold defaultstate="collapsed" desc="Configuration Bits Setup">



// Configuration Bits to make the part run from Internal FRCDIV
// Oscillator.
_FBS
(
        BWRP_OFF & // Boot Segment Write Protect (Disabled)
        BSS_OFF // Boot segment Protect (No boot flash segment)
        )

_FGS
(
        GWRP_OFF & // General Segment Flash Write Protect (General segment may be written)
        GCP_OFF // General Segment Code Protect (No Protection)
        )

_FOSCSEL
(
        FNOSC_FRCPLL &
        //FNOSC_FRCDIV &  // Oscillator Select (8MHz FRC with Postscaler (FRCDIV))
        //Note that the default for CLKDIV is to divide by 2 (4 MHz clock => Fcy = 2 MHz)
        SOSCSRC_DIG & // SOSC Source Type (Analog Mode for use with crystal)
        LPRCSEL_HP // LPRC Power and Accuracy (High Power/High Accuracy)
        & IESO_OFF // Internal External Switch Over bit (Internal External Switchover mode enabled (Two-speed Start-up enabled))
        )

_FOSC
(
        POSCMOD_NONE & // Primary Oscillator Mode (Primary oscillator disabled)
        //OSCIOFNC_IO &   // CLKO Enable Configuration bit (CLKO output signal enabled)
        POSCFREQ_MS & // Primary Oscillator Frequency Range Configuration bits (Primary oscillator/external clock frequency between 100kHz to 8MHz)
        //SOSCSEL_SOSCHP &// SOSC Power Selection Configuration bits (Secondary Oscillator configured for high-power operation)
        FCKSM_CSECME // Clock Switching and Monitor Selection (Clock Switching and Fail-safe Clock Monitor Enabled)
        )

_FWDT
(
        WDTPS_PS32768 & // Watchdog Timer Postscale Select bits (1:32768)
        FWPSA_PR128 & // WDT Prescaler bit (WDT prescaler ratio of 1:128)
        FWDTEN_OFF & // Watchdog Timer Enable bits (WDT disabled in hardware; SWDTEN bit disabled)
        WINDIS_OFF // Windowed Watchdog Timer Disable bit (Standard WDT selected (windowed WDT disabled))
        )

// Warning:
// Always enable MCLRE_ON config bit setting so that the MCLR pin function will
// work for low-voltage In-Circuit Serial Programming (ICSP). The Microstick
// programming circuitry only supports low-voltage ICSP. If you disable MCLR pin
// functionality, a high-voltage ICSP tool will be required to re-program the
// part in the future.
_FPOR
(
        BOREN_BOR3 & // Brown-out Reset Enable bits (Enabled in hardware; SBOREN bit disabled)
        PWRTEN_ON & // Power-up Timer Enable (PWRT enabled)
        I2C1SEL_PRI & // Alternate I2C1 Pin Mapping bit (Default SCL1/SDA1 Pins for I2C1)
        BORV_V18 // Brown-out Reset Voltage bits (Brown-out Reset at 1.8V)
        // & MCLRE_ON        // MCLR Pin Enable bit (RA5 input disabled; MCLR enabled)
        )

_FICD
(
        ICS_PGx3 // ICD Pin Placement Select (EMUC/EMUD share PGC3/PGD3)
        )
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Function Prototypes">
```

```c

void SetupSPIa(void);
void WriteSPIa(unsigned int* dacinput, char select);
void SetupUART(void);
void WriteScreen(char s[50]);
void StartupCycle(void);
void WriteVcntrla(void);
void WriteMenu();
void SetupTimer(void);

    // </editor-fold>



typedef struct {
  unsigned Timer1 :1;
  unsigned ADC1 :1;
   unsigned U1RX :1;
   unsigned Auto :1;
   unsigned WriteTimes :1;
   unsigned BitSix :1;
   unsigned BitSeven :1;
   unsigned BitEight :1;
}InterruptFlags;

static volatile InterruptFlags IntFlags;



int main(void) {

  IntFlags.Timer1 = 0;
  IntFlags.ADC1 = 0;
  IntFlags.U1RX = 0;
  IntFlags.WriteTimes = 0;
  IntFlags.BitSix = 0;
  IntFlags.BitSeven = 0;
  IntFlags.BitEight = 0;



  //EnableA should be digital, output, on
  TRISBbits.TRISB10 = 0;
  LATBbits.LATB10 = 1;

  //Set Manual A to digital, "off"
  ANSBbits.ANSB8 = 0; //Set pin 42 (OC1F) to digital output  (manual A)
  TRISBbits.TRISB8 = 0;
  LATBbits.LATB8 = 0; //Should always be off unless PWM-ing

  CLKDIV = 0x0000; //Stop dividing clock by 2 (to achieve 32 MHz)


  //These variables go from 0 to 2^16 = 65,536.
  //Vcntrl values indicate times ranging from 0 to .
  //Vzvs values indicate turn-on triggers from 0 to 500 V. (x100 step down ratio)


  unsigned int vcntrla = 23000;
  unsigned int vzvsa = 6000;


  //Make B2 and B7 (U1RX,U1TX) digital pins so UART can take them over
  ANSBbits.ANSB2=0;
  //TRISBbits.TRISB2 = 1;
  LATBbits.LATB2=0;

  ANSBbits.ANSB7=0;
  //TRISBbits.TRISB7 = 0;
  LATBbits.LATB7=0;



  SetupUART();
  SetupSPIa();
  WriteSPIa(&vzvsa,'z');
  WriteSPIa(&vcntrla,'c');
  //SetupTimer();




  char selection;

```

```
168   WriteMenu();
169
170   while (1) {
171
172
173     if(IntFlags.U1RX == 1) {
174
175       WriteScreen("\n\r");
176       selection = U1RXREG; //Echo selection to the screen
177       U1TXREG = selection;
178
179       switch (selection) {
180         case 'a':
181           WriteScreen("No B versions");
182            break;
183         case 'z':
184           WriteScreen("No B versions");
185           break;
186         case 's':
187           vzvsa = vzvsa + 300;
188           WriteSPIa(&vzvsa,'z');
189
190           break;
191         case 'x':
192           vzvsa = vzvsa - 300;
193           WriteSPIa(&vzvsa,'z');
194           break;
195         case 'q':
196           StartupCycle();
197           break;
198         case 'd':
199           WriteScreen("No ADC Function");
200           break;
201         case 'c':
202           WriteScreen("No ADC Function");
203           break;
204         case 'g':
205           WriteScreen("No B versions");
206           break;
207         case 'b':
208           WriteScreen("No B versions");
209           break;
210         case 'h':
211           vcntrla = vcntrla + 500;
212           WriteSPIa(&vcntrla,'c');
213           break;
214         case 'n':
215           vcntrla = vcntrla - 500;
216           WriteSPIa(&vcntrla,'c');
217           break;
218         case 'e':
219           WriteScreen("No Auto/Manual");
220           break;
221         case 'y':
222           WriteScreen("No Auto/Manual");
223           break;
224         default:
225           WriteScreen("Invalid Selection");
226       }
227
228       WriteMenu();
229
230       IntFlags.U1RX = 0;
231
232     } // End of if(U1RX)
233
234
235     if(IntFlags.Timer1 == 1) {
236
237       IntFlags.Timer1 = 0;
238     }  //End of timer interrupt
239
240
241
242   } // End of infinite while
243
244   return 0;
245 }
246
247
248
249
250
251
252
```

```
253 void __attribute__((__interrupt__, no_auto_psv)) _U1RXInterrupt(void) {
254     IntFlags.U1RX = 1;
255
256     IFS0bits.U1RXIF = 0;
257 }
258
259
260
261 void __attribute__((__interrupt__, no_auto_psv)) _T1Interrupt(void) {
262
263     IntFlags.Timer1 = 1;
264
265     IFS0bits.T1IF=0;  //Clear interrupt flag
266 }
267
268
269 void SetupSPIa(void) {
270
271     //SCKa
272     //ANSBbits.ANSB11 = 0;
273     TRISBbits.TRISB11 = 0;
274
275     //SDOa
276     ANSBbits.ANSB13 = 0;
277     TRISBbits.TRISB13 = 0;
278
279     //CS-LDa
280     ANSBbits.ANSB12 = 0; // (pin 15)
281     TRISBbits.TRISB12 = 0;
282     LATBbits.LATB12 = 1; //Idle high
283
284     //A functions are on SSP1 (SCK1,SDO1)
285     SSP1STAT = 0b0000000001000000;
286     //Status register for MSSP2
287     //Bit 6: CKE - some confusion on this point, but setting to 1 to match graph on 58 page 18
288
289     SSP1CON1 = 0b0000000000000000;
290     //Bit 15-8: Unimplemented (00000000)
291     //Bit 7: WCOL No Collision (0)
292     //Bit 6: SSPOV No overflow (0)
293     //Bit 5: SSPEN Not enabled (0) (will enable last thing)
294     //Bit 4: CKP clock idle low (0)
295     //Bit 3-0: SSPM<3:0> SPI master mode with clock Fosc/2 = Fcy (0000)
296
297     SSP1CON3 = 0b0000000000000000;
298     //Bit 15-8: Unimplemented (00000000)
299     //Bit 7: ACKTIM unused in SPI (0)
300     //Bit 6: PCIE unused in SPI (0)
301     //Bit 5: SCIE unused in SPI (0)
302     //Bit 4: BOEN unused in SPI master (0)
303     //Bit 3: SDAHT unused in SPI (0)
304     //Bit 2: SBCDE unused in SPI (0)
305     //Bit 1: AHEN unused in SPI (0)
306     //BIt 0: DHEN unused in SPI (0)
307
308     SSP1CON1bits.SSPEN = 1;
309 }
310
311 void SetupTimer(void) {
312     T1CON = 0b0000000000110000;
313     //Bit 15: TON turns on or off (turn on last thing)
314     //Bit 14: Unimplemented (0)
315     //Bit 13: TSIDL continues in idle mode (0)
316     //Bit 12-10: Unimplemented (000)
317     //Bit 9-8: TECS extended clock option, set to 00 but unused if TCS=0
318     //Bit 7: Unimplemented (0)
319     //Bit 6: TGATE enables gating, unused if TCS=0
320     //Bit 5-4: TCKPS prescale to 1:1 (00) or 1:256 (11)
321     //Bit 3: Unimplemented (0)
322     //Bit 2: TSYNC external synchronization, unused if TCS=0
323     //Bit 1: TCS - use the internal clock Fosc/2  (0)
324     //Bit 0: Unimplemented (0)
325
326
327     //Note that Fosc = 32 MHz after postscaling, so Fcy=Fosc/2 = 16 MHz.
328     //To achieve 1 kHz update frequency, scale by
329     //      a factor of 16000 = 0b0011111010000000;
330     //For testing, use a 1 second update frequency, scale by
331     //      a factor of 0xFFFF after a prescale of 1:256
332     //PR1 = 0b0011111010000000;
333     PR1 = 0b0000100000000000; // About 0.13s refresh rate (0.25s was a little clumsy)
334
335
336     T1CONbits.TON=1;  //Turns on the module.
337     IFS0bits.T1IF=0;
```

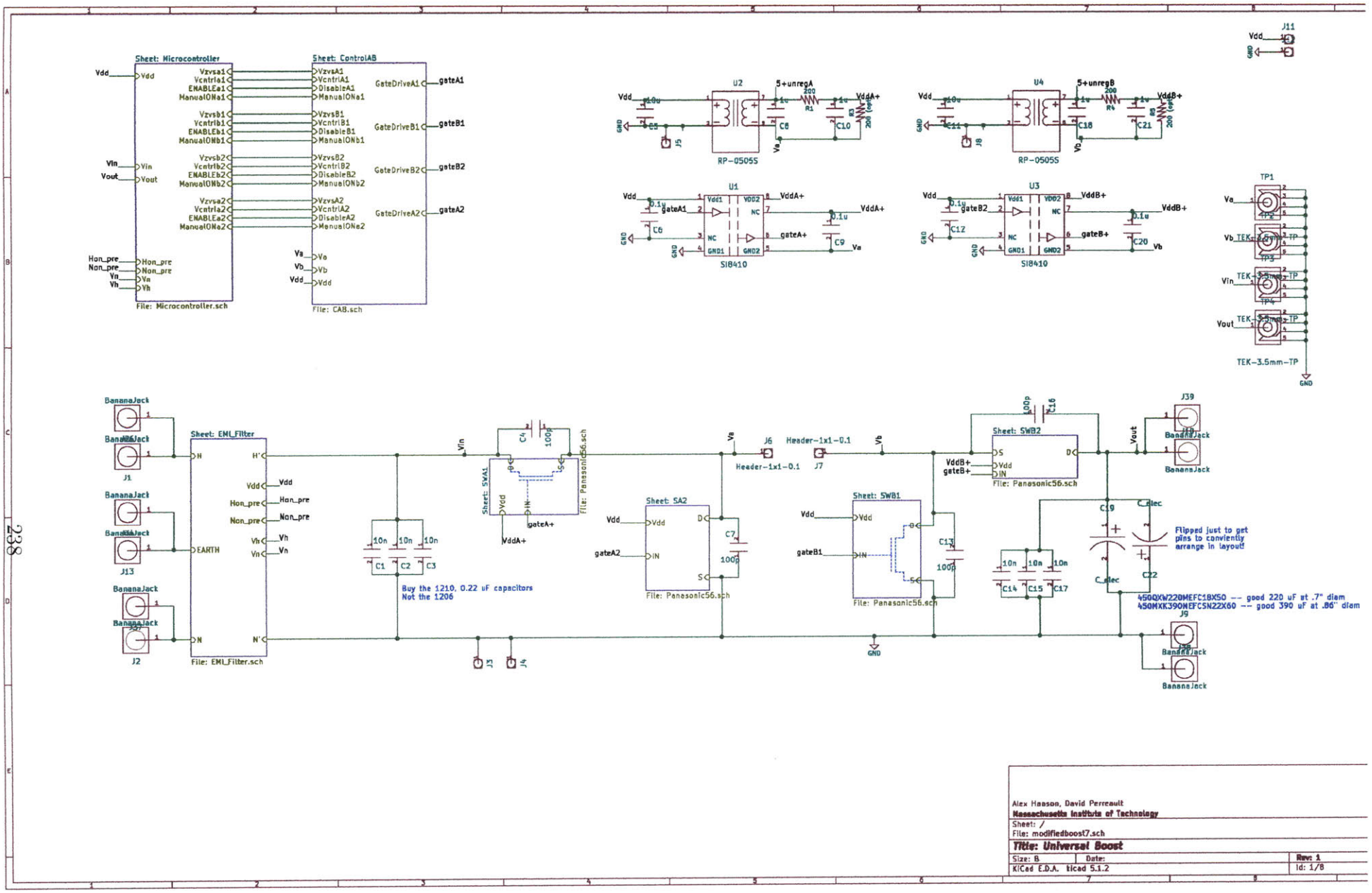
```
    IEC0bits.T1IE=0; //Enable interrupt

}



void WriteSPIa(unsigned int* dacinput, char select) {
    //Write Sequence
    //Writing to SSP1BUF should get 8 bits into the transmit register
    //Writing 3 times should transmit 24 bits, or one "word" for the LTC2602
    int i;
    unsigned int temp;
    //while(SSP1STATbits.BF == 1);  This line (sometimes) caused infinite delays

    int mode;



    //    char buf[10];
    //    sprintf(buf,"%u",*dacinput);
    //    WriteScreen("\n\r");
    //    WriteScreen(buf);

    switch (select) {
      case 'c':
        mode = 0b0000000000110001; //VcntrlB
        break;
      case 'z':
        mode = 0b0000000000110000; //VcntrlB
        break;
      default:
        ;
    }



    LATBbits.LATB12 = 0; //Active low

    SSP1BUF = mode;
    //SSP1BUF = 0x0000 & p[0];
    //Data out
    //Bit 15-8: Unused for 8 bit SPI I think
    //Bit 7-4: COMMAND C<3:0> (0011) - Write and update module n
    //Bit 3-0: ADDRESS A<3:0> )0001) - DAC B

    //while(SSP1STATbits.BF == 1);
    for (i = 0; i < 20; i++) {
    }

    temp = *dacinput;
    temp = temp >> 8;
    //temp = temp & 0x0011;

    //    sprintf(buf,"%u",temp);
    //    WriteScreen("\n\r");
    //    WriteScreen(buf);

    //SSP1BUF = 0b0000000011010001;
    //SSP1BUF = 0x0000 & temp;
    SSP1BUF = temp;
    //Bit 15-8: Unused for 8 bit SPI I think
    //Bit 7-0; 8 MSBs of data

    for (i = 0; i < 20; i++) {
    }

    //while(SSP1STATbits.BF == 1);

    temp = (*dacinput) & 0x0011;
    //SSP1BUF = 0b0000000001110100;
    //SSP1BUF = 0x00 & p[0];
    SSP1BUF = temp;
    //Bit 15-8: Unused for 8 bit SPI I think
    //Bit 7-0; 8 LSBs of data

    for (i = 0; i < 20; i++) {
    }

```

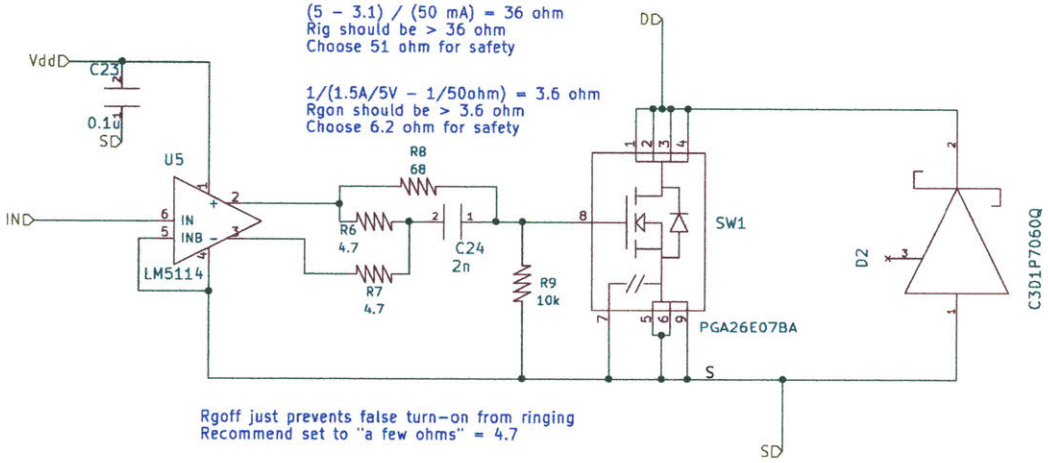
```
  LATBbits.LATB12 = 1; //back to idle high
  //TriggerPulse();
}

void SetupUART(void) {
  U1MODE = 0b0100000010000000;
  //Bit 15: UEN - UART disable (0) (will turn on last)
  //Bit 14: UFRZ - freeze in debug mode on (1)
  //Bit 13: USIDL - Do not stop in idle mode (0)
  //Bit 12: IREN - IrDA disabled (0)
  //Bit 11: RTSMD - flow control mode (0) (won't use those pins anyway)
  //Bit 10: ALTIO - do not use alternate pins (0)
  //Bit 9-8: UEN<1:0> - Enable RX and TX pins; not CTS, RTS or BCLK (00)
  //Bit 7: WAKE - wake up during sleep enabled (1)
  //Bit 6: LPBACK - loopback mode disabled (0)
  //Bit 5: ABAUD - one-time auto baud measurement not taking place (0)
  //Bit 4: RXINV - idle state is '1' (0)
  //Bit 3: BRGH - low speed (0)
  //Bit 2-1: PDSEL<1:0> - 8 bit data, no parity (00)
  //Bit 0: STSEL - one stop bit (0)

  U1STA = 0b0000010000000000;
  //Bit 15,13: UTXISEL<1:0> - interrupt when transfer to Tshift (00)
  //Bit 14: UTXINV - transmit idle state is '1' (0)
  //Bit 12: Unimplemented (0)
  //Bit 11: UTXBRK - sync break transmission disabled (0)
  //Bit 10: UTXEN - transmitter disabled (0) (will enable later)
  //Bit 9: UTXBF - status bit for full buffer register (0)
  //Bit 8: TRMT - status bit for full shift register (0)
  //Bit 7-6: URXISEL<1:0> - interrupt flag set when character received (00)
  //Bit 5: ADDEN - address detect mode disabled (0)
  //Bit 4: RIDLE - status bit for receiver idle (0)
  //Bit 3: PERR - status bit for parity error (0)
  //Bit 2: FERR - status bit for framing error (0)
  //Bit 1: OERR - status bit for buffer overflow (0)
  //Bit 0: URXDA - status bit for receive buffer available (0)

  U1RXREG = 0b0000000000000000;
  //Bit 15-9: Unimplemented (0000000)
  //Bit 8: Data bit 8 in 9-bit mode (0)
  //Bit 7-0: Data bits 7-0 (00000000)
  //Not sure if I can set these bits; just helps bookkeep in code

  U1TXREG = 0x0000000000000000;
  //Bit 15-9: Unimplemented (0000000)
  //Bit 8: Data bit 8 in 9-bit mode (0)
  //Bit 7-0: Data bits 7-0 (00000000)

  U1BRG = 0b0000000001100111;
  //Bit 15-0: Baud Rate Generator Divisor bits
  //For 8 MHz internal clock, Fcy = 4 MHz -- see baud rate tables
  //Choose BRG = 0d25 in this case for Baud = 9600
  //0d25 = 0b0000000000011001
  //Actual measured Fcy = 2 MHz: for Baud of 9600 with BRGH=0, want 0d12
  //0d13 = 0000000000001100;
  //Fcy = F/2.  F = 4 MHz if divided by 2 in non-volatile memory.
  //Works at 4 MHz: 0b0000000000001100 corresponds to 9600
  //At 32 MHz (Fcy = 16 MHz) for 9600, want 0d103 => 1100111

  //Enable the U1 module
  U1MODEbits.UARTEN = 1;
  Nop();

  //Enable transmit for the U1 module
  U1STAbits.UTXEN = 1;
  Nop();

  //Turn off
  IFS0bits.U1RXIF = 0;
  IFS0bits.U1TXIF = 0;
  IEC0bits.U1RXIE = 1;
  IEC0bits.U1TXIE = 0;
  IPC2bits.U1RXIP = 0b111;


}

void WriteMenu(void) {

  WriteScreen("\n\n\n\n\r***Modified Boost Menu***");
  WriteScreen("\n\ra:IncVzvsB\tz:DecVzvsB");
  WriteScreen("\n\rs:IncVzvsA\tx:DecVzvsA");

  WriteScreen("\n\rd:WriteVout\tc:WriteVin");
  WriteScreen("\n\rg:IncVcntrlb\tb:DecVcntrlb");
```

```
    WriteScreen("\n\rh:IncVcntrla\tn:DecVcntrla");
    WriteScreen("\n\rq:Startup\t");
    WriteScreen("\n\re:Auto/Manual\ty:TimeDispToggle");

    WriteScreen("\n\n\r");

}

void WriteScreen(char s[50]) {
    char *p;
    p = s;
    while (*p) {
        while (!(U1STAbits.TRMT)) {
        }
        U1TXREG = *p++;
    }
}

void StartupCycle(void) {


    //Ensure good initial condition
    CCP1CON2Hbits.OCBEN = 0; //ManualON A disconnected from PWM
    LATBbits.LATB8=0; //Make sure ManualON A is off


    //EnableA should be digital, output, on
    TRISBbits.TRISB10 = 0;
    LATBbits.LATB10 = 1;


    LATBbits.LATB8 = 1;  //Turn on A
    LATBbits.LATB10= 0;  //Quickly disable A before timer triggers
    LATBbits.LATB8 = 0;  //Remove Manual A

}
```

Appendix T

PFC Schematic


Sheet: Microcontroller
File: Microcontroller.sch
Sheet: ControlAB
File: CAB.sch
Vzvsa1
Vcntrla1
ENABLEa1
ManualONa1
Vzvsb1
Vcntrlb1
ENABLEb1
ManualONb1
Vzvsb2
Vcntrlb2
ENABLEb2
ManualONb2
Vzvsa2
Vcntrla2
ENABLEa2
ManualONa2
GateDriveA1
GateDriveB1
GateDriveB2
GateDriveA2
gateA1
gateB1
gateB2
gateA2
Vdd
Vin
Vout
Hon_pre
Non_pre
Vn
Vh
Va
Vb
U2
U4
5+unregA
5+unregB
VddA+
VddB+
RP-0505S
U1
U3
SI8410
TP1
TP2
TP3
TP4
TP5
TEK-3.5mm-TP
GND
BananaJack
J1
J13
J2
Sheet: EMI_Filter
File: EMI_Filter.sch
EARTH
H
N
H'
N'
Header-1x1-0.1
J6
J7
Sheet: SWA1
Sheet: SA2
Sheet: SWB1
Sheet: SWB2
File: Panasonic56.sch
gateA+
gateB+
J39
J9
Buy the 1210, 0.22 uF capacitors
Not the 1206
Flipped just to get pins to conviently arrange in layout!
450QXW220MEFC18X50 -- good 220 uF at .7" diam
450MXK390MEFCSN22X60 -- good 390 uF at .86" diam
Alex Hanson, David Perreault
Massachusetts Institute of Technology
Sheet: /
File: modifiedboost7.sch
Title: Universal Boost
Size: B
Date:
Rev: 1
KiCad E.D.A. kicad 5.1.2
Id: 1/8



(5 − 3.1) / (50 mA) = 36 ohm
Rig should be > 36 ohm
Choose 51 ohm for safety

1/(1.5A/5V − 1/50ohm) = 3.6 ohm
Rgon should be > 3.6 ohm
Choose 6.2 ohm for safety

Rgoff just prevents false turn−on from ringing
Recommend set to "a few ohms" = 4.7

Vgs negative recovery time constant
should be 1/3 of minimum off time
(maybe 1/3 of 100 ns = 33.3 ns)
Cs = tau/(Rgon+Rig) − Cg
= 33.3ns/(36ohm+6.2ohm) − 400pF
= 389 pF

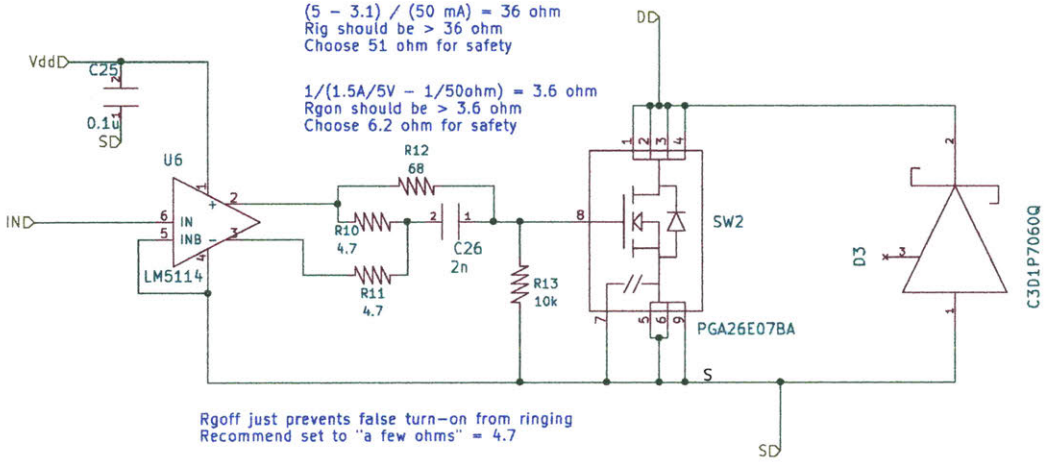
New recommendation from the low
resistance panasonic GaN app note
and own calculations:

6.2 ohm turn−on
100 ohm steady on
1500 pF
4.7 ohm turn−off
10k ohm pull−down

4/17/19
What I actually installed is

1.0 ohm pulldown
0.0 ohm pullup
68 ohm delivery
3.3n capacitor
10k on bottom

Sheet: /SWB1/		
File: Panasonic56.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 5.1.2		Id: 2/8



(5 - 3.1) / (50 mA) = 36 ohm
Rig should be > 36 ohm
Choose 51 ohm for safety

1/(1.5A/5V - 1/50ohm) = 3.6 ohm
Rgon should be > 3.6 ohm
Choose 6.2 ohm for safety

Rgoff just prevents false turn-on from ringing
Recommend set to "a few ohms" = 4.7

Vgs negative recovery time constant
should be 1/3 of minimum off time
(maybe 1/3 of 100 ns = 33.3 ns)
Cs = tau/(Rgon+Rig) - Cg
= 33.3ns/(36ohm+6.2ohm) - 400pF
= 389 pF

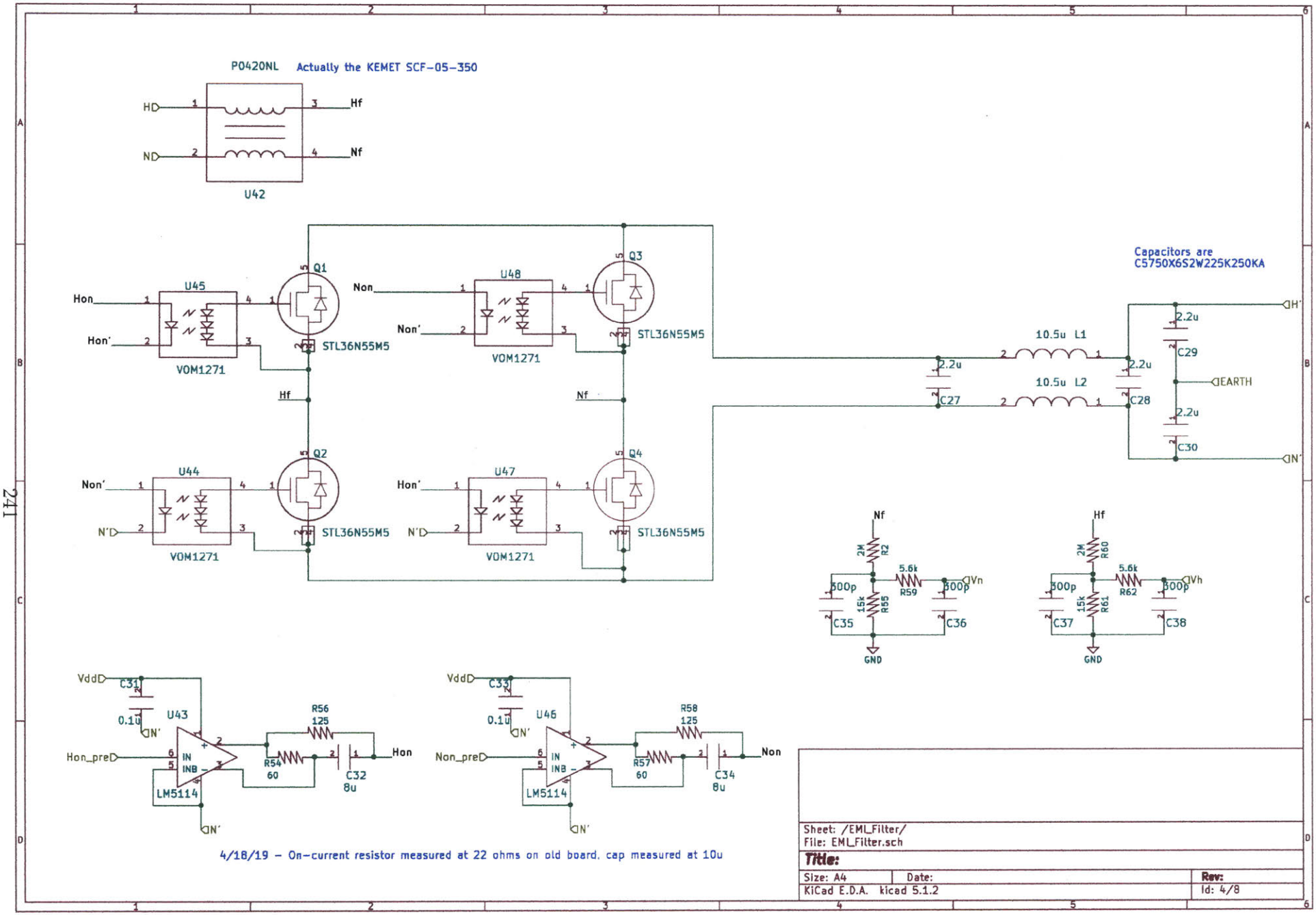
New recommendation from the low
resistance panasonic GaN app note
and own calculations:

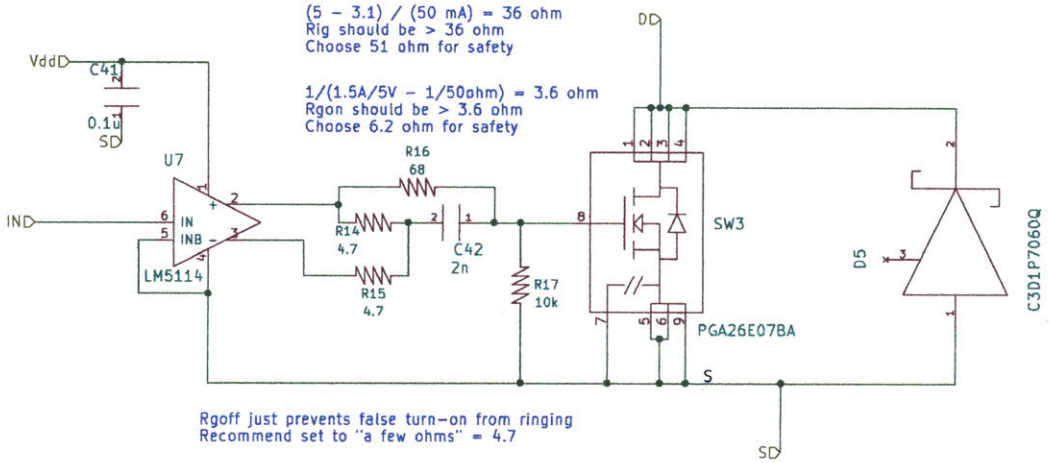
6.2 ohm turn-on
100 ohm steady on
1500 pF
4.7 ohm turn-off
10k ohm pull-down

4/17/19
What I actually installed is

1.0 ohm pulldown
0.0 ohm pullup
68 ohm delivery
3.3n capacitor
10k on bottom

Sheet: /SWA1/		
File: Panasonic56.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 5.1.2		Id: 3/8


P0420NL Actually the KEMET SCF-05-350
HI
NI
Hf
Nf
U42
U45
U44
U48
U47
VOM1271
Hon
Hon'
Non'
N'
Non
Q1
Q2
Q3
Q4
STL36N55M5
Hf
Nf
Capacitors are
C5750X6S2W225K250KA
10.5u L1
10.5u L2
2.2u
C27
C28
C29
C30
EARTH
H'
N'
2M
R2
R60
5.6k
R59
R62
300p
C35
C36
C37
C38
15k
R55
R61
Vn
Vh
GND
Vdd
C31
C33
0.1u
U43
U46
LM5114
IN
INB
R56
R58
125
R54
R57
60
C32
C34
8u
Hon_pre
Non_pre
Hon
Non
4/18/19 - On-current resistor measured at 22 ohms on old board, cap measured at 10u
Sheet: /EMI_Filter/
File: EMI_Filter.sch
Title:
Size: A4
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 4/8



Rgoff just prevents false turn-on from ringing
Recommend set to "a few ohms" = 4.7

Vgs negative recovery time constant
should be 1/3 of minimum off time
(maybe 1/3 of 100 ns = 33.3 ns)
Cs = tau/(Rgon+Rig) - Cg
= 33.3ns/(36ohm+6.2ohm) - 400pF
= 389 pF

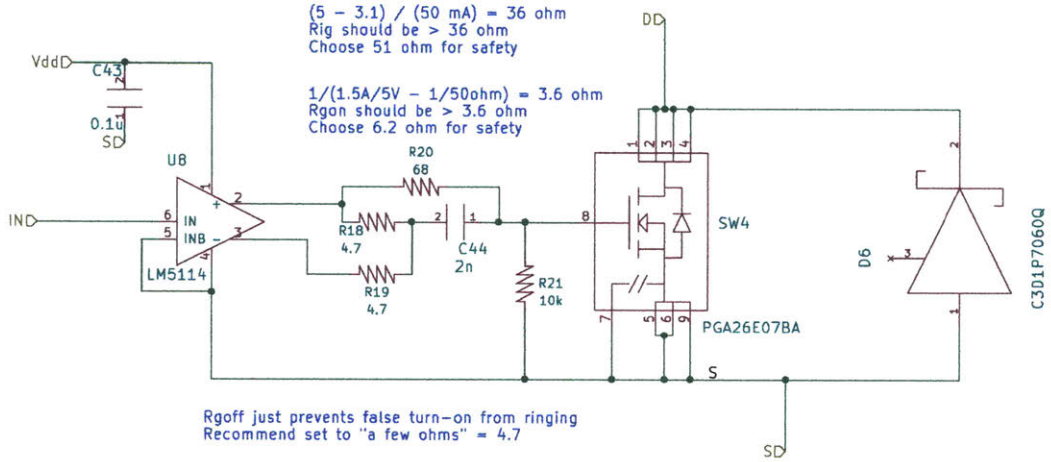
New recommendation from the low
resistance panasonic GaN app note
and own calculations:

6.2 ohm turn-on
100 ohm steady on
1500 pF
4.7 ohm turn-off
10k ohm pull-down

4/17/19
What I actually installed is

1.0 ohm pulldown
0.0 ohm pullup
68 ohm delivery
3.3n capacitor
10k on bottom

Sheet: /SA2/		
File: Panasonic56.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 5.1.2		Id: 5/8



Rgoff just prevents false turn-on from ringing
Recommend set to "a few ohms" = 4.7

Vgs negative recovery time constant
should be 1/3 of minimum off time
(maybe 1/3 of 100 ns = 33.3 ns)
Cs = tau/(Rgon+Rig) - Cg
= 33.3ns/(36ohm+6.2ohm) - 400pF
= 389 pF

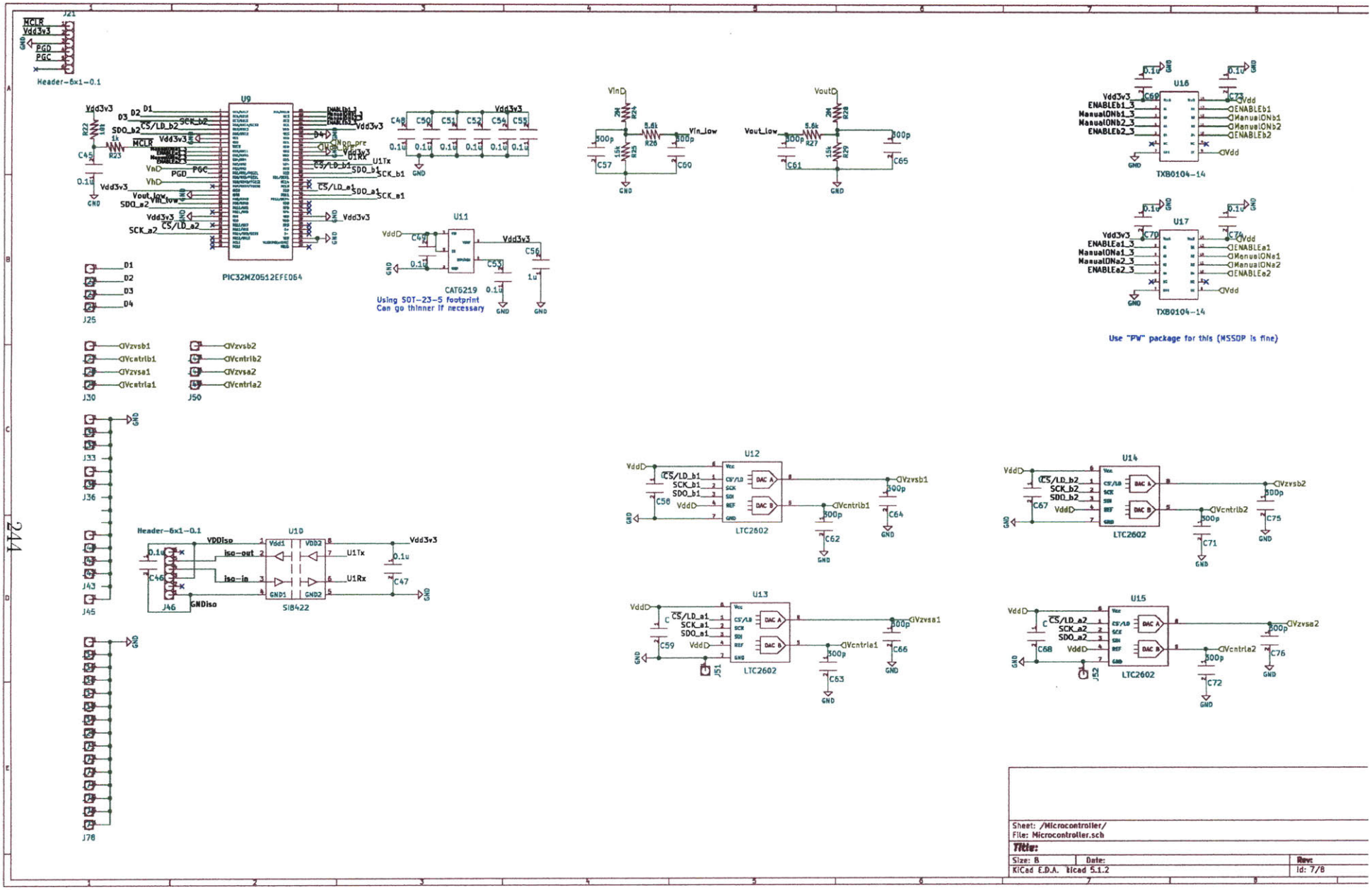
New recommendation from the low
resistance panasonic GaN app note
and own calculations:

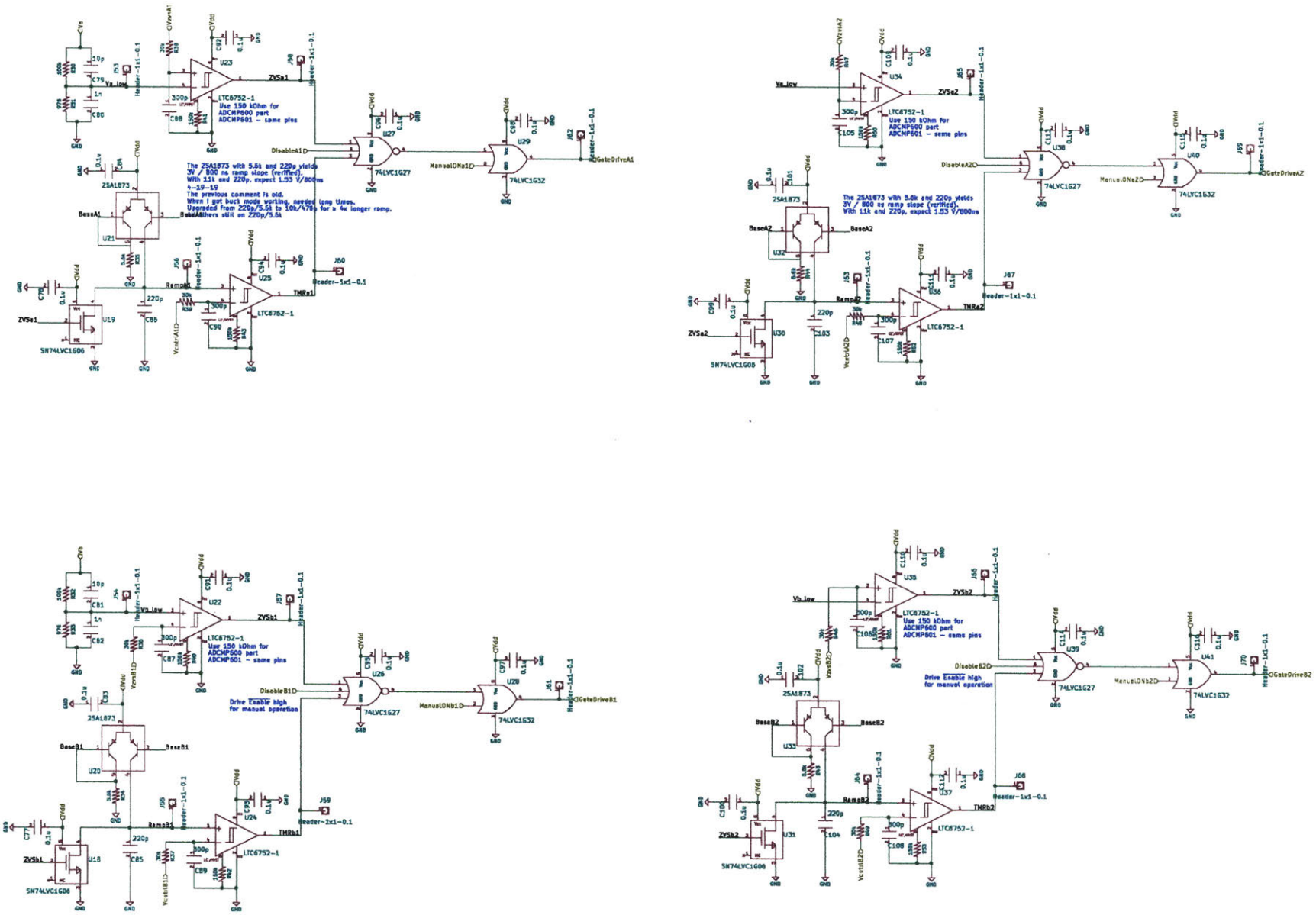
6.2 ohm turn-on
100 ohm steady on
1500 pF
4.7 ohm turn-off
10k ohm pull-down

4/17/19
What I actually installed is

1.0 ohm pulldown
0.0 ohm pullup
68 ohm delivery
3.3n capacitor
10k on bottom

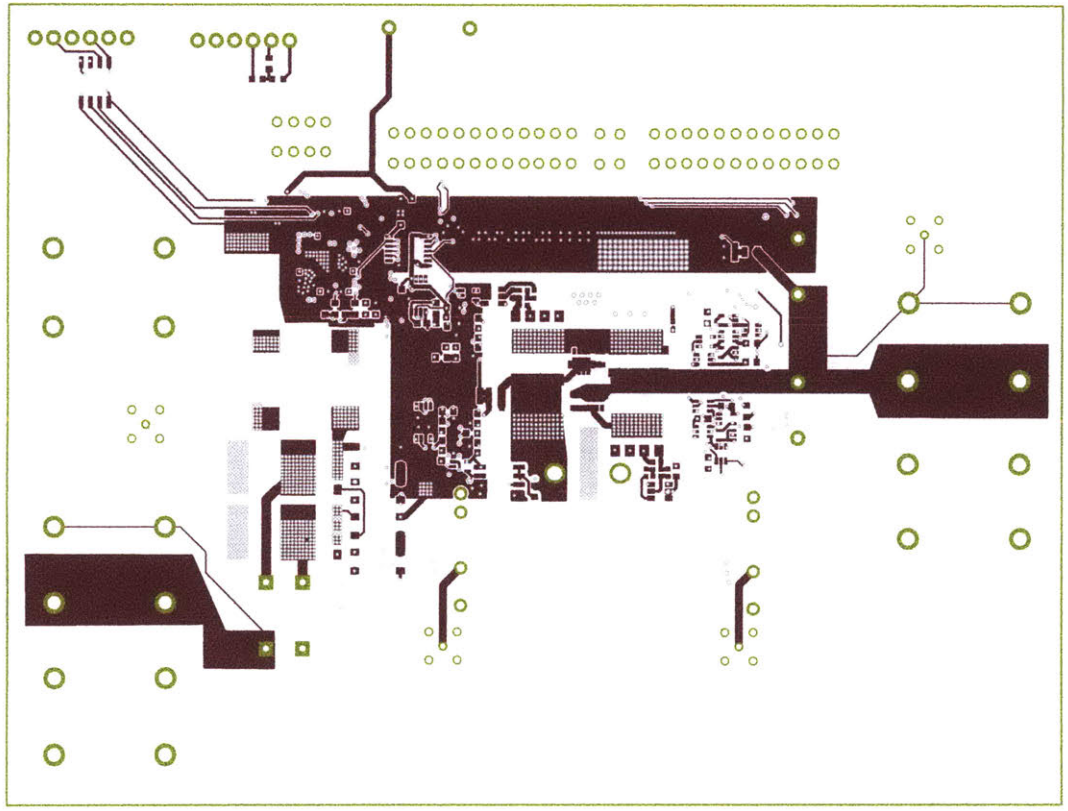
Sheet: /SWB2/		
File: Panasonic56.sch		
Title:		
Size: A4	Date:	**Rev:**
KiCad E.D.A. kicad 5.1.2		Id: 6/8


J21
MCLR
Vdd3v3
PGD
PGC
Header-6x1-0.1
U9
PIC32MZ0512EFE064
U11
CAT6219
Using SOT-23-5 footprint
Can go thinner if necessary
J25
J30
J50
J33
J36
J43
J45
J46
J78
Header-6x1-0.1
U10
SI8422
VDDiso
GNDiso
iso-out
iso-in
U1Tx
U1Rx
Vin
Vout
Vin_low
Vout_low
U16
U17
TXB0104-14
Use "PW" package for this (MSSOP is fine)
U12
U13
U14
U15
LTC2602
DAC A
DAC B
Vzvsb1
Vcntrib1
Vzvsa1
Vcntria1
Vzvsb2
Vcntrib2
Vzvsa2
Vcntria2
J51
J52
Sheet: /Microcontroller/
File: Microcontroller.sch
Title:
Size: B
Date:
Rev:
KiCad E.D.A. kicad 5.1.2
Id: 7/8

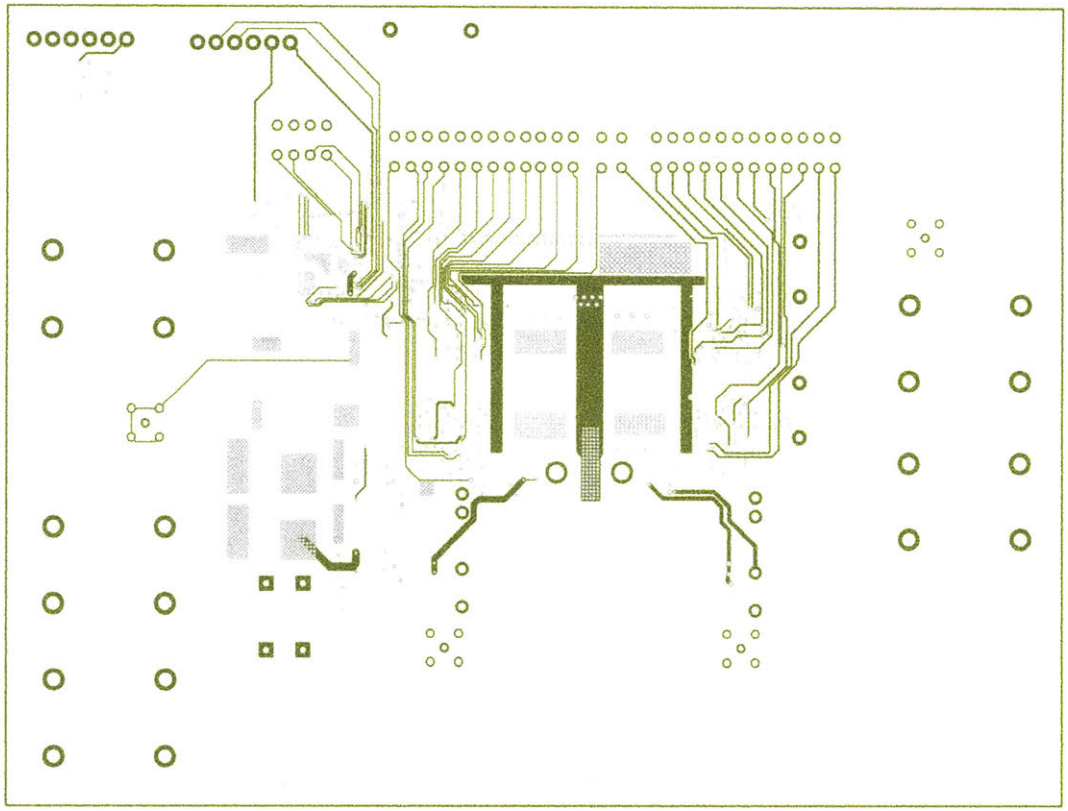

LTC6752-1
Use 150 kOhm for ADCMP600 part ADCMP601 - same pins
The 25A1873 with 5.6k and 220p yields 3V / 800 ns ramp slope (verified).
With 11k and 220p, expect 1.53 V/800ns
4-19-19
The previous comment is old.
When I got buck mode working, needed long times.
Upgraded from 220p/5.6k to 10k/470p for a 4x longer ramp.
Others still on 220p/5.6k
The 25A1873 with 5.6k and 220p yields 3V / 800 ns ramp slope (verified).
With 11k and 220p, expect 1.53 V/800ns
Drive Enable high for manual operation
Drive Enable high for manual operation
74LVC1G27
74LVC1G32
SN74LVC1G06
25A1873
ZVSa1
ZVSa2
ZVSb1
ZVSb2
DisableA1D
DisableA2D
DisableB1D
DisableB2D
ManualONa1D
ManualONa2D
ManualONb1D
ManualONb2D
GateDriveA1
GateDriveA2
GateDriveB1
GateDriveB2
TMRa1
TMRa2
TMRb1
TMRb2
RampA1
RampA2
RampB1
RampB2
BaseA1
BaseA2
BaseB1
BaseB2
Va_low
Vb_low
Header-1x1-0.1
Sheet: /ControlAB/
File: CAB.sch
Title:
Size: C
Date:
KiCad E.D.A. kicad 5.1.2
Rev:
Id: 8/8

Appendix U

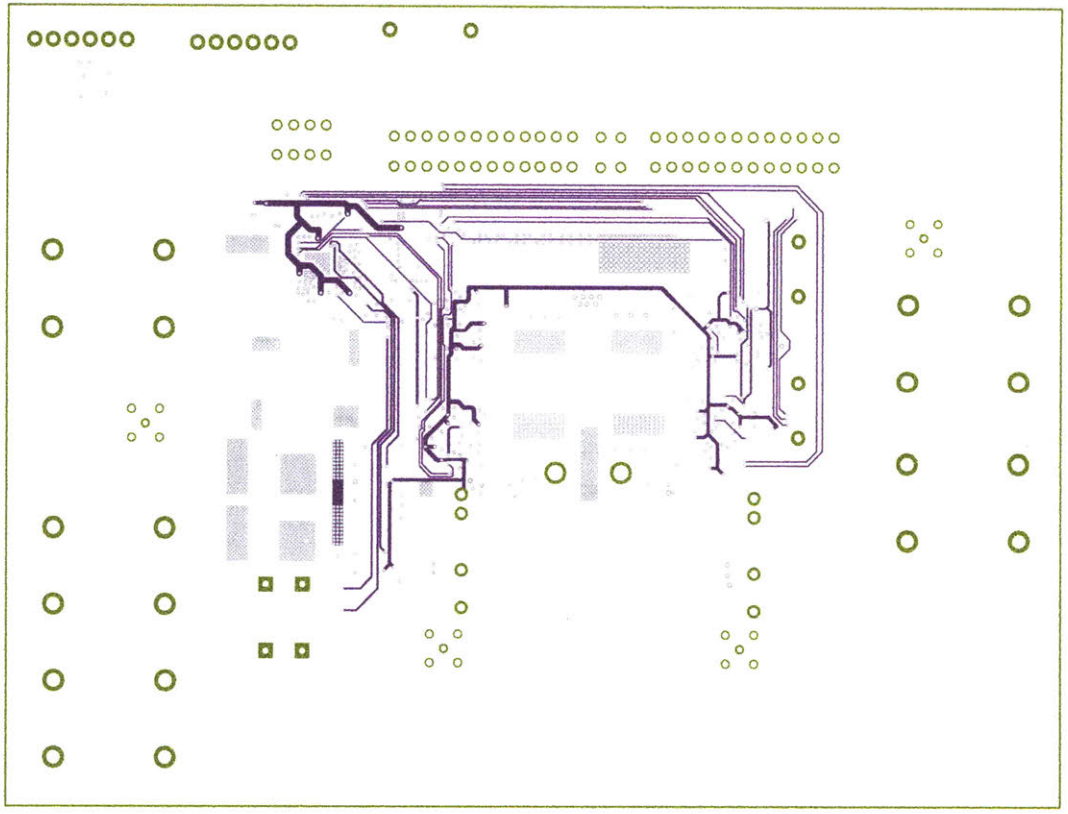
PFC Layout



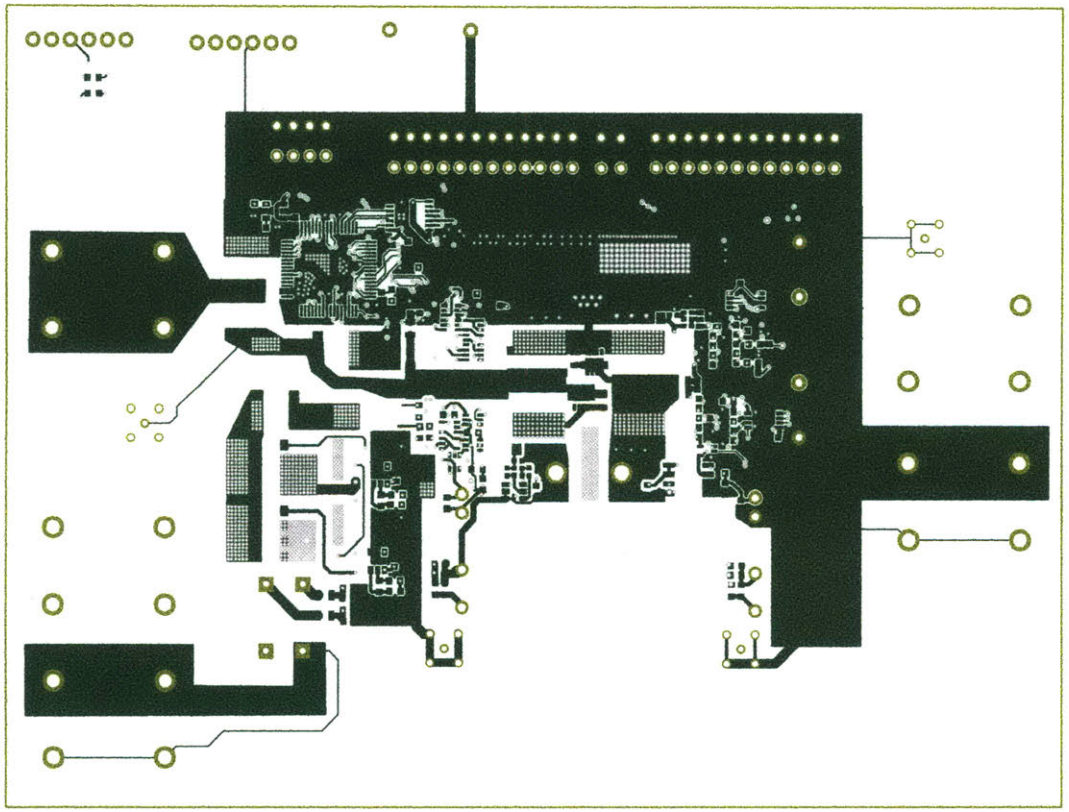
Sheet:		
File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



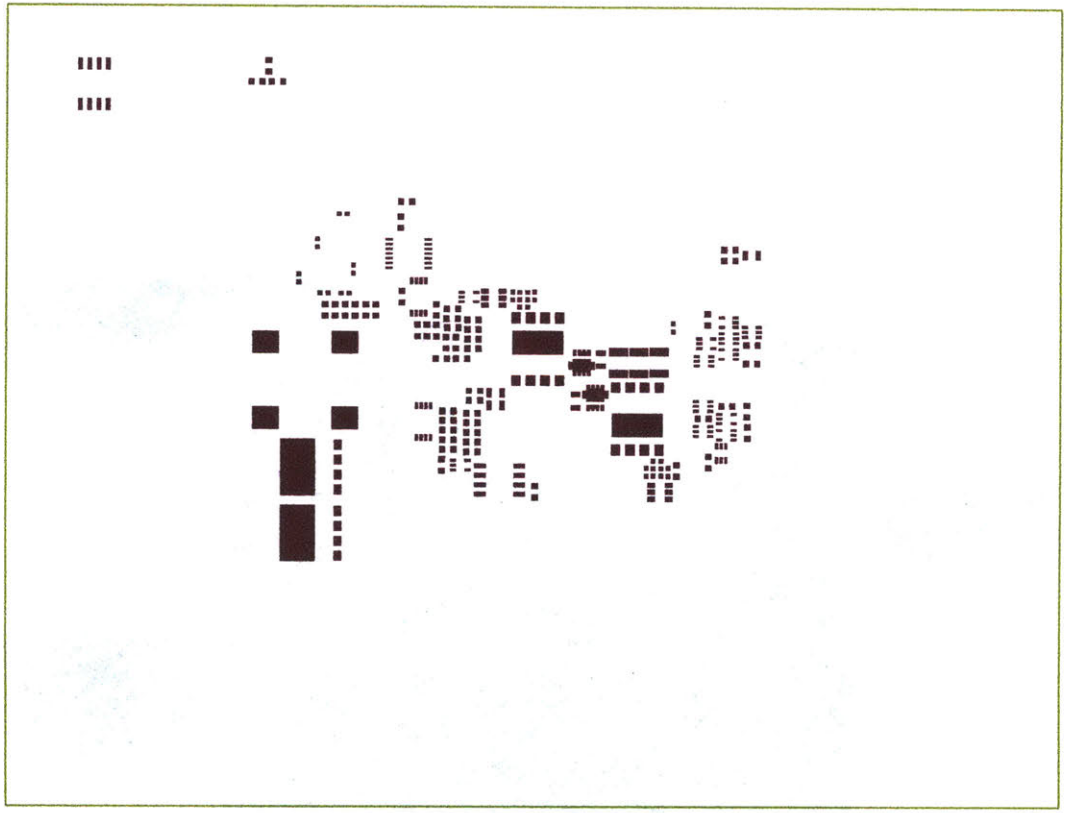
Sheet:		
File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



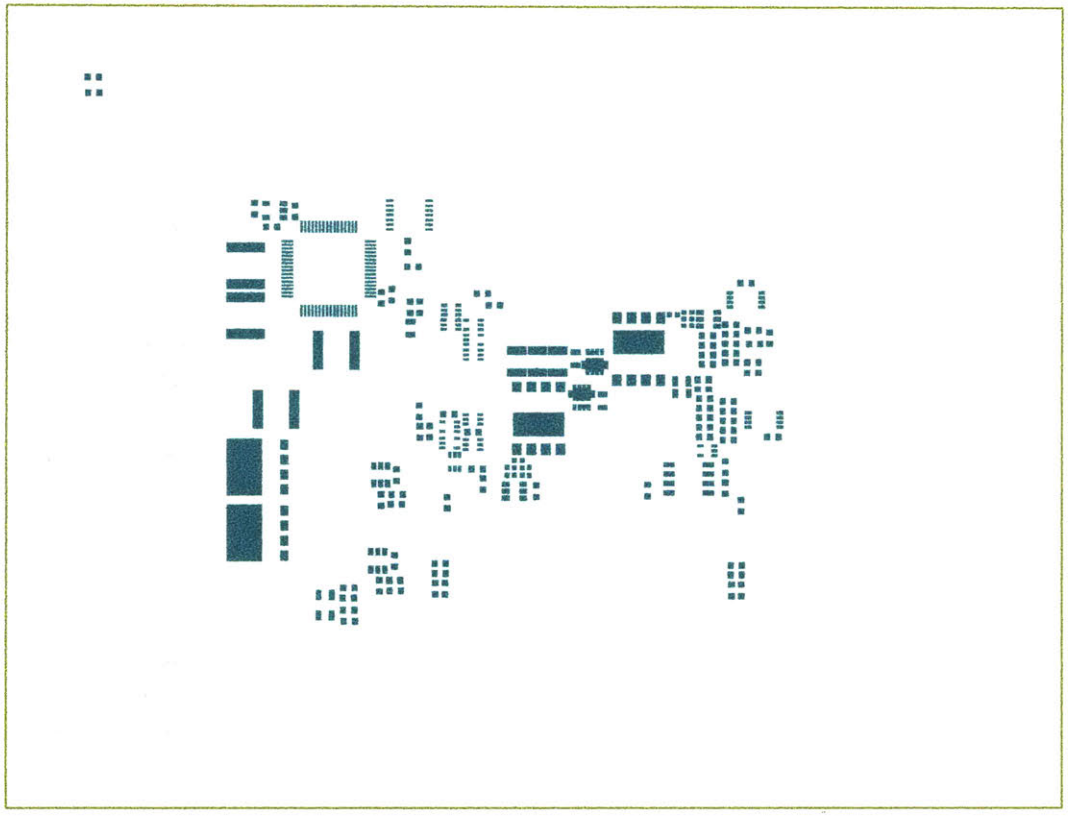
Sheet: File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



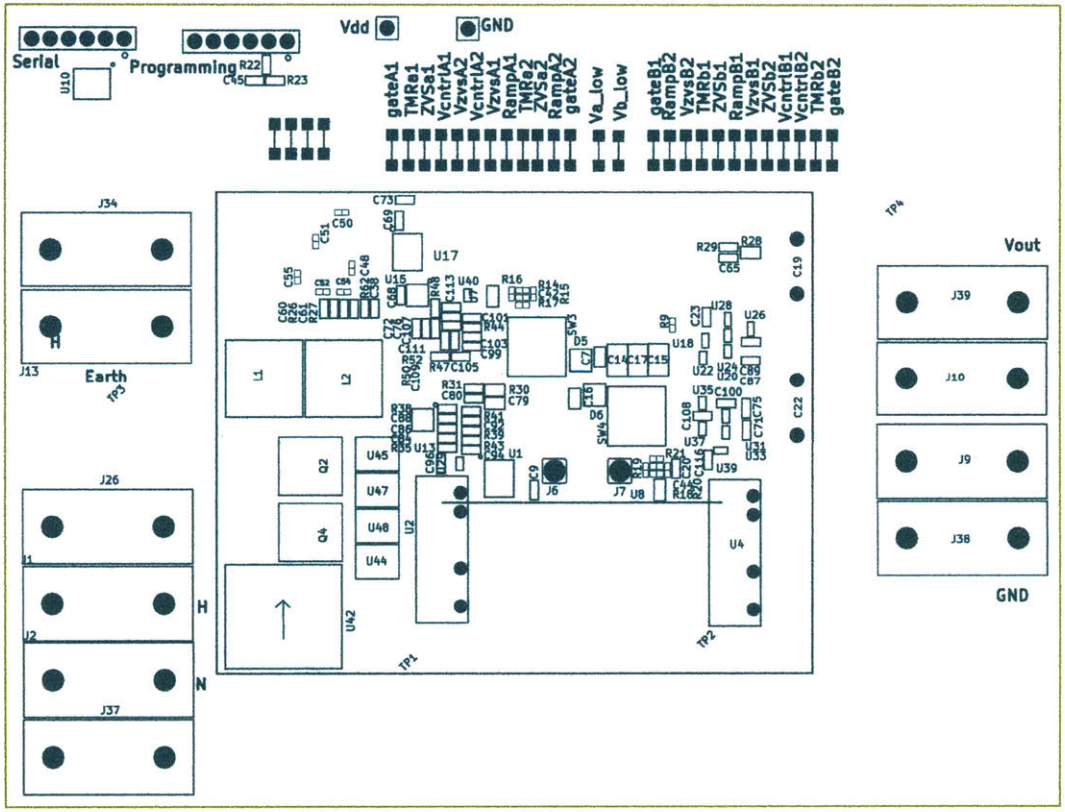
Sheet:		
File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet: File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



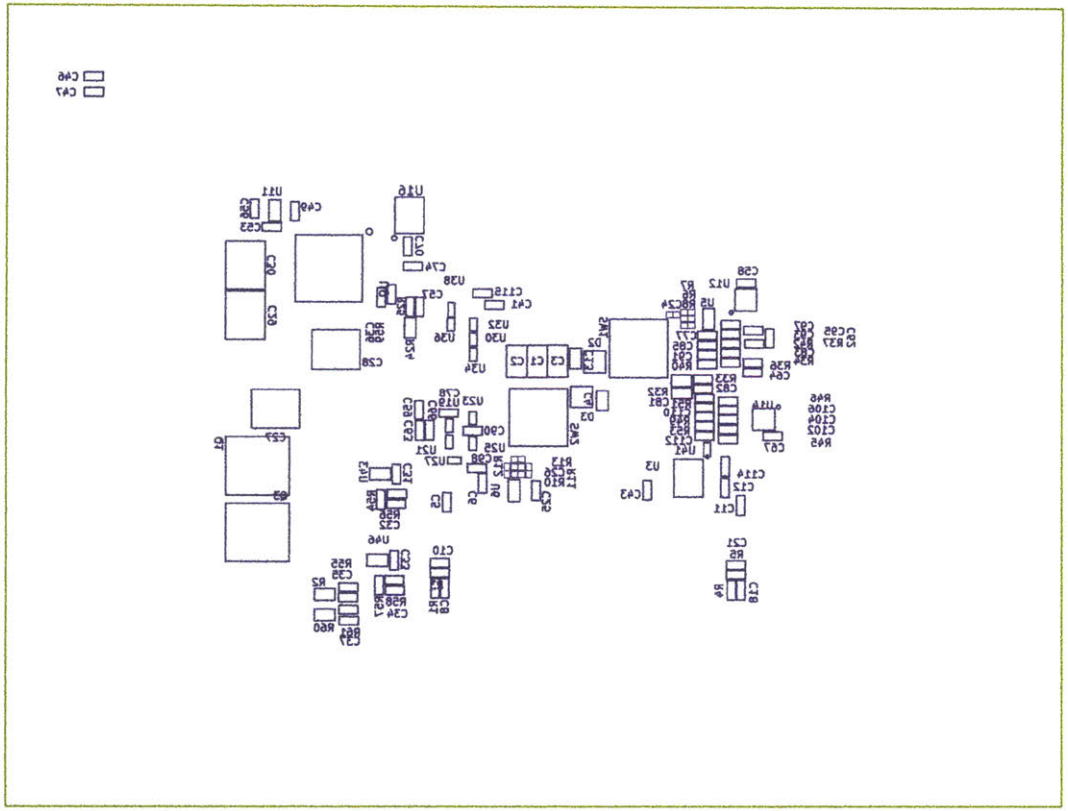
Sheet:		
File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet:
File: modifiedboost7.kicad_pcb

Title:

Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1



Sheet:		
File: modifiedboost7.kicad_pcb		
Title:		
Size: USLetter	Date:	Rev:
KiCad E.D.A. kicad 5.1.2		Id: 1/1

Appendix V

PFC Code

```
/*
 * File:   modboost32_refactored.c
 * Author: Alex Hanson
 *
 * Created on May 2, 2017, 11:36 AM
 */



// <editor-fold defaultstate="collapsed" desc="Configuration Bits">

// DEVCFG3
// USERID = No Setting


#pragma config FMIIEN = OFF          // Ethernet RMII/MII Enable (RMII Enabled)
#pragma config FETHIO = ON           // Ethernet I/O Pin Select (Default Ethernet I/O)
#pragma config PGL1WAY = OFF         // Permission Group Lock One Way Configuration (Allow multiple reconfigurations)
#pragma config PMDL1WAY = OFF        // Peripheral Module Disable Configuration (Allow multiple reconfigurations)
#pragma config IOL1WAY = OFF         // Peripheral Pin Select Configuration (Allow multiple reconfigurations)
#pragma config FUSBIDIO = OFF        // USB USBID Selection (Controlled by Port Function)

// DEVCFG2
#pragma config FPLLICLK = PLL_FRC     // System PLL Input Clock Selection (FRC is input to the System PLL)
#pragma config FPLLIDIV = DIV_1      // System PLL Input Divider (1x Divider)
#pragma config FPLLRNG = RANGE_5_10_MHZ // System PLL Input Range (5-10 MHz Input)
#pragma config FPLLMULT = MUL_64      // System PLL Multiplier (PLL Multiply by 4)
#pragma config FPLLODIV = DIV_8      // System PLL Output Clock Divider (2x Divider)
#pragma config UPLLFSEL = FREQ_24MHZ  // USB PLL Input Frequency Selection (USB PLL input is 24 MHz)

// DEVCFG1
#pragma config FNOSC = SPLL          // Oscillator Selection Bits (Fast RC Osc w/Div-by-N (FRCDIV))
#pragma config DMTINTV = WIN_127_128   // DMT Count Window Interval (Window/Interval value is 127/128 counter value)
#pragma config FSOSCEN = OFF         // Secondary Oscillator Enable (Disable SOSC)
#pragma config IESO = OFF            // Internal/External Switch Over (Disabled)
#pragma config POSCMOD = OFF         // Primary Oscillator Configuration (Primary osc disabled)
#pragma config OSCIOFNC = OFF        // CLKO Output Signal Active on the OSCO Pin (Disabled)
#pragma config FCKSM = CSDCMD        // Clock Switching and Monitor Selection (Clock Switch Disabled, FSCM Disabled)
#pragma config WDTPS = PS1048576     // Watchdog Timer Postscaler (1:1048576)
#pragma config WDTSPGM = STOP        // Watchdog Timer Stop During Flash Programming (WDT stops during Flash programming)
#pragma config WINDIS = NORMAL       // Watchdog Timer Window Mode (Watchdog Timer is in non-Window mode)
#pragma config FWDTEN = OFF          // Watchdog Timer Enable (WDT Disabled)
#pragma config FWDTWINSZ = WINSZ_25   // Watchdog Timer Window Size (Window size is 25%)
#pragma config DMTCNT = DMT31        // Deadman Timer Count Selection (2^31 (2147483648))
#pragma config FDMTEN = OFF          // Deadman Timer Enable (Deadman Timer is disabled)

// DEVCFG0
#pragma config DEBUG = OFF           // Background Debugger Enable (Debugger is disabled)
#pragma config JTAGEN = OFF          // JTAG Enable (JTAG Disabled)
#pragma config ICESEL = ICS_PGx1     // ICE/ICD Comm Channel Select (Communicate on PGEC1/PGED1)
#pragma config TRCEN = ON            // Trace Enable (Trace features in the CPU are disabled)
#pragma config BOOTISA = MIPS32      // Boot ISA Selection (Boot code and Exception code is MIPS32)
#pragma config PECCCON = OFF_UNLOCKED  // Dynamic Flash ECC Configuration (ECC and Dynamic ECC are disabled (ECCCON bits are writable))
#pragma config FSLEEP = OFF          // Flash Sleep Mode (Flash is powered down when the device is in Sleep mode)
#pragma config DBGPER = PG_ALL       // Debug Mode CPU Access Permission (Allow CPU access to all permission regions)
#pragma config SMCLR = MCLR_NORM     // Soft Master Clear Enable bit (MCLR pin generates a normal system Reset)
#pragma config SOSCGAIN = GAIN_2X    // Secondary Oscillator Gain Control bits (2x gain setting)
#pragma config SOSCBOOST = ON        // Secondary Oscillator Boost Kick Start Enable bit (Boost the kick start of the oscillator)
#pragma config POSCGAIN = GAIN_2X    // Primary Oscillator Gain Control bits (2x gain setting)
#pragma config POSCBOOST = ON        // Primary Oscillator Boost Kick Start Enable bit (Boost the kick start of the oscillator)
#pragma config EJTAGBEN = NORMAL     // EJTAG Boot (Normal EJTAG functionality)

// DEVCP0
#pragma config CP = OFF              // Code Protect (Protection Disabled)


// </editor-fold>

//Windows
#include "../proc/p32mzc0512efe064.h"
#include "xc.h"
#include <sys/attribs.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

///Linux
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/proc/p32mzc0512efe064.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/xc.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/stdio.h"
//#include "/opt/microchip/xc32/v1.43/pic32mx/include/math.h"
//#include <sys/attribs.h>


// <editor-fold defaultstate="collapsed" desc="Pin #defines">

#define DIO1_set LATESET = 1<<5
#define DIO2_set LATESET = 1<<6
#define DIO3_set LATESET = 1<<7
#define DIO4_set LATESET = 1<<0

#define DIO1_clr LATECLR = 1<<5
#define DIO2_clr LATECLR = 1<<6
#define DIO3_clr LATECLR = 1<<7
#define DIO4_clr LATECLR = 1<<0


//There is no more ZVS_select on the white buck board
//#define ZVS_select_set  LATFSET= 1<<5

#define CSa1_set       LATDSET= 1<<0
#define CSa2_set       LATBSET= 1<<13
#define CSb1_set       LATDSET= 1<<3
#define CSb2_set       LATGSET= 1<<7
#define ENABLEa1_set    LATBSET= 1<<5
#define ENABLEa2_set    LATBSET= 1<<3
#define ENABLEb1_set    LATESET= 1<<4
#define ENABLEb2_set    LATESET= 1<<1
#define MANUALa1_set    LATGSET= 1<<9
#define MANUALa2_set    LATBSET= 1<<4
#define MANUALb1_set    LATESET= 1<<3
#define MANUALb2_set    LATESET= 1<<2
#define NONPRE_set      LATFSET= 1<<1
#define HONPRE_set      LATFSET= 1<<0

#define CSa1_clr       LATDCLR= 1<<0
#define CSa2_clr       LATBCLR= 1<<13
#define CSb1_clr       LATDCLR= 1<<3
#define CSb2_clr       LATGCLR= 1<<7
#define ENABLEa1_clr    LATBCLR= 1<<5
#define ENABLEa2_clr    LATBCLR= 1<<3
#define ENABLEb1_clr    LATECLR= 1<<4
#define ENABLEb2_clr    LATECLR= 1<<1
#define MANUALa1_clr    LATGCLR= 1<<9
#define MANUALa2_clr    LATBCLR= 1<<4
#define MANUALb1_clr    LATECLR= 1<<3
#define MANUALb2_clr    LATECLR= 1<<2
#define NONPRE_clr      LATFCLR= 1<<1
#define HONPRE_clr      LATFCLR= 1<<0


//Define ADCdata registers
//Will take care of Uart and SPI in functions rather than #defines
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Function Prototypes">
double Compute11(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double I2, double limitpercent, double I3, double I5);
double Compute11buck(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double I2, double limitpercent, double I3, double I5);
double Computetbuck(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double I2, double limitpercent, double I3, double I5);
double Computetbtran(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double I2, double limitpercent, double I3, double I5, double ta);
void SetupSPIa1(void);
void SetupSPIa2(void);
void SetupSPIb1(void);
void SetupSPIb2(void);
void SetupADC(void);
void SetupUART(void);
void SetupTimer(void);
void StartupBoost(void);
void StartupBuck(void);
void StartupCycle(void);
void StartupLong(void);
void WriteMenu(void);
void WriteScreen(char s[50]);
void WriteSPIa1(unsigned int* dacinput, char select);
void WriteSPIa2(unsigned int* dacinput, char select);
void WriteSPIb1(unsigned int* dacinput, char select);
void WriteSPIb2(unsigned int* dacinput, char select);
void WriteVin(void);
void WriteVout(void);
void WriteVh(void);
void WriteVn(void);
// </editor-fold>


// <editor-fold defaultstate="collapsed" desc="Static Volatile Initializations/Declarations">
```

```
typedef struct {
    unsigned Timer :1;
    unsigned ADC1 :1;
    unsigned U1RX :1;
    unsigned Rising :1;
    unsigned WriteTimes :1;
    unsigned Running :1;
    unsigned Mode :1;
    unsigned Control :1;
}FlagStruct;
//Mode and Running form a state machine with states Running,Mode
//State 0,0 = Everything is off, converter hasn't started
//State 0,1 = SA1 is on for initial charging of output capacitor
//State 1, 1 = Operation in mod boost mode
//State 1,0 = Operation in regular boost mode
//See code for state transitions
static volatile FlagStruct Flags;

static volatile double adccount;
static volatile double vinaccum;
static volatile double voutaccum;
static volatile int protect = 0;
static volatile char selection;
// </editor-fold>


int main(int argc, char** argv) {

// <editor-fold defaultstate="collapsed" desc="Global Interrupt Enable">
  //Enable uC to receive interrupts
  // macros come from this page:  http://microchipdeveloper.com/32bit:mz-arch-exceptions-usage
  INTCONSET = _INTCON_MVEC_MASK; //Set interrupt controller to Multi Vector Mode through macro
  __builtin_enable_interrupts(); // Globally enable interrupts through CP0 StatusIE bit
// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Oscillator Setup">


  REFO1CONbits.OE = 0;
  OSCCONbits.FRCDIV = 000;  //Same as default, divide by 1

  //REFCLK1 can be used for SPI (or PBCLK2) or as an output
  //REFCLK2 can be used for output or SQI (unused)
  //REFCLK3 can be used for ADC (or FRC or SYSCLK) or as an output

  REFO1CONbits.RODIV = 0;
  REFO1CONbits.RSLP = 1; //Run during sleep
  if(REFO1CONbits.ACTIVE == REFO1CONbits.ON){ //Shouldn't write if active != ON
    REFO1CONbits.ROSEL = 0b0000;  //Use SYSCLK output as source for reference
    REFO1CONbits.RODIV = 0;  //No divider for reference clock
    REFO1TRIM = 0;
    REFO1CONbits.ON = 0;   //Turn on to activate
  }
  REFO1CONbits.OE = 0; //Turn on to drive the REFCLKO1 pin (defined by PPS)
  //Note that SPLLCON should be completely defined by config registers on reset




  // <editor-fold defaultstate="collapsed" desc="Unlock Sequence for Modifying Oscillator">
  volatile unsigned int int_status;
  volatile unsigned int dma_suspend;

  // Disable global interrupt
  int_status = __builtin_get_isr_state();
  __builtin_disable_interrupts();

  // Suspend DMA
  dma_suspend = DMACONbits.SUSPEND;
  if (dma_suspend == 0)
  {
    DMACONSET = _DMACON_SUSPEND_MASK;
    while (DMACONbits.DMABUSY == 1);
  }

  /* Unlock */
  SYSKEY = 0x00000000;
  SYSKEY = 0xAA996655;
  SYSKEY = 0x556699AA;
  // </editor-fold>


  PB3DIVbits.PBDIV = 0;  //PBCLK3 = SYSCLK  (no scaling)  runs TMR1 which triggers ADC
  PB3DIVbits.ON = 1;  //Activate PBCLK3

  PB2DIVbits.PBDIV = 0;  //Divide by 2; UART
  PB2DIVbits.ON = 1;

  PB1DIVbits.PBDIV = 0;
  //PB1DIVbits.ON = 1;


  // <editor-fold defaultstate="collapsed" desc="Re-Lock Sequence After Modifying Oscillator">
  SYSKEY = 0x33333333;

  if (dma_suspend == 0)
  {
    DMACONCLR = _DMACON_SUSPEND_MASK;
  }

  __builtin_set_isr_state(int_status);
  // </editor-fold>


// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Analog Pin Assignment">
  //Note that pins with analog option default to analog input unless changed manually
  //Analog pins only appear on B, E, G ports
  ANSELB = 0;
  ANSELE = 0;
  ANSELG = 0;
  ANSELBbits.ANSB6 = 1;   //Vh
  ANSELBbits.ANSB2 = 1;   //Vn
  ANSELBbits.ANSB8 = 1;   //Vout_low
  ANSELBbits.ANSB9 = 1;   //Vin_low


// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Pin State Initialization">
  TRISGbits.TRISG6 = 0; LATGbits.LATG6 = 0;   //Pin 4 output low (SCKb2)
  TRISGbits.TRISG7 = 0; LATGbits.LATG7 = 0;   //Pin 5 output low (CS/LDb2)
  TRISGbits.TRISG8 = 0; LATGbits.LATG8 = 0;   //Pin 6 output low (SDOb2)
                        //Pin 7 VSS
                        //Pin 8 VDD
                        //Pin 9 MCLR
  TRISGbits.TRISG9 = 0; LATGbits.LATG9 = 0;   //Pin 10 output low (Manual_a1)
  TRISBbits.TRISB5 = 0; LATBbits.LATB5 = 0;   //Pin 11 output low (Enable_a1)
  TRISBbits.TRISB4 = 0; LATBbits.LATB4 = 0;   //Pin 12 output low (Manual_a2)
  TRISBbits.TRISB3 = 0; LATBbits.LATB3 = 0;   //Pin 13 output low (Enable_a2)
  TRISBbits.TRISB2 = 1;                       //Pin 14 Input (Vh new pin)
                        //Pin 15 PGEC1
                        //Pin 16 PGED1
  TRISBbits.TRISB6 = 1;                       //Pin 17 input (Vh new pin)
  TRISBbits.TRISB7 = 0; LATBbits.LATB7 = 0;   //Pin 18 output low (unused)
                        //Pin 19 AVDD
                        //Pin 20 AVSS
  TRISBbits.TRISB8 = 1;                       //Pin 21 Input (Vout_low new pin)
  TRISBbits.TRISB9 = 1;                       //Pin 22 Input (Vin_low)
  TRISBbits.TRISB10 = 0; LATBbits.LATB10 = 0; //Pin 23 output low (SDOa2)
  TRISBbits.TRISB11 = 0; LATBbits.LATB11 = 0; //Pin 24 output low (unused)
                        //Pin 25 VSS
                        //Pin 26 VDD
  TRISBbits.TRISB12 = 0; LATBbits.LATB12 = 0; //Pin 27 output low (unused)
  TRISBbits.TRISB13 = 0; LATBbits.LATB13 = 0; //Pin 28 output low (CS/LDa2)
  TRISBbits.TRISB14 = 0; LATBbits.LATB14 = 0; //Pin 29 output low (SCKa2)
  TRISBbits.TRISB15 = 0; LATBbits.LATB15 = 0; //Pin 30 output low (unused)
  TRISCbits.TRISC12 = 0; LATCbits.LATC12 = 0; //Pin 31 output low (unused)
  TRISCbits.TRISC15 = 0; LATCbits.LATC15 = 0; //Pin 32 output low (unused)
                        //Pin 33 VBUS (unused)
                        //Pin 34 VUSB3V3 (grounded)
                        //Pin 35 VSS
                        //Pin 36 D- (unused)
                        //Pin 37 D+ (unused)
  TRISFbits.TRISF3 = 0; LATFbits.LATF3 = 0;   //Pin 38 output low (unused)
                        //Pin 39 VDD
                        //Pin 40 VSS
  TRISFbits.TRISF4 = 0; LATFbits.LATF4 = 0;   //Pin 41 output low (unused)
  TRISFbits.TRISF5 = 0; LATFbits.LATF5 = 0;   //Pin 42 output low (unused)
  TRISDbits.TRISD9 = 0; LATDbits.LATD9 = 0;   //Pin 43 output low (unused)
  TRISDbits.TRISD10 = 0; LATDbits.LATD10 = 0; //Pin 44 output low (SCKa1)
```

```
TRISDbits.TRISD11 = 0; LATDbits.LATD11 = 0; //Pin 45 output low (SDOa1)
TRISDbits.TRISD0 = 0; LATDbits.LATD0 = 0;  //Pin 46 output low (CS/LDa1)
TRISCbits.TRISC13 = 0; LATCbits.LATC13 = 0; //Pin 47 output low (unused)
TRISCbits.TRISC14 = 0; LATCbits.LATC14 = 0; //Pin 48 output low (unused)
TRISDbits.TRISD1 = 0; LATDbits.LATD1 = 0;  //Pin 49 output low (SCKb1)
TRISDbits.TRISD2 = 0; LATDbits.LATD2 = 0;  //Pin 50 output low (SDOb1)
TRISDbits.TRISD3 = 0; LATDbits.LATD3 = 0;  //Pin 51 output low (CS/LDb1)
TRISDbits.TRISD4 = 0; LATDbits.LATD4 = 0;  //Pin 52 output low (U4TX)
TRISDbits.TRISD5 = 1;                      //Pin 53 input (U4RX)
                                           //Pin 54 VDD
                                           //Pin 55 VSS
TRISFbits.TRISF0 = 0; LATFbits.LATF0 = 0;  //Pin 56 output low (Hon_pre)
TRISFbits.TRISF1 = 0; LATFbits.LATF1 = 0;  //Pin 57 output low (Non_pre)
TRISEbits.TRISE0 = 0; LATEbits.LATE0 = 0;  //Pin 58 output low (unused)
                                           //Pin 59 VSS
                                           //Pin 60 VDD
TRISEbits.TRISE1 = 0; LATEbits.LATE1 = 0;  //Pin 61 output low (Enable_b2)
TRISEbits.TRISE2 = 0; LATEbits.LATE2 = 0;  //Pin 62 output low (Manual_b2)
TRISEbits.TRISE3 = 0; LATEbits.LATE3 = 0;  //Pin 63 output low (Manual_b1)
TRISEbits.TRISE4 = 0; LATEbits.LATE4 = 0;  //Pin 64 output low (Enable_b1)

TRISEbits.TRISE5 = 0; LATEbits.LATE5 = 0;  //Pin  1 output low (DIO1)
TRISEbits.TRISE6 = 0; LATEbits.LATE6 = 0;  //Pin  2 output low (DIO2)
TRISEbits.TRISE7 = 0; LATEbits.LATE7 = 0;  //Pin  3 output low (DIO3)
TRISEbits.TRISE0 = 0; LATEbits.LATE0 = 0;  //Pin 58 output low (DIO4)

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Peripheral Pin Select Assignments">
U4RXR = 0b0110;    //Set U4RX to be pin RPD5

RPD4R = 0b0010;    //Set RPD4 to be U4TX

RPD2R = 0b0101;    //Set RPD2 to be SDO1
RPG8R = 0b0110;    //Set RGP8 to be SDO2
RPB10R = 0b0111;   //Set RPB10 to be SDO3
RPD11R = 0b1000;   //Set RPD11 to be SDO4
//RPE5R = 0b1111;    //Set RPE5 (pin 1) to REFCLKO1
                   //REFCLKO1 can also go to RPG7 (pin 5)
                   //REFCLKO3 can go to RPG6 (pin 4)

// </editor-fold>

// <editor-fold defaultstate="collapsed" desc="Variable Definition and Initialization">
Flags.Timer = 0;
Flags.ADC1 = 0;
Flags.U1RX = 0;
Flags.WriteTimes = 0;
Flags.Running = 0;
Flags.Mode = 0; //
Flags.Rising = 1;
Flags.Control = 0;

ENABLEb1_set;
ENABLEa1_set;
ENABLEb2_set;
ENABLEa2_set;
MANUALb1_clr;
MANUALa1_clr;
MANUALb2_clr;
MANUALa2_clr;

//Note that I can start up with R = 1100 kicking in at 350 V
//Cb starts at 8000*65536; Ca starts at 4850*65536
//Expect slightly higher than 400 at first; very close after engaging control
//Or same Cb, Ca and R = 850 for a 240 V in put


//(variable)_16 corresponds to 16 bit codes to send to 5V DAC
//They are converter from "normal" variables by (variable in volts) * 2^16/5
//unsigned int vzvsb_16 = 5500;  This worked forever before removing linear caps
unsigned int vcntrlb1_16 = 20000; //Was 14000 before 10-13-18
unsigned int vzvsb1_16 = 7500;
unsigned int vcntrla1_16 = 23000;
unsigned int vzvsa1_16 = 40000;

//For now I'm just initializing these so it'll compile
unsigned int vcntrlb2_16 = 7500;
unsigned int vzvsb2_16 = 7500;
unsigned int vcntrla2_16 = 7500;
unsigned int vzvsa2_16 = 6500;

unsigned int timeb_pause_16;
unsigned int vout;
unsigned int i = 0;
unsigned int ControlCount = 0;
unsigned int KickCount = 0;
unsigned int VrmsCount = 0;

double Vin_prevSign = 0;
int signstate = 1;
int syncrecenable = 0;
int bridgeenable = 0;
unsigned int peakwaitcount = 0;
unsigned int SlgnCount = 0;
//double zvsamult = 0.8;  //This worked forever before removing linear caps
double zvsamult = 0.8;  //Was 0.75 before 10-13-18, trouble with A reaction in startup
double temp;
double timeb;
double timeb2 = 0;
double timea;
//double Cb = 8000*65536;//7000*65536;  = L*i, where L approx "4000"

//double L = 16600; //Inductance, units ns*V/A (= nH) was 16600
double L = 14000;  //Rachel says inductor is actually like 13.9 uH
double i1 = 1.9;  //Current, units A  (2.8 worked for a long time)
double i2 = 2.1;  //Current, units A
//double Cb = 40000; //Units ns*V; 32000 corresponds to 16 uH * 2.5 A
//double Ca = 5150*65536;//5000*65536;  = L*i2
//double Ca = 28800; //Units ns*V; 28800 corresponds to 16 uH * 1.8 A
double i1Startup = i1;
double integral = i1;
double derivative;
double error;
double preverror;
double correctiona = 0;  //ns
double correctionb = 0;  //ns
double correctionb_prefactor = 0.133; //  ns/V
double correctiona_prefactor = 0.216 //  ns/V
double Vout;
double Vin;
double Vb;
double Va;
//double rampslope = 0.00375;  //3 Volt per 800 ns, verified for 5.6 kohm, 220 pF
//double rampslope = 0.00191;  //1.53 Volt per 800 ns, assumed for 11 kohm, 220 pF
double rampslope = 0.00364;  //3.64 volt per us, as measured on 10/2/18 for all 4 switches
double rampslopea = 0.000954;  // Changed from 220p, 5.6k to 470p, 10k, which should yield this 4x lower rampslope
double R = 125;  //125 worked well for a long time //R=250 gives 40 W at 120 Vrms  //10-25-18:  250 is too high almost to be practical (esp when transitioning)
double Peff = 220*220/R;
double I3 = 0.0034;  //A/W  (spec given in mA/W)
double I5 = 0.0019;
double I7 = 0.0010;
double I9 = 0.0005;
double I11 = 0.00035;
double limitpercent = 0;

double tnormal;
double tforres;
double tforcap;
double ttotal;
double tbuck;
double i1buck;
double timea2;
double C_in = 4500 ;  //3-18-19 worked fine at 3500 for a long time, but not enough i think //Used to be 2000 = 2000 ns * A / V = 2 uF
//Technically should be 4400 for 4.4 uF, but I don't think I usually use the real value b/c derating
double w_line = 0.0000003142; // per ns  (50 hz)
double Vrms = 220; //volts
double VinMax = 0;
//double Cres = 0.12;  //100 pF = 0.1 ns * A/V
//double Cres = 0.15;  //0.12 worked well for a long time; needed higher for zero crossing, stitching
//double Cres = 0.05;  //Used 0.15 before removing linear caps
double Cres = 0.125;  //Used 0.05 (50 pF) for Navitas; using .125 (125 pF) for large panasonic parts

double Iin = 0;
double iconv = 0;
double x = 0;
double D = 0;
double test = 1;

double Vin_prev = 0;
double Vout_prev = 0;
double cap_sign = 1;
double captoggle = 0;

double bridge_hyst = 80;

char buff[16];

// </editor-fold>
```

```
  SetupSPIa1();
  SetupSPIa2();
  SetupSPIb1();
  SetupSPIb2();
  SetupADC();
  SetupTimer();
  SetupUART();

  PRECONbits.PREFEN = 0b11;  //Allow Predictive Prefetch of CPU instructions and data
  PRECONbits.PFMWS = 0b000;  //Zero wait states for PFM access time (see Table 37-13)

  WriteSPIa1(&vzvsa1_16,'z');
  WriteSPIa1(&vcntrla1_16,'c');
  WriteSPIa2(&vzvsa2_16,'z');
  WriteSPIa2(&vcntrla2_16,'c');

  WriteSPIb1(&vzvsb1_16,'z');
  WriteSPIb1(&vcntrlb1_16,'c');
  WriteSPIb2(&vzvsb2_16,'z');
  WriteSPIb2(&vcntrlb2_16,'c');

  for(i=0;i<10000;i++){Nop();} //Letting modules get startad up, clean WriteMenu
  WriteScreen("\n\r");
  WriteMenu();


  while(1) {


    if(Flags.U1RX == 1) {
      // <editor-fold defaultstate="collapsed" desc="User Interface Update">
      WriteScreen("\n\r");
      //selection = U4RXREG; U4TXREG = selection;  //Echo selection to the screen

      switch (selection) {
        case 'u':
          limitpercent = limitpercent + 0.1;
          //sprintf(buf, "%.1f", limitpercent);
          //WriteScreen(buf);
          break;
        case 'r':
          limitpercent = limitpercent - 0.1; if(limitpercent<0){limitpercent=0;}
          //%sprintf(buf, "%.1f", limitpercent);
          //WriteScreen(buf);
          break;
        case 'a':
          zvsamult = zvsamult + 0.05; break;
        case 'x':
          zvsamult = zvsamult - 0.05; break;
        case 'q':
          Flags.Control = 1; break;
        case 'd':
          WriteVout(); break;
        case 'c':
          WriteVin(); break;
        case 'v':
          WriteVh(); break;
        case 'f':
          WriteVn(); break;
        case 'g':
          Peff = Peff + 25;
          R = (220*220)/Peff; break;
        case 'b':
          Peff = Peff - 25;
          R = (220*220)/Peff; break; break;
        case 'h':
          I2 = I2*1.05; break; //WriteSPIa(&vzvsb_16, 'c'); break;
        case 'n':
          I2 = I2*0.95; break; //WriteSPIa(&vzvsb_16, 'c'); break;
        case 'l':
          if(captoggle == 0) {captoggle=1; WriteScreen("1");}
          else{captoggle = 0; WriteScreen("0");}
          break;
        case 'r':
          if(syncrecenable == 0) {syncrecenable=1; WriteScreen("1");}
          else{syncrecenable = 0; WriteScreen("0");}
          break;
        case 'e':
          if(bridgeenable == 0) {bridgeenable=1; WriteScreen("1");}
          else{bridgeenable = 0; WriteScreen("0");}
          break;
        case 'y':
          if(RCONbits.EXTR  == 1){WriteScreen("EXTR\n\r");}
          if(RCONbits.SWR   == 1){WriteScreen("SWR\n\r");}
          if(RCONbits.WDTO  == 1){WriteScreen("WDTO\n\r");}
          if(RCONbits.DMTO  == 1){WriteScreen("DMTO\n\r");}
          if(RCONbits.SLEEP == 1){WriteScreen("SLEEP\n\r");}
          if(RCONbits.IDLE  == 1){WriteScreen("IDLE\n\r");}
          if(RCONbits.CMR   == 1){WriteScreen("CMR\n\r");}
          if(RCONbits.BOR   == 1){WriteScreen("BOR\n\r");}
          if(RCONbits.POR   == 1){WriteScreen("POR\n\r");}
          RCON = 0b0000000000000000111111111111111;
          RCONbits.BOR = 0;
          RCONbits.POR = 0;
          break;
        case 'm':
          WriteMenu(); break;
        default:
          WriteScreen("Invalid Selection");
      } //End of switch statement

      WriteScreen("\n\r");
      Flags.U1RX = 0;
      // </editor-fold>
    } // End of if(Flags.U1RX)


    if(Flags.Timer == 1) {
      // <editor-fold defaultstate="collapsed" desc="State Update">
      Flags.Timer = 0;


      ////
      //JUST FOR TESTING, DELETE BEFORE RUNNING
      //WriteSPIa2(&vzvsa2_16,'z');
      //WriteSPIa2(&vcntrla2_16,'c');
      //WriteSPIb1(&vzvsb1_16,'z');
      //WriteSPIb1(&vcntrlb1_16,'c');
      //WriteSPIb2(&vzvsb2_16,'z');
      //WriteSPIb2(&vcntrlb2_16,'c');


      ////




      ////////////// Voltage Update ////////////////////////
     //Taking ADC data and convert to 16-bit (double) where number is in volts
      Vout_prev = Vout;
      Vout = (double) (ADCDATA3>>16); //.0067575 = 3.3 * 134.3 / 2^16
      Vout = Vout*(3.3 * 127 / 65536);
      Vout = (Vout_prev + Vout)/2;

      Vin_prev = Vin;
      Vin = (double) (ADCDATA4>>16);
      Vin = Vin*(3.3 * 127 / 65536);  //134.3 was a little high
      if(Vin_prev > 20 && Vin > 1.15*Vin_prev){Vin = 1.15*Vin_prev;}
      else if(Vin_prev > 20 && Vin < 0.85*Vin_prev){Vin = 0.85*Vin_prev;}
      //Vin = (Vin_prev + Vin)/2;

      Vh = (double) (ADCDATA1>>16);
      Vh = Vh*(3.3 * 127 / 65536);

      Vn = (double) (ADCDATA2>>16);
      Vn = Vn*(3.3 * 127 / 65536);


      ////////////////  Setting the input bridge ////////////
      NONPRE_clr;
      HONPRE_clr; //Active code replaced by diode operation 10-14-18
              //Active code disabled 4-18-19 for final blue board to start

      if(bridgeenable == 1 && Peff > 200){
        if(Vh > Vn + bridge_hyst && Vin > 110) {
          NONPRE_clr;
          HONPRE_set;
          DIO1_set;
        } else if(Vn > Vh + bridge_hyst && Vin > 110) {
          HONPRE_clr;
          NONPRE_set;
```

```
            DIO3_set;
          } else{
            HONPRE_clr;
            NONPRE_clr;
            DIO1_clr;
            DIO3_clr;
          }
        }
        else{
          HONPRE_clr;
          NONPRE_clr;
        }

        if(Peff < 200){syncrecenable = 0;}



        ////////////////////  Vrms Detection ////////////////////
        if(Vin > Vrms*1.414) {Vrms = Vin * 0.7071;}
        if(Vin > VinMax) {VinMax = Vin;}
        VrmsCount = VrmsCount + 1;
        if(VrmsCount > 1000){    //Timer goes off every 32 us, or 259 times per half-period
                        //So waiting for every 4 half-cycles is about 1000 counts

          //if(VinMax > 350) { Vrms = 240; }  //Don't want it to accidentally think the voltage is crazy high
          //else { Vrms = VinMax * 0.7071 + 5; }
          if(VinMax < Vrms*1.414){Vrms = VinMax*0.7071;}

          VinMax = 0;
          VrmsCount = 0;
        }


        ////////////////////  dV/dt sign Detection ////////////////


        //THIS IS THE DIRECT, OLD WAY
//        if(Vin-Vin_prev > 0) {Flags.Rising = 1; cap_sign = -1;}
//        else {Flags.Rising = 0; cap_sign = 1;}
//
//        if(captoggle == 0) {cap_sign = 0;}




        ////This is the state-machine, new way.  More robust.
        if(SignCount == 2) {  //32 us * 8 = 256 us  WAS 8

          switch(signstate) {

            case 1: //voltage rising
              cap_sign = -1;
              if(Vin > Vrms*1.41*0.85) {  //WAS 0.85
                signstate = 2; peakwaitcount = 0; //WriteScreen("2");
                if(Flags.Control == 1) {
                  R = R + (Vout-400)*0.04;  //0.1 worked fine at low power
                }
                //StartupBoost();
              }//WriteScreen("2");  //Moving to top state. #2
              break;
            case 2: //voltage near peak
              cap_sign = 0;
              if(peakwaitcount > 8) {signstate = 3; peakwaitcount = 0;  }// USED TO WAIT FOR 2 peakwaitcounts
              else {peakwaitcount = peakwaitcount + 1;}
              break;
            case 3: //Voltage falling
              cap_sign = 1;
              if(Vin < Vrms*1.41*0.25) {signstate = 4;  }//WriteScreen("4");}
              break;
            case 4:
              cap_sign = 1;  //Was -1, could be 0.  I want +1 because it will add more on both sides for zero crossing distortion
              if(Vin > Vin_prevSign-3){
                signstate = 1;
                StartupBoost();
              }//WriteScreen("1");}
              break;
            default:
              cap_sign = -1;
              signstate = 1;
          } //close switch

          SignCount = 0;
          Vin_prevSign = Vin;

        }
        else {
          SignCount = SignCount + 1;
        }

        if(Vrms < 70){cap_sign=1;}





        ////////////////////// Sanitize Vin to avoid sqrt problems ////////////
        if(Vin > Vrms*1.41) {Vin = 1.41*Vrms-1;}  //
        //If Vin is even close to Vrms*sqrt(2), make it just smaller than Vrms*sqrt(2)
        //Because Vrms updates every cycle, this should only engage at the very edges


        if(Flags.Running==0 && Flags.Mode==0){
          // <editor-fold defaultstate="collapsed" desc="Charging State">


          if(Vout > 100) { //350 normally -- above this Vout regular boost safe
            //WriteScreen("B");
            Flags.Running = 1;
            Flags.Mode = 0;
            timeb = 300;



            StartupBoost();
          } else {

            //Make sure A2 and B2 are off
            ENABLEb2_set;
            ENABLEa2_set;
            MANUALb2_clr;
            MANUALa2_clr;


            vzvsa1_16 = (unsigned int) (Vin/100 * zvsamult * 65536/5); // LV * 16-bit/5V
            WriteSPIa1(&vzvsa1_16,'z');


            if(Vin<350) {correctionb = correctionb_prefactor * (400-Vin-50);} //10-13-18 do I really need this?
            else {correctionb = 0;}

            //timeb = L*(1Startup/(Vin)+correctionb;  Commented 10-13-18
            timea = timeb+50;  //startup with triangular waveform

            //Commented 10-13-18 -- shouldn't a1 just be on for b1 + a little (from initialization)?
            //vcntrla1_16 = (unsigned int) (timea * rampslope * 65536/5);  // ns * V/ns * code/V
            //WriteSPIa1(&vcntrla1_16,'c');


            for(i=0;i<3;i++) {Nop();}
            StartupLong();



          }
          // </editor-fold>
        }


        //Mod Boost State should be 1 1
        if(Flags.Running==1 && Flags.Mode==1){
          // <editor-fold defaultstate="collapsed" desc="Mod Boost State">

          //Make sure A2 and B2 are off
          //ENABLEb2_set;
          ENABLEa2_set;
          MANUALb2_clr;
          MANUALa2_clr;


          if(Vout<100){  //19660 => 150 V
            // <editor-fold defaultstate="collapsed" desc="Mod -> Charge">
            ENABLEb2_set;
            Flags.Running = 0;
```

```
            Flags.Mode = 0;
            //WriteScreen("C");

            ENABLEb1_set; //Enable B should be on
            ENABLEa1_set; //EnableA should be on
            MANUALb1_clr; //Set Manual B to off
            MANUALa1_clr; //Set Manual A to off
            // </editor-fold>
          }
          else if(Vin < Vout*0.47 ){  //4-20-19 I used 95 for dedicated 200V mode, next comment old.  //Under 193 V => turn on boost state
            // <editor-fold defaultstate="collapsed" desc="Mod -> Boost">

            ENABLEb2_set;
            ENABLEa2_set;
            Flags.Mode = 0; //WriteScreen("B");
            Flags.Running = 1;

            //if(Vin<350) {correctionb = correctionb_prefactor * (400-Vin-50); }
            //else {correctionb = 0; }

            //timeb = L*I1/(Vin)+correctionb;

            //timeb = timeb*0.6;  //THIS NEEDS TO CHANGE WITH NEW ALGORITHM

            //No need to calculate R = R(i1) with new and improved HV mode
            //R = 200 / ( ( 0.5*timeb*I1 + 0.5*(I1+I2)*(timea-timeb) )  / (  timea + L*I2/Vout + 3.1415*sqrt(L*Cres/2) )    );
            //Note the use of Cres/2 since the net capacitance is 2 Cres in series when in Mod Boost mode
            //R = Vin / <lin>


            Flags.Rising = 0;
            //cap_sign = 1;  I was fixing cap_sign because of Vin measurement error

            ttotal = 2*L/R + (Vout/Vin)*sqrt(L*Cres)*(sqrt(1 - 2*Vin/Vout) + 1 - Vin/Vout) + cap_sign * 2*L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
            if(ttotal > (4.5/rampslope) )  ttotal = 4.5/rampslope;
            else if (ttotal < 0) ttotal = 0;
            vcntrlb1_16 = (unsigned int) (ttotal * rampslope * 65536/5);
            WriteSPIb1(&vcntrlb1_16,'c');



            //timeb_pause_16 = (unsigned int) (0.6 * timeb * rampslope * 65536/5);
            //WriteSPIb(&timeb_pause_16,'c');

            StartupBoost();
            // </editor-fold>
          }
          else if(Vin > Vout+20){  //
            // <editor-fold defaultstate="collapsed" desc="Mod -> Buck">

            ENABLEb2_set;
            ENABLEa2_set;
            Flags.Mode = 1; //WriteScreen("B");
            Flags.Running = 0;


            tbuck = 3600;
            vcntrla1_16 = (unsigned int) (tbuck * rampslopea * 65536/5);
            WriteSPIa1(&vcntrla1_16,'c');

            StartupBuck();

            // </editor-fold>
          }
          else {  //If no change required, set new values and give a new kick
            // <editor-fold defaultstate="collapsed" desc="Keep Mod">
            //Old System
//            if(Vin<350) {correctionb = correctionb_prefactor * (400-Vin-50); }
//            else {correctionb = 0; }
//            timeb = L*I1/(Vin)+correctionb;  //Cb is in ns*V, total is in ns
//            vcntrlb_16 = (unsigned int) (timeb * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
//            WriteSPIb(&vcntrlb_16,'c');

//            if(Vin<343) {correctiona = correctiona_prefactor * (400-Vin-57); }
//            else {correctiona = 0; }
//            timea = (L*I1-L*I2)/(400-Vin) + timeb + correctiona;
//            vcntrla_16 = (unsigned int) (timea * rampslope * 65536/5); // ns * (V/ns) => convert to 16 bit 5 V DAC output
//            WriteSPIa(&vcntrla_16,'c');

//            vzvsa_16 = (unsigned int) (Vin * zvsamult/100 * 65536/5); // Step down to 5 V domain, => convert to 16 bit 5V DAC output
//            WriteSPIa(&vzvsa_16,'z');

            //New System
//            x = Vin/Vout;
//            D = (3.1415*Vout/(L*I2)) * sqrt(L*Cres/2);


//            iin = (Vin/R) * (  1 + (3*I3*limitpercent+5*I5*limitpercent)*Vrms - (4*I3*limitpercent+20*I5*limitpercent)*Vin*Vin*0.5/Vrms   + (16*I5*limitpercent)*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  );
//            iconv = iin + cap_sign*C_in*w_line*sqrt(2*Vrms*Vrms-Vin*Vin);
//            i1 = iconv + sqrt(iconv*iconv + I2*I2*x - 2*iconv*I2*x*x + 2*iconv*I2*D*x*(1-x));

            i1 = Compute1(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, I2, limitpercent, I3, I5);
            x = Vin/Vout;
            timeb = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin;
            //timea = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin + (I1-I2)*L/(Vout-Vin) + sqrt(L*Cres/2)*(3.1415/2 + (1-2*Vin/Vout));  //Approximation for arccos
            //USED TO USE Cres/2
            timea = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin + (I1-I2)*L/(Vout-Vin) + sqrt(L*Cres/2)*(  4*(1-x)/(sqrt(x-x*x) + .707*(1-x))  );
            // NOTE THE ABOVE EQUATION SHOULD USE 1.4 for SQRT(2) -- why was I using 0.707?  Probably brain fart
            // Not sure when I wrote the above.  As of 10/8/18, still says 0.707 ... now not sure if I should change it


            ///THIS IS THE TRANSITION MODE
            if(Vin > (Vout-17)){
              timea = 750; //4-20-19  worked well at 750 for a long time, longer is more efficient but risks failing to oscillate
              if(Peff < 250) {timea = 200;}//4-20-19 added b/c low power failing at 240 Vin
              timeb = Computetbtran(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, I2, limitpercent, I3, I5,timea);
              if(timeb > timea) {timeb = timea;} //Just as a precaution
            }

            vcntrla1_16 = (unsigned int) (timea * rampslopea * 65536/5); // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIa1(&vcntrla1_16,'c');


            vzvsa1_16 = (unsigned int) (Vin * zvsamult/100 * 65536/5); // Step down to 5 V domain, => convert to 16 bit 5V DAC output
            WriteSPIa1(&vzvsa1_16,'z');

            vcntrlb1_16 = (unsigned int) (timeb * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIb1(&vcntrlb1_16,'c');


            ////////////////////////////////////////////////////////////////////
            //This should get SB2 to synchronously rectify during the mod-boost mode
            vzvsb2_16 = (unsigned int) (Vout/100 * 0.75 * 65536/5); // Step down to 5 V domain, => convert to 16 bit 5V DAC output
            WriteSPIb2(&vzvsb2_16,'z');


            timea2 = I2 * L / Vout * 0.4;  //the 0.6 is for safety
            vcntrla2_16 = (unsigned int) (timea2 * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIa2(&vcntrla2_16,'c');

            timeb2 = timea-timeb + timea2/2; // the /2 is for safety
            if(timeb2 > 1000){timeb2 = 1000;}
            vcntrlb2_16 = (unsigned int) (timeb2 * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIb2(&vcntrlb2_16,'c');

            //if(syncreceivable == 1){ENABLEb2_clr;}

            ////////////////////////////////////////////////////////////////////


            Nop();

            //if(KickCount > 1) {
            if(1){
              //I'm enclosing StartupCycle with disabling B2
              //Having B2 operational can keep the "kick" from working
              ENABLEb2_set;
              StartupCycle();
              for(i=0;i<10;i++){Nop();}
              if(syncreceivable == 1 && Vin > Vout*0.52){ENABLEb2_clr; ENABLEa2_clr;}
              KickCount = 0;
            } else {
              KickCount = KickCount + 1;
            }
            // </editor-fold>
          }
          // </editor-fold>
        }


        //Boost State
        if(Flags.Running==1 && Flags.Mode==0) {
          // <editor-fold defaultstate="collapsed" desc="Boost State">

          if(Vout<100){  //10000 => 76 V
            // <editor-fold defaultstate="collapsed" desc="Boost -> Charge">
            //WriteScreen("C");
            Flags.Running = 0;
```

```
1018            Flags.Mode = 0;
1019
1020            ENABLEb1_set; //Enable B should be on
1021            ENABLEa1_set; //EnableA should be on
1022            MANUALb1_clr; //Set Manual B to off
1023            MANUALa1_clr; //Set Manual A to off
1024            // </editor-fold>
1025          }
1026          else if(Vin > Vout*.47 ){  //4-20-19 I used 95 for dedicated 200V mode, next comment old.  Over 195 V => turn on mod boost state
1027            // <editor-fold defaultstate="collapsed" desc="Boost -> Mod Boost">
1028
1029            if(Flags.Control == 1) {
1030              // <editor-fold defaultstate="collapsed" desc="Control">
1031              if(ControlCount > 10){
1032                error = 405 - Vout;
1033
1034                //If output voltage gets beyond 450, knock it down faster
1035                if(error < -50) {  integral = (integral + error)*.8;  }
1036                else {  integral = integral + error;  } // error (V) }
1037
1038                i1 = (1/50) * integral + (1/15) * error; // coefficients are (A/V)
1039                if(i1 < 2) {i1 = 2;} // minimum i1 = 2A
1040                //Note that 1/15 means every 15 volts of error produces a 1A change in i1
1041                //This is actually pretty strong...wouldn't be surprised if unstable
1042
1043                ControlCount = 0;
1044              } else {
1045              ControlCount = ControlCount + 1;
1046              }
1047              // </editor-fold>
1048            }
1049
1050
1051
1052            //Old way
1053 //           if(Vin<350) { correctionb = correctionb_prefactor*(400-Vin-50);  //Answer in ns
1054 //           else{correctionb = 0;}
1055 //           timeb = L*i1/(Vin)+correctionb;
1056 //
1057 //
1058 //           if(Vin<343) {correctiona = correctiona_prefactor*(400-Vin-57);}
1059 //           else{correctiona = 0;}
1060 //           timea = (L*i1-L*i2)/(400-Vin) + timeb + correctiona;
1061 //
1062 //           vcntrla_16 = (unsigned int) (timea * rampslope * 65536/5);
1063 //           WriteSPIa(&vcntrla_16,'c');
1064 //
1065 //           vzvsa_16 = (unsigned int) (Vin * zvsamult/100 * 65536/5);
1066 //           WriteSPIa(&vzvsa_16,'z');
1067 //
1068 //           vcntrlb_16 = (unsigned int) (timeb * rampslope * 65536/5);
1069 //           WriteSPIb(&vcntrlb_16,'c');
1070
1071
1072
1073            i1 = Compute11(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, i2, limitpercent, i3, i5);
1074            x = Vin/Vout;
1075            timeb = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin;
1076            //timea = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin + (i1-i2)*L/(Vout-Vin) + sqrt(L*Cres/2)*(3.1415/2 + (1-2*Vin/Vout));  //Approximation for arccos
1077            timea = L*(i1+Vout*sqrt(Cres/L)*(1-Vin/Vout))/Vin + (i1-i2)*L/(Vout-Vin) + sqrt(L*Cres/2)*(  4*(1-x)/(sqrt(x-x*x) + .707*(1-x))  );
1078            //USED TO USE CRES/2.  Also why is it still 0.707 (see other comment)
1079
1080
1081            ENABLEb2_set; // Don't want to accidentally cause problems
1082
1083            vcntrla1_16 = (unsigned int) (timea * rampslopea * 65536/5); // ns * (V/ns) => convert to 16 bit 5 V DAC output
1084            WriteSPIa1(&vcntrla1_16,'c');
1085
1086            vzvsa1_16 = (unsigned int) (Vin * zvsamult/100 * 65536/5);
1087            WriteSPIa1(&vzvsa1_16,'z');
1088
1089            vcntrlb1_16 = (unsigned int) (timeb * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
1090            WriteSPIb1(&vcntrlb1_16,'c');
1091
1092
1093
1094
1095            Flags.Mode = 1;
1096            Flags.Running = 1;
1097            KickCount = 0;
1098
1099            StartupCycle();
1100            // </editor-fold>
1101          }
1102 //       else if(Vin < 25 ) { //Under 25 V => pause
1103 //          // <editor-fold defaultstate="collapsed" desc="Boost -> Pause">
1104 //          //WriteScreen("P");
1105 //          Flags.Running = 0;
1106 //          Flags.Mode = 1;
1107 //          StartupBoost();
1108 //          // </editor-fold>
1109 //       }
1110          else {//If no change required, then give another kick to keep going
1111            // <editor-fold defaultstate="collapsed" desc="Keep Boost">
1112
1113            //if(Vin<60 && cap_sign == 1){test = 1.3;} else{test=1;}
1114
1115            //ttotal = 2*L/R + 2*(Vout/Vin)*sqrt(L*Cres * (1 - 2*Vin/Vout)) + cap_sign * L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
1116            //ttotal = 2*L/R * (1 + 3*Vrms * 0.0034 - 2 * Vin*Vin * .0034 / Vrms) + 2*(Vout/Vin)*sqrt(L*Cres * (1 - 2*Vin/Vout)) + cap_sign * L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
1117            //ttotal = 2*L/R * (1 + limitpercent*(3*i3+5*i5)*Vrms - limitpercent*(4*i3+20*i5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*i5)*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  ) + (Vout/Vin)*sqrt(L*Cres)*(sqrt(1 - 2*Vin/Vout)+1-Vin/Vout) + cap_sign * 2*L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
1118            ttotal = 2*L/R  + (Vout/Vin)*sqrt(L*Cres)*(sqrt(1 - 2*Vin/Vout)+1-Vin/Vout) + cap_sign * 2*L * C_in * w_line * sqrt(2*Vrms*Vrms/(Vin*Vin) - 1);
1119            //When I ran this before I didn't have the 2 before the C_in term, but it should be there from my notes
1120            //I'm trying with the factor of 2 but reducing C_in as a variable
1121            //This will clean up the HV mode (which clearly is acting like C_in is too big) and also matches what I think the actual C_in is better
1122            if(ttotal > (4.5/rampslope) )  ttotal = 4.5/rampslope;
1123            else if (ttotal < 0) ttotal = 0;
1124
1125            vcntrlb1_16 = (unsigned int) (ttotal * rampslope * 65536/5);
1126            WriteSPIb1(&vcntrlb1_16,'c');
1127
1128            ////Sync Rsc
1129            vzvsb2_16 = (unsigned int) (Vout/100 * 0.75 * 65536/5); // Step down to 5 V domain, => convert to 16 bit 5V DAC output
1130            WriteSPIb2(&vzvsb2_16,'z');
1131
1132            timeb2 = ttotal * Vin / (Vout - Vin) * 0.6;  //The 0.6 is for safety
1133            vcntrlb2_16 = (unsigned int) (timeb2 * rampslope * 65536/5);
1134            WriteSPIb2(&vcntrlb2_16,'c');
1135            //////
1136
1137
1138            StartupBoost();
1139            for(i=0;i<5;i++){Nop();}
1140            if(syncreconable == 1 && Vin > 45) {ENABLEb2_set;}
1141
1142            if(Vin < 45){ENABLEb2_set;}
1143            // </editor-fold>
1144          }
1145          // </editor-fold>
1146        }
1147
1148
1149      //Buck State
1150      if (Flags.Running==0 && Flags.Mode==1){
1151        // <editor-fold defaultstate="collapsed" desc="Buck State">
1152
1153        if(Vout<150){
1154          // <editor-fold defaultstate="collapsed" desc="Buck -> Charging">
1155
1156
1157            ENABLEa2_set;
1158            Flags.Running = 0;
1159            Flags.Mode = 0;
1160
1161            ENABLEb1_set; //Enable B should be on
1162            ENABLEa1_set; //EnableA should be on
1163            MANUALb1_clr; //Set Manual B to off
1164            MANUALa1_clr; //Set Manual A to off
1165            // </editor-fold>
1166          }
1167          else if(Vin < Vout+20 ){
1168            // <editor-fold defaultstate="collapsed" desc="Buck -> Mod Boost">
1169
1170            if(Flags.Control == 1) {
1171              // <editor-fold defaultstate="collapsed" desc="Control">
1172              if(ControlCount > 10){
1173                error = 405 - Vout;
1174
1175                //If output voltage gets beyond 450, knock it down faster
1176                if(error < -50) {  integral = (integral + error)*.8;  }
1177                else {  integral = integral + error;  } // error (V) }
1178
1179                i1 = (1/50) * integral + (1/15) * error; // coefficients are (A/V)
1180                if(i1 < 2) {i1 = 2;} // minimum i1 = 2A
1181                //Note that 1/15 means every 15 volts of error produces a 1A change in i1
1182                //This is actually pretty strong...wouldn't be surprised if unstable
1183
1184                ControlCount = 0;
1185              } else {
1186              ControlCount = ControlCount + 1;
1187              }
```

```
                // </editor-fold>
            }


            ///THIS IS THE TRANSITION MODE
            if(Vin > (Vout-20)){
                timea = 800; //4-20-19  worked well at 750 for a long time, longer is more efficient but risks failing to oscillate
                if(Poff < 250){timea = 200;}//4-20-19 added b/c low power failing at 240 Vin
                timeb = Computetbtmn(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, I2, limitpercent, I3, I5,timea);
                if(timeb > timea) {timeb = timea;} //just as a precaution
            }


            vcntrla1_16 = (unsigned int) (timea * rampslopea * 65536/5); // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIa1(&vcntrla1_16,'c');

            vzvsa1_16 = (unsigned int) (Vin * zvsamult/100 * 65536/5);
            WriteSPIa1(&vzvsa1_16,'z');

            vcntrlb1_16 = (unsigned int) (timeb * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIb1(&vcntrlb1_16,'c');

            ////////////////////////////////////////////////////////////////////////
            ///This should get SB2 to synchronously rectify during the mod-boost mode
//          vzvsb2_16 = (unsigned int) (Vout * zvsamult/100 * 65536/5); // Step down to 5 V domain, => convert to 16 bit 5V DAC output
//          WriteSPIb2(&vzvsb2_16,'z');
//
//          timeb2 = timea-timeb;
//          if(timeb2 > 1000){timeb2 = 1000;}
//          vcntrlb2_16 = (unsigned int) (timeb2 * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
//          WriteSPIb2(&vcntrlb2_16,'c');

            //if(syncrecenable == 1){ENABLEb2_clr;}

            ////////////////////////////////////////////////////////////////////////



            ENABLEa2_set;
            MANUALa2_clr;
            ENABLEb2_set;
            MANUALb2_clr;
            Flags.Mode = 1;
            Flags.Running = 1;

            StartupCycle();
            // </editor-fold>
        }
        else{
            // <editor-fold defaultstate="collapsed" desc="Keep Buck">

            I1buck = ComputeI1buck(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, I2, limitpercent, I3, I5);
            tbuck = Computetbuck(Vin, Vout, Vrms, R, L, Cres, cap_sign, C_in, w_line, I2, limitpercent, I3, I5);
            tbuck = tbuck*1.2;

            if(tbuck>4720){tbuck = 4720;}  //This is 4.5 V / (rampslopea = 0.000954) = 4.72 us

            vcntrla1_16 = (unsigned int) (tbuck * rampslopea * 65536/5);
            WriteSPIa1(&vcntrla1_16,'c');


            ////////////////////////////////////////////////////////////////////////
            ///This should get SA2 to synchronously rectify during the buck mode

            timea2 = I1buck*L/Vout * 0.5;  //The 0.5 is for safety
            if(timea2 > 1000){timea2 = 1000;}  //As of 3-16-19, the SA2 setup only has a ramp for about 1 us, not several
            vcntrla2_16 = (unsigned int) (timea2 * rampslope * 65536/5);  // ns * (V/ns) => convert to 16 bit 5 V DAC output
            WriteSPIa2(&vcntrla2_16,'c');


            //if(syncrecenable == 1){ENABLEa2_clr;}

            ////////////////////////////////////////////////////////////////////////



            ENABLEa2_set;
            StartupBuck();
            for(i=0;i<10;i++){Nop();}
            if(syncrecenable == 1){ENABLEa2_clr;}


            // </editor-fold>
        }
        // </editor-fold>
    }
    // </editor-fold>
  }  //End of timer interrupt


  } //End of infinite while

  return (EXIT_SUCCESS);
}



void __ISR_AT_VECTOR(_UART4_RX_VECTOR, IPL2AUTO) IntUart4Handler(void) {

  Flags.U1RX = 1;
  selection = U4RXREG; U4TXREG = selection;  //Echo selection to the screen
  //WriteScreen("hi");
  IPS5bits.U4RXIF = 0;

}

void __ISR_AT_VECTOR(_TIMER_1_VECTOR, IPL2AUTO) IntTimer1Handler(void) {

  IPS0bits.T1IF = 0;

}

void __ISR_AT_VECTOR(_TIMER_2_VECTOR, IPL2AUTO) IntTimer2Handler(void) {

  Flags.Timer = 1;
  IPS0bits.T2IF = 0;

}

void SetupADC(void){

  //First thing, load calibration data into the CFG registers
  //DEVADC is stored in flash memory
  //The *only* thing the FRM says is that the user must copy the data
  //Examples in the FRM have only these exact commands
  ADC1CFG = DEVADC1;  //Vh, Pin 17 (B6), AN46
  ADC2CFG = DEVADC2;  //Vn, Pin 14 (B2), AN2
  ADC3CFG = DEVADC3;  //Vout_low, Pin 21 (B8), AN48
  ADC4CFG = DEVADC4;  //Vin_low, Pin 22 (B9), AN49

  //Page 474 "When an alternate input is used as the input source for a
  //  dedicated ADC module, the data output is still read from the primary
  //  input data output register"



  //Check section 22.4.3 of FRM "Selecting the Format of ADC result
  //Fractional is left justified, while integer is right-justified
  ADCCON1bits.FRACT = 1;  //Fractional output (left justified, 16-bit number with zeros for LSB)
  ADCIMCON1bits.SIGN1 = 0;
  ADCIMCON1bits.SIGN2 = 0;
  ADCIMCON1bits.SIGN3 = 0;
  ADCIMCON1bits.SIGN4 = 0;

  ADCCON1bits.SIDL = 0;   //Keep running in idle mode
  ADCCON1bits.AICPMPEN = 0; //Charge pump disabled (for Vdd > 2.5, see P427)
  CPGCONbits.IOANCPEN = 0;  //Charge pump disabled (for Vdd > 2.5, see P427, 596)
  ADCCON1bits.PSSCLKEN = 1; //Past System Clock to ADC Control Clock Enable (see FRM 22 p.77)
  ADCCON1bits.PSPBCLKEN = 1; //Peripheral clock to ADC enable (specific conditions FRM22 p.77)

  ADCCON3bits.ADCSEL = 0b11; //Use SYSclk as input (can use FRC, PBCLK3, REFCLK3)
  ADCCON3bits.CONCLKDIV = 1; //SYSclk/2 = 32 MHz = TQ
  ADC1TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC1TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
  ADC2TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC2TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
  ADC3TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC3TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
  ADC4TIMEbits.ADCDIV = 0b0000001;  //2 * Tq = Tad  = 16 MHz
  ADC4TIMEbits.SAMC = 0b0000001000;  //Sample time = 8*Tad
```

```
//Recall that 12-bit conversion takes 13 Tad
//(8+13)/(16 MHz) = 1.3 us
//With Timer1 running at 64 MHz, need to count to at least 84

ADCCON3bits.VREFSEL = 0;  //Use AVdd and AVss as pos/neg references

ADCTRGMODEbits.SH1ALT = 0b01;  //Use AN46 on ADC1 (Vh)
ADCTRGMODEbits.SH2ALT = 0b00;  //Use AN2  on ADC2 (Vn)
ADCTRGMODEbits.SH3ALT = 0b01;  //Use AN48 on ADC3 (vout_low)
ADCTRGMODEbits.SH4ALT = 0b01;  //Use AN49 on ADC4 (vin_low)


ADCIMCON1 = 0;  //All inputs use single-ended, unsigned data
ADCIMCON2 = 0;
ADCIMCON3 = 0;

ADCGIRQEN1 = 0;  //No interrupts from any ADC
ADCGIRQEN2 = 0;

ADC1TIMEbits.SELRES = 0b11; //12 bit resolution
ADC2TIMEbits.SELRES = 0b11; //12 bit resolution
ADC3TIMEbits.SELRES = 0b11; //12 bit resolution
ADC4TIMEbits.SELRES = 0b11; //12 bit resolution

//From Section 22 of FRM:  "Each Class 1 input has a unique trigger and
//  upon arrival of the trigger, ends sampling and starts conversion.
//  Upon completion of conversion, the ADC module reverts back to
//  sampling mode.  When a Class 1 input is enabled and is not being
//  converted, it is always sampled."
//  Exception:  SAMC is a minimum sample time.  If it is not met when
//  a trigger arrives, the ADC will wait until it is met.
//  See FRM Sec 22.4.4.2.1-2

//ADC1CFG and ADC4CFG need to be written with DEVADC1, DEVADC4 prior to turn on?
//ADCFLTRx?  May use in averaging or oversampling mode
//Could you use the AFRDY (or the standard RDY signal) to trigger a new conversion?
//  and effectively create a continual measurement scheme?
//  (might need to go through a CPU interrupt)
//Use ADCDATA1 and ADCDATA4 to access 46 and 49 (merely alternates)

ADCTRG1bits.TRGSRC1 = 0b00101; //AN1(=467) using TMR1 match as trigger source
ADCTRG1bits.TRGSRC2 = 0b00101; //AN2(=477) using TMR1 match as trigger source
ADCTRG1bits.TRGSRC3 = 0b00101; //AN3(=467) using TMR1 match as trigger source
ADCTRG2bits.TRGSRC4 = 0b00101; //AN4(=497) using TMR1 match as trigger source

ADCFSTAT = 0;  //FIFO and all associated properties are disabled


/*****************************************************************/
T1CONbits.SIDL = 0;  //Run T1 even in idle mode
T1CONbits.TCS = 0;  //Run on peripheral clock PBCLK 3 (instead of external clock)
T1CONbits.TCKPS = 0b00; //1:1 prescaling
T1CONbits.TGATE = 0;
T1CONbits.TSYNC = 0;
PR1 = 100;  //Set this value to the number of counts.
        //Note that the ADC takes (SAMC+24)*Tad for sample/convert
        //Tad = 2*TQ based on ADCXTIME.ADCDIV
        //TQ = TCLK based on ADCCON3.CONCLKDIV
        //And lastly TCLK = PRC or System Clock based on ADCCON3.ADCSEL
        //So need roughly 30 Tad counts, or 60 TQ = 60 TCLK
        //So maybe let PBCLK3 count to 100

T1CONbits.ON = 1;  //Turn timer on

//IPC1bits.T1IP = 2;
//IEC0bits.T1IE = 1;


/*****************************************************************/



//Digital and analog can be disabled to conserve power
//Digital starts up quickly and is easily enabled
//Further power saving by shutting down analog biasing, but takes time to start up again
//I assume the default is off, but better set them that way anyway
ADCANCONbits.ANEN0 = 0;
ADCANCONbits.ANEN1 = 1;  //ADC1 analog and bias circuitry enabled
ADCANCONbits.ANEN2 = 1;  //ADC2 analog and bias circuitry enabled
ADCANCONbits.ANEN3 = 1;  //ADC3 analog and bias circuitry enabled
ADCANCONbits.ANEN4 = 1;  //ADC4 analog and bias circuitry enabled
ADCANCONbits.ANEN7 = 0;

ADCCON3bits.DIGEN0 = 0;
ADCCON3bits.DIGEN1 = 1;  //ADC1 is digitally enabled (for vout_low)
ADCCON3bits.DIGEN2 = 1;  //ADC2 analog and bias circuitry enabled
ADCCON3bits.DIGEN3 = 1;  //ADC3 analog and bias circuitry enabled
ADCCON3bits.DIGEN4 = 1;  //ADC4 is digitally enabled (for vin_low)
ADCCON3bits.DIGEN7 = 0;


ADCCON1bits.ON = 1;  //Turn on last thing


}

void SetupSPIa1(void) {

CSa1_set:   //Idle High

SPI4CONbits.MSSEN = 0;  //Manually do slave select
SPI4CONbits.MCLKSEL = 0; //Use PBCLK2 as CLK (as opposed to REFCLKO1)
SPI4CONbits.SIDL = 0; //Continue in idle mode


SPI4CONbits.MODE32 = 0; //These three bits define 8-bit communication
SPI4CONbits.MODE16 = 0;
SPI4CON2bits.AUDEN = 0; //AUDEN is for audio codecs

SPI4CONbits.CKE = 1;  //Change data on active->idle transition
SPI4CONbits.CKP = 0;  //Clock idle is low

SPI4CONbits.SSEN = 0; //SSx pin is unused (will manually do chip select)
SPI4CONbits.MSTEN = 1; //Master mode
SPI4CONbits.DISSDI = 1; //SDI pin is unused (never expecting to receive)

SPI4BRG = 0;  //Baud rate divisor (Fsck = Fpb/(2*(BRG+1))
        //Fastest possible is Fpb/2 (=32 MHz in this case)

SPI4CONbits.ON = 1;  //Turn on unit last thing


}

void SetupSPIa2(void) {

CSa2_set:   //Idle High

SPI3CONbits.MSSEN = 0;  //Manually do slave select
SPI3CONbits.MCLKSEL = 0; //Use PBCLK2 as CLK (as opposed to REFCLKO1)
SPI3CONbits.SIDL = 0; //Continue in idle mode


SPI3CONbits.MODE32 = 0; //These three bits define 8-bit communication
SPI3CONbits.MODE16 = 0;
SPI3CON2bits.AUDEN = 0; //AUDEN is for audio codecs

SPI3CONbits.CKE = 1;  //Change data on active->idle transition
SPI3CONbits.CKP = 0;  //Clock idle is low

SPI3CONbits.SSEN = 0; //SSx pin is unused (will manually do chip select)
SPI3CONbits.MSTEN = 1; //Master mode
SPI3CONbits.DISSDI = 1; //SDI pin is unused (never expecting to receive)

SPI3BRG = 0;  //Baud rate divisor (Fsck = Fpb/(2*(BRG+1))
        //Fastest possible is Fpb/2 (=32 MHz in this case)

SPI3CONbits.ON = 1;  //Turn on unit last thing


}

void SetupSPIb1(void) {

CSb1_set:   //Idle High

SPI1CONbits.MSSEN = 0;  //Manually do slave select
SPI1CONbits.MCLKSEL = 0; //Use PBLK2 as CLK (as opposed to REFCLKO1)
SPI1CONbits.SIDL = 0; //Continue in idle mode

SPI1CONbits.MODE32 = 0; //These three bits define 8-bit communication
SPI1CONbits.MODE16 = 0;
SPI1CON2bits.AUDEN = 0; //AUDEN is for audio codecs

SPI1CONbits.CKE = 1;  //Change data on active->idle transition
SPI1CONbits.CKP = 0;  //Clock idle is low

SPI1CONbits.SSEN = 0; //SSx pin is unused (will manually do chip select)
SPI1CONbits.MSTEN = 1; //Master mode
SPI1CONbits.DISSDI = 1; //SDI pin is unused (never expecting to receive)
```

```
1528
1529    SPI1BRG = 0;  //Baud rate divisor (Fsck = Fpb/(2*(BRG+1))
1530            //Fastest possible is Fpb/2 (=32 MHz in this case)
1531
1532    SPI1CONbits.ON = 1;  //Turn on unit last thing
1533
1534
1535 }
1536
1537 void SetupSPIb2(void) {
1538
1539    CSb2_set;    //Idle High
1540
1541    SPI2CONbits.MSSEN = 0;  //Manually do slave select
1542    SPI2CONbits.MCLKSEL = 0; //Use PBLK2 as CLK (as opposed to REFCLKO1)
1543    SPI2CONbits.SIDL = 0; //Continue in idle mode
1544
1545    SPI2CONbits.MODE32 = 0; //These three bits define 8-bit communication
1546    SPI2CONbits.MODE16 = 0;
1547    SPI2CON2bits.AUDEN = 0; //AUDEN is for audio codecs
1548
1549    SPI2CONbits.CKE = 1;  //Change data on active->idle transition
1550    SPI2CONbits.CKP = 0;  //Clock idle is low
1551
1552    SPI2CONbits.SSEN = 0; //SSx pin is unused (will manually do chip select)
1553    SPI2CONbits.MSTEN = 1; //Master mode
1554    SPI2CONbits.DISSDI = 1; //SDI pin is unused (never expecting to receive)
1555
1556    SPI2BRG = 0;  //Baud rate divisor (Fsck = Fpb/(2*(BRG+1))
1557            //Fastest possible is Fpb/2 (=32 MHz in this case)
1558
1559    SPI2CONbits.ON = 1;  //Turn on unit last thing
1560
1561
1562 }
1563
1564 void SetupTimer(void) {
1565    //This is the main timer which sets the time between DAC updates
1566    //Not the same as Timer1 which is used to trigger the ADC (much faster)
1567
1568    T2CONbits.SIDL = 0;  //Run T1 even in idle mode
1569    T2CONbits.TCS = 0;  //Run on peripheral clock PBCLK 3 (instead of extrnal clock)
1570    T2CONbits.TCKPS = 0; //1:1 prescaling
1571    T2CONbits.T32 = 0;  //Each timer is separate 16 bit timer, not combo 32 bit
1572    PR2 = 2048;  //Set this value to the number of counts.
1573    //This is nominally the same as from the PIC24 code which used
1574    //0b0000010000000000
1575    //For PBCLK3 = SYSCLK = 64 MHz, 2048/64 MHz = 32 us
1576    //Which is anywhere between 32 and 96 switching periods worth
1577
1578
1579    T2CONbits.ON = 1;  //Turn timer on
1580
1581    IPC2bits.T2IP = 2;
1582    IEC0bits.T2IE = 1;
1583
1584
1585 }
1586
1587 void SetupUART(void) {
1588
1589    U4MODEbits.STSEL = 0;       //1 stop bit
1590    U4MODEbits.PDSEL = 0b00;    //8-bit, no parity
1591    U4MODEbits.BRGH = 0;        //High speed mode
1592    U4MODEbits.UEN = 0b00;      //Use TX and RX but not RTS/CTS/BCLK
1593    U4MODEbits.RXINV = 0;       //Idle state is high
1594
1595
1596    U4BRG = 72;  //Based on PBCLK2 = SYSCLK (= 64 MHz)
1597            //See FRM 21.3 for tables
1598            //72 yields baud rate of 56000
1599
1600    U4MODEbits.ON = 1;
1601    U4STAbits.UTXEN = 1;
1602    U4STAbits.URXEN = 1;
1603    U4STAbits.URXISEL = 0;  //Interrupt asserted while buffer has any characters
1604
1605
1606
1607    IPS5bits.U4RXIP = 0;
1608    IPS5bits.U4TXIP = 0;
1609    IEC5bits.U4RXIE = 1;
1610    IEC5bits.U4TXIE = 0;
1611    IPC42bits.U4RXIP = 0b010;
1612
1613
1614 }
1615
1616 void StartupBoost(void){
1617    int i;
1618
1619    ENABLEb2_set;
1620    MANUALb2_clr;
1621    //MANUALa1_clr; //Make sure ManualON A is off.  Commented out 10-14-18
1622    MANUALb1_clr;  //Make sure ManualON B is off (doesn't actually turn switch off)
1623
1624
1625    ENABLEb1_set;
1626    ENABLEa1_set;
1627
1628
1629    MANUALb1_set;  //Turn on B
1630    MANUALa1_set;  //Turn on A -- this opens a path through the inductor
1631    for(i=0; i<6; i++) {Nop();}  //Changed from 3 on 10-14-18
1632
1633    MANUALb1_clr;  //Remove Manual B
1634
1635    ENABLEb1_clr;  //Quickly disable B before reset action
1636 }
1637
1638 void StartupBuck(void){
1639    int i;
1640
1641    MANUALa1_clr; //Make sure ManualON A is off.  Commented out 10-14-18
1642    MANUALa2_clr;
1643    MANUALb1_clr;  //Make sure ManualON B is off (doesn't actually turn switch off)
1644    MANUALb2_clr;  ////////////////*****************************************
1645
1646    ENABLEb1_set;
1647    ENABLEa1_set;
1648    ENABLEb2_set; ////////////////*****************************************
1649
1650    //MANUALb1_set;  //Turn on B
1651    MANUALa1_set;  //Turn on A -- this opens a path through the inductor
1652    MANUALb2_set;  ////////////////*****************************************
1653    for(i=0; i<7; i++) {Nop();}  //Changed from 3 on 10-14-18
1654    MANUALa1_clr;  //
1655
1656    ENABLEa1_clr;  //Quickly disable B before reset action
1657 }
1658
1659 void StartupCycle(void){
1660
1661    int i;
1662
1663    MANUALb2_clr;
1664
1665    //Make sure both aren't manually controlled
1666    MANUALa1_clr; //Make sure ManualON A is off
1667    MANUALb1_clr;  //Make sure ManualON B is off
1668
1669    //Enable manual control
1670    ENABLEb1_set;
1671    ENABLEa1_set;
1672
1673
1674    MANUALb1_set;  //Turn on B -- approximately nothing should happen
1675    MANUALa1_set;  //Turn on A -- this opens a path through the inductor
1676    //Should wait until current definitely positive sufficient
1677    //Then, Turn off manual, let "ZVS" signal reset
1678    //Then, turn off enable
1679    for(i=0; i<7; i++) {Nop();}  // Was 6 before 10-13-18
1680
1681    MANUALb1_clr;  //Remove Manual B
1682    MANUALa1_clr;  //Remove Manual A
1683
1684    //Nop();
1685
1686    ENABLEb1_clr;  //Quickly disable B before reset action
1687    ENABLEa1_clr;  //Quickly disable A before reset action
1688    //For some reason, disabling A first was too fast and sometimes it turned back on
1689
1690
1691
1692
1693
1694    ////ERASE ALL THIS BEFORE OPERATION
1695
1696
1697
```

```
}

void StartupLong(void){

  int i;

  //Make sure both aren't manually controlled
  MANUALa1_clr; //Make sure ManualON A is off
  MANUALb1_clr; //Make sure ManualON B is off

  //Enable manual control
  ENABLEb1_set;
  ENABLEa1_set;


  MANUALb1_set;  //Turn on B -- approximately nothing should happen
  MANUALa1_set;  //Turn on A -- this opens a path through the inductor
  //Should wait until current definitely positive sufficient
  //Then, Turn off manual, let "ZVS" signal reset
  //Then, turn off enable
  for(i=0; i<16; i++) {Nop();}  // Was 6 before 10-13-18

  MANUALb1_clr;  //Remove Manual B
  MANUALa1_clr;  //Remove Manual A

  //Nop();

  ENABLEb1_clr;  //Quickly disable B before reset action
  ENABLEa1_clr;  //Quickly disable A before reset action
  //For some reason, disabling A first was too fast and sometimes it turned back on




  ////ERASE ALL THIS BEFORE OPERATION



}

void WriteMenu(void) {

  WriteScreen("\n\n\n\n\r***Modified Boost Menu***");
  WriteScreen("\n\ra:IncLimit%\tz:DecLimit%");
  WriteScreen("\n\rs:IncVzvsA\tx:DecVzvsA");

  WriteScreen("\n\rq:ControlOn\ti:ToggleCin");
  WriteScreen("\n\rd:WriteVout\tc:WriteVin");
  WriteScreen("\n\rv:WriteVhtf:WriteVz");
  WriteScreen("\n\rg:IncPf\t\tb:DecP");
  WriteScreen("\n\rh:IncI2\t\tn:DecI2");
  WriteScreen("\n\rr:SyncRect\te:Bridge");
  WriteScreen("\n\ry:Shtdn\t\te:Menu");

  WriteScreen("\n\n\r");


}

void WriteScreen(char s[50]) {

  char *p;
  p = s;
  while (*p) {
    while (!(U4STAbits.TRMT)) {
    }
    U4TXREG = *p++;
  }

}

void WriteSPIa1(unsigned int* dacinput, char select) {

  int i;
  int mode;
  unsigned int temp;

  switch (select) {
    case 'c':
      mode = 0b0000000000110001; //Control word to select right DAC for VcntrlA
      break;
    case 'z':
      mode = 0b0000000000110000; //Control word to select right DAC for VzvsA
      break;
    default:
      ;
  }



  CSa1_clr;  //Chip select A active low

  //Transmit 8-bit mode word

  SPI4BUF = mode; while(SPI4STATbits.SPITBE == 0){}


  temp = *dacinput;
  temp = temp >> 8;

  //Transmit 8-bit data word

  SPI4BUF = temp; while(SPI4STATbits.SPITBE == 0){}


  temp = (*dacinput) & 0x0011;

  //Transmit 8-bit data word

  SPI4BUF = temp; while(SPI4STATbits.SPITBE == 0){}
  for (i=1; i<10; i++) {Nop();}  //Necessary delay for LTC2602 timing

  CSa1_set;


}

void WriteSPIa2(unsigned int* dacinput, char select) {

  int i;
  int mode;
  unsigned int temp;

  switch (select) {
    case 'c':
      mode = 0b0000000000110001; //Control word to select right DAC for VcntrlA
      break;
    case 'z':
      mode = 0b0000000000110000; //Control word to select right DAC for VzvsA
      break;
    default:
      ;
  }



  CSa2_clr;  //Chip select A active low

  //Transmit 8-bit mode word

  SPI3BUF = mode; while(SPI3STATbits.SPITBE == 0){}


  temp = *dacinput;
  temp = temp >> 8;

  //Transmit 8-bit data word

  SPI3BUF = temp; while(SPI3STATbits.SPITBE == 0){}


  temp = (*dacinput) & 0x0011;

  //Transmit 8-bit data word

  SPI3BUF = temp; while(SPI3STATbits.SPITBE == 0){}
  for (i=1; i<10; i++) {Nop();}  //Necessary delay for LTC2602 timing

  CSa2_set;


}

void WriteSPIb1(unsigned int* dacinput, char select) {

  int i;
  int mode;
  unsigned int temp;
```

```c
    switch (select) {
      case 'c':
        mode = 0b0000000000110001; //Control word to select right DAC for VcntrlB
        break;
      case 'z':
        mode = 0b0000000000110000; //Control word to select right DAC for VzvsB
        break;
      default:
        ;
    }



    CSb1_clr;  //Chip select B active low

    //Transmit 8-bit mode word

    SPI1BUF = mode; while(SPI1STATbits.SPITBE == 0){}


    temp = *dacinput;
    temp = temp >> 8;

    //Transmit 8-bit data word

    SPI1BUF = temp; while(SPI1STATbits.SPITBE == 0){}


    temp = (*dacinput) & 0x0011;

    //Transmit 8-bit data word

    SPI1BUF = temp; while(SPI1STATbits.SPITBE == 0){}
    for (i=1; i<10; i++) {Nop();}  //Necessary delay for LTC2602 timing

    CSb1_set;



}

void WriteSPIb2(unsigned int* dacinput, char select) {

    int i;
    int mode;
    unsigned int temp;

    switch (select) {
      case 'c':
        mode = 0b0000000000110001; //Control word to select right DAC for VcntrlB
        break;
      case 'z':
        mode = 0b0000000000110000; //Control word to select right DAC for VzvsB
        break;
      default:
        ;
    }



    CSb2_clr;  //Chip select B active low

    //Transmit 8-bit mode word

    SPI2BUF = mode; while(SPI2STATbits.SPITBE == 0){}


    temp = *dacinput;
    temp = temp >> 8;

    //Transmit 8-bit data word

    SPI2BUF = temp; while(SPI2STATbits.SPITBE == 0){}


    temp = (*dacinput) & 0x0011;

    //Transmit 8-bit data word

    SPI2BUF = temp; while(SPI2STATbits.SPITBE == 0){}
    for (i=1; i<10; i++) {Nop();}  //Necessary delay for LTC2602 timing

    CSb2_set;



}

void WriteVin(void) {


    unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    float ADCValF;
    //ADCVal = ADCDATA4;
    ADCValF = (double) (ADCDATA4>>16) ;
    //ADCValF = ADCValF / 1.127;
    ADCValF = ADCValF*(3.3 * 134.3 / 65536);
    //ADC Values are saved as
    // dddd dddd dddd 0000 0000 0000 0000 0000
    //So divide by 2^32 to get fraction of full voltage which is 3.3


    char buf[16];
    sprintf(buf, "%f", ADCValF);
    WriteScreen("\n\rVin  = ");
    WriteScreen(buf);

}

void WriteVout(void) {

    unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    float ADCValF;
    //ADCVal = ADCDATA1;
    ADCValF = (double) (ADCDATA3>>16);  //Convert 32 bit to 16 bit
    //ADCValF = ADCValF / 1.127;
    ADCValF = ADCValF*(3.3 * 134.3 / 65536);


    //ADCValF = 3.3 * ADCValF / (0x100000000);
    char buf[16];
    sprintf(buf, "%f", ADCValF);
    WriteScreen("\n\rVout  = ");
    WriteScreen(buf);


}


void WriteVh(void) {


    unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    float ADCValF;
    //ADCVal = ADCDATA4;
    ADCValF = (double) (ADCDATA1>>16) ;
    //ADCValF = ADCValF / 1.127;
    ADCValF = ADCValF*(3.3 * 134.3 / 65536);
    //ADC Values are saved as
    // dddd dddd dddd 0000 0000 0000 0000 0000
    //So divide by 2^32 to get fraction of full voltage which is 3.3


    char buf[16];
    sprintf(buf, "%f", ADCValF);
    WriteScreen("\n\rVh  = ");
    WriteScreen(buf);

}

void WriteVn(void) {


    unsigned int ADCVal;   //int is 32 bit in XC32 compiler;  short's are 16 bit
    float ADCValF;
    //ADCVal = ADCDATA4;
    ADCValF = (double) (ADCDATA2>>16) ;
    //ADCValF = ADCValF / 1.127;
    ADCValF = ADCValF*(3.3 * 134.3 / 65536);
    //ADC Values are saved as
    // dddd dddd dddd 0000 0000 0000 0000 0000
    //So divide by 2^32 to get fraction of full voltage which is 3.3


    char buf[16];
    sprintf(buf, "%f", ADCValF);
    WriteScreen("\n\rVn  = ");
    WriteScreen(buf);


}


double Compute11(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double I2, double limitpercent, double I3, double I5) {

```

```
    double x = Vin/Vout;
    double D = (3.1415*Vout/(L*i2)) * sqrt(L*Cres/2);  //Used to have Cres/2

    double iin = (Vin/R) * (  1 + limitpercent*(3*i3+5*i5)*Vrms - limitpercent*(4*i3+20*i5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*i5)*Vin*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  );
    double iconv = iin + cap_sign*C_in*w_line*sqrt(2*Vrms*Vrms-Vin*Vin);
    double i1 = iconv + sqrt(iconv*iconv + i2*i2*x - 2*iconv*i2*x*x + 2*iconv*i2*D*x*(1-x));

    return(i1);
}



double Computei1buck(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double i2, double limitpercent, double i3, double i5) {


    double x = Vin/Vout;

    //double D = (3.1415*Vout/(L*i2)) * sqrt(L*Cres/2);  //Used to have Cres/2
    double w_res = 1/sqrt(Cres*L);

    double i0 = -Cres*Vout*w_res*sqrt(x*(2-x));
    double t0 = (1/w_res)*( 3.1415/2 + (x-1) + (x-1)*(x-1)*(x-1)/6 ); // good approximation to arccos as long as x doesn't get too close to 2

    double iin = (Vin/R) * (  1 + limitpercent*(3*i3+5*i5)*Vrms - limitpercent*(4*i3+20*i5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*i5)*Vin*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  );

    //if(cap_sign == 1) {Vin = Vin - 10;}
    //This is a correction for late measurements and steep slopes.
    //Compute large iin from large Vin, then compute longer times with smaller Vin


    double iconv = iin + cap_sign*C_in*w_line*sqrt(2*Vrms*Vrms-Vin*Vin);

    double i1 = iconv*x + sqrt(iconv*iconv*x*x - 2*iconv*i0 + i0*i0 + 2*iconv*t0*Vout*(x-1)/L);

    return(i1);
}


double Computetbuck(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double i2, double limitpercent, double i3, double i5) {

    double x = Vin/Vout;
    //double D = (3.1415*Vout/(L*i2)) * sqrt(L*Cres/2);  //Used to have Cres/2
    double w_res = 1/sqrt(Cres*L);

    double i0 = -Cres*Vout*w_res*sqrt(x*(2-x));
    double t0 = (1/w_res)*( 3.1415/2 + (x-1) + (x-1)*(x-1)*(x-1)/6 ); // good approximation to arccos as long as x doesn't get too close to 2

    double iin = (Vin/R) * (  1 + limitpercent*(3*i3+5*i5)*Vrms - limitpercent*(4*i3+20*i5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*i5)*Vin*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  );
    double iconv = iin + cap_sign*C_in*w_line*sqrt(2*Vrms*Vrms-Vin*Vin);

    double i1 = iconv*x + sqrt(iconv*iconv*x*x - 2*iconv*i0 + i0*i0 + 2*iconv*t0*Vout*(x-1)/L);

    double tbuck = (i1-i0)*L/(Vin-Vout);

    return(tbuck);
}

double Computetbtran(double Vin, double Vout, double Vrms, double R, double L, double Cres, double cap_sign, double C_in, double w_line, double i2, double limitpercent, double i3, double i5, double ta) {

    double x = Vin/Vout;
    double D = (3.1415*Vout/(L*i2)) * sqrt(L*Cres/2);  //Used to have Cres/2
    double tres = 3.1415*sqrt(L*Cres/2);

    double iin = (Vin/R) * (  1 + limitpercent*(3*i3+5*i5)*Vrms - limitpercent*(4*i3+20*i5)*Vin*Vin*0.5/Vrms   + limitpercent*(16*i5)*Vin*Vin*Vin*Vin/(4*Vrms*Vrms*Vrms)  );
    double iconv = iin + cap_sign*C_in*w_line*sqrt(2*Vrms*Vrms-Vin*Vin);
    //double i1 = iconv + sqrt(iconv*iconv + i2*i2*x - 2*iconv*i2*x*x + 2*iconv*i2*D*x*(1-x));
    double tbtran = (ta - iconv*L/Vout) - sqrt( iconv*iconv*L*L/(Vout*Vout) + ta*ta - 4*iconv*L*ta/Vout - 2*iconv*L*tres/Vout  );


    return(tbtran);
}

//After checking all functionality, implement SartupCycle and StartupBoost
```

Bibliography

[1] A. J. Hanson, J. A. Belk, S. Lim, C. R. Sullivan, and D. J. Perreault, "Measurements and performance factor comparisons of magnetic materials at high frequency," in *IEEE Energy Conversion Congress and Exposition (ECCE)*. IEEE, 2015.

[2] ——, "Measurements and performance factor comparisons of magnetic materials at high frequency," *IEEE Transactions on Power Electronics*, vol. 31, no. 11, pp. 7909–7925, Nov 2016.

[3] A. Hanson, "Enabling HF Power Conversion: Magnetic Components and a Wide Voltage Range Converter," Master's thesis, Massachusetts Institute of Technology, Cambridge, MA, 2016.

[4] R. S. Yang, A. J. Hanson, D. J. Perreault, and C. R. Sullivan, "A low-loss inductor structure and design guidelines for high-frequency applications," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2018, pp. 579–586.

[5] R. S. Yang, A. J. Hanson, B. A. Reese, C. R. Sullivan, and D. J. Perreault, "A low-loss inductor structure and design guidelines for high-frequency applications," *IEEE Transactions on Power Electronics*, pp. 1–1, 2019.

[6] *AC Current Probes CT1, CT2, CT6 Datasheet*, Tektronix, 12 2013. [Online]. Available: https://www.tek.com/link-click-count?nid=73041&url=https%3A//download.tek.com/datasheet/AC_Current_Probes.pdf

[7] "About efficient power conversion corporation (epc)," https://epc-co.com/epc/AboutEPC.aspx, accessed: 2019-04-15.

[8] M. Chen, K. K. Afridi, S. Chakraborty, and D. J. Perreault, "Multitrack power conversion architecture," *IEEE Transactions on Power Electronics*, vol. 32, no. 1, pp. 325–340, Jan 2017.

[9] J. Santiago, D. Otten, and D. J. Perreault, "Single phase universal input pfc converter operated at hf," in *IEEE Applied Power Electronics Converence (APEC)*, 2018.

[10] W. Inam, K. K. Afridi, and D. J. Perreault, "Variable frequency multiplier technique for high-efficiency conversion over a wide operating range," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 4, no. 2, pp. 335–343, June 2016.

[11] Y. . Wu, J. Gritters, L. Shen, R. P. Smith, J. McKay, R. Barr, and R. Birkhahn, "Performance and robustness of first generation 600-v gan-on-si power transistors," in *The 1st IEEE Workshop on Wide Bandgap Power Devices and Applications*, Oct 2013, pp. 6–10.

[12] G. Zulauf, S. Park, W. Liang, K. Surakitbovorn, and J. M. R. Davila, "C_{OSS} losses in 600 v gan power semiconductors in soft-switched, high- and very-high-frequency power converters," *IEEE Transactions on Power Electronics*, vol. PP, no. 99, pp. 1–1, 2018.

[13] B. Galapon, A. J. Hanson, and D. J. Perreault, "Measuring dynamic on resistance in gan transistors at mhz frequencies," in *2018 IEEE 19th Workshop on Control and Modeling for Power Electronics (COMPEL)*, June 2018, pp. 1–8.

[14] A. F. Martin, A. J. Hanson, and D. J. Perreault, "Energy and size reduction of grid-interfaced energy buffers through line waveform control," in *2018 IEEE 19th Workshop on Control and Modeling for Power Electronics (COMPEL)*, June 2018, pp. 1–8.

[15] A. J. Hanson, A. F. Martin, and D. J. Perreault, "Energy and size reduction of grid-interfaced energy buffers through line waveform control," *IEEE Transactions on Power Electronics*, pp. 1–1, 2019.

[16] A. J. Hanson and D. J. Perreault, "A high frequency power factor correction converter with soft switching," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2018, pp. 2027–2034.

[17] ——, "A high frequency power factor correction converter with soft switching," in *IEEE Applied Power Electronics Conference (APEC)*, 2018.

[18] A. Hanson, J. Belk, S. Lim, C. Sullivan, and D. Perreault, "Measurements and performance factor comparisons of magnetic materials at high frequency," *IEEE Transactions on Power Electronics*, vol. PP, no. 99, pp. 1–1, 2016.

[19] C. R. Sullivan, B. A. Reese, A. L. F. Stein, and P. A. Kyaw, "On size and magnetics: Why small efficient power inductors are rare," in *2016 International Symposium on 3D Power Electronics Integration and Manufacturing (3D-PEIM)*, June 2016, pp. 1–23.

[20] C. R. Sullivan, "Prospects for advances in power magnetics," in *CIPS 2016; 9th International Conference on Integrated Power Electronics Systems*, March 2016, pp. 1–9.

[21] P. A. Kyaw, A. L. F. Stein, and C. R. Sullivan, "High-q resonator with integrated capacitance for resonant power conversion," in *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2017, pp. 2519–2526.

[22] A. L. F. Stein, P. A. Kyaw, and C. R. Sullivan, "High-q self-resonant structure for wireless power transfer," in *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2017, pp. 3723–3729.

[23] C. R. Sullivan, "Aluminum windings and other strategies for high-frequency magnetics design in an era of high copper and energy costs," in *APEC 07 - Twenty-Second Annual IEEE Applied Power Electronics Conference and Exposition*, Feb 2007, pp. 78–84.

[24] J. Qiu, A. J. Hanson, and C. R. Sullivan, "Design of toroidal inductors with multiple parallel foil windings," in *14th Workshop on Control and Modeling for Power Electronics (COMPEL)*. IEEE, 2013, pp. 23–26.

[25] Y. Han and D. J. Perreault, "Inductor design methods with low-permeability rf core materials," *IEEE Transactions on Industry Applications*, vol. 48, no. 5, pp. 1616–1627, Sept 2012.

[26] J. Hu and C. R. Sullivan, "The quasi-distributed gap technique for planar inductors: design guidelines," in *Industry Applications Conference, 1997. Thirty-Second IAS Annual Meeting, IAS '97., Conference Record of the 1997 IEEE*, vol. 2, Oct 1997, pp. 1147–1152 vol.2.

[27] "Distributed air gaps in ferrite cores," EPCOS AG – a TDK Group Company, Tech. Rep., June 2017. [Online]. Available: https://de.tdk.eu/download/2113422/321697054fce0c768ea66959fde3b3db/ferrites-air-gaps-pb.pdf

[28] P. L. Dowell, "Effects of eddy currents in transformer windings," *Electrical Engineers, Proceedings of the Institution of*, vol. 113, no. 8, pp. 1387–1394, August 1966.

[29] C. R. Sullivan, "Optimal choice for number of strands in a litz-wire transformer winding," *IEEE Transactions on Power Electronics*, vol. 14, no. 2, pp. 283–291, Mar 1999.

[30] W. G. Hurley, E. Gath, and J. G. Breslin, "Optimizing the ac resistance of multilayer transformer windings with arbitrary current waveforms," *IEEE Transactions on Power Electronics*, vol. 15, no. 2, pp. 369–376, Mar 2000.

[31] T. Delaforge, H. Chazal, J. L. Schanen, and R. J. Pasterczyk, "Increasing windings efficiency at high frequencies: Hollow conductors and clad metal conductors formal solution based on the magnetic potential," in *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Sept 2015, pp. 5689–5695.

[32] X. Nan and C. R. Sullivan, "An improved calculation of proximity-effect loss in high-frequency windings of round conductors," in *Power Electronics Specialist Conference, 2003. PESC '03. 2003 IEEE 34th Annual*, vol. 2, June 2003, pp. 853–860 vol.2.

[33] M. Chen, M. Araghchini, K. K. Afridi, J. H. Lang, C. R. Sullivan, and D. J. Perreault, "A systematic approach to modeling impedances and current distribution in planar magnetics," *IEEE Transactions on Power Electronics*, vol. 31, no. 1, pp. 560–580, Jan 2016.

[34] K. Venkatachalam, C. R. Sullivan, T. Abdallah, and H. Tacca, "Accurate prediction of ferrite core loss with nonsinusoidal waveforms using only steinmetz parameters," in *2002 IEEE Workshop on Computers in Power Electronics, 2002. Proceedings.*, June 2002, pp. 36–41.

[35] J. Muhlethaler, J. Biela, J. W. Kolar, and A. Ecklebe, "Improved core-loss calculation for magnetic components employed in power electronic systems," *IEEE Transactions on Power Electronics*, vol. 27, no. 2, pp. 964–973, Feb 2012.

[36] C. Fei, F. C. Lee, and Q. Li, "High-efficiency high-power-density llc converter with an integrated planar matrix transformer for high output current applications," *IEEE Transactions on Industrial Electronics*, vol. PP, no. 99, pp. 2428–2435, 2017.

[37] R. Yang, "Low-loss inductor design for high-frequency power applications," Master's thesis, Massachusetts Institute of Technology, 2019.

[38] T. H. Lee, *Planar Microwave Engineering: A Practical Guide to Theory, Measurement, and Circuits*, 1st ed. Cambridge University Press, 2004, pp. 140–142.

[39] F. E. Terman, *Radio Engineers' Handbook*, 1st ed. McGraw-Hill Book Company, Inc., 1943, pp. 52–55.

[40] V. Valchev and A. Van den Bossche, *Inductors and Transformers for Power Electronics.* CRC Press, 2005.

[41] J. W. Kolar, F. Krismer, M. Leibl, D. Neumayr, L. Schrittwieser, and D. Bortis, Impact of Magnetics on Power Electronics Converter Performance – State-of-the-Art and Future Prospects, Keynote Presentation at the PSMA Workshop on "Power Magnetics @ High Frequency – Transforming the Black Magic to Engineering", Tampa, FL, USA, March 25, 2017.

[42] C. R. Sullivan and R. Y. Zhang, "Simplified design method for litz wire," in *IEEE Applied Power Electronics Conference and Exposition (APEC).* IEEE, 2014, pp. 2667–2674.

[43] B. A. Reese, R. Joseph, and C. R. Sullivan, "Improved litz wire designs for the MHz range," in *2018 IEEE 19th Workshop on Control and Modeling for Power Electronics (COMPEL)*, June 2018.

[44] J. D. Pollock, T. Abdallah, and C. R. Sullivan, "Easy-to-use CAD tools for litz-wire winding optimization," in *IEEE App. Pow. Electr. Conf.*, 2003.

[45] C. R. Sullivan and R. Y. Zhang, "Analytical model for effects of twisting on litz-wire losses," in *IEEE Workshop on Control and Modeling for Pow. Electr. (COMPEL)*, 2014.

[46] Mingjuan Zhu, D. J. Perreault, V. Caliskan, T. C. Neugebauer, S. Guttowski, and J. G. Kassal, "Design and evaluation of an active ripple filter with rogowski-coil current sensing," in *30th Annual IEEE Power Electronics Specialists Conference. Record. (Cat. No.99CH36321)*, vol. 2, July 1999, pp. 874–880 vol.2.

[47] C. R. Hewson, W. F. Ray, and J. Metcalfe, "Optimising high frequency integrator operation of rogowski current transducers," in *2007 European Conference on Power Electronics and Applications*, Sep. 2007, pp. 1–9.

[48] "EN 61000-3-2:2018 (edition 5.0) electromagnetic compatibility (emc) – part 3-2: Limits – limits for harmonic current emissions (equipment up to and including 16 a per phase)," 2018.

[49] O. Garcia, J. A. Cobos, R. Prieto, P. Alou, and J. Uceda, "Single phase power factor correction: a survey," *IEEE Transactions on Power Electronics*, vol. 18, no. 3, pp. 749–755, May 2003.

[50] B. Whitaker, A. Barkley, Z. Cole, B. Passmore, T. McNutt, and A. B. Lostetter, "High-frequency ac-dc conversion with a silicon carbide power module to achieve high-efficiency and greatly improved power density," in *2013 4th IEEE International Symposium on Power Electronics for Distributed Generation Systems (PEDG)*, July 2013, pp. 1–5.

[51] L. Huber, Y. Jang, and M. M. Jovanovic, "Performance evaluation of bridgeless pfc boost rectifiers," *IEEE Transactions on Power Electronics*, vol. 23, no. 3, pp. 1381–1390, May 2008.

[52] S. Qin, Y. Lei, C. Barth, W. C. Liu, and R. C. N. Pilawa-Podgurski, "A high power density series-stacked energy buffer for power pulsation decoupling in single-phase converters," *IEEE Transactions on Power Electronics*, vol. 32, no. 6, pp. 4905–4924, June 2017.

[53] Y. Sun, Y. Liu, M. Su, W. Xiong, and J. Yang, "Review of active power decoupling topologies in single-phase systems," *IEEE Transactions on Power Electronics*, vol. 31, no. 7, pp. 4778–4794, July 2016.

[54] M. Chen, K. K. Afridi, and D. J. Perreault, "Stacked switched capacitor energy buffer architecture," *IEEE Transactions on Power Electronics*, vol. 28, no. 11, pp. 5183–5195, Nov 2013.

[55] K. K. Afridi, M. Chen, and D. J. Perreault, "Enhanced bipolar stacked switched capacitor energy buffers," *IEEE Transactions on Industry Applications*, vol. 50, no. 2, pp. 1141–1149, March 2014.

[56] S. Li, W. Qi, J. Wu, S. C. Tan, and R. Hui, "Minimum active switch requirements for single-phase pfc rectifiers without electrolytic capacitors," *IEEE Transactions on Power Electronics*, pp. 1–1, 2018.

[57] S. Li, W. Qi, S. Tan, and S. Y. . Hui, "Integration of an active filter and a single-phase ac/dc converter with reduced capacitance requirement and component count," *IEEE Transactions on Power Electronics*, vol. 31, no. 6, pp. 4121–4137, June 2016.

[58] "Rtca do-160g environmental conditions and test procedures for airborne equipment," 2010.

[59] H. Yuan, S. Li, W. Qi, S. C. Tan, and R. Hui, "On nonlinear control of single-phase converters with active power decoupling function," *IEEE Transactions on Power Electronics*, pp. 1–1, 2018.

[60] L. Gu, X. Ruan, M. Xu, and K. Yao, "Means of eliminating electrolytic capacitor in ac/dc power supplies for led lightings," *IEEE Transactions on Power Electronics*, vol. 24, no. 5, pp. 1399–1408, May 2009.

[61] A. Shagerdmootaab and M. Moallem, "Filter capacitor minimization in a flyback led driver considering input current harmonics and light flicker characteristics," *IEEE Transactions on Power Electronics*, vol. 30, no. 8, pp. 4467–4476, Aug 2015.

[62] X. Ruan, B. Wang, K. Yao, and S. Wang, "Optimum injected current harmonics to minimize peak-to-average ratio of led current for electrolytic capacitor-less ac-dc drivers," *IEEE Transactions on Power Electronics*, vol. 26, no. 7, pp. 1820–1825, July 2011.

[63] B. Wang, X. Ruan, K. Yao, and M. Xu, "A method of reducing the peak-to-average ratio of led current for electrolytic capacitor-less ac-dc drivers," *IEEE Transactions on Power Electronics*, vol. 25, no. 3, pp. 592–601, March 2010.

[64] G. M. Soares, P. S. Almeida, J. M. Alonso, and H. A. C. Braga, "Capacitance minimization in offline led drivers using an active-ripple-compensation technique," *IEEE Transactions on Power Electronics*, vol. 32, no. 4, pp. 3022–3033, April 2017.

[65] M. Nassary, M. Orabi, E. M. Ahmed, E. S. Hasaneen, and M. Gaafar, "Modified harmonic injection technique for electrolytic capacitor-less led driver," in *2017 Nineteenth International Middle East Power Systems Conference (MEPCON)*, Dec 2017, pp. 1459–1464.

[66] Q. Hu and R. Zane, "Minimizing required energy storage in off-line led drivers based on series-input converter modules," *IEEE Transactions on Power Electronics*, vol. 26, no. 10, pp. 2887–2895, Oct 2011.

[67] S. Lim, D. M. Otten, and D. J. Perreault, "New ac-dc power factor correction architecture suitable for high-frequency operation," *IEEE Transactions on Power Electronics*, vol. 31, no. 4, pp. 2937–2949, April 2016.

[68] Y. Tang and F. Blaabjerg, "Power decoupling techniques for single-phase power electronics systems: An overview," in *2015 IEEE Energy Conversion Congress and Exposition (ECCE)*, Sept 2015, pp. 2541–2548.

[69] K. Yao, X. Ruan, X. Mao, and Z. Ye, "Reducing storage capacitor of a dcm boost pfc converter," *IEEE Transactions on Power Electronics*, vol. 27, no. 1, pp. 151–160, Jan 2012.

[70] Y. C. Li, F. C. Lee, Q. Li, X. Huang, and Z. Liu, "A novel ac-to-dc adaptor with ultra-high power density and efficiency," in *2016 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2016, pp. 1853–1860.

[71] D. G. Lamar, J. Sebastian, M. Arias, and A. Fernandez, "On the limit of the output capacitor reduction in power-factor correctors by distorting the line input current," *IEEE Transactions on Power Electronics*, vol. 27, no. 3, pp. 1168–1176, March 2012.

[72] K. Surakitbovorn and J. R. Davila, "Evaluation of gan transistor losses at mhz frequencies in soft switching converters," in *2017 IEEE 18th Workshop on Control and Modeling for Power Electronics (COMPEL)*, July 2017, pp. 1–6.

[73] T. Foulkes, T. Modeer, and R. C. N. Pilawa-Podgurski, "Developing a standardized method for measuring and quantifying dynamic on-state resistance via a survey of low voltage gan hemts," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2018, pp. 2717–2724.

[74] A. J. Hanson and D. J. Perreault, "A high frequency power factor correction converter with soft switching," in *2018 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2018, pp. 2027–2034.

[75] K. Li, P. Evans, and M. Johnson, "Gan-hemt dynamic on-state resistance characterisation and modelling," *Control and Modeling for Power Electronics (COMPEL), 2016 IEEE 17th Workshop*, 27-30 June 2016.

[76] D. Jin and J. A. del Alamo, "Mechanisms responsible for dynamic on-resistance in gan high-voltage hemts," *Proceedings of the 2012 24th International Symposium on Power Semiconductor Devices and ICs*, pp. 333–336, 3-7 June 2012.

[77] P. Wright and M. Thorsell, "A novel technique for gan hemt trap states characterisation," *Compound Semiconductor Integrated Circuit Symposium (CSICS), 2013 IEEE*, 13-16 October 2013.

[78] O. C. Spro, D. Peftitsis, O.-M. Midtgard, and T. Undeland, "Modelling and quantification of power losses due to dynamic on-state resistance of gan e-mode hemt," *Control and Modeling for Power Electronics (COMPEL), 2017 IEEE 18th Workshop*, 9-12 July 2017.

[79] N. Badawi and S. Dieckerhoff, "A new method for dynamic ron extraction of gan power hemts," *PCIM Europe 2015*, pp. 1010–1015, 19-20 May 2015.

[80] N. Badawi, O. Hilt, E. Behat-Treidel, J. Böcker, J. Würfl, and S. Dieckerhoff, "Investigation of the dynamic on-state resistance of 600v normally-off and normally-on gan hemts," *Energy Conversion Congress and Exposition (ECCE), 2015 IEEE*, pp. 913–919, 20-24 September 2015.

[81] B. Lu, T. Palacios, and D. Risbud, "Extraction of dynamic on-resistance in gan transistors: Under soft- and hard-switching conditions," *Compound Semiconductor Integrated Circuit Symposium (CSICS), 2011 IEEE*, pp. 333–336, 16-19 October 2011.

[82] J. Böcker, H. Just, O. Hilt, N. Badawi, J. Würfl, and S. Dieckerhoff, "Experimental analysis and modeling of gan normally-off hfets with trapping effects," *Power Electronics and Applications (EPE'15 ECCE-Europe), 2015 17th European Conference*, 8-10 September 2015.

[83] R. Gelagaev, P. Jacqmaer, and J. Driesen, "A fast voltage clamp circuit for the accurate measurement of the dynamic on-resistance of power transistors," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 2, pp. 1241–1250, 20 August 2014.

[84] L. Roslaniec, A. S. Jurkov, A. A. Bastami, and D. J. Perreault, "Design of single-switch inverters for variable resistance-load modulation operation," *IEEE Transactions on Power Electronics*, vol. 30, no. 6, pp. 3200–3214, June 2015.

[85] M. Madsen, A. Knott, and M. A. E. Andersen, "Low power very high frequency resonant converter with high step down ratio," in *2013 Africon*, Sept 2013, pp. 1–6.

[86] J. M. Burkhart, R. Korsunsky, and D. J. Perreault, "Design methodology for a very high frequency resonant boost converter," *IEEE Transactions on Power Electronics*, vol. 28, no. 4, pp. 1929–1937, April 2013.

[87] A. S. Jurkov, A. Radomski, and D. J. Perreault, "Tunable impedance matching networks based on phase-switched impedance modulation," in *2017 IEEE Energy Conversion Congress and Exposition (ECCE)*, Oct 2017, pp. 947–954.

[88] J. C. Hertel, Y. Nour, and A. Knott, "Integrated very-high-frequency switch mode power supplies: Design considerations," *IEEE Journal of Emerging and Selected Topics in Power Electronics*, vol. 6, no. 2, pp. 526–538, June 2018.

[89] S. Lim, J. Ranson, D. M. Otten, and D. J. Perreault, "Two-stage power conversion architecture suitable for wide range input voltage," *IEEE Transactions on Power Electronics*, vol. 30, no. 2, pp. 805–816, Feb 2015.

[90] *AN-2020 Thermal Design By Insight, Not Hindsight*, Texas Instruments, 4 2013.

[91] D. Jin and J. A. del Alamo, "Methodology for the study of dynamic on-resistance in high-voltage gan field-effect transistors," *IEEE Transactions on Electron Devices*, vol. 60, no. 10, pp. 3190–3196, 2 August 2013.

[92] G. Moschopoulos and P. Jain, "Single-phase single-stage power-factor-corrected converter topologies," *IEEE Transactions on Industrial Electronics*, vol. 52, no. 1, pp. 23–35, Feb 2005.

[93] O. Garcia, J. A. Cobos, R. Prieto, P. Alou, and J. Uceda, "Single phase power factor correction: a survey," *IEEE Transactions on Power Electronics*, vol. 18, no. 3, pp. 749–755, May 2003.

[94] L. Huber, Y. Jang, and M. M. Jovanovic, "Performance Evaluation of Bridgeless PFC Boost Rectifiers," *IEEE Transactions on Power Electronics*, vol. 23, no. 3, pp. 1381–1390, May 2008.

[95] A. J. Hanson, R. S. Yang, S. Lim, and D. J. Perreault, "A soft-switched high frequency converter for wide voltage and power ranges," in *2016 IEEE International Telecommunications Energy Conference (INTELEC)*, Oct 2016, pp. 1–8.

[96] "High performance power factor preregulator," Texas Instruments, 2005. [Online]. Available: http://www.ti.com/lit/ds/symlink/uc2855a.pdf

[97] "Natural interleaved dual-phase transition-mode pfc controller," Texas Instruments, 2008. [Online]. Available: http://www.ti.com/lit/ds/symlink/ucc28060.pdf

[98] M. Scherbaum, M. Reddig, R. Kennel, and M. Schlenk, "An Isolated, bridgeless, quasi-resonant ZVS-switching, buck-boost single-stage AC-DC converter with power factor correction (PFC)," in *2017 IEEE Applied Power Electronics Conference and Exposition (APEC)*, March 2017, pp. 74–81.

[99] Z. Liu, X. Huang, M. Mu, Y. Yang, F. C. Lee, and Q. Li, "Design and evaluation of GaN-based dual-phase interleaved MHz critical mode PFC converter," in *2014 IEEE Energy Conversion Congress and Exposition (ECCE)*, Sept 2014, pp. 611–616.

[100] S. Lim, D. M. Otten, and D. J. Perreault, "New AC-DC Power Factor Correction Architecture Suitable for High-Frequency Operation," *IEEE Transactions on Power Electronics*, vol. 31, no. 4, pp. 2937–2949, April 2016.

[101] M. Chen, K. Afridi, S. Chakraborty, and D. Perreault, "MultiTrack Power Conversion Architecture," *IEEE Transactions on Power Electronics*, vol. PP, no. 99, pp. 1–1, 2016.

[102] O. Knecht, D. Bortis, and J. W. Kolar, "ZVS Modulation Scheme for Reduced Complexity Clamp-Switch TCM DC-DC Boost Converter," *IEEE Transactions on Power Electronics*, vol. PP, no. 99, pp. 1–1, 2017.

[103] M. Marvi and A. Fotowat-Ahmady, "A Fully ZVS Critical Conduction Mode Boost PFC," *IEEE Transactions on Power Electronics*, vol. 27, no. 4, pp. 1958–1965, April 2012.

[104] B. Su, J. Zhang, and Z. Lu, "Totem-Pole Boost Bridgeless PFC Rectifier With Simple Zero-Current Detection and Full-Range ZVS Operating at the Boundary of DCM/CCM," *IEEE Transactions on Power Electronics*, vol. 26, no. 2, pp. 427–435, Feb 2011.

[105] "Solution for designing a transition mode pfc preregulator with the l6562a," ST Microelectronics, Tech. Rep., 2009.

[106] A. J. Hanson, J. A. Belk, S. Lim, C. R. Sullivan, and D. J. Perreault, "Measurements and Performance Factor Comparisons of Magnetic Materials at High Frequency," *IEEE Transactions on Power Electronics*, vol. 31, no. 11, pp. 7909–7925, Nov 2016.

[107] M. C. Ghanem, K. Al-Haddad, and G. Roy, "A new single phase buck-boost converter with unity power factor," in *Conference Record of the 1993 IEEE Industry Applications Conference Twenty-Eighth IAS Annual Meeting*, Oct 1993, pp. 785–792 vol.2.

[108] U. Anwar, D. Maksimovic, and K. Afridi, "A Simple Control Architecture for Four-Switch Buck-Boost Converter Based Power Factor Correction Rectifier," in *2017 IEEE Control and Modeling of Power Electronics (COMPEL) Workshop*, July 2017, pp. 1–6.

[109] A. Murthy and M. Badaway, "State Space Averaging Model of a Dual Stage Converter in Discontinuous Conduction Mode," in *2017 IEEE Control and Modeling of Power Electronics (COMPEL) Workshop*, July 2017.

[110] R. S. Yang, A. J. Hanson, C. R. Sullivan, and D. J. Perreault, "A low-loss inductor structure and design guidelines for high-frequency applications," in *IEEE Applied Power Electronics Converence (APEC)*, 2018.

[111] Y. Han, G. Cheung, A. Li, C. R. Sullivan, and D. J. Perreault, "Evaluation of magnetic materials for very high frequency applications," *IEEE Transactions on Power Electronics*, pp. 425–435, 2008.

[112] "Distributed air gaps in ferrite cores," Vishay, Tech. Rep., June 2018. [Online]. Available: https://www.vishay.com/docs/53048/pprachp.pdf

[113] R. G. Medhurst, "Q of solenoid coils," *Wireless Engineer*, p. 281, 1947.

[114] T. H. Lee, *Planar Microwave Engineering.* Cambridge University Press, 2004, ch. 6. Passive Components.

[115] "Coil32," http://coil32.net/detail/coil32-more.html, 2014.