# INDUCTIVELY-COUPLED POWER TRANSFER FOR ELECTROMECHANICAL SYSTEMS

by

**Deron Keith Jackson**

B.S., University of California at Davis, 1991

S.M., Massachusetts Institute of Technology, 1994

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

DOCTOR OF PHILOSOPHY

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

March 1998



Signature of Author _____
Department of Electrical Engineering and Computer Science
March 10, 1998

Certified by _____
Steven B. Leeb
Associate Professor of Electrical Engineering
Thesis Supervisor

Accepted by _____
Arthur C. Smith
Chairman, Department Committee on Graduate Students

# INDUCTIVELY-COUPLED POWER TRANSFER FOR ELECTROMECHANICAL SYSTEMS

by

**Deron Keith Jackson**

Submitted to the Department of Electrical Engineering and Computer Science
on March 10, 1998 in partial fulfillment of the requirements for the degree of
Doctor of Philosophy in Electrical Engineering

## ABSTRACT

Advances in power electronics and magnetic materials have made it feasible to consider replacing conventional ohmic couplings in electromechanical systems with non-contact, inductive couplings. Energy transfer through an electromagnetic field opens the door to a new range of rotating, sliding, and/or separable couplings. Inductive couplings can offer safe, reliable, and efficient power transmission that is largely immune to the effects of wear and the environment.

This thesis investigates inductive coupling as a general approach to power transfer for electromechanical systems. The focus is on three different applications: single-stage inductively-coupled power transfer, multi-stage inductively-coupled power transfer, and an inductively-coupled polymer-gel actuator. These applications span a spectrum of engineering challenges associated with inductively-coupled electromechanical systems. Power-electronic drives and control strategies are developed, which are appropriate for a range of applications that transfer power through an electromagnetic field. Hardware prototypes are built to demonstrate the systems. Electrical and mechanical issues surrounding the design of these systems are investigated. The coupling systems considered are capable of transferring power from the milliwatt to the kilowatt level, and the control strategies developed can operate reliably and with demonstrable stability in the face of large-signal changes in the operating parameters.

Thesis Supervisor: Steven B. Leeb
Title: Associate Professor of Electrical Engineering

# ACKNOWLEDGEMENTS

I thank my advisor Steven B. Leeb, who over the last three years not only has been an outstanding advisor, but also has become a good friend. He has made himself constantly available, and his energy and excitement are inexhaustible. His help and encouragement have been invaluable. I also thank my two thesis committee readers, James L. Kirtley and Richard D. Thornton, who offered many helpful and useful comments.

I thank AMP, Incorporated, for generously supporting this research, with special recognition to Bill Fritz, Bob Grebe, Howard Pieffer, Joseph Sweeney, Bob Trump, and Jim Wise.

I thank those in the LEES lab who provided assistance and friendship during my years there. Steve Shaw deserves special recognition because he is an outstanding engineer and he has been a constant source of help and amusement. Ahmed Mitwalli, "the gel man," deserves thanks not only for his good cheer, but also for his assistance with parts of this thesis. I also thank some of my other colleagues in the lab who both made thesis work more enjoyable and put up with my many questions, in particular, Steve Nagle, Tim Denison, Dave Perreault, John Ofori, and Judy Cardell.

Dahlene Fusco, Paolo Narvaez, and Chris Laughman assisted with parts of this thesis work. I thank them for their contributions. I would like to acknowledge Aaron Schultz for laying the groundwork for some parts of this thesis and also Brett Miwa and Loveday Mweene whose fine theses provided me with valuables ideas and insight.

I would especially like to thank my parents for their love and constant support. Without them none of this would be possible. But most of all, I offer thanks to my beautiful fiancée Lisa Sereno. She is my partner in everything I do.

# Contents

# List of Figures

# List of Tables

# Chapter 1

# Introduction

An inductive coupling is a device that transfers electrical energy from one subsystem to another through a magnetic field. This energy transfer can be accomplished without physical or ohmic contact between subsystems. The technique has been used for well over a century to transfer power in a variety of commercial products and systems. Commercial products employing inductively-coupled energy transfer range from motors and transformers to electric toothbrushes and *in vivo* medical implants. Inductively-coupled connectors can be found in power systems for aircraft passenger seats, mining equipment, underwater exploration, and electric vehicles.

Specifically, an inductively-coupled connector couples two or more system ports without any conductor-to-conductor (ohmic) contact. The lack of any ohmic connection has advantages in many applications. First, it provides galvanic isolation between the subsystems, and therefore allows the safe interconnection of circuits operating with respect to different ground potentials. Second, provided the magnetic path can be split, the subsystems may be physically separated and reconnected without breaking the electric circuit paths. Also, since no metal contacts are exposed, the power transfer capabilities of inductive couplings are essentially immune to environmental factors, such as dirt, dust, and water. They can also be more reliable, robust, and spark-free than ohmic connectors, and they do not necessarily suffer from mechanical wear or friction.

Recent applications are the result of a rising interest in applying separable, sliding, or rotating inductive couplings to safe, relatively high-power transfer problems. Many of these applications are now possible because the development of power-electronic components and materials has experienced a tremendous amount of growth in the last three

decades [113]. The introduction of power MOSFETs and high-performance magnetic materials has led to remarkable improvements in the achievable switching speed, size, and efficiency of power-electronic converters. These improvements have facilitated the use of separable inductive couplings at power levels far in excess of those used in familiar applications like the electric toothbrush charger.

## 1.1 Thesis Scope

This thesis investigates inductive coupling as a general approach to power transfer for electromechanical systems. The scope encompasses a full range of issues that must be considered when implementing an inductively-coupled system, including:

- Power-electronics design and analysis,
- Digital control and adaptive "self-tuning" control,
- Mechanical "coupling" design and modeling.

The focus is on non-contact separable, sliding, or rotating inductive couplings applicable to such systems. Electrical and mechanical issues surrounding the design of these non-contact couplings are investigated. Suitable power-electronic drives and control strategies are developed. The coupling systems considered are capable of transferring power from the milliwatt to the kilowatt range. The control strategies developed can operate reliably and with demonstrable stability in the face of large-signal changes in the operating parameters (voltage and current) of the coupling.

A growing number of references and applications discuss or use inductive couplings as a means to transfer power. The power-electronic drives and control strategies used in these systems may be quite different depending on the specific application. A goal of this thesis is to shed light on some of the fundamental parameters or desired operating conditions that must be identified in any application of inductively-coupled power transfer with a separable connector. Once basic conditions, such as permissible electrical and mechanical parameters of the separable connector, (e.g., mechanical separation, leakage inductance, and operating frequency) have been determined, intelligent choices for the power train and control strategy can be made.

This thesis focuses on three different applications of inductively-coupled connectors. These are a single-stage inductively-coupled power transfer system in the kilowatt range for electromechanical and servomechanical systems, a multi-stage transfer system for electromechanical systems, and a poorly-coupled system for energizing a linear polymer-gel actuator through induction heating. These example applications span a spectrum of engineering challenges associated with inductive couplings. The designs presented for these inductively-coupled applications are inter-related and expose key system characteristics that must be identified before choices about drive and control strategy can be made intelligently.

## 1.2 Previous Work

Inductive couplings are found in a variety of commercial products and systems. This section highlights a number of products and systems currently in existence or under development. These examples demonstrate the wide range of applications for inductively-coupled power transfer.

Perhaps one of the most familiar applications of inductive coupling is the electric toothbrush. This and similar devices have been widely available since the mid-1970's, and several U.S. patents have been issued [28, 92]. In this application, an inductive coupling recharges batteries within the device. Figure 1.1 (a) illustrates one incarnation of this technology from Optiva. A cup-shaped base unit inductively couples power to batteries within the hand-held toothbrush when it is not in use. Although the power level is low, inductive coupling is perfectly suited to the application. Recharging is accomplished without any exposed metal contacts on the device. Since the toothbrush is necessarily exposed to water, dirt, and grime, a non-contact electrical connection maximizes reliability and operator safety.

Figure 1.1 (b) illustrates another system where the use of an inductive coupling maximizes safety, reliability, and convenience. In [54], Kelly and Owens present a "connectorless" power system for aircraft-passenger seats. The system supplies power to entertainment units located within each seat, each unit consuming approximately 50 W. This application requires that the power connections can be easily separated and reat-

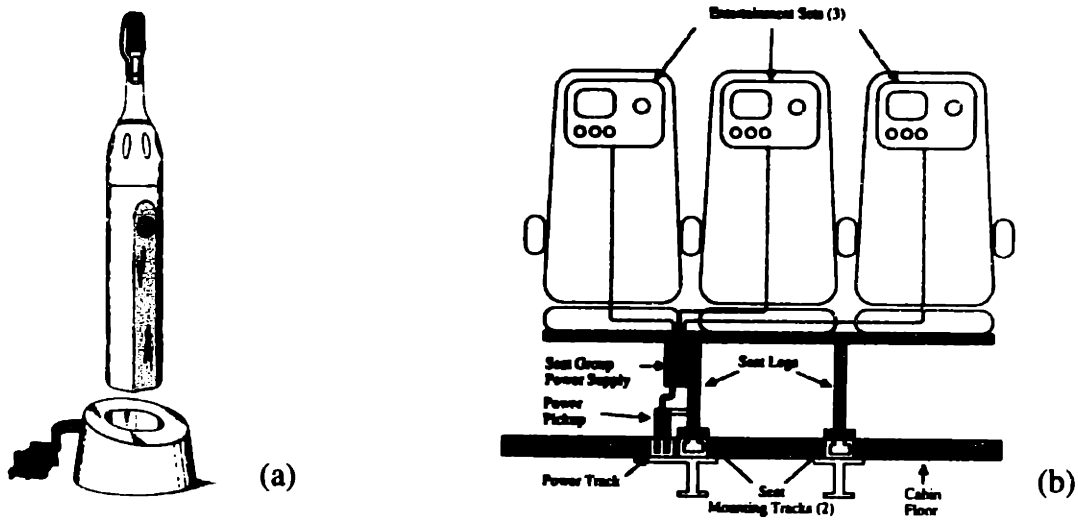



**Figure 1.1:** (a) Inductively-coupled toothbrush [28]. (b) "Connectorless" inductive pickups for powering aircraft entertainment systems [54].

tached because seat groups are frequently moved about the aircraft. The authors describe a flexible system where a sealed power track is incorporated into the floor of the passenger cabin. Multiple seat groups can then be attached using inductively-coupled "power-pickups," which mate to the power track. Power is transferred across the couplings at a frequency of 28 kHz. The system uses a current-fed coupling topology, which is discussed later in Part II of this thesis. The resulting system is flexible, compact, and reliable.

Figure 1.2 illustrates a "sliding" inductively-coupled power system presented in [57]. The authors of [57] describe a system that utilizes a "coaxial-winding transformer" (CWT) to deliver high power to moving loads. Multiple loads can receive power through separate CWTs that slide freely along the length of a single primary conductor. The system is proposed for use with hauling-vehicles, conveyors, and other support equipment in an underground mining operation. (See Figure 1.2.) This application involves up to 10 loads rated at 100 kW each, for a total system rating of 1 MW. Power is supplied by a large-scale current-source inverter running at 2 kHz. A similar "sliding" CWT concept is described in [34] for supplying power to underwater equipment for deep-sea drilling, mining, and exploration. Both applications require the immunity to environmental factors and increased safety offered by inductive coupling. The references clearly illustrate the CWT as a superior "sliding" coupling. The CWT is shown to have low and controlled leakage inductance, which leads to improved performance and efficiency.

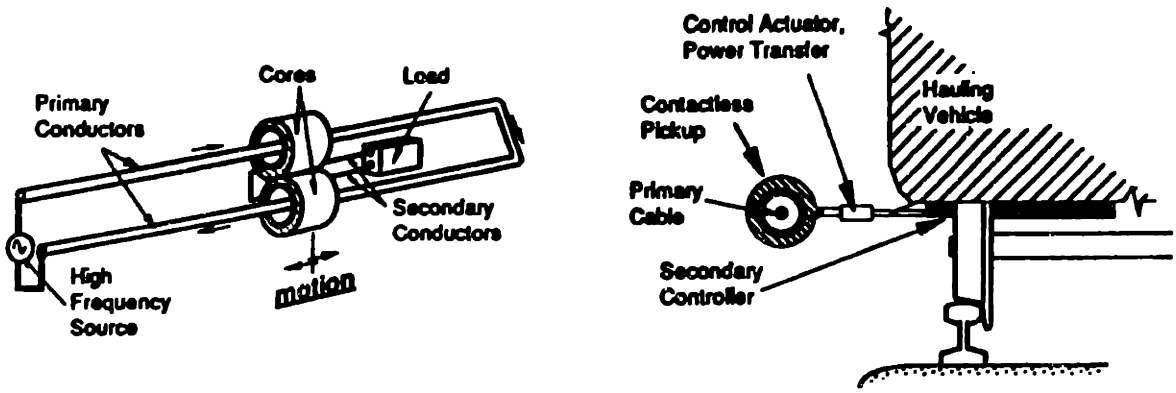



**Figure 1.2:** Inductively-coupled "sliding" power system for mining equipment [57].

Another application for inductive coupling has been found in the automotive industry. Recent environmental mandates in California, New York, and Massachusetts have spurred significant developments in electric-vehicle (EV) technology. Inductively-coupled power transfer has been demonstrated as a viable means for charging EV batteries [23, 58, 59]. In fact, a commercial inductively-coupled EV battery charging system is now in production. The system, named MAGNE-CHARGE™, is sold by Delco specifically for use with General Motor's electric vehicle, the EV1 [15]. Figure 1.3 illustrates the charger in action. A separable inductive coupling transfers power from the charger to the vehicle batteries. Inserting the charging "paddle" into a port on the vehicle is all that is required to initiate charging. Power transfer through the coupling occurs at a variable frequency between 80 kHz and 350 kHz. Since Part I of this thesis specifically covers the design of EV bat-

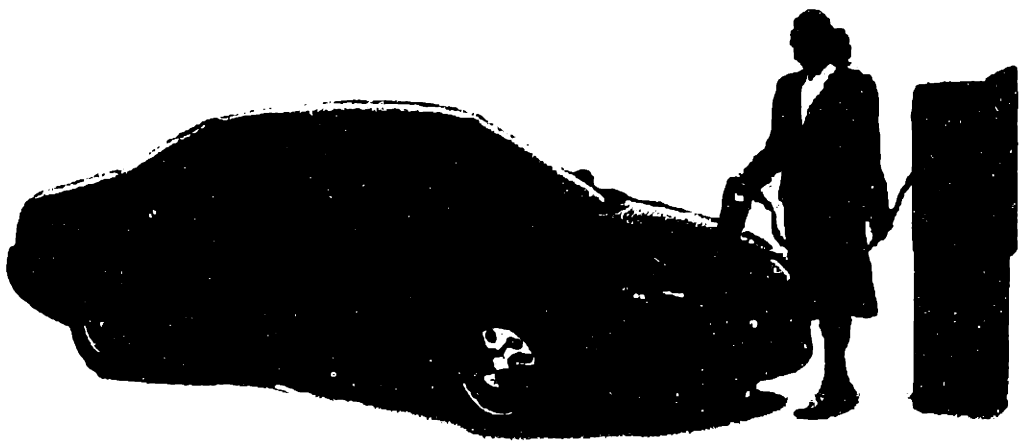


**Figure 1.3:** The EV1 from General Motors, and the MAGNE-CHARGE™ inductive charging system from Delco [15].

tery chargers, more specific details about the system's design will be provided later, along with a more extensive review of background work in the EV area.

The applications discussed so far have all involved separable or sliding inductive couplings where, in each case, the coupling "halves" are held within a relatively close spacing. However, a wide range of applications exist where the space between coupling halves is significant, varying, or obstructed. The quality of the magnetic coupling in these applications is relatively poor. Figure 1.4 illustrates two medical applications that demonstrate such a situation.

Figure 1.4 (a) shows an artificial heart under development by Abiomed [1]. The heart pump is driven by an electric motor and controlled by circuitry implanted underneath the patient's skin. Power to these internal devices is passed through the skin by an inductive coupling worn around the patient's waist. Since the placement of the coupling and the thickness of the patient's skin may vary, the power-electronic drive for the system must tolerate wide variations in the coupling parameters. Approaches for dealing with this variability are described in references [27] and [74].

Figure 1.4 (b) illustrates another *in vivo* medical application of inductive coupling that has been widely applied and discussed in the literature [55, 73, 101]. Radio-frequency

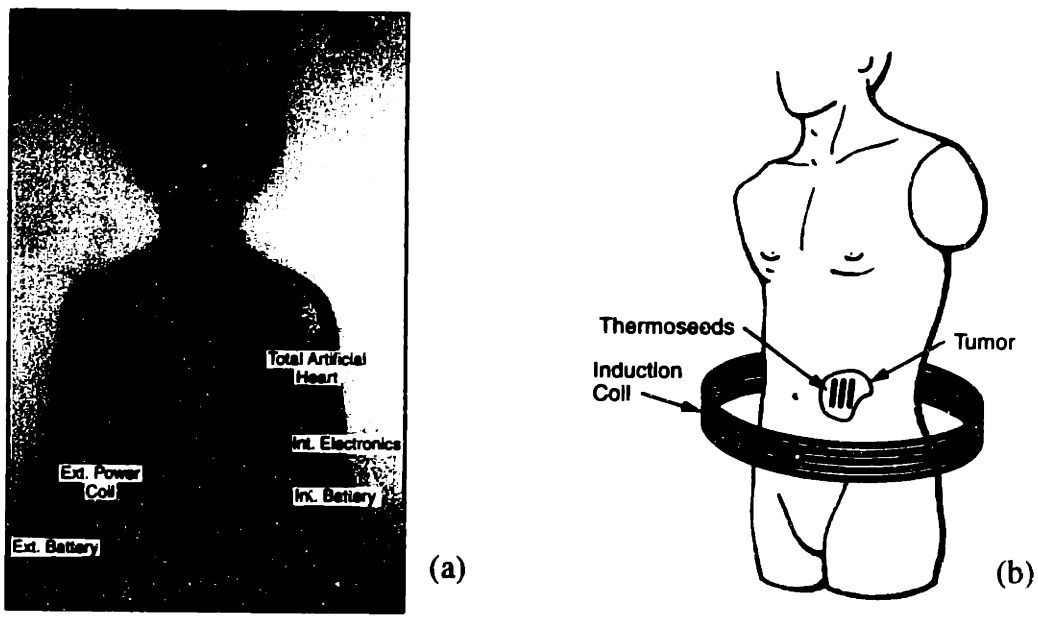



**Figure 1.4:** (a) Inductively-coupled artificial heart from Abiomed [1]. (b) Induction heating of thermoseeds for clinical hyperthermia [31].

(10 – 500 kHz) induction heating is used to raise the temperature of implanted thermoseeds within the human body. As a therapy for malignant tumors, thermoseeds are implanted into a tumor and induction heated until their temperature is raised above 42 °C. This causes local hyperthermia, which destroys the tumor cells. The system is non-contact and inductively coupled. Power is transferred from a primary excitation coil to thermoseeds within the body, where it is converted to heat. The seed and primary coil are poorly coupled in this system, which presents special challenges for the drive electronics. These challenges are addressed in Part III of this thesis.

The products and systems reviewed above represent only a fraction of the many examples of inductively-coupled power systems. However, these cases demonstrate the wide range of possible applications.

## 1.3 Contributions

Contributions from this thesis work span several research areas related to the design and construction of inductively-coupled systems, including power-electronic design and analysis, control techniques, and mechanical design.

Journals within the field of power-electronics are packed with articles describing "new" or "novel" power-converter architectures for a myriad of applications. In fact, many of these architectures are slight variations on existing techniques. However, their contributions are often vital, since they keep modern designs on par with ever-changing component technologies. This thesis presents several power-converter architectures that offer the potential for improvements in efficiency and capabilities over existing designs. The topologies presented here are suitable for a wide range of coupling conditions common to inductively-coupled applications. This range spans from well-coupled low-Q systems, to poorly-coupled high-Q systems. Detailed analyses on select converter topologies are also presented. Some of these analyses expand on oversimplified approaches that have appeared in the literature. For example, two operating modes for the "capacitive-only" filtered rectifier are revealed and described analytically in Chapter 3 [82].

Hardware prototypes were constructed to verify the designs and analyses. Two single-stage inductively-coupled systems were built: a 1.5-kW unidirectional power system and

a 600-W bidirectional power system. Three prototype induction-heating systems were also built. The largest of these prototypes is capable of supplying poor power-factor loads with as much of 15 kVA of apparent power.

This thesis compares and contrasts some of the assumptions and decisions regarding important system design issues that have been made in prior work. It will be shown that many of the designs presented in previous work were arrived at by assuming certain helpful conditions, e.g., low leakage inductance in the inductive coupling, or a regulation environment, where the converter can be assumed to be operating around a fixed output voltage or current. Often such assumptions are not guaranteed. As a result, *ad hoc* arguments have often been used to assure the performance of these systems.

This thesis develops, implements, and demonstrates reliable, large-signal linear tracking control algorithms for the prototype inductively-coupled power systems. Good bandwidth and tracking performance are achieved. Specific examples are worked out in detail, and general multirate algorithms are described for more complex loads. Adaptive digital control implementations are also developed to handle parameter variations. Specific recursive formulations are presented that provide adequate noise immunity in a power-electronic environment. This issue is commonly overlooked in previous papers on adaptive control for power-electronic systems. All the control techniques were implemented on a digital microcontroller and demonstrated using prototype power electronics. Model servomechanical systems were also constructed and used to demonstrate the adaptive techniques.

Mechanical issues are critical to the design of many inductively-coupled systems. Separable, sliding, and rotating inductive-couplings all present different mechanical challenges. This thesis contributes several novel designs to the field of mechanical design. In Part I, a high-power separable coupling for a prototype EV charger is developed. Since the electrical properties of such a coupling vary significantly with the mechanical arrangement, proper design is critical. In Part II, original designs for two-axis rotating inductively-coupled joints are presented. The joints offer a range of motion similar to that, for example, of the human shoulder. Finally, Part III introduces a new inductively-coupled

linear actuator based on polymer gels. This actuator directly combines electrical, thermal, and mechanical interactions in a single inductively-coupled system.

## 1.4 Thesis Outline

As was mentioned in Section 1.1, this thesis focuses on three different inductively-coupled applications. These applications span a spectrum of engineering challenges and issues relevant to inductively-coupled power transfer. Accordingly, the remainder of this thesis is divided into three parts:

**Part I — Single-Stage Inductively-Coupled Power Transfer,**
**Part II — Multi-Stage Inductively-Coupled Power Transfer,**
**Part III — An Inductively-Coupled Polymer-Gel Actuator.**

In Part I, single-stage inductively-coupled power transfer is thoroughly investigated. Part I is composed of Chapters 2 through 6, which include detailed discussions of two prototype hardware systems and extensive coverage of control issues. Chapter 2 motivates the topic of single-stage inductive coupling with a specific application, an inductively-coupled EV battery charger. Existing commercial systems and relevant academic research are reviewed. Then a prototype inductively-coupled connector is described, and equivalent circuit models are furnished.

Chapters 3 and 4 cover, in detail, the hardware design of the two prototype systems. Chapter 3 focuses on a 1.5-kW system with unidirectional power flow. This system was built using printed-circuit-board (PCB) construction techniques, bench tested with a variety of loads, and then packaged in portable demonstration cases. The architecture of the system can be split into two parts, an interleaved boost pre-regulator and a symmetrical half-bridge DC/DC converter. Both parts are described in detail, and their operation is analyzed. Experimental waveforms that verify expected operation are included.

A second hardware prototype is developed in Chapter 4. This unit is a 600-W system with bidirectional power flow. It was built using PCB techniques and then bench tested on a small EV battery pack. Because this unit is bidirectional, both charge and discharge modes (forward and reverse power flow) are possible. A third AC-inverter mode is also

demonstrated, where the unit draws power from the EV batteries and generates a synthesized 60-Hz AC waveform. The architecture of this unit can also be split into two parts, a bidirectional boost-buck-inverter and a symmetrical full-bridge DC/DC converter. The topologies of both parts are discussed, and all three operating modes are analyzed. Experimental waveforms that verify expected operation are included.

Chapter 5 focuses on digital control schemes for the single-stage inductively-coupled architectures. The control techniques presented extend to applications beyond the EV battery charger. Control schemes for power-electronic systems are often based on small-signal models, linearized about some nominal operating point. While this may be suitable in a regulation environment, many applications require a converter that can reliably track a widely varying command reference, where no nominal operating point may exist. Chapter 5 develops large-signal linear voltage- and current-control techniques, which provide stable, well-characterized performance over wide variations in voltage and current. A "pole-placement" voltage-control loop is introduced, and shown to have desirable performance characteristics independent of load power. Specific variations on the voltage-control techniques are then introduced for high-bandwidth current control for battery charging. Finally, general multirate techniques are presented. These simplify the formulation of controllers for complex loads.

Chapter 6 applies these general multirate techniques to adaptive-control schemes that estimate load-model parameters and adaptively "tune" the controller. Specific recursive formulations that provide adequate noise immunity in a power-electronic environment are presented. The techniques were demonstrated by constructing model inductively-coupled servomechanical demonstration systems: a water-bath temperature control system and a motor-speed control system. Both systems allow for real-time parameter variation under closed-loop control. Experimental results are presented using the 1.5-kW hardware prototype from Chapter 3.

Part II of this thesis, Chapters 7 and 8, broadens the scope to include multi-stage inductively-coupled power transfer. Such systems are applicable to, among other possibilities, robotic limbs. The creation of an inductively-coupled AC bus, where "pickups" or power taps can be placed at any point along the length of the limb, is explored. Such a sys-

tem might be ideal for future robotic manipulators that incorporate synthetic (gel) muscles as actuators. Chapter 7 begins with a overview of the robotic limb application and a review of relevant literature. Mechanical designs for two-axis inductively-coupled joints are illustrated. Chapter 8 investigates two possible architectures for multi-stage inductively-coupled systems, voltage-fed and current-fed topologies. Since both topologies have been demonstrated by previous work, the goal of this thesis work is to determine how important design parameters affect the performance of these architectures. This is accomplished using the results of mathematical and simulated circuit analyses.

Part III deals directly with the development of inductively-coupled polymer-gel actuators. Polymer gels can be made to exhibit reversible volume changes that can be triggered by variations in a number of environmental factors. They could, in principle, be used as actuators in servomechanisms and sensors, which range in size from (silicon) mechanisms to larger devices comparable in size and force density to biological systems. Chapters 9 and 10 describe new techniques for remotely triggering gels using a non-contact inductive coupling. Chapter 9 describes the design of thermally responsive polymer gels "seeded" with ferromagnetic material. The resulting "mag-gels" can be coupled to an alternating magnetic field where losses within the seed material generate heat and trigger a volume-phase transition. This volume-phase transition can be harnessed to provide direct linear motion with useful force densities. The magnetic excitation coil and seed material form a poorly-coupled non-contact inductive coupling, which presents special challenges for the drive electronics. Chapter 10 presents suitable power-electronic drives for this "induction heating" application, including three prototype systems, which were built and used to demonstrate the techniques. Test apparatus used to characterize the performance of the mag-gels is described, and measured results are presented. A position control system is also described and used to demonstrate the gels potential as a controllable linear actuator.

Chapter 11 concludes the thesis by summarizing key contributions and identifying areas for future work.

# PART I

# SINGLE-STAGE INDUCTIVELY-COUPLED POWER TRANSFER

## Chapter 2

# Overview

Part I of this thesis focuses on single-stage inductively-coupled power transfer. Special emphasis is placed on inductively-coupled power transfer for the charging of electric-vehicle batteries. However, the work is applicable to a wide range of inductively-coupled electromechanical systems. Since Part I covers a broad range of design issues, the presentation is separated into three categories: connector design, power electronic design, and control design. Connector design is addressed in Section 2.4 below. Power electronics is covered in Chapters 3 and 4, where the design and construction of two prototype hardware systems are discussed. In particular, new circuit topologies that offer improved efficiency and the capability for bidirectional power flow are presented. Chapters 5 and 6 introduce advanced digital-control techniques, which offer flexible, high-performance control. This performance is verified experimentally using the prototype hardware.

In recent years there has been a renewed interest in developing electric vehicles (EVs) for commercial transportation. This development has included research into new systems for charging EV batteries. This chapter motivates the use of inductively-coupled power transfer for the charging of EV batteries. Existing commercial systems and relevant academic research are reviewed. Then the two prototype systems are overviewed in brief. A prototype inductively-coupled "connector" is described, and electrical equivalent-circuit models are furnished. This introductory material provides the necessary background for the chapters that follow.

## 2.1 Motivation — EV Battery Charging

Electric vehicles have been around since the 1800's. In fact, the first EV was built in 1834, and by 1900 they were a dominant automotive technology. However, this initial popularity soon diminished as developing gas-powered vehicles prevailed as the primary form of highway transportation. Electric vehicles were relegated to a relatively small number of specialty tasks. Environmental awareness and the possibility of diminishing fossil-fuel resources have recently fostered a renewed interest.

In 1990, the California Air Resources Board (CARB)—a department of the California Environmental Protection Agency—created a mandate that requires 2% of the cars offered for sale in California in 1998 to be zero-emission vehicles. The mandate increases to 5% in 2001, and 10% in 2003 and subsequent years. Massachusetts and New York have followed suit with similar mandates. Currently, electric vehicles are the only vehicles that qualify as zero-emission vehicles. So, despite arguments that EV technology, and in particular battery technology, is not up to the challenge, automakers have begun a serious effort to develop an EV competitive with gas-powered automobiles. The CARB, in a March 1996 decision, relaxed the mandated requirements. In contracts with seven major manufacturers—GM, Ford, Chrysler, Toyota, Nissan, Honda, and Mazda—specific vehicle quotas were removed until 2003 in exchange for other conciliations. This has not, however, deterred development in the field.

The push to bring EVs to market has spurred research and development by private companies with close ties to the automotive industry. Their interest stems in part from the fact that EVs, with the possible exception of fuel-cell vehicles and hybrids, store their "fuel" in batteries, and these batteries must be recharged. A typical EV battery pack, such as that in the "EV1" electric vehicle from GM, is composed of 26 lead-acid batteries and holds about 16 kWh of energy. This is the energy equivalent of what an efficient generator could produce from roughly one gallon of gasoline. As a result, recharging will occur frequently, and the process must be simple, safe, and efficient.

### 2.1.1 Inductive vs. Conductive

A conventional battery charger is connected to the utility using a plug and receptacle, i.e., an ohmic contact. While this method is well suited for a wide range of low-power applications, the charging of EV batteries presents a more significant challenge. A typical EV battery pack may have a capacity on the order of tens of kilowatt-hours. Thus, even a moderate eight hour "overnight" charge cycle requires several kilowatts of steady charging power. Due to the high power levels, a standard conductive (metal-to-metal) connection has several disadvantages. First, the lack of galvanic isolation and the presence of exposed metal surfaces presents a possible shock hazard to the user. Second, the connection is susceptible to the affects of corrosion, water, dirt, and dust. These disadvantages, along with a perceived strong demand from consumers, have motivated the automotive industry to examine non-ohmic connectors for charging EV batteries. In many ways, an inductive coupling provides a nearly ideal solution.

A separable inductively-coupled connector can serve as a charge port, which connects the electric-vehicle batteries to an external charger. Since power is transferred using a magnetic field, the primary and secondary halves of the connector can be totally insulated, eliminating a potentially important shock hazard to the user. A suitable covering also ensures protection from environmental factors. In order to be accepted as an alternative, an inductive system must be just as lightweight, powerful, and efficient as a conductive system. This requires high-frequency, high-power switching circuits with efficiencies on the order of 90%. In addition, operating frequencies must be in the 10's to 100's of kilohertz so that magnetic components are of reasonable size and weight for a multi-kilowatt system. All of these requirements can be met with current technology.

There is little reason why an inductive system should be more complex or substantially more expensive than an equivalent conductive system. Since a conductive system typically incorporates a fixed (non-separable) transformer for safety isolation, both approaches share similar construction. The primary difference between the two approaches is that the conductive system typically separates at the AC utility, leaving nearly the entire system "on-board" the vehicle. On the other hand, an inductive system separates near to the vehicle batteries, so a majority of the electronics can be located out-

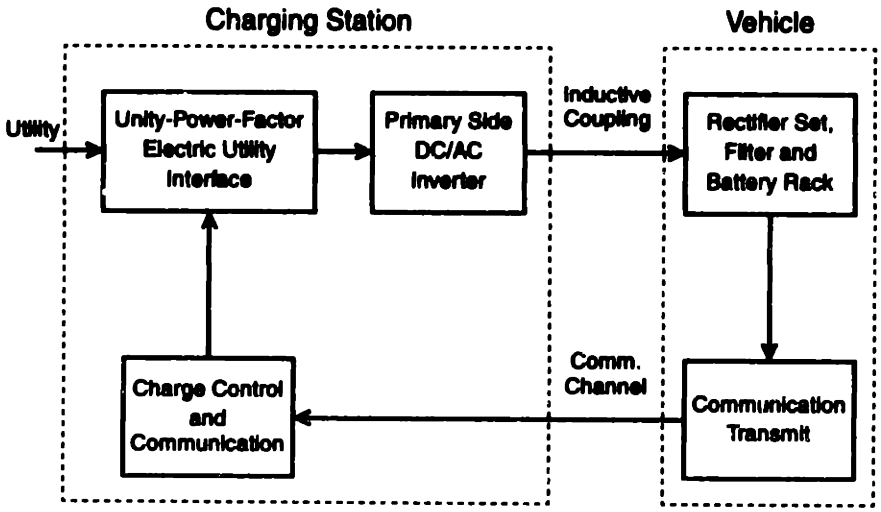



**Figure 2.1:** Block diagram of an inductively-coupled EV charging system.

side the vehicle. It has been projected that this arrangement can reduce the weight of the on-board components by as much as 90% in comparison to an on-board charger [23]. Given that a modern 5-kW EV charger weighs about 33 pounds, this favors the inductive approach [10]. Economic issues such as the warranty and service of the on-vehicle components must also be considered.

A block diagram of a typical inductive architecture is illustrated in Figure 2.1. The system is divided into two parts, the charging station and the vehicle inlet. The charging station is located outside the vehicle, and it contains an AC-to-DC utility interface, a charge controller, a high-frequency DC-to-AC inverter, and the primary side of the inductive coupling. The vehicle inlet contains the secondary side of the inductive coupling and an AC-to-DC converter (typically a set of rectifiers and a filter), which mates the secondary to the vehicle batteries. Isolated communication channels between the charger and the vehicle are also provided so that the charge cycle can be monitored. Although an inductive "channel" is used to transfer power to the vehicle, information may be relayed by a number of techniques, such as radio-frequency transmission, inductive coupling, capacitive coupling, or optical coupling.

### 2.1.2 SAE J-1773

As modern EVs have moved closer to widespread introduction into the marketplace, the Society of Automotive Engineers (SAE) has begun the process of developing stan-

dards for EV battery charging. Two specifications are currently in review: the SAE J-1773 *Electric Vehicle Inductive Coupling Recommended Practice* and SAE J-1772 *Electric Vehicle Conductive Coupling Recommended Practice* [98, 99]. The SAE specifications are an attempt by the automotive industry to define a standard for both inductive and conductive EV chargers. Standards may ultimately help to ensure compatibility between chargers and automobiles from different manufacturers. It is essential, therefore, that these standards are considered when designing an EV battery charger.

The present draft version of the SAE J-1773 specification incorporates a surprising number of elements specific to one circuit topology. This occurred because the draft document was structured around an existing commercial system, the MAGNE-CHARGE system, described in the next section. The specification, therefore, contains details specific to this system's circuit topology. It is expected that the final form of the specification will be modified, or its scope narrowed, to remove this specificity. As is, the standard would severely limit the ability for different manufacturers to develop their own technologies.

| Charging Level | Maximum Input Voltage | Maximum Current Draw | Maximum Power |
|---|---|---|---|
| Level 1 | 120 $V_{AC}$, 1$\Phi$ | 15 Amps | 1.9 kW |
| Level 2 | 240 $V_{AC}$, 1$\Phi$ | 40 Amps | 7.6 kW |
| Level 3 | 208-600 $V_{AC}$, 3$\Phi$ | As Required | 160 kW |

**Table 2.1:** SAE J-1773 EV inductive-charging levels.

The SAE J-1773 specification outlines three standard charging levels for EV systems. They are listed in Table 2.1. This thesis covers the development of a charger that is compatible with Level 1 and Level 2 charging. Techniques to extend the designs to Level 3 will also be discussed.

## 2.2 Existing Technologies

### 2.2.1 Commercial Systems

When this thesis work began in 1994, development was already underway in all areas of EV technology, and battery charging was no exception. Conductive systems, which had

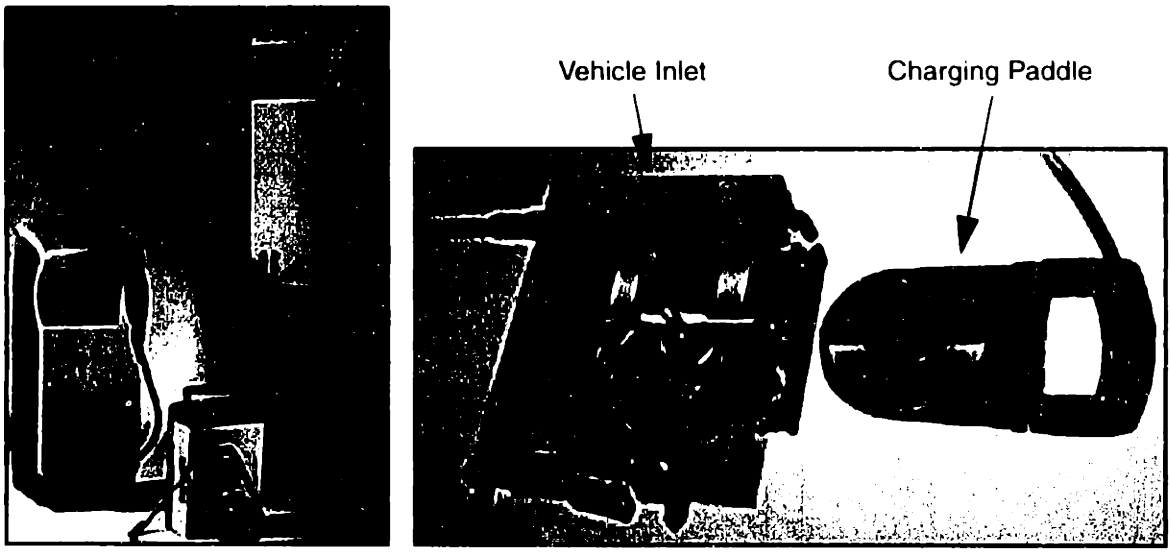



**Figure 2.2:** Photographs of the MAGNE CHARGE system from GM/Delco [15].

been around since the introduction of EVs, were being redesigned to meet modern demands for size, safety, capacity, quality, and efficiency. At the same time inductive systems were becoming available. For example, Hughes designed an inductively-coupled battery charging system named MAGNE-CHARGE™ for General Motor's prototype electric vehicle the "Impact." Because the MAGNE-CHARGE system was one of the first commercial inductively-coupled systems to be developed, its design was used as the basis for the SAE J-1773 recommended practice, as mentioned in the proceeding section. The GM test vehicle, now beyond the prototype stage, has become the GM "EV1," which is available for lease in limited areas of California and Arizona. The MAGNE-CHARGE systems supplied with the EV1 are now manufactured by Delco, a division of GM.

Figure 2.2, on the left, shows a photograph of the available models. The smallest unit is a portable 1.2-kW "emergency" charger. This unit plugs into a 110-$V_{AC}$ outlet and takes approximately 15 hours to recharge the 16.2-kWh battery pack of the EV1. The two larger units are floor- and wall-mount versions of a the "standard" 6.6-kW charger. These units require 220-$V_{AC}$ single-phase power, and they can recharge the EV1 in approximately 3 hours. A 25-kW commercial-use charger (not shown in the figure) based on the same technology also exists. However, it is not currently available for use with the EV1. The photograph on the right in Figure 2.2 shows a cut-away of the charging paddle and vehicle inlet, which together make up the inductive coupler used by the MAGNE-CHARGE system.

The specific dimensions and ergonomic design of this charging paddle have currently been adopted by the SAE J-1773 specification.

### 2.2.2 Academic Research

Many academic papers have been published on battery charging systems for electric vehicles. Relatively few have focused on inductively-coupled systems. The most prominent of these are [23], [58], and [59]. This section briefly reviews these papers and summarizes their contributions.

In [23], Esser describes an inductively-coupled charging *and* communication system for electric vehicles. Power transfer is accomplished inductively at 25 kHz across a separable "contactless charging plug". Simultaneously, communication signals are transmitted at 25 MHz using a *separate* system of inductive and capacitive channels. A two-stage power-converter architecture similar to Figure 2.1 was selected, and a 5-kW laboratory system was built to demonstrate the concept. The system operates from a three-phase 230-$V_{AC}$ source. A three-phase boost rectifier is used for power-factor correction, and a half-bridge series-resonant DC/DC converter transfers power across the inductive coupling. Control of the charging current was accomplished by varying the voltage at the output of the boost converter.

In [58], Klontz *et al.* provide a well-organized paper that surveys a number of suitable converter topologies for inductively-coupled EV charging. The paper focuses on topology selection, with an emphasis on converter utilization and efficiency. Both single- and two-stage converter architectures were compared; an "optimum" topology (based on efficiency and converter utilization) was selected; and a prototype system was built. Their comparison favored a two-stage converter design operated from single-phase 240 $V_{AC}$. A hard-switched continuous-conduction-mode boost converter performed power-factor correction and pre-regulation. A soft-switched half-bridge inverter accomplished DC-to-DC conversion across the inductive coupling. The half-bridge also provided a means for charge control by asymmetrically operating the active switches, thus allowing for modulation of the voltage transfer ratio. The demonstration system operated at a power level of 3.7 kW and approximately 90% efficiency. The results presented in [58] are impressive. However, a

number of simplifying assumptions were made, and certain topologies were not considered. As a result, there is still room for improvement on their "optimal" design.

In [59], Kutkut *et al.* survey converter topologies for 120-kW "Level 3" converters for EV fast charging. Because the power levels are so high, only high-voltage soft-switching IGBT converter architectures were considered. The final topology selection was based almost solely on minimizing the IGBT losses at the proposed power level. As a result, the techniques presented in this paper are not generally applicable to medium and low power systems.

As a whole, this academic work along with many papers on other inductive coupling applications has contributed much to the field. Several suitable converter architectures have been presented, and prototype systems have been built to demonstrate their feasibility. The presentations, however, all have a limited focus. Not one has addressed potentially critical control issues that arise, including stability and tracking performance. Instead the focus has been on comparing converter topologies. Such comparisons, in general, require simplifying assumptions in order to judge dissimilar systems. In addition, critical parameters of the inductive coupling, such as leakage and magnetizing inductance, have been largely ignored because they were assumed to be insignificant or of secondary concern. As a result, none of the papers includes an accurate analysis of how the transfer characteristic across the coupling varies with load power. Also, despite the focus on topology selection, there is still room for improvements in the efficiency and functionality of the power train. For instance, interleaved and bidirectional power conversion has not been considered. All of these deficiencies will be addressed by the designs presented in chapters 3 through 6.

## 2.3 Prototype Systems

Two prototype systems were constructed for the first part of this thesis: a 1.5-kW unidirectional system and a 600-W bidirectional system. The hardware prototypes were built to provide a platform on which different circuit topologies and control strategies could be evaluated and compared. At the same time they were required to demonstrate an enabling technology that could be used to design and construct inductively-coupled systems. The

systems were designed to demonstrate a number key features, which are summarized in Table 2.2.

The different features of the two prototypes affected what circuit topologies were selected for each. Figure 2.3 illustrates schematically the key power train components for each system. Both designs employ two stages of power conversion: a UPF utility interface and a DC/DC converter. In total, four different component stages were built. Chapters 3 and 4 review the design of all four.

## 2.4 Prototype "Connector"

An inductively-coupled connector is a transformer, and transformers have been widely studied and applied for over a hundred years. However, an inductively-coupled connector is special in that its two "halves" or winding ports are designed to separate, rotate, or slide with respect to one another. As a result, many traditional transformer design practices do not apply. For instance, a separable connector requires the presence of an airgap in the magnetic path, and so it becomes impossible to interleave the primary and secondary windings. These constraints limit the values of leakage and magnetizing inductance. This problem has been analyzed by a colleague, Scott Rhodes, in [91].

Rhodes compares a variety of separable transformer geometries. He uses analytical, experimental, and finite-element techniques to compare their inductance parameters. His results reveal that separable transformers, in general, are characterized by relatively low ratios of magnetizing to leakage inductance. Therefore, given a fixed target for the magnetizing inductance, the resulting leakage inductance of a separable transformer will be higher than a comparable non-separable design. This ratio impacts significantly on the design of power electronic systems which drive them, as will be made clear in Chapter 3. Rhodes work also suggests appropriate core and winding geometries to minimize undesired effects.

### 2.4.1 Coupling Design

A prototype inductively-coupled connector was designed and built for use with the prototype hardware systems. Many factors were considered during its design, including

| Feature | Unidirectional Prototype | Bidirectional Prototype |
|---|---|---|
| Microprocessor-based digital control | Yes | Yes |
| Architecture | Two-stage | Two-stage |
| Unity-power-factor operation | Yes | Yes |
| Power handling | 1.5 kW | 600 W |
| Power flow | One way | Reversible |
| Efficiency | Very high | High[1] |
| 120-Hz ripple cancellation | No | Yes |
| Inverter Operation | No | Yes |
| Packaging | Portable | Bench |

**Table 2.2:** Feature summary for the hardware prototypes.

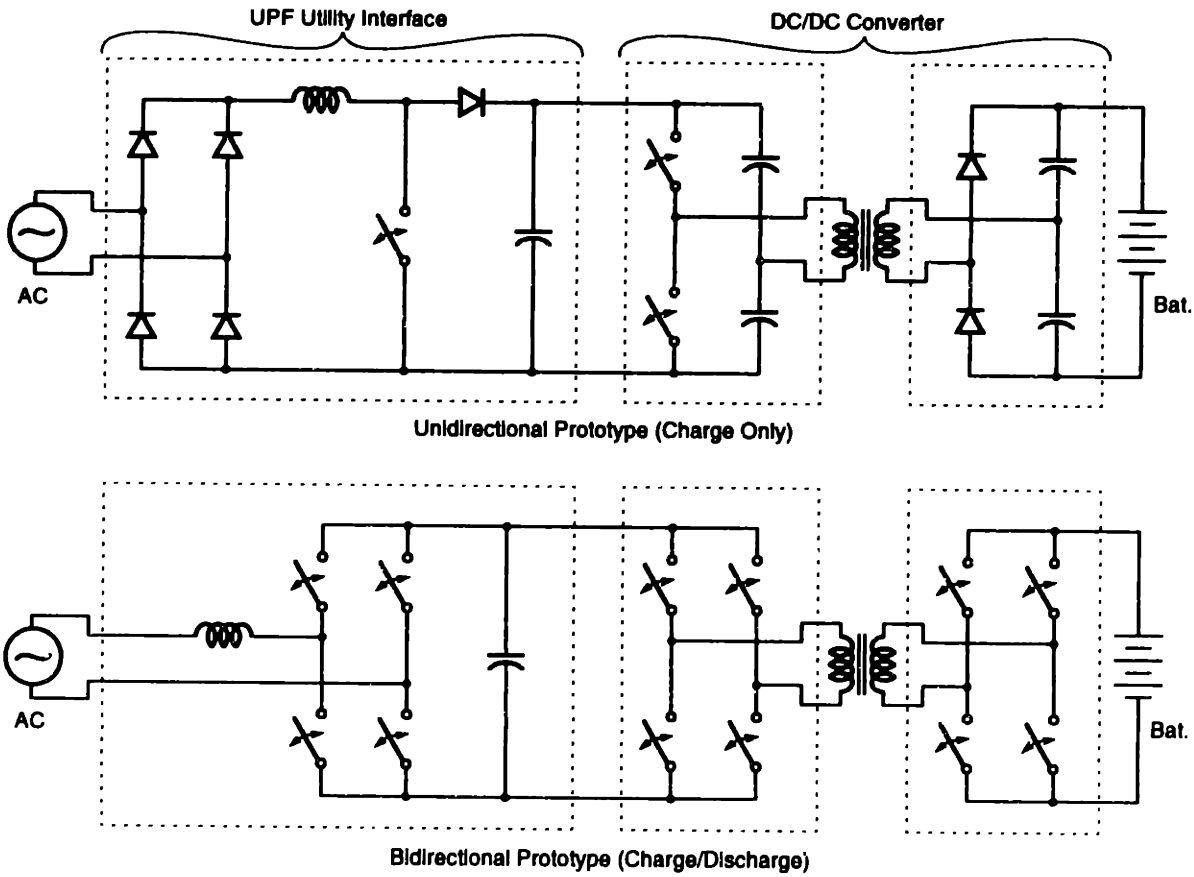



**Figure 2.3:** Circuit topologies for the prototype inductively-coupled systems.

1. The difference in efficiency can be attributed to the 120-Hz ripple cancellation feature (see Chapter 4).

inductance parameters, power handling, size, appearance, and component availability. However, several fixed constraints made it possible to quickly narrow the design space. In particular, it was desired to maintain a "paddle" and "inlet" design, similar to that pictured in Figure 2.2, for compatibility with the SAE J-1773 specification. The specification recommends a ferrite E-Core magnetic structure. While other core shapes, such as the CWT, offer the possibility for better electrical performance, they generally do not offer the mechanical simplicity of the separable E-core.

The specific dimensions of the core depend on a number of factors. The overall weight must be balanced with power handling and performance criteria. At the same time, it is critical to minimize the leakage inductance of the coupling while maintaining the magnetizing inductance above a specified value. This requires that the airgap is made as small as possible and that the cross-sectional area is maximized. Reasons for these requirements are provided in [91]. The cores linearity and power handling ability, on the other hand, are improved by increasing the airgap and the core volume. The separable nature of the core also influences its dimensions. The best compromise proves to be a core with a moderate airgap and a large cross-sectional area. Figure 2.4 illustrates the prototype separable core design.

A commercially available core from TDK was modified to fit the dimensional requirements [106]. As shown in Figure 2.4, two EC90 core halves were modified to create a three-piece assembly. The vertical height was reduced, which results in a core with a large cross-sectional area in proportion to its magnetic path length. A removable "puck" was created in the center leg with a total airgap of 0.50 mm. The wiring arrangement is detailed in Figure 2.5. Copper Litz wire is wound in a spiral pattern to produce the primary and secondary windings. The primary turns are wound around the removable centerpiece or puck. Two 11-turn double-layer windings in parallel form the primary. The spiral patterns rotate counter to each other. This is intended to increase the path length for the leakage flux and improve the leakage specification. The secondary uses two 5-turn single-layer windings in parallel, one above and one below the primary. There is a physical separation between the primary and secondary so that the primary can easily slide in and out of the structure.

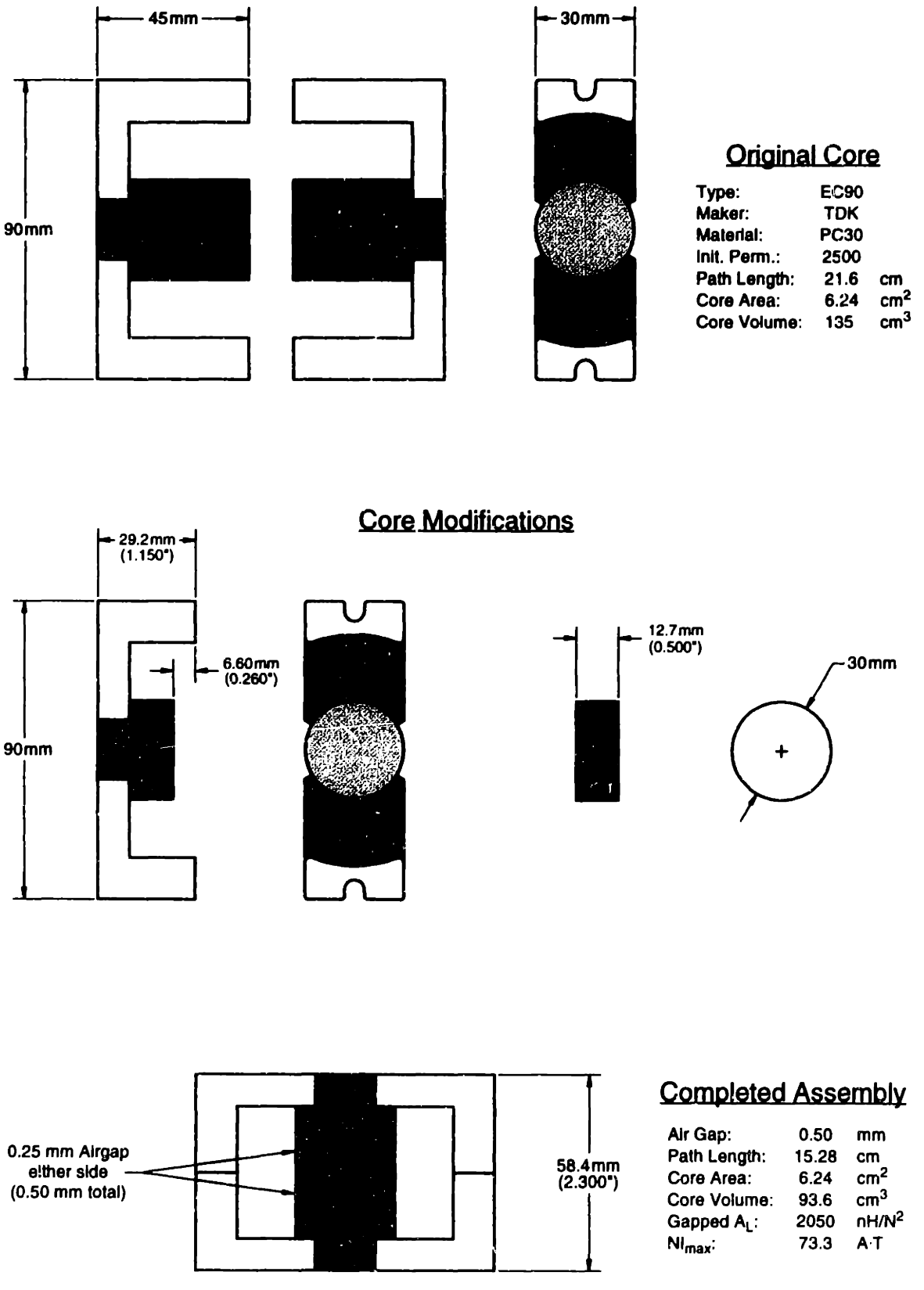



**Figure 2.4:** Modified ferrite E-Core used for the prototype inductive coupler.

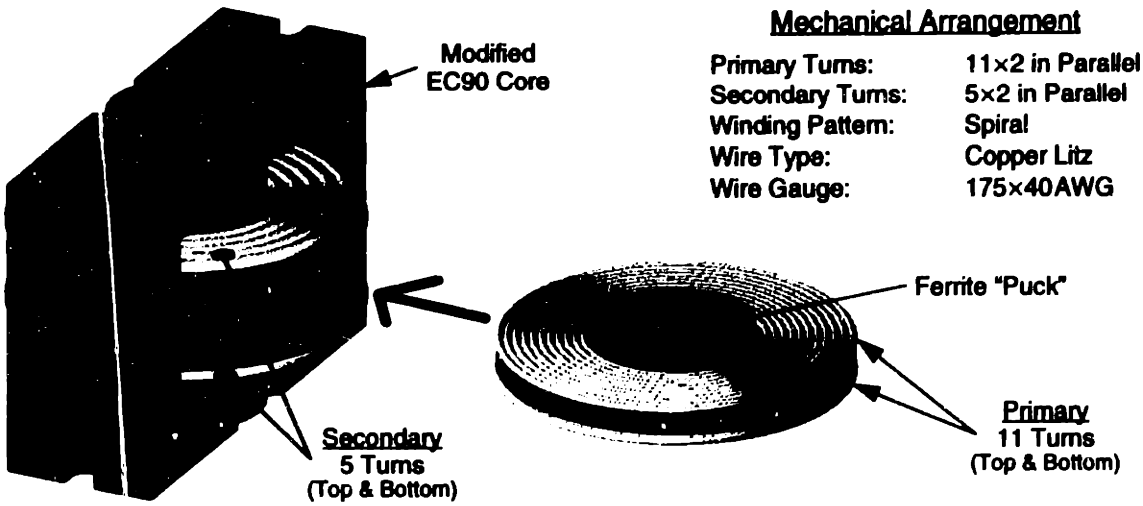



**Figure 2.5:** Mechanical wiring details for the prototype coupling.

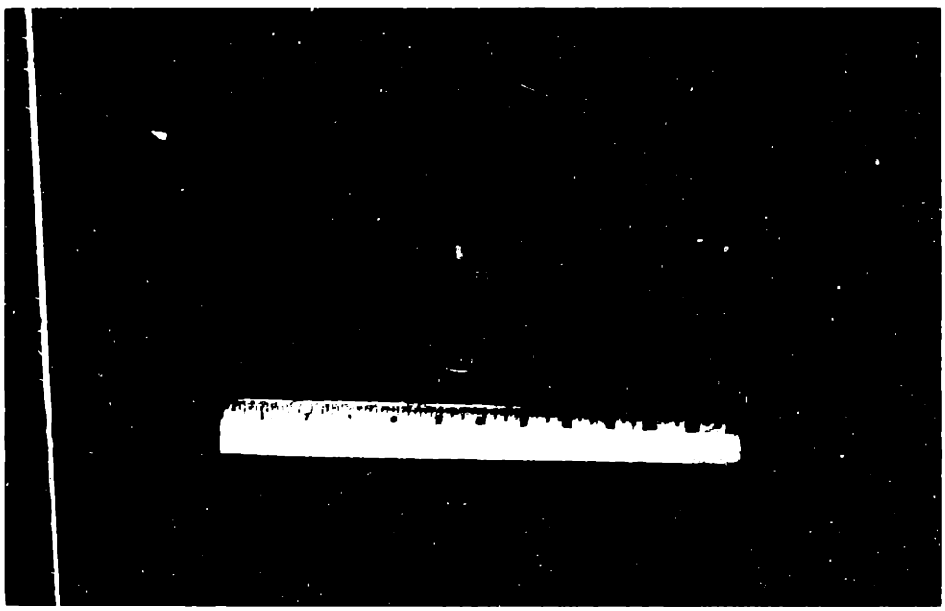


**Figure 2.6:** EV charging "paddle" constructed for the prototype charger.

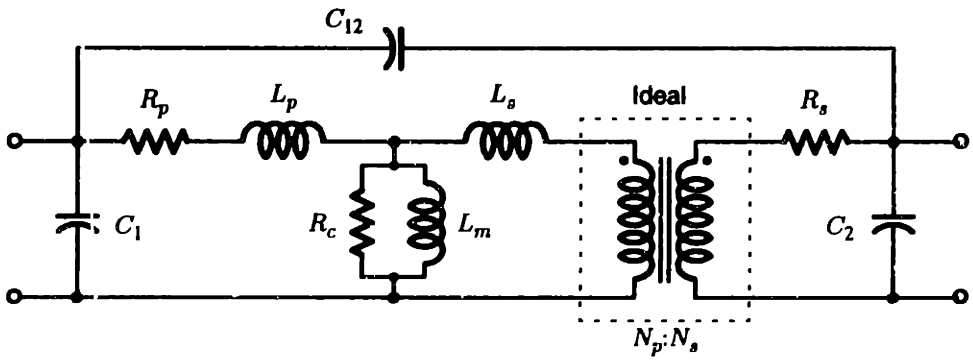



**Figure 2.7:** IEEE equivalent circuit model for a transformer.

The paddle-like primary structure was encased in a molded plastic housing. A photograph of the paddle is shown in Figure 2.6. The housing provides complete electrical isolation as well as mechanical reinforcement. The secondary half of the transformer was mounted in an aluminum enclosure that models a vehicle "inlet." A photograph of the inlet is provided in Chapter 3. The housing also includes a number of optical fibers. The fibers are used to establish an optical communication link, the details of which are provided in Appendix B.

The power transfer capability of this connector is well in excess of the 1.5-kW design specification. An oversized core was selected because a large cross-sectional area helps to reduce the leakage inductance. Also, the 0.50 mm airgap places the magnetic saturation point at well over three times the expected operating level. When driven using the prototype circuit in Chapter 3, the power dissipation within the core is estimated to be between 1.5 and 3.5 W depending on the voltage level of the excitation. The core volume, approximately 94 $cm^3$, amply dissipates this power in free air with only a slight rise in temperature.

### 2.4.2 Equivalent Circuit Models

The same electrical equivalent circuit models that apply to standard transformers also apply to inductive couplings. Figure 2.7 shows the transformer equivalent circuit model recommended by the IEEE [40]. This model includes nine parasitic elements arranged around an ideal transformer. However, the parasitic capacitive elements prove insignificant for the frequencies and circuit topologies studied in Part I, and they may be ignored. The core loss is small as well. So a reduced model may be used for analysis. Three simplified equivalent circuit models are shown in Figure 2.8. These three models are electrically identical. The only difference is in how each represents the inductance parameters.

The top circuit in Figure 2.8 is referred to as the "T" model. The "T" model splits the leakage inductance into two parts, $L_p$ and $L_s$. The two lower circuits are both "L" models, with the inductance parameters lumped on the primary and secondary sides, respectively. Equations relating the inductance parameters for the three models also appear in the fig-

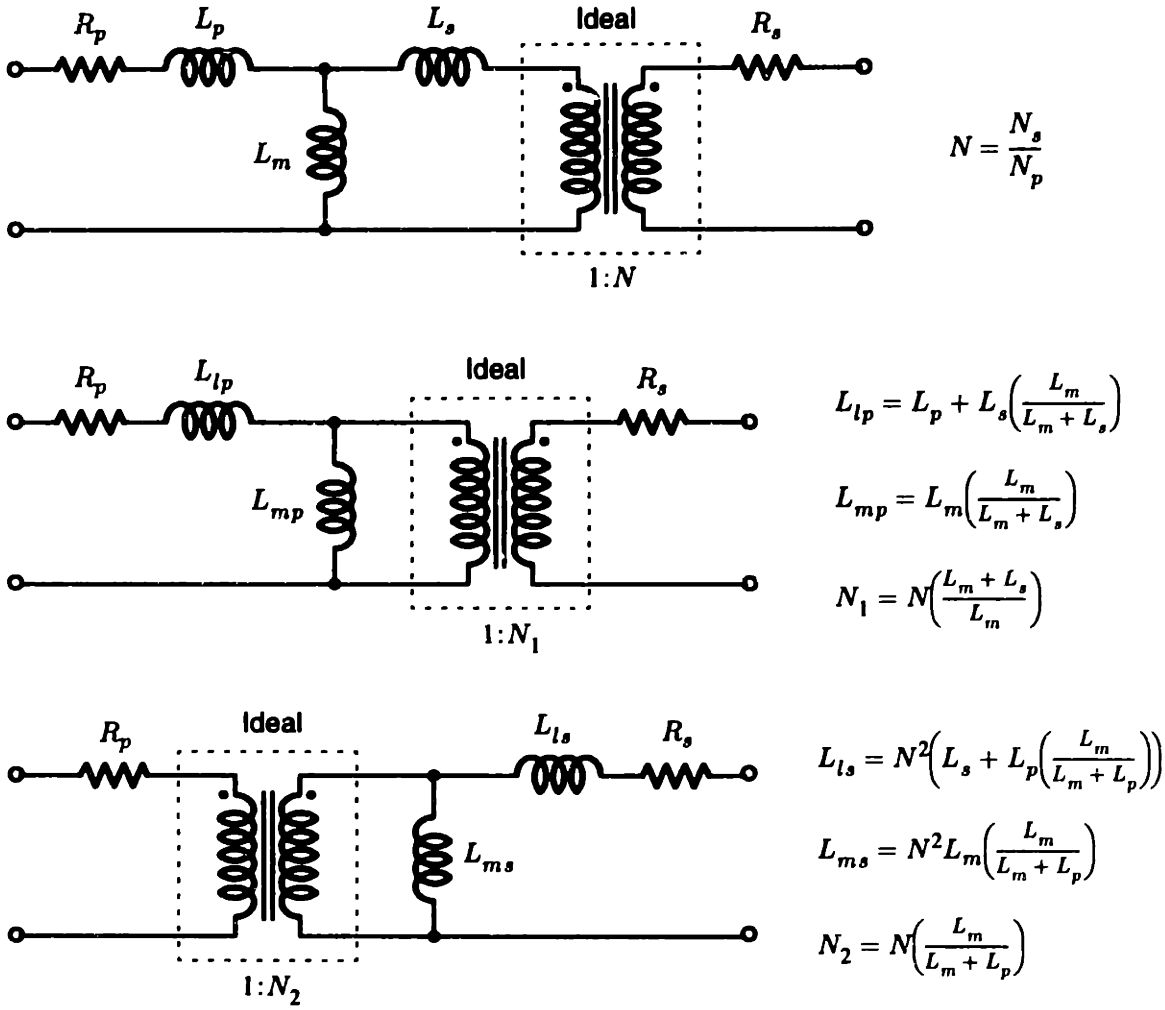



**Figure 2.8:** Simplified equivalent circuit models for the inductive coupling.

| Model Parameter | AMP/M.I.T. Paddle & Inlet | Bench Test Pot-Core |
|---|---|---|
| Primary turns $N_p$ | 11 | 25 |
| Secondary Turns $N_s$ | 5 | 12 |
| "L" model leakage $L_{lp}$ (meas.) | 10 $\mu$H | 23 $\mu$H |
| "L" model magnetizing $L_{mp}$ (meas.) | 250 $\mu$H | 195 $\mu$H |
| "L" model turns ratio $N_1$ (approx.) | 0.458 | 0.491 |
| "T" model leakage $L_p$ (approx.) | 8.3 $\mu$H | 18.8 $\mu$H |
| "T" model leakage $L_s$ (approx.) | 1.7 $\mu$H | 4.3 $\mu$H |
| "T" model magnetizing $L_m$ (approx.) | 251.7 $\mu$H | 199.3 $\mu$H |

**Table 2.3:** Inductive coupling model parameters.

ure. All three representations will be used at various points throughout this thesis. Table 2.3 lists parameter values for the AMP/M.I.T. prototype inductive coupling.

Also included in Table 2.3 are parameter values for a pot-core transformer. A pot-core transformer was constructed for bench testing of the prototype systems. This transformer was built so that its inductance parameters could be easily adjusted for specific testing procedures. Therefore, a non-separable off-the-shelf ferrite "pot core" from Philips (model 6656-3C8) was used as the magnetic material. The windings were wound using copper Litz wire, 25 turns for the primary and 12 turns for the secondary. The primary and secondary windings were placed adjacent to each other along the centerpost of the pot core. This intentionally poor arrangement approximates the high leakage that would arise if the coupling were allowed to separate. A large 125-mil airgap ensures linear operation of the transformer over a wide range of voltage and current conditions. Despite the mechanical differences between this substitute transformer and the modified E-core transformer of Figure 2.5, they are electrically quite compatible.

# Chapter 3

# Unidirectional Prototype

This chapter describes the *hardware* design and operation of a unidirectional 1.5-kW inductively-coupled system. Switching circuit topologies are presented, and their operation is analyzed. Experimental data from the prototype system is then compared with expected results. The understanding of the hardware aspects of the system provided in this chapter allows Chapters 5 and 6 to focus solely on algorithms for digital control.

## 3.1 System Overview

The unidirectional prototype system was designed to demonstrate high-power single-stage inductively-coupled power transfer. From the outset, this prototype was intended to become a portable demonstration system for our sponsor. Therefore, high-quality PCB construction was employed throughout, and the final system was packaged into travel-hardened aluminum cases.

A photograph of the completed portable demonstration system appears in Figure 3.1. The prototype circuits are packaged in two aluminum cases. The packaging was designed to model an inductively-coupled EV battery charging system. Therefore, the case on the left in Figure 3.1 is referred to as the "charge station," and the case on the right is the "vehicle inlet." The demonstration system delivers power to a bank of halogen lamps, which can be seen lit-up in Figure 3.1. The lamps serve as a blindingly bright indicator of the power transfer. It was decided that even a small, several kilowatt-hour, battery pack weighed enough that the system could no longer be considered portable. The halogen lamps offered a convenient high-power alternative. Three 300-W or 500-W halogen bulbs

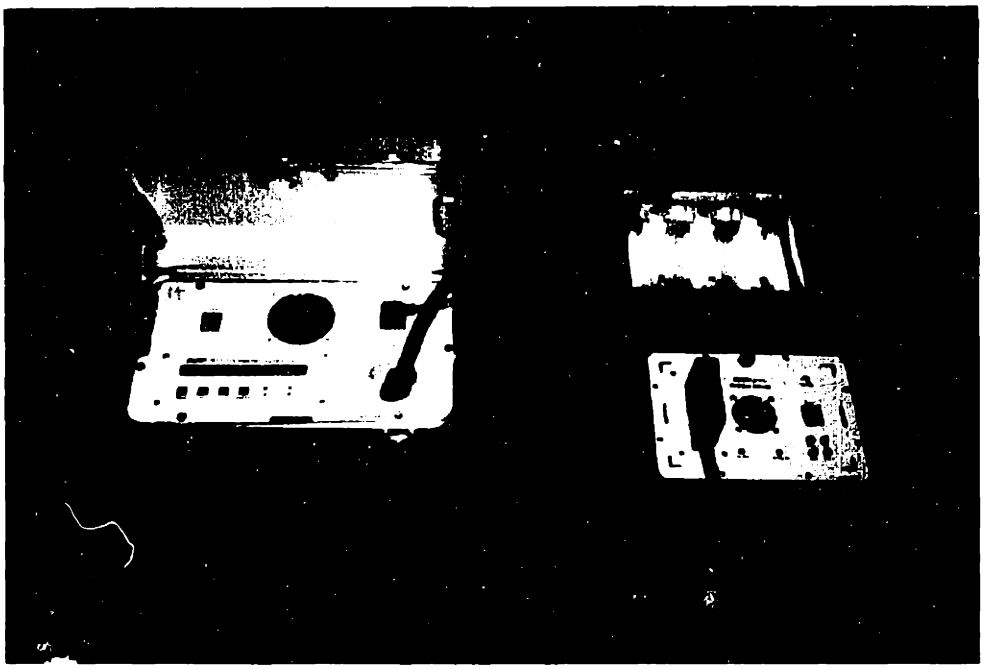


**Figure 3.1:** Photograph of the AMP/M.I.T. prototype.

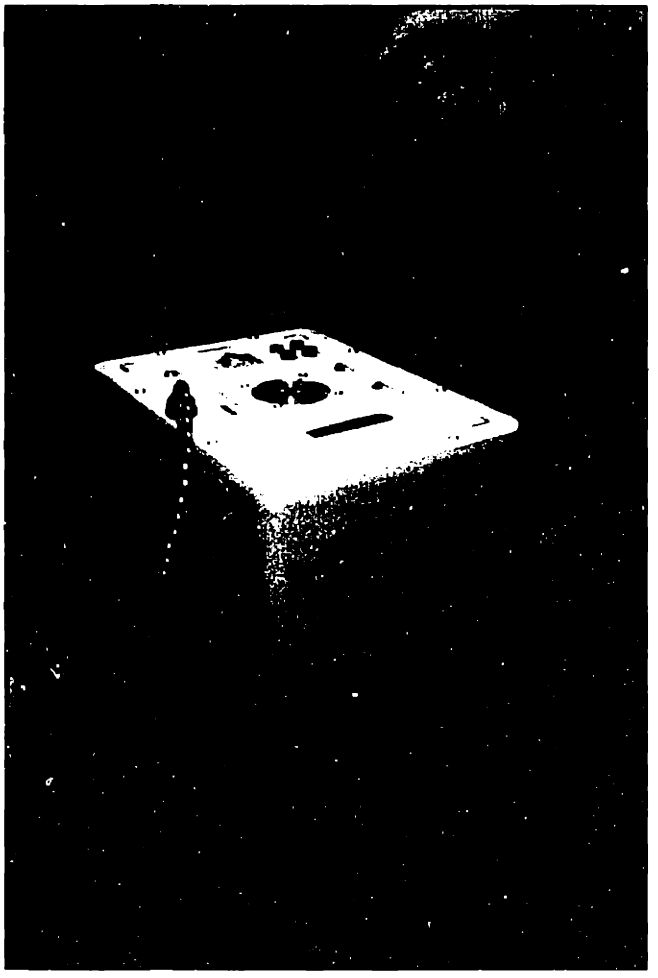


**Figure 3.2:** Photograph of the charging "paddle" and "vehicle inlet."

mount directly into a custom made reflector in the lid of the vehicle inlet enclosure. The result is an easily portable system with a self-contained load of up to 1500 W.

Figure 3.2 shows a close-up view of the vehicle inlet and the charging paddle. The paddle slides easily into its mating receptacle and seats firmly. An optical communications link that aligns only when the connector is seated properly, allows the system to determine the status of the coupling before engaging power transfer. Not surprisingly, no metal-to-metal connection exists between the paddle and the inlet.

Figures 3.3 and 3.4 diagram the contents of the respective enclosures. Although the diagrams contain a significant amount of detail, it is possible to identify the two critical power circuit blocks: the interleaved boost converter and the half-bridge DC/DC converter. These two blocks are described in detail in the following sections. The remaining blocks, such as the microcontroller board and the DC/DC controllers, are touched on briefly here and in Chapter 5. However, detailed descriptions are left to Appendices B and C.

A number of specific design goals were targeted for this system:

1. Demonstrate power transfer across a separable inductive coupling.
2. Deliver a minimum of 1500 W from single-phase 120 $V_{AC}$.
3. Draw power from the utility with near unity power factor.
4. Operate with high efficiency.
5. Demonstrate flexible microprocessor-based digital control.

The circuit topologies presented here in combination with the control strategies in Chapters 5 and 6 accomplish all these goals. The topologies selected for this unidirectional prototype capitalize on a number of proven circuit techniques. These techniques have been modified and fused together in a way particularly suited to the inductively-coupled application. The sections that follow describe and analyze the power-circuit topologies in detail. At the same time, design issues particular to the inductively-coupled application are identified and explained.

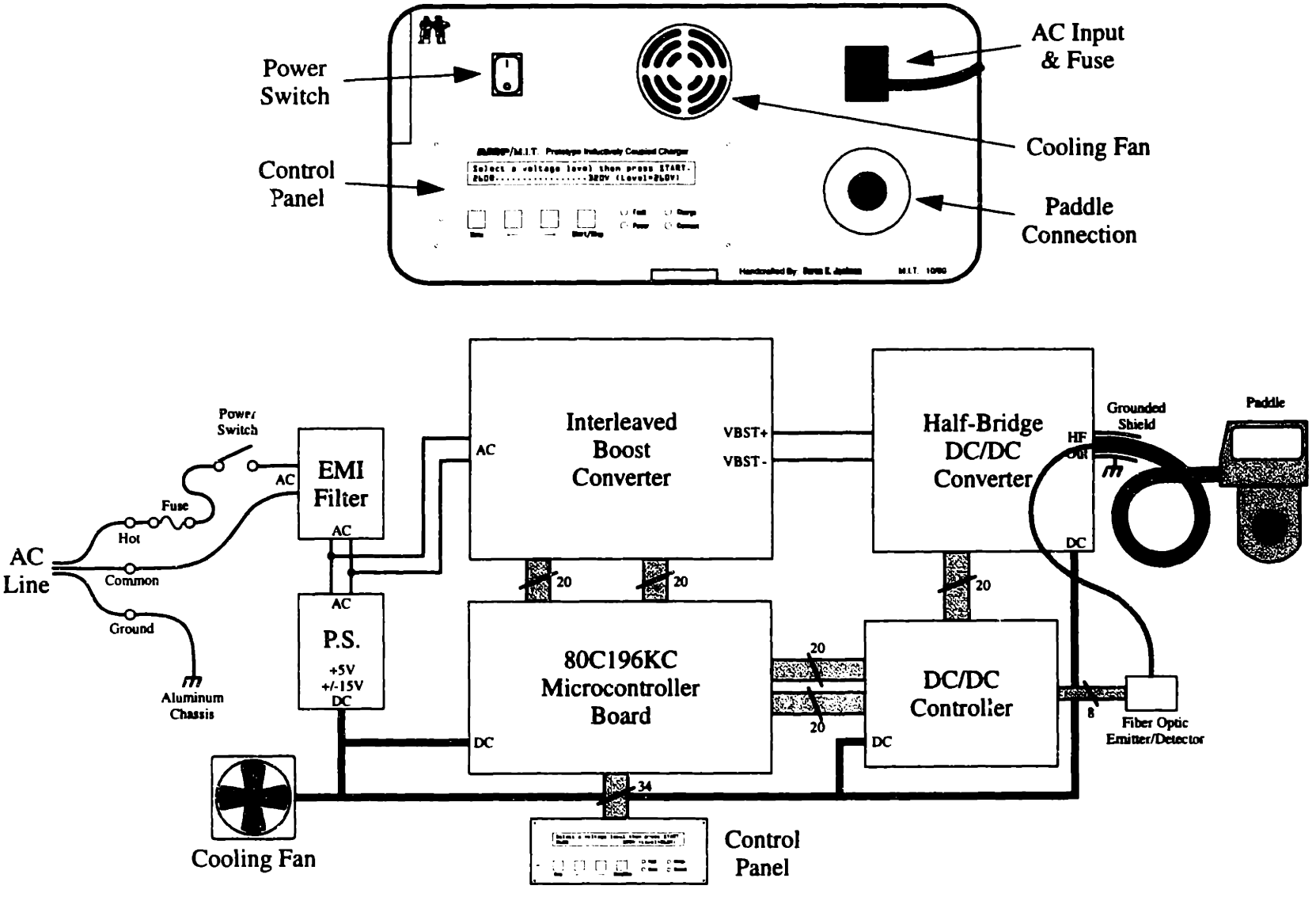



**Figure 3.3:** Block diagram of the AMP/M.I.T. prototype "charge station".

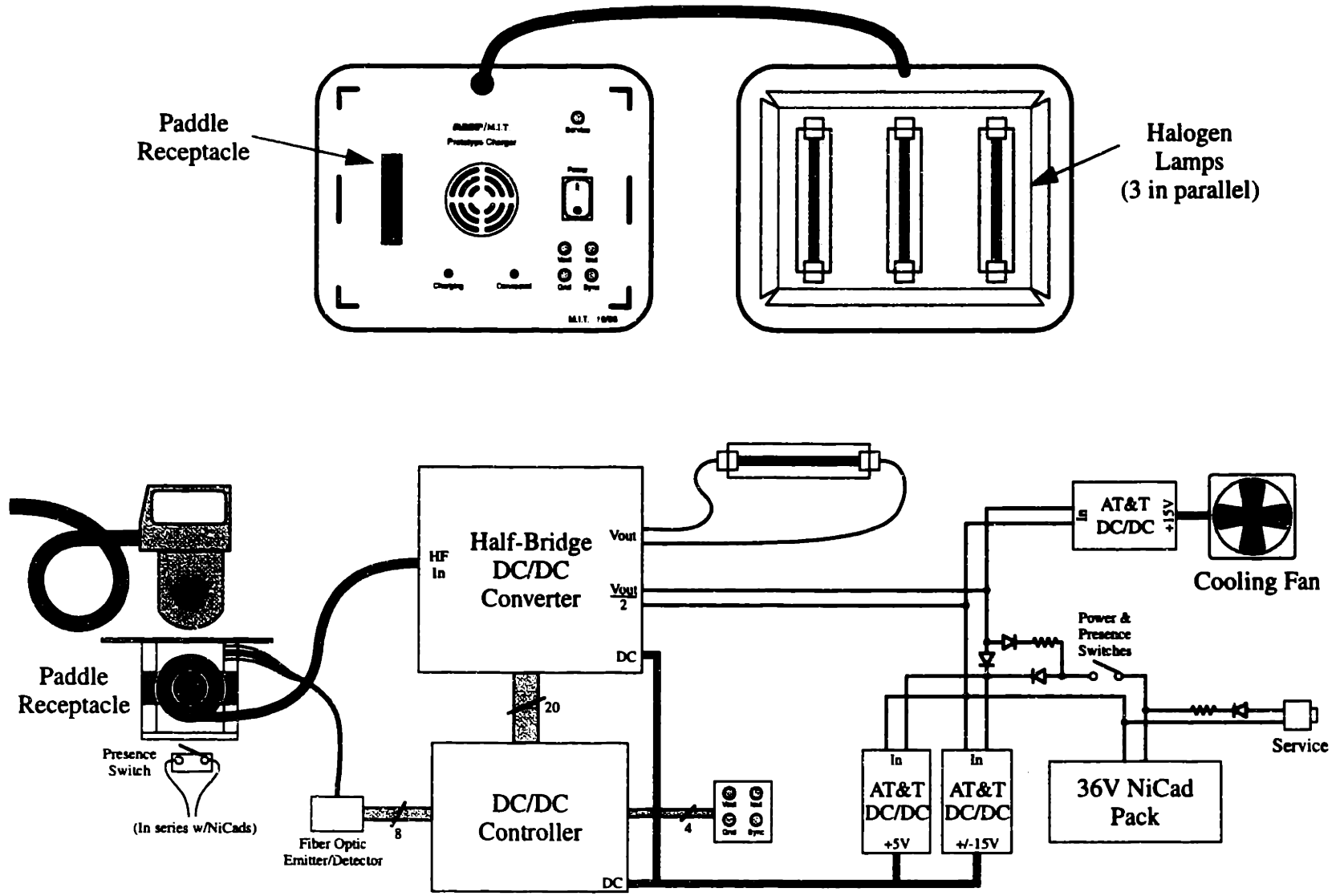



**Figure 3.4:** Block diagram of the AMP/M.I.T. prototype "vehicle inlet".

## 3.2 HPF Interleaved Boost Converter

Power transfer from the AC utility, across the inductive coupling, to the load occurs through two-stages of conversion. The first stage takes the 50/60-Hz sinusoidal voltage from the utility, rectifies it and boosts it to a high DC voltage. In addition, the input current is controlled to match the shape and phase of the input voltage. This conversion is accomplished using a high-power-factor (HPF) interleaved boost converter.

The interleaved high-power-factor boost converter presented here is based on a design by Brett Miwa [79]. His thesis describes interleaved power conversion techniques that offer improved conversion efficiency, power density, and ripple attenuation. Similar techniques are also presented in [71]. Miwa designed and built a prototype 1.5-kW interleaved HPF boost pre-regulator. The power stage in his design achieved a conversion efficiency of approximately 96%. The power stage was so well-suited to this application that very little redesign was necessary in order to use it as the first stage of the unidirectional prototype system. This section describes the design and operation of the interleaved boost converter in enough detail so that the reader should gain a working understanding. However, many of the analytical details are left intentionally brief, since Miwa's work provides a thorough reference [79]. Critical design equations are presented in Appendix A.

### 3.2.1 Interleaved Power Conversion

In general, increasing the frequency of a switching power converter reduces the required energy storage and increases the conversion density of the converter. However, these benefits are typically offset by increases in switching device and core losses. Interleaved power conversion realizes the benefits of higher frequency without increasing device losses. This is accomplished by interleaving multiple switching cells in such a way that the individual ripple components largely cancel each other out. Interleaving can often be implemented with little increase in cost or complexity. Because multi-kilowatt power converters already use multiple semiconductor and magnetic devices to meet current ratings and energy storage requirements, interleaving might involve little more than a rearrangement of existing components into multiple cells.

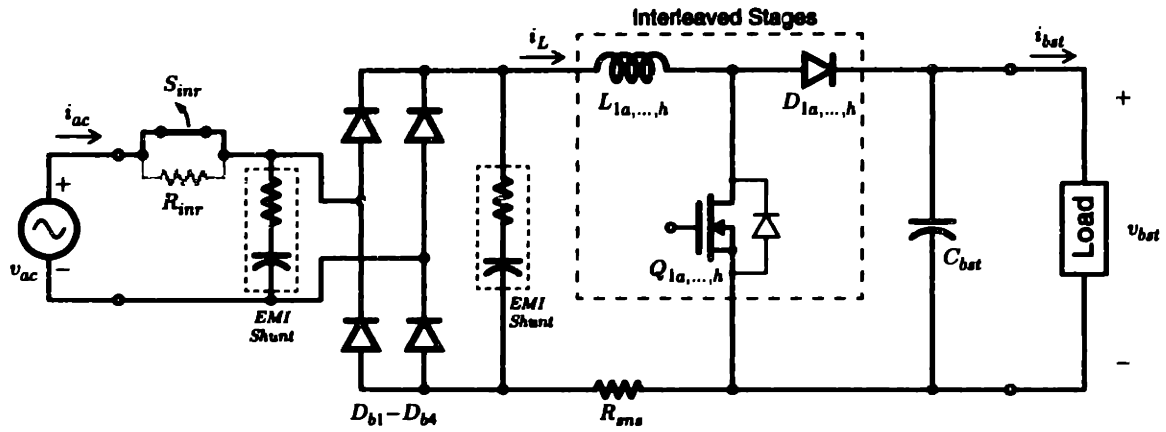



**Figure 3.5:** Simplified schematic of the interleaved HPF boost converter.

Figure 3.5 shows a simplified schematic of a high-power-factor boost converter. To understand the operation of the converter, first consider a single-cell design. The key circuit components are a full bridge rectifier $D_{b1}-D_{b4}$, a boost inductor $L_1$, a controllable switch $Q_1$, a diode $D_1$, and an output capacitor $C_{bst}$. The full-bridge rectifier arrangement ensures that, in steady state, $v_{bst} \geq |v_{ac}|$. Since a single cell has only one active switch, only two switching states exist. When the switch is on, diode $D_1$ is off, and the inductor current $i_L$ ramps upward in response to the rectified source voltage $|v_{ac}|$ applied across it. The output current in this state is sourced by $C_{bst}$. When the switch is off, diode $D_1$ turns on, and the inductor current $i_L$ flows to the output and the filter capacitor $C_{bst}$. The current $i_L$ now ramps toward zero in response to the negative voltage difference $|v_{ac}| - v_{bst}$. Thus, it is possible to modulate the inductor current up or down by varying the duty ratio of the switch $Q_1$. If the switch is pulse-width-modulated (PWM) at a fixed frequency, the duty ratio $d(t)$ can be controlled so that the inductor current tracks a desired reference, with a sawtooth shaped ripple. This provides a means for power-factor correction.

An interleaved boost converter operates in a similar fashion. However, there are $N$ switching cells connected in parallel. Figure 3.6 illustrates a connection with three interleaved cells. Only the inductor, switch, and diode are duplicated in each cell. All cells operate at a fixed frequency as described above. However, the gates drives to each cell are offset by $T_s/N$, where $T_s$ is the switching period. Consequently, the ripple currents within each cell are shifted in phase. Figure 3.7 illustrates example waveforms for the case when

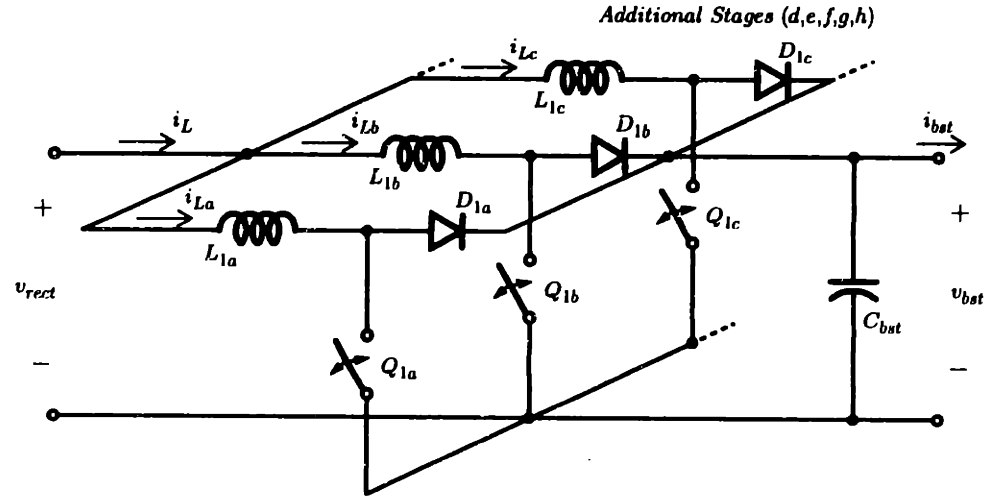



**Figure 3.6:** An illustration of the interleaved boost converter structure.

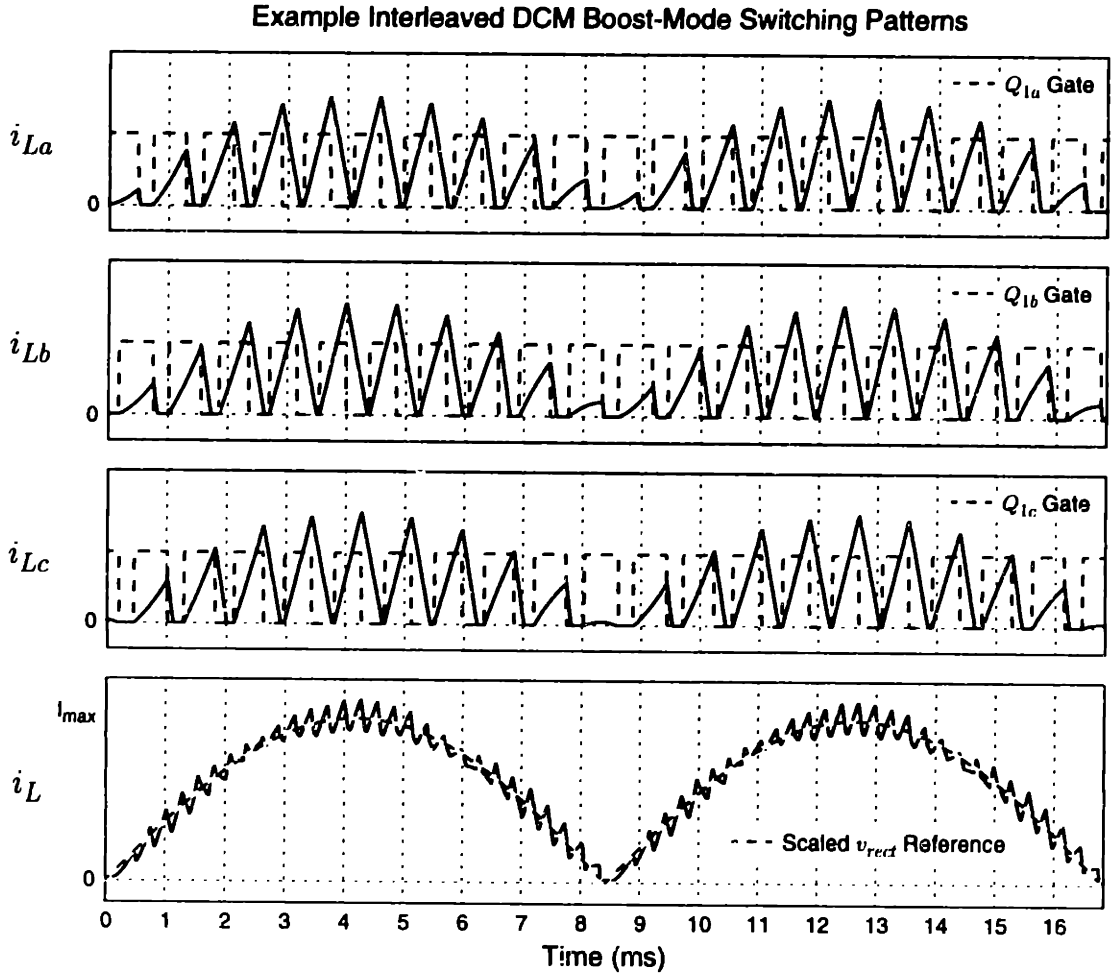



**Figure 3.7:** Example switching patterns for a three-cell interleaved boost converter. Note that each stage operates in discontinuous conduction.

$N=3$. Each cell in the example switches at a frequency of 1.2 kHz. Upon close inspection, it can be seen that the ripple patterns of the three inductor currents, $i_{La}$, $i_{Lb}$, and $i_{Lc}$, are each shifted in phase by precisely 1/3$^{rd}$ the clock period. As a result, significant ripple cancellation occurs between cells. The net current ripple is at least $N$ times smaller than the peak ripple of a single cell and $N$ times higher in effective frequency. This is demonstrated in the bottom trace in Figure 3.7. The exact behavior of the ripple amplitude is described in Appendix A.

Ripple cancellation gives interleaving its many advantages. For example, an interleaved converter can be designed to switch at $1/N^{th}$ the frequency and use $1/N^{th}$ the magnetic energy storage of an equivalent single-cell converter, but still meet the same ripple specifications [79]. In addition, a dramatic increase in efficiency is possible if the interleaved converter is operated in discontinuous-conduction mode (DCM).

Discontinuous conduction occurs when the per-cell current ripple is large enough that the inductor current returns to zero before the end of each switching cycle.[1] Since the controllable switch turns on at zero current, switch turn-on losses and diode reverse-recovery losses are essentially eliminated. These losses would otherwise dominate in a comparable single-cell continuous-conduction-mode (CCM) converter. Also, because each cell in an interleaved design can be operated at a lower frequency, all frequency dependent losses are reduced accordingly.

A range of trade-offs can be made between the number of interleaved cells $N$, efficiency, ripple amplitude, and converter size. This complex topic is discussed at length in [79] and [80]. In short, increasing $N$ will significantly improve efficiency until DCM operation is guaranteed. Beyond that, more stages may still be used to improve the ripple amplitude or decrease the required energy storage of the converter. However, Miwa suggests component mismatches between cells will most likely limit the effectiveness for $N>8$.

1. DCM can be ensured by increasing $N$ or by decreasing the inductance or switching frequency. Limiting relationships are provided in Appendix A.

### 3.2.2 The Prototype Circuit

The prototype circuit was built using 4-layer PCB construction. The topology of the prototype is essentially that shown in Figure 3.5. Naturally, a practical implementation includes a number of additional components, which are not shown on the simplified schematic. Full design schematics along with a photograph of the completed board are provided in Appendix C.

The prototype uses eight interleaved cells, where each cell switches at 25 kHz. Each switching cell is built from a 548-$\mu$H inductor, a 500-V IRFP450 power MOSFET, and a MUR1560 high-speed bipolar diode. The cell currents operate in DCM, within limits defined in Appendix A. The net input-current ripple is continuous with an effective frequency of 200 kHz and an amplitude less than 10% of the rms. input current. The inductors were wound on gapped ferrite cores from Magnetics [66]. Five strand, 26-AWG Litz wire was used to lower conduction losses, and precise matching between inductances (within 1%) was achieved by hand winding the inductors. The output capacitor $C_{bst}$ was built-up from a number of capacitors in parallel. Three large electrolytic capacitors provided a bulk energy storage of approximately 1410 $\mu$F.

Due to the high input-ripple frequency, very little input filtration was required to meet basic conducted EMI specifications set by the utility. Small *R/C* filters were used to provide a low-impedance shunt that diverts high-frequency ripple from passing through to the utility. The location of these filters is shown in Figure 3.5. In addition, a thermistor $R_{inr}$ and relay $S_{inr}$ were used to provide inrush current limiting during start-up. The thermistor limits the inrush of current that occurs when power is applied to the initially uncharged bulk output capacitors. The relay shunts the thermistor during normal operation to prevent undue loss.

### 3.2.3 Inner-Current-Loop Compensation

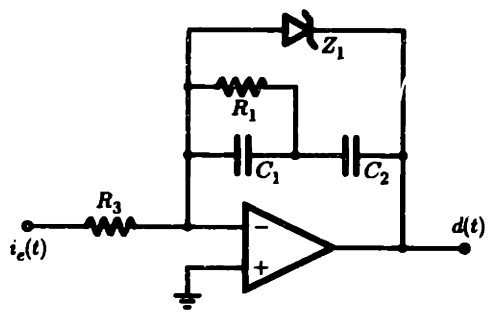
The duty ratio $d(t)$ of the power MOSFET in each cell must be precisely controlled to shape the input current for unity power factor. Since the switching frequency of each cell is high relative to the bandwidth of the voltage loop, this waveshaping control is commonly accomplished using an analog circuit. Direct digital control of the PWM switching

patterns has been shown to incur significant penalties because the required clock speed is related to the product of the switching frequency and the number of possible duty ratios [79]. In addition, if this high-bandwidth waveshaping control is performed in analog *hardware*, the relatively low-bandwidth tasks of output voltage or current control can be readily accomplished in *software* using digital techniques.

Although digital control techniques are discussed thoroughly in Chapters 5 and 6, it is appropriate to briefly address the analog inner-current loop here. Several standard inner-current control techniques exist: hysteresis control, constant off-time control, peak-current control, and averaged current-mode control [70, 111]. Hysteresis control and off-time control are unsuitable for interleaved topologies because their variable switching frequency impedes ripple cancellation. Peak-current control requires slope compensation and is often prone to noise problems. Therefore, because of its simplicity and high noise immunity, "averaged" current-mode control is favored. The fact that the converter is interleaved changes the approach very little from that employed for single-stage converters [111].

Because the converter operates in DCM, the individual cell currents reset to zero before the end of the each switching cycle. Thus, current errors do not accumulate between cells, and the interleaved stages will automatically share current equally. Only the net or input current needs to be measured for feedback. A sense resistor located in the common return path to the bridge rectifier provides this signal. (See $R_{sns}$ in Figure 3.5.) The voltage across $R_{sns}$ is gained, filtered, and then subtracted from a reference trajectory. The resulting current error $i_e(t)$ passes through a compensation amplifier to produce a duty-ratio command $d(t)$. This single duty-ratio command acts as a reference for eight phase-shifted PWM generators. The PWM generators produce the 25-kHz gate-drive signals for the eight interleaved switching cells of the converter. (Please see the Appendix C for detailed schematics of the actual construction.)

A simplified schematic of the analog compensation amplifier appears in Figure 3.8. The transfer function of the compensator, Eq. (3.1), has two poles and a single zero. An integrator ensures perfect steady-state tracking, and a pole/zero pair provides phase lead near crossover. Both the pole/zero locations and the gain of the transfer function must be chosen carefully because too high a gain can lead to subharmonic oscillations. Design




$$H_c(s) = \frac{D(s)}{I_e(s)} = -\frac{(C_1 + C_2)}{C_1 C_2 R_3} \frac{(s + \frac{1}{R_1(C_1 + C_2)})}{s(s + \frac{1}{R_1 C_1})} \quad (3.1)$$

**Figure 3.8:** Analog circuit and transfer function for the inner-current-loop compensation.

equations are provided in Appendix A. A properly designed compensator provides nearly perfect tracking, which is demonstrated in the next subsection.

### 3.2.4 Power-Factor Correction

The power factor of a load is defined as the ratio of the average power delivered to a load $\langle p(t) \rangle$ and the apparent power at its terminals $V_{rms} I_{rms}$ [49].

$$k_p = \frac{\langle p(t) \rangle}{V_{rms} I_{rms}} \quad (3.2)$$

For a load to receive maximum power from the service it must have unity power factor ($k_p = 1.0$). Unity power factor (UPF) implies that the load impedance appears to be purely resistive. The waveforms in Figure 3.9 illustrate a case where the power factor is not unity. The figure shows the input voltage and current to the prototype HPF interleaved boost converter with all switches held off. The resulting rectifier circuit behaves like a peak detector, and large current spikes are drawn from the utility during the peaks in the AC waveform. Using Eq. (3.2) the input power factor for the experimental data is approximately 0.70. This is substantially shy of unity and would limit the available DC load power to about 1200 W.

A further deficiency is that the distorted current waveform is rich with harmonics. These harmonics put undue stress on the utility and can lead to voltage distortion, which occurs when the harmonic currents flow through the source impedance. This voltage distortion is clearly visible in Figure 3.9, and it may affect nearby loads.[2] Standards such as

2. The source voltage distortion is exaggerated by the impedance of an isolation transformer.

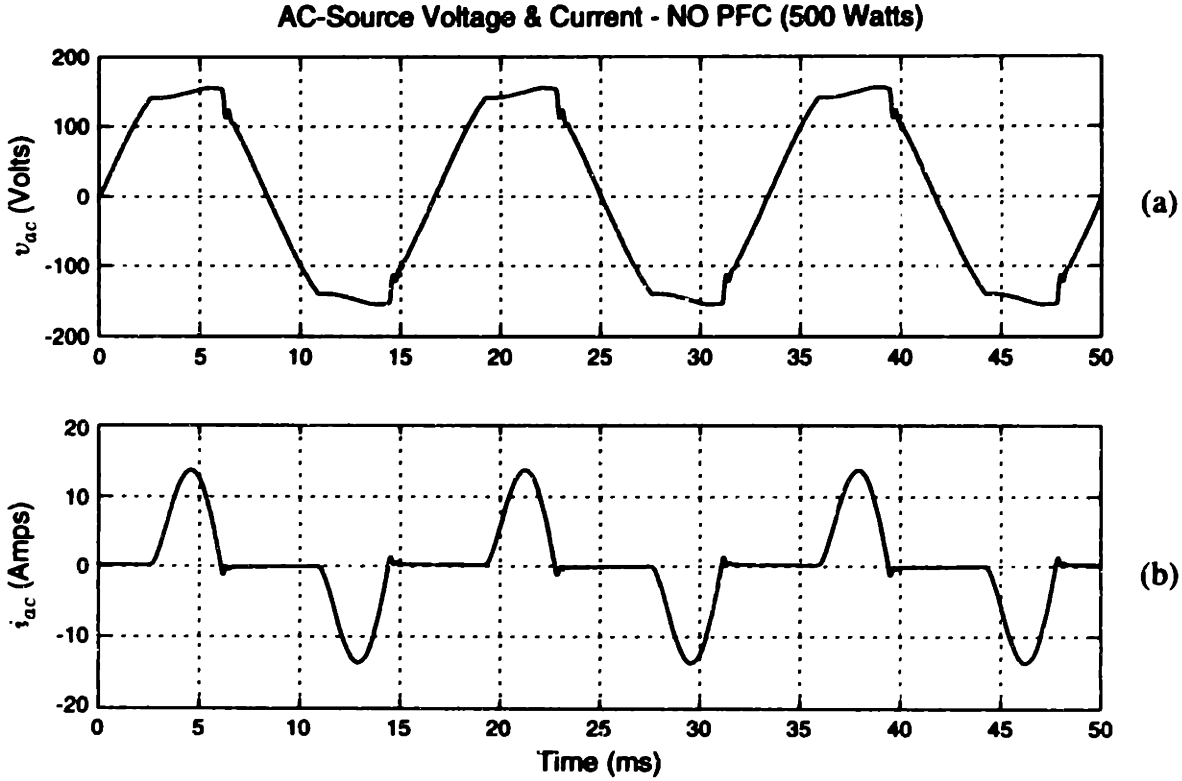



**Figure 3.9:** Input voltage and current without power-factor correction. ($k_p$=0.70)

IEC 555-2, IEC 1000-3-2, and IEEE 519 have been established in an attempt to set harmonic limits.[3] Compliance with IEC 555-2 has been enforced in Europe since January 1997, and it is considered the standard for commercial systems [21].

As mentioned earlier, the HPF interleaved boost topology allows for power-factor correction. The experimental results of this correction technique are shown in Figure 3.10. The figure shows the input voltage and current ($v_{ac}$ and $i_{ac}$) to the interleaved boost converter with power-factor correction enabled. The waveforms demonstrate an experimental power factor of 0.999. The total harmonic distortion (THD) of the current waveform is only 1.36%. The waveforms in the figure show the converter operating from 120 $V_{AC}$ supplied by a Hewlett-Packard 6834B AC source/analyzer. This accounts for the nearly perfect sinusoidal voltage. The input power is approximately 1000 W. The output voltage is nominally 380 V with a small but unavoidable 120-Hz ripple component. The measured efficiency of the converter under these conditions is above 96%.

3. IEC 555-2 and IEC 1000-3-2 are also referred to as EN 60555-2 and 61000-3-2.

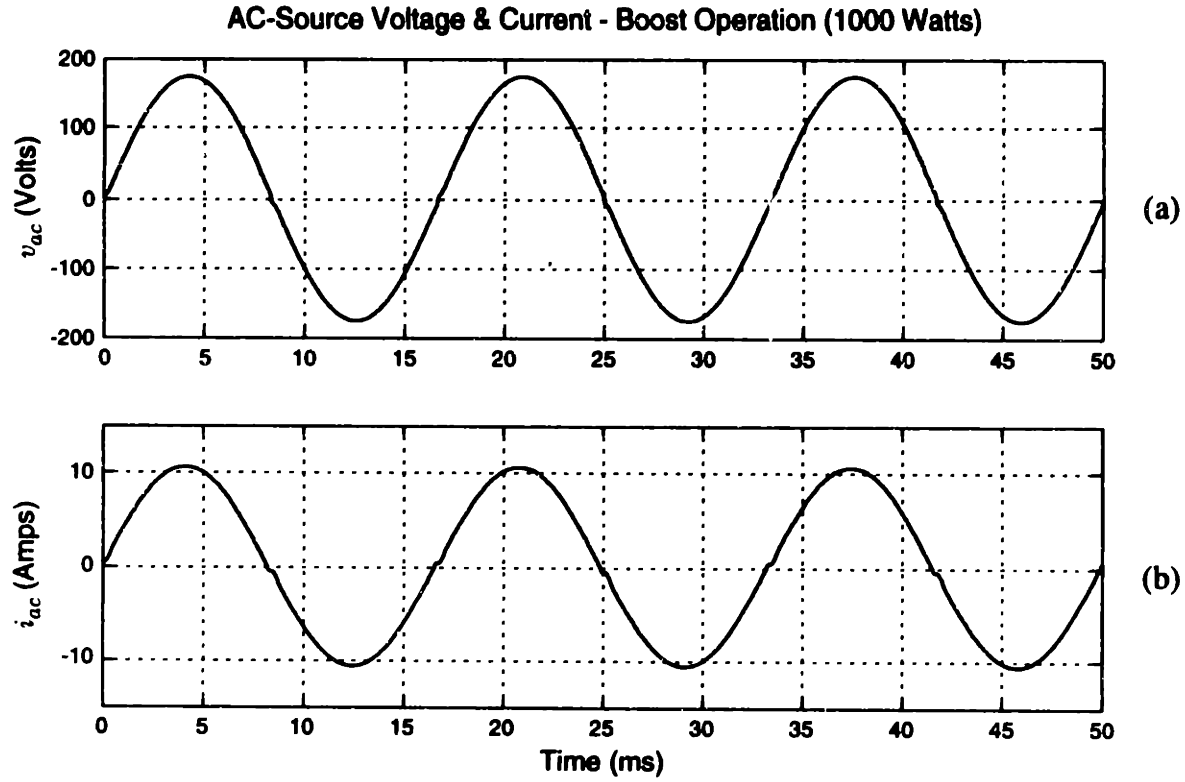



**Figure 3.10:** Experimental input voltage and current with PFC enabled. ($k_p$=0.999)

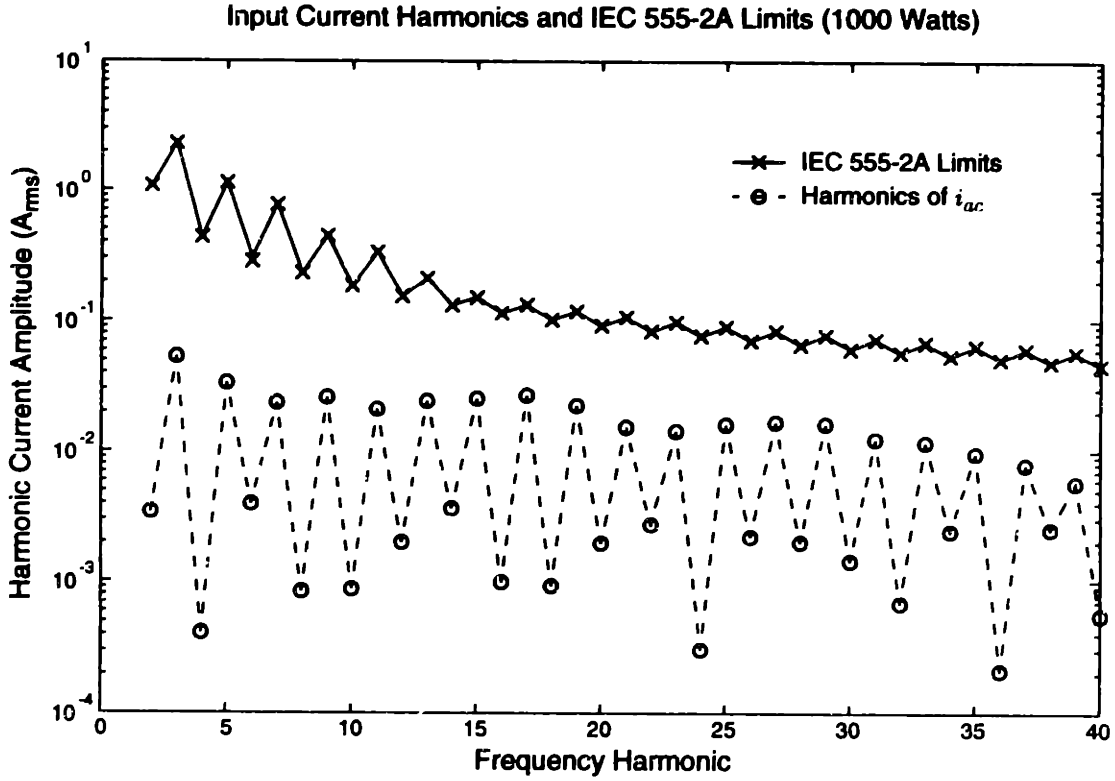



**Figure 3.11:** Magnitude of the input current harmonics versus the IEC 555-2A limits.

Figure 3.11 graphs the magnitude of the input current harmonics along with the limits imposed by IEC 555-2A. Category 2A of the IEC standard applies to low-harmonic rectifiers drawing over 300 W. Explicit limits on the magnitude of the input current harmonics are specified for harmonic numbers 2 through 40. Figure 3.11 clearly illustrates compliance. Notice that the vertical axis is logarithmic. The harmonic current magnitudes are all well below the IEC limits.

## 3.3 Symmetrical Half-Bridge DC/DC Converter

As illustrated in Figure 3.3, the pre-regulated DC voltage at the output of the interleaved boost converter feeds the second power-conversion stage of the unidirectional prototype. The second stage takes power from this DC bus and transfers it across the inductive coupling, using a high-frequency half-bridge DC/DC converter. Since the DC/DC conversion is actually accomplished in two conversion steps, it is more accurately described as a DC/AC/DC converter. A DC/AC inverter generates a high-frequency square wave, which is applied across the primary winding of the inductive coupling. This AC waveform inductively couples to the secondary winding where it is then rectified back to DC and supplied to the load.

This section describes the design and operation of the second stage and presents experimental results from the hardware prototype. The section proceeds in four parts. First, topology selection is discussed, and a topology is selected to meet system design goals. The chosen topology is then described, and its advantages are explained. Next, the efficient switching behavior of the circuit is described and demonstrated experimentally. Finally, the DC transfer characteristic of the converter is analyzed. This analysis provides key results that are needed later to develop digital control algorithms.

### 3.3.1 Topology Selection

There are a large number of high-frequency transformer-isolated DC/DC converter topologies. They can be grouped into two categories: indirect converters (e.g., flyback converters), where the transformer stores energy, and direct converters, where the transformer does not store energy [49]. The flyback configurations were quickly ruled out

because of their high switch stresses and magnetic energy storage requirements at the desired power levels. Thus, only forward converter topologies were considered. Before arriving at a final topology, a number of criteria were considered, such as conversion efficiency, tolerance for high leakage inductance, fixed- or variable-frequency operation, and bidirectional capability. Table 3.1 summarizes these criteria for five popular voltage-fed topologies.[4] A check mark indicates that a particular topology minimizes that criterion.

| Topology | Conversion Losses | | | Fixed Frequency | Bidirect. | Power Range w/leakage |
|---|---|---|---|---|---|---|
| | Switch | Cond. | Rect. | | | |
| 1) Parallel Resonant (Freq. Control) | ✓ | | ✓ | No | No | High |
| 2) Series/Parallel Resonant. (Freq. Control) | ✓ | | ✓ | No | No | High |
| 3) ZVS Bridge (PWM Control) | ✓ | | | Yes | Yes | Moderate |
| 4) ZVS Bridge (Unity duty ratio) | ✓ | ✓ | ✓ | Yes | Yes | Moderate |
| 5) Series Resonant ZVS Bridge (Unity duty ratio) | ✓ | ✓ | ✓ | Yes | Yes | High |

**Table 3.1:** A comparison of DC/DC converter topologies.

The first three topologies in Table 3.1 use a variable frequency or PWM control technique to provide buck or buck-boost control of the output voltage. The fourth and fifth, however, are operated at fixed frequency and unity duty ratio.[5] Therefore, voltage control must be supplied externally. This difference critically affects the relative efficiencies of the topologies.

Since high efficiency was paramount, Table 3.1 divides conversion losses into three categories: switching losses, conduction losses, and rectifier/filter losses. Only soft-switched topologies were considered so that the switching losses would be minimized for all the topologies. Conduction losses, on the other hand, are higher for the first three topologies because their control action (either variable frequency or PWM) tends to increase

4. Current-fed topologies were not considered because they require a significant external inductance, resulting in a poor power-to-weight ratio. Also, they can generate excessive transient voltage excursions, which can easily breakdown semiconductor switching devices.
5. This is equivalent to fixing the PWM duty ratio at 100%.

the rms value of the switch currents as the output voltage is reduced. Example topologies of this type are the variable-frequency LLCC topology in [33] and a number of PWM topologies in [58]. Rectifier losses are high for the third topology because it is the only topology that uses an $LC$ (inductor-capacitor) output filter. An inductor in the output filter generally induces a "lossy" ring with the rectifier capacitance as described in [81].

Fixed-frequency, as opposed to variable-frequency, operation is important because it allows the magnetic design of the coupling to be optimized around a single frequency, which may put the magnetic material in its best operating regime. Parallel and series/parallel resonant topologies, which use a variable frequency to achieve voltage control, do not allow for this optimization. These topologies are also unsuited to bidirectional operation because they incorporate parallel-resonant capacitive elements. So reverse operation requires a current-fed topology to prevent undue current stress on the switching devices.

The only topologies that meet all the criteria are the fourth and fifth, the zero-voltage switched (ZVS) topologies operated at unity duty ratio. These highly-efficient "pseudo-resonant" converters have been described in the literature [82, 83, 86] and by Loveday Mweene in a 1992 M.I.T. Ph.D. thesis [81]. They are economical, have a low parts count, and can be scaled for almost any power range from milliwatts to kilowatts. Even higher power is possible if series-resonant elements are included. High efficiency comes from almost zero switching losses, which is due to recovery of the energy in the parasitic switch and rectifier capacitances. Both half-bridge and full-bridge configurations are possible, and, in fact, prototypes of both were built. The half-bridge topology is described here, and a full-bridge topology is described in Chapter 4. The half-bridge design presented here differs notably from published work, and thus warrants a complete description.

### 3.3.2 Converter Topology

A simplified sketch of the converter topology is shown in Figure 3.12. Only the components necessary for unidirectional power flow are illustrated. MOSFETs $Q_1$ and $Q_2$ in combination with capacitors $C_{b1}$ and $C_{b2}$ form a half-bridge inverter. Switches $Q_1$ and $Q_2$ are operated during alternate half-cycles at a fixed switching frequency (100 kHz in the prototype). Capacitors $C_{b1}$ and $C_{b2}$ are large; thus, node B appears as a small-signal

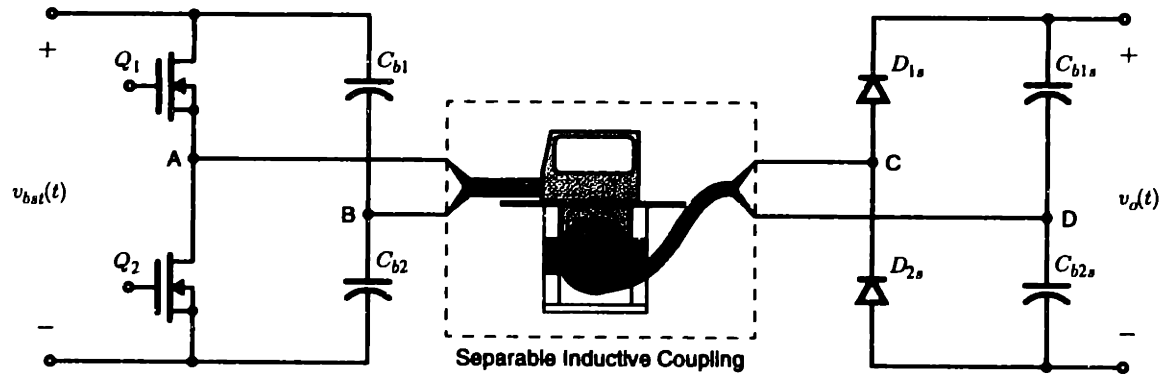



**Figure 3.12:** Simplified schematic of the half-bridge prototype.

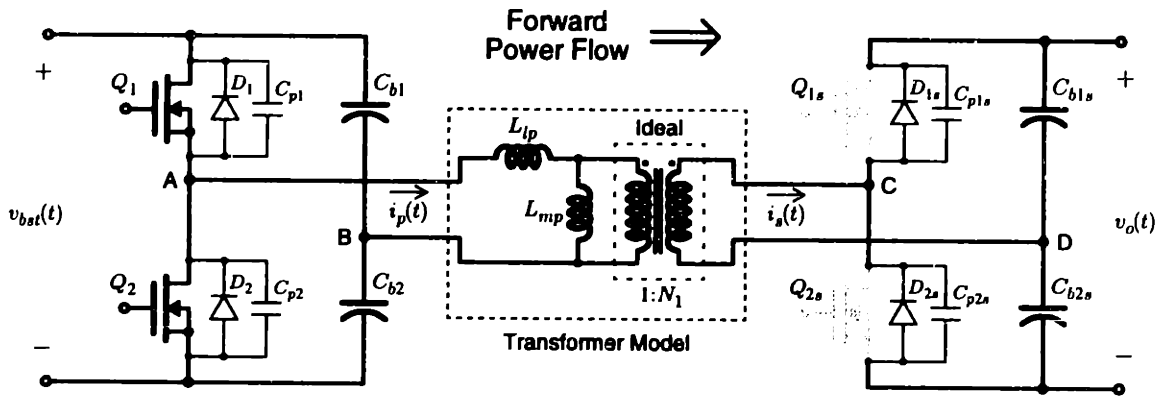



**Figure 3.13:** Bidirectional half-bridge converter FORWARD operation.

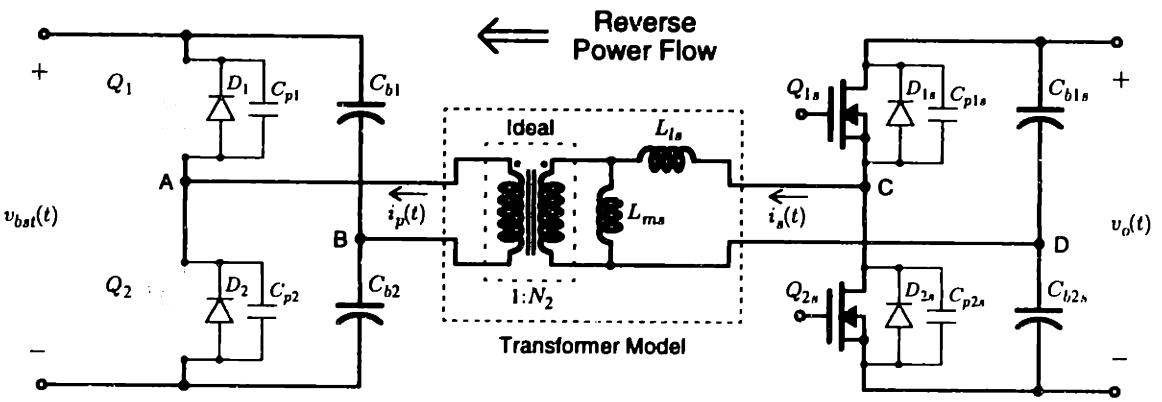



**Figure 3.14:** Bidirectional half-bridge converter REVERSE operation.

ground. The switching action impresses a square wave across the primary of the inductive coupling. On the secondary side, diodes $D_{1s}$, $D_{2s}$ and capacitors $C_{b1s}$, $C_{b2s}$ rectify the AC voltage and produce an output voltage $v_o(t)$. The output is nominally a DC voltage $V_o$ plus a small ripple component. This converter does not use duty-ratio control to regulate the output voltage. Instead, $V_o$ is controlled by varying the bus voltage at the input to the inverter. This is accomplished by controlling the output voltage of the boost pre-regulator, which is nominally DC at $v_{bst}(t) \cong V_{bst}$. The control technique will be explained further in Chapter 5

The rectifier circuit is unconventional in two ways. First, the two diode-capacitor pairs ($D_{1s}$, $C_{b1s}$ and $D_{2s}$, $C_{b2s}$) form a stacked pair of half-wave rectifiers. Consequently, the output voltage is twice that of other rectifier configurations, such as a full-bridge. Second, there is no output filter inductor. An output filter inductor is unnecessary at unity duty ratio because the leakage inductance can support the small AC voltage difference between the transformer secondary voltage and the DC output voltage as seen through the output rectifiers [82].

This unconventional rectifier has several advantages. One is that it allows efficient coupling of a source and a load that are approximately equal in voltage. For example, a 26-cell EV battery has a nominal voltage of 312 V, which is right in the middle of the voltage range of the output voltage $V_{bst}$ of the boost converter prototype. Thus, the transformer turns ratio will be very near to one-to-one, which optimizes the leakage inductance parameter. Also, because there is no output filter inductor, there is no "lossy" ring with the parasitic rectifier capacitances. This has been shown to dramatically improve efficiency [81]. In addition, it makes the primary and secondary circuits symmetrical. Thus, a bidirectional topology can be realized simply by substituting MOSFET switches in place of the two rectifier diodes in Figure 3.12.

Figures 3.13 and 3.14 schematically illustrate the bidirectional topology. Since the MOSFET anti-parallel diodes and parasitic capacitances are integral to the circuit operation, they are shown explicitly.[6] Figure 3.13 demonstrates forward power flow. During

6. Although the figures show individual MOSFETs, multiple devices are actually used. This does not alter the operation of the circuit. The switch blocks are shown in detail in Appendix C.

forward operation $Q_{1s}$ and $Q_{2s}$ are held off, leaving their anti-parallel diodes to perform rectification. Figure 3.14 demonstrates reverse power flow. In this case $Q_1$ and $Q_2$ are held off, and $Q_{1s}$ and $Q_{2s}$ drive the circuit from the secondary side. The inductive coupling in each figure has been replaced by an equivalent transformer "L" model. Reversing the side on which the model's inductances are shown illustrates the complete symmetry of the topology.

Full design schematics along with a photograph of the completed circuit are provided in Appendix C. The schematics detail the design for a bidirectional DC/DC converter. However, only unidirectional operation was necessary for this first prototype.

### 3.3.3 Zero-voltage Switching

Zero-voltage switching (ZVS) implies that the active MOSFETs turn on and off with zero volts across them. Energy in the MOSFET parasitic capacitances is recovered rather than dissipated during switching. The half-bridge circuit described here uses a "pseudo-resonant" switching technique, which can ensure ZVS over the entire operating range of the converter [86]. Since the converter operation is identical for both forward and reverse power flow, only the forward case will be described. Please refer to the schematic in Figure 3.13 during the description below.

Experimental waveforms showing the prototype circuit in operation appear in Figure 3.15. The circuit is operating with input and output voltages of $V_{bst}=380$ V and $V_o=116$ V, respectively. About 1000 W is being delivered to the load with an efficiency of 93.4% across the pot-core inductive coupling. The traces on the left of Figure 3.15 show one and a half switch cycles of the 100-kHz switching waveforms. The traces on the right show expanded views around the negative switch transition.

The primary-side behavior is as follows. Beginning with switch $Q_1$ on and $Q_2$ off, the primary current $i_p(t)$ ramps up linearly.[7] MOSFET $Q_1$ is turned off at the end of the half-cycle with the current $i_p(t)$ at its peak value of $I_p$. For a short period, approximately 300 ns in the prototype, both MOSFETs remain off, and the current built up in the leakage induc-

7. In fact $i_p(t)$ displays a damped second-order ring between $L_{lp}$, the output filter capacitors, and the parasitic resistance in the circuit. However, the switching period is much shorter than the second-order time constants, hence the linear assumption is reasonable.

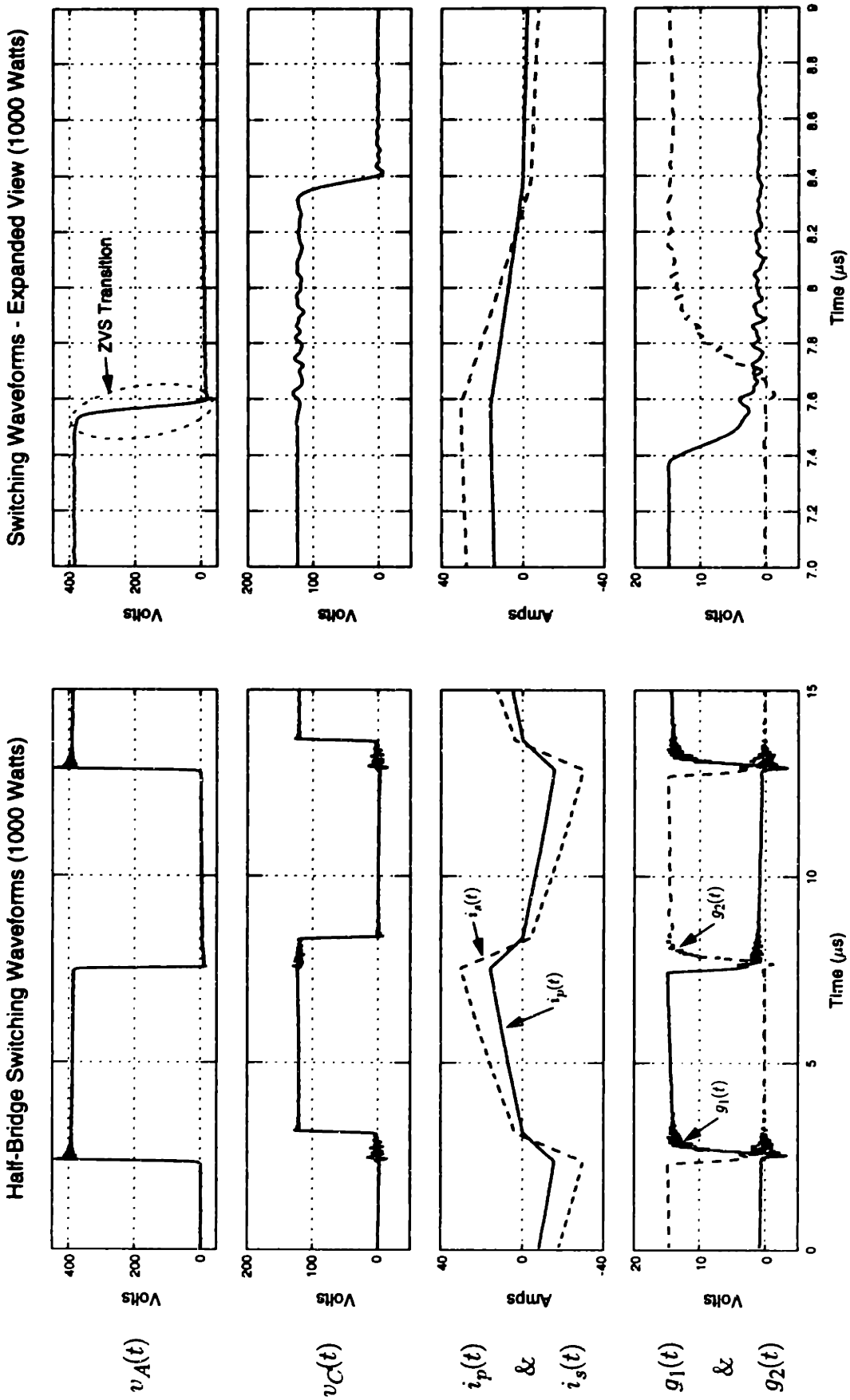



**Figure 3.15:** Experimental switching waveforms from the half-bridge prototype converter at 1000 W.

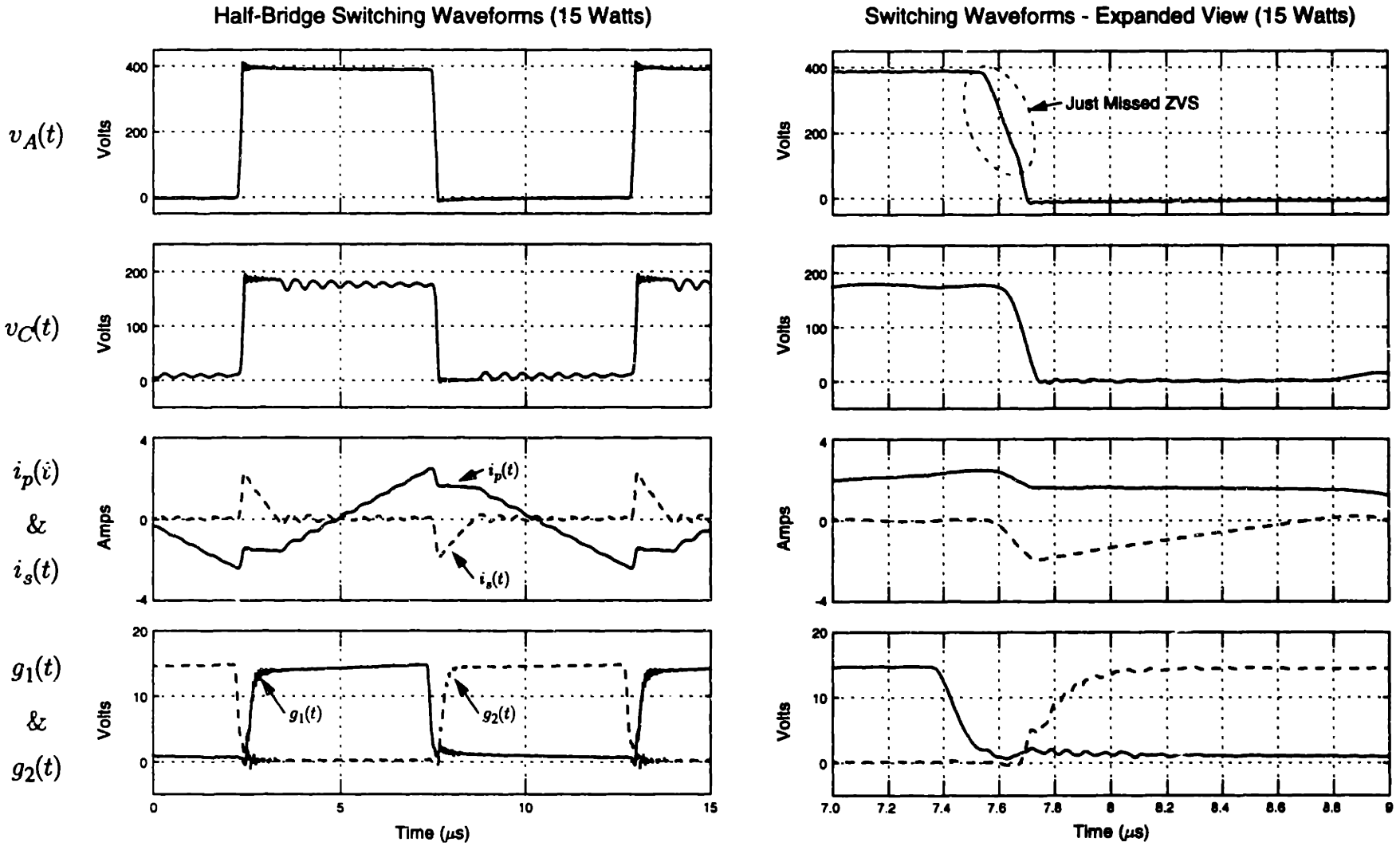



**Figure 3.16:** Experimental switching waveforms from the half-bridge prototype converter at 15 W.

tance $L_{lp}$ rings with the parasitic switch capacitances. This resonant transition rings the voltage at node A to the negative rail, where it is clamped by the anti-parallel diode $D_2$. MOSFET $Q_2$ can then be turned on with "zero" volts across it and no loss. Shortly after $Q_2$ begins to conduct, the secondary-side rectifiers commutate, and $i_p(t)$ begins to ramp linearly toward $-I_p$. So the process repeats for the negative half-cycle.

Equations (3.3) and (3.4) describe the resonant transition. They were derived assuming linear capacitors as in [81].

$$i_p(t) = I_p \cos(\omega t) \tag{3.3}$$

$$v_A(t) = V_{bst} - (I_p \sqrt{L_{lp}/C_A}) \sin(\omega t) \tag{3.4}$$

The resonant frequency $\omega = 1/\sqrt{L_{lp}C_A}$ and $C_A = C_{p1} + C_{p2}$. Using (3.4) it is clear that ZVS is guaranteed if

$$I_p > V_{bst}\sqrt{C_A/L_{lp}}. \tag{3.5}$$

This condition is easily satisfied under load. However, with little or no load current, $I_p$ is equal to the peak magnetizing current. Thus, a single constraint on the magnetizing inductance can ensure ZVS under all loading conditions.

$$L_{mp} < \frac{1}{8 f_s}\sqrt{\frac{L_{lp}}{C_A}} \tag{3.6}$$

There is a conduction loss penalty for making the magnetizing inductance too small. Therefore, in practice, ZVS can be missed at very light loads with little affect on the total conversion losses. The waveforms in Figure 3.16 demonstrate that with only a 15-W load the converter just misses ZVS on the downslope of $v_A(t)$.

Similar resonant transitions occur at node C on the secondary side. After $v_A(t)$ drops to zero, $i_p(t)$ and $i_s(t)$ begin a steep linear decrease toward zero. When $i_s(t)$ reaches zero, the current must commutate from diode $D_{1s}$ to $D_{2s}$. The commutation occurs resonantly because $L_{ls}$ rings with the parasitic capacitance at node C. For this analysis it is convenient to refer the transformer model inductances to the secondary side (see Figure 2.8). Equations (3.7) and (3.8) describe the resonant transition.

$$v_C(t) = V_o(1 - \cos(\omega_s t)) \tag{3.7}$$

$$i_s(t) = -\frac{V_o}{\sqrt{L_{ls}/C_C}} \sin(\omega_s t) = -I_s \sin(\omega_s t) \tag{3.8}$$

The resonant frequency $\omega_s = 1/\sqrt{L_{ls}C_C}$ and $C_C = C_{p1s} + C_{p2s}$. Since node C rings from $V_o$ to zero on the downward transition, commutation occurs in 1/4 cycle of the frequency $\omega_s$. The secondary current $i_s(t)$ then begins a linear ramp from a starting level $I_s$ for the next half-cycle of the switching frequency. The next subsection examines this commutation in more detail.

### 3.3.4 DC Transfer Characteristic

The 'DC transfer characteristic' refers to how the ratio of the converter output to input voltage changes with loading. Because the leakage inductance of the inductive coupling is non-zero, the output voltage droops with increasing load power. This effect can significantly limit the deliverable load power if the leakage inductance is too large. Therefore, it is important to have an analytical tool to predict the droop and to aid in system design. Two analytical approaches are presented below: a commutating reactance model and a switching model.

Figure 3.17 (a) shows a simplified model of the DC/DC converter. The primary-side inverter has been replaced by an equivalent square-wave voltage source,

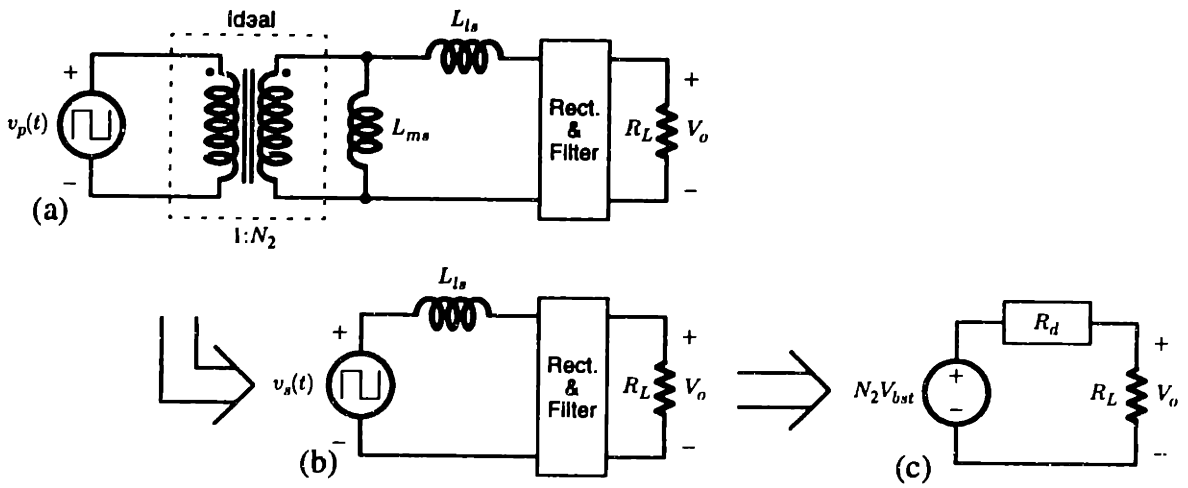



**Figure 3.17:** Commutating reactance model for the DC/DC droop calculation.

$$v_p(t) = \frac{V_{bst}}{2}\,\mathrm{sgn}\left[\sin(2\pi f_s t)\right], \tag{3.9}$$

where $V_{bst}$ is the DC bus voltage supplied by the boost pre-regulator and $f_s$ is the switching frequency of the half-bridge inverter. Also, the inductive coupling has been replaced by a transformer "L" model with secondary-side inductances. Since the magnetizing inductance $L_{ms}$ does not influence the output, and the ideal transformer simply scales the source voltage, Figure 3.17 (a) can be simplified further. As shown in Figure 3.17 (b), all that remains is an AC source driving the rectifier block through a commutating reactance $L_{ls}$.

The droop characteristic or "load regulation curve" for this configuration has been studied in the context of line-frequency rectifiers [49]. It has been shown that for sinusoidal excitation the effect of the commutating reactance on the output voltage can be modeled as a lossless resistor. The same technique can be applied here if the square-wave excitation of the DC/DC converter is approximated by its fundamental sinewave component. The resulting DC model is shown in Figure 3.17 (c). A DC voltage source of amplitude $N_2 V_{bst}$ drives a resistive divider between $R_d$ and the load $R_L$. The droop resistor $R_d$, defined in (3.10), models the lossless effect of the commutating reactance.

$$R_d = \frac{\pi^3 f_s L_{ls}}{2}. \tag{3.10}$$

The output voltage can be written as a function of $I_o$, as follows:

$$V_o = N_2 V_{bst} - R_d I_o. \tag{3.11}$$

Substituting $I_o = P_o / V_o$ into (3.11) yields a quadratic equation, which can be solved to find the output voltage as a function of output power, as follows:

$$V_o = \frac{N_2 V_{bst}}{2} \pm \frac{1}{2}\sqrt{(N_2 V_{bst})^2 - 4 R_d P_o}. \tag{3.12}$$

The real solutions to (3.12) trace out the curves plotted in Figure 3.19.

Figure 3.19 shows two pairs of predicted and experimental curves. One pair of curves correspond to the pot-core transformer from Table 2.3. For the second pair an additional 20.6-$\mu$H inductor was added in series with the transformer primary in order to approxi-

mate a larger leakage inductance. Both experimental curves where measured with $V_{bst}=380$ V. The output power was swept using an electronic load, and the output voltage was recorded at a number of points. Although the point-by-point match between predicted and experimental results is poor, the reactance model is useful as an approximation that captures the overall transfer behavior as a function of load.

The peak power transfer across the coupling occurs when the expression under the square-root in (3.12) is equal to zero. Rewriting this expression leads to a simple closed-form solution for the peak power across the coupling:

$$P_{o,\,max} = \frac{(N_2 V_{bst})^2}{4R_d} = \frac{(N_2 V_{bst})^2}{2\pi^3 f_s L_{ls}}. \tag{3.13}$$

From (3.13) it is clear the peak power capability of the inductive coupling is inversely proportional to the leakage inductance.

A more exact analytical solution can be obtained using a switching model of the output rectifier. Figure 3.18 shows a simplified schematic of the DC/DC stage that will aid the development of this model. The inverter circuit and transformer model have been replaced by a properly scaled square-wave source,

$$v_s(t) = V_s \operatorname{sgn}[\sin(2\pi f_s t)], \tag{3.14}$$

where $V_s = N_2 V_{bst}/2$.

The secondary current $i_s(t)$ can be classified into one of two characteristic modes. The characteristic shapes for Modes 1 and 2 are sketched in Figure 3.18. Although somewhat idealized, the shapes match well with the experimental data in Figures 3.15 and 3.16. In both modes the time average of the rectified secondary current is related to $I_o$ by (3.15).

$$\frac{\langle |i_s(t)| \rangle}{2} = I_o \tag{3.15}$$

But recall that $i_s(t)$ undergoes 1/4 cycle of a resonant ring at the moment of rectifier commutation. The amplitude of this ring, described by (3.8), is $I_s = V_o\sqrt{C_{ps}/L_{ls}}$, and it determines the starting level of $i_s(t)$ at the beginning of each half-cycle. Two operating modes arise because $i_s(t)$ starts each half-cycle at a finite current.

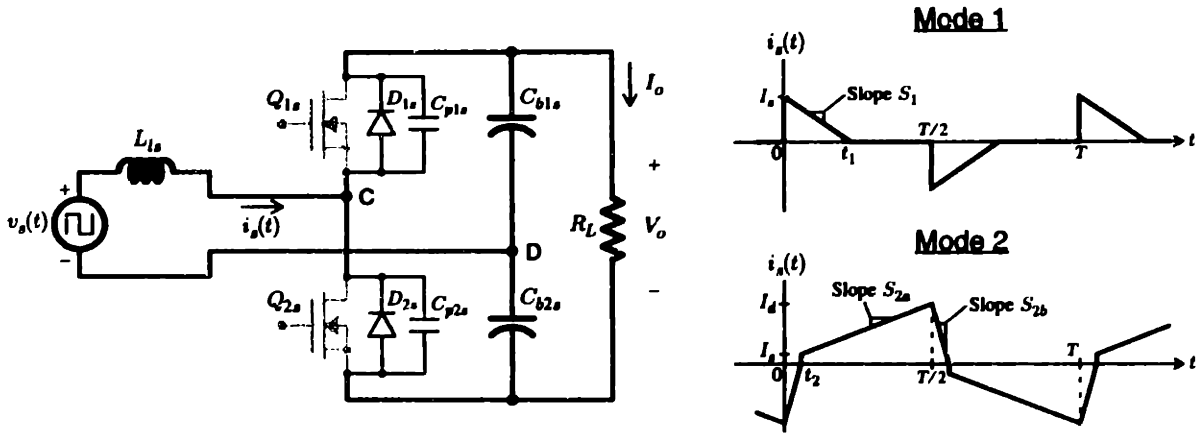



**Figure 3.18:** Two operating modes for the half-bridge rectifier with a capacitive-only filter.

Mode 1 occurs when the output is lightly loaded. Since $i_s(t)$ is odd harmonic (i.e., its positive and negative half-cycles are mirrors of each other), only the positive half-cycle needs to be described. As sketched in Figure 3.18, the secondary current begins the half-cycle at a level $I_s$. It then ramps quickly to zero so that the time-average relationship in (3.15) is maintained. The negative slope of $i_s(t)$ is determined by the voltage difference across the leakage inductance $L_{ls}$.

$$S_1 = (V_s - V_o/2)\frac{1}{L_{ls}} \tag{3.16}$$

For $S_1$ to be negative $V_o$ must be greater than $N_2 V_{bst}$. Thus, the output voltage actually lifts instead of droops at light load. Given the starting value $I_s$ and the slope $S_1$, the time average of $|i_s(t)|$ can be found from the average height of the Mode 1 waveform between $t = 0$ and $t = T/2$. The Mode 1 output current is

$$I_o = \frac{\langle i_s(t)\rangle}{2} = -\frac{I_s^2}{2S_1 T} = \frac{I_s^2 L_{ls}}{2(V_o/2 - V_s)T} = \frac{C_C V_o^2}{(V_o - 2V_s)T}, \tag{3.17}$$

where $T$ is the switching period $1/f_s$, and $C_C = C_{p1s} + C_{p2s}$. Equation (3.17) can be solved to find $V_o$ for Mode 1 as,

$$\text{(Mode 1)} \qquad V_o = \frac{I_o T}{2C_C}\left(1 - \sqrt{1 - \frac{8V_s C_C}{I_o T}}\right). \tag{3.18}$$

A mode transition occurs when the slope $S_1$ becomes shallow enough that $i_s(t)$ is no longer discontinuous. In other words, when $t_1$ in Figure 3.18 exceeds $T/2$, the secondary current enters Mode 2. This amounts to a condition on $I_o$, whereby the operation is Mode 2 if

$$I_o \geq \frac{V_s T}{2T\sqrt{L_{ls}/C_C} - 8L_{ls}}. \tag{3.19}$$

The shape of $i_s(t)$ during Mode 2 operation is also shown in Figure 3.18. The Mode 2 current appears shifted in phase with respect to Mode 1 because the diode commutation does not occur until $i_s(t)$ crosses zero, which is $t_2$ seconds after $v_s(t)$ changes state. From time $t_2$ to $T/2$, $i_s(t)$ ramps from $I_s$ to $I_d$ with a slope $S_{2a}$. After $T/2$, $v_s(t)$ changes state again, and $i_s(t)$ slopes down at $S_{2b}$, returning to zero at time $T/2+t_2$. The slopes are again related to the voltage difference across $L_{ls}$.

$$S_{2a} = \frac{(V_s - V_o/2)}{L_{ls}} \qquad S_{2b} = \frac{(-V_s - V_o/2)}{L_{ls}} \tag{3.20}$$

The time $t_2$ is related to the slopes as follows:

$$t_2 = \frac{-S_{2b}T/2 - I_s}{S_{2a} - S_{2b}}. \tag{3.21}$$

The time average of $|i_s(t)|$ can be found from the average height of the Mode 2 waveform between $t = t_2$ and $t = T/2+t_2$. The Mode 2 output current is

$$\begin{aligned} I_o &= \frac{\langle i_s(t)\rangle}{2} = \frac{(I_s - S_{2b}t_2)(T-2t_2)}{4T} - \frac{(S_{2b}t_2^2)}{2T} \\ &= \frac{1}{4}\frac{-4L_{ls}C_C V_o^2 + (8TV_s\sqrt{L_{ls}C_C} - V_s T^2)V_o + 2(V_s T)^2}{TL_{ls}(6V_s - V_o)}. \end{aligned} \tag{3.22}$$

Solving (3.22) for $V_o$ yields,

$$\text{(Mode 2)} \quad V_o = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \qquad \begin{aligned} a &= 4L_{ls}C_C \\ b &= T(V_s T - 4I_o L_{ls} - 8V_s\sqrt{L_{ls}C_C}). \\ c &= 24I_o L_{ls}V_s T - 2(V_s T)^2 \end{aligned} \tag{3.23}$$

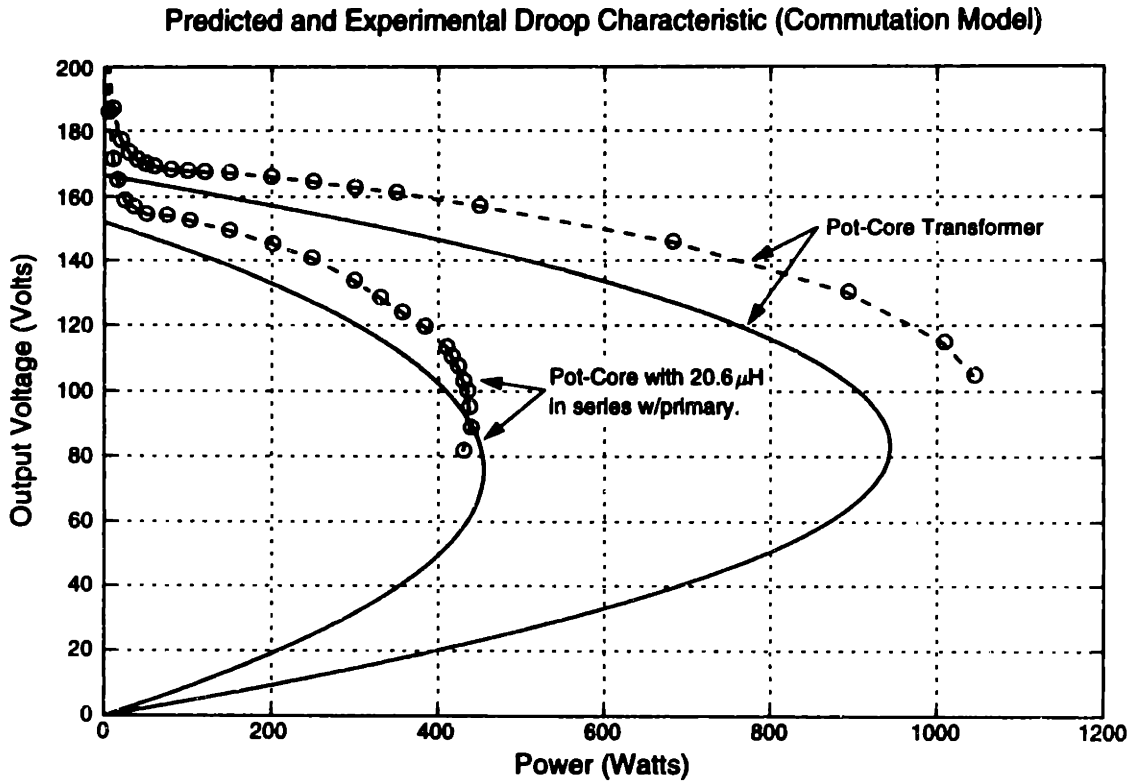



**Figure 3.19:** Predicted and experimental droop using the commutating reactance model.

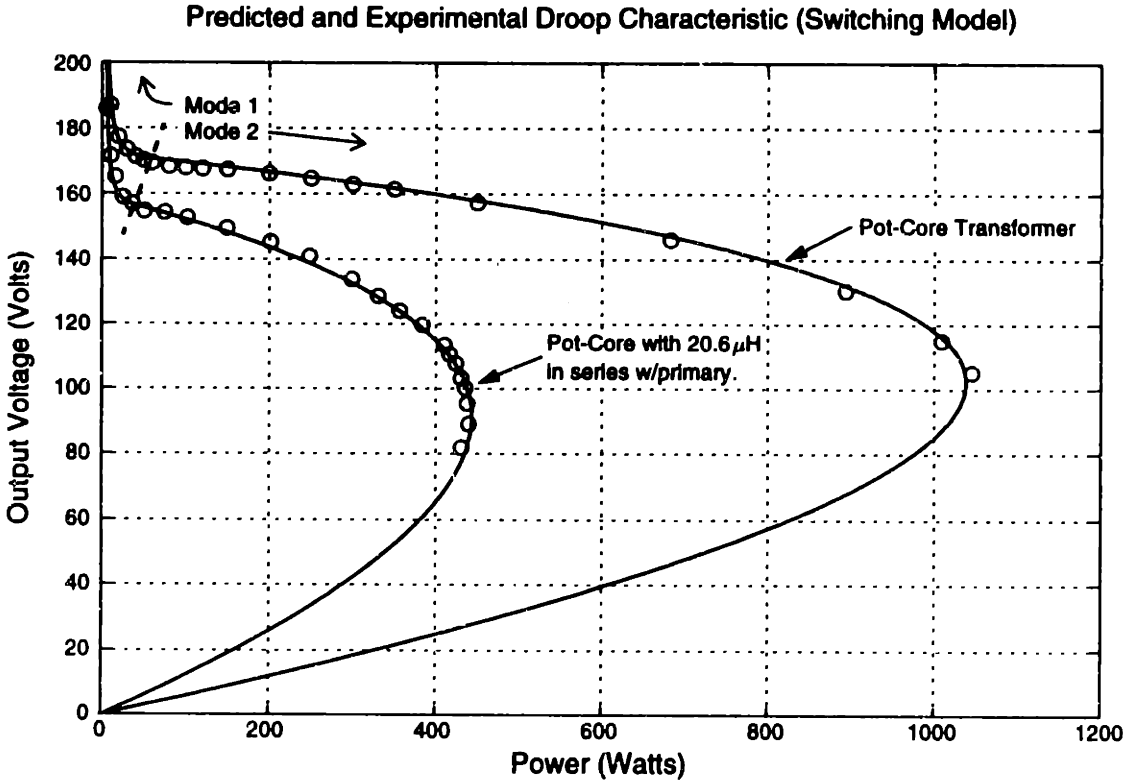



**Figure 3.20:** Predicted and experimental droop using the Mode 1/2 switching model.

The results for Modes 1 and 2 from (3.18) and (3.23) can be used to generate a droop characteristic curve of $V_o$ versus $P_o$. However, the predicted voltage droop is larger than that measured experimentally. This is not surprising because the idealized waveforms in Figure 3.18 do not account for voltage fluctuations at node D due to the finite value of the filter capacitors. Ideally, the voltage at node D should be precisely $V_o/2$. In practice, there is an AC ripple component at the switch frequency. The effect of this ripple is to slightly increase the effective output voltage from the converter. Thus, a precise match between the predicted and experimental data is possible if an additional factor is added to $V_o$ as defined below. The effective output voltage is

$$V_{o,eff} = V_o + k_r \frac{T}{C_f} I_o - 2V_{dd}, \tag{3.24}$$

where $C_f = (C_{b1s} + C_{b2s})/2$ and $V_{dd}$ represents the voltage drop across the output rectifiers. A value of $k_r$=2.30 yielded the predicted curves in Figure 3.20. The predictions match closely with the experimental data.

The same technique was used to obtain the predicted curves in Figure 3.21. The figure shows the predicted output voltage versus power curves for the AMP/M.I.T. inductive coupling. The leakage inductance of this coupling (see Table 2.3) is almost half that of the pot-core transformer. As a result, it is possible to transfer considerably more power across the coupling. With $V_{bst}$ at 390 V the leakage limited power capability is over 4 kW.

The shaded region in Figure 3.21 outlines the range the converter might operate over in order to charge a 120-V battery rack at a power level from 0 to 2 kW. A typical lead-acid battery experiences a 30% rise in voltage during charge. The data in the figure predicts that the variation in $V_{bst}$ from 267 V to 358 V (a 34% increase) spans the load range. Thus, the voltage droop of the converter leads to a slight 4% increase in the voltage range at the boost converter output.

### 3.3.5 Series-Resonant Operation

As just shown the leakage inductance of the inductive coupling tends to limit the maximum power capability of the DC/DC stage. In a single-stage inductively-coupled system this limit can be mitigated by carefully optimizing the connector design for low leakage.

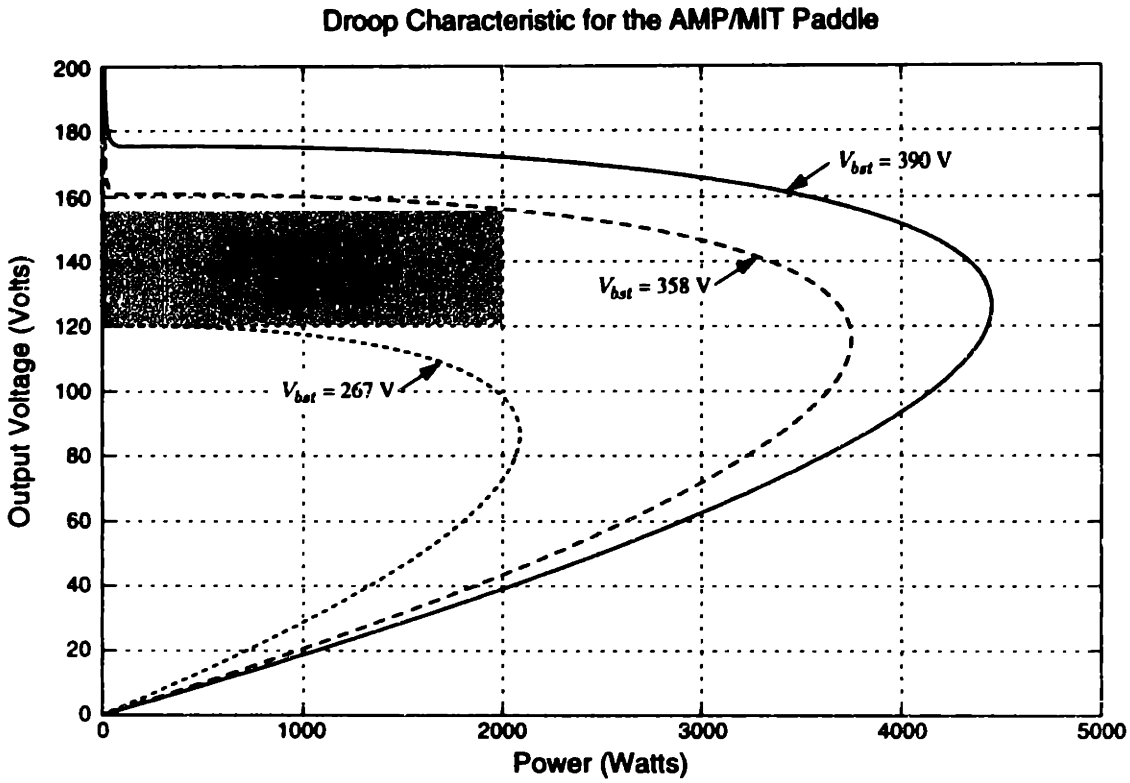



**Figure 3.21:** Voltage droop characteristic for the AMP/M.I.T. prototype "paddle."

However, it is possible to exceed the predicted power capability for a fixed leakage inductance by inserting a series capacitor. This series-resonant element effectively lowers the impedance of the leakage inductance and increases the power capability of the converter.

The half-bridge topology of Figure 3.12 can be easily made series-resonant by properly scaling capacitors $C_{b1}$ and $C_{b2}$ to resonate with $L_{lp}$. Provided that the resonant frequency is maintained slightly below the switching frequency, the primary current will lead the voltage, and ZVS can still be achieved. The effect of the series-resonant capacitor on the droop characteristic of the converter is discussed in Part II of this thesis.

## 3.4 Summary

This chapter described the hardware design and operation of the first of two prototype inductively-coupled systems. The system uses two stages of power conversion: an interleaved HPF boost converter and a half-bridge DC/DC converter. Both stages were described, and their operation was analyzed. Experimental data from the prototype stages was compared with expected results. The results were excellent. The prototype system

was shown to be highly efficient, with first- and second-stage efficiencies of 96.3% and 93.4%. The combined efficiency, including logic and gate-drive losses, is 89% at 1000 W. The understanding of the hardware aspects of the system provided in this chapter allows Chapters 5 and 6 to focus solely on algorithms for digital control.

# Chapter 4

# Bidirectional Prototype

This chapter describes the hardware design and operation of a 600-W bidirectional inductively-coupled system. This system demonstrates capabilities unavailable on the first prototype, including bidirectional power transfer. Circuit topologies for this second prototype are presented, and their operation is analyzed. Experimental data from the prototype system is then compared with expected results. This chapter, as with Chapter 3, provides an understanding of the hardware aspects of the system that will clarify the development of digital control algorithms in Chapters 5 and 6.

## 4.1 System Overview

The unidirectional hardware system described in Chapter 3 performed as designed. A second prototype was built to explore topologies and design possibilities not addressed by the first prototype. This prototype, like the first, was built with the EV charging application in mind. However, the circuit topologies are by no means limited to this application.

The second prototype is capable of bidirectional power flow across the inductive coupling. Bidirectional power flow expands the possible applications for inductively-coupled power transfer. It becomes possible to consider complex loads and electromechanical devices, which source as well as sink power. The EV charger is one example. Traditionally, power is transferred into a battery, raising its state of charge. Reversing this power flow makes it possible to discharge the battery, returning the power back to the utility or perhaps to an external load.

It has been shown that, for certain battery technologies, the lifetime of a battery can be extended by properly discharging before recharging [64, 68]. This ability may prove even

more vital as new high-density battery chemistries are developed. A bidirectional inductively-coupled charger could provide a safe and reliable "total charging solution."

The second prototype was designed with a number of specific goals in mind:

1. Demonstrate *bidirectional* battery charging and discharging.
2. Charge/discharge at a 600-W level from single-phase 120 $V_{AC}$.
3. Source or sink power from the utility with unity power factor.
4. Operate with high efficiency.
5. Demonstrate flexible microprocessor-based digital control.
6. Actively cancel 120-Hz ripple components during battery charging.
7. Implement an inverter mode to synthesize 60-Hz AC for external loads.

Although some of these goals overlap with those for the first prototype, many do not. In particular, bidirectional power flow, 120-Hz ripple cancellation, and AC-inverter operation are specific to this prototype. The 600-W power level was selected as a credible number for demonstration purposes, and a two-stage power-electronic system that meets all the above design goals has been developed for that power level. Every effort was made to interface the new power-electronic stages with the existing digital microcontroller board, so that as little redesign as possible was necessary. With a few software changes, the two new bidirectional power-electronic stages can simply act as drop in replacements for the earlier unidirectional stages. This was possible with only minor design compromises. Sections 4.2 and 4.3 describe the two stages of the bidirectional inductively-coupled power system in detail.

## 4.2 Bidirectional Boost-Buck-Inverter

The first stage, or utility interface, of the second prototype is a unique topology, which can be operated in three distinct modes. Forward power flow is possible in a boost mode, and reverse power flow is possible in a buck mode or an AC-inverter mode. Although slightly more complex than the unidirectional stage presented in Chapter 3, this stage is considerably more flexible.

### 4.2.1 Converter Topology

A simplified schematic of the converter topology is shown in Figure 4.1. The basic circuit uses two half-bridge legs to form an H-bridge, which connects to the AC utility through an inductor $L_1$. The topology has many advantages over alternative designs, including:

1. Bidirectional power flow.
2. One high-frequency switching leg and one line-frequency switching leg.
3. Low conduction losses. (Only two semiconductor drops instead of three.)
4. No input full-bridge rectifier.
5. An $N$-cell interleaved topology uses only $2N+2$ switches.

In order to accomplish bidirectional power flow, the conventional boost topology (recall Figure 3.5 from Chapter 3) must be extended. A traditional approach is to replace the diode $D_1$ with a high-frequency switch complementary to $Q_1$ and to replace the full-bridge rectifier at the input with an H-bridge of line-frequency switches, which allow for unfolding of the waveform during reverse operation. In total, the number of active switches for a single-cell converter would increase from one to six.

With only four active switches, the topology presented in Figure 4.1 offers the same performance and better efficiency. The rectifier bridge at the input of a conventional topology has been removed entirely. The inductor $L_1$ instead connects directly to the AC source, and the unfolding operation is performed by two line-frequency switches $Q_3$ and

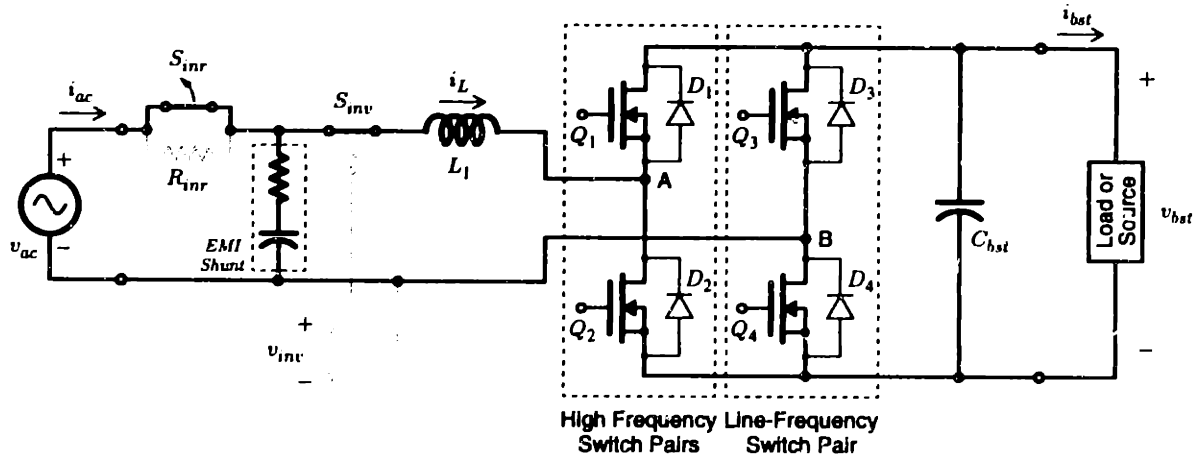



**Figure 4.1:** Simplified schematic of the boost/buck topology.

$Q_4$. The resulting circuit has lower switching and conduction losses than its traditional counterpart. Since two of the four switches are operated at the line-frequency, their switching losses are negligible. Also, the current path from input to output has only two semiconductor drops. A conventional boost topology, unidirectional or bidirectional, has three. Thus, with MOSFET switches the conduction losses can be significantly reduced.

In addition, the topology in Figure 4.1 can easily be extended to include multiple interleaved switching cells. A simplified schematic showing three cells appears in Figure 4.2. The interleaved topology requires that the wave-shaping switches ($D_1$ and $D_2$) and the inductor $L_1$ are duplicated for each cell. The line-frequency "unfolding" switches $Q_3$ and $Q_4$ do not need to be duplicated. Therefore, the number of active switches for a $N$-cell interleaved converter is $2N+2$, and the current through the $2N$ high-frequency switches is derated by a factor $N$. This topology could easily take advantage of modern integrated power semiconductors, which combine as many as six IGBT switching devices into a single package. This would minimize the parts count and keep the cost competitive with traditional designs. At the same time all of benefits described in Chapter 3—ripple amplitude, power density, and efficiency—still apply.

The single-cell prototype converter shown in Figure 4.1 was built and used to demonstrate all the desired conversion techniques. The new converter was switched at 25 kHz so that a common PWM control circuit could be shared by both the unidirectional and bidirectional UPF interfaces. This made it possible to considerably reduce development time by connecting both prototypes to the same digital microcontroller board. The following

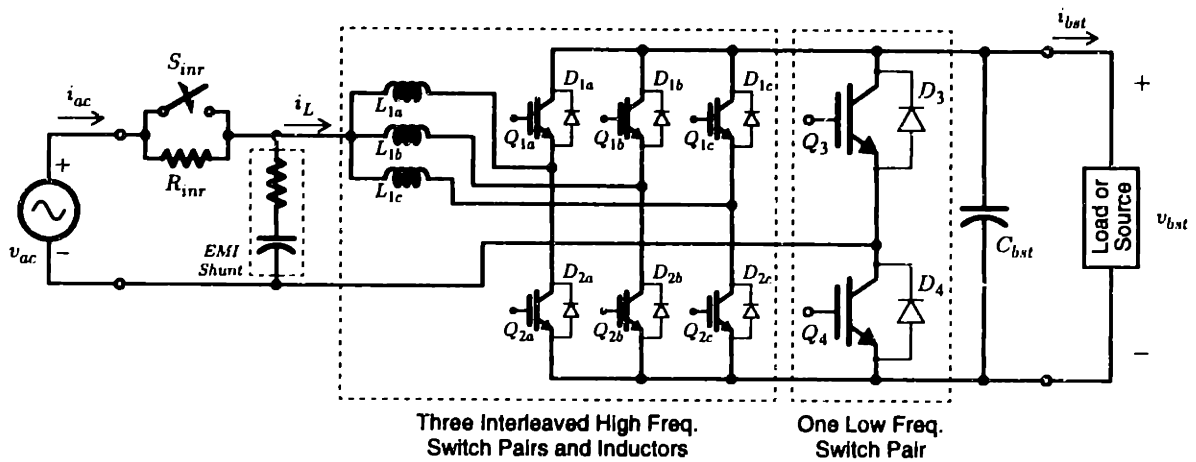



**Figure 4.2:** Simplified schematic of an *interleaved* boost/buck topology.

three subsections describe in detail the three operating modes of the prototype boost-buck-inverter.

### 4.2.2 Boost- and Buck-Mode Switching

The boost and buck switching modes are used to accomplish power transfer out of and into the AC utility with near unity power factor. Because the voltages at the input and output of the converter are nearly identical for both modes of operation, the switching patterns are closely related. In fact, the same inner-loop controller is used to perform power-factor correction in both directions. The examples below illustrate the switching methodologies. Experimental results are provided later in subsection 4.2.3.

The boost mode of the converter takes the AC voltage from $v_{ac}$ and converts it to a DC voltage at $v_{bst}$. Ideally, the current $i_{ac}$ is modulated to track the shape and phase of $v_{ac}$ in order to achieve unity power factor. The waveforms in Figure 4.3 illustrate how this can be accomplished. During the positive half-cycle of $v_{ac}$, MOSFET $Q_4$ is held on and $Q_3$ is held off. Consequently, node B is pulled low, effectively connecting the negative terminals of $v_{bst}$ and $v_{ac}$. With $Q_1$ held off, $Q_2$ is switched at a constant frequency and a duty ratio $d(t)$. When $Q_2$ is on, the current $i_L$ ramps up at a rate $v_{ac}/L_1$. When $Q_2$ is off, the current commutates to diode $D_1$ and ramps down at a rate $(v_{bst}-v_{ac})/L_1$. During the negative half-cycle of $v_{ac}$, the pattern is altered slightly to allow $i_L$ to go negative. MOSFETs $Q_3$ and $Q_4$ are toggled, bringing node B up to $v_{bst}$. By reversing the roles of $Q_1$ and $Q_2$, the current $i_L$ is identically controlled in the negative direction.

The example waveforms in Figure 4.3 illustrate an important source of distortion. Because the prototype is non-interleaved it operates in CCM to ensure low input-current ripple. However, a large input inductor is required, and cusp distortion results. Cusp distortion, which is highlighted in Figure 4.3, occurs when the current reference exceeds the available slew rate $v_{ac}/L_1$. This occurs just after $v_{ac}$ crosses zero. The overall affect on the power factor is shown to be negligible in the next subsection. Cusp distortion is largely non-existent in an interleaved converter because the individual conversion cells have high slew rates due to their much smaller inductors. This was illustrated in Chapter 3.

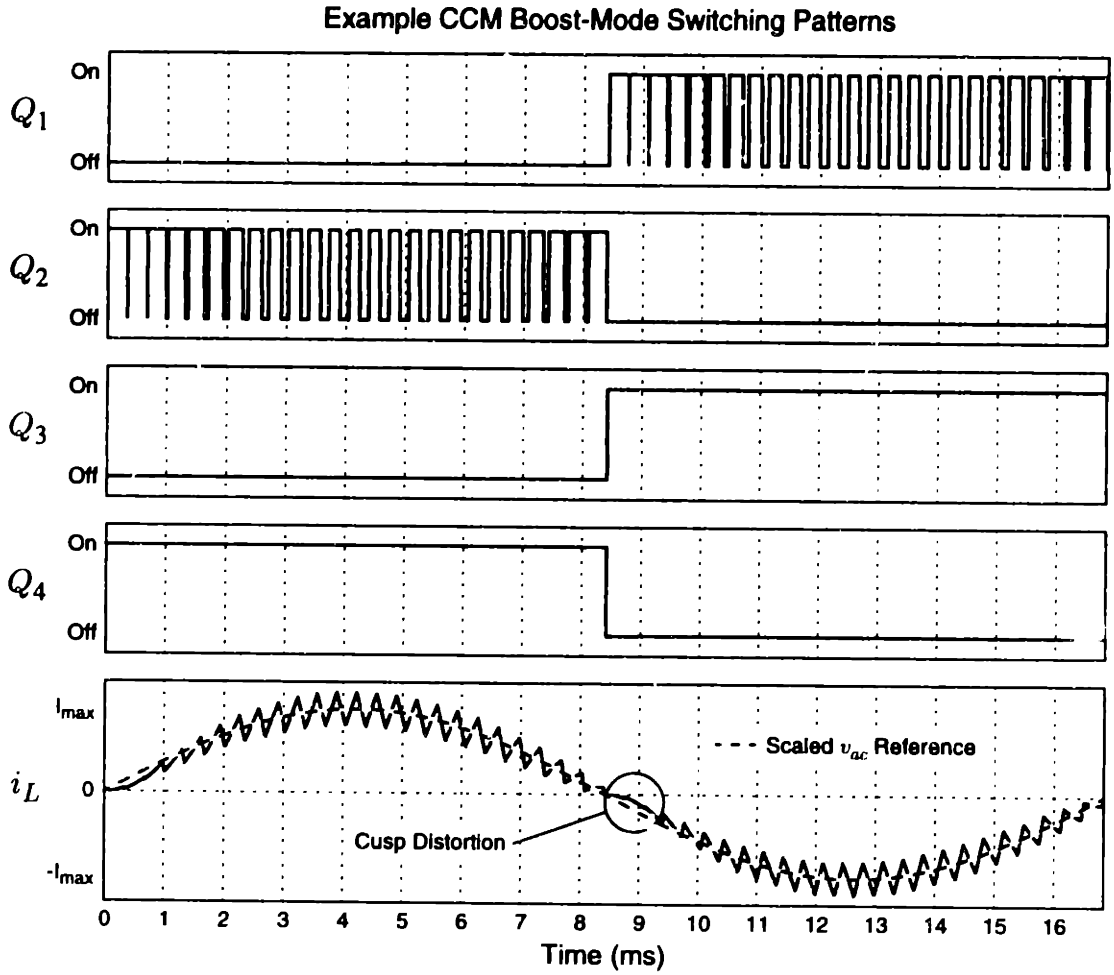



**Figure 4.3:** Simulated switch patterns for boost-mode operation in continuous conduction mode.

The buck mode of the converter operates in a similar fashion, but the power flow is reversed. The waveforms in Figure 4.4 illustrate how this can be accomplished. During the positive half-cycle of $v_{ac}$, current is fed into the AC source. As with the boost mode, node B is held low by MOSFET $Q_4$, but now switch $Q_1$ is modulated by a duty ratio $d(t)$. When $Q_1$ is on, the current $i_L$ ramps in a negative direction with a slope $(v_{bst} - v_{ac})/L_1$. When $Q_1$ is off, the free wheeling current is carried by diode $D_2$, and $i_L$ ramps positive at a rate $v_{ac}/L_1$. During the negative half-cycle of $v_{ac}$, the pattern is reversed. MOSFETs $Q_3$ and $Q_4$ are toggled, and the roles of $Q_1$ and $Q_2$ are reversed.

The example waveforms in Figure 4.4 illustrate the operation. A similar cusp distortion effect can be observed just before the zero crossing. In this case the current reference

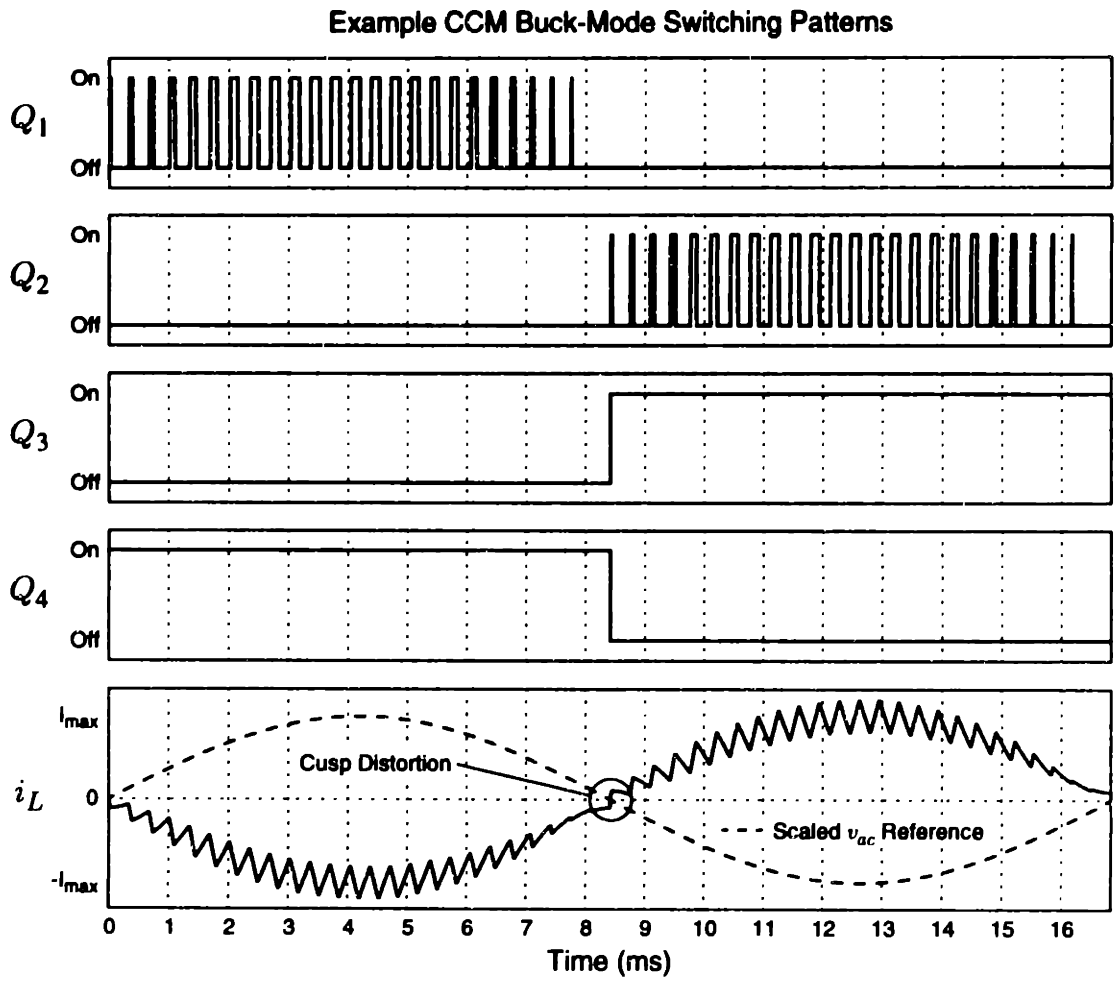



**Figure 4.4:** Simulated switch patterns for buck-mode operation in continuous conduction mode.

falls below the available negative current slew rate $v_{ac}/L_1$. Again, the overall affect on the power factor is shown to be negligible in the next subsection.

The operation of the hardware prototype is identical to the example waveforms in Figures 4.3 and 4.4 except for the switching frequency. The prototype operates at a switching frequency of 25 kHz. This frequency is relatively low for a single-cell converter, and a large 2 mH inductor was required for $L_1$. Due to the size and current requirement of this inductor, it was necessary to place six small inductors in series to achieve the required energy storage. The 25 kHz frequency was selected for compatibility with existing control electronics. Each cell of the interleaved converter discussed in Chapter 3 operated at 25 kHz. Therefore, it was possible to use the same PWM controller for this prototype. Complete details of the hardware design are available in Appendix C.

### 4.2.3 Boost/Buck-Mode PFC

An averaged current-mode controller was used to control the duty ratio $d(t)$ of the switches during forward boost-mode operation and reverse buck-mode operation. This controller provides effective power-factor correction in both directions. The controller was implemented with an analog circuit, identical to that described in subsection 3.2.3. Because the forward and reverse operating modes are so similar, the same controller can be used for both modes. A minor amount of additional logic is used to route the switch signals to the appropriate gates. Design equations for the controller are given in Appendix A.

As stated in subsection 3.2.4, the goal of power-factor correction is to achieve unity power factor. Figure 4.5 shows the experimental results measured from the system during forward, boost-mode operation. The waveforms demonstrate an experimental power factor of 0.997. The measured THD of the current waveform is 4.44%. The waveforms in the figure show the converter operating from 120 $V_{AC}$ supplied by the HP 6834B AC source/analyzer. The expanded region in Figure 4.5 (b) shows the current ripple measured before the EMI shunt filter. The filter effectively bypasses the 25-kHz ripple current so that it does not pass through to the utility. Figure 4.6 graphs the magnitude of the input current harmonics along with the IEC 555-2A limits. The magnitudes of the input current harmonics all fall well below the IEC limits.

Figure 4.7 shows the experimental results measured from the system during reverse, buck-mode operation. Notice that the current waveform is 180 degrees out of phase with the voltage waveform. This indicates negative power, meaning the converter is actually returning power from the batteries to the AC utility. Approximately 210 W is shown returning to the utility with an experimental power factor of −0.996. The measured THD of the current waveform is 5.43%. However, this distortion number is slightly misleading since the input voltage is not a perfect sinusoid.

Because the HP 6834B AC source/analyzer used in the previous measurements is incapable of sinking power, it was necessary to make the PFC measurements with the unit directly connected to the AC utility. The voltage distortion observed in Figure 4.7 (a) is caused by a variety of loads (computers etc.) that also receive power from the AC utility and draw current waveforms with substantial harmonics content. The end result is that the

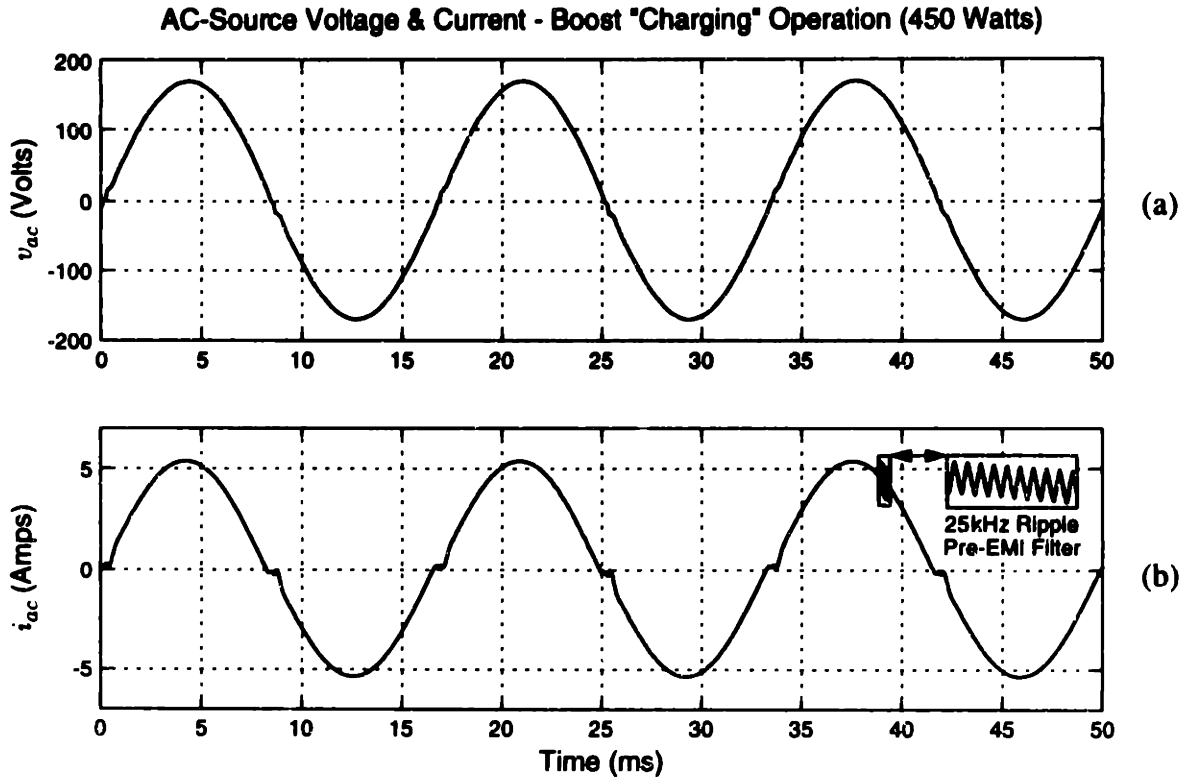



**Figure 4.5:** Experimental input voltage and current with PFC enabled. ($k_p$=0.997)

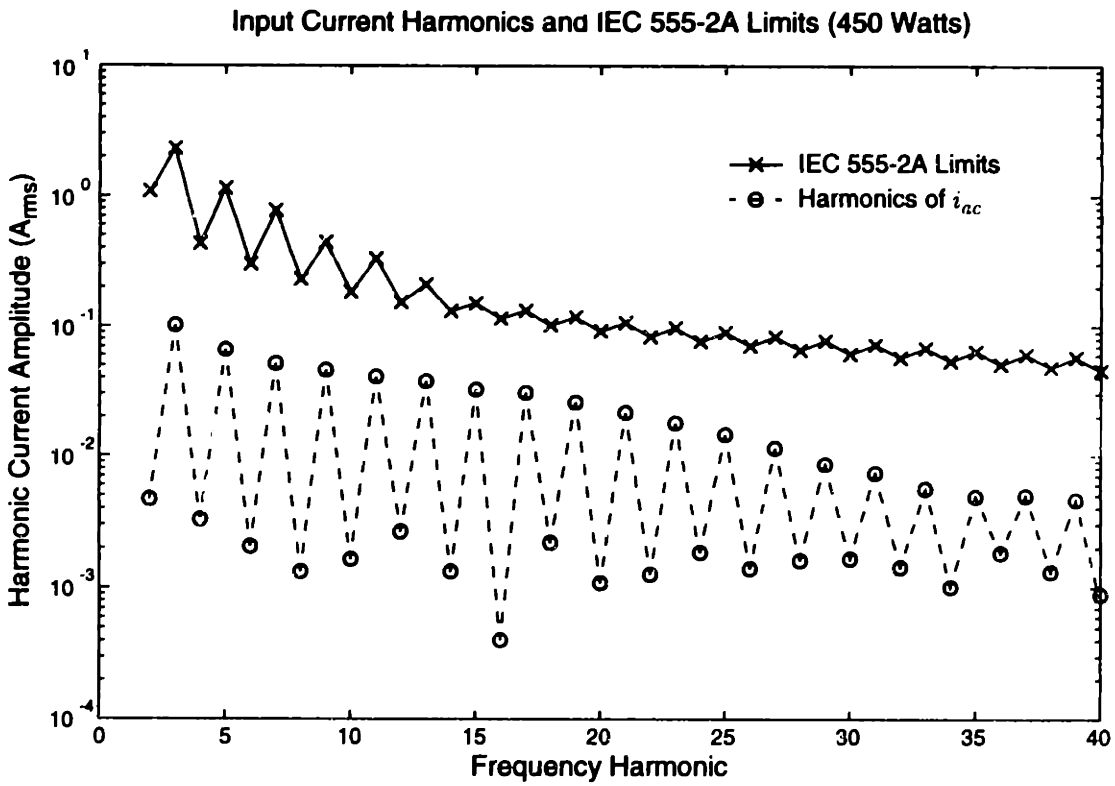



**Figure 4.6:** Magnitude of the input current harmonics versus the IEC 555-2A limits.

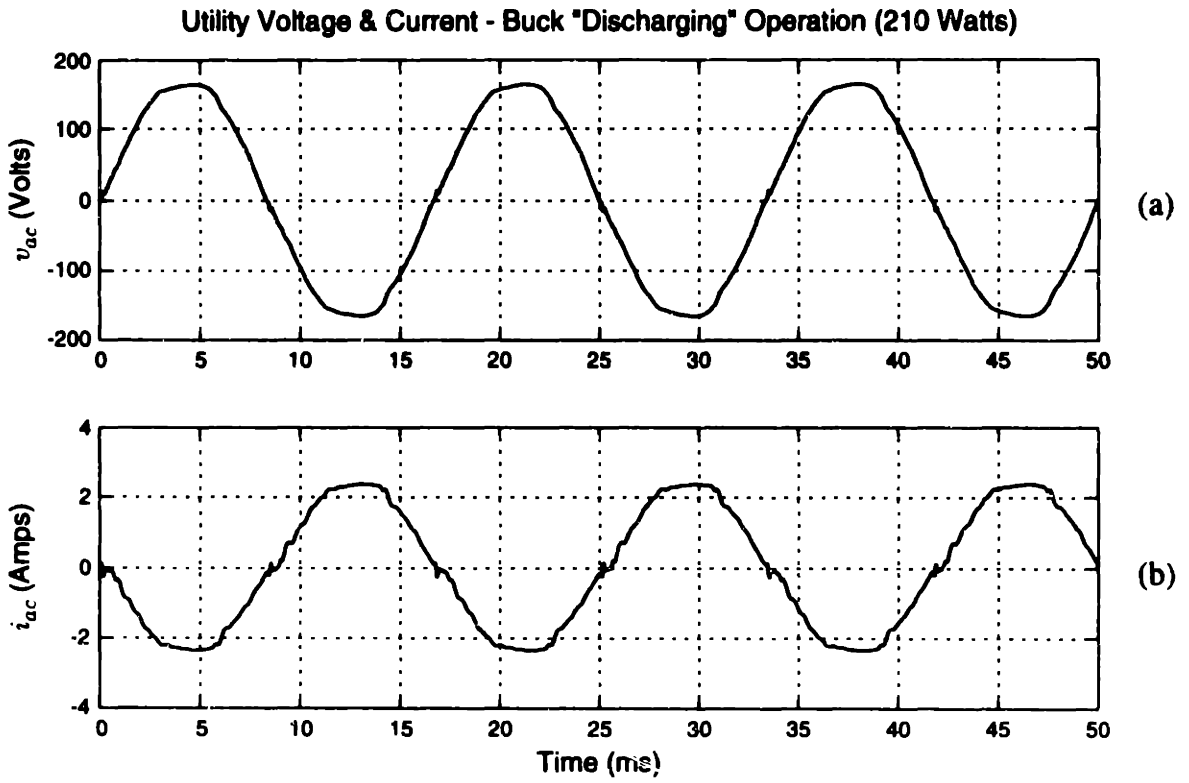



**Figure 4.7:** Experimental utility voltage and current during buck operation. ($k_p = -0.996$)

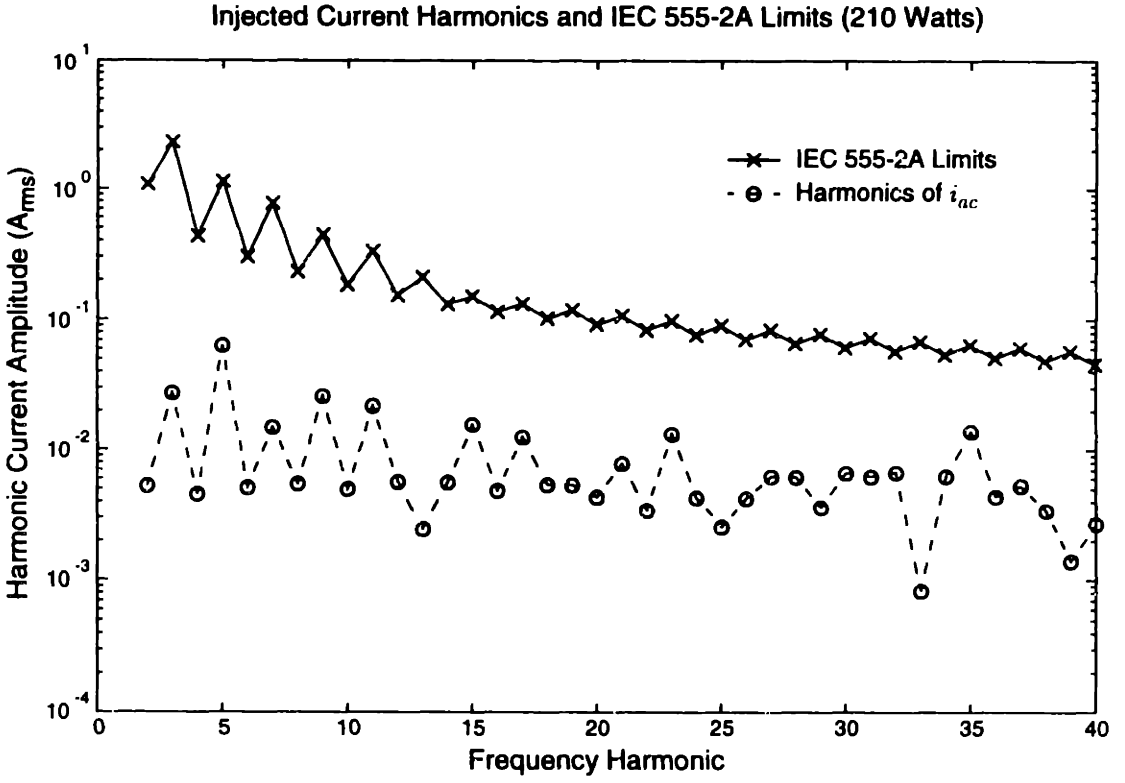



**Figure 4.8:** Magnitude of the output current harmonics versus the IEC 555-2A limits.

current harmonic measurements are worse than could be expected from a clean AC source. Nevertheless, the input current harmonic magnitudes plotted in Figure 4.8 still indicate compliance with the IEC 555-2A limits.

### 4.2.4 AC-Inverter Mode

An optional mode of operation for the converter is the AC-inverter mode. In this mode, power is extracted from the batteries and used to create a synthesized 60-Hz AC waveform. This mode could be used, for example, to power household AC equipment or supplement utility power during a short-term outage. Because the AC utility normally supplies a fixed voltage between $110-120\ V_{AC}$, where the load power is determined by the load, there is no reason for feedback control. It is generally possible to synthesize the 60-Hz voltage waveform using an open-loop switching technique.

Figure 4.9 illustrates the circuit configuration for the AC-inverter mode. The relay $S_{inv}$, shown in the figure, disconnects the existing AC utility and routes the two half-bridge midpoints (nodes A and B) to the external AC load. The converter transfers power from the DC voltage at $v_{bst}$ to the external load or loads connected across the $v_{inv}$ terminals. The boost/buck inductor $L_1$ does not interfere with the inverter operation. In fact, it provides a small amount of filtering, which helps to smooth the output waveforms. The MOSFET switches $Q_1$ through $Q_4$ are modulated to produce a tri-state voltage waveform measured between nodes A and B of the full-bridge.

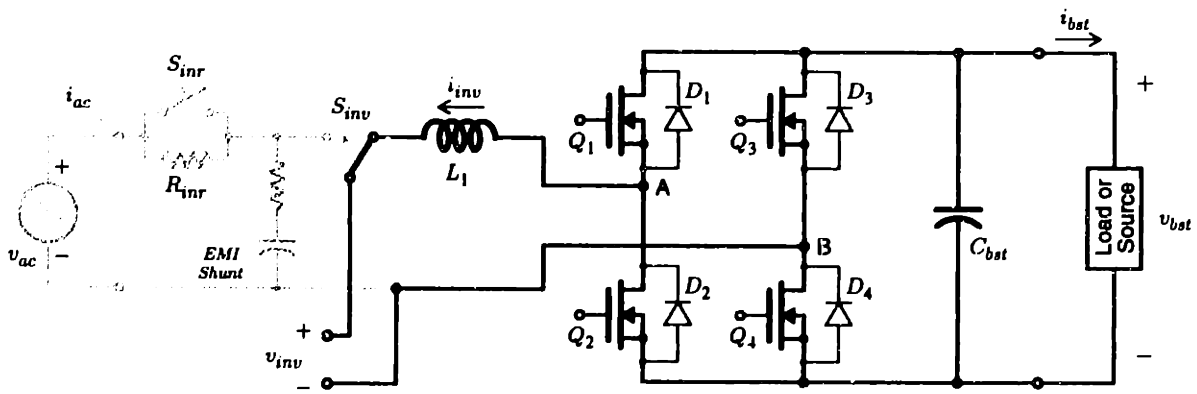



**Figure 4.9:** Simplified schematic of the inverter topology.

The three voltage states are zero, $V_{bst}$, or $-V_{bst}$, where $V_{bst}$ is the DC component of the voltage across the $v_{bst}$ terminals; the small 120-Hz AC component is heavily filtered by $C_{bst}$ and can safely be ignored. In keeping with the switching methods described for the boost and buck operation, switches $Q_3$ and $Q_4$ change state only once per half-cycle of the synthesized 60-Hz output. This allows low-frequency devices to be used. Switches $Q_1$ and $Q_2$ are switched at a high frequency to accomplish the sinusoidal modulation.

The full-bridge topology shown in Figure 4.9 is commonly used in line-frequency and low-frequency inverter applications, such as uninterruptible power supplies, solar-cell inverters, and motor drives. The traditional switching methodology is a PWM technique, where a fixed-frequency duty-ratio command is used to gate the MOSFETs $Q_1$ and $Q_2$. Since the frequency is fixed, a set number of switch transitions occur during each cycle of the reference waveform. These switch transitions are "lossy" because zero-voltage switching is not guaranteed by this topology. In addition, the output waveform will have *significant* harmonic energy around and above the switching frequency. In some cases the amplitude of the switching frequency harmonic can be larger than the fundamental. A low-pass filter can be used to attenuate switch-frequency harmonics, but low-frequency harmonics in the passband still affect the waveform quality.

Several methods have been proposed to minimize the low-frequency harmonic content of PWM circuits [63, 85]. The methods use analytical techniques, such as Fourier or Walsh function analysis, to express the harmonic content in terms of the PWM duty ratio. The resulting equations are then solved to minimize distortion. In many cases the resulting fixed-frequency PWM waveforms are discretized in time so that they may be implemented using digital circuitry [20]. These "programmed" PWM waveforms are essentially long, repeating, discrete-time switching sequences.

In general, switching sequences can be created with switch transitions that do not occur at a fixed frequency. A technique in [49], which eliminates harmonics by zeroing or "notching" a square wave, is one example. Other possibilities and design criteria are described in [18] and [62]. For example, a fixed-length sequence can be used to construct a discrete approximation to a reference sinewave. The number and position of the "ones" and "zeros" can be selected to minimize distortion without regard to the fixed-frequency

constraints. It is possible to create low distortion sinewave approximations that use fewer switch transitions per cycle than a comparable PWM pattern. It is also possible to tailor or place the harmonic content of a switching sequence for a particular application. The bit patterns of the binary sequence can be stored in a memory and clocked out at a fixed rate or generated in real time using a simple state machine or microcontroller.

An example 30-bit pattern is shown in Figure 4.10. Symmetry yields an underlying 15-bit pattern '001011111111010', which is inverted and repeated for each half-cycle of the waveform. If this example pattern is used with the circuit shown in Figure 4.9 to synthesize a 60-Hz sinusoid, then the bits are clocked out at a rate of 1800 Hz. The gate drives for MOSFETs $Q_1$ through $Q_4$ are related to the bit pattern as shown in the figure. Thus, very little additional logic is needed. The bottom trace in Figure 4.10 shows the resulting tri-state waveform measured across nodes A and B. The voltage inversion during the sec-

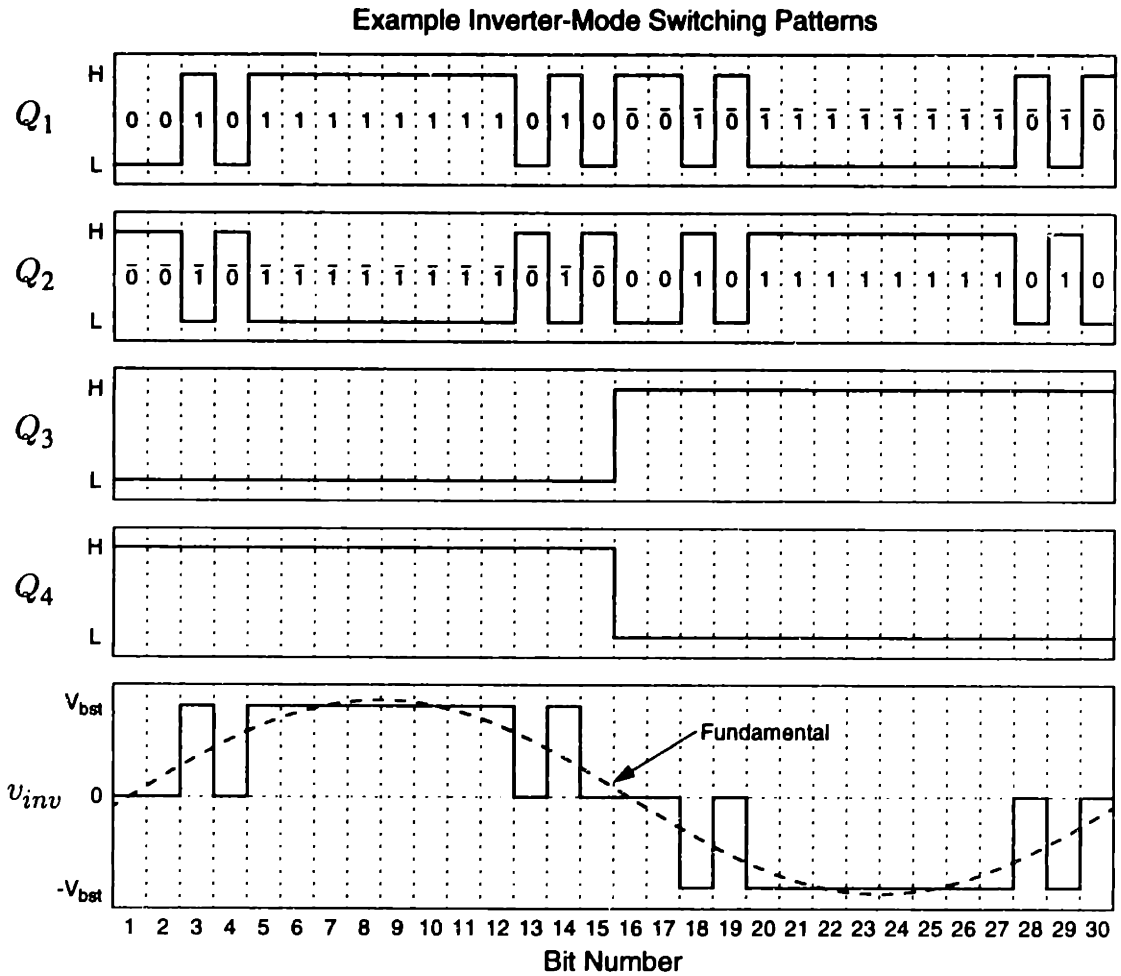



**Figure 4.10:** Example 30-bit inverter switching sequence.

ond half-cycle is due to node B's transition from 0 to $V_{bst}$ as $Q_3$ and $Q_4$ change state. The fundamental harmonic of this example waveform is a 60-Hz sinusoid with a peak amplitude approximately 6% greater than $V_{bst}$. The fundamental is sketched with a dashed line in the figure.

The 30-bit waveform in Figure 4.10 is special in that an amazing number of its higher-order harmonics are zero. Harmonic numbers 2, 3, 4, 5, 6, 8, 9, and 10 are all zero, and the seventh is only 12% of the fundamental. Not surprisingly, the distortion must show up somewhere, and indeed many of the remaining higher harmonics are significant in amplitude. Since so many of the harmonics just above the fundamental are zero or approximately zero, a mild low-pass filter will result in a low distortion output. In many cases the load itself is low-pass and provides some of this filtering. In general, longer sequences will allow many more harmonics to be eliminated and thus lessen the filter requirements.

The achievable amplitude of the target fundamental is proportional to the percentage of ones in the binary sequence. Therefore, it is possible in many cases to simply search all $N$-point sequences with the desired number of ones. However, long sequences quickly become unwieldy. The multiplicity of possible bit patterns has been shown to grow exponentially with $N$. Thus, an approximation technique must be employed. A fast algorithm has been developed by a colleague, Steven Shaw [96]. His algorithm employs simulated annealing to quickly minimize the number of transitions and harmonic content. Please see Appendix A for further discussion on the design of annealed switching sequences.

The algorithm in [96] was used to produce a 1024-point switching sequence for use with the prototype circuit. Appendix E lists a MATLAB function for generating the sequences. The number of ones in the sequence was set to 384, which yields a peak amplitude for the fundamental sinewave of $0.591V_{bst}$. This corresponds to approximately 110 $V_{AC}$ at the inverter output given a DC bus voltage of $V_{bst}$=265 V.[1] The sequence was coded into a finite-state machine built from programmable-logic devices. The result was a simple digital circuit that interfaced easily with the digital microcontroller and provided the gate-drive signals for all four MOSFET switches. Small, several hundred nano-second,

1. In practice, sequences of varying amplitudes could be pre-computed to accommodate for a range of DC voltage levels.

delays were built into the MOSFET gate-drive circuitry to prevent the possibility of shoot-through due to overlap edges. Appendix C presents complete hardware schematics.

The experimental output of the system is shown in Figure 4.11. A 1-HP shop vacuum was used as a load for the experiment. A shop vacuum was selected because its motor inductance acts as a low-pass filter. No external filtering was used except for the 2-mH boost inductor shown in the schematic. The top trace shows the inverter output voltage measured between nodes A and B (see Figure 4.9). The dashed line is the fundamental component of the voltage waveform. The lower axis shows the current $i_{inv}$ drawn by the shop vacuum. The solid line is the measured current using the inverter output as a source. The dashed line is the ideal current, which was measured using the HP 6834 AC source to provide a near-perfect sinusoidal voltage.

A fast-fourier-transform (FFT) analysis of the voltage waveform harmonics is provided in Figure 4.12. The figure demonstrates the extremely low harmonic content of the 1024-bit inverter output voltage. The first 40 harmonics have an amplitude smaller than 1% of the fundamental, and the higher harmonics are well distributed. The THD of the voltage waveform is approximately 113%. However, the small amount of filtering provided by the boost inductor and the load inductance results in only about 10% current distortion. Furthermore, a capacitor could be added across the $v_{inv}$ terminals of the converter to provide second-order filtering. A second-order low-pass filter centered at 2 kHz reduces the voltage THD to approximately 5%.

The experiments shown here demonstrate that the AC-inverter mode works as expected. Good harmonic elimination was achieved using a 1024-bit annealed switching sequence, and the distortion is comparable or better than traditional PWM techniques. The sequence chosen has only 300 switch transitions per 60-Hz cycle, which makes it efficient. A practical system might further refine the techniques by improving the optimization of the switching sequences and/or using a significantly longer sequence. It might, for example, be desirable to push the significant frequency harmonics above the audible range, thereby relaxing the constraints on the output low-pass filter.

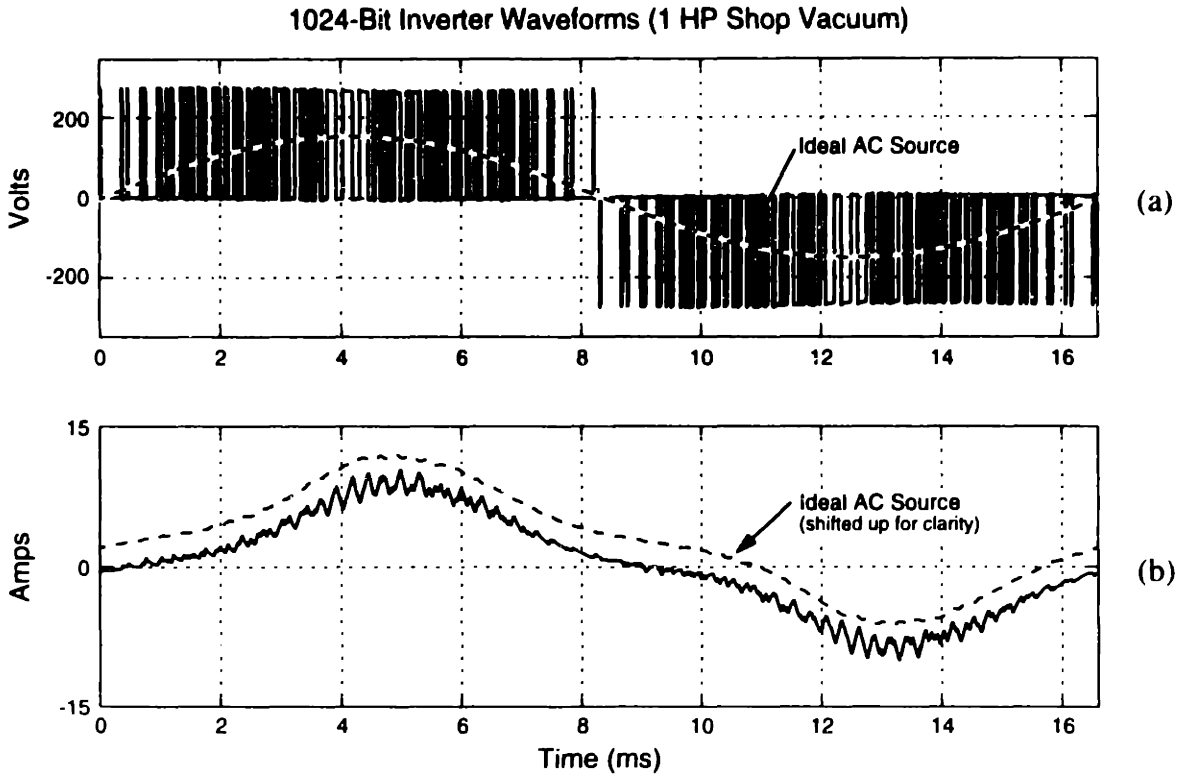



**Figure 4.11:** Experimental waveforms with the converter driving a 1-HP shop vacuum in inverter mode. (a) Inverter voltage. (b) Inverter current.

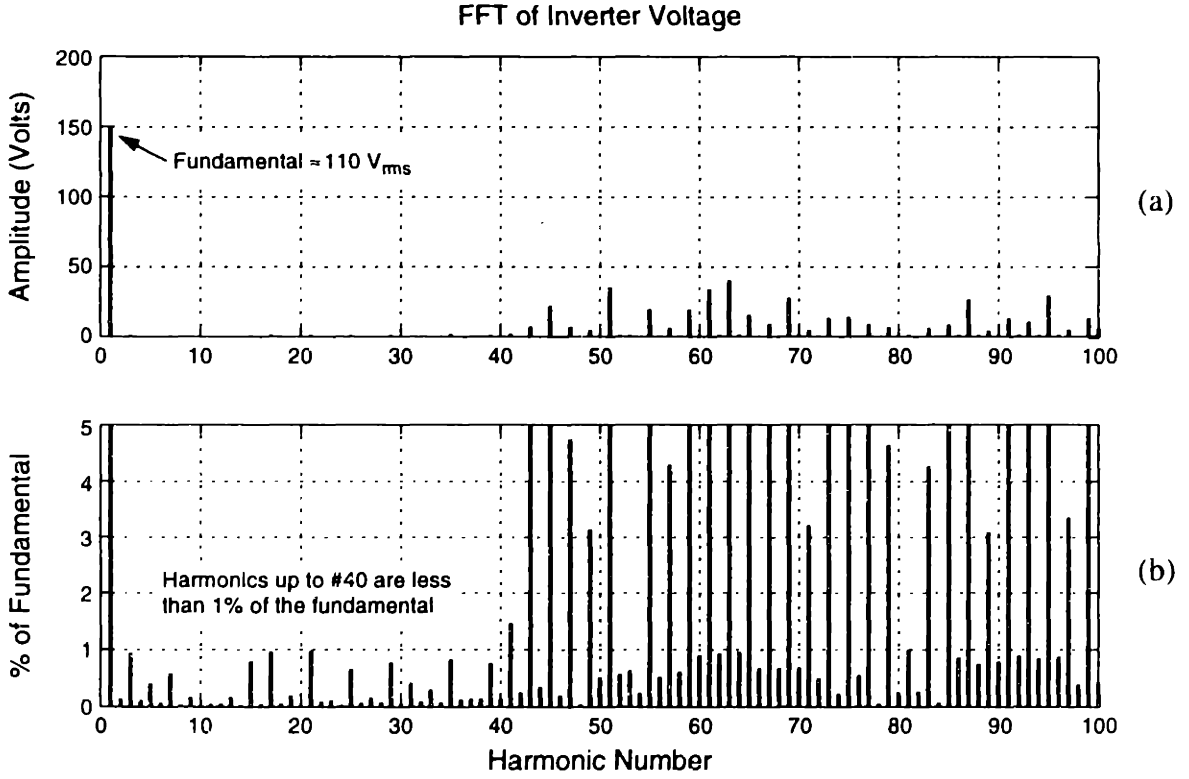



**Figure 4.12:** Fourier transform of the inverter voltage. (a) FFT with amplitude in volts. (b) An expanded view showing harmonics as a percent of the fundamental.

## 4.3 Symmetrical Full-Bridge DC-DC Converter

The half-bridge converter described in Chapter 3 meets all the design criteria for the first prototype. It is simple and very efficient. However, it operates with a constant (unity) duty ratio. Therefore, the 120-Hz ripple component of the voltage $v_{bst}$ passes through the DC/DC converter unaffected. This ripple voltage, when applied to the terminals of a battery, induces a significant current ripple into the battery. In order to demonstrate that this ripple could be actively cancelled, a second converter was necessary. At least two options were possible. The first was to make the half-bridge topology controllable by asymmetrically operating the duty ratio of the switches. The second was to design a new phase-shifted full-bridge topology. Both provide PWM control and operate with nearly identical efficiency.

The phase-shifted full-bridge topology is preferable. The output voltage of this topology is linearly related to the switch duty ratio, and this considerably simplifies active ripple cancellation. Furthermore, a single-chip PWM controller from Unitrode greatly simplifies the design process [110]. The asymmetrical half-bridge control technique, discussed in [42], is promising, but the output voltage is non-linearly related to the switch duty ratio and so would considerably complicate the ripple cancellation technique described below.

### 4.3.1 Converter Topology

A simplified sketch of the full-bridge converter topology is shown in Figure 4.13. Only the components necessary for unidirectional forward power flow are illustrated. MOSFETs $Q_1$ through $Q_4$ form a full-bridge inverter. The inverter drives the primary of the inductive coupling through a large DC blocking capacitor $C_{bl}$.[2] The four switches are gated at a frequency of 100 kHz using a phase-shifted PWM pattern. This pattern is described in the next subsection. The resulting inverter voltage $v_{AB}$ appears across the primary of the inductive coupling. As sketched in Figure 4.16, $v_{AB}$ is a square wave with a variable "on" time, which can be controlled by adjusting the phase shift between the gate-

2. The blocking capacitor $C_{bl}$ is selected so that its resonant frequency with the leakage inductance $L_l$ is well below the switching frequency of the converter.

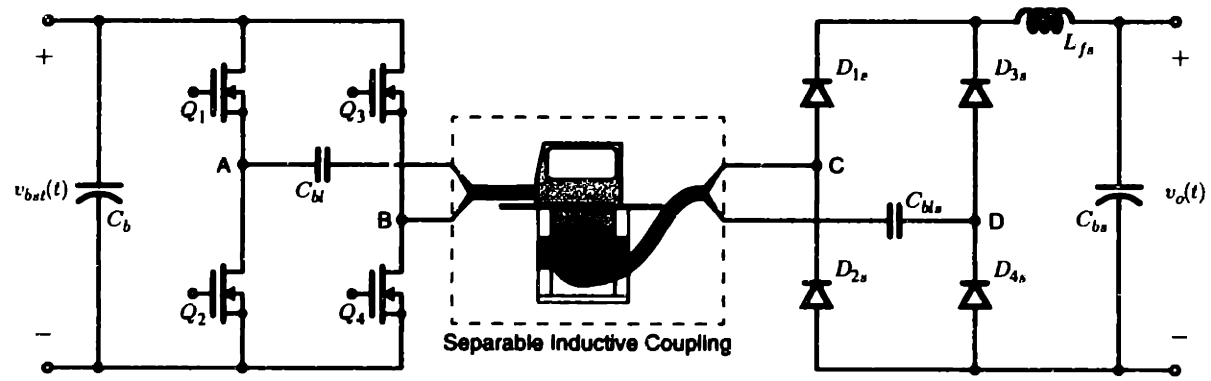



**Figure 4.13:** Simplified schematic of the full-bridge prototype.

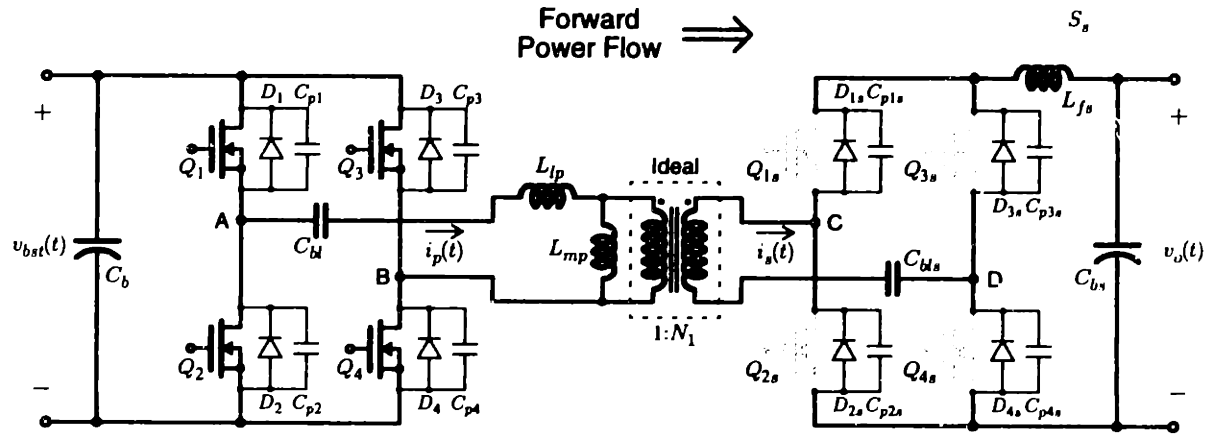



**Figure 4.14:** Full-bridge converter FORWARD operation.

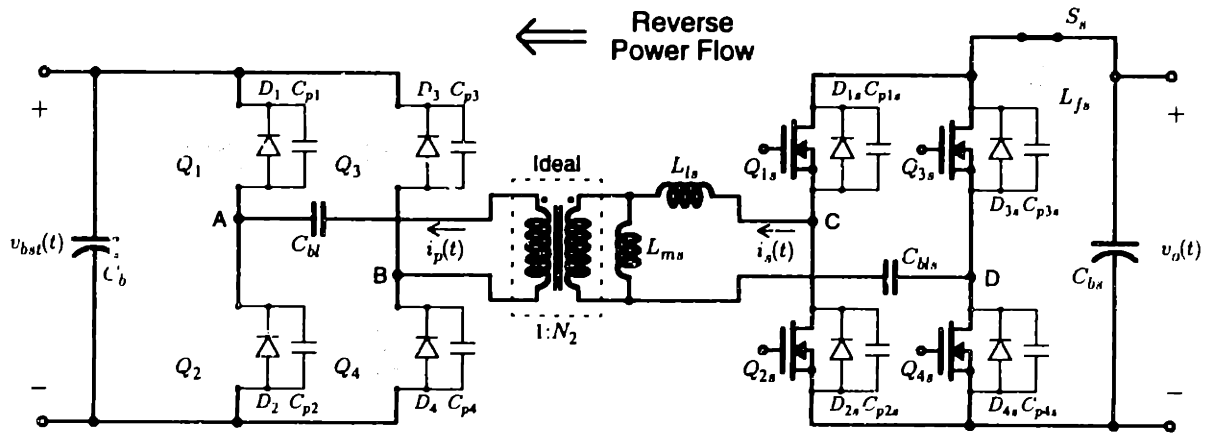



**Figure 4.15:** Full-bridge converter REVERSE operation.

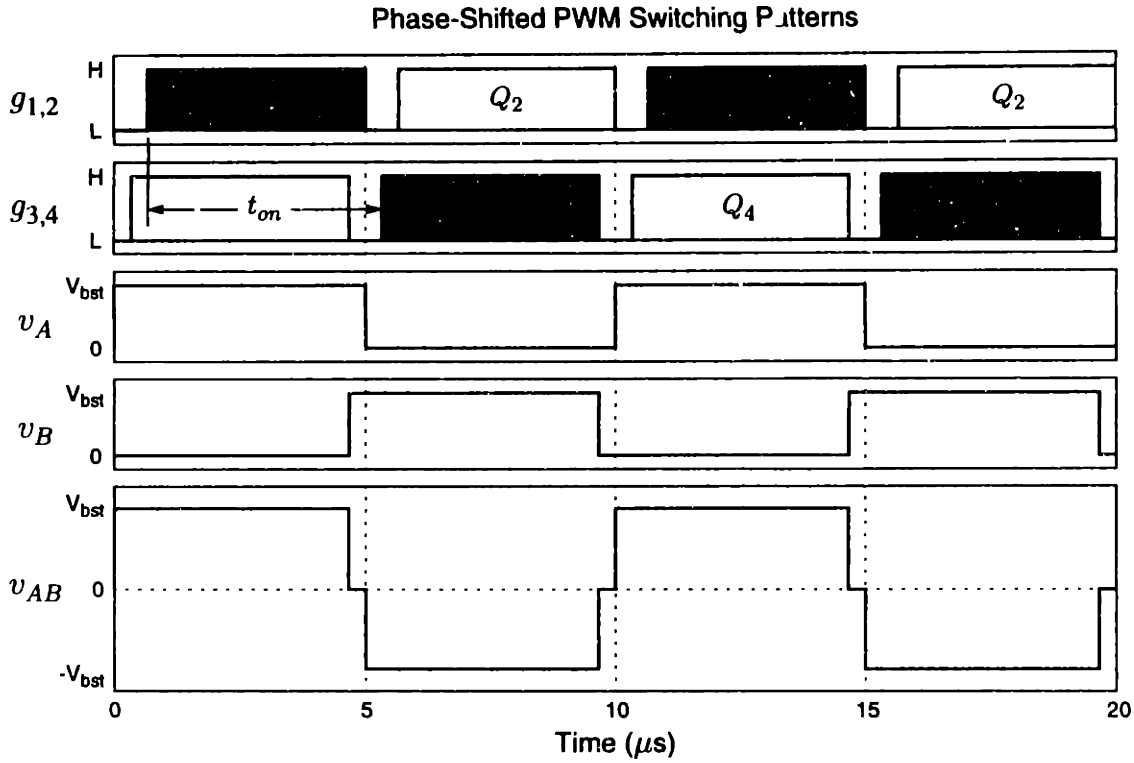



**Figure 4.16:** Idealized phase-shifted PWM switching patterns.

drive patterns for each half-bridge "leg" of the inverter. The 100-kHz AC voltage passes across the inductive coupling where it is rectified by a full-bridge of rectifiers, $D_{1s}$,...,$D_{4s}$. The filter inductor $L_{fs}$ and capacitor $C_{bs}$ extract the DC average of the PWM voltage waveform.

In principle, PWM control could be used to vary the output voltage from its maximum all the way down to zero. However, the conversion efficiency decreases as the PWM duty ratio is lowered from unity. The decrease in efficiency is caused by an increase in the r.m.s. switch currents relative to the output current. Therefore, duty-ratio control is used only to cancel the 120-Hz small-signal voltage ripple. The nominal duty ratio remains within a few percent of unity. Large signal voltage control is accomplished by varying the bus voltage $V_{bst}$ at the input to the inverter. This is the same technique that was used for the half-bridge converter in Chapter 3.

Figure 4.14 and 4.15 schematically illustrate a complete bidirectional full-bridge topology.[3] Since active ripple cancellation was only desired during battery charging, the

topology was designed to provide PWM control only during forward power flow. During reverse power flow, the duty ratio is fixed at unity.

Figure 4.14 shows the configuration for forward power flow. In this mode MOSFETs $Q_1$ through $Q_4$ are active while $Q_{1s}$ through $Q_{4s}$ are held off, leaving their anti-parallel diodes to perform rectification. The relay $S_s$ is opened so that an output $LC$ (inductor-capacitor) filter is formed by $L_{fs}$ and $C_{bs}$. A filter inductor $L_{fs}$ is required for PWM operation because, when the duty ratio is less than unity, $L_{fs}$ supports the voltage difference between the transformer secondary and the output. Along with this filter inductor comes an unwanted ringing between $L_{fs}$ and the rectifier capacitances. This ring is lossy, and it typically increases the total DC/DC conversion losses by about 20% [81]. In addition, the ring increases the voltage stress on the secondary-side rectifiers. An external clamp circuit may be added if necessary.

The configuration for reverse power flow is shown in Figure 4.15. In this case the roles of the switches are reversed. MOSFETs $Q_{1s}$ through $Q_{4s}$ are active, while $Q_1$ through $Q_4$ are held off. The relay $S_s$ is closed so that $v_o$, which is now the "input" voltage to the full bridge, forms a stiff bus voltage for $Q_{1s}$ through $Q_{4s}$. Also, no output filter inductor (analogous to $L_{fs}$) is required. Just as with the half-bridge in Chapter 3, unity duty-ratio operation is possible with a capacitor-only filter. Thus, there is no lossy ring, and the energy in the parasitic rectifier capacitances is transferred to the load.

Full design schematics along with a photograph of the completed circuit are provided in Appendix C.

### 4.3.2 Zero-Voltage Switching

Phase-shifted PWM operation of the full-bridge inverter provides zero-voltage switching of the active MOSFETs. This reduces switching losses by permitting full recovery of the energy in the MOSFET parasitic capacitances. Provided the leakage inductance, magnetizing inductance and parasitic capacitances are sized appropriately, ZVS can be guaranteed over a wide operating range. As a result, the full-bridge and half-bridge inverter

3. Although the figures show individual MOSFETs, multiple devices are actually used. This does not alter the operation of the circuit. The switch blocks are shown in detail in Appendix C.

topologies achieve nearly identical efficiencies.[4] Since the ZVS operation of the full bridge has been covered in published literature, only a brief discussion is presented here (see [81] and [110] for further discussion).

Figure 4.16 shows the phase-shifted PWM patterns for the full-bridge converter. The upper two traces show the gate-drive waveforms for the MOSFETs $Q_1$ through $Q_4$. The MOSFETs in each half-bridge leg are driven at a switching frequency of 100 kHz with a unity duty ratio. The small dead time between the low and high gate drives allows sufficient time for the voltages at nodes A and B to ring losslessly from one rail to the other. The resulting clamped pseudo-resonant switch transitions are nearly identical to those described for the half-bridge converter in Chapter 3. In fact, the governing equations (3.3) through (3.5) can be applied directly.

Pulse-width modulation of the voltage $v_{AB}$ — seen across the primary of the inductive coupling — is controlled by adjusting the phase shift between the gate drives applied to each leg of the converter. As illustrated in Figure 4.16, the phase shift is equal to $2\pi t_{on}/T$, where $T$ is the switching period (10 $\mu$s in the prototype). Zero phase shift yields an effective duty ratio of zero, while a phase shift of $\pi/2$ yields a 100% duty ratio. By adjusting $t_{on}$ for PWM control, active ripple cancellation is possible.

Experimental waveforms from the prototype full-bridge converter are plotted in Figure 4.17. The six traces on the left show waveforms during FORWARD operation at a power level of approximately 900 W from a DC bus voltage of approximately 310 V. The voltage waveforms $v_A$ and $v_B$ demonstrate the ZVS transitions. The phase shift between $v_A$ and $v_B$ is very near to $\pi/2$ as indicated by the approximately 95% duty ratio of primary voltage $v_{AB}$. Subsection 4.3.3 below describes how small perturbations in this duty ratio cancel the ripple in $v_{bst}$. The bottom two traces at the left of Figure 4.17 show the voltage and current measured at the input to the secondary-side rectifier. The waveforms clearly show the expected oscillatory ring of the leakage inductance and the parasitic rectifier capacitances.

The six traces on the right of Figure 4.17 show experimental waveforms during REVERSE operation at a power level of approximately 500 W from a DC battery voltage

4. This assumes equivalent MOSFET die areas for both designs. Therefore, the on-state resistance of a switch block in the half-bridge would be 1/2 that of a switch block in the full-bridge.

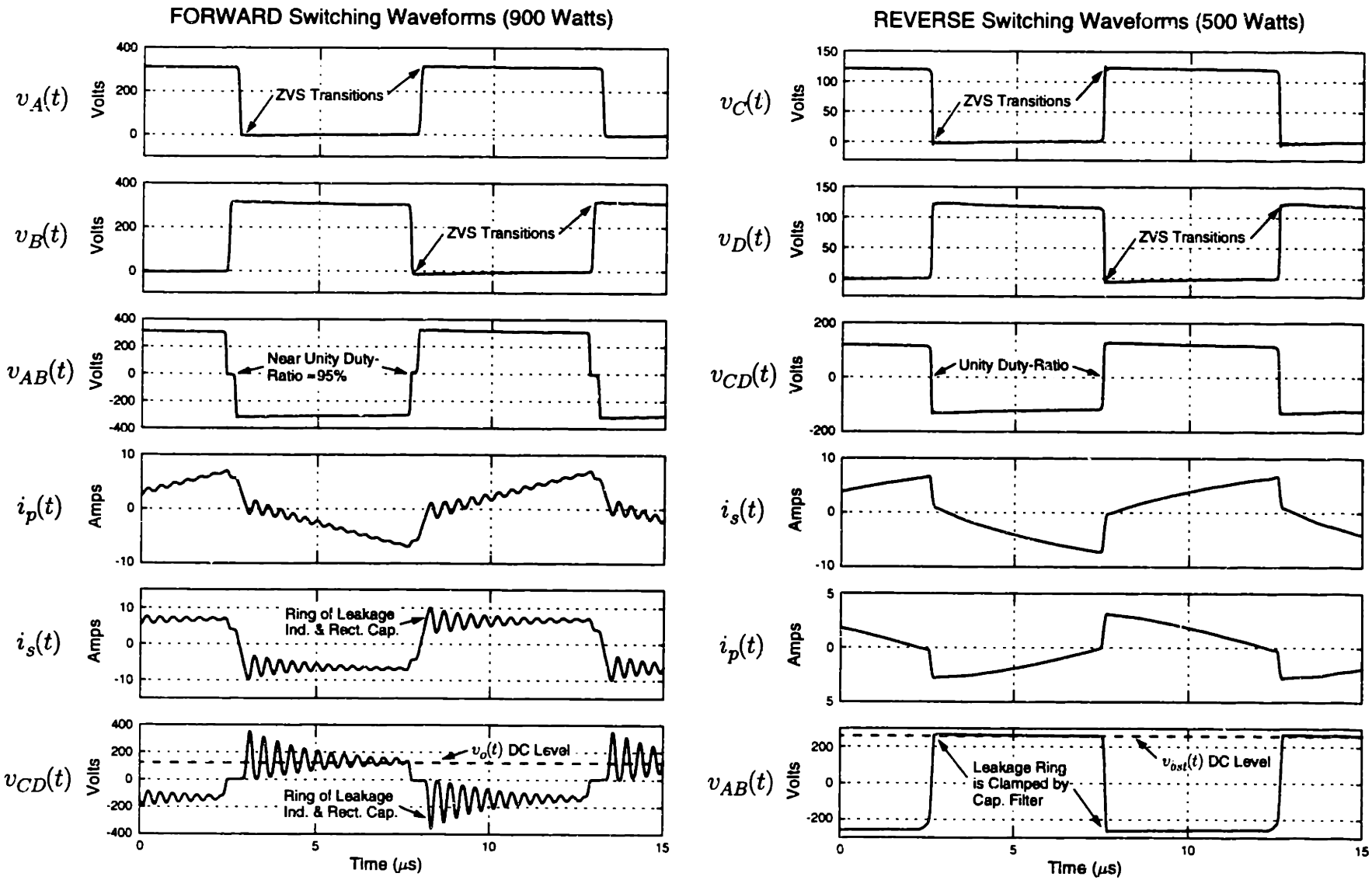



**Figure 4.17:** Experimental switching waveforms for FORWARD and REVERSE operation of the full-bridge prototype.

of approximately 120 V. In this case the inductive coupling is driven from the secondary side at 100 kHz with unity duty ratio. Zero-voltage switching is illustrated by the waveforms $v_C$ and $v_D$. Since reverse operation occurs at unity duty ratio, an output filter inductor is unnecessary. As a result, the voltage $v_{AB}$, shown at the bottom right in the figure, is effectively clamped by the filter capacitor $C_b$.

### 4.3.3 Feedforward Ripple Cancellation

One of the objectives for this second prototype was to demonstrate active cancellation of the 120-Hz ripple components. This ripple is a byproduct of the PFC stage. If the ripple is not eliminated during the second stage DC/DC conversion, it passes directly to the load. Generally, the peak-to-peak amplitude of the voltage ripple at $v_{bst}$ is on the order of 1% of its DC value. For a resistive load, such a small variation is of little consequence. However, in a battery charging application the load behaves, at least to a first order approximation, as a voltage source with a relatively small series resistance. The series resistance is typically so small that even a slight voltage ripple will induce significant current ripple into the battery.

This is exemplified by the experimental waveforms in Figure 4.18 (a). The experimental data was recorded using the prototype hardware with the ripple cancellation circuitry disabled. The waveforms show a 25% (peak-to-peak) current ripple while charging a 120-V lead-acid battery pack with a nominal current of 2.3 A. The corresponding voltage ripple during this experiment is approximately 0.5% (peak-to-peak). Although the voltage ripple is relatively small, the current ripple is significant. As the nominal charge current is increased, the ripple can easily exceed 100%, resulting in a discontinuous battery current.

Discontinuous or high-ripple charging current has not been shown to damage lead-acid batteries. In fact, many lead-acid battery chargers currently operate with significant 120-Hz ripple. However, such ripple may increase the temperature and pressure of the battery during charge. And, it is quite possible that future battery chemistries and high-rate charge profiles may require that such ripple is eliminated. The PWM voltage control capability of the phase-shifted full-bridge DC/DC converter makes it possible to provide active ripple cancellation. The prototype full-bridge converter uses a feedforward control technique to accomplish this.

The steady-state bus voltage at the input to the DC/DC stage can be described as

$$v_{bst}(t) = V_{bst} + r(t) \tag{4.1}$$

where $V_{bst}$ is the nominal or DC level and $r(t)$ is the 120-Hz ripple component. Under PWM control the output voltage of the DC/DC stage is linearly related to the duty-ratio command by

$$v_o(t) = d(t)Nv_{bst}(t) = d(t)N(V_{bst} + r(t)) \tag{4.2}$$

where $d(t)$ is the duty ratio and $N$ is the effective transformer turns ratio.[5] Exact ripple cancellation requires a duty ratio of

$$d(t) = \frac{DV_{bst}}{V_{bst} + r(t)} \tag{4.3}$$

where $D$ is a constant slightly less than one. As desired, the expression for the steady-state output voltage becomes a constant $v_o(t) = DNV_{bst}$.

However, the nonlinear expression in (4.3) is expensive to implement in analog hardware. A linear approximation of the expression in (4.3) can be found by taking the first term in its binomial series expansion, as follows:

$$d(t) = D - \frac{D}{V_{bst}} r(t) \tag{4.4}$$

The linear approximation is simpler to implement and yields nearly ideal performance. The resulting output voltage is found by substituting (4.4) into (4.2).

$$v_o(t) = DNV_{bst} - \frac{DN}{V_{bst}} r(t)^2 \tag{4.5}$$

Although the expression in (4.5) is not a constant, the relative amplitude of the AC component is greatly reduced. Significant reduction in the AC component requires only that $V_{bst}$ is much greater than the amplitude of $r(t)$, and this condition is automatically satisfied by the PFC stage. Thus, if the amplitude of the input voltage ripple is 1%, the output ripple amplitude will be reduced by over 99%. The amplitude of the remaining ripple is insignificant.

---

5. Voltage droop across the coupling leakage inductance does not affect the ripple cancellation. Therefore, it has been ignored to simplify the discussion.

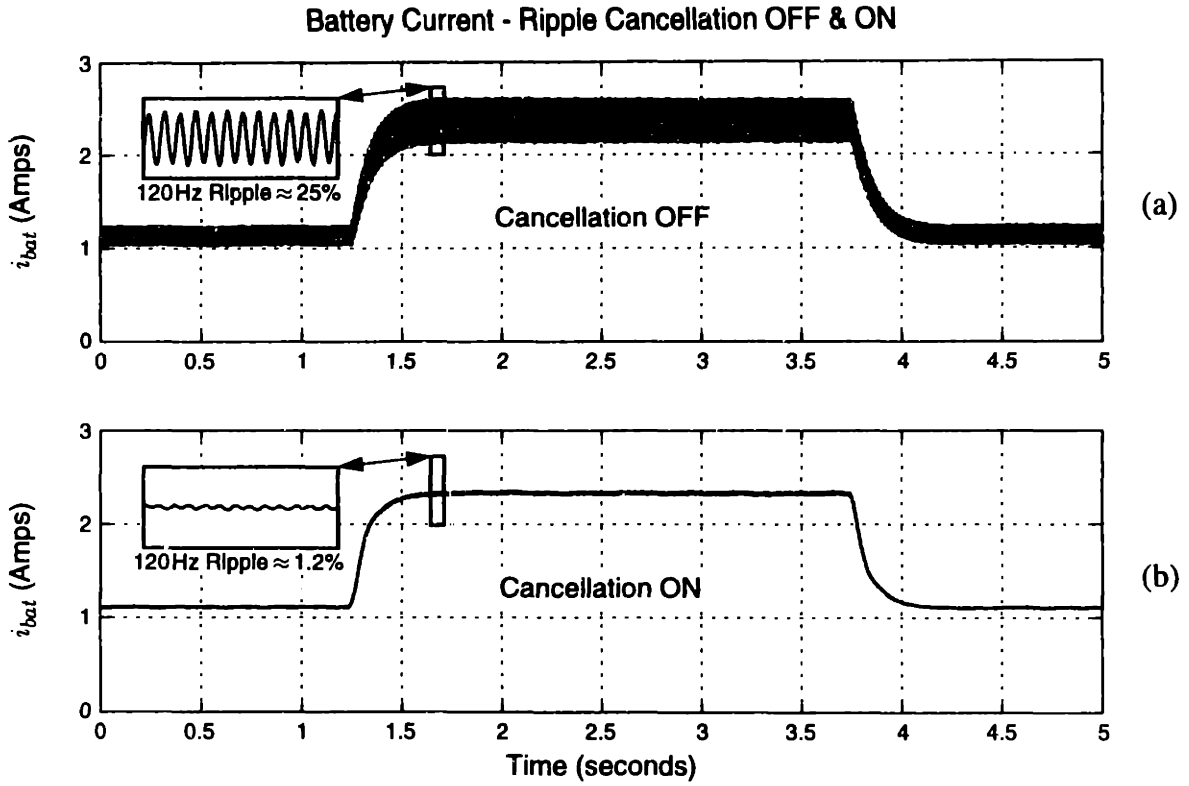



**Figure 4.18:** Experimental waveforms showing the effect of ripple cancellation.

In practice, the feedforward cancellation scheme just described is implemented as follows. A first-order analog high-pass filter, with a corner frequency of approximately 20 Hz,[6] is used to separate $r(t)$ from a measurement of $v_{bst}(t)$. The measured ripple component is scaled by $D/V_{bst}$ and subtracted from a constant duty-ratio command $D$. The required perturbation in $d(t)$ is small, so $D$ is selected as close to unity as possible.[7] Setting $D$ near unity ensures that there is little or no conduction loss penalty from the freewheeling current in the DC/DC stage.

Experimental waveforms demonstrating the effectiveness of the ripple cancellation technique are shown in Figure 4.18. The expanded views demonstrate a measured reduction in current ripple from 25% to 1.2%. The remaining ripple can be attributed to the linear approximation of the optimum $d(t)$ and to a slight phase shift from the filtered

6. The corner frequency must be far below 120 Hz so that the amplitude and phase of $r(t)$ are unaffected. However, too low a corner frequency may adversely affect the digital charge-current controller (see Chapter 5).
7. $D$ was set to 95% for the prototype. (See Figure 4.17). In practice, $d(t)$ is clipped just above and below $D$ to limit any dynamic interaction between the ripple cancellation and digital control.

measurement of $r(t)$, which is used to generate the feedforward command. Also, note that the transient step changes in the output current have essentially no affect on the ripple cancellation.

### 4.3.4 DC Transfer Characteristic

The DC transfer characteristic of the full-bridge inverter can be predicted using the analytical techniques presented in subsection 3.3.4. Because the voltage across the coupling primary from a full-bridge inverter is twice that from a half-bridge inverter, Equations (3.9) to (3.24) require minor modifications. The $LC$ filter used during forward PWM operation alters the analysis of the rectifier operation. The voltage difference that used to appear across $L_{ls}$ now appears across the series combination of $L_{ls}$ and $L_{fs}$. However, the change does not appreciably affect the droop characteristic of the converter. Provided the conduction losses due to the magnetizing current are normalized for both designs the droop characteristics will be nearly identical.

### 4.3.5 Series-Resonant Operation

Just as with the half-bridge topology of Chapter 3, it is possible to extend the power capability of the DC/DC stage by properly scaling the blocking capacitors $C_{bl}$ and $C_{bls}$ to resonate with the leakage inductance of the coupling. Provided that the resonant frequency is maintained slightly below the switching frequency, the primary current will lead the voltage, and ZVS can still be achieved. The affect of the series-resonant capacitors on the droop characteristic of the converter is discussed in Part II of this thesis.

## 4.4 Summary

This chapter described the hardware design and operation of a 600-W bidirectional inductively-coupled system. The system uses two stages of power conversion: a bidirectional boost-buck-inverter and a full-bridge DC/DC converter. Both stages were described, and their operation was analyzed. Experimental data from the prototype stages was compared with expected results. Given the hardware background provided in this chapter, Chapters 5 and 6 will focus on algorithms for digital control of the prototype hardware.

# Chapter 5

# Digital Control Schemes

This chapter describes the design and operation of *software* algorithms for digital control of the prototype hardware systems discussed in Chapters 3 and 4. This chapter develops specific current-control algorithms for the charging and discharging of EV batteries. In addition, large-signal linear voltage- and current-control techniques are developed, which provide stable, well-characterized performance over wide variations in voltage and current. These techniques form the basis for general multirate control, which will be discussed further in Chapter 6. The performance of each algorithm is confirmed through simulation and experimental verification.

## 5.1 Digital Control Overview

The power circuits described in Chapters 3 and 4 can both be controlled by a common digital microcontroller board, built specifically for the prototype inductively-coupled systems. The board contains the analog inner current-loop hardware described in Chapters 3, an Intel 80C196KC embedded applications microprocessor, and a number of support components. An attached control panel with buttons, lights, and a liquid-crystal display (LCD) acts as a user interface. Please see Figure 3.3 for a diagram of the system.

The Intel 80C196KC microcontroller is the "brain" of the inductively-coupled systems [43]. The 80C196KC is low-cost 16-bit integer math processor, which runs at 16 MHz. All high-level control, command, and user interface functions are performed by software running on the microcontroller, with plenty of extra processing power. Voltage and current measurements required for control are accomplished using an eight channel 10-bit A/D converter, which is integrated into the 80C196KC. Analog circuits are used to filter and

pre-scale measured quantities in order to minimize the effects of noise and quantization. Control of the power electronics is accomplished using a 12-bit multiplying D/A converter, which programs a variable gain $k$ within the analog inner current loop. The effect of this variable on control is made clear in the sections that follow. The mixed analog/digital control hardware used here is similar to that described in [78] and [94]. Thus, a detailed description is omitted. However, complete schematics, which include the details of this implementation, are provided in Appendix C.

Code for the microcontroller was developed in the C programming language using cross-compilers from Intel and Hi-Tech [39, 43]. All digital control algorithms were implemented using interrupt driven routines to ensure synchronization with the operation of the power electronics. Interim processing cycles are used for low priority tasks such as updating the LCD display on the user interface. A complete discussion of the microcontroller code is presented in Appendix F.

The voltage control algorithms described in the following section demonstrate the core digital control methodology. The battery-current control and general multirate control algorithms presented in later sections build on this basic approach.

## 5.2 Voltage Control

A simplified schematic of a single-cell boost converter appears in Figure 5.1. Although somewhat simpler than the boost stages described in Chapters 3 and 4, the configuration diagrammed here is operationally equivalent to both for control and modeling purposes. An analog inner current loop, described in subsection 3.2.3, shapes the inductor current $i_L(t)$ to follow a desired reference waveform $i_p(t)$. To ensure UPF operation, the reference waveform $i_p(t)$ is a scaled copy of the rectified input voltage waveform. An outer, voltage-loop controller can adjust $v_{bst}$ to a desired value by varying the scale factor $k$ used to compute $i_p(t)$. Changing $k$ is tantamount to changing input power.

In [67] and [78], a large-signal linear, "power balance" model of the boost UPF rectifier was derived using Tellegen's theorem [49]. The power balance model relates the rate of change in capacitor energy to the input power, output power, and the rate of change in inductor energy, as follows:

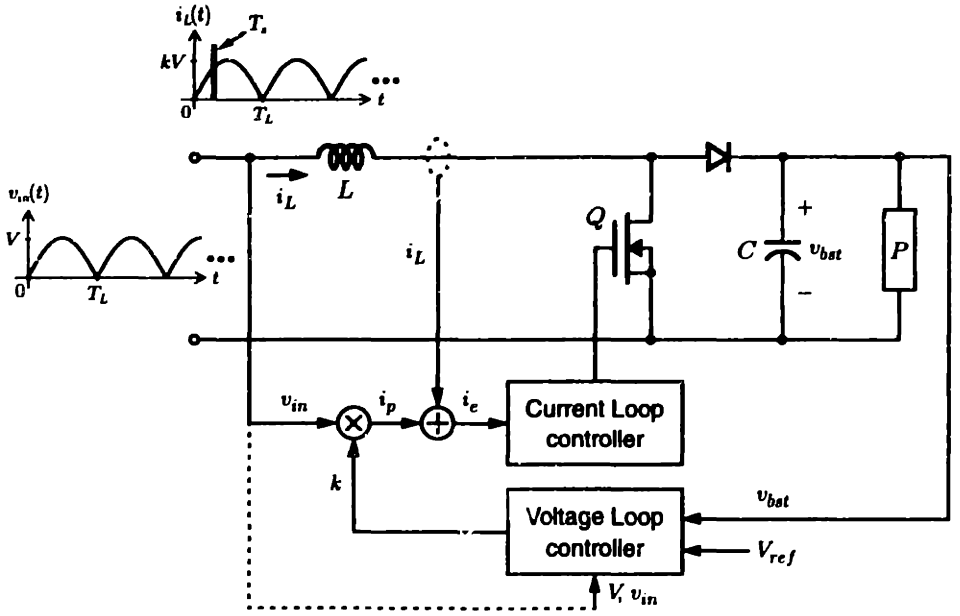



**Figure 5.1:** High-power-factor pre-regulator.

$$\frac{1}{2}C\frac{dv_{bst}^2(t)}{dt} = v_{in}(t)i_L(t) - \frac{1}{2}L\frac{di_L^2(t)}{dt} - P(t)\,. \tag{5.1}$$

Assuming that the inner current loop works well, the inductor current $i_L(t)$ is presumed to track $i_p(t)$ perfectly, provided that the switch frequency ripple is ignored.[1] Thus, $i_L(t) \cong i_p(t) = k(t)v_{in}(t)$. Substituting for $i_L(t)$ in (5.1) yields

$$\frac{1}{2}C\frac{dv_{bst}^2(t)}{dt} = k(t)v_{in}^2(t) - \frac{1}{2}L\frac{d[k^2(t)v_{in}^2(t)]}{dt} - P(t)\,. \tag{5.2}$$

Assuming that $k$, $v_{bst}^2$ and $P$ do not vary significantly over one rectified line cycle, Equation (5.2) can be greatly simplified by integrating over one rectified line cycle.[2]

Integrating (5.2) over the $n^{\text{th}}$ cycle of length $T_L$ yields the following sampled-data model:

$$x[n+1] = x[n] + \frac{T_L V^2}{C}k[n] - \frac{2T_L}{C}P[n] \tag{5.3}$$

---

1. The switching period $T_s$ is considerable shorter than the period of the rectified line cycle $T_L$. Therefore, switching ripple can generally be safely ignored on the longer time scale.
2. These assumptions are valid because the gain $k(t)$ is fixed during each line cycle for UPF operation. Furthermore, since the boost capacitor is large, $v_{bst}$ will vary slowly, and the rate of change in $P$ can be controlled by the system dynamics.

where the state variable $x[n]$ denotes the value of $v_{bst}^2$ at the beginning of the $n^{\text{th}}$ cycle. Similarly, $k[n]$ and $P[n]$ are respectively the values of the gain $k$ and the load power during the $n^{\text{th}}$ cycle. The variable $T_L$ is the period of one rectified input line cycle, i.e., $1/T_L = 120$ Hz. The index $n$ in the sampled data model of (5.3) increments once every $T_L$ seconds. The sampled-data power balance model of the boost rectifier is illustrated schematically with the use of the $z$-transform in Figure 5.2 [97].

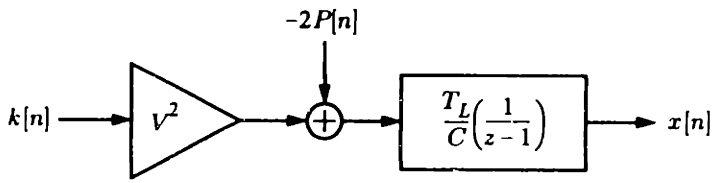



**Figure 5.2:** Boost converter sampled-data model.

Because this model is not a small-signal approximation, it is especially suitable for developing a controller for tracking applications like the battery charger. Also, it is a sampled-data model, so it is a convenient starting point for developing a digital controller. The subsections below outline two discrete-time (DT) algorithms for controlling the squared output voltage. The voltage control systems utilize the multirate structure illustrated in Figure 5.1. A high-bandwidth analog inner current loop shapes the input current, and a lower bandwidth voltage loop controls the output bus voltage.

### 5.2.1 PI Control

A discrete-time version of a *proportional-integral* (PI) controller has been used in [49], [78], and [94] to stabilize similar voltage loops. Formulation of the control loop begins with the control command $k$:

$$k[n] = \frac{C}{T_L V^2}\Big(g_1(X[n] - x[n]) + g_2\sigma_v[n]\Big) \tag{5.4}$$

where $X[n]$ is the squared-voltage reference, $g_1$ and $g_2$ are feedback gains, and $\sigma_v[n]$ is the state of a DT accumulator. The accumulator serves to "integrate" the squared output voltage error. The future accumulator state $\sigma_v[n+1]$ is computed as

$$\sigma_v[n+1] = \sigma_v[n] + (X[n] - x[n]). \tag{5.5}$$

Combining (5.4) and (5.3) yields the following state-space description of the closed voltage loop:

$$\begin{bmatrix}\sigma_v[n+1]\\ x[n+1]\end{bmatrix} = \begin{bmatrix}1 & -1\\ g_2 & 1-g_1\end{bmatrix}\begin{bmatrix}\sigma_v[n]\\ x[n]\end{bmatrix} + \begin{bmatrix}0\\ g_1\end{bmatrix}X[n] + \begin{bmatrix}0\\ \frac{-2T_L}{C}\end{bmatrix}P[n]\,. \tag{5.6}$$

This compensation unfortunately results in a voltage loop that is dependent on the load power $P[n]$. It will be shown in later sections that the inability to guarantee the voltage-loop dynamics independently of the load significantly complicates later control developments.

To make the voltage-loop dynamics independent of load power, the control command is computed as in (5.4), but with the addition of a power feedforward term:

$$k[n] = \frac{C}{T_L V^2}\Big(g_1(X[n]-x[n]) + g_2\sigma_v[n]\Big) + \frac{2}{V^2}P[n]\,. \tag{5.7}$$

In the prototype circuit, both the load terminal voltage and current are available, and computing the load power requires little additional computational effort. Substituting (5.7) into (5.3) yields a new second-order, large-signal linear model for the actively controlled boost converter:

$$\begin{bmatrix}\sigma_v[n+1]\\ x[n+1]\end{bmatrix} = \begin{bmatrix}1 & -1\\ g_2 & 1-g_1\end{bmatrix}\begin{bmatrix}\sigma_v[n]\\ x[n]\end{bmatrix} + \begin{bmatrix}0\\ g_1\end{bmatrix}X[n]\,. \tag{5.8}$$

A closed-loop DT transfer function from $X[n]$ to $x[n]$ is obtained using the $z$-transform:

$$\tilde{H}_{PI}(z) = \frac{g_1\left(z + \frac{g_2-g_1}{g_1}\right)}{z^2 + (g_1-2)z + (1+g_2-g_1)}\,. \tag{5.9}$$

The closed-loop poles are

$$z_1, z_2 = \frac{(2-g_1) \pm \sqrt{g_1^2 - 4g_2}}{2}, \tag{5.10}$$

and a finite zero exists at $z = \frac{g_1-g_2}{g_1}$.

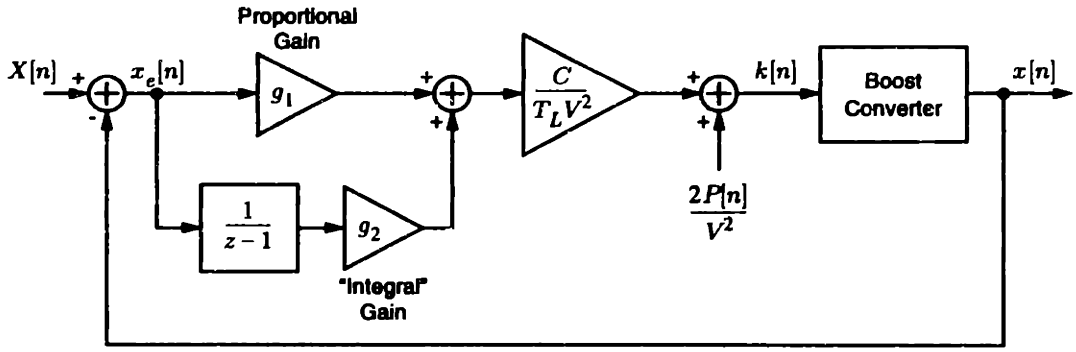



**Figure 5.3:** Block diagram of the PI voltage loop.

The discrete-time PI voltage loop is diagramed in Figure 5.3. Selecting the gains $g_1$ and $g_2$ so that the closed-loop poles have magnitude less than one results in a stable system. Because (5.8) is independent of load power, a stable system will remain stable for practically any load. Thus, given sufficient time it will converge to any reference.

The PI voltage loop was implemented and tested using the 1.5-kW unidirectional prototype hardware described in Chapter 3. Gains $g_1$ and $g_2$ were calculated to place the closed-loop poles at $z_1 = z_2 = 0.90$ with a zero located at $z = 0.95$. With the DC/DC stage disconnected, a 143.8-$\Omega$ load resistor was connected directly across the output terminals of the interleaved boost converter. The voltage-loop command was programmed to vary the output voltage $v_{bst}$ periodically between 300 and 350 V, which corresponds to a change in load power from approximately 625 to 850 W.

Experimental results are plotted in Figure 5.4 for both square-wave and sawtooth voltage commands. The first few seconds of each trace show the soft-start mechanism of the converter. Closed-loop command following begins after the soft start. The dashed lines represent the voltage command in each case. The results demonstrate perfect tracking of constant and ramp commands in the steady state. However, the transient performance exhibits a significant amount of overshoot due to the relatively low-frequency zero in the system. This overshoot is an unavoidable consequence of the discrete-time PI implementation. The next subsection introduces an improved *pole-placement* (PP) algorithm.

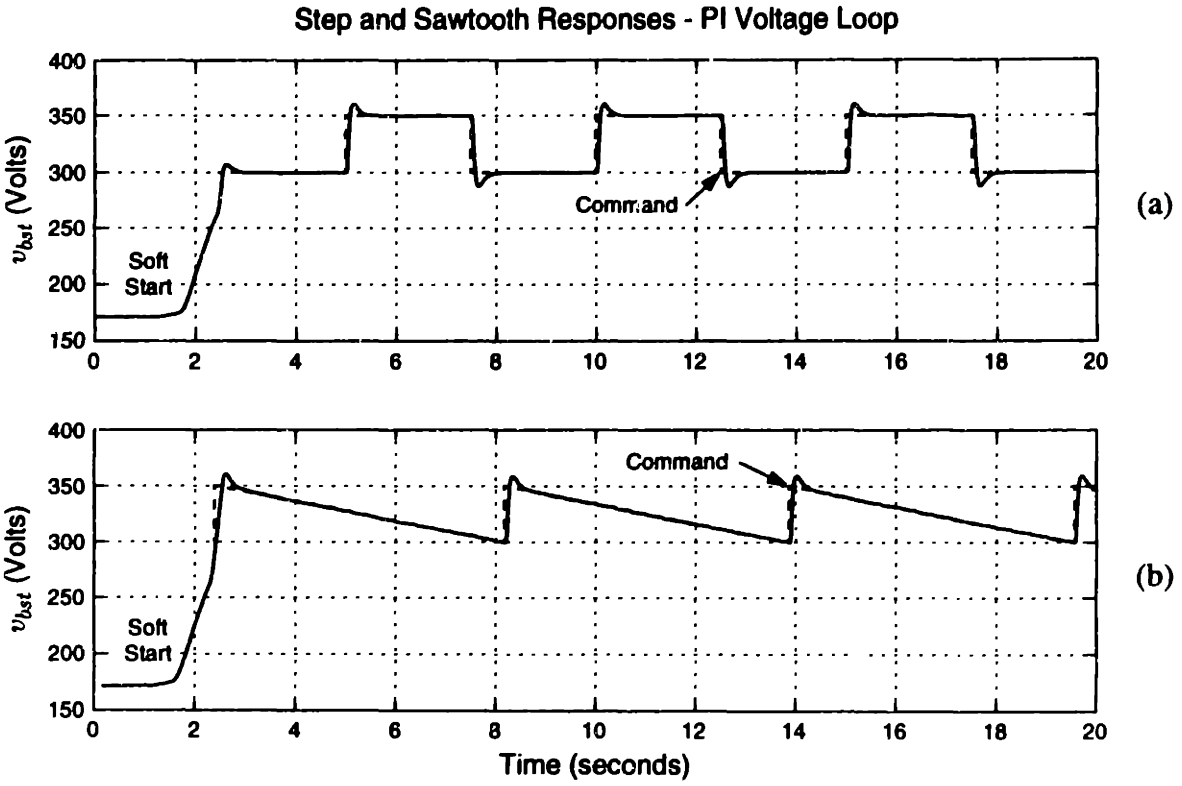



**Figure 5.4:** Step and sawtooth responses for the voltage loop under PI control.

### 5.2.2 PP Control

The pole-placement algorithm employs full-state feedback, which, in principle, allows for arbitrary placement of the closed-loop poles. It differs from the PI controller in that its closed-loop zero is located at the origin in the $z$-plane. As a result, overshoot is eliminated without affecting the bandwidth of the closed-loop system.

The control command $k$ for the PP controller is

$$k[n] = k[n-1] + \frac{2}{V^2}(P[n] - P[n-1]) + \frac{C}{T_L V^2}\Big(g_1(X[n] - x[n]) + g_2(X[n] - x[n-1])\Big). \quad (5.11)$$

Note that the load power feedforward term has already been included. The $k[n-1]$ term directly incorporates an accumulator. Therefore, $k[n]$ can only be in steady-state when the two error terms, $(X[n] - x[n])$ and $(X[n] - x[n-1])$, are both zero. Combining (5.11) with (5.3) results in the following closed-loop state-space description:

$$\begin{bmatrix} \sigma_x[n+1] \\ x[n+1] \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -(1+g_2) & 2-g_1 \end{bmatrix} \begin{bmatrix} \sigma_x[n] \\ x[n] \end{bmatrix} + \begin{bmatrix} 0 \\ g_1+g_2 \end{bmatrix} X[n] \quad (5.12)$$

where $\sigma_x[n]$ is defined as

$$\sigma_x[n+1] = x[n]\,. \tag{5.13}$$

The variable $\sigma_x[n]$ accounts for the added state from the $x[n-1]$ term in (5.11). The closed-loop DT transfer function for the PP controller is

$$\tilde{H}_{PP}(z) = \frac{(g_1+g_2)z}{z^2+(g_1-2)z+(g_2+1)}, \tag{5.14}$$

and the closed-loop poles are

$$z_1, z_2 = \frac{(2-g_1) \pm \sqrt{g_1^2 - 4(g_1+g_2)}}{2}. \tag{5.15}$$

As discussed, the finite zero is located at $z=0$. The discrete-time PP voltage loop is diagramed in Figure 5.5 below.

The relative performance of the two voltage loops can be judged by comparing their step responses. For each system, values for $g_1$ and $g_2$ were calculated so that the closed-loop poles were located at $z_1 = z_2 = 0.85$. The unit-step responses of the resulting systems are plotted in Figure 5.6. Figure 5.6 demonstrates two clear advantages of the PP loop in contrast to its PI counterpart. First, the response of the PP loop has zero overshoot versus about 18% for the PI loop. Second, with a much lower peak control command, the PP loop achieves settling times equal to the PI loop. Figure 5.6 shows the peak value of $k[n]$ for the

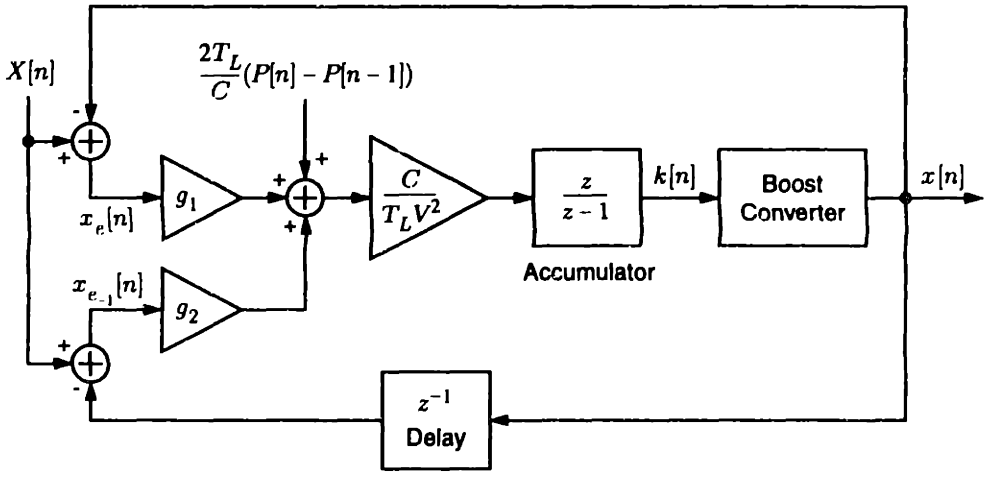



**Figure 5.5:** Block diagram of the PP voltage loop.

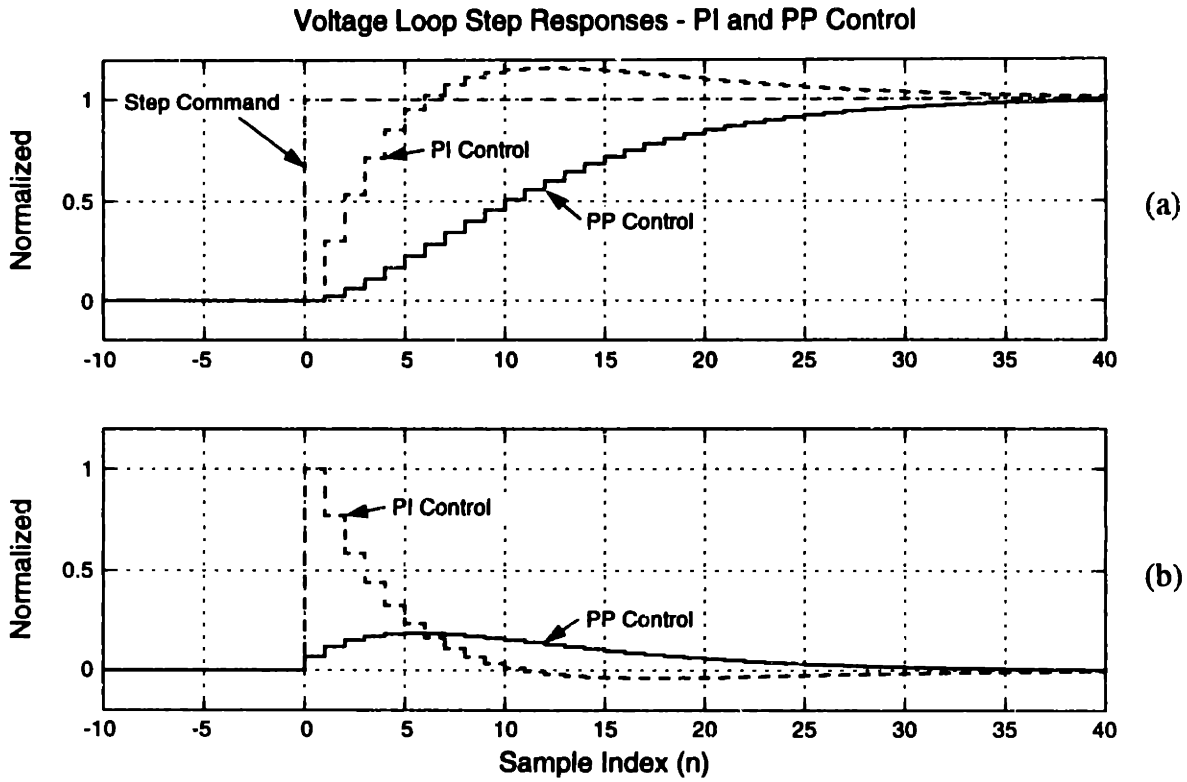



**Figure 5.6:** Simulated responses for PI and PP control. (a) Voltage-loop unit step responses. (b) Normalized control commands $k[n]$.

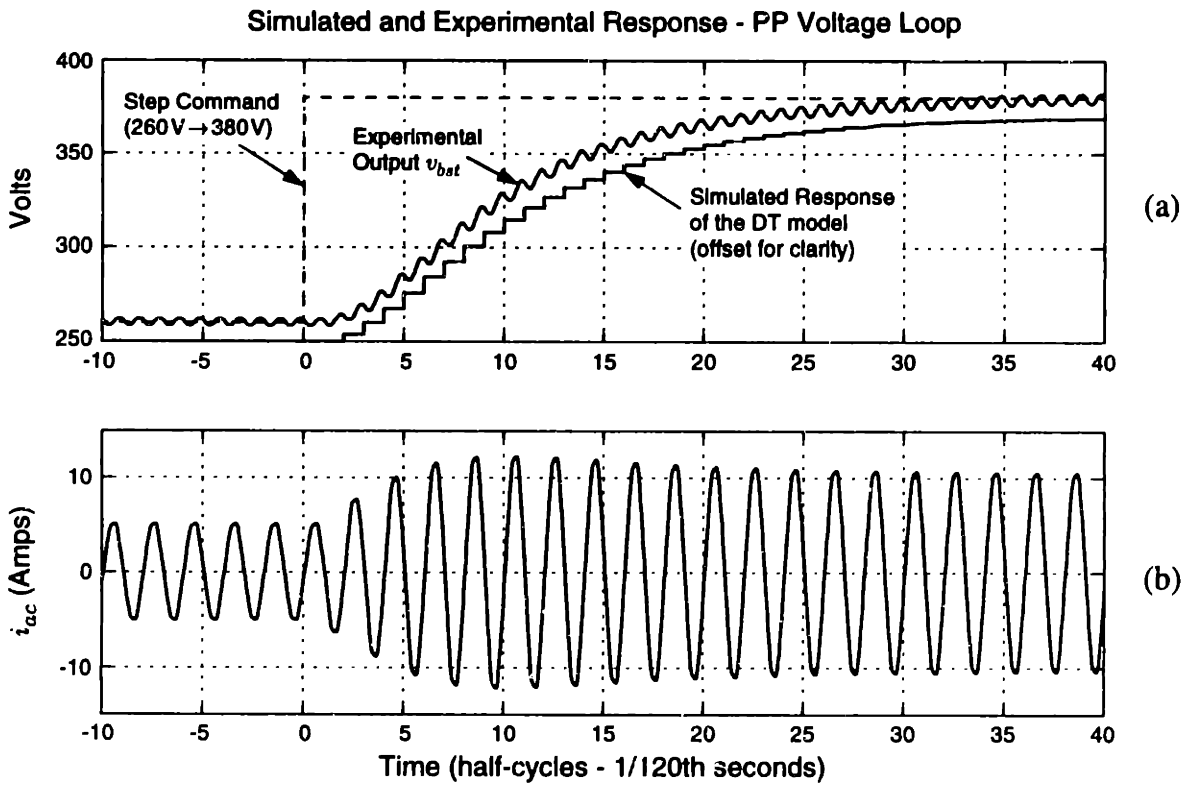



**Figure 5.7:** Simulated and experimental step responses for the PP voltage loop. (a) Boost converter output voltage. (b) Power-factor corrected input current.

PP loop to be only 19% of the peak PI command. This is critical to avoid input command (and hence input power) saturation.

The PP voltage loop was implemented and tested using the 1.5-kW unidirectional prototype hardware. The closed-loop poles were set at $z_1 = z_2 = 0.85$, and the response of $v_{bst}$ to a step change in the voltage command from 260 to 380 V was measured experimentally. This corresponds to a step change in load power from approximately 500 to 1000 W. Figure 5.7 (a) plots the experimental response alongside a simulated response, which was computed using the close-loop DT model in (5.12). The experimental and simulated results are nearly identical at each sample point. The 120-Hz ripple component that rides along $v_{bst}$ has no affect on the transient response because the DT controller operates synchronously with the period $T_L$ of the rectified line voltage. The corresponding experimental line current $i_{ac}$ is plotted in Figure 5.7 (b). Because the gain command $k$ is updated once per rectified line cycle — in sync with the zero crossings of $v_{ac}$ — the inner-loop power-factor correction operates flawlessly throughout the transient. This is shown clearly in the figure.

Figure 5.8 shows the experimental responses of the PP voltage loop to periodic step and sawtooth voltage commands. Again, the first few seconds in each figure show the soft-start mechanism of the converter, after which closed-loop command following begins. The responses demonstrate zero overshoot and good command following. The transient response of the PP voltage loop was also tested in response to step changes in the load power. This was accomplished using a semiconductor switch to abruptly alter the load resistance. With the voltage-loop command fixed at 360 V, the load power was momentarily doubled from 500 W to 1000 W. The transient response of the prototype is plotted in Figure 5.9. A peak voltage disturbance of 5% was observed with a settling time of less than ten half-line cycles. The settling time is far shorter than the time constant associated with the closed-loop poles of the DT controller.[3] The speedy transient decay is due to the feedforward term, which rapidly compensates for changes in load power.

3. This voltage transient is unrelated to the voltage-loop dynamics. The speed of convergence depends on the bandwidth of noise filters, which delay the measurement of the load power.

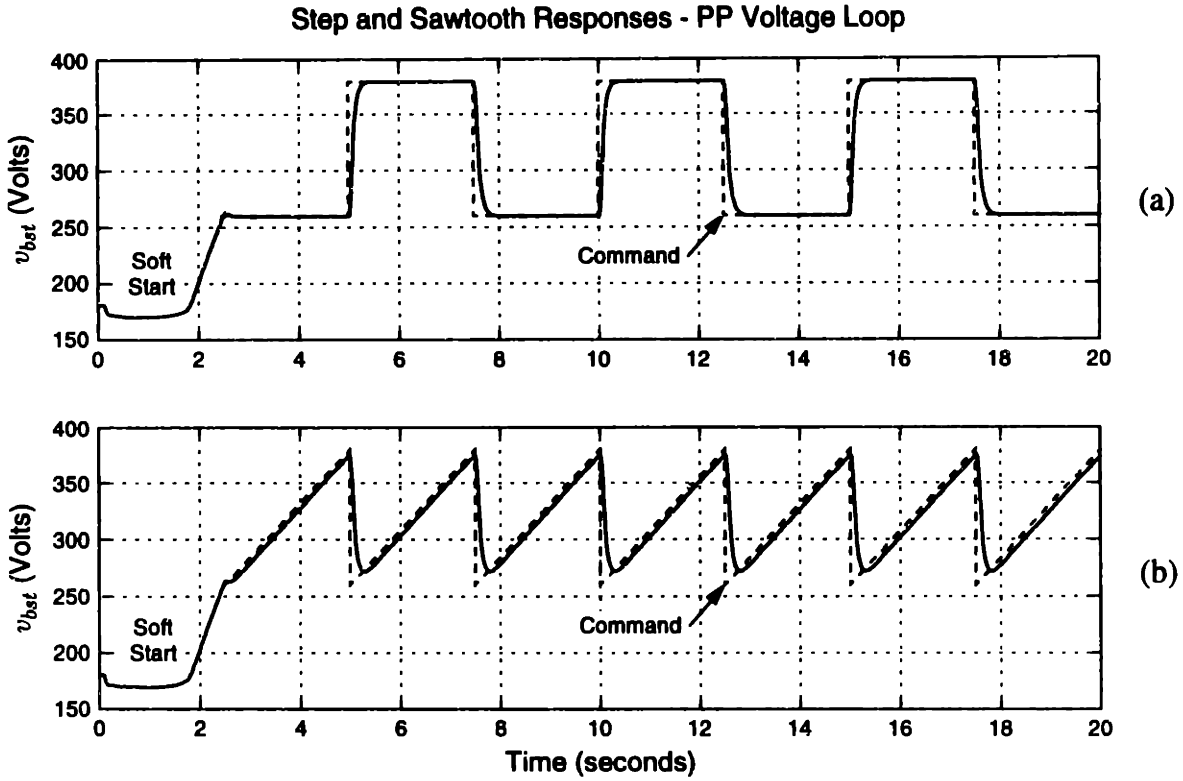



**Figure 5.8:** Experimental step and sawtooth responses for the PP voltage loop.

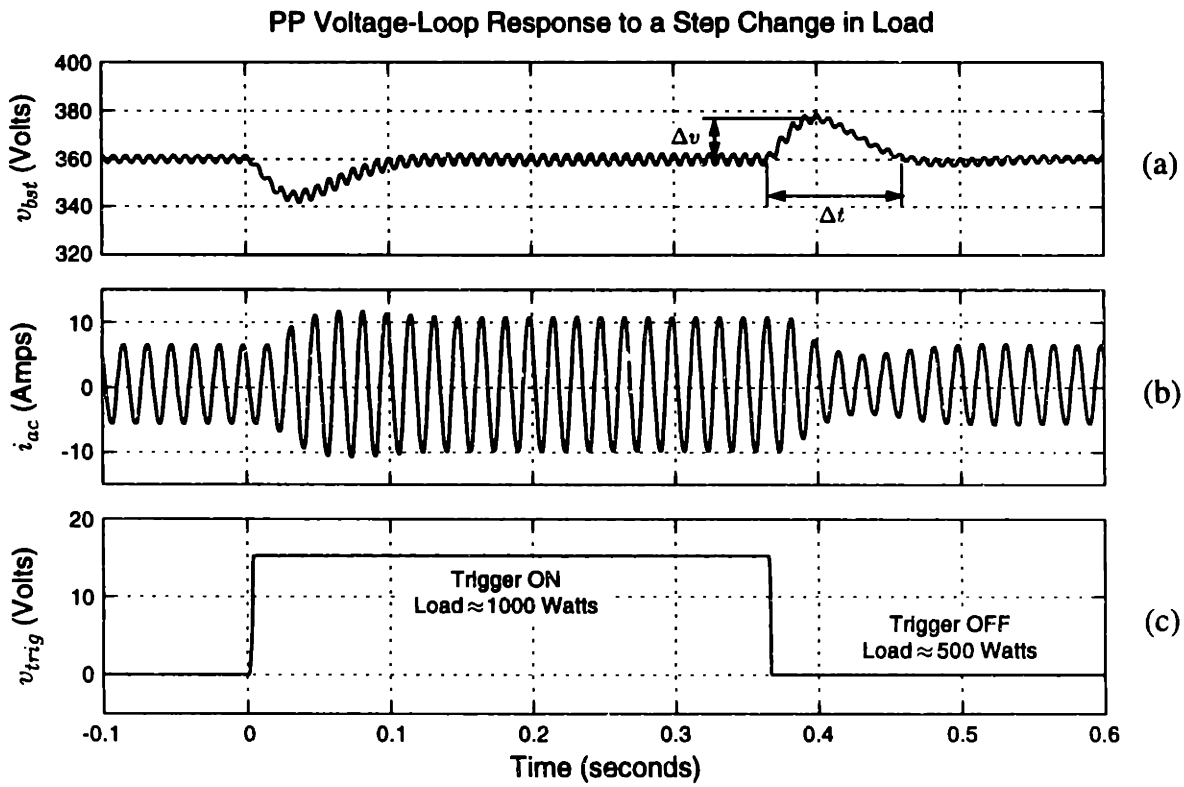



**Figure 5.9:** Experimental response of the PP voltage loop to a load step. (a) Output voltage distortion. (b) Change in input current. (c) Trigger for the step load change.

The PP voltage loop demonstrated superior overall performance. This unconventional feedback structure eliminates the undesirable zero and resulting overshoot of the closed-loop PI transfer function. In addition, the recursive structure of the control command eliminates accumulator windup during saturation. Because of its advantages, PP compensation is used repeatedly throughout this thesis.

## 5.3 Battery-Current Control

One of the primary applications for the inductively-coupled prototypes was for the charging and discharging of EV batteries. To that end, digital algorithms were developed for current control during battery charging and discharging. These algorithms provide stable, closed-loop control, which will track desired current profiles. The charge and discharge algorithms were tested using the 600-W bidirectional inductively-coupled system. The hardware for this system was described in Chapter 4. A lead-acid battery pack, with a nominal voltage of 120 V and a capacity of 2.1 kWh, was used to test the charge/discharge capabilities of the converter. All algorithms were implemented in C on the 80C196KC microcontroller, and complete source-code listings can be found in Appendix F.

### 5.3.1 Charge-Current Control

The charge-current control loop is an extension of the voltage-loop described in the proceeding section. Charge-current control is accomplished using an additional feedback loop, which maps a desired current command to an appropriate voltage command. The control structure appears in Figure 5.10. Two feedback loops are shown. An outer current feedback loop drives the current error to zero, and an inner voltage feedback participates in the command mapping.

Figure 5.11 schematically illustrates an equivalent load model for the DC/DC converter and battery. The DC/DC converter is represented by an ideal (DC capable) transformer in series with an equivalent droop resistance $R_d$. This representation is based on the commutating reactance model described in subsection 3.3.4. The LTI lead-acid battery model in Figure 5.11 has been suggested in [4], and similar models for nickel-cadmium batteries appear in [107]. This model is substantially simplified; it does not include, for

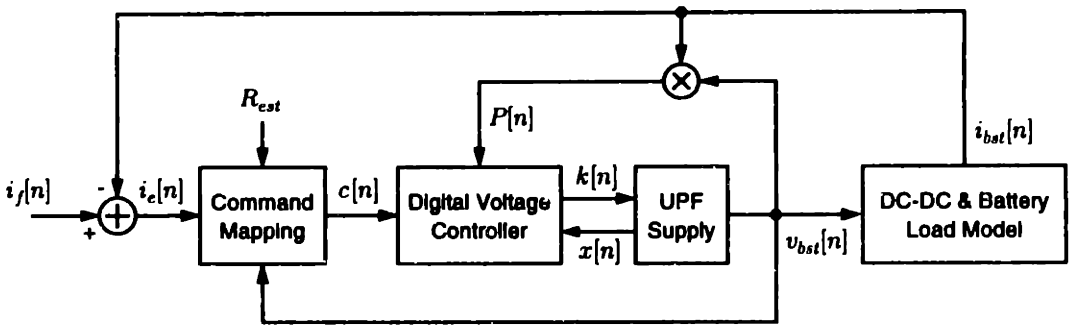



**Figure 5.10:** Block diagram of the charge-current control loop.

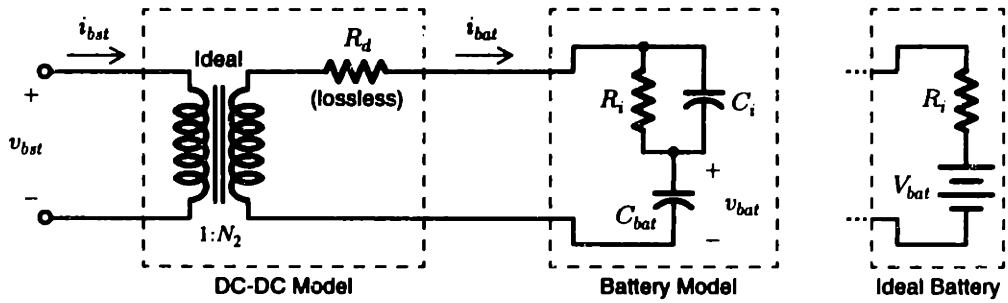



**Figure 5.11:** Simplified load model for the DC-DC converter and battery.

example, battery self-inductance nor the detail of the slow (minutes to hours) electrochemical processes. However, these effects are irrelevant given the control bandwidth. Self-inductance is generally insignificant, except at frequencies well in excess of 120 Hz. And, slow electrochemical effects fall well within the tracking bandwidth of the controller.

Assume, for the moment, the battery is ideal and that $N_2 = 1$. The ideal battery model, also shown in Figure 5.11, is a voltage source in series with a resistance. Since $N_2 = 1$, the ideal transformer may be ignored, and $i_{bat} = i_{bst}$. The boost voltage is then related to the battery voltage and current as

$$v_{bst} = i_{bat}R + V_{bat} \tag{5.16}$$

where $R = R_d + R_i$. This straightforward relationship between $i_{bat}$ and $v_{bst}$ suggests that it may be possible to indirectly control the battery current by supplying an appropriate reference to the voltage-loop controllers already described. Consider, for example, a function that maps a reference current command $i_f[n]$ to a reference voltage command $c[n]$, as follows:

$$c[n] = i_f[n]R + V_{bat}[n]. \tag{5.17}$$

If the voltage loop command is set as $X[n] = c^2[n]$, then $x[n]$ will track the new command $c^2[n]$. Since $x[n] = v_{bst}^2[n]$, the convergence of $x[n]$ and $X[n]$ drives the error between $i_{bat}[n]$ and $i_f[n]$ to zero.

Unfortunately, (5.17) requires that $V_{bat}[n]$ can be measured directly, which is not possible. However, $V_{bat}[n]$ can be eliminated from (5.17) using (5.16). The new command can be written as

$$c[n] = (i_f[n] - i_{bst}[n])R + v_{bst}[n] \tag{5.18}$$

where $i_{bst}[n]$ is now the control variable. Equation (5.18) involves only the readily measured quantities $v_{bst}[n]$ and $i_{bst}[n]$. If the above steps are repeated without the assumption that $N_2 = 1$, $c[n]$ becomes

$$c[n] = (i_f[n] - i_{bst}[n])\frac{R}{N_2^2} + v_{bst}[n] \tag{5.19}$$

where

$$i_{bat}[n] = \frac{i_{bst}[n]}{N_2}. \tag{5.20}$$

Equation (5.19) actually drives $i_{bst}[n]$ and not $i_{bat}[n]$ to track the current reference $i_f[n]$. However, the two currents are related by a constant, so it is a simple matter to replace one in favor of the other. There is a practical advantage to selecting $i_{bst}[n]$ as the control variable because $i_{bst}$ can be directly measured on the "charger" side of the inductively-coupled system. The reliability and accuracy of this measurement can be guaranteed with relative ease. The alternative, a direct measurement of $i_{bat}$, must be accomplished from the "vehicle" side of the system. All measurements on the vehicle side are relayed back to the charger's digital controller. Consequently, the reliability of the intervening communications channel becomes an issue.[4] The prior choice avoids this potential weak link in the feedback loop.

4. Both prototype systems used an optically-coupled communications system to relay information across the coupling.

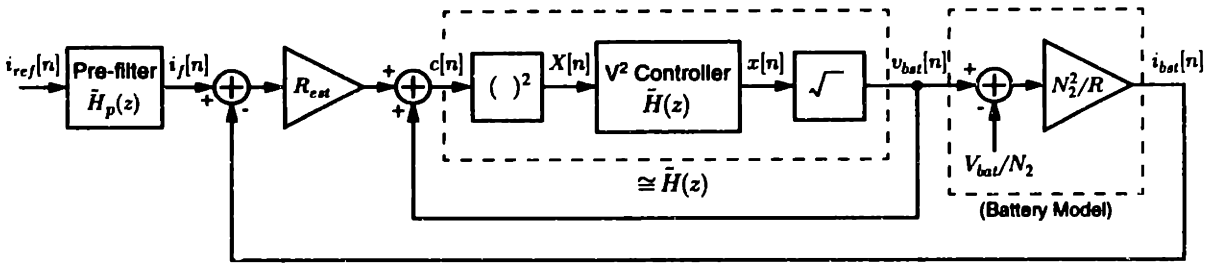



**Figure 5.12:** Block diagram of simplified charge-current control loop.

Applying (5.19) results in the control structure illustrated in Figure 5.12. Note that $R/N_2^2$ in (5.19) has been replaced by $R_{est}$ and that the battery is still assumed to be ideal. Also, a command pre-filter $\tilde{H}_p(z)$ has been added, which for now can be assumed to have a value of $\tilde{H}_p(z) = 1$. At first glance it is clear that the transfer function relationship between $c[n]$ and $v_{bst}[n]$ is nonlinear. In general, this is problematic. However, in this case, the transfer function can be linearized with reasonable accuracy since transients in the boost voltage $v_{bst}[n]$ are constrained by the battery voltage $V_{bat}$. The maximum deviation is typically less than 5% (see Figure 5.15). Within this range, $c[n]$ and $v_{bst}[n]$ are accurately related by the closed-loop transfer function $\tilde{H}(z)$ of the $V^2$ voltage-loop controller. This approximation is based on the first-order binomial expansion $(1+x)^2 \cong 1+2x$ applied to the difference equation relating $c[n]$ and $v_{bst}[n]$. The approximation error is less than 0.23%. Multirate control techniques, which do not depend on linearized approximations, are described in Section 5.4.

The block diagram in Figure 5.12 can now be reduced to a single transfer function relating $i_{bst}[n]$ to $i_f[n]$:

$$\frac{\tilde{I}_{bst}(z)}{\tilde{I}_{ref}(z)} \cong \frac{\tilde{H}_p(z)\tilde{H}(z)R_{est}N_2^2/R}{1-\tilde{H}(z)+\tilde{H}(z)R_{est}N_2^2/R}. \tag{5.21}$$

Provided that $R_{est} = R/N_2^2$ and the pre-filter $\tilde{H}_p(z) = 1$, the closed-loop transfer function for the charge-current loop is approximated by $\tilde{H}(z)$ of the $V^2$ voltage-loop controller, where $\tilde{H}(z)$ is either $\tilde{H}_{PI}(z)$ or $\tilde{H}_{PP}(z)$ from (5.9) or (5.14), respectively. The value of $R_{est}$ may be pre-computed or estimated on-line using the estimation techniques in Chapter 6. The stability of the charge-current loop is guaranteed, even if the more complex battery

model is substituted. Intuitively, the LTI battery model from Figure 5.11 behaves as a lead network, which adds at most 90° of phase margin to the loop transfer function. It can be shown that the closed-loop transfer function remains stable.

Simulated responses for the charge-current control loop are plotted in Figure 5.13. Trace (a) plots the response of $i_{bst}[n]$ to a step in $i_{ref}[n]$ with the pre-filter set to $\tilde{H}_p(z) = 1$. The responses for both voltage-loop controllers, $\tilde{H}_{PI}(z)$ and $\tilde{H}_{PP}(z)$, are shown. The responses are very similar to the earlier voltage-loop responses from Figure 5.6, with one exception. There is a distinct steady-state error when the PP loop is used. This error is caused by the bulk capacitance $C_{bat}$ in the battery model. During constant current conditions the voltage across $C_{bat}$ increases linearly. Thus, for $i_{bst}$ to have zero steady-state error, the inner voltage loop must track a ramp with zero steady-state error. The PI loop has this ability, but unfortunately it suffers from a secondary problem, transient overshoot. This overshoot can be remedied in two ways: by feedforward clipping of the voltage-loop accumulator or by low-pass pre-filtering the current reference. The outcome using both techniques is plotted in Figure 5.13 (b).

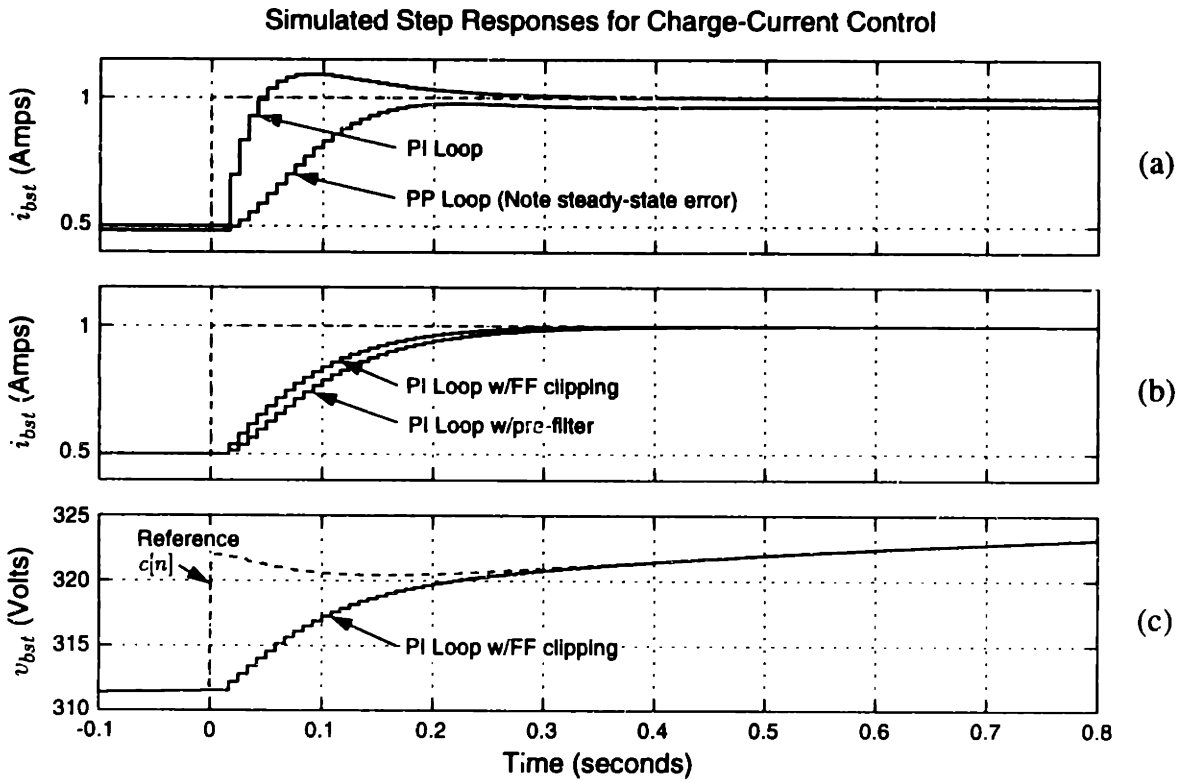



**Figure 5.13:** Simulated step responses for the various charge-current control schemes. (a) Raw PI and PP loops. (b) Modified PI loops. (c) Voltage profile.

The feedforward method is implemented by clipping the voltage accumulator of the PI voltage-loop controller. The update equation for the PI accumulator becomes

$$\sigma_v[n+1] = \sigma_v[n] + \begin{cases} \Delta_{max} & \text{if } (X[n] - x[n]) > \Delta_{max} \\ -\Delta_{max} & \text{if } (X[n] - x[n]) < -\Delta_{max} \\ X[n] + x[n] & \text{elsewhere} \end{cases} \tag{5.22}$$

where $\Delta_{max}$ is the clipping level. In practice, $\Delta_{max}$ is set to a value that is proportional to the amplitude of the current-command profile. The exact relationship was determined experimentally. A second method to eliminate the overshoot uses a pre-filter to effectively cancel the zero in the PI transfer function responsible for the overshoot. This can be accomplished using the low-pass pre-filter below:

$$\tilde{H}_p(z) = \frac{z}{z + \frac{g_2 - g_1}{g_1}}, \tag{5.23}$$

where $g_1$ and $g_2$ are the PI control gains from (5.7). The second method is preferred because it does not require prior knowledge of the current-command profile.

The charge-current control algorithm was tested using the 600-W bidirectional prototype and a lead-acid battery pack with a nominal voltage of 120 V and a capacity of 2.1 kWh. The final algorithm used PI control for the inner voltage loop. To eliminate overshoot, feedforward clipping was performed on the PI accumulator. The value of $R_{est}$ was pre-computed from measured parameters. Experimental results are plotted in Figures 5.14 and 5.15. Figure 5.14 shows the experimental response of the system to square-wave and sawtooth charging profiles. In each case the profiles vary between 0.5 and 1.0 A. The closed-loop transient and tracking performance is excellent. The transient behavior is nearly identical to that of the voltage loop plotted in Figure 5.8.

Figure 5.15 shows an expanded view of the charge current step response. The top trace in the figure clearly shows that $i_{bat}$ and $i_{bst}$ are proportional to each other as predicted by (5.20). The ratio is approximately 2.3, which agrees precisely with the predicted value of $N_2$ based on the coupling parameters in Table 2.3. Figure 5.15 (b) and (c) show the corresponding voltages $v_{bst}$ and $v_o$. (The voltage $v_o$ is measured across the battery terminals.) The effect of the ripple cancellation circuitry is clearly illustrated in this figure. The

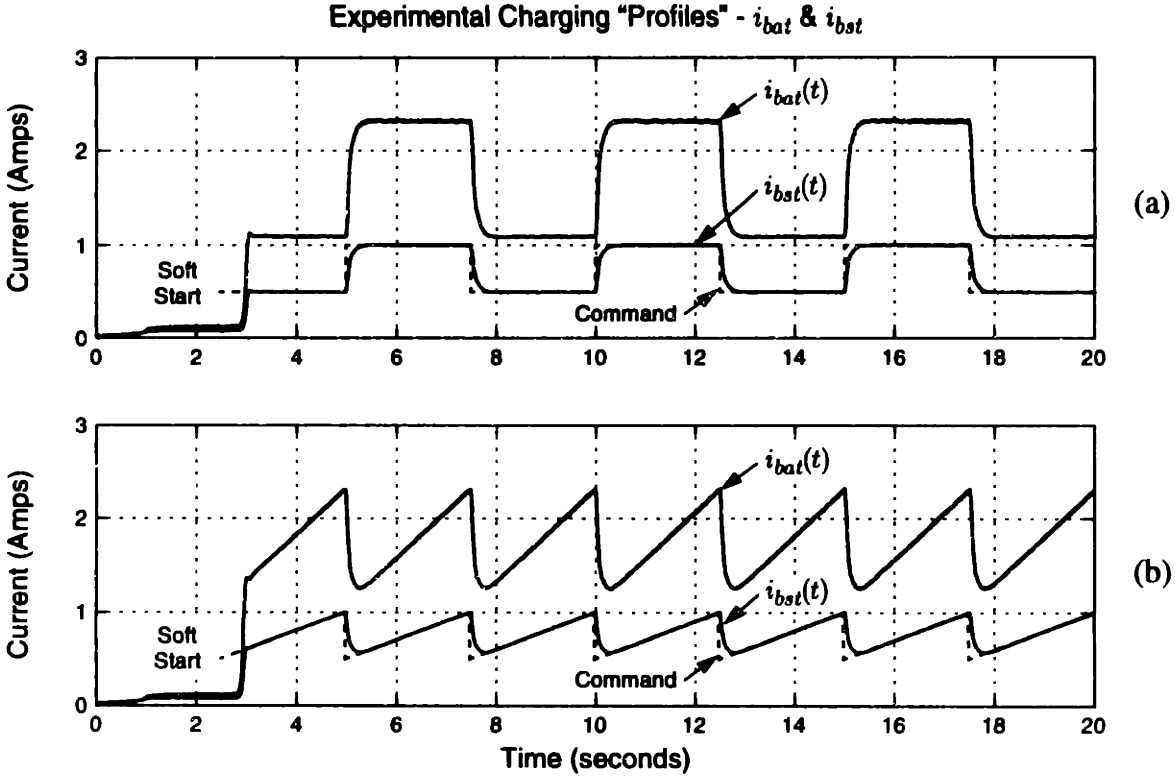



**Figure 5.14:** Experimental waveforms showing battery charging current profiles. (a) Step current profile. (b) Sawtooth current profile.

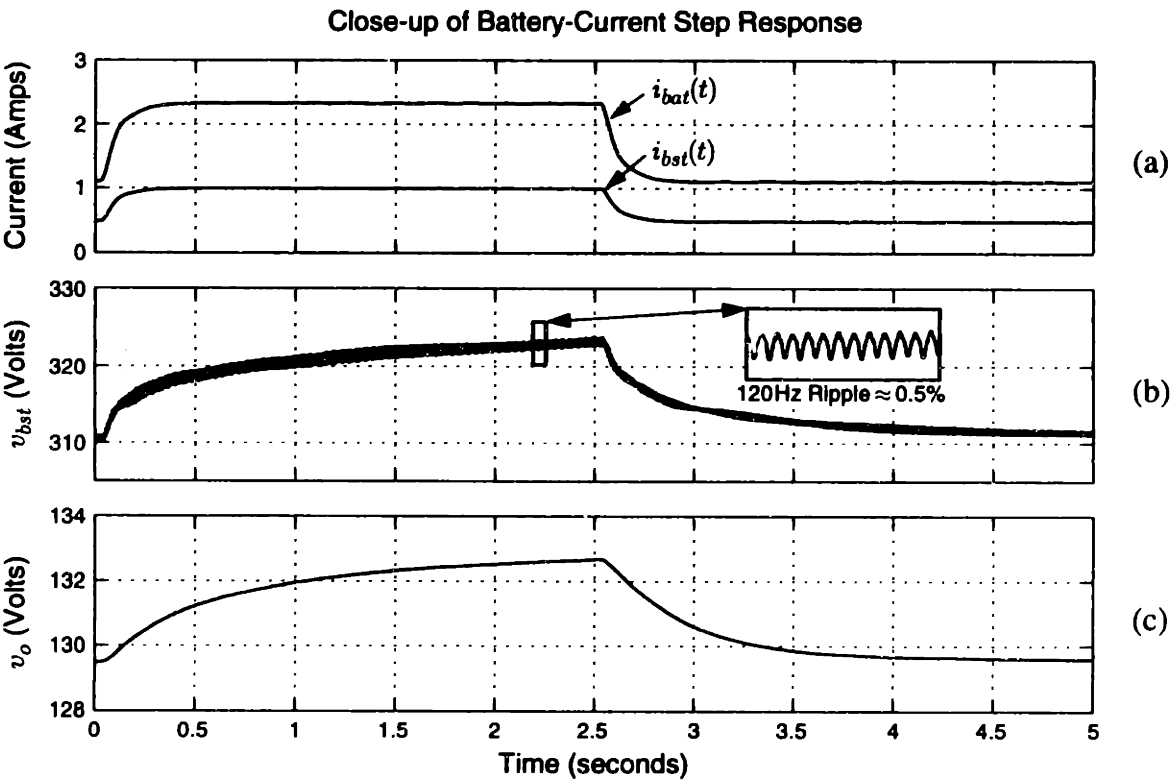



**Figure 5.15:** Close-up of experimental waveforms during a step change in charge current. (a) Boost and battery current. (b) Boost voltage. (c) Battery voltage.

120-Hz voltage ripple has an amplitude of approximately 0.5% (peak-to-peak) at the boost output. The ripple is essentially nonexistent in both current waveforms.

### 5.3.2 Discharge-Current Control

Control of the current during a battery discharge cycle is considerably less complex. Since power flows in reverse, from the battery to the AC utility, the "load" in the system is now the AC utility. The buck power-factor-correction mode described in Section 4.2 allows for direct control over the amplitude of the wave-shaped current injected into the utility. As a result, the dynamics of the battery and the DC/DC converter can be ignored.

Ideally, the AC utility voltage is sinusoidal with an amplitude $V_{ac}$ and a frequency $f_L$, e.g., $f_L = 60$ Hz.

$$v_{ac}(t) = V_{ac}\sin(2\pi f_L t) \tag{5.24}$$

Due to the action of the analog inner current-loop controller, the utility current is sinusoidal as well. Because the power flow is reversed, the current is 180 degrees out of phase with the voltage. The current can be described as

$$i_{ac}(t) = -I_{ac}\sin(2\pi f_L t) \tag{5.25}$$

where $I_{ac}[n] = k[n]V_{ac}$ and $k[n]$ is the programmable current gain of the inner current-loop controller. The gain $k[n]$ is computed by a DT controller synchronized with the half-line-cycle period $T_L$. Every $n^{\text{th}}$ cycle $k[n]$ is computed according to

$$k[n] = k[n-1] + g_1(I_{ref}[n] - I_{ac}[n]) \tag{5.26}$$

where $g_1$ is a control gain and $I_{ref}[n]$ is a target reference amplitude. Substituting $I_{ac}[n]/V_{ac}$ for $k[n]$ in (5.26) yields a first-order DT model for the amplitude of the AC utility current.

$$I_{ac}[n] = \alpha I_{ac}[n-1] + (1-\alpha)I_{ref}[n] \tag{5.27}$$

$$\alpha = \frac{1}{1 + g_1 V_{ac}} \tag{5.28}$$

A $z$-transform representation of (5.27) reveals a single DT pole at $z = \alpha$.

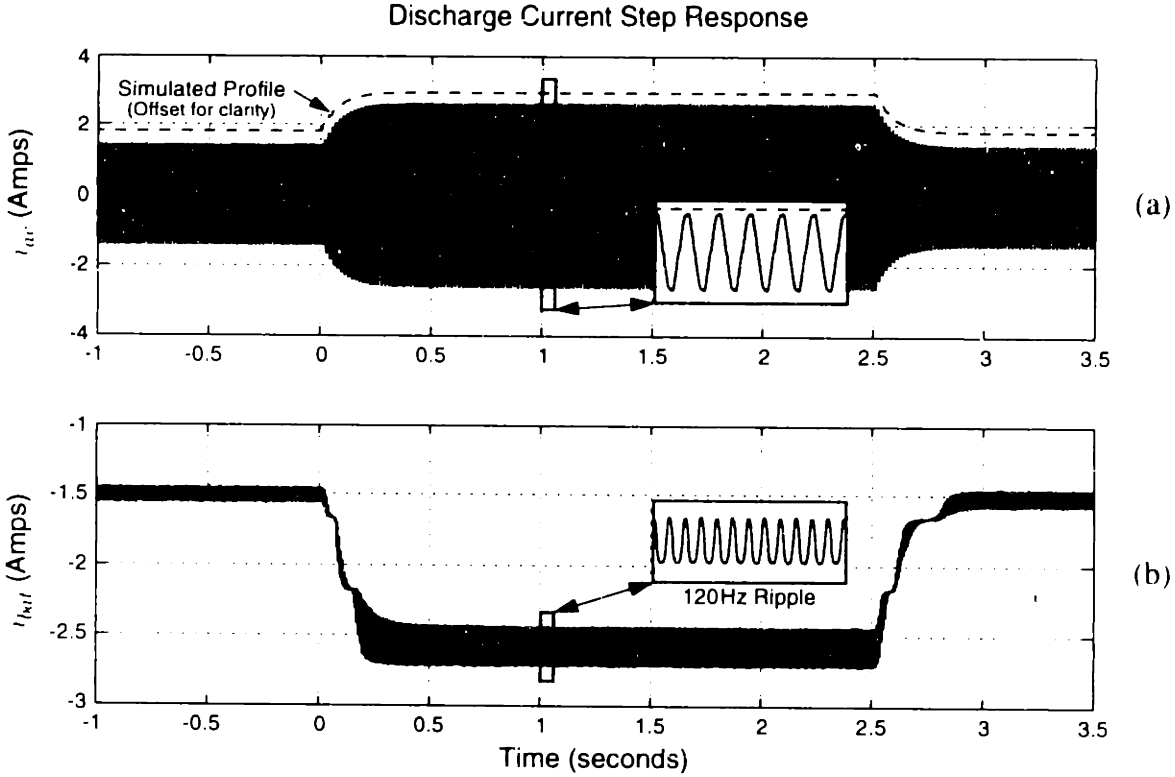



**Figure 5.16:** Experimental discharge current step response.

$$\tilde{H}(z) = \frac{I_{ac}(z)}{I_{ref}(z)} = \frac{1-\alpha}{1-\alpha z^{-1}} \tag{5.29}$$

The above algorithm was implemented and tested using the bidirectional prototype system. The gain $g_1$ is computed on-line to place the closed-loop pole at $z = 0.90$. With the system running, the reference current amplitude was stepped from 1.5 A to 2.5 A and back. The measured response is plotted in Figure 5.16.

Figure 5.16 (a) shows the step response of the utility current $i_{ac}$. A simulation of the expected current profile (dashed line) is included for comparison. Note that the figure plots a span of several seconds, so several hundred cycles of the sinusoidal current waveform are included. A small segment has been enlarged to reveal the underlying sinusoidal pattern. The envelope of $i_{ac}$ can be distinguished clearly in the figure, and it matches the simulated response precisely. Figure 5.16 (b) plots the corresponding step change in battery current. The enlarged segment highlights the 120-Hz ripple component to $i_{bat}$. The presence of 120-Hz ripple is no surprise since active ripple cancellation functions only during battery charging. The amplitude variations of the ripple component during step

changes in the current are caused by an intriguing interaction between the capacitance $C_{bst}$ and the mode 1 and 2 operating conditions of the DC/DC converter. This does not, however, impact on the performance of the discharge cycle.

## 5.4 General Multirate Control

The previous section described current-control algorithms well-suited to the battery charging application. The development of the charging-current algorithm, in particular, relied on convenient simplifications of the battery load model. This largely avoided potentially serious complications from the fact that the voltage loop operates on the *squared* output voltage and not directly on the output voltage. Similar simplifications are likely to be impossible for loads other than the typical EV battery.

The prototype inductively-coupled electronics are well-suited for supplying power to a wide range of loads outside of an EV battery. Many of these loads, electromechanical systems in particular, present complex load dynamics, and a more general approach is required to incorporate these dynamics into the control design. Two approaches for general multirate control design are described below. Their structure is defined and the techniques are demonstrated with an example design. A multirate digital current controller is designed for a resistive load. More complex electromechanical load examples are considered in Chapter 6, where the techniques presented here provide a building block for adaptive control.

The structure of a general multirate current-control algorithm is diagrammed in Figure 5.17. Three separate control loops are nested. When listed from highest to lowest control-loop tracking bandwidth, they are as follows: an analog inner-current loop to

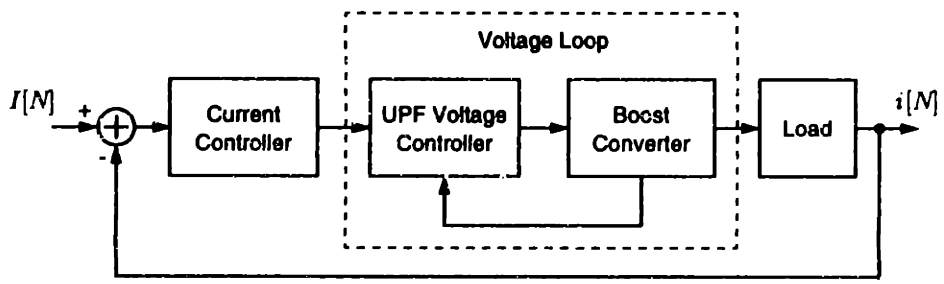



**Figure 5.17:** Current loop block diagram.

shape the boost converter input current; a DT voltage loop to control the output bus voltage; and an outermost DT loop to accomplish the final current control. In this case, the outermost DT current loop is wrapped around the PP voltage-loop controller described in Section 5.2. So the dashed "Voltage Loop" box in Figure 5.17 represents the boost converter circuitry combined with the PP voltage-loop controller from Figure 5.5.

The load dynamics for a wide range of loads are easily represented in the outer current loop by a driving-point admittance. Given the availability of a function relating applied terminal voltage to load current, a natural and convenient formulation for the current loop is to assume that load current will be sensed, and that a desired terminal voltage will be created by the action of the current-loop computation and the voltage amplifier (boost converter and voltage loop). However, the voltage loop controls the squared output voltage, not the output voltage. This complicates the formulation of a complete state-space description for the full three-loop system.

Since the voltage-loop convergence is guaranteed independently of load dynamics, simplifying assumptions are possible. Recall that the DT voltage loop operates with sample step index $n$. The DT current loop will be designed to operate with sample index $N$, where $n = QN$ and $Q$ is a positive integer. That is, every step of the current loop corresponds to $Q$ steps of the voltage loop. This multirate arrangement makes it possible to model the voltage-loop dynamics, from the standpoint of the outer current loop, in any of several simplified ways. Two approaches will be considered here: a delay model of the voltage loop appropriate for resistive loads, and a zero-order-hold model appropriate for loads representable as combinations of linear time-invariant (LTI) circuit elements and independent sources.

### 5.4.1 Delay Model

A delay model was employed in our prototype with a resistive load by selecting $Q$ and the closed-loop pole locations of the voltage loop so that the output bus voltage will converge to a new reference $X$ in a single step of the current-loop index $N$. The current loop computes a voltage reference at time $N$, which is squared and supplied as the command reference to the inner voltage loop. With the proper choice of $Q$ and the voltage-loop

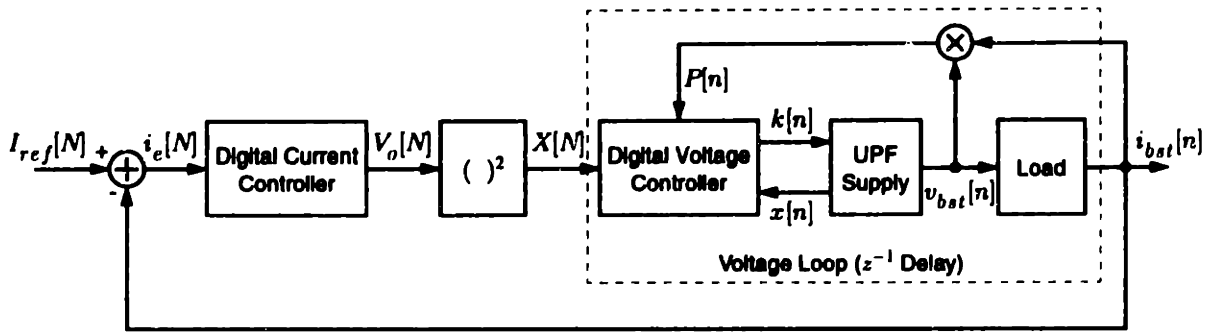



**Figure 5.18:** Multirate current-loop control structure.

poles, the output bus voltage will have converged to the reference command supplied by the current loop by the time $N+1$. Under these assumptions, the voltage loop may be modeled as a unit delay on the slow, current-loop time scale. This arrangement is illustrated in Figure 5.18. Signals in the figure are indexed by $n$ or $N$, depending on whether they are part of the fast voltage loop or slow current loop, respectively.

With a resistive load the current loop may be stabilized most simply with a first order DT compensator similar to PP compensator used in the voltage loop. The reference voltage $V_o$ is defined as

$$V_o[N] = V_o[N-1] + g_3(I_{ref}[N] - i_{bst}[N]) \tag{5.30}$$

where $g_3$ is a constant gain.

Since the action of the voltage loop is modeled as a unit delay on the time scale of the current loop, the output voltage applied to the load is equivalent to the delayed command signal, i.e.,

$$v_{bst}[N+1] = V_o[N] = V_o[N-1] + g_3(I_{ref}[N] - i_{bst}[N])\,. \tag{5.31}$$

Assuming a load resistance $R$, a state equation for $i_{bst}$ can be written using (5.31) and Ohm's law:

$$i_{bst}[N+1] = i_{bst}[N] + \frac{g_3}{R}(I_{ref}[N] - i_{bst}[N])\,. \tag{5.32}$$

Application of the $z$-transform to (5.32) yields a transfer function from $I_{ref}[N]$ to $i_{bst}[N]$ for the charging current loop:

$$\tilde{H}_I(z) = \frac{g_3/R}{z - 1 + g_3/R}. \tag{5.33}$$

In the $z$-plane, the closed-loop system has a single pole at

$$z = 1 - g_3/R. \tag{5.34}$$

The stability and transient characteristics of the current loop may be adjusted by selecting an appropriate gain $g_3$.

The delay-model technique just described was implemented using the 1.5-kW unidirectional prototype and the same 143.8-Ω load resistance as in the previous voltage loop experiments. The constant $Q$ was selected so that the current-loop index $N$ increments once for every 25 steps of the voltage-loop index $n$. The gain $g_3$ was selected so that the closed-loop pole of the current loop was located at $z = 0.20$. Figure 5.19 (a) and (b) show the current-loop response to square-wave and sawtooth current profiles, respectively. Each trace shows the current reference (dashed line) and the measured current $i_{bst}$ (solid line). As usual, the first approximately three seconds of the response in each case is dominated

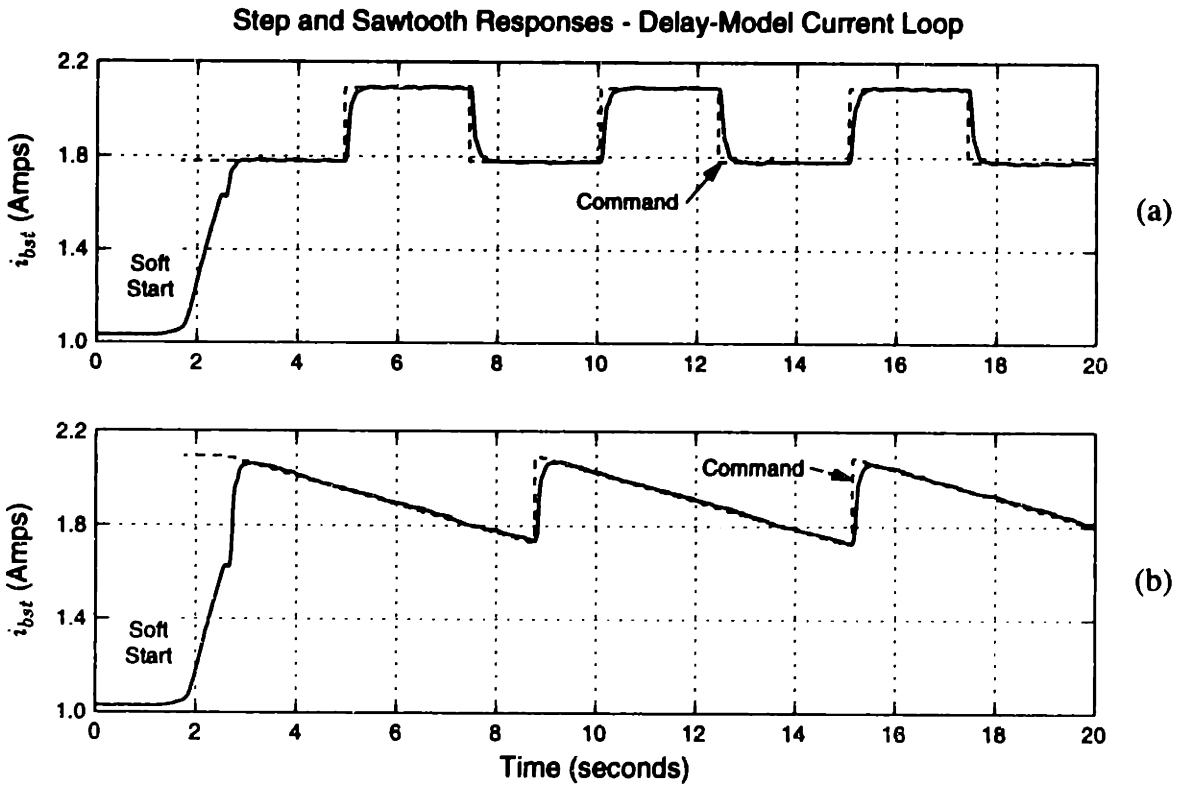



**Figure 5.19:** Step and sawtooth responses for the delay-model current loop.

by the soft-start mechanism built into the software. The closed-loop performance is excellent in both cases. The current-loop step response settles to within 10% of its final value in two steps of the index $N$. This response is consistent with the first-order behavior modeled by (5.32).

### 5.4.2 Zero-Order-Hold Model

For a resistive load, since the load terminal voltage and current are proportionally related, it is easy to step from (5.31) to (5.32) while developing the current loop in the previous subsection. For a more complicated LTI load model, the delay model of the voltage loop may not be as easy to apply. In this case, we can exploit the guaranteed, large-signal transient characteristics of the voltage loop to develop a second approach.

Once again, the DT current loop steps with index $N$, where $n = QN$ and $Q$ is a positive integer. Now, add the additional constraint that the voltage loop, when supplied with a new reference, will drive the output voltage to this reference in substantially less than $Q$ steps of the index $n$. This condition is ensured through judicious selection of $Q$ and the closed-loop pole locations of the voltage loop. Given these conditions, the voltage loop may be modeled as a zero-order-hold (ZOH) on the time scale of the outer current loop.

For an LTI load, the load terminal current can be related to the applied terminal voltage by an expression for the driving-point admittance of the load, represented henceforth by the CT Laplace transform $H(s)$.[5] Assuming that $Q$ steps of the index $n$ are substantially longer than the time required for the voltage loop to settle to a new command reference, the CT voltage applied to the load will appear "pulse like" throughout one step of the index $N$, i.e., the operation of the voltage loop will closely approximate that of a zero-order-hold. On each step of the current-loop index $N$, the current-loop controller will provide a command reference to the voltage loop and will also sample the load current. Figure 5.20 schematically illustrates this arrangement. The zero-order-hold block represents the boost converter with voltage loop.

---

5. Many loads of interest can be modeled as circuits consisting of LTI circuit elements, or as switched circuits that are piecewise LTI. Most batteries, on the other hand, tend to exhibit nonlinear driving-point current/voltage characteristics. It is often possible, however, to develop LTI or piecewise LTI battery models; see [107] for example.

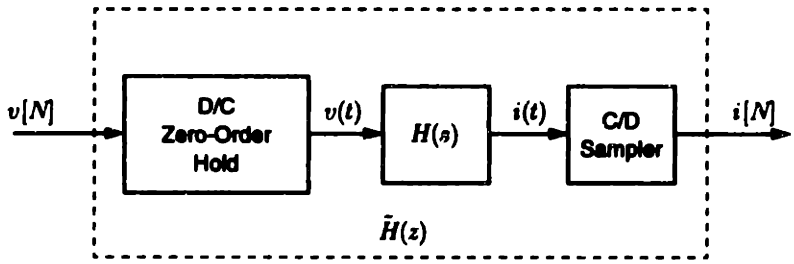



**Figure 5.20:** Driving-point characteristic.

The ZOH and sampling operations in Figure 5.20 model the interface between the CT driving-point characteristic of the load and the DT current loop [5]. The DT driving-point admittance can be described as a $z$-transform $\tilde{H}(z)$. The admittance $\tilde{H}(z)$ is related to $H(s)$ by a *step-invariant transformation* [97].

Given $Y(s)$, the DT transfer function $\tilde{H}(z)$ may be computed as follows:

- Compute the step response of $H(s)$. That is, calculate the inverse Laplace of $\frac{H(s)}{s}$.
- Sample the resulting continuous-time step response $y(t)$ to obtain $y[N] = y(NT)$.
- Determine the $z$-transform of $y[N]$, denoted by $\tilde{Y}(z)$.
- The $z$-transform $\tilde{Y}(z)$ represents the step response of the DT transfer function $\tilde{H}(z)$, i.e. $\frac{z\tilde{H}(z)}{z-1}$. To find $\tilde{H}(z)$, multiply $\tilde{Y}(z)$ by $\frac{z-1}{z}$.

Tables relating common functions $H(s)$ to their "pulse" transfer functions $\tilde{H}(z)$ may be found in many texts. (See [5], for example.)

In general, complex load models will add state variables to the overall current-loop state-space description. In such cases, the current-loop state equations will generally be more complicated than those summarized in (5.32). Gains might have to be adapted, and/or different compensation schemes might be needed for different loads. Fortunately, the digital implementation of the current-loop controller accommodates these changes.

The approach outlined in this section requires that the voltage loop converge to its reference in substantially less than $Q$ steps of the index $n$. This limits the performance of the current loop. In principle, however, the voltage loop can be made *deadbeat* [49], i.e., the voltage loop can converge in two steps of the index $n$ or one electrical input line cycle. This, of course, is subject to the limitations imposed by the maximum current command

that can be followed by the inner current loop and by the discharge rate made possible by the loading conditions. Nevertheless, the achievable, practical performance appears to be more than adequate for even high performance electromechanical applications.

## 5.5 Summary

This chapter described digital control techniques for the inductively-coupled architectures discussed in Chapters 3 and 4. Specific current-control algorithms were developed for the charging and discharging of EV batteries. In addition, large-signal linear voltage- and current-control techniques were also developed. These techniques form the basis for general multirate control, which will be developed further in Chapter 6.

# Chapter 6

# Adaptive Control and Estimation

This chapter presents adaptive-control schemes, which estimate load-model parameters and adaptively "tune" a digital controller. These methods for adaptive control both demonstrate and extend the general multirate techniques presented in Chapter 5. Estimation algorithms are presented that utilize specific recursive formulations, which provide adequate noise immunity in a power-electronic environment. The techniques are demonstrated using two example inductively-coupled servomechanical systems and the 1.5-kW prototype electronics described in Chapter 3.

## 6.1 Background

The general multirate control techniques developed in Chapter 5 presume that the driving-point impedance of the load can be described by a LTI transfer-function model. The time invariance constraint implies that the load model consists of constant coefficients. In addition, these coefficients must be known so that conventional design techniques apply. Not surprisingly, a number of desirable control applications do not fit this description. In particular, an array of servomechanical control problems exist where variations in mechanical properties, loading, or external disturbances lead to unpredictable changes in the load model parameters. In some cases the entire structure of the load model may change, increasing or decreasing in order. Such changes can adversely affect even the most robust control designs, possibly leading to instability.

Adaptive or "self-tuning" control provides a mechanism whereby the controller design adjusts automatically in response to system changes. Adaptive control systems have been widely studied in published literature [29, 48, 112]. The name generally refers to a two step process of system parameter estimation and control adaptation. Both of these tasks

must be accomplished on-line to permit the controller to track time-varying system parameters. This imposes certain limitations on the possible implementations. In addition, the adaptation algorithm must be numerically suitable for digital implementation and sufficiently robust to noise and model inaccuracies. This chapter addresses these concerns and presents techniques specifically suited to the power electronic and digital control environment of the prototype inductively-coupled systems.

The remainder of this chapter is organized as follows. Section 6.2 discusses two approaches to parameter estimation. Algorithms are presented along with a discussion of the trade-offs associated with each approach. Sections 6.3 and 6.4 present two different experimental implementations using example servomechanical systems: a water-bath temperature control system and a motor-speed control system. The example systems allow for time-varying changes in the load characteristics so that the control adaptation can be evaluated. In each case experimental results are presented, and the adaptive control performance is compared to that of a non-adaptive design.

## 6.2 Parameter Estimation

Parameter estimation refers to the process by which the coefficients of a parameterized system model can be estimated from observations of the input and output waveforms. These estimates can then be used as a basis on which to update the controller design. Thus, parameter estimation is fundamental to the process of adaptive control. However, it also finds wide use outside the control arena. As a general approach for finding unknown model parameters, estimation can be applied in both controlled or uncontrolled dynamic systems for process monitoring, error detection, and performance analysis.

The process of parameter estimation involves computing or refining a set of parameter estimates, which minimize the error between experimental and modeled responses according to a prescribed criteria. This criteria may vary depending on the expected properties of the error. Examples include the absolute error and the absolute squared error or stochastic properties, such as the maximum likelihood or Bayesian criteria. The absolute squared error or "least-squared error" is by far the most widely used. It can be shown to produce unbiased parameter estimates in the presence of white noise [48].

Parameter estimation algorithms can be divided into two categories: recursive and non-recursive. Non-recursive algorithms, such as ordinary least-squares, require that a block of $N$ observations is stored or otherwise made available prior to the calculation. Matrix algebra is then applied to the block of data as a whole, yielding a single accurate estimate. For a typical application, $N$ may be several hundred or several thousand. Such high numbers can quickly swamp the storage space and computational ability of many embedded microcontrollers. As a result, non-recursive algorithms are often considered unsuitable for on-line or "real-time" applications. The following two subsections describe recursive algorithms, which avoid these pitfalls.

### 6.2.1 Recursive Least Squares

Recursive least squares (RLS) estimation is very attractive for real-time applications. Recursive least-squares estimation is an iterative reformulation of conventional least-squares estimation [48]. Because the algorithm is recursive, very few data points need to be stored between iterations, and computations occur at specified intervals with a predictable computational complexity. Figure 6.1 diagrams the basic layout of a DT recursive estimator.

The object is to estimate the parameters of the DT transfer function $\tilde{H}(z)$, which takes $u[k]$ as an input and produces $y[k]$ as an output. The variable $k$ is used here as a DT index. The transfer function is proper and can be written as

$$\tilde{H}(z) = \frac{b_1 z^{m-1} + b_2 z^{m-2} + \ldots + b_m}{z^m + a_1 z^{m-1} + a_2 z^{m-2} + \ldots + a_m} \tag{6.1}$$

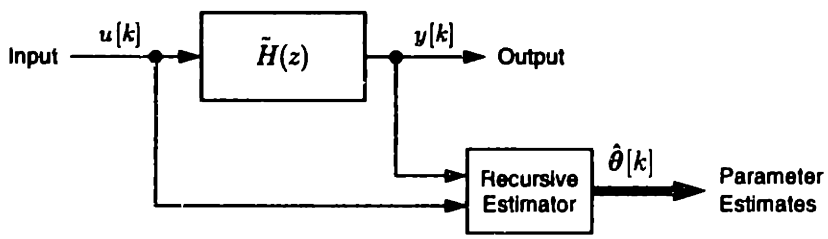



**Figure 6.1:** Block diagram of a DT recursive parameter estimator.

where the denominator polynomial has order $m$ and the coefficients $a_i$ and $b_i$ may be slowly *time varying*. The RLS estimator requires observations of the present output $y[k]$ and a regressor $\boldsymbol{r}[k]$. The regressor $\boldsymbol{r}[k]$ is a vector that contains a short set of previous input and output samples. The output of the estimator block is a another vector $\hat{\boldsymbol{\theta}}[k]$, which contains the parameter estimates $\hat{a}_i$ and $\hat{b}_i$. (The "hat" notation is used throughout to indicate estimated quantities.) The corresponding regressor and parameter vectors for the expression in (6.1) are

$$\boldsymbol{r}[k] = \begin{bmatrix} -y[k-1] \\ \vdots \\ -y[k-m] \\ u[k-1] \\ \vdots \\ u[k-m] \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{\theta}}[k] = \begin{bmatrix} \hat{a}_1 \\ \vdots \\ \hat{a}_m \\ \hat{b}_1 \\ \vdots \\ \hat{b}_m \end{bmatrix}. \tag{6.2}$$

The recursive least-squares algorithm is shown in (6.3) below:

$$\begin{aligned} \varepsilon[k] &= y[k] - \boldsymbol{r}[k]^T \hat{\boldsymbol{\theta}}[k-1] \\ \boldsymbol{g}[k] &= \frac{\boldsymbol{P}[k-1]\boldsymbol{r}[k]}{1 + \boldsymbol{r}[k]^T \boldsymbol{P}[k-1]\boldsymbol{r}[k]} \\ \hat{\boldsymbol{\theta}}[k] &= \hat{\boldsymbol{\theta}}[k-1] + \boldsymbol{g}[k]\varepsilon[k] \\ \boldsymbol{P}[k] &= \boldsymbol{P}[k-1] - \boldsymbol{g}[k]\boldsymbol{r}[k]^T \boldsymbol{P}[k-1] \end{aligned} \tag{6.3}$$

where $\varepsilon[k]$ is a scalar prediction error, $\boldsymbol{g}[k]$ is a gain vector and $\boldsymbol{P}[k]$ is a weighting matrix, typically referred to as the "covariance matrix" [29]. The algorithm in (6.3) is executed from top to bottom at each iteration of the index $k$.

After $N$ iterations, the recursive algorithm can, in principle, return the exact parameter estimates that would occur using non-recursive least squares on the same $N$ data points. However, a perfect match requires very specific initial conditions for $\boldsymbol{P}[k]$ and $\hat{\boldsymbol{\theta}}[k]$. In practice, the initial value of the covariance matrix is set to a large constant multiplied by an identity matrix, i.e., $\boldsymbol{P}[0] = p_i \boldsymbol{I}$. Although this can cause large initial transients in $\hat{\boldsymbol{\theta}}[k]$, the large covariance assures rapid parameter convergence.

The RLS algorithm just described refines its parameter estimates at each iteration. So after $N$ iterations the accuracy of the estimate accumulates the contributions from all $N-1$ previous iterations. This prevents the algorithm from accurately tracking time-varying parameters unless the effect of distant data points can somehow be eliminated. This can be accomplished by incorporating a "forgetting factor" $\rho$ to the algorithm in (6.3). The resulting formulation is often referred to a RLS with exponential forgetting and can be written as

$$\begin{aligned}
\varepsilon[k] &= y[k] - \boldsymbol{r}[k]^T\hat{\boldsymbol{\theta}}[k-1] \\
\boldsymbol{g}[k] &= \frac{\boldsymbol{P}[k-1]\boldsymbol{r}[k]}{\rho + \boldsymbol{r}[k]^T\boldsymbol{P}[k-1]\boldsymbol{r}[k]} \\
\hat{\boldsymbol{\theta}}[k] &= \hat{\boldsymbol{\theta}}[k-1] + \boldsymbol{g}[k]\varepsilon[k] \\
\boldsymbol{P}[k] &= \frac{1}{\rho}\Big(\boldsymbol{P}[k-1] - \boldsymbol{g}[k]\boldsymbol{r}[k]^T\boldsymbol{P}[k-1]\Big)
\end{aligned} \tag{6.4}$$

where $\rho$ is a constant very close to 1, with a value of $\rho = 0.99$ being typical. The forgetting factor modifies the minimization criteria of the RLS algorithm so that errors that occurred at time index $k-i$ are weighted by $\rho^i$. The number of samples effectively kept in "memory" is roughly proportional to $1/(1-\rho)$ [48].

Exponential forgetting does, however, have its disadvantages. If the forgetting factor $\rho$ is chosen too large, it will lead to slow convergence of the estimates, and if the forgetting factor is chosen to small, it will result in noisy estimates, which are based on few data points. Thus, the choice of $\rho$ represents a trade-off between parameter tracking and disturbance rejection. A related problem, known as "covariance matrix explosion," may occur during moments of low excitation is [48]. In the absence of a significantly exciting input, the quantity $\boldsymbol{P}[k-1]\boldsymbol{r}[k]$ will approach zero, in which case $\boldsymbol{g}[k]$ and $\boldsymbol{P}[k]$ begin to increase exponentially. In turn, the parameter estimates become increasingly sensitive to the prediction error $\varepsilon[k]$, and the slightest noise or model inaccuracy can lead to erratic or erroneous parameter estimates. In practice, this effect can be mitigated to a large degree by skipping the RLS update if a defined variable, such as the output, changes by less than an amount $\delta$ between samples. This topic is discussed again near the end of this chapter.

### 6.2.2 The Lambda Method

The term 'lambda method' (LM) is used here to describe a modification to the RLS algorithm, which will be referred to as LM-RLS. The technique relies on an "operator transformation" that allows the differential operator in a continuous-time (CT) transfer function model to be replaced without approximation. The result is a DT estimation algorithm, which operates on transformed or "filtered" observations of the input and output data. An estimation algorithm can then be designed around the transformed model. This novel technique is described briefly in [48], but otherwise appears to have been largely overlooked in the power-electronics literature. However, experiments have demonstrated a substantial reduction in noise sensitivity can be achieved over conventional RLS.

The LM-RLS technique is developed around a CT transfer function model of the target system. A suitably parameterized rational CT transfer can be written as

$$H(s) = \frac{b_1 s^{m-1} + \ldots + b_m}{s^m + a_1 s^{m-1} + \ldots + a_m} \tag{6.5}$$

where the denominator polynomial has order $m$ and the coefficients $a_i$ and $b_i$ may be *time varying*. The coefficients of $H(s)$ equivalently describe a linear differential equation model of the system written as

$$s^m y(t) + a_1 s^{m-1} y(t) + \ldots + a_m y(t) = b_1 s^{m-1} u(t) + \ldots + b_m u(t) \tag{6.6}$$

where $s$ now acts as the derivative operator $d/dt$. An operator transformation can be applied to (6.6), whereby the $s$ operator is replaced by a new "lambda" operator

$$\lambda = \frac{1}{\tau_\lambda s + 1} \tag{6.7}$$

where $\tau_\lambda$ is a positive time constant. Using (6.7) to eliminate $s$ from (6.6) yields a new linear model, which can be written as

$$y(t) + \alpha_1 \lambda y(t) + \ldots + \alpha_m \lambda^m y(t) = \beta_1 \lambda u(t) + \ldots + \beta_m \lambda^m u(t) \tag{6.8}$$

where the reformulated parameters $\alpha_i$ and $\beta_i$ are algebraically related to the starting parameters $a_i$, $b_i$ and the time constant $\tau_\lambda$. Equation (6.8) is special in that the addends $\lambda y(t)$, $\lambda u(t)$, etc., represent inputs and outputs operated on by $\lambda$. More specifically, the $\lambda$

operator specifies a first-order low-pass filter with a time constant $\tau_\lambda$. Furthermore, powers of $\lambda$, i.e., $\lambda^2$, $\lambda^3$, etc., denote quantities that have been double-filtered, triple-filtered and so on.

The reformulated parameters from (6.8) can be estimated directly using a discrete-time RLS algorithm. Assuming that the filtered CT quantities are available, they can be sampled every $T_k$ seconds, in sync with the corresponding DT algorithm iterated by the index $k$. The sampled data is compiled into a regressor vector $\boldsymbol{r}_\lambda[k]$, and the RLS algorithm of (6.4) can be applied directly provided the vectors $\boldsymbol{r}[k]$ and $\hat{\boldsymbol{\theta}}[k]$ are replaced by

$$\boldsymbol{r}_\lambda[k] = \begin{bmatrix} -\lambda y(kT_k) \\ \vdots \\ -\lambda^m y(kT_k) \\ \lambda u(kT_k) \\ \vdots \\ \lambda^m u(kT_k) \end{bmatrix} \quad \text{and} \quad \hat{\boldsymbol{\theta}}_\lambda[k] = \begin{bmatrix} \hat{\beta}_1 \\ \vdots \\ \hat{\beta}_m \\ \hat{\alpha}_1 \\ \vdots \\ \hat{\alpha}_m \end{bmatrix}. \tag{6.9}$$

The final step is to recover estimates of the starting parameters $\hat{a}_i$ and $\hat{b}_i$ from their known relationships to $\hat{\alpha}_i$, $\hat{\beta}_i$ and $\tau_\lambda$.

The convergence properties of the LM-RLS method are strongly affected by the selection of the filter time constant $\tau_\lambda$. Experimental evidence suggests that $\tau_\lambda$ should not exceed the nominal settling time of the target system. Doing so will unnecessarily slow the convergence rate of the parameter estimates. A compromise may be necessary, however, if $\tau_\lambda$ is increased to attenuate noise from an external source. Conversely, decreasing $\tau_\lambda$ can speed up the convergence rate, but the convergence transient can be expected to become increasingly violent [48]. Nevertheless, the convergence properties of the LM-RLS method demonstrate a marked improvement over the direct RLS approach. This will be demonstrated by the experimental tests in the following sections.

## 6.3 Bath Temperature Control

A water-bath temperature control system was devised in order to apply the parameter estimation techniques just described. The goal was to implement real-time adaptive control, which delivers stable and predictable closed-loop tracking performance subject to load

parameter variations. The temperature control of a water-bath was selected as an example application because it is realistic, yet not overly complicated. Thus, it provides a clear illustration of the techniques. Also, because the application transfers power to an aqueous environment, it benefits from the inherent safety and isolation provided by inductive coupling. The example is representative of a number of real-world applications, which could apply inductive-coupling as a non-contact means to transfer power across an environmental boundary.

### 6.3.1 Experimental Setup

The experimental setup is illustrated in Figure 6.2. A 1-kW immersion heater was placed inside a two gallon water bucket. Electrical power for this heater was supplied via an inductive coupling using the 1.5-kW unidirectional prototype described in Chapter 3. Chilled water was supplied to the bucket at a constant rate from a cold water supply, and excess water was actively pumped out through a drain hose. Adjusting the level of the drain up or down allowed for external control of the water volume, with a range from approximately 0.6 to 1.5 gallons. A mixer kept the water in constant circulation so that the temperature throughout the bucket was homogeneous.

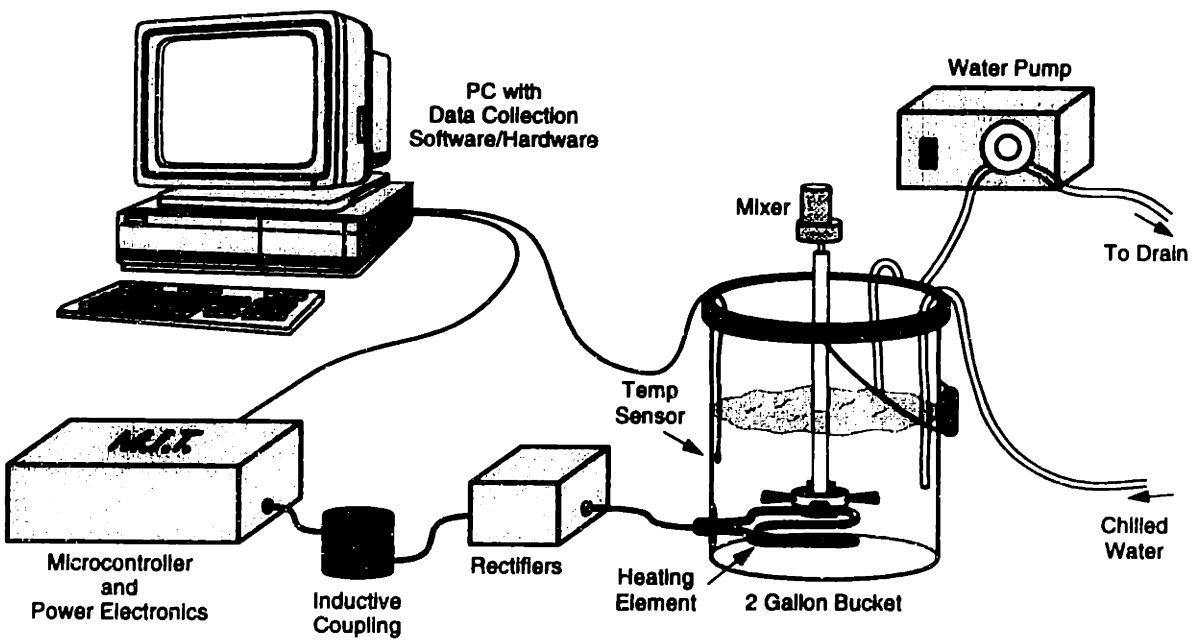



**Figure 6.2:** Setup for the water-bath temperature control experiments.

The PC illustrated in Figure 6.2 was used to record experimental data for later analysis. This was accomplished using a commercially-available data collection system from Advantech [2]. This system recorded two electrical signals: the boost voltage $v_{bst}(t)$ from the prototype electronics and the water temperature $T_b(t)$ from a precision temperature sensor. For use in control, the liquid temperature $T_b(t)$ was also relayed via the PC to the digital microcontroller board. It is important to emphasize that the PC in this experimental setup simply provided data collection for the experiment. *All digital control and estimation algorithms were implemented on-board the embedded 80C196KC microcontroller board.* A control panel attached to the microcontroller board provided a user interface for setting control options, temperature setpoints, etc.

### 6.3.2 Control Design

The goal of the control system is to provide stable, predictable tracking performance subject to load parameter variation. Naturally, this can be accomplished using adaptive control. The design of such a controller begins with the general multirate procedure described in Chapter 5. Subsection 5.4.2 calls for a nested-loop multirate control structure, where the "fast" voltage-loop response of the boost converter can be modeled as a ZOH on the time scale of a "slow" outer loop. This procedure allows an LTI model of the driving-point impedance of the load to be incorporated into the DT control model using a step-invariant transformation.

A diagram of the overall adaptive temperature control structure appears in Figure 6.3. If the adaptive control additions are neglected, the structure is similar to the multirate structure suggested in Figure 5.17. However, in this case the "load" is the thermal system of the water bath. The "power electronics" block in the figure represents the boost converter and PP voltage-loop controller as described in Chapter 5. The voltage-loop controller, which is not explicitly illustrated in Figure 6.3, acts to drive the boost output voltage $v_{bst}$ to track a reference command $V_c$. This loop operates on a time index $n$ in sync with the period $T_L$ of the rectified utility voltage. An outermost loop controls the actual water-bath temperature by sending appropriate reference commands $V_c$ to the power electronics. This outer loop operates on an index $k$, where $n = Qk$ and $Q$ is a positive integer. Thus, if

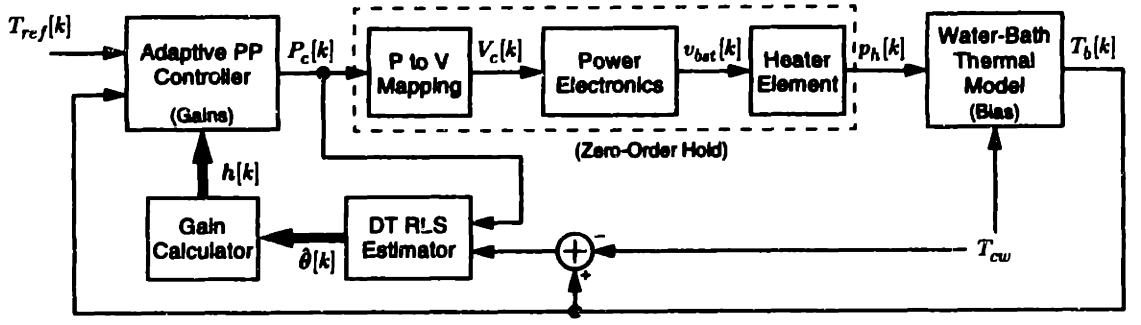



**Figure 6.3:** Block diagram of the temperature control system.

$Q$ is made large, the voltage loop will converge in substantially less than $Q$ steps of the index $n$. This is the basis for the ZOH approximation.

Figure 6.4 shows equivalent load models for the heater and water-bath systems. The circuit on the left models the DC/DC converter with a resistive submersion heater attached. The model includes an ideal transformer, a droop resistance $R_d$ and a heater resistance $R_h$. The power dissipated in the heating element can be written as

$$P_h = i_h^2 R_h = \frac{N_2^2 v_{bst}^2 R_h}{(R_d + R_h)^2} \tag{6.10}$$

where $N_2$ is the effective turns ratio of the inductive coupling. Equation (6.10) shows that $P_h$ is proportional to the square of the output voltage $v_{bst}$. This relationship can easily be inverted so that the appropriate voltage reference $V_c$ can be found to achieve a target power dissipation $P_c$. That relationship is

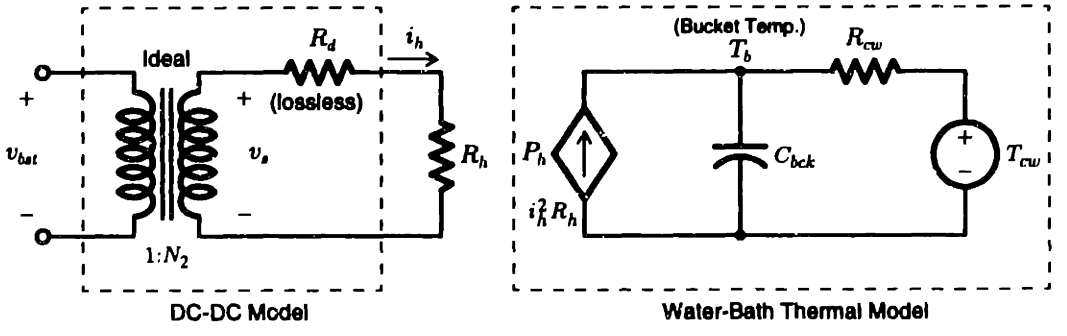



**Figure 6.4:** Equivalent load models for the heater and water-bath system.

$$V_c = \frac{R_d + R_h}{N_2}\sqrt{\frac{P_c}{R_h}}. \tag{6.11}$$

Equation (6.11) is used in the "P-to-V mapping" block of Figure 6.3. As a result it is possible to collapse the dashed box in Figure 6.3 into a single ZOH model relating the command $P_c$ to the heater dissipation $P_h$.

The thermal circuit in Figure 6.4 models the relationship between the power dissipated in the heater $P_h$ (i.e., the heat flow) and the water-bath temperature $T_b$. The capacitance $C_{bck}$ in the model represents the heat capacity of the water. The voltage source $T_{cw}$ is the temperature of the cooling water supply, and $R_{cw}$ is the effective thermal resistance between the bucket and cooling water temperatures. Applying Kirchhoff's current law to this thermal equivalent circuit yields the following first-order differential equation:

$$P_h(t) + \frac{T_{cw} - T_b(t)}{R_{cw}} = C_{bck}\frac{dT_b(t)}{dt}. \tag{6.12}$$

From (6.12) a transfer function relating $P_h(t)$ to $T_b(t)$ can be written as

$$H(s) = \frac{T_b(s)}{P_h(s)} = \frac{1/C_{bck}}{s + 1/(R_{cw}C_{bck})}. \tag{6.13}$$

The transfer function $H(s)$ is the driving-point characteristic of the load. The DT driving-point characteristic can now be found using the step-invariant transformation as outlined in subsection 5.4.2:

$$\tilde{H}(z) = \frac{T_b(s)}{P_c(s)} = \frac{b_1}{z + a_1} \tag{6.14}$$

where

$$a_1 = -e^{-T_k/(R_{cw}C_{bck})} \qquad \text{and} \qquad b_1 = R_{cw}(1 + a_1). \tag{6.15}$$

Because $\tilde{H}(z)$ includes the ZOH block, the DT transfer function relates the commanded power $P_c[k]$ to the sampled temperature $T_b[k]$. The DT load model in (6.14) can now be used to design a DT compensator which yields the desired closed-loop performance. Since the performance of the PP compensation scheme has already been proven, it is a natural choice for the temperature-loop compensation.

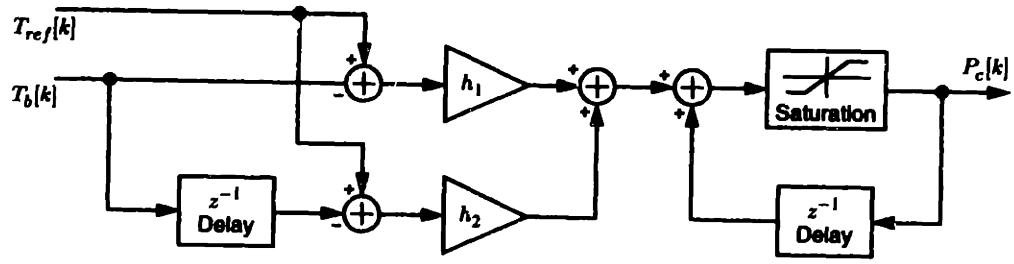



**Figure 6.5:** Block diagram of the first-order PP controller.

The control command $P_c[k]$ for a first-order PP temperature controller is

$$P_c[k] = P_c[k-1] + h_1(T_{ref}[k] - T_b[k]) + h_2(T_{ref}[k] - T_b[k-1]) \tag{6.16}$$

where $T_{ref}[k]$ is the temperature reference and $h_1$ and $h_2$ are feedback gains. A block diagram of this control structure appears in Figure 6.5. The saturation block illustrated in Figure 6.5 was added in practice to limit the maximum and minimum power command. The closed-loop transfer function for the temperature controller is

$$\tilde{H}_{CL}(z) = \frac{(1 + x_1 + x_2)z}{z^2 + x_1 z + x_2} \tag{6.17}$$

provided the gains $h_1$ and $h_2$ are selected according to

$$h_1 = \frac{x_1 - a_1 + 1}{b_1} \qquad \text{and} \qquad h_2 = \frac{x_2 + a_1}{b_1}. \tag{6.18}$$

Clearly, it is possible to locate the closed-loop poles arbitrarily in the $z$-plane.

### 6.3.3 Adaptive Updating

The previous subsection described the design of a fixed temperature controller provided the coefficients of the load model are known. As might be expected, it is possible to estimate these coefficients in real-time and use them to adaptively update the controller gains. The procedure has already been illustrated in Figure 6.3.

A discrete-time RLS estimator with exponential forgetting was used to estimate the parameters of DT transfer function in (6.14) on-line. The regressor and parameter estimate vectors for this case are

$$\boldsymbol{r}[k] = \begin{bmatrix} -T_b[k-1] + T_{cw} \\ p_h[k-1] \end{bmatrix} \qquad \hat{\boldsymbol{\theta}}[k] = \begin{bmatrix} \hat{a}_1 \\ \hat{b}_1 \end{bmatrix}. \tag{6.19}$$

It should be noted that the cooling water temperature $T_{cw}$ must be subtracted from the observed temperature in order to avoid an unwanted bias in the estimates. The RLS algorithm updates the estimates $\hat{\boldsymbol{\theta}}[k]$ on each step of the index $k$. The control gains $h_1$ and $h_2$ are updated after each iteration in order to achieve a closed-loop transfer function with two real poles at identical locations on the $z$-axis, i.e., $z_1 = z_2 = p$. Thus, $h_1$ and $h_2$ are computed as

$$h_1 = \frac{-2p - \hat{a}_1 + 1}{\hat{b}_1} \qquad \text{and} \qquad h_2 = \frac{p^2 + \hat{a}_1}{\hat{b}_1}. \tag{6.20}$$

### 6.3.4 Results

The adaptive temperature control system was implemented in software on the 80C196KC microcontroller board. The C-code source listings can be found in Appendix F. So that direct performance comparisons could be made, both fixed and adaptive controllers were implemented. The fixed controller was optimized for operation at the low end of the water volume range, approximately 0.6 gallons. The load model parameters for $H(s)$ in (6.13) were approximated experimentally at this level. The heat capacity, thermal resistance, and cooling water temperature were measured at

$$C_{bck} = 6672 \ \frac{\text{J}}{^\circ\text{F}}, \qquad R_{cw} = 0.0625 \ \frac{^\circ\text{F}}{\text{W}}, \qquad \text{and} \qquad T_{cw} = 75 \ ^\circ\text{F}. \tag{6.21}$$

The closed-loop performance of both the DT temperature loops was targeted to have two real poles at $z_1 = z_2 = 0.80$. A sample period of $T_k = 20$ s was used, so the inner- and outer-loop indices are related by $n = Qk$, where $Q = 2400$. The RLS parameter estimation for the adaptive controller was configured with a forgetting factor $\rho$ of 0.99. Thus, approximately 60 minutes of data is retained in "memory." No initial guess was supplied for $\hat{\boldsymbol{\theta}}[0]$. Rather, the covariance matrix $\boldsymbol{P}$ was initialized as 10,000 times an identity matrix, and the estimates were allowed to converge on their own. In order to assure stable control during start-up, the fixed controller was engaged for the first 10 minutes. This allowed time for the estimates to converge before adaptive control was engaged.

Simulated and experimental results appear in Figures 6.6 and 6.7. A square wave in temperature was commanded in order to demonstrate the control performance. In each case, the system was initiated with the fluid level set at its lowest level of 0.6 gallons. At this level the closed-loop performance of both the adaptive and fixed temperature controllers is nearly ideal. The step transients are consistent with the closed-loop pole locations, and the tracking performance is good. After approximately 90 minutes of operation the fluid level was increased to 1.5 gallons. This volume change takes approximately 5 minutes to occur. The change drives the fixed controller away from its optimal operating point and significant overshoot results. The adaptive controller, on the other hand, quickly adapts to the altered load model, and the desired closed-loop performance is maintained.

The simulated responses in Figure 6.6 indicate the expected parameter convergence rate. The initial transient in the parameter estimates settles in approximately 20 minutes, or about 60 iterations. However, after the abrupt change in parameters at $t=90$ minutes, the convergence rate has slowed to approximately 60 minutes, which is consistent with the programmed value of the forgetting factor. The initial high convergence rate can be attributed to large starting value for the covariance matrix $\boldsymbol{P}$.

Several attempts were made to improve the convergence rate by decreasing the forgetting factor $\rho$. While it was found that some improvement was possible, the increased noise sensitivity quickly becomes intolerable. As noise susceptibility increases the accuracy of the parameter estimates degrades, which leads to poor closed-loop performance or even instability. Since parameter tolerance tends to decreases as control complexity increases, higher-order systems offer even less flexibility. The optimal value for $\rho$ requires a trade-off between performance and the adaptation rate.

## 6.4 Motor-Speed Control

The water-bath example was intended as a simple demonstration of the adaptive control techniques. However, many practical systems are higher order. As a result, they are often substantially more susceptible to noise, performance degradation and instability. The experimental work presented here suggests that adaptive control based on the lambda

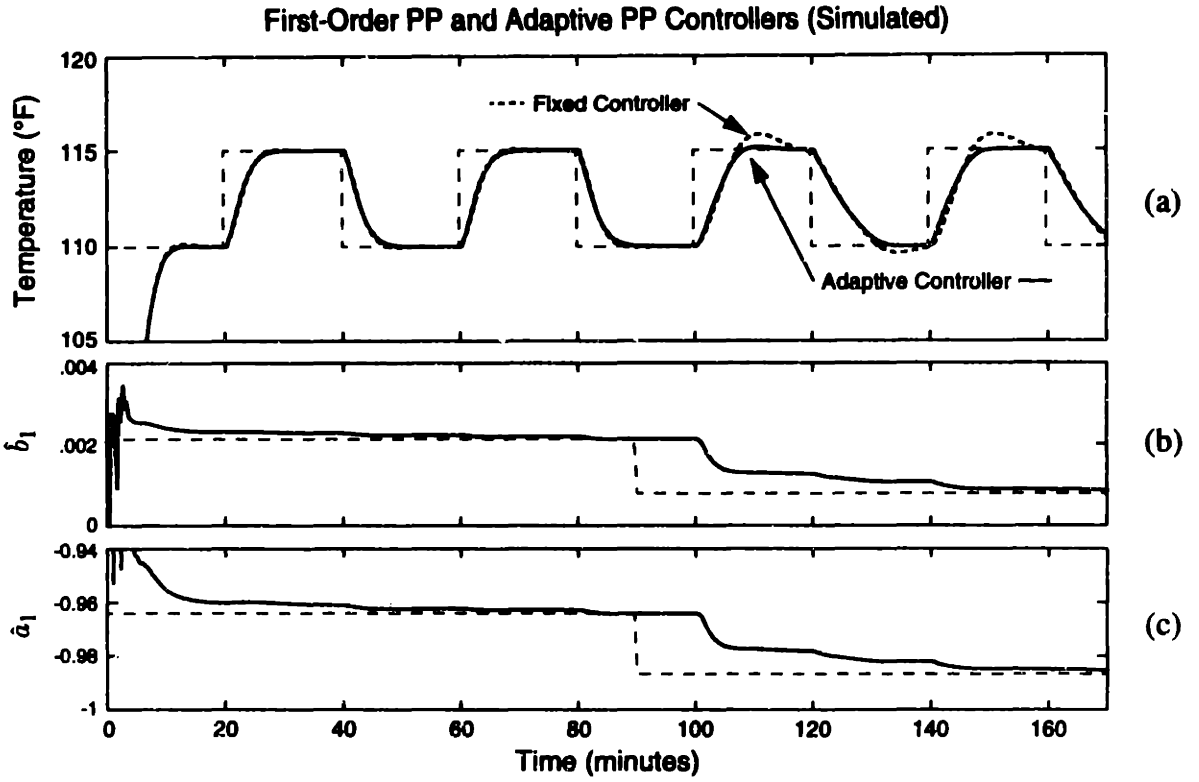



**Figure 6.6:** Simulated temperature responses. (a) Temperature using fixed and adaptive control. (b, c) System parameter estimates.

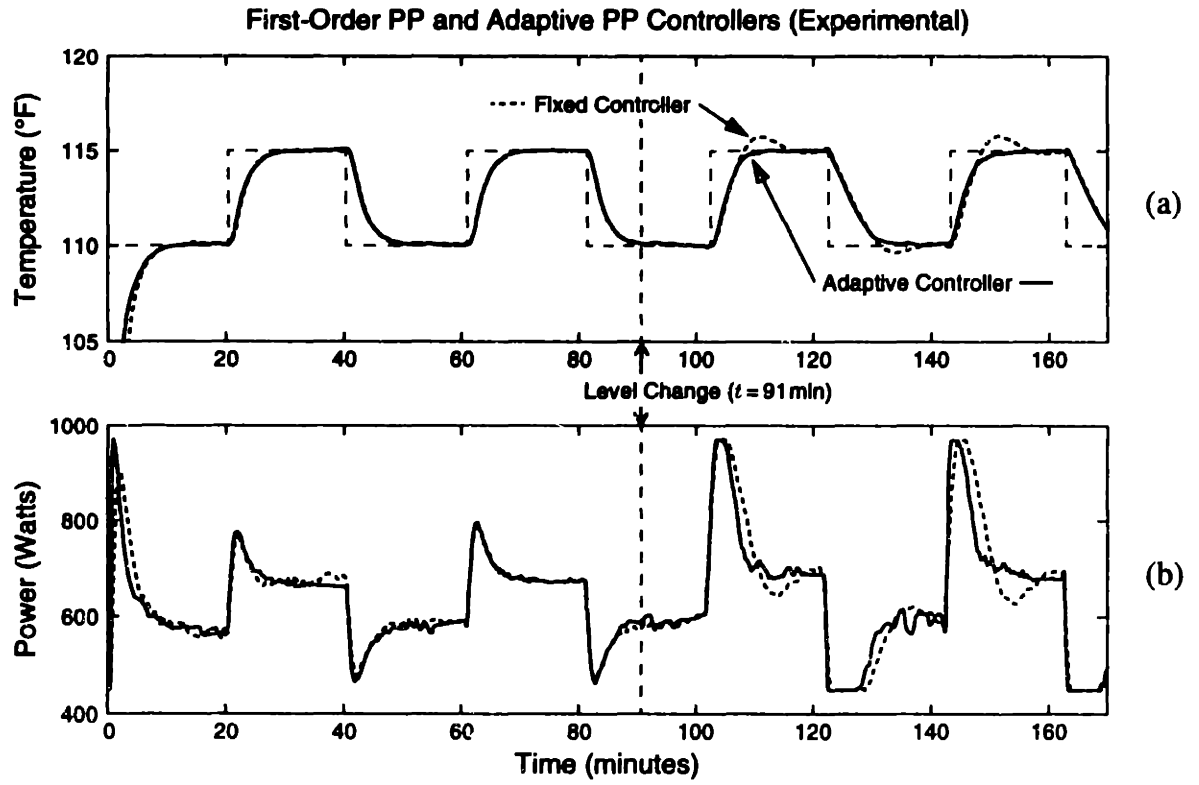



**Figure 6.7:** Experimental temperature responses using fixed and adaptive control.

method of parameter estimation demonstrates considerably improved performance subject to a high noise power-electronic environment.

A motor-speed control system was devised to model an electromechanical servo system. This example approaches the complexity of a number of potential real-world applications. For example, it could model the drive system for an underwater vehicle, where electrical power passes contactlessly through the vehicle's hull to an external drive motor. Other examples might include process control or automation systems, where non-contact inductive coupling is used to transfer power across an environmental boundary.

### 6.4.1 Experimental Setup

Figure 6.8 illustrates the motor-speed control apparatus used for the experiments. Many of the components are unchanged from the earlier water-bath setup. The 1.5-kW inductively-coupled power electronics and the PC data collection system remain the same. The load has been replaced with an electromechanical combination, consisting of a DC motor, a variable-inertia load and a tachometer circuit. The motor was affixed securely to a support beam and used to spin a variable number of circular weights. By dropping additional weights onto the rotating shaft, the mass and hence the inertia could be changed

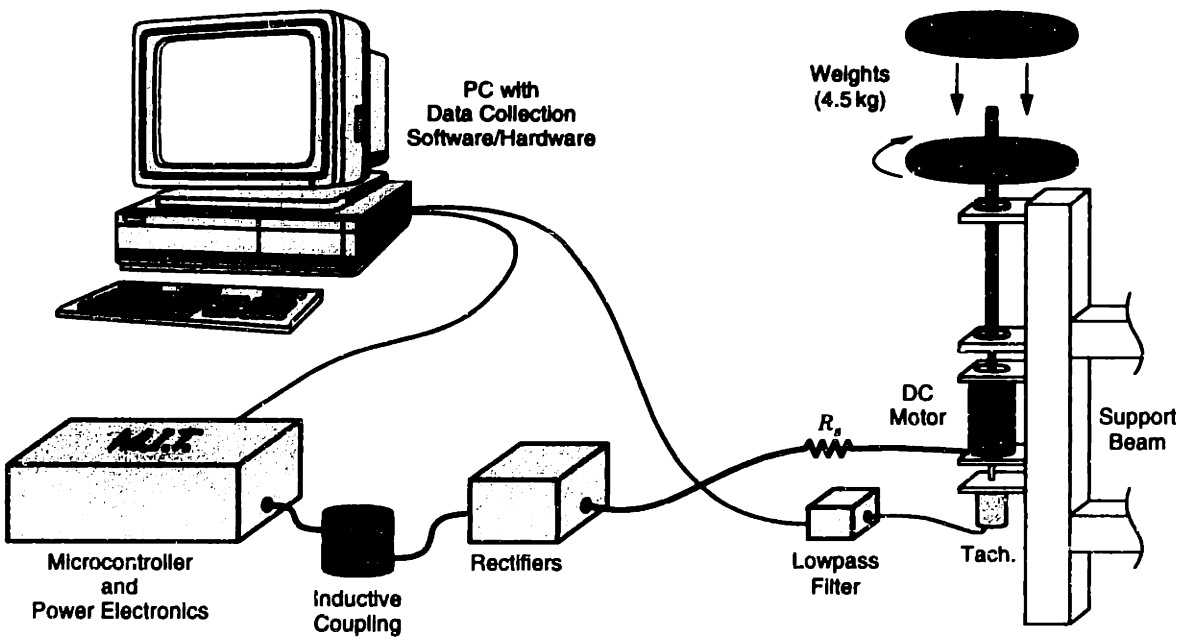



**Figure 6.8:** Setup for the motor-speed control experiments.

abruptly. Mass could be added or removed in increments of 4.5 kg (10 lb). The shaft speed of the rotating system was sensed using a small DC motor as a tachometer. A low-pass filter circuit was used to remove brush and harmonic noise from the tachometer voltage.

### 6.4.2 Control Design

Figure 6.9 diagrams the multi-loop adaptive speed control system that was developed. The structure is analogous to the general multirate approach applied to the temperature control system in the previous section. The "power electronics" block represents the boost converter with its inner voltage-loop controller, and the dashed box represents the electromechanical load. The remaining blocks in Figure 6.9 represent the adaptive speed controller. Parameter estimation for this controller was performed using the LM-RLS algorithm, as indicated by the "lambda estimator" block.

A driving-point characteristic for the load model was derived with the aid of Figure 6.10. As before, the DC/DC stage has been reduced to an ideal transformer and a series resistance $R_d$. The output voltage of the DC/DC stage drives a permanent magnet DC motor. The model for this DC motor includes a winding resistance $R_f$, an inductance $L_f$ and a motor constant $K_m$. The motor's torque acts to rotate an inertia $J$ against a frictional damping $f$. The final speed of this assembly is sensed by a small tachometer mounted to the opposite end of the motor shaft. The output voltage of the tachometer is low-pass filtered to smooth out brush noise. The transfer function of this LPF is

$$\frac{\Omega_f(s)}{\Omega_m(s)} = \frac{1}{\tau_f s + 1} \tag{6.22}$$

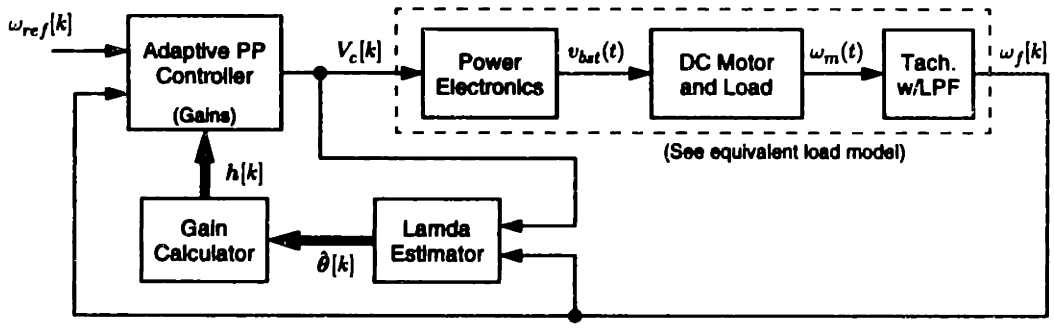



**Figure 6.9:** Block diagram of the motor-speed control system.

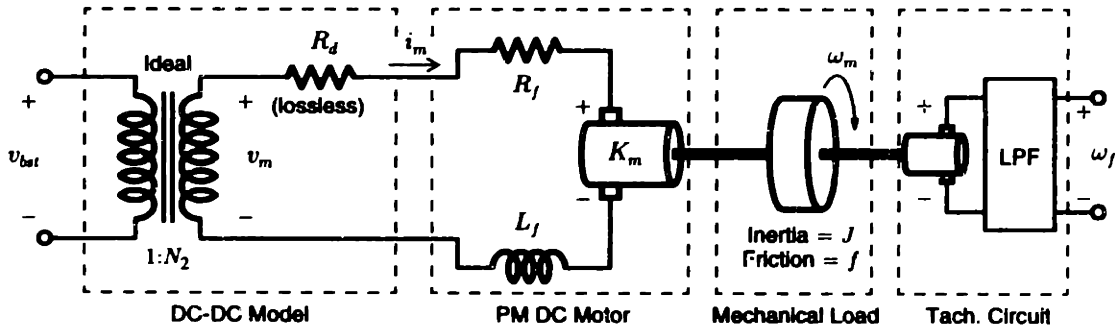



**Figure 6.10:** Block diagram of the equivalent electromechanical load.

where the time constant $\tau_f$ is set at approximately 10 s, and $\Omega_f(s)$ and $\Omega_m(s)$ are the transforms of $\omega_f(t)$ and $\omega_m(t)$, respectively.

From Figure 6.10, the motor torque can be written as

$$T_m(t) = K_m i_m(t) = K_m\left(\frac{N_2 v_{bst}(t) - K_m\omega_m(t)}{R_d + R_f}\right). \tag{6.23}$$

The winding inductance $L_f$ has been dropped from (6.23) because the resulting time constant, $L_f/(R_d + R_f)$, is considerably shorter than the sampling interval of the DT speed controller. The load torque is determined by the inertia, the frictional damping, and the shaft speed, as follows:

$$T_L(t) = J\frac{d\omega_m(t)}{dt} + f\omega_m(t)\,. \tag{6.24}$$

Combining (6.22) and the Laplace transforms of (6.23) and (6.24) yields a single CT transfer function relating the transforms of $v_{bst}(t)$ and the tachometer speed voltage $\omega_f(t)$:

$$H(s) = \frac{\omega_f(s)}{V_{bst}(s)} = \frac{g_m}{(\tau_m s + 1)(\tau_f s + 1)} \tag{6.25}$$

where

$$\tau_m = \frac{J}{f + K_m^2/(R_d + R_f)} \qquad \text{and} \qquad g_m = \frac{N_2 K_m}{f(R_d + R_f) + K_m^2}\,. \tag{6.26}$$

Next, applying a CT to DT transformation to $H(s)$ yields the DT transfer function

$$\tilde{H}(z) = \frac{a_1 z}{z^2 + b_1 z + b_2} \tag{6.27}$$

where

$$a_1 = \frac{g_m}{\tau_m - \tau_f}\left(e^{-\frac{T_s}{\tau_m}} - e^{-\frac{T_s}{\tau_f}}\right), \qquad b_1 = -\left(e^{-\frac{T_s}{\tau_m}} - e^{-\frac{T_s}{\tau_f}}\right), \qquad \text{and} \qquad b_2 = e^{-\left(\frac{1}{\tau_m} + \frac{1}{\tau_f}\right)T_s}. \tag{6.28}$$

The DT load model in (6.27) can now be used to design a DT compensator which yields the desired closed-loop performance. Since $\tilde{H}(z)$ is second-order, a second-order PP compensator was selected. The PP control command is

$$\begin{aligned} V_c[k] = d_1 V_c[k-1] + d_2 V_c[k-2] + h_1(\omega_{ref}[k] - \omega_f[k]) + \ldots \\ h_2(\omega_{ref}[k] - \omega_f[k-1]) + h_3(\omega_{ref}[k] - \omega_f[k-2]) \end{aligned} \tag{6.29}$$

where $w_{ref}[k]$ is the motor-speed reference command and $d_1$, $d_2$, $h_1$, $h_2$, and $h_3$ are constant gains. A block diagram of this command structure appears in Figure 6.11. The saturation block illustrated in Figure 6.11 is added in practice to limit that maximum and minimum voltage command.

Combining (6.29) and the load model in (6.27) results in a closed-loop transfer function of the form

$$\tilde{H}_{CL}(z) = \frac{(1 + x_1 + x_2 + x_3 + x_4)z^3}{z^4 + x_1 z^3 + x_2 z^2 + x_3 z + x_4} \tag{6.30}$$

when the control gains in (6.29) are assigned as follows:

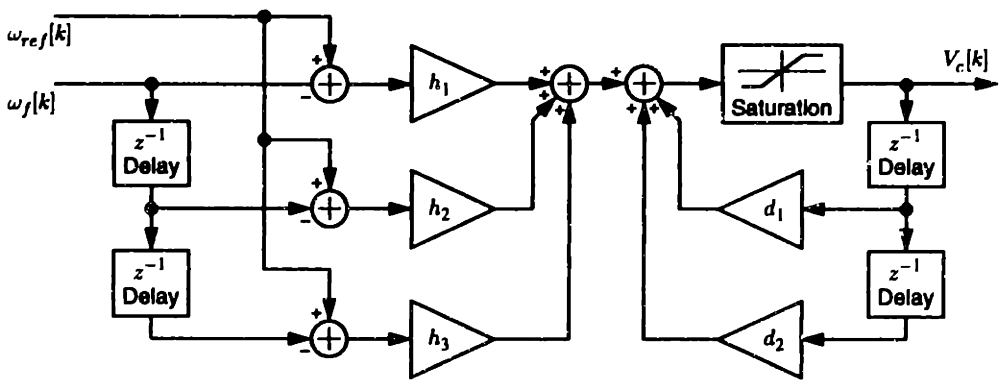



**Figure 6.11:** Block diagram of the second-order PP controller.

$$
\begin{aligned}
d_1 &= x_4/b_2 \\
d_2 &= d_1 - 1 \\
h_1 &= (x_1 + d_1 - b_1 + 1)/a_1 \\
h_2 &= (x_2 + d_1(b_1 - 1) + b_1 - b_2)/a_1 \\
h_3 &= (x_3 + d_1(b_2 - b_1) + b_2)/a_1.
\end{aligned}
\tag{6.31}
$$

Since (6.31) allows the coefficients of (6.30) to be assigned arbitrarily, the closed-loop poles may be located freely on the $z$-plane. A logical choice for a stable, well-damped response is to place the four closed loop poles at identical locations on the real axis, i.e., $z_{1,2,3,4} = p$.

### 6.4.3 Adaptive Updating

Adaptive updating of the speed controller gains was accomplished using an LM-RLS parameter estimation scheme. A block diagram of this estimator appears in Figure 6.12. The LM-RLS estimator contains as its center a RLS estimator. However, it is the coefficients of a *transformed* system, and not the actual system, that are estimated. As described in Section 6.2, a low-pass model transformation is applied to the CT load model. The coefficients of this new model are then estimated. As a result the regressors used for the RLS estimation must be filtered observations of the input and output. For this example, the regressor $\boldsymbol{r}_\lambda[k]$ and the estimate vector $\hat{\boldsymbol{\theta}}_\lambda[k]$ are

$$
\boldsymbol{r}_\lambda[k] = \begin{bmatrix} -(\lambda\omega_f)[k] \\ -(\lambda^2\omega_f)[k] \\ (\lambda^2 V_c)[k] \end{bmatrix} \qquad \text{and} \qquad \hat{\boldsymbol{\theta}}_\lambda[k] = \begin{bmatrix} \hat{\beta}_1 \\ \hat{\beta}_2 \\ \hat{\alpha}_2 \end{bmatrix}. \tag{6.32}
$$

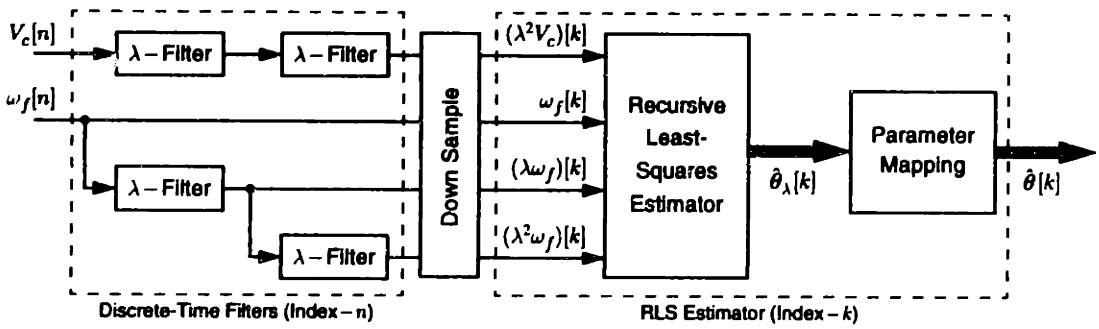



**Figure 6.12:** Block diagram of the Lambda Estimator.

In (6.32) $\lambda$ indicates that a particular observation is pre-filtered by the $\lambda$ operator. Recall from (6.7) that the $\lambda$ operator is a first-order low-pass filter with a time constant $\tau_\lambda$ of 10 s. Ordinarily, such a pre-filtering operation would require that the signals are passed through an analog low-pass filter prior to being sampled. This approach is undesirable for two reasons. First, it increases the circuitry complexity. Although the filters themselves are straightforward, the number of A/D channels increases because each filtered quantity must be sampled separately. In this case four A/D channels would be required versus just two for conventional RLS. Secondly, external filters would be ineffective in removing noise caused by sampling and quantization.

The multirate nature of this digital implementation provides an elegant alternative. The $\lambda$-filtering operation can be implemented digitally at the rate of the "fast" inner voltage loop. Since the voltage-loop time index $n$ samples at a rate $Q$ times that of the outer loop, the DT filters will appear continuous on the "slow" outer-loop time scale. The transfer function of each digital "$\lambda$-filter" in Figure 6.12 is

$$\tilde{H}_\lambda(z) = \frac{z}{z - e^{-T_L/\tau_\lambda}} \tag{6.33}$$

where $T_L$ is the period between steps of the inner loop index $n$. As shown in the figure, the outputs are all down-sampled to the DT time index $k$ before entering the RLS estimator block.

The output of the RLS block is a vector $\hat{\boldsymbol{\theta}}_\lambda[k]$ of parameter estimates for the transformed load model. Estimates for the coefficients of the original CT load model in (6.25) can be computed according to

$$\hat{g}_m = \frac{\tau_\lambda \hat{\alpha}_2}{\hat{\beta}_1 + \hat{\beta}_2 + 1}, \qquad \text{and} \qquad \hat{\tau}_{m,f} = \frac{\hat{g}_m}{2\hat{\alpha}_2}\left(2 + \hat{\beta}_1 \pm \sqrt{\hat{\beta}_1^2 - 4\hat{\beta}_2}\right) \tag{6.34}$$

The parameter estimates that are necessary to update the speed controller can be obtained from the relationships in (6.28) and (6.31). Their application yields formulas for the quantities in $\hat{\boldsymbol{\theta}}[k]$ and $\boldsymbol{h}[k]$ below:

$$\hat{\boldsymbol{\theta}}[k] = \begin{bmatrix} \hat{a}_1 \\ \hat{b}_1 \\ \hat{b}_2 \end{bmatrix} \qquad \text{and} \qquad \boldsymbol{h}[k] = \begin{bmatrix} \hat{d}_1 \\ \hat{d}_2 \\ \hat{h}_1 \\ \hat{h}_2 \\ \hat{h}_3 \end{bmatrix}. \tag{6.35}$$

The gains in $\boldsymbol{h}[k]$ are used to update the coefficients of the PP control command.

### 6.4.4 Results

The adaptive speed-control system was implemented in software on the 80C196KC microcontroller board. The C-code source listings can be found in Appendix F. As before, both fixed and adaptive controllers were implemented to allow for direct comparison. The fixed controller was optimized for operation with an inertial mass of 4.5 kg. The load model parameters for $H(s)$ in (6.25) were approximated experimentally at this mass setting. The results were

$$\tau_m = 11.7 \text{ s}, \qquad \tau_f = 10.0 \text{ s}, \qquad \text{and} \qquad g_m = 0.011. \tag{6.36}$$

The closed-loop performance of both speed-control loops was targeted to have four real poles at $z_{1,2,3,4} = 0.70$. A sample period of $T_k = 1.0$ s was used, so the inner- and outer-loop indices are related by $n = Qk$, where $Q = 120$. The LM-RLS parameter estimation for the adaptive controller was configured with a forgetting factor $\rho$ of 0.97. Thus, approximately 33 seconds of data is retained in "memory." No initial guess was supplied for $\hat{\boldsymbol{\theta}}_\lambda[k]$. Rather, the covariance matrix $\boldsymbol{P}$ was initialized as 10,000 times an identity matrix, and the estimates were allowed to converge on their own. In order to assure stable control during start-up, the fixed controller was engaged for the first 1.5 minutes. This allowed time for the estimates to converge before adaptive control was engaged.

Simulated and experimental results appear in Figures 6.13 and 6.14. A square wave in speed was commanded in order to demonstrate the control performance. In each case, the system was initiated with the inertial mass set at 4.5 kg. At this level the closed-loop performance of both the adaptive and fixed speed controllers is nearly ideal. The step transients are consistent with the closed-loop pole locations and the tracking performance is

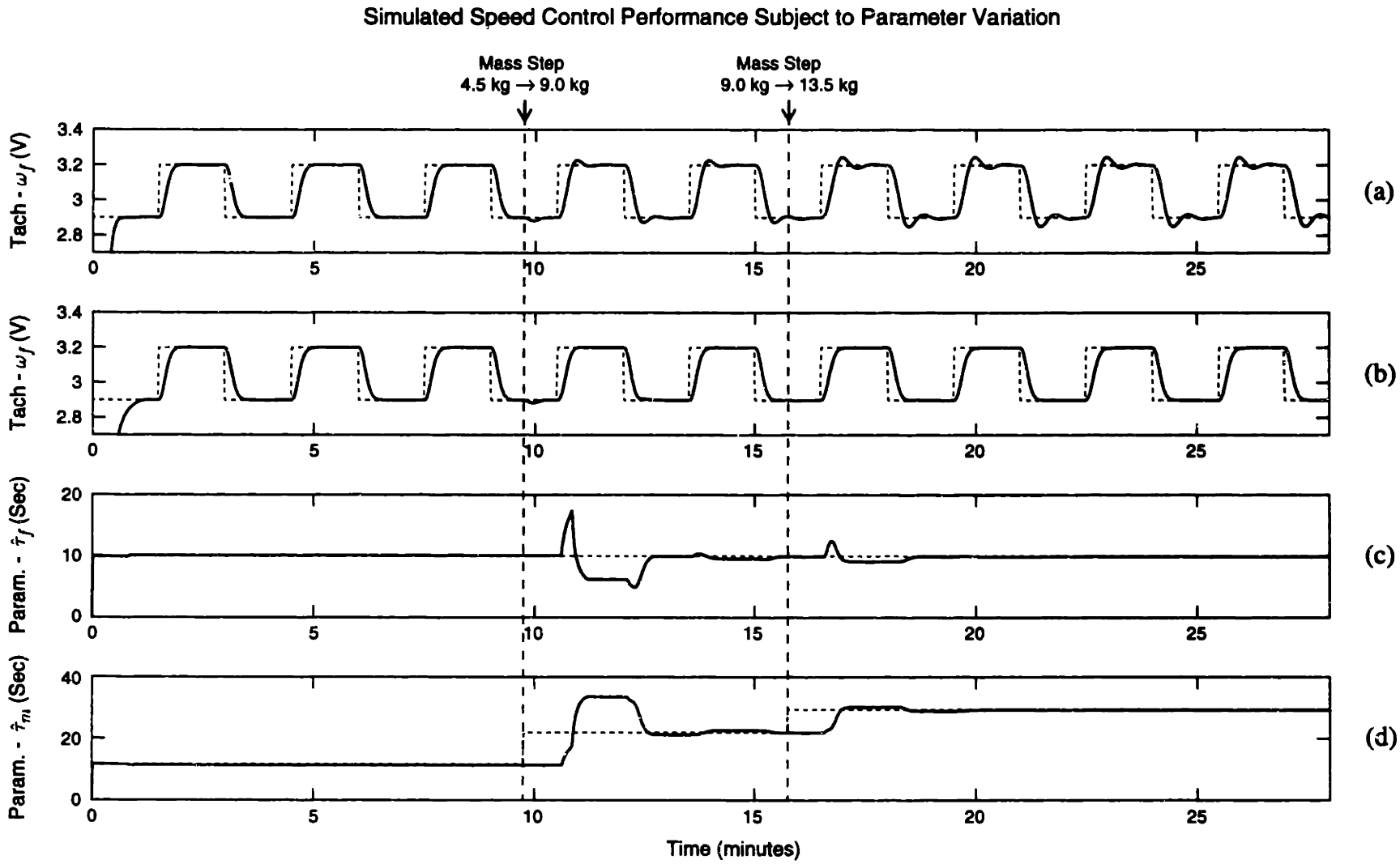



**Figure 6.13:** Simulated motor-speed control responses. (a) Fixed PP controller tuned for a mass of 4.5 kg. (b) Adaptive PP controller. (c, d) Estimated time constants for the adaptive controller..

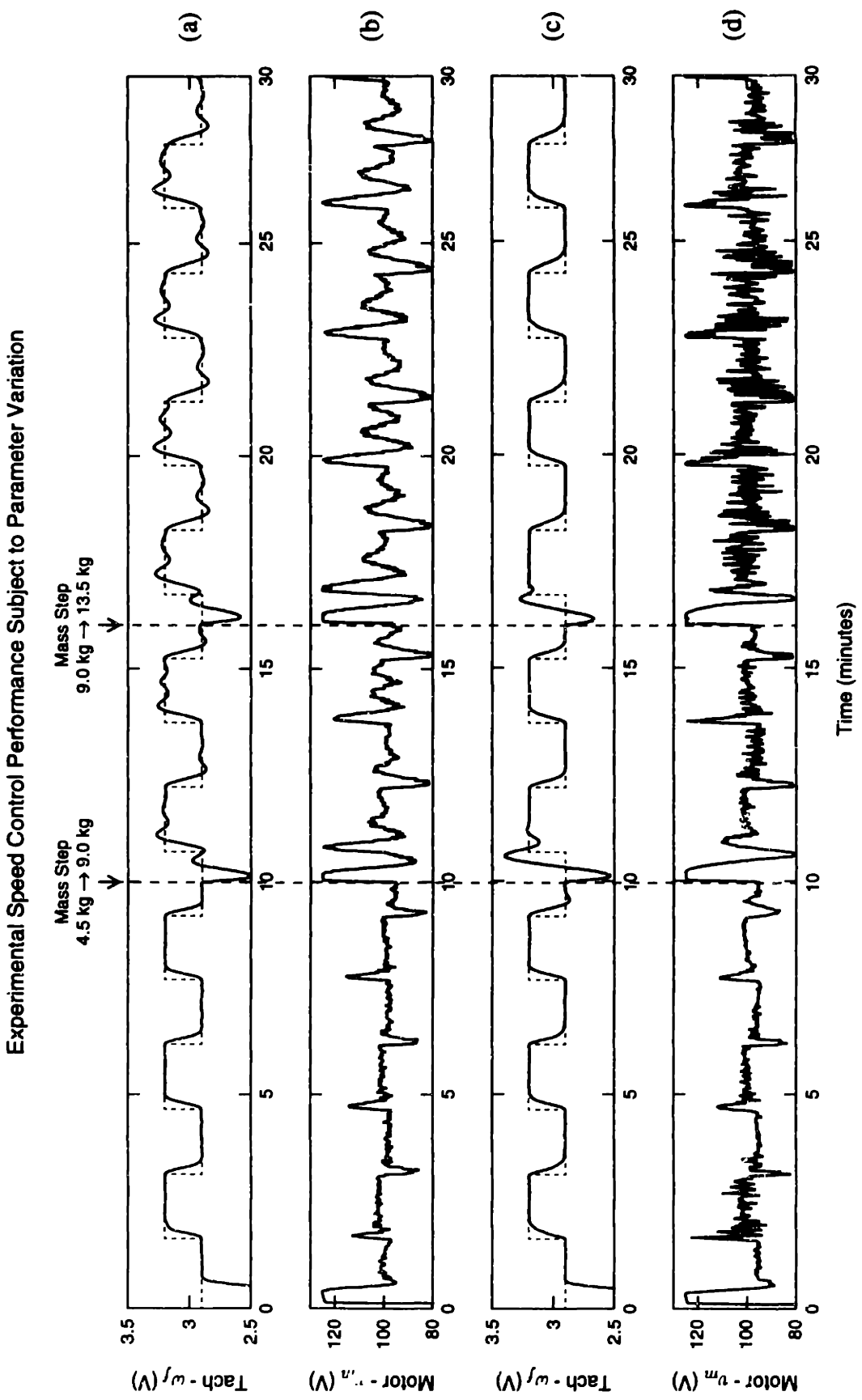



**Figure 6.14:** Experimental motor-speed control responses. (a, b) Fixed PP controller tuned for a mass of 4.5 kg. (a) Tachometer voltage, (b) commanded drive volatage. (c, d) Adaptive PP controller. (c) Techometer voltage, (d) commanded drive voltage.

good. The inertial mass of the system was abruptly increased at 10 minutes and again at 16 minutes into each experiment. Each increase added 4.5 kg to the rotating mass. A momentary loss of speed can be observed at the instants were the mass increased. This effect appears only in the experimental data because the simulation did not take the change in momentum into account.

The experimental results in Figure 6.14 demonstrate that the adaptive controller quickly adapts to the changing inertia of the system, and the tracking performance remains essentially constant throughout. The fixed controller fairs poorly as might be expected. The closed-loop response begins to exhibit a damped oscillatory behavior. The simulated responses in Figure 6.13 show the expected parameter convergence rate. Each step change in the load inertia causes a definite disturbance in all three parameter estimates.[1] The estimates quickly converge on their new values after approximately two cycles of the step input of the system. In practice, the actual convergence time will vary depending on the excitation level of the system, the amplitude of the parameter changes, and the LM-RLS settings $\rho$ and $\tau_\lambda$. Unlike the direct RLS estimator in the previous section, the time constant $\tau_\lambda$ of the "$\lambda$-filter" has an effect on the parameter convergence rate. An optimal setting for $\tau_\lambda$ is equal to or slightly smaller than the fastest pole in the load model.

The performance of the LM-RLS parameter estimation proved to be superior in a high noise environment. The motor-speed control system, for example, was subject to significant electrical noise. Noise sources included brush commutation and switching spikes as well as potential modeling errors and unmodeled dynamics. Prior to the LM-RLS experiments, an adaptive motor-speed controller was implemented using direct RLS estimation. Under nearly identical conditions the RLS estimation proved to be erratic and unreliable. The problem was traced back to the covariance matrix explosion described in subsection 6.2.1. It was hoped that a judicious selection of the dead-band width $\delta$ and the forgetting factor $\rho$ would yield satisfactory results. This was not the case. The LM-RLS method improves the situation dramatically because the regressor vector is composed of filtered measurements. Their noise content is proportionately reduced, and a much narrower dead-band $\delta$ can be tolerated.

---

1. A third parameter, the gain $g_m$, was estimated as well although it does not appear in the figure.

## 6.5 Summary

This chapter formulated adaptive "self-tuning" control techniques, which estimate load-model parameters and automatically adjust a digital controller in response to system changes. Recursively formulated estimation algorithms were presented that are well suited to real-time implementation and demonstrate increased noise immunity, which is essential in a power-electronic environment.

Chapter 6 also demonstrated and extended the general multirate techniques developed in Chapter 5. Two different example servomechanical systems were constructed: a water-bath temperature control system and a motor-speed control system. The examples are illustrative of a number of potential real-world applications, which combine the flexibility of adaptive control and the advantages of non-contact inductive power transfer.

# PART II

# MULTI-STAGE INDUCTIVELY-COUPLED POWER TRANSFER

## Chapter 7

# Overview

Part I of this thesis focuses on single-stage inductively-coupled power transfer. Part II broadens the focus to include multi-stage inductively-coupled power transfer. The creation of an inductively-coupled AC bus, where "pickups" or power taps can be placed at any point along the length of the limb, is explored. Such systems are applicable to, among other possibilities, robotic limbs [22]. Multi-stage power transfer may be ideal for future robotic manipulators which incorporate synthetic (gel) muscles as actuators. Part II uses multi-axis robotic limbs as a context in which multi-stage inductively-coupled power transfer can be studied. The goal is to determine how important design parameters affect the system performance. The results, then, are not limited to the robotic application.

The presentation in Part II addresses several aspects of multi-stage inductively-coupled systems. Mechanical arrangements for inductively-coupled joints are explored. Particular attention is focused on robotic joints with two degrees of freedom, specifically joints similar to human hip or shoulder joints. In addition, two possible multi-stage circuit architectures are studied: voltage-fed power transfer and current-fed power transfer. Due to the breadth of this topic, the analysis is more general than that presented for the single-stage couplings in the first section. The circuit work includes both analytical analysis and computer simulations of model systems. Relevant issues and trade-offs for both architectures are discussed in detail. The results should serve as design guidelines for future experimental work in the area.

## 7.1 Background

### 7.1.1 Multi-Axis Robotic Limbs

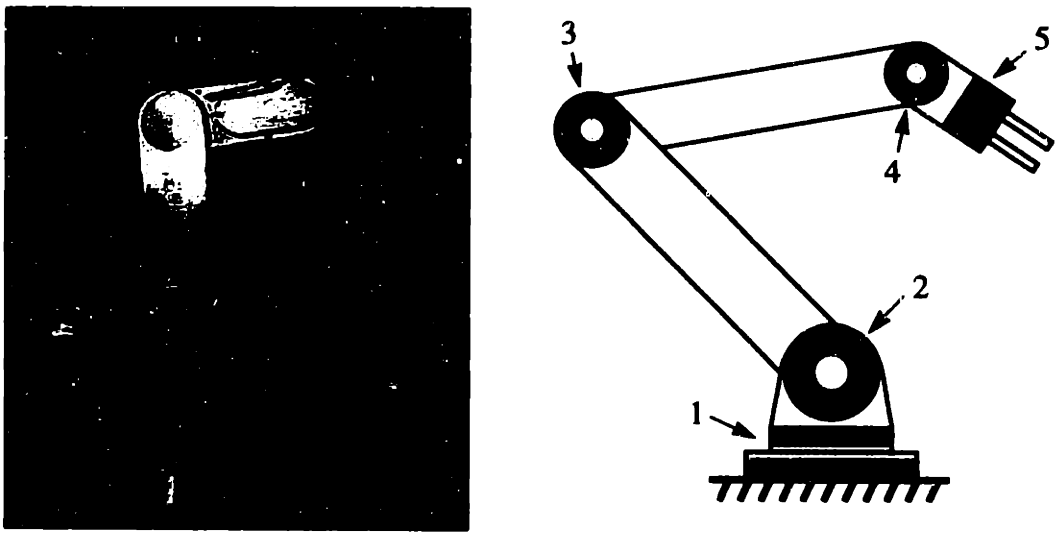



**Figure 7.1:** A multi-axis robotic limb with five rotating joints. The photograph is of a Mitsubishi Melfa ME-RD1 robot [61].

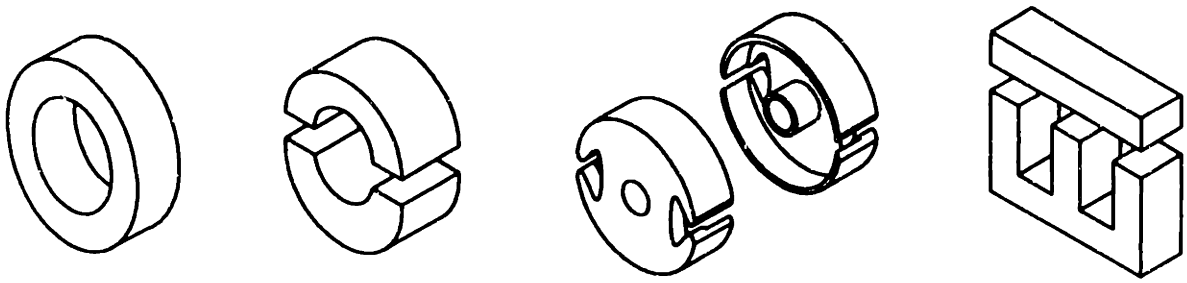
Multi-axis robotic limbs have become a mainstay in American industry. Their application spans a wide range from manufacturing and research to arts and entertainment. The tasks best suited for robotic manipulators are often repetitive and labor intensive. Other tasks may call for high precision or exposure to harsh and otherwise inaccessible environments. Whatever the application, flexibility and durability are primary concerns. A common multi-axis industrial robotic limb is pictured in Figure 7.1. Five rotating joints provide the manipulator with an almost unlimited range of motion throughout its reach. As the photograph clearly illustrates, two flexible coiled-wire assemblies provide electrical connections across rotating joints, and a large number of flexible-wire connections, not visible in the photograph, are certain to exist within the limb. This wiring is responsible for the transmission of power to the various actuators within the arm as well as communication or feedback from those actuators to their respective controllers.

Flexible-wire connections introduce several problems. First, the repetitive motions typical of robotic applications cause excessive wear and stress upon flexible connections, and this wear and stress will eventually lead to failure. Second, the flexibility or range of

motion for any particular joint or coupling will be compromised. A rotating joint at the base of a manipulator or in the wrist, for example, is typically constrained to revolve through an angle less than 360 degrees. This limitation is necessary to prevent tangling or accidental damage. Eliminating the flexible connections would remove such constraints and allow motions currently impossible. For instance, the wrist joint on the manipulator might be made to rotate through many revolutions and thereby allow a continuous twisting motion of the wrist, which could be used to unscrew nuts and bolts in a single motion.

Inductive coupling opens the door for a variety of rotating, sliding, and/or separable links. Advances in switching circuits during the last decade have reduced the size of magnetic components, and modern ferrite materials allow for a wide variety of magnetic core shapes. Thus, it is now possible to incorporate an inductive coupling into the joint of a robotic limb and still maintain the mechanical rigidity of the joint. Ideally, it is possible to transmit power as well as all necessary communications signals through a single inductive coupling at each joint. However, a number of complications, some of which will be discussed shortly, arise in a practical implementation.

Because robotic systems are typically composed of several joints, a suitable inductively-coupled architecture is required to couple signals across multiple links. One possible architecture is to create an inductively-coupled AC bus, where "pickups" or power taps can be placed at any point along the length of the system. Although in today's market the advantages of adding such a system to a typical industrial robot might not outweigh the cost, an expandable power distribution system, perhaps incorporating communication as well, might be ideal for future robotic manipulators. Such a design seems especially desirable for limbs incorporating synthetic muscles as actuators, where many independent actuators are distributed along the length of a limb. The development of gel-based actuators, which are similar in many respects to human muscle tissue, is an active area of research [9, 77]. Inductively-coupled polymer-gel actuators, which are discussed in Part III of this thesis, are one example.



a) Toroid b) 'C' core c) Pot core d) 'E' core

**Figure 7.2:** Commercially available transformer core geometries illustrated in [16].

### 7.1.2 Inductively-Coupled Joints

Good magnetic coupling requires that a magnetic core material, such as MnZn ferrite, is used to contain the magnetic flux linking the electrical circuits. Ferrite cores are ceramic structures that are pressed into shape and then fired in a kiln at 2000 °F [66]. They are widely used in switching power circuits operating at frequencies from 10 kHz to 50 MHz. Standard "off-the-shelf" core geometries are relatively inexpensive, and custom cores can be pressed into limitless shapes for a price. Figure 7.2 shows four commercially available ferrite core geometries.

A number of criteria need to be considered in order to create a well-designed inductively-coupled joint. The following are of particular importance:

- Mechanical stiffness,
- Smooth motion between any two points,
- Ohmic isolation between the coupled electrical circuits,
- Good magnetic coupling between electrical circuits,
- No flexing of the electrical connections.

A uni-axial joint that meets all of the above criteria may be relatively trivial to design. However, the design of a multi-axis joint that does the same can be quite difficult, especially when factors such as complexity, cost, and manufacturability are considered. Several colleagues have been involved in the design of robotic limbs for the testing of polymer gel muscles [16, 50]. Although there are numerous issues that still need to be

worked out before gel muscles can readily be applied as actuators in robotic limb, several new mechanical joint designs have been considered.

In [50], a deceptively simple mechanical arrangement was developed for a uni-axial revolute joint. An off-the-shelf ferrite "pot core" was used to form the inductive coupling. Later work in [16] extended the same design to a multi-axis joint with two degrees of freedom. It was decided that since gel muscles are being developed to simulate human motion, the joint should be designed to function similarly to the human shoulder or hip joint. This requires a joint with two-degrees of freedom such as those illustrated in Figure 7.3. A simple combination of two uni-axial joints was the first attempt. The resulting joint is illustrated in Figure 7.4. The design places the two pot-core halves interior to the joint, in line with a pair of bearings. The bearings support all the mechanical stress while the surrounding structures hold the pot-core halves in perfect alignment. An accurate airgap is maintained between the two pot-core halves. Wires that form the electrical circuits are confined to respective halves of the pot core, and they are free from any mechanical stress or strain.

One undesirable feature of the joint shown in Figure 7.4 is that the two rotating axes are separated by a small distance. A smoother motion is possible if the two axes intersect each other perpendicularly in the same plane, similar to Figure 7.3 (b). An inductively-coupled version of such a "universal" joint might be constructed as shown in Figure 7.5. A custom, folded toroid core would form the magnetic circuit. Although this arrangement is not separable, the core and the windings do not touch.

A variety of mechanical arrangements are possible. However, the same electrical equivalent-circuit models described in Chapter 2 (subsection 2.4.2) still apply. These models are used extensively throughout the discussions in Chapter 8.

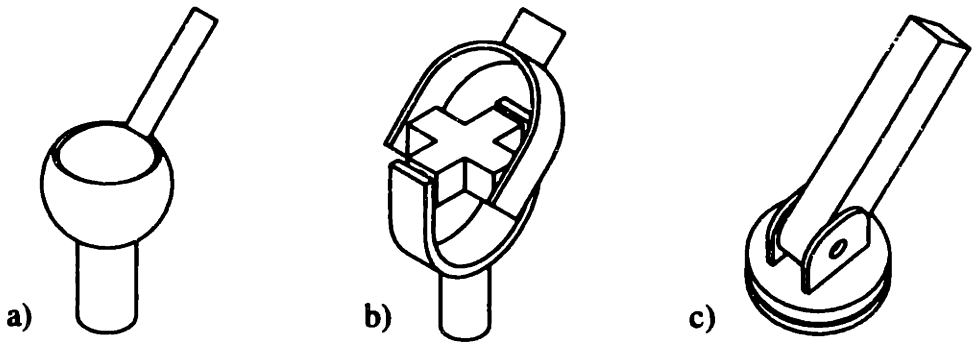



**Figure 7.3:** Common two degree of freedom joints. (Illustration from [16].)

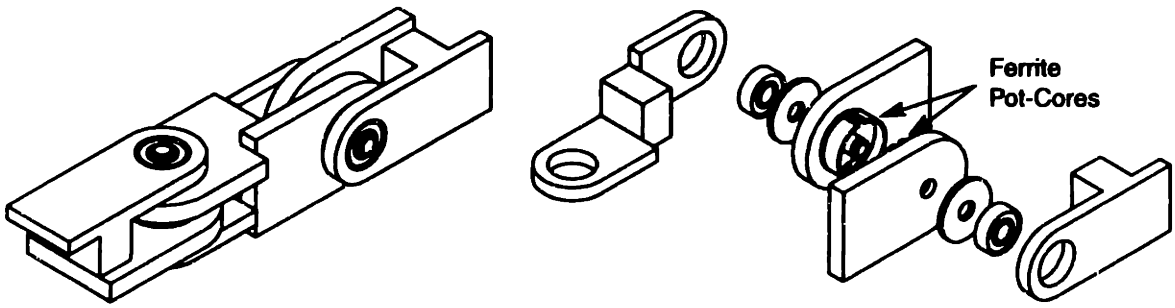
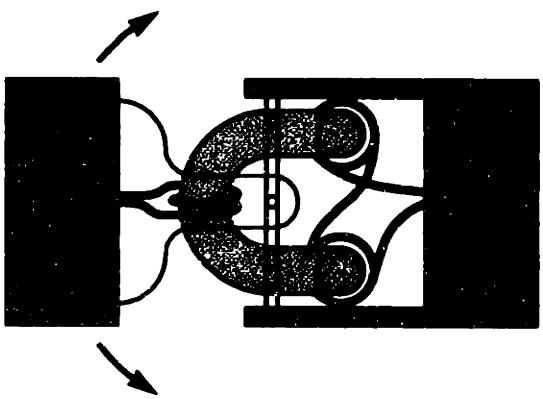
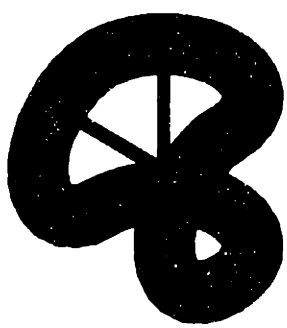



**Figure 7.4:** An inductively-coupled, two-axis joint proposed in [16]. Note that the axes are not in the same plane.



a) Proposed universal joint.



b) Folded toroid core.

**Figure 7.5:** A proposed inductively-coupled, two-axis universal joint. In this case the two axes share the same plane.

# Chapter 8

# Power-Transfer Architectures

Chapter 7 outlined the basic mechanical structure of an inductively-coupled joint. This chapter studies two possible multi-stage architectures for inductively-coupled systems: voltage-fed power transfer and current-fed power transfer.

## 8.1 Voltage-Fed Power Transfer

### 8.1.1 Theory

A voltage-fed inductively-coupled circuit transfers power by applying a constant amplitude AC voltage across the primary terminals of a transformer. This induces an AC voltage across the secondary terminals that, in the ideal case, is exactly equal to the turns ratio times the primary-side voltage. This secondary-side voltage is rectified to produce a "stiff" DC voltage, which can be used to power a servo motor drive, etc. In practice, however, the additional effects of a real transformer (leakage inductances and parasitic resistances) will always result in a secondary voltage that droops as increasing amounts of power are transferred. Quite often, this drooping effect severely limits the ability to transfer power.

For example, Figure 8.1 shows a "T" model of a one-to-one (1:1) transformer with its secondary loaded by a resistance $R_L$ and its primary driven by a sinusoidal voltage source. The resulting LTI transfer function, relating the input and load voltages, is given in the figure. If values for the various circuit elements are known, the load voltage can be readily calculated with a package such as MATLAB. In order to allow for comparisons between a number of simulations, a set of "ball-park" values were selected. These values were arrived at assuming a system designed to transfer $2-3$ kW across a rotating inductively-

coupled joint. A gapped ferrite-core transformer was also assumed. Specific values for the transformer model parameters were decided upon after reviewing the range of values found in the literature. The model parameters are listed in Table 8.1 below.

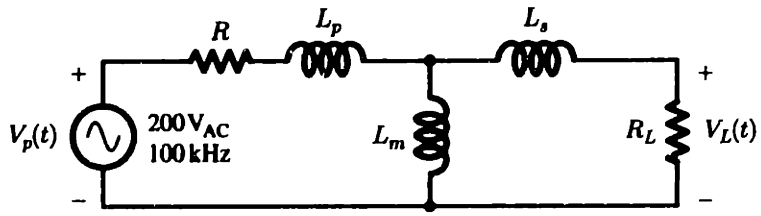
| | |
|---|---|
| Inverter Voltage | $V_p = 200\ V_{AC}$ |
| Switching Frequency | $f_s = 100$ kHz |
| Magnetizing Inductance | $L_m = 200\ \mu H$ |
| Leakage Inductances | $L_p = L_s = 5\ \mu H$ |
| Series Resistance (combined) | $R = 0.2\ \Omega$ |

**Table 8.1:** Model parameters.

With the parameters above, MATLAB was used to evaluate the voltage droop of the circuit in Figure 8.1. The result is plotted in Figure 8.3 (a). The plot shows that the load voltage decreases with decreasing $R_L$. Furthermore, the load power peaks at 1500 W and then "knees over," actually decreasing with smaller $R_L$. This result is essentially the same as that obtained using the commutating reactance model for the DC/DC droop calculation presented in Chapter 3. The voltage droop is caused by a voltage division between the impedances of the leakage inductance, magnetizing inductance, and the load. Luckily, it is possible to counter the effects of the leakage inductances by adding a series-resonant capacitor or capacitors. For example, Figure 8.2 again shows the "T" model of a 1:1 transformer, but this time two series capacitors $C_p$ and $C_s$ are added. If the capacitor values are selected according to (8.1) to resonate at the driving frequency ($f_r = f_s$), then the leakage impedances are effectively cancelled out.

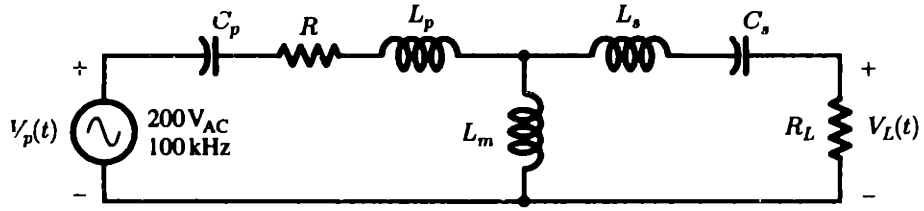
$$C_p = \frac{1}{L_p(2\pi f_r)^2} \qquad C_s = \frac{1}{L_s(2\pi f_r)^2} \quad \rightarrow \quad (C_p = C_s = 0.507\mu F) \tag{8.1}$$

Again, MATLAB was used to calculate the voltage droop, and the result is plotted in Figure 8.3 (b). The effect is quite dramatic. The small remaining voltage droop, which cannot be avoided, is due solely to the series resistance $R$. It should be noted that a similar effect can be achieved with a single series capacitor placed only on the primary side (or only on the secondary side). This solution is arrived at using the equivalent "L" model of




$$H(s) = \frac{V_L(s)}{V_p(s)} = \frac{R_L L_m s}{(L_m L_s + L_p L_s + L_p L_m)s^2 + (RL_m + L_m R_L + RL_s + L_p R_L)s + RR_L}$$

**Figure 8.1:** Voltage-fed transformer without a series capacitance.




$$H(s) = \frac{R_L L_m C s^3}{as^4 + bs^3 + cs^2 + ds + 1}$$

$$a = (L_m L_s + L_p L_s + L_p L_m) C_p C_s \qquad b = (L_m R_L + RL_s + RL_m + L_p R_L) C_p C_s$$

$$c = [(L_m + L_s) C_s + (L_m + L_p) C_p + RR_L C_p C_s] \qquad d = (R_L C_s + RC_p)$$

**Figure 8.2:** Voltage-fed transformer with split series capacitance.

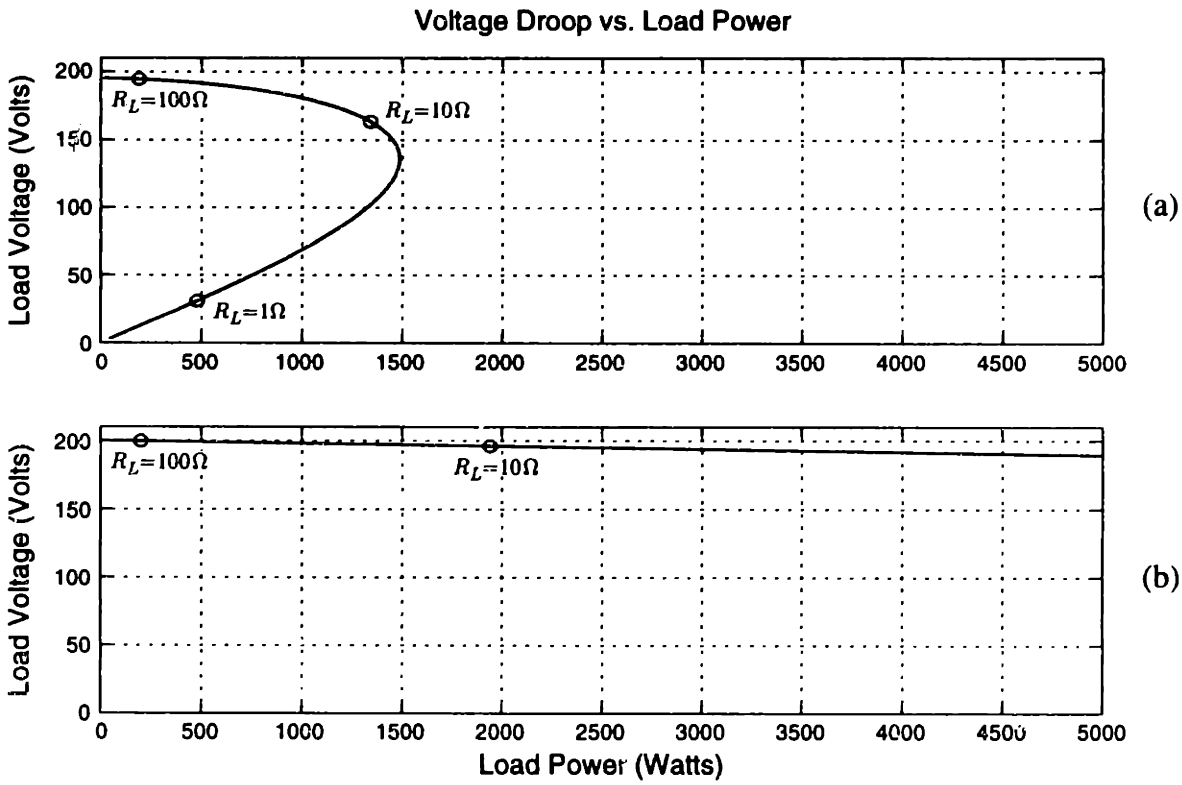



**Figure 8.3:** Load voltage vs. load power, plotted for decreasing load resistance. (a) Figure 8.1 - No series capacitor(s). (b) Figure 8.2 - Split series capacitors.

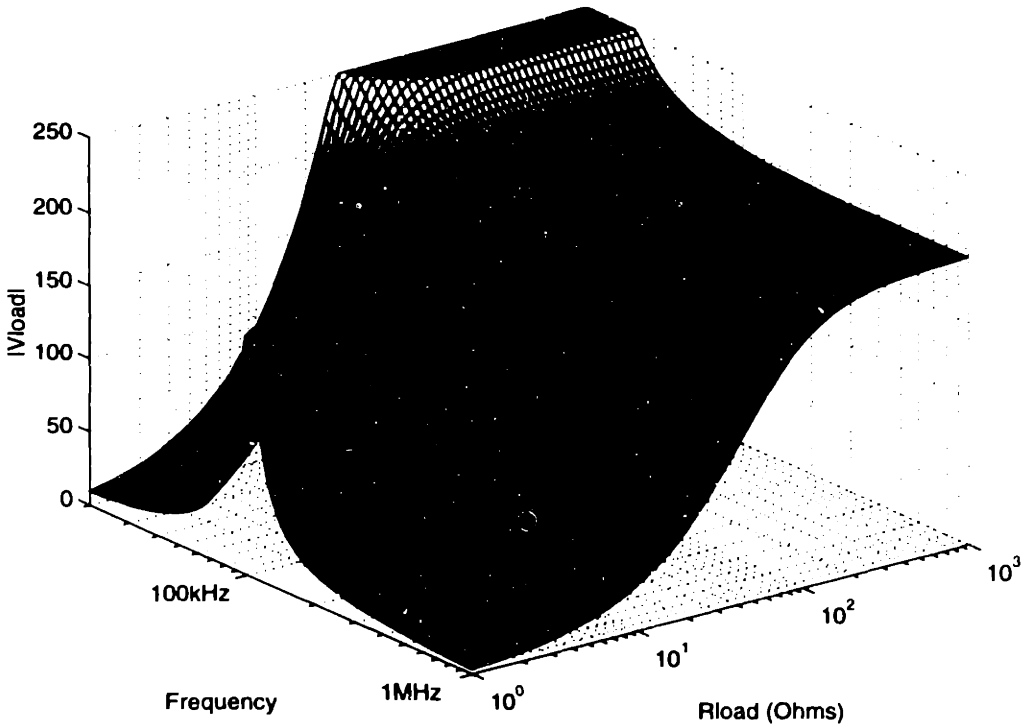



**Figure 8.4:** Load voltage for versus frequency and load resistance. (The voltage axis has been clipped at 250 V.)

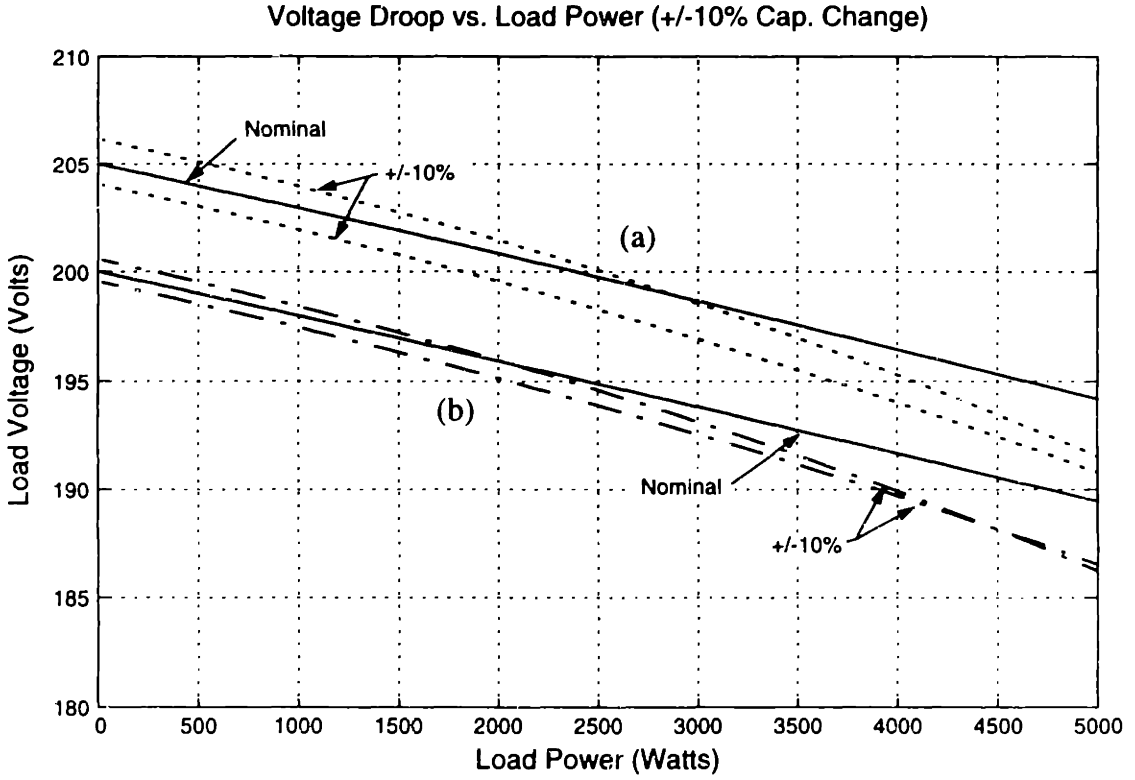



**Figure 8.5:** Load voltage droop provided a +/-10% deviation in series capacitor value(s). (a) Single series capacitor. (b) Split series capacitors.

the transformer. However, the effective turns ratio of the coupling becomes $N_1$ (or $N_2$), instead of $N$ (see Figure 2.8). A second difference will be discussed shortly.

There are some setbacks to this approach. Since complete cancellation of the leakage impedances occurs only at the resonant frequency, in this case $f_r = 100$ kHz, any deviation in the parameter values will yield worse performance. Figures 8.4 and 8.5 help to emphasize this point. Figure 8.4 plots the variation in load voltage as a 3D surface, with load resistance and frequency on the horizontal axes. This plot illustrates that low values of $R_L$ (i.e., high power to the load) increase the resonant $Q$ value, thus narrowing the acceptable tuning range. Nevertheless, Figure 8.5 (b) shows that, for our sample system, a $\pm 10\%$ deviation on both $C_p$ and $C_s$ causes very little detuning over a sweep in load power from $0-5$ kW. The load voltage drops by only 0.6% at 3 kW.

The remaining plot, Figure 8.5 (a), graphs the performance of a similar circuit that employs only a single ($C_{lp} = 0.256$ $\mu$F) capacitor placed in series with the transformer primary. This series capacitance is sized to cancel the "L" model leakage inductance of $L_{lp} = 9.88$ $\mu$H. It is important to note that this time the same $\pm 10\%$ deviation in $C_{lp}$ results in nearly double (0.8%) the load voltage drop at 3 kW. This result is significant because it implies that a split-capacitor system is more robust to parameter variations. A second point to consider is that the model element $L_{lp}$ is a function of the true leakage inductances $L_p$ and $L_s$ as well as the magnetizing inductance $L_m$. Since the value of $L_m$ in a transformer is a sensitive function of the airgap, where $L_p$ and $L_s$ are not, this could pose a tuning problem if variations in the airgap make $L_m$ difficult to control. Both effects suggest that a split-capacitor system is more desirable. The trade-off is that the total capacitance needed for a split-capacitor system is generally four times that of an equivalent single-capacitor system.

### 8.1.2 Practical Considerations

A practical system is built using a DC/DC converter circuit similar to the half-bridge or full-bridge designs in Chapters 3 and 4. These designs, however, were not series-resonant, and this difference requires certain practical considerations. The current through a series-resonant circuit is very nearly sinusoidal because the $RLC$ resonant circuit forms a

high-Q filter, which (for all but very light loads) passes only the fundamental component of the current with any significant magnitude. It is important that this current *lags* the input voltage in order to ensure ZVS and maximize the efficiency of the inverter circuit. Zero-voltage-switching schemes, such as the pseudo-resonant scheme described in Chapters 3 and 4, require that the inverter load is somewhat inductive. A lagging current implies that the load is more inductive than capacitive. Normally, leakage inductances in the transformer ensure this condition, but the series-resonant design effectively cancels the leakage inductance. Fortunately, the magnetizing inductance can be sized so that ZVS is still guaranteed, which is an important practical consideration.

Another important consideration is the effect the load $R_L$ has on the resonant circuit. The LTI analysis in the previous subsection assumed that $R_L$ was resistive. In a practical circuit, the load is composed of a set of rectifiers, a filter, and a DC load (a motor drive, etc.). The non-linear nature of this load can affect the "tuning" of the resonant circuit. A series-resonant circuit is current "stiff"—i.e., the impedance of the resonant circuit is high everywhere except around the resonant frequency. Therefore, the current is sinusoidal, and any rectifier circuit that is attached to the output must be able to support a sinusoidal current. A rectifier followed by an $LC$ (inductor-capacitor) filter is incompatible with this requirement.

Figure 8.6 illustrates the problem. Assume that the filter inductor $L_f$ in Figure 8.6 is large enough so that the inductor current is essentially DC. The current that passes through $L_s$ and $C_s$ prior to the rectifiers must then be a squarewave. But, this is incompatible with the sinewave requirement just mentioned. In practice, the inductance $L_f$ affects the reso-

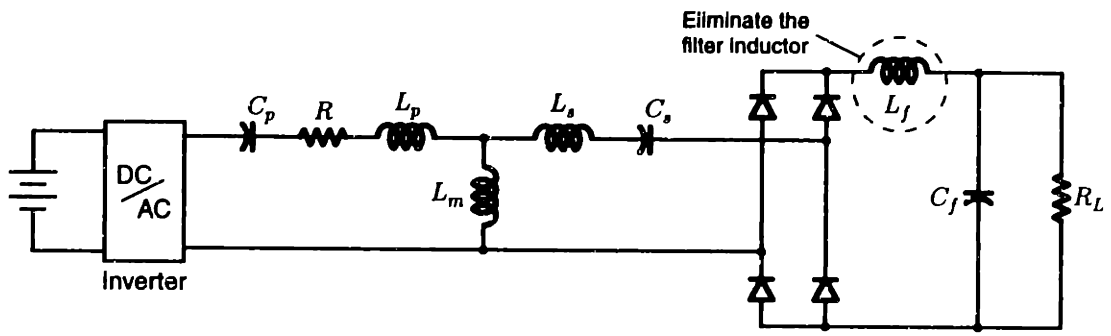



**Figure 8.6:** Rectifier circuit precautions in a series-resonant system.

nant circuit in a non-linear manner, effectively "detuning" the circuit. A capacitive-only filter, however, can support a sinusoidal current without detuning the circuit. Thus, a capacitive-only filter must follow the rectifier in a series-resonant circuit.

### 8.1.3 Multi-Stage Architecture

The topology for a single-stage, voltage-fed system has been described in detail in Part I of this thesis. There are a number of possibilities for extending the topology to a segmented or multi-stage architecture. Work in [22] and [53] suggested the three possible architectures for a segmented system. These architectures are illustrated in Figure 8.7. Notice that all three architectures have in common a series capacitor or capacitors to nullify the leakage inductances at each transformer coupling.

Systems (a) and (b) are referred to as "direct connections" because a single inverter drives a chain of transformer-coupled "joints," which form an AC bus down the length of

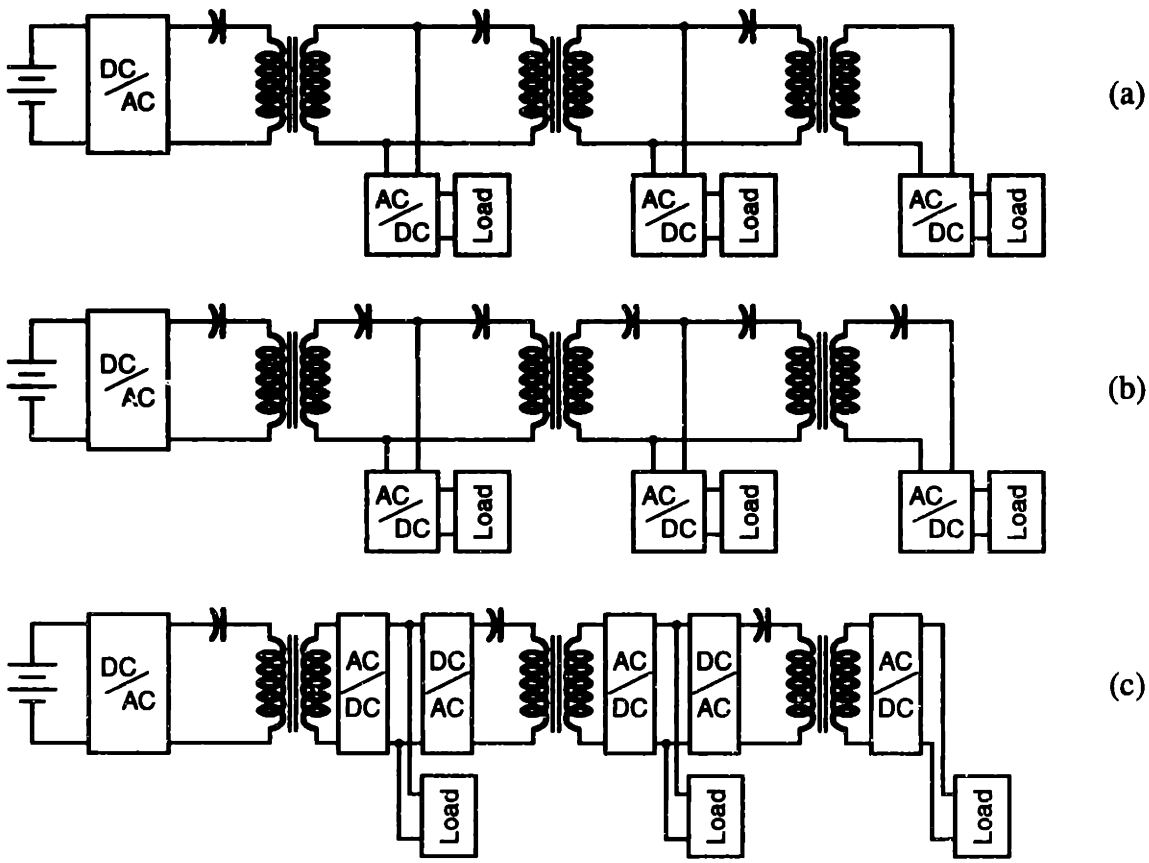



**Figure 8.7:** Three voltage-fed circuit architectures. (a) Direct connection w/series compensation. (b) Direct connection w/split compensation. (c) Indirect connection.

the system. Power is tapped off anywhere along a given segment by rectifying the AC voltage at that point.[1] System (c), on the other hand, is called an "indirect connection," and it is nothing more than a cascade connection of multiple single-stage blocks. Since the entire DC-AC-DC conversion process is repeated across each coupling, an indirect system is considerably more complex. This added complexity offers no clear advantages, and for this reason the indirect method is considered impractical.

It should be noted that all of the architectures illustrated in Figure 8.7 offer the potential for bidirectional power flow. A symmetrical DC/DC converter topology like the half-bridge described in Chapter 3 would make this possible. Bidirectional power flow would make schemes such as regenerative motor breaking possible. This may improve the overall efficiency in specific cases where actuators operate in a generator regime. However, the discussion here will limit itself to unidirectional systems.

### 8.1.4 Simulations

The circuit simulation package PSPICE was used to simulate the operation of a three segment direct connection system. The parameters from Table 8.1 above where used to model the inverter circuit and first-stage coupling. The modeled leakage inductances for the second- and third-stage couplings were increased slightly to reflect the fact that smaller magnetic cores would likely be used in downstream stages. Four different loads, with varying power consumption, were attached to the system. One load was connected along the first segment, two to the second and a final one to the third. A complete schematic of the circuit is shown in Figure 8.8. Simulation results are presented in Figures 8.9 – 8.12. Although not revealed by the figures, an extreme number of simulation steps (approximately 20,000 per trace) are computed during these simulations. Each simulation takes approximately 1/2 hour to complete on a Pentium PC.

Figure 8.9 displays the output of a PSPICE transient analysis. The analysis spans a 4.5-ms period of operation. The load power at each output is doubled and then returned to its original state during the course of the simulation. The intent was to examine the load

1. The specific voltage level at the output of any given power tap may be tailored by adding a low-leakage step-up/down transformer before rectification.

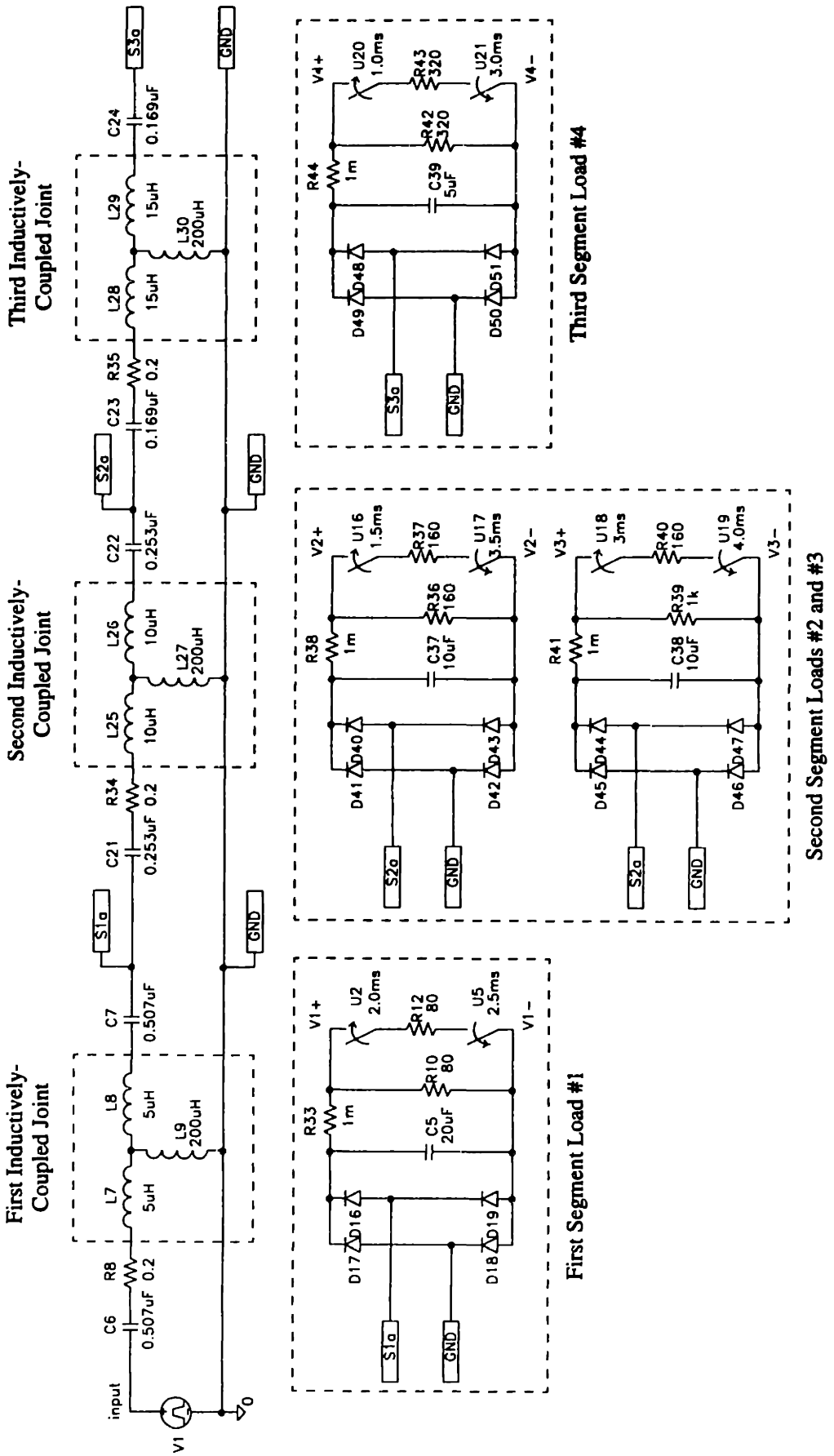



**Figure 8.8:** Schematic of a voltage-fed, inductively-coupled power system. Power is transferred at 100 kHz across three "joints."

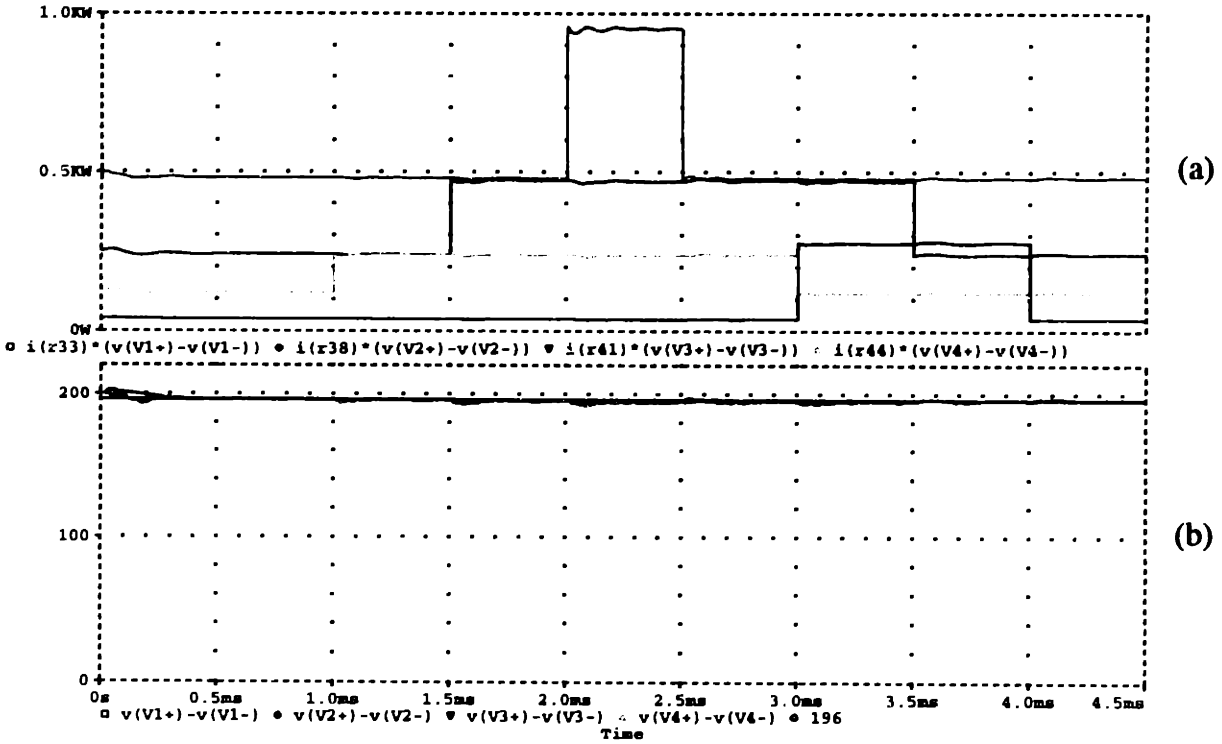



**Figure 8.9:** Simulation results for the schematic in Figure 8.8. (a) Power delivered to the four loads. (b) Output voltages at the four loads.

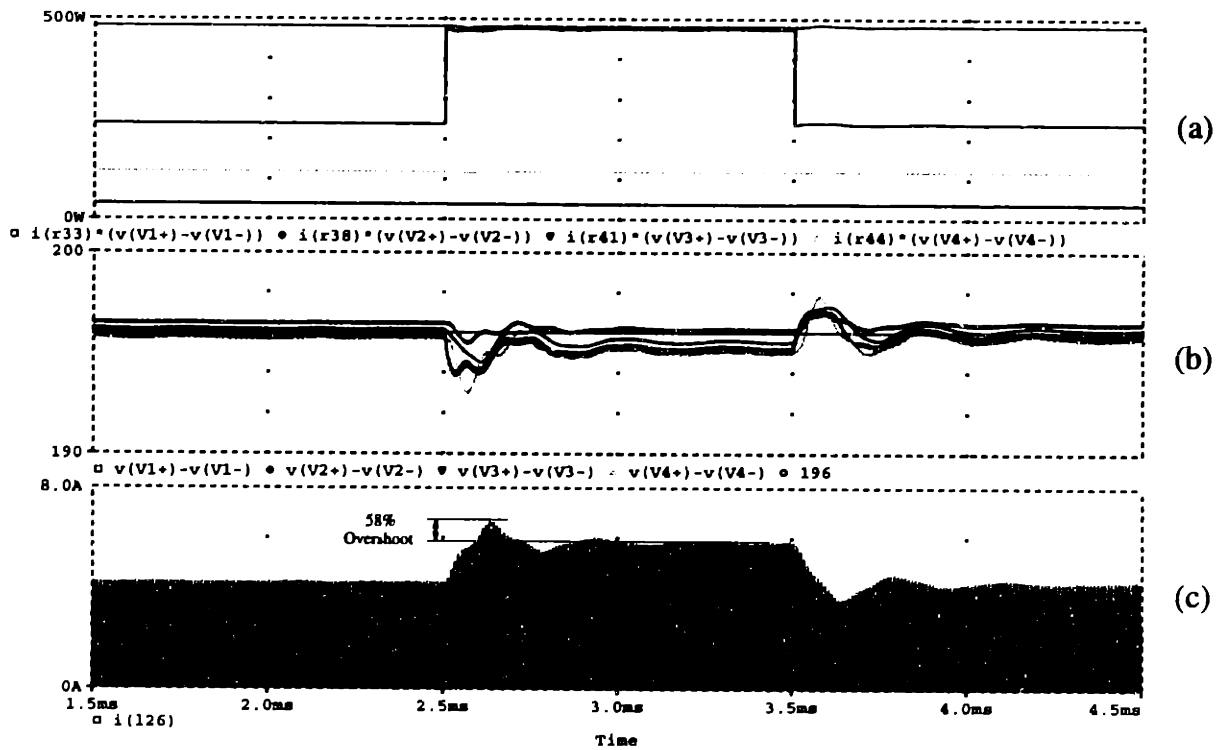



**Figure 8.10:** Simulation results for the schematic in Figure 8.8. (a) Power to the four loads. (b) Output voltages at the four loads. (c) Current through $L_{26}$.

voltage transients at each output in response to changes in load power. The step changes in power can be seen in Figure 8.9 (a). Load #1 is stepped from 480 W to 960 W and back, and Load #2 from 240 W to 480 W and back, etc. Transient variations in the output voltage at each of the four loads are plotted in Figure 8.9 (b). It is clear that each of the four voltages experiences only small perturbations despite a variation from 850 W to 1.7 kW in total system power. A closer examination of the output voltage transients appears in Figure 8.10. During this simulation Loads #1, #3, and #4 were held constant at 480 W, 40 W and 120 W, respectively, while Load #2 was stepped from 240 W to 480 W and back. Again the load powers are plotted in Figure 8.10 (a) and the output voltages in (b). The vertical axis in (b) has been greatly magnified. The simulation reveals peak-to-peak voltage ripples for the four loads of 0.9%, 1.7%, 1.4%, and 2.5%, respectively. Although all four outputs show a relatively small change in voltage, it is important to note that a load change on a *single* output causes voltage transients on *all four* outputs.

It is possible to estimate the transient performance by approximating the transient behavior as a second-order LTI response. Clearly, the true system is significantly higher order and, in fact, non-linear due to the rectified loads. However, the complexity of the true system makes an exact analytical solution intractable. The second-order approximation yields a simple approximate analytical solution, which can serve as a design aid. A second-order damping factor can be approximated for a given load, as follows:

$$\zeta = \frac{1}{2} R_e \sqrt{\frac{C_e}{L_e}}. \tag{8.2}$$

This approximation will be demonstrated for the multi-stage system simulated in Figure 8.10. The simulation depicts a step change in Load #2. We will attempt to predict the transient current overshoot at the output of the second stage. Values for $R_e$, $L_e$, and $C_e$ in (8.2) are taken directly from the simulation schematic in Figure 8.8. The appropriate values for the second stage are given in (8.3).

$$\begin{aligned} R_e &= R_8 + R_{34} + 2R_{\text{diode}} = 0.42\,\Omega \\ L_e &= 2(L_{26} + L_{25} + L_8 + L_7) = 60\mu\text{H} \\ C_e &= C_{37} + C_{38} = 20\mu\text{F}. \end{aligned} \tag{8.3}$$

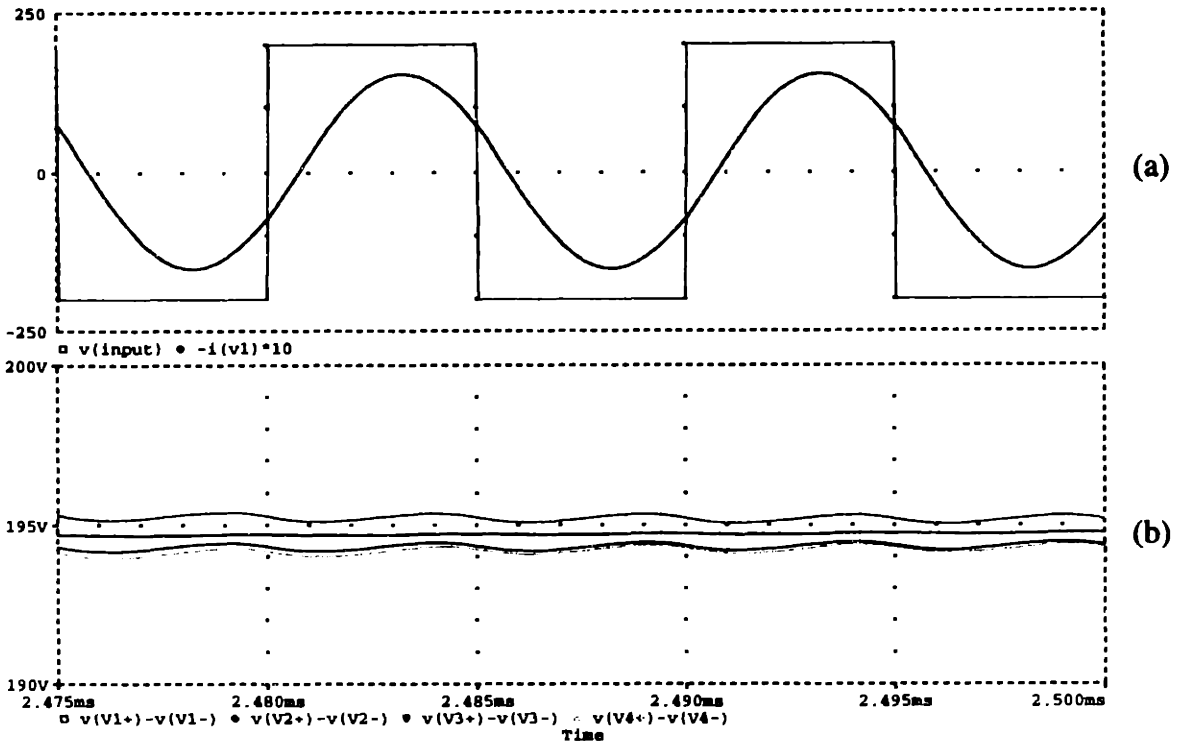



**Figure 8.11:** Simulation results for the schematic in Figure 8.8. (a) Input voltage and current. (b) Output voltages at the four loads.

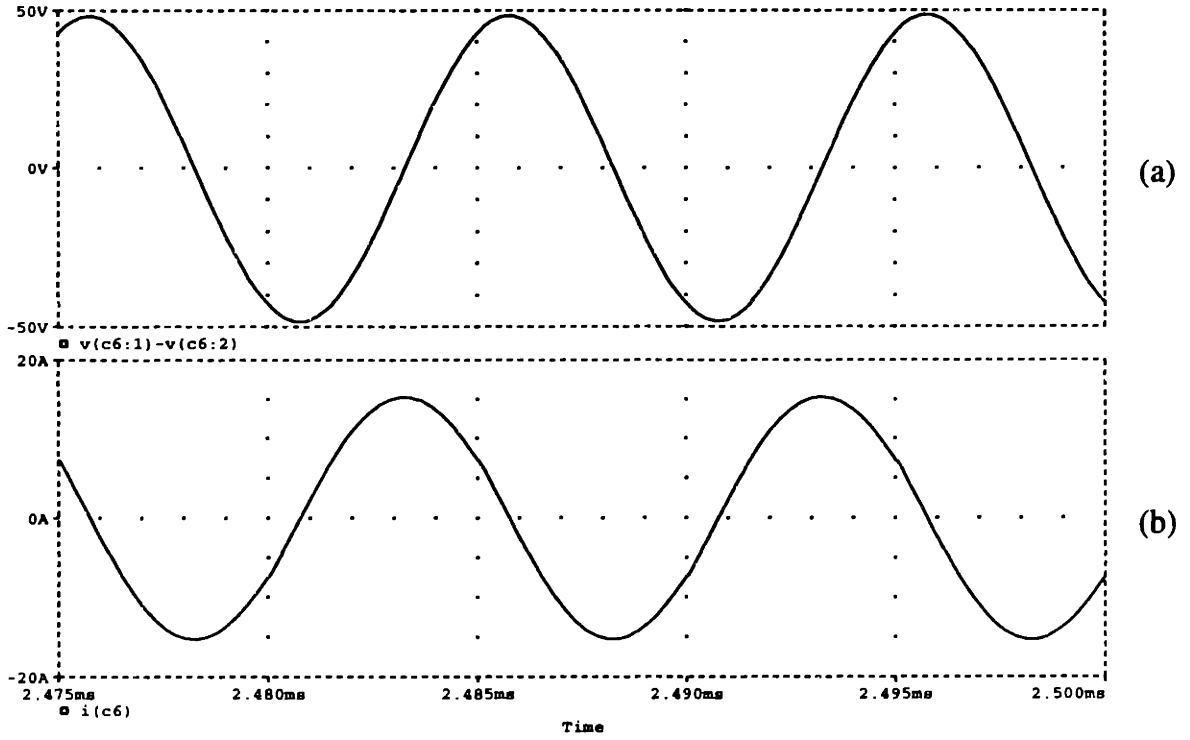



**Figure 8.12:** Simulation results for the schematic in Figure 8.8. (a) Voltage across capacitor $C_6$. (b) Current through capacitor $C_6$.

The equivalent element values $R_e$ and $L_e$ represent the total series resistance and inductance seen looking at the source from the load. The equivalent capacitance $C_e$ is the sum of the filter capacitances at the second-stage output. Plugging the values from (8.3) into (8.2) yields an estimated damping factor $\zeta = 0.121$ for Load #2 in Figure 8.8. The expected peak current overshot, assuming a second-order response is

$$\text{P.O.} = 100e^{\frac{-\zeta\pi}{\sqrt{1-\zeta^2}}} = 68\%. \tag{8.4}$$

The estimated overshoot in (8.4) compares well with the simulated current overshoot of 58% shown in Figure 8.10 (c). Figure 8.10 (c) plots the simulated current in the inductor $L_{26}$ from Figure 8.8. Such a close match is surprising since the current envelope in (c) clearly reveals higher-order behavior. The relative match, however, does suggest that the response behaves dominately as expected. This approximation should work well as a back-of-the-envelope design aid.

The remaining two figures in this section, Figure 8.11 and Figure 8.12, offer expanded views of the operating waveforms in steady state. The drive frequency is 100 kHz. Figure 8.11 (a) shows the simulated inverter output voltage and current. Note that the inverter current lags the voltage. Thus, the sign of the inverter current at the switch transitions is appropriate for ZVS. Trace (b) in the same figure magnifies the steady-state output-voltage ripple on all four loads. Figure 8.12 (a) and (b) shows the voltage and current, respectively, for the first-stage series-resonant capacitor $C_6$. Since $C_6$ carries the highest r.m.s. current in the circuit, it was checked to make sure the amplitude was within reason. The figure reveals that at 1.7 kW of total system power, this capacitor supports a sinusoidal current of about 10 $A_{rms}$ and a sinusoidal voltage of 35 $V_{rms}$. Capacitors with ratings far in excess of this are readily available [41].

### 8.1.5 Voltage-Fed Design Guidelines

The simulations in the proceeding subsection indicate that a multi-stage voltage-fed system is feasible. A typical system that delivers several kilowatts to four independent loads was shown to have good performance. The specifics of this typical system—inverter voltage, operating frequency and transformer parameters—were chosen somewhat arbitrarily. In order to extend this design methodology it is necessary to know how

changes in these design parameters will influence system performance. Table 8.2 provides a set of design guidelines that outline the influence various design changes have on performance. These guidelines allow a designer to pick and size system components in order to meet desired performance criteria.

| Design Parameter (Change) | Effects |
|---|---|
| Series capacitor scheme (Split caps vs. single cap) | **Positive Effects:**<br>• More robust to parameter variation.<br>• No dependence on $L_m$ (i.e., insensitive to airgap changes.)<br>**Negative Effects:**<br>• Increased capacitor volume and cost. |
| Operating frequency (increase) | **Positive Effects:**<br>• Lighter/smaller system components.<br>• Increased power handling for a fixed A·T rating.<br>• Less energy storage and better transient performance.<br>**Negative Effects:**<br>• More core, ohmic, and switching losses.<br>**No Effect:**<br>• Droop characteristics do not change. (If inductances are proportionally reduced, i.e., fixed magnetizing current.) |
| Inverter voltage (increase) | **Positive Effects:**<br>• Lower current = lower ohmic losses.<br>• Lower radiated EMI.<br>**Negative Effects:**<br>• Increased insulation and safety requirements.<br>**No Effect:**<br>• Droop characteristics do not change. (If inductances are increased by the square of the voltage, i.e., fixed A·T.) |
| Leakage inductances (decrease) | **Positive Effects:**<br>• Droop characteristics are less stringent.<br>• Less resonant energy and better transient performance.<br>**Negative Effects:**<br>• Achieved through expensive windings, more core volume, or a smaller airgap. |
| Magnetizing inductance (increase) | **Positive Effects:**<br>• Lower ohmic losses.<br>• Minimal improvement in voltage droop.<br>**Negative Effects:**<br>• May decrease the ZVS range of the inverter. |

**Table 8.2:** Voltage-fed design guidelines.

## 8.2 Current-Fed Power Transfer

### 8.2.1 Theory

A current-fed circuit establishes a constant amplitude AC current through the primary terminals of a transformer. If a load resistance $R_L$ is connected across the secondary terminals, a voltage will develop across it that strongly depends on the value of $R_L$. However, a simple compensation scheme, suggested in [54] and illustrated in Figure 8.13, allows for a nearly constant load voltage despite changes in the value of $R_L$. A single capacitor $C_o$ is added in series with the load $R_L$. The effect of $C_o$ is clear when the circuit is viewed in its Thevenin equivalent form. The reflected capacitance $C_{o*}$ "cancels" the impedance of $L_m$ and $L_s$, leaving only the secondary-side resistance $R_{s*}$ in series with the Thevenin voltage source $V_T$.

It has been suggested that $C_o$ should not be sized to *exactly* cancel $L_m$ and $L_s$ at the drive frequency, "because the secondary winding and the load dissipate identical amounts of power which leads to poor efficiency and overheating of the secondary winding" [54]. However, identical dissipation only occurs if $R_L$ is equal to $R_s$, but these quantities would *never* be equal in a practical system. Instead, $R_L \gg R_s$ in a practical system, so the power dissipated in $R_s$ will be only a tiny fraction of that delivered to the load.

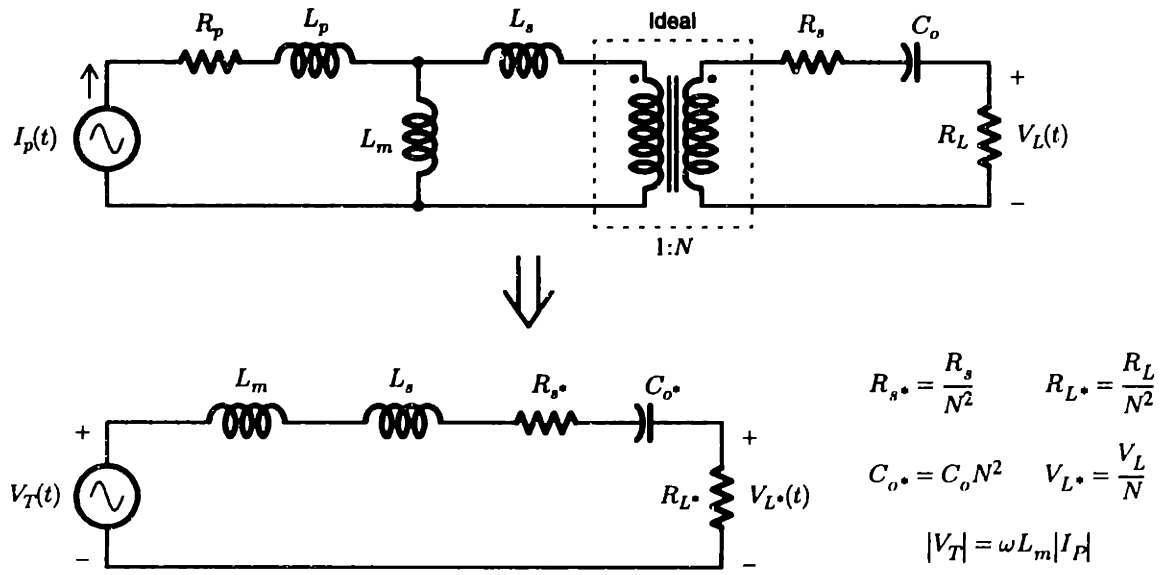



**Figure 8.13:** Thevenin equivalent reduction of a current-fed transformer with added output capacitance [54].

Placing the primaries of multiple transformers in series allows power to be delivered to a number of loads simultaneously. A single-loop primary can be used to source multiple "pickups" [54]. A pickup contains the magnetic core of a transformer and all other secondary-side components. A complete transformer is formed by clamping the pickup around a segment of the primary loop. Multiple pickups on the same loop will appear as though they are in series electrically. The ability to add or remove pickups without ohmic connections is the most significant advantage to the current-fed approach. Recall that ohmic connections to the AC bus are required in order to add/remove loads with a voltage-fed system. A single-loop current-fed system, with three pickups in place, is illustrated in Figure 8.14 (a).

Such a system is applicable to a segmented robotic limb only if it is extended to support multiple loops, which are inductively coupled at each joint in the limb. Figure 8.14 (b) shows a two-loop extension of the single-loop architecture. More than two loops could be linked in a similar fashion. Provided the current-output inverter can tolerate the new load, the addition of a second loop, as shown, has no affect on the operation of the first loop. However, if Loops #1 and #2 are to carry the same current, then the magnetizing current $I_m$, in the figure, must be approximately zero. Since the current $I_m$ is a result of the finite magnetizing inductance of the transformer that couples the two loops at the joint, it

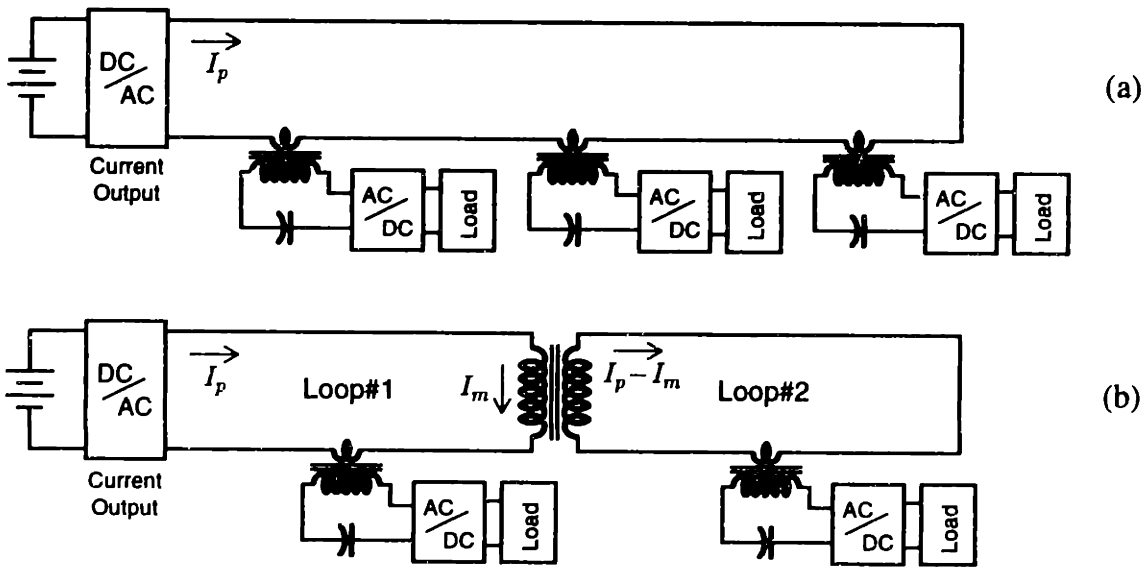



**Figure 8.14:** (a) A single-loop current-fed architecture. (b) An extended architecture with multiple series-connected loops.

can be brought close to zero by making the magnetizing inductance at the joint much larger than the total inductance of Loop #2. This condition becomes increasingly difficult to maintain as the number of linked loops is increased beyond two.

### 8.2.2 Practical Considerations

In a practical system, a current-source inverter is often realized by wrapping a feedback controller around a voltage-source inverter. The controller adjusts the inverter voltage in order to maintain a desired current. The suitability of such an inverter for this application may depend on the transient performance of the feedback controller. Certain inverter topologies, such as the class-E topology in Part III of this thesis, are well-suited for current-source, series-resonant operation.

In addition, the output capacitor $C_o$ in a current-fed system forms a series-resonant tank just as $C_p$ and $C_s$ did in the voltage-fed system. Therefore, the same caution applies when $R_L$ in Figure 8.13 is replaced by rectifiers, a filter, and a DC load. A *capacitive-only* filter must be used, as an $LC$ filter will detune the circuit. An explanation of the problem is provided in Section 8.1.2.

### 8.2.3 Simulations

The circuit simulation package PSPICE was used to simulate a two-loop current-fed system as shown in Figure 8.14 (b). As in the previous simulations, certain assumptions were made in order to obtain specific component values for use in the simulation. In particular, the model parameters for a pickup transformer were scaled from experimental data presented in [54]. The inverter current feeding our sample system was set at 5 A and 50 A in each of two simulations. The turns ratios of the pickup transformers were adjusted in each case to deliver a constant 125-V DC voltage to the loads. The final component values are listed in Table 8.3, and a complete schematic of the PSPICE circuit appears in Figure 8.15. A single power pickup is placed on each of the system's two loops. A second load resistor on each pickup is switched in parallel midway through the simulation in order to observe the effect of load doubling. Note that the coupling transformer, used to join the two loops, has a large magnetizing inductance, $L_m = 1000\ \mu$H. This ensures that

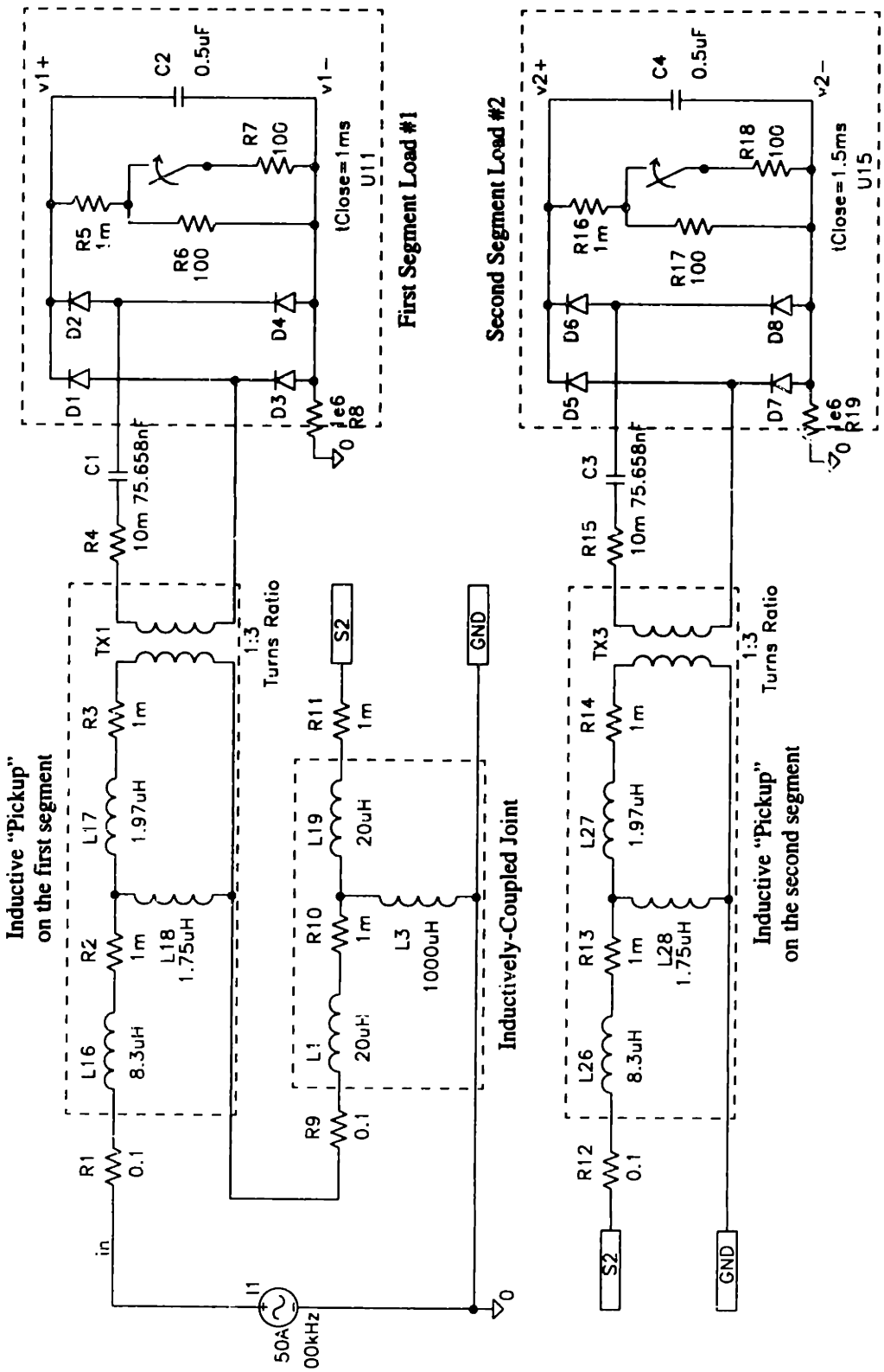



**Figure 8.15:** Schematic of a current-fed, inductively-coupled power system. Power is transferred at 100 kHz down two joined loops. Two "pickups," one for each loop, draw power from the system.

| Inverter | | | |
|---|---|---|---|
| Inverter Current | $I_p = 5$ A & 50 A | | |
| Switching Frequency | $f = 100$ kHz | | |
| **Pickup Parameters** | | **Joint Coupling Parameters** | |
| $L_p = 8.3\ \mu$H | $R_p = 0.1\ \Omega$ | $L_p = 20\ \mu$H | $R_p = 0.1\ \Omega$ |
| $L_s = 1.97\ \mu$H | $R_s = 10\ \text{m}\Omega$ | $L_s = 20\ \mu$H | $R_s = 1\ \text{m}\Omega$ |
| $L_m = 1.75\ \mu$H | | $L_m = 1000\ \mu$H | |

**Table 8.3:** Model parameters

loop currents are essentially equal in both segments. Simulation results are presented in Figures 8.16 and 8.17.

Figure 8.16 displays the output of a PSPICE transient analysis spanning 2 ms. The circuit in this simulation is driven by a 50-A current source, and pickups with a 1:3 turns ratio yield a nominal 125-V output. The power delivered to each load is stepped from 160 W to 320 W and then back to 160 W during the course of the simulation. The intent was to examine the load voltage transients at each output in response to these changes. The step changes in power can be seen in Figure 8.16 (a). The second axis (b) shows inverter voltage and current, and the third axis (c) shows the voltage at each output. A transient voltage oscillation appears at each output during the respective step-changes in load power. Although the oscillation rapidly decays, its amplitude peaks at approximately 14 V or 11%. Such a large perturbation might exceed the specified operating voltage of an attached load. Note, however, that a step change in Load #1 does not result in a voltage transient at Load #2. Unlike a voltage-fed system, transients at the various outputs are largely decoupled. (An exception to this rule may occur if the inverter that feeds the system experiences a transient during load swings.)

A disturbing feature appears in Figure 8.16 (c). The amplitude of the voltage at the output terminals of the inverter peaks near 2 kV. This means that the VA product at the inverter output is an incredible 46 kW, while the real power is only about 650 W. The poor power factor is caused by the large inductive load presented to the inverter output. The VA product can be decreased in a number of ways. Two of the most promising are to reduce

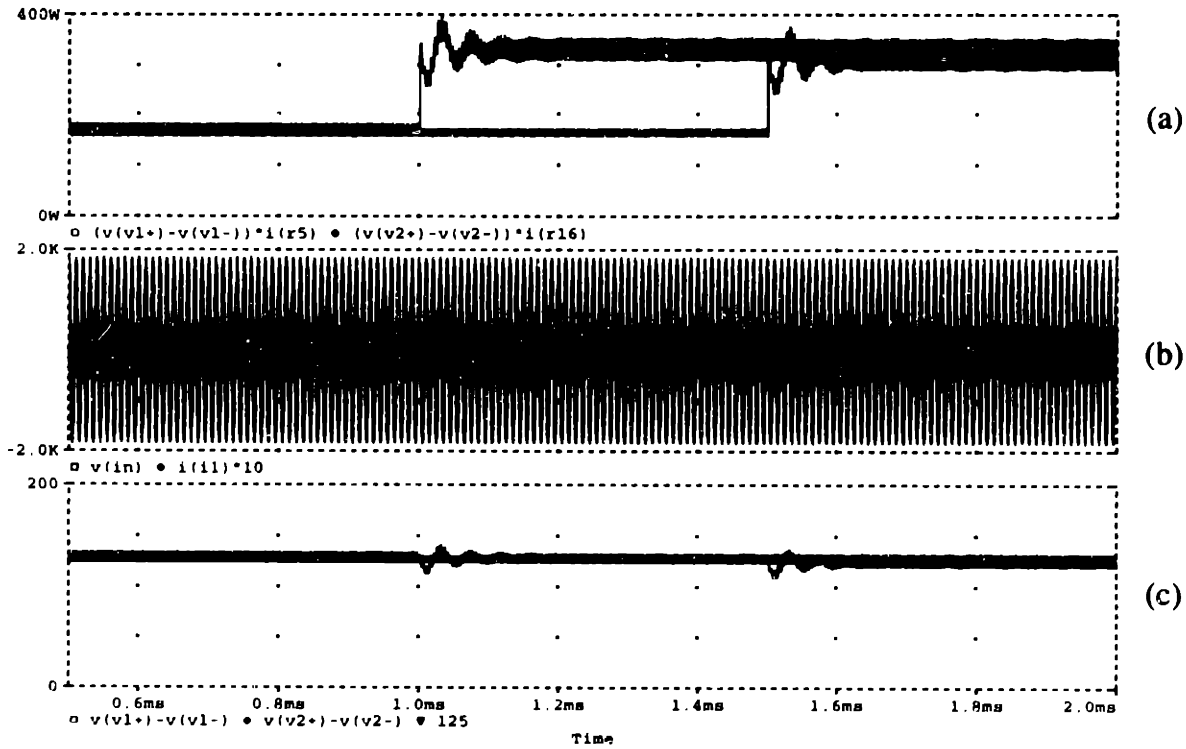



**Figure 8.16:** Simulation results for Figure 8.15 driven by a 50-A current source. (a) Load power. (b) Source voltage and current. (c) Load voltage.

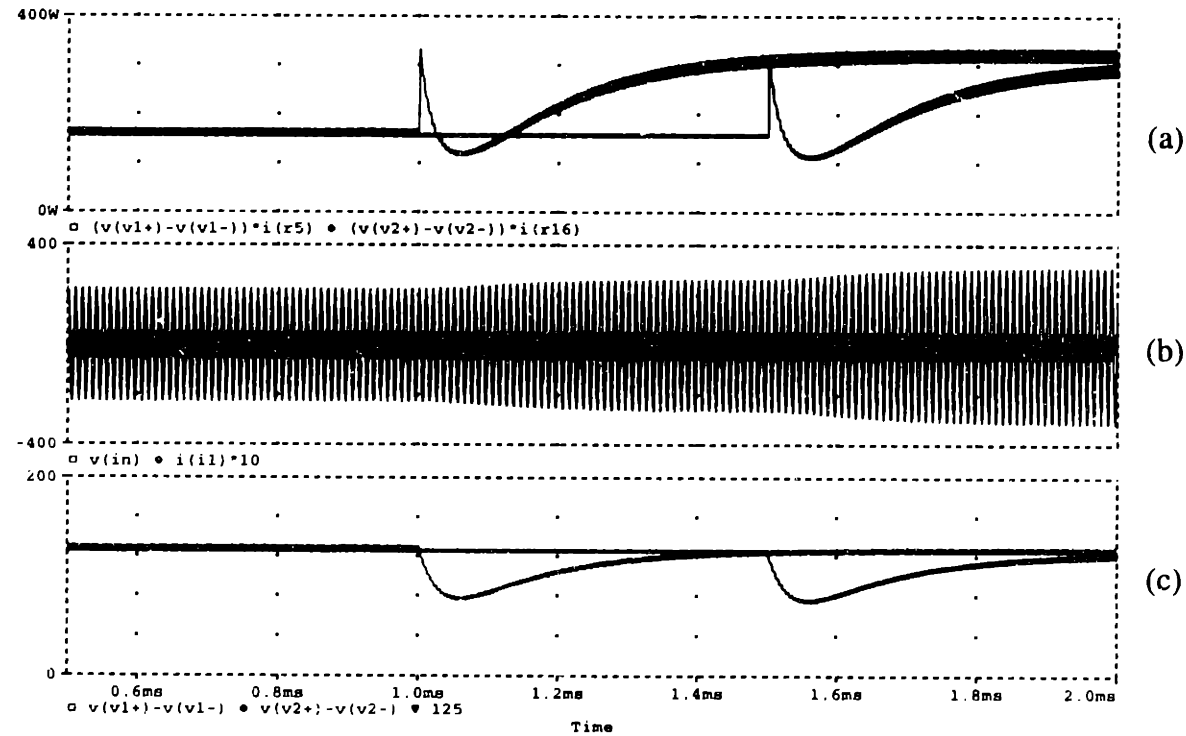



**Figure 8.17:** Simulation results for Figure 8.15 driven by a 5-A current source. (a) Load power. (b) Source voltage and current. (c) Load voltage.

the loop current or to resonant the extra inductance with a capacitor. Since adding a resonant capacitor requires specific details about the inverter design, it will not be discussed. The effect of reducing the loop current can easily be checked by simulation.

Figure 8.17 shows the results of a second simulation. This time the drive current was reduced to 5 A, but the same 125-V nominal output voltage was maintained by increasing the turns ratio on the pickups from 1:3 to 1:30. Again, the same steps in load power were applied, shown this time in Figure 8.17 (a). The scales on the second axis (b) indicate that the inverter VA product is significantly lower. The exact VA product is 822 W for the same 650 W of real power delivered to the load, a definite improvement. However, an unfortunate side-effect is visible in Figure 8.17 (c). The momentary oscillations that occurred in the previous simulation now appear as large dips (over 0.2 ms wide) in the output voltage. The low points on these dips are approximately 53 V (or 42%) below nominal. The oscillations are caused by an interaction between inductances within the pickup transformer, the load filter capacitor, and the load. A method for estimating the magnitude of these oscillations would surely aid system design.

Figure 8.18 show a simple circuit that models the transient behavior of the oscillations. The following steps show how this model can be used to estimate a bound on the peak of the voltage transient. The steps assume a doubling in load power on Load #1 in Figure 8.15. The source $V_S$ is set equal to the nominal DC output voltage, in this case 125 V. The elements $R_s$, $C_f$ and $R_L$ represent the secondary-side resistance of the pickup, the filter capacitor and the load, respectively. Values for these elements can be taken directly from Figure 8.15 as follows:

$$R_s = R_4 = 10\,\text{m}\Omega \qquad C_f = C_2 = 0.5\,\mu\text{F} \qquad R_L = R_6 = 100\,\Omega\,. \tag{8.5}$$

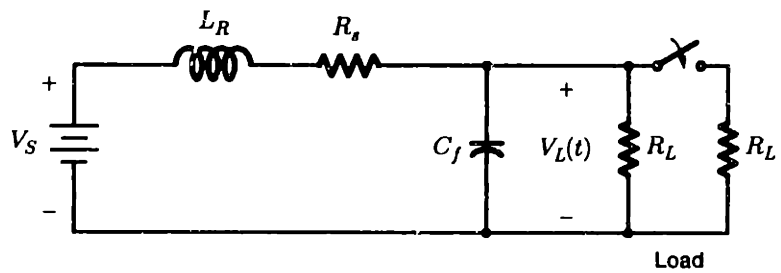



**Figure 8.18:** A simplified model for load transient calculation.

The inductance $L_R$ represents the impedance seen when looking in the secondary terminals of the pickup transformer, thus

$$L_R = (L_m + L_s)N^2 \tag{8.6}$$

where $L_m$ and $L_s$ are 1.75 $\mu$H and 1.97 $\mu$H, respectively. The transformer turns ratio $N$ takes on values of 3 and 30 in the two simulations.

Closing the switch on the model in Figure 8.18 causes an identical voltage transient to that observed in the simulations. It can be shown that, neglecting $R_s$, an upper bound on the peak amplitude of this perturbation is given by

$$|D| < V_{dc}\frac{L_R}{R_{dc}}\left(\frac{\omega_n}{\sqrt{1-\zeta^2}}\right)e^{-\zeta} \tag{8.7}$$

where

$$\zeta = \frac{1}{R_{dc}}\sqrt{\frac{L_R}{C_f}} \tag{8.8}$$

and

$$\omega_n = \frac{1}{\sqrt{L_R C_f}}. \tag{8.9}$$

Substituting values into (8.7) for the two simulated cases, results in bounds of 9.5 V for the first simulation ($I_p = 50$ A) and 78 V for the second ($I_p = 5$ A). These values overestimate the PSPICE results by as much as 50%, but they are close enough to serve as design aids. The model reveals that the bound $|D|$ scales proportionally with $L_R$. So the smaller $L_R$, the better we expect the disturbance rejection of the system to be, and it has just been shown that $L_R$ can be decreased by increasing the inverter current $I_p$.

### 8.2.4 Current-Fed Design Guidelines

The simulations in the proceeding subsection indicate that a multi-stage current-fed system is feasible. A typical two-stage system was simulated, which delivers power to two independent loads. Since the specifics of the system—inverter current, operating frequency and transformer parameters—where chosen somewhat arbitrarily, it is necessary

to know how changes in these design parameters will influence system performance. Table 8.4 provides a set of design guidelines that outline the effects.

| Design Parameter (Change) | Effects |
|---|---|
| Operating frequency (increase) | **Positive Effects:**<br>• Lighter/smaller system components.<br>• Increased system power handling for a fixed A·T rating.<br>• Less energy storage and better transient performance.<br>**Negative Effects:**<br>• More core, ohmic, and switching losses. |
| Inverter current (increase) | **Positive Effects:**<br>• Improved transient performance.<br>**Negative Effects:**<br>• Increased inverter VA requirement.<br>• More ohmic losses.<br>• More EMI. |
| Coupling transformer parameters (increase $L_m$) (decrease $L_p$, $L_s$) | **Positive Effects:**<br>• Less voltage droop on downstream loops.<br>• Lower inverter VA requirement.<br>**Negative Effects:**<br>• Achieved through expensive windings, more core volume, or a smaller airgap. |

**Table 8.4:** Current-fed design guidelines.

## 8.3 Summary

A multi-stage inductively-coupled power transfer system might be ideal for future robotic manipulators. Eliminating direct ohmic connections opens the door for a variety of rotating, sliding, and/or separable links. Such designs seem especially desirable for limbs incorporating synthetic muscles as actuators. Part II of this thesis has examined many aspects of multi-stage inductively-coupled systems, with particular focus on power-transfer architectures. The mechanical and electrical issues associated with inductively-coupled joints were briefly discussed. Then two different approaches to power transfer were reviewed, and an extended version of the current-fed architecture in [54] was introduced. In each case possible design trade-offs were explained.

Based on the discussion in the preceding sections, it is possible to draw some general conclusions about which approach, voltage-fed or current-fed, is more suitable for a given application. Table 8.5 below, compares eight key features of voltage-fed and current-fed systems. The table allows the reader to draw his own conclusions based on the needs of his particular application. Systems marked by a solid circle offer an advantage with respect to the adjacent feature. Two empty circles indicates that neither system offers an advantage. If all the features are weighted equally, the voltage-fed approach is preferred.

| System Feature | Voltage-Fed | Current-Fed |
|---|---|---|
| • Simple (inexpensive) inverter circuit | ● | ○ |
| • Low inverter VA requirement (i.e. high efficiency) | ● | ○ |
| • Contactless addition/removal of power "pickups" | ○ | ● |
| • Good load disturbance rejection | ● | ○ |
| • Lighter and smaller system components | ● | ○ |
| • Expandable to any number of segments | ● | ○ |
| • Adjustable load voltage and current | ○ | ○ |
| • Conveniently expanded to bidirectional power flow | ● | ○ |

**Table 8.5:** Feature summary of the two power-transfer architectures.

# PART III

# AN INDUCTIVELY-COUPLED POLYMER-GEL ACTUATOR

## Chapter 9

# Overview

Parts I and II of this thesis have focused on inductively-coupled systems where the coupling "halves" are held within a relatively close spacing. As a result, the quality of the magnetic coupling is good. Part III of this thesis focuses on the opposite end of the spectrum of inductively-coupled applications, where the quality of the magnetic coupling is poor. Specifically, an inductively-coupled system is developed that supplies activation energy to a polymer-gel actuator. The poor power factor and large apparent power required for this application present special challenges for the drive electronics.

Polymer gels are composed of a crosslinked network of polymers suspended in a solvent [103]. Under certain conditions, gels exhibit reversible changes in volume (by as much as 1000-fold) that can be triggered by variations in a number of environmental factors, including temperature, solvent composition, or pH [38, 102, 105]. Part III of this thesis describes new techniques for remotely triggering polymer gels using a non-contact inductive coupling. Demonstration systems are developed and issues in the design of circuit topologies are discussed. Experimental results are presented that characterize the gel performance, and a position control system is used to demonstrate the gels potential as a controllable linear actuator.

## 9.1 Background

Thermally responsive polymer gels are "seeded" with a ferromagnetic material. The resulting magnetically-activated gels (mag-gels) are then remotely activated through

induction heating. In particular, when the mag-gels are coupled to an alternating magnetic field, losses within the seed material generate heat. This heat raises the temperature of the surrounding gel and triggers a volume-phase transition. When the field is removed, the seed/gel system cools, and the volume-phase transition reverses. This system is an inductively-coupled electromechanical system because energy transfer to the seed/gel system occurs contactlessly through a magnetic field.

The mag-gel system has many possible applications. It could be used to trigger a gel under the skin or in any remote location where an electromagnetic field could penetrate. For example, gels loaded with appropriate, beneficial solvents could be used in controlled drug-release applications either *in vivo* or *in vitro*. In addition, gels could in principle be used as actuators in servomechanisms and sensors, which range in size from microscopic (silicon) mechanisms to larger devices comparable in size and force density to biological systems. For example, polymer gels could act as synthetic muscles that provide direct, quiet, and swift linear motion with useful force densities [77]. Gel actuators may enable design techniques and approaches that differ from those used for traditional systems employing electromagnetic actuators. Since gels could be layered and routed conformally with an underlying mechanical structure, gel actuators might lead to servomechanisms with a range and complexity of motion that is difficult to attain with bulk electromagnetic actuators. Because the mass of a gel actuator can be distributed, gels may permit the construction of servomechanical systems with reduced moments of inertia and improved dynamic performance.

The development of magnetically-activated gels presents its own unique set of design issues. The presentation that follows describes on-going work in the area. It includes detailed descriptions of all power-electronic designs, seeding methods, loss mechanisms, and gel formulations. This original research introduces new techniques that combine the fields of power electronics and polymer gels.

## 9.2 Gel Design

Experimentation was conducted with gels composed of *N*-isopropylacrylamide (NIPA). Gels were prepared by a free radical polymerization process in water with NIPA mono-

mer, a crosslinker, and an appropriate polymerization initiator and accelerator. Typical formulas for the preparation of this widely studied gel are presented in [38]. Pre-gel solutions were permitted to polymerize in capillary tubes of 1.073 mm in diameter. After gelation, the gels are removed from the capillary tubes and washed to remove residual chemicals. The gels are then placed in a solvent, usually deionized water, for experimentation.

If an appropriate seed material is incorporated in or around the gel matrix, at least three loss mechanisms will induce heating from a magnetic field. The loss mechanisms include ohmic heating from eddy currents, hysteresis losses and mechanical (frictional) losses. These mechanisms will be discussed shortly. Depending on the seeding method, the seed material is incorporated either during or after gel formation. Three seeding methods are considered below.

### 9.2.1 Method 1 — Lumped Seed Material

In this approach, the gel is seeded by inserting a conductive, possibly ferromagnetic, material directly into the gel sample. For example, a short segment of a steel pin can be inserted. Due to the macroscopic size of the seed, significant heating can be induced with moderate field strength at frequencies of only a few hundred kilohertz. Ohmic heating from eddy currents tends to be the dominant loss mechanism with this approach. Induced losses are strongly dependent on the shape, conductivity and permeability of the seed target. In [31], an excellent analysis is presented of the relative heating efficiencies of cylindrically-shaped ferromagnetic seeds with varying material properties and form factors.

### 9.2.2 Method 2 — Distributed or Powdered Seed Material

In this approach, the gel is seeded with conductive powder or flakes that are small compared to the gel dimensions. In our experiments, a concentration of ferromagnetic particles, approximately 5% by weight (0.5% by volume), is first coated with polyvinyl alcohol (PVA) and then mixed into the pre-gel solution. We have found that coating with PVA or a similar polymer is essential to ensure good suspension and dispersal of the metal flakes during the free radical polymerization process [76]. When the gel is formed, these particles are permanently trapped in the crosslinked polymer network. Particle shape and material properties must be selected carefully to provide desired heating, which comes pri-

marily from eddy current and hysteresis losses. Our experiments with readily available commercial powders have revealed that ferromagnetic nickel flakes (approximately 30 $\mu$m in diameter and 0.4 $\mu$m in thickness) sold by Novamet [84] provided adequate heating at the given concentration and magnetic excitation both of which are described in the next sections. These observations agree qualitatively with published results in [90].

### 9.2.3 Method 3 — Ferrofluid Solvent

In Methods 1 and 2, the seed material is entrained in the polymer matrix. During a phase transition, the polymer network collapses around the seed material(s) as the gel solvent—deionized water—leaves the interstitial spaces of the polymer network. In Method 3, the gel is not seeded by a trapped material as in Methods 1 and 2. Instead, the solvent is replaced by a ferrofluid. A typical ferrofluid is composed of water, a surfactant, and a small amount (1 – 4% by volume) of microscopic ferromagnetic material (roughly 87 Å across). The surfactant suspends the ferromagnetic particulants and prevents them from separating out of the mixture. When used as a gel solvent, ferrofluid diffuses into the expanded gel. The microscopic ferromagnetic particles fill and surround the expanded polymer network. We have found that when ferrofluid is excited by a strong magnetic field in the range of 2 – 3 MHz, significant heating occurs. Microscopic mechanical or hydrodynamic motion may be a substantial source of dissipation in this method.

## 9.3 Experimental Overview

The diagram in Figure 9.1 illustrates the test apparatus used for the preliminary experiments. It may be helpful to refer to this diagram throughout the discussion in this section. A power-electronic inverter generates a high-frequency AC voltage from a DC source. The voltage waveform is impressed across the terminals of a solenoid. The applied AC voltage causes an alternating current to flow in the winding. This current, in turn, generates a high-frequency alternating magnetic field inside the coil. For a finite length solenoid, the time-varying magnetic field $H(t)$ at the center of the coil is

$$H(t) = \left[\frac{l/d}{\sqrt{1 + (l/d)^2}}\right]\frac{Ni(t)}{l} \tag{9.1}$$

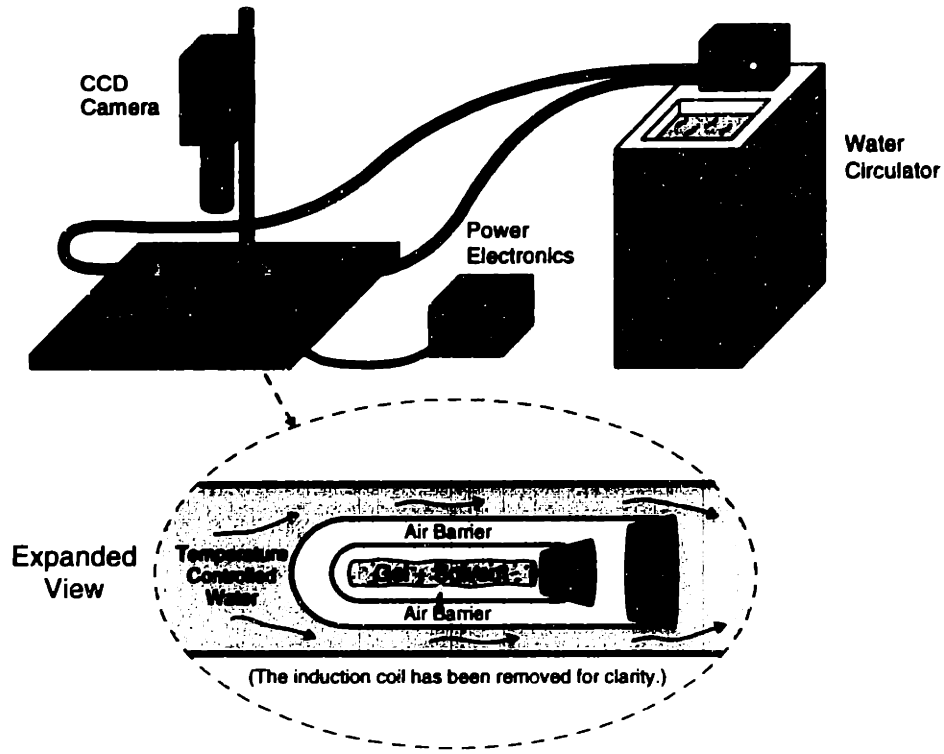



**Figure 9.1:** Experimental setup for equilibrium measurements.

where $N$ is the number of turns in the coil, $i(t)$ is the coil current, and $l$ and $d$ are the length and diameter of the coil, respectively. Note that in a practical field application, other magnetic coil arrangements, e.g., a Helmholtz coil pair, could be employed instead of the solenoid. High-frequency, high-permeability materials could be used with a winding, or winding set, to guide magnetic flux to a target location.

The solenoid in our experiments is formed of litz wire wound on a triple-walled test vessel. As Figure 9.1 illustrates, temperature-controlled cooling water circulates in the outer jacket of the test vessel to prevent coil heat from influencing the temperature of the inner test chamber. An insulating air space separates the inner test chamber from the outer water jacket. Finally, a sample under test, such as a gel and its solvent, is inserted into the test chamber. The magnetic field created by the solenoid is used in our experiments to excite losses in the target seed material in the test chamber. Three predominant loss mechanisms are illustrated in Figure 9.2 (see [8], [32] and [89] for more details).

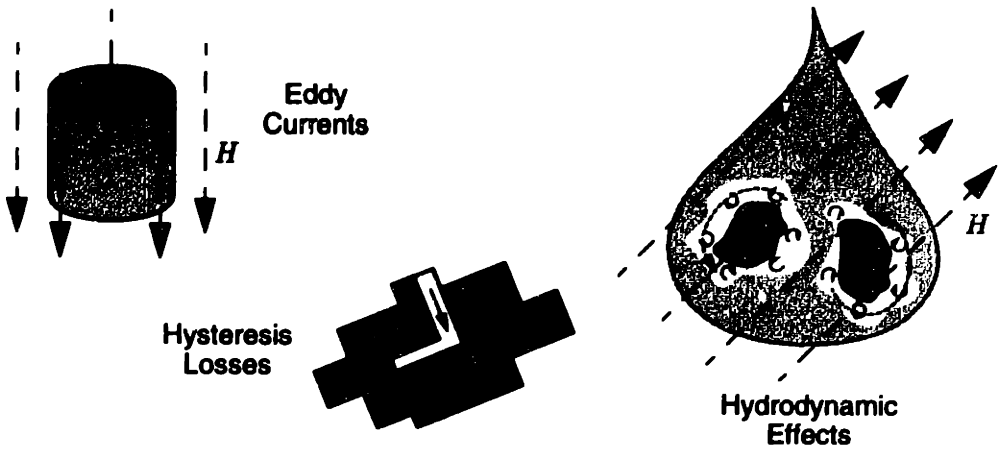



**Figure 9.2:** Inductively-coupled loss mechanisms.

### 9.3.1 Loss Mechanisms

**Eddy Currents:**

High-frequency magnetic fields induce eddy currents in conductive materials. At sufficiently high frequencies, these currents flow in a thin surface region because of skin effect and can result in substantial ohmic power dissipation. When this surface region is small compared to the dimensions of the target seed, the current can be modeled with reasonable accuracy as flowing uniformly in a region with a depth equal to the "skin depth." The externally applied magnetic field is essentially excluded from the interior, bulk regions of the seed, i.e., from regions located several skin depths from the surface. The skin depth for a magnetically linear material is

$$\delta = \sqrt{\frac{2}{\omega\mu\sigma}} \tag{9.2}$$

where $\omega$ is the magnetic excitation frequency, $\mu$ is the magnetic permeability of the material and $\sigma$ is the conductivity of the material.

During early experimentation with different power-electronic drives and coil configurations, single cylinders of test materials were employed in the test stand to verify theoretical heating predictions and circuit performance. These tests were functionally identical to the heating techniques outlined above in Method 1. Consider, for example, an enamel-coated type-4403 aluminum rod (25.4 mm in length and 1.59 mm in diameter) in 2 cc of

deionized water in the induction heating test stand.[1] Gels used in our experiments were often well over 90% water by volume, so the performance of a cylinder of seed material in pure water is a reasonable indicator of anticipated performance in a gel.

In the experiments with the aluminum rod, a 2.46-MHz sinusoidal magnetic field with a peak strength of 3900 A·T/m was applied axially by the solenoid in the test stand. Type-4403 aluminum has a measured conductivity of $2.87 \times 10^7$ mho/m and, at this frequency, exhibits a skin depth of approximately 60 $\mu$m. The ohmic power dissipation can be computed by solving precisely for the field and current distributions in the rod and then integrating a continuum version of Ohm's Law in the current-carrying regions. This approach is taken, for example, in the analyses presented in [8] and [114]. Since the skin depth is relatively small, we employ a simplification that gives excellent results in comparison to the exact solution (in terms of Bessel's functions) of the magnetic diffusion equation in cylindrical coordinates.

We assume that the external, axial magnetic $H$-field is terminated by a uniform current density $J$ flowing in the skin region and that the current density and $H$-field in deeper, interior regions of the rod are zero. The magnitude of the current density can be approximated by first finding an azimuthal surface current of magnitude $K$ sufficient to terminate the externally applied $H$-field and then distributing this surface current uniformly over the width of a skin depth. Applying the appropriate magnetic field boundary condition at the surface of the rod, we find that $K = H_o$, where $H_o$ is the magnitude of the external axial $H$-field. The current density $J$ is approximately $K/\delta$, and the instantaneous ohmic dissipation $P(t)$ can now be found by applying a continuum version of Ohm's law integrated over the thin shell of volume $V$ in which current flows, as follows:

$$P(t) = \frac{J^2}{\sigma} V = \frac{K^2}{\sigma \delta^2} V. \tag{9.3}$$

Under the assumption that the current flows entirely in a region the thickness of one skin depth, the volume of the thin shell is

1. Due to an increase in skin depth and, thus, reduced ohmic losses, a non-ferromagnetic seed like aluminum generally provides inferior heating when compared to an otherwise similar but ferromagnetic seed. However, aluminum is magnetically linear, and empirically observed heating performance can, therefore, be reconciled with analytical predictions.

$$V = \pi(r^2 - (r - \delta)^2)h \tag{9.4}$$

where $r$ and $h$ are the radius and length of the rod, respectively. The $H$-field applied in our experiments varies sinusoidally with time. The steady-state time-averaged power dissipation $P_d$ may, therefore, be found by replacing the peak surface current $K$ with the rms surface current $K_{rms} = K/\sqrt{2}$ as follows:

$$P_d = \frac{K_{rms}^2}{\sigma\delta^2}V. \tag{9.5}$$

For the type-4403 sample described above, with a peak induction coil current of 8.88 A and externally applied peak $H$-field of 3900 A·T/m, the final temperature in the 2-cc water bath was 10.87 °C above ambient. Using (9.5), we compute a predicted steady-state power dissipation in the rod of $P_d = 0.539$ W.

To verify the accuracy of the computed power dissipation, the rod in the 2-cc water vessel was replaced with an ohmic resistor driven from a DC supply. With the induction coil deactivated, a power dissipation of 0.555 W in the resistor was required to achieve the same 10.87 °C temperature rise as in the induction heating experiment. We presume that this power dissipation level is the actual power dissipated in the aluminum rod during our induction heating experiments. The predicted power dissipation is within 2.9% of this actual dissipation.

Several factors complicate the calculation of eddy current power dissipation when the seed material is not a single, relatively long rod. Closed-form determination of the externally applied field may be difficult when the seed material is irregularly shaped. Also, neighboring seeds can locally distort the magnetic field around a particular seed, complicating the determination of the externally applied $H$-field. If the seed material is ferromagnetic with a nonlinear magnetization characteristic, the classic linear solution of the magnetic diffusion equation employing skin depth is inaccurate. Approaches for dealing with nonlinear magnetization typically follow the approach in [3], which involves substantial simplification of the material behavior.

**Hysteresis Losses:**

An applied time-varying magnetic field constantly reorients the domains in a ferromagnetic material. The resulting power dissipation is proportional to the area of the $B$-$H$ curve for the block material and the field excitation frequency. These losses can be very substantial or even exceed eddy current losses at sufficiently high frequencies.

**Hydrodynamic Effects:**

Small magnetizable particles suspended in a viscous solvent can physically spin in a time-varying magnetic field. The resulting mechanical motion may create significant frictional heating. We speculate that this may be a heating mechanism in some ferrofluids [108].

# Chapter 10

# Experimental Results

## 10.1 Power Electronics

Three prototype power circuits were built to demonstrate the activation methods described in Chapter 9. Key considerations in the design of the prototypes were flexibility, power consumption, and portability. The resulting designs employ two different circuit topologies: the zero-voltage-switching full-bridge shown schematically in Figure 10.2 and the resonant class-E converter shown in Figure 10.4. Each circuit is discussed in the subsections that follow.

The function of the power circuit is to impress an AC waveform across an air-core solenoid (the induction coil) which surrounds the sample to be induction heated. The combination of the solenoid and test sample form an inductively-coupled system that is most easily thought of as an air-core transformer with a shunted secondary winding. Schematically, this load is modeled as shown in Figure 10.1. An "L" model for the transformer is shown, where the resistance $R_p$ represents the parasitic winding resistance of the solenoid. The resistance $R_L$ accounts for the real power dissipation in the test sample. It is important to note that, in our experiments, poor coupling exists between the primary and secondary of the transformer because the sample occupies only a tiny fraction of the volume within

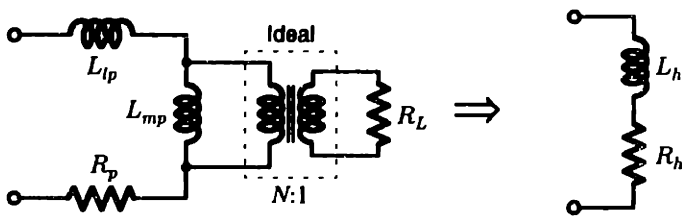



**Figure 10.1:** Simplified load model of the induction coil.

the solenoid. This makes it reasonable to ignore the effects of the load resistance $R_L$ and, thus, simplify the load model as shown. Also, any change in inductance that results from insertion or removal of the sample is small. The parameters $L_h$ and $R_h$ will vary with frequency. Nevertheless, they may be treated as constants under fixed-frequency operation.

## 10.2 ZVS Full Bridge

The first prototype, a full-bridge inverter, impresses a 240-kHz AC square wave across the load using four switches in the H-Bridge configuration shown in Figure 10.2. The switches were implemented with IRFP450 MOSFETs. Gate-drive isolation for the high-side switches was accomplished through pulse transformers. The MOSFETs $Q_1 - Q_4$ are switched at almost 100% duty ratio using a phase-shifted pattern as discussed in Chapter 4. This technique allows the inductor current to ring with the parasitic capacitances $C_p$ of the MOSFETs. This ring enables ZVS at nodes A and B and significantly reduces switching losses. The primarily inductive load provided by the induction coil allows for ZVS under all heating conditions.

The bridge operates from a 120-V DC bus, which is stepped-up from a 24-V battery pack using a boost converter. As a result, the entire system is portable. The bus voltage is flexible, but limited by the 500-V breakdown voltage of the MOSFETs. The prototype outputs a $\pm 120$-V square wave at 240 kHz. This results in a triangular AC current ($\approx \pm 4$ A) through the load solenoid. Output waveforms, $v_o$ and $i_o$, from the first prototype are plotted in Figure 10.3. The inductance of the load solenoid $L_h$ was 30.2 $\mu$H. The coil

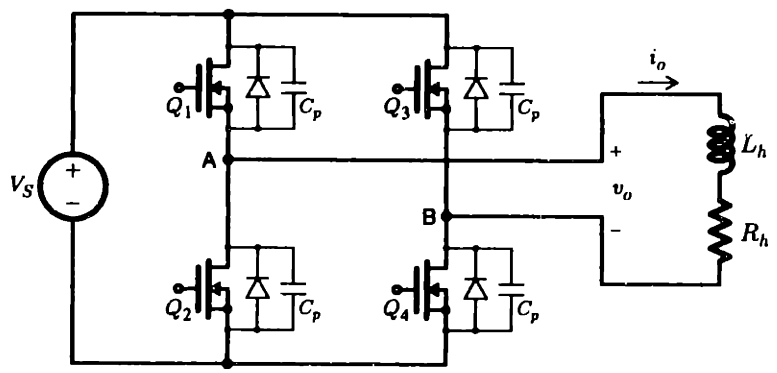



**Figure 10.2:** Simplified schematic of the ZVS full-bridge inverter.

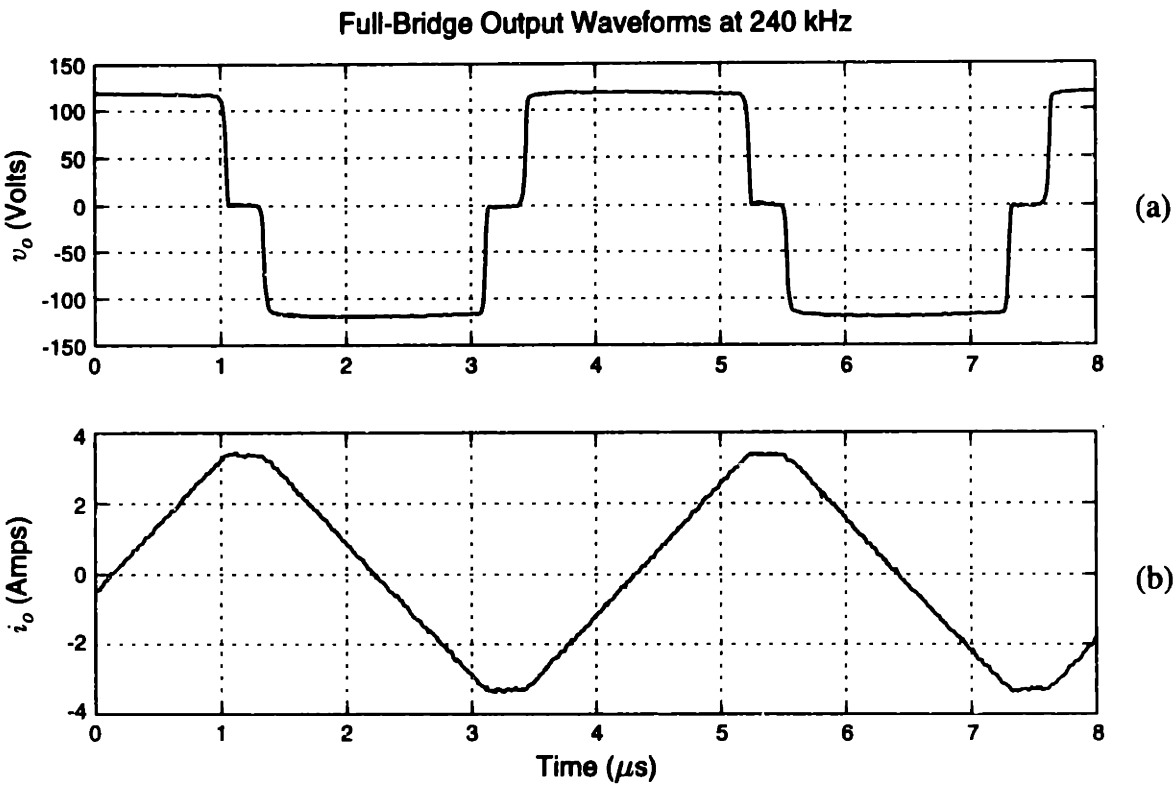



**Figure 10.3:** ZVS full-bridge experimental waveforms at 240 kHz.

was constructed by winding 44 turns of multi-strand litz wire on a 5.5-cm long plexiglass former. The system develops a peak $H$-field intensity of approximately 2500 A·T/m at the center of the solenoid.

The first prototype is well-suited for triggering gel samples seeded with lumped materials (Method 1). Its most valuable asset is its ability to operate with ZVS over a wide range of load inductance. This allows the user to exchange, for instance, the 30.2-$\mu$H output solenoid with a 60.0-$\mu$H Helmholtz coil without adjusting the circuit parameters. Zero-voltage-switching is possible as long as the amplitude of the voltage ring $V_r$ at nodes A and B exceeds the bus voltage $V_S$ as shown by

$$V_r = \left( I_{pk} \sqrt{\frac{L_h}{2C_p}} \right) > V_S . \tag{10.1}$$

Assuming a square-wave output and a purely inductive load, $I_{pk}$ can be expressed as

$$I_{pk} = \frac{V_S}{4 L_h f_s} \tag{10.2}$$

where $f_s$ is the switching frequency. It is now possible to define an upper limit for $L_h$ under which ZVS can be sustained as follows:

$$L_h < \frac{1}{32 C_p f_s^2}. \tag{10.3}$$

Substituting values for $C_p$ and $f_s$ of 720 pF and 240 kHz into (10.3), we find an upper limit on $L_h$ of 754 $\mu$H.

| Source | Loss |
|---|---|
| Gate Drive & Logic | 5.4 W |
| FET Conduction | 4.5 W |
| Coil Conduction | 5.5 W |
| Boost Converter | 2.7 W |
| Total | 18.1 W |

**Table 10.1:** ZVS full-bridge losses.

Operating as in Figure 10.3, the prototype consumes about 18.1 W, excluding power delivered to the load. A breakdown of the losses are given in Table 10.1. The relative performance of the circuit for the induction-heating application can be judged by comparing the reactive power $P_r$ flowing through the induction coil to the total real power dissipation in the circuit. A relationship between $H$-field intensity and reactive power is derived in the next subsection. The measured waveforms in Figure 10.3 demonstrate a reactive power $P_r$ of 253 VA. Therefore, we can define a figure of merit $Q_{\text{circuit}}$ for this prototype as the ratio of reactive power to power loss as follows:

$$Q_{\text{circuit}} = \frac{253 \text{ VA}}{18.1 \text{ W}} = 14.0. \tag{10.4}$$

The full-bridge circuit meets all the design goals. The circuit is flexible, tolerant to load variations, portable, and consumes relatively little power. However, the circuit is not easily adapted for use at the higher frequencies (several MHz) required to activate gels seeded with Methods 2 and 3. As the switching speed is increased, it becomes apparent that the full-bridge topology has several disadvantages. First, the complex gate-drive circuitry of the full bridge quickly leads to considerable gate-drive losses, and tolerances

must be tight in order to avoid catastrophic shoot-through. Second, high current and voltage derivatives during switching often excite unwanted resonances between device parasitic inductances and capacitances. Extreme care must be taken during circuit layout to minimize these unwanted resonances and the resulting increase in losses. Third, the bus voltage can quickly reach unreasonable levels (thousands of volts) when only a few amps are driven through a 30.2-$\mu$H coil at 2 or 3 MHz. A matching transformer is the obvious solution to this last problem, but the design of a low-loss, high-power, high-frequency transformer presents its own challenges. The most logical solution to this problem is, then, to employ a resonant converter topology.

## 10.3 Resonant Class-E Converter

Two additional prototypes were built using a resonant, class-E topology: a portable 500-kHz converter and a bench-top 2.5-MHz converter. The class-E converter—developed by Sokal and Sokal in [100]—is a current-fed, resonant, DC-to-AC converter. It has been studied extensively in [11], [35], and [88], for example. The converter is particularly suited to this application because of its simple design, resonant operation, and high efficiency. A simplified schematic of the converter is shown in Figure 10.4. The basic circuit consists of a power MOSFET, a $LCR$ series-resonant circuit, a shunt capacitor $C_s$ and a filter inductor $L_S$.

The circuit is driven at a fixed frequency by a square wave $q(t)$ to the MOSFET gate. The filter inductor $L_S$ is assumed to be large enough so that the supply current $I_S$ is DC, neglecting a small ripple component. When the MOSFET is on, $C_s$ is short-circuited, and

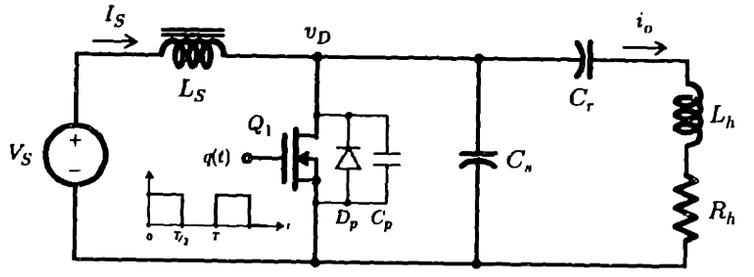



**Figure 10.4:** Simplified class-E schematic.

a resonant circuit is formed by $L_h$, $C_r$, and $R_h$. During this time the resonant frequency of the load is

$$f_{r1} = \frac{1}{2\pi\sqrt{L_h C_r}}. \tag{10.5}$$

When the MOSFET is off, $C_s$ and $C_r$ appear in series and the load resonant frequency shifts to

$$f_{r2} = \frac{1}{2\pi\sqrt{L_h C_r C_s/(C_r + C_s)}} \tag{10.6}$$

where $f_{r2}$ is higher than $f_{r1}$. The switching frequency $f_s$ is fixed somewhere between $f_{r1}$ and $f_{r2}$. The exact frequency depends on the component values of the load circuit.

The parasitic load resistance $R_h$ is small in this application, so the $Q$ of the series-resonant load circuit is very high. The load circuit $Q$ is define as

$$Q = \frac{2\pi f_s L_h}{R_h}, \tag{10.7}$$

which is essentially the $Q$ of the induction coil at the drive frequency. The high $Q$ ensures a current $i_o$ through the load that is approximately sinusoidal at the drive frequency. During the half-cycle when the MOSFET is off, the drain-to-source voltage $v_D$ will ring, almost sinusoidally, from zero to its peak and then back towards zero. Optimum operation of the class-E circuit occurs when both the value and the slope of the voltage $v_D$ are zero at the instant the MOSFET turns on [51]. For this to occur the resistance $R_h$ and the capacitance $C_s$ must be defined as follows:

$$R_h = R_{opt} \approx 0.87\left(2\pi f_s L_h - \frac{1}{2\pi f_s C_r}\right) \quad \text{and} \quad C_s \approx \frac{0.18}{2\pi f_s R_{opt}}. \tag{10.8}$$

The above relationships between $C_r$, $C_s$, $L_h$, $R_h$, and $f_s$ cannot be easily guaranteed in practice. Generally, the voltage $v_D$ will hit zero with a negative slope before the MOSFET turns on. In this "suboptimum" mode of operation, the MOSFET antiparallel diode will clamp $v_D$ near zero, ensuring zero-voltage turn-on of the switch. Despite its name, suboptimum operation is still very efficient.

Assuming that $R_h$ is variable, ZVS can be achieved over the range $0 \leq R_h \leq R_{opt}$. In practice, the circuit is "tuned" by adjusting $f_s$ between $f_{r1}$ and $f_{r2}$ until ZVS operation occurs. Once tuned, the circuit can be operated open loop. The output current amplitude can be controlled by adjusting the source voltage $V_S$. The MOSFET switch must be sized to carry a peak current of $i_o + I_S$ and a peak voltage significantly above $V_S$. This results in a high switch stress, which is a disadvantage of the class-E circuit.

Two class-E prototype converters were built. However, only the 2.5-MHz bench-top model was used for experimentation. Its construction is briefly described here. Full schematic details and photographs of both prototypes are available in Appendix C. The bench-top prototype was designed to operate over a range of 1 – 3 MHz. The higher frequency was necessary to test seeding Methods 2 and 3. Switch $Q_1$ was realized using two IRFP450 MOSFETs in parallel for a low on-state resistance and, thus, increased current capacity. The 500-V drain-to-source breakdown voltage of the IRFP450 provided a more than sufficient range for experimentation. Two new induction coils were wound. The first, coil A, is a 4.54-$\mu$H solenoid wound with 29 turns of 1700-strand, 48-AWG litz wire on a former 5.85 cm in length and 1.6 cm in diameter. The second, coil B, is a 12.89-$\mu$H solenoid wound with 28 turns of 1700-strand, 48-AWG litz wire on a former 5.72 cm in length and 2.83 cm in diameter. The exceptionally high strand count of the litz wire is required to minimize ohmic power losses, which are a result of high-frequency skin effects in the wire. The $Q$ values of the two solenoids, measured at 2.5 MHz, are 117 and 170, respectively. The resonant capacitor $C_r$ was selected according to (10.5) (720 pF for coil A and 300 pF for coil B). The capacitor $C_s$ was eliminated, relying instead on the parasitic drain-to-source capacitance $C_p$ of the paralleled MOSFETs (approximately 1440 pF).

Experimental waveforms in Figure 10.5 demonstrate the operation of the converter. The figure shows the circuit operating with coil B and $f_s = 2.46$ MHz. The output current is sinusoidal at $\pm 12.5$ A, yielding a peak $H$-field intensity of 5500 A·T/m at the center of the solenoid. The DC input voltage $V_S$ was set to 28 V, and the input current $I_S$ was measured at 4.0 A. Under the operating conditions just described, the second prototype consumes 135.2 W, excluding power delivered to the load. A breakdown of the losses is provided in Table 10.2.

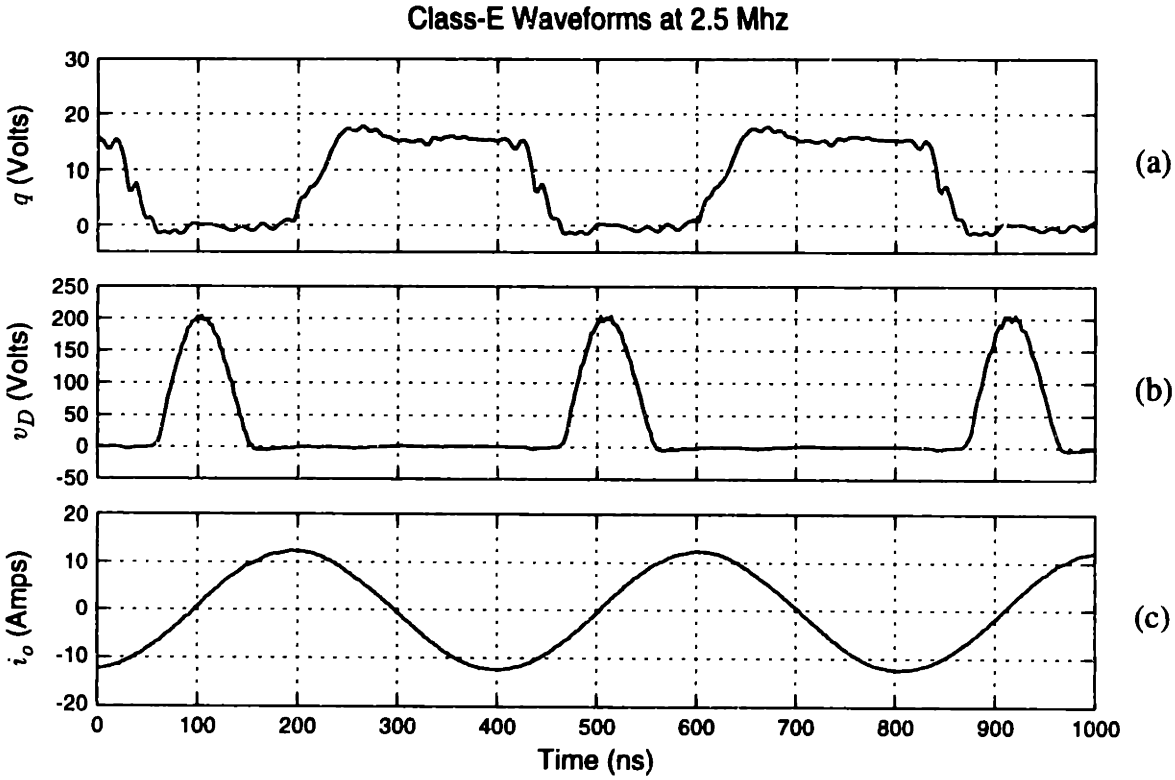



**Figure 10.5:** Class-E experimental waveforms at 2.5 MHz.

| Source | Loss |
|---|---|
| Gate Drive & Logic | 18.2 W |
| FET Conduction | 22.2 W |
| Coil Conduction | 89.8 W |
| Cooling Fan | 5.0 W |
| Total | 135.2 W |

**Table 10.2:** Class-E Circuit Losses

As with the ZVS full bridge, the ratio of the reactive power $P_r$ across the coil and the total real power dissipation in the circuit serves as a performance indicator. Since the class-E converter produces a sinusoidal load voltage and current, a relationship between the $H$-field intensity inside the induction coil and the reactive power across it can be analytically derived as follows. The reactive power across the coil is

$$P_r = I_{rms}^2 X_l = I_{\text{pk}}^2 \pi L f_s \tag{10.9}$$

where $X_l$ is the impedance of the coil inductance and $f_s$ is the switching frequency. The variables $I_{rms}$ and $I_{pk}$ represent the rms and peak values of the coil current, respectively. Now, substitute $L$ in (10.9) with the inductance of a long solenoid,

$$L = \frac{\mu_o N^2 A}{l} \tag{10.10}$$

where $\mu_o$ is the permeability of free space, $N$ is the number of turns, $A$ is the cross-sectional area and $l$ is the length. Solving the result for $I_{pk}$ and substituting into (9.1), we find the peak $H$-field intensity to be

$$H_{pk} = k_d \sqrt{\frac{P_r}{\pi f_s \mu_o A l}} = k_d \sqrt{\frac{P_r}{\pi f_s \mu_o V_l}} \tag{10.11}$$

where $V_l$ is the volume contained by the solenoid and

$$k_d = \frac{l/d}{\sqrt{1 + (l/d)^2}}. \tag{10.12}$$

Equation (10.11) shows that the peak $H$-field intensity increases with the square-root of the reactive power $P_r$. Therefore, under the stated assumptions, the reactive power absorbed by the coil inductance is an indicator of the real power dissipation which could be induced in the target seed.

The waveforms in Figure 10.5 demonstrate a reactive power $P_r$ of 15.3 kVA. A figure of merit $Q_{\text{circuit}}$ can again be defined for this prototype as the ratio of reactive power to power loss:

$$Q_{\text{circuit}} = \frac{15.3 \text{ kVA}}{135.2 \text{ W}} = 113.2. \tag{10.13}$$

The performance of the class-E circuit is clearly superior to that of the ZVS full bridge presented above. The theoretical limit for $Q_{\text{circuit}}$ is the $Q$ value of the induction coil itself. Despite the disadvantages of high switch-stress and "tuning" difficulties, the class-E circuit appears to be the best choice for this induction heating application.

## 10.4 Mag-gel Characterization

This section describes preliminary experiments that were intended to verify the operation and performance of the power-electronic drives and the magnetic triggering methods described in Section 9.2.

### 10.4.1 Results

Initial evaluations of the ZVS full-bridge and class-E converters were conducted in the test stand shown in Figure 9.1 with gels seeded by Method 1. Both converters were adequate to trigger gels seeded with macroscopic pins and other lumped seed materials since the seed dimensions are large in comparison to the skin depth. However, for the smaller seed powders employed with Method 2, the high-frequency ($1-3$ MHz) capability of the class-E converter was essential to achieve adequate dissipation in the seed materials. High-frequency fields were also found to be essential to generate sufficient heating in the ferrofluids employed in Method 3.

Figure 10.6 shows the equilibrium phase-transition curve of a NIPA/PVA mag-gel. Gels were seeded with Novamet nickel "leafing grade" flakes using the procedure described for Method 2. The curve in Figure 10.6 shows the normalized gel diameter on the vertical axis versus the peak current in induction coil (coil A) on the horizontal axis. A CCD video micrometer, illustrated in Figure 9.1, was used during the course of the experiment to observe the gel diameter through a fractional gap in the solenoidal winding. A total of 18 diameter measurements were made. At each point, the gel was exposed to a 2.46-MHz magnetic field induced by the indicated peak coil current until the gel diameter equilibrated. The peak coil current can be related to magnetic field intensity using (9.1). The dashed line interpolating the points in the figure serves as a guide for the eye. Under the described conditions, a volume-phase transition occurs at approximately $I_{pk} = 7.5$ A. The resulting volume change is approximately eight-fold.

Tests to verify the efficacy of seeding Method 3 were also performed. Experiments were conducted with gels employing Ferrofluidics EMG 705 ferrofluid as a solvent [25]. Through before-and-after observation of the gel diameter, we were able to observe magnetically-activated phase transitions in these gels. However, it was not possible to make

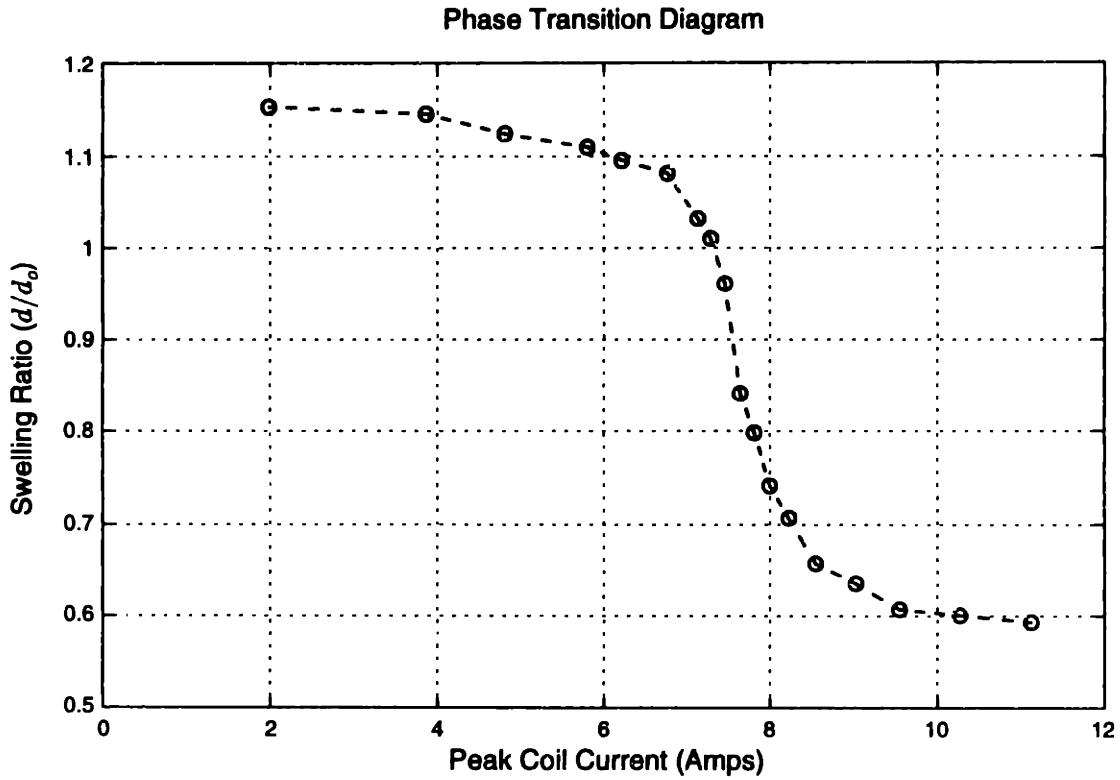



**Figure 10.6:** Equilibrium gel diameter versus current. The circulating water temperature is fixed at 30.0 °C.

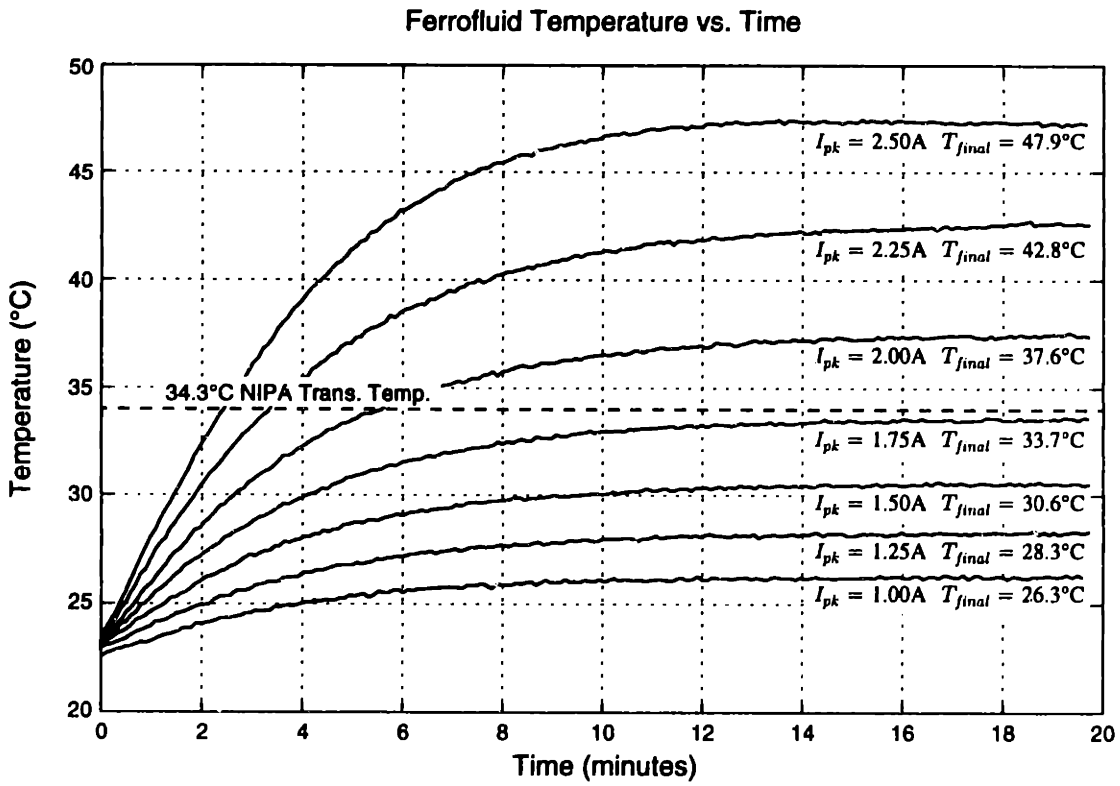



**Figure 10.7:** Ferrofluid temperature versus time for multiple current levels. The circulating water temperature is fixed at 22.0 °C.

real-time observations of the gel dimensions due to the extreme opacity of the EMG 705 fluid. Instead, quantitative measurements of the ferrofluid temperature were made using an Everest 4000.4GH infrared pyrometer in place of the video microscopy system described above [24]. The class-E prototype with coil B, formed on a Liebig condenser, was used in the test stand both to accommodate the larger size of the ferrofluid samples and also to provide an unobstructed axial view for the pyrometer. A test sample composed of 2 cc of ferrofluid in a glass vial was used for the experimental results shown in Figure 10.7.

Figure 10.7 shows the temperature rise in the ferrofluid sample versus time at seven different induction coil currents. The excitation frequency from the class-E converter was again set at 2.46 MHz. NIPA polymer gels undergo a phase transition around 34.3 °C [38]. The dashed horizontal line in Figure 10.7 indicates the NIPA phase-transition temperature for comparison with the dynamic and steady-state induction heating performance of the ferrofluid. It is clear from the figure that for current amplitudes above 2.0 A, ample dissipation occurs in the ferrofluid to trigger a NIPA gel. Significantly higher dissipations are possible as the excitation current is increased. Sustained exposure to the 12.5-A full-power capability of the prototype converter drives the ferrofluid sample to its vapor point in a matter of seconds.

## 10.5 PWM Position Control

In order to utilize polymer gels as actuators in a servomechanical application, it is critical to be able to regulate the length or volume of the gel. In some cases the length may need to follow a trajectory with predefined position and velocity as functions of time. This section describes on-going research into the closed-loop position control of magnetically-activated polymer gels. The work presented in this section is a collaborative effort with colleagues Ahmed Mitwalli and Timothy Denison. The presentation below summarizes the results to date of this combined effort. More complete coverage can be found in [75] and [76].

### 10.5.1 Experimental Setup

Two new mag-gel samples were fabricated for a series of position control experiments. A nickel-seeded gel, 0.51 cm in diameter and 7.37 cm in length, and a ferrofluid gel, 0.51 cm in diameter and 6.67 cm in length, were created. The gel under test was mounted vertically in the apparatus illustrated in Figure 10.8. The experimental apparatus was constructed to regulate the temperature, magnetic field, and mechanical load applied to the gel. The gel sample was recessed within the Liebig condenser and surrounded by a small tube filled with the appropriate solvent—deionized water or ferrofluid. The base of the gel was affixed to a mechanical support, and a silk fiber was attached to the top of the gel with epoxy. An induction coil (coil B from Section 10.3) was wound around the Liebig condenser, and water was circulated through the outer jacket to prevent any direct heat transfer from the induction coil to the gel.

In a series of experiments with the nickel-seeded gel, an air barrier was left between the inner tube containing the gel sample and the outer jacket of the Liebig condenser. This stabilizes the effective "ambient" temperature of the inner tube. The circulating Lauda water was set to 30.0 °C to "bias" the gel near to, but below, its phase-transition temperature (approximately 34.3 °C). In a second series of experiments with the ferrofluid gel, the air barrier was filled with water to increase the coupling with the thermal bath, the temperature of which was adjusted to 15 °C to increase the cooling rate of the gel.

The silk fiber attached to the top of the gel was passed over a low-friction pulley system and connected to an adjustable weight and a light steel rod. The rod was suspended within a linear-variable-differential-transformer (LVDT) that sensed the relative position of the gel [65]. The LVDT allowed for position measurements sensitive to displacements under 0.1 mm. The LVDT output was sampled and incorporated directly into a PC-based data-acquisition and control system. Software running under MATLAB provided full control over the experimentation. In particular, MATLAB sampled the relative gel position and provided a command signal that enables and disables the class-E prototype inverter. In all of the experiments, the field frequency was 2.46 MHz, and the field strength was set to 2460 A·T/m ($I_{pk} \approx 6.0$ A). The use of MATLAB as a control and computation environment allowed for relatively easy synthesis of different feedback compensation schemes.

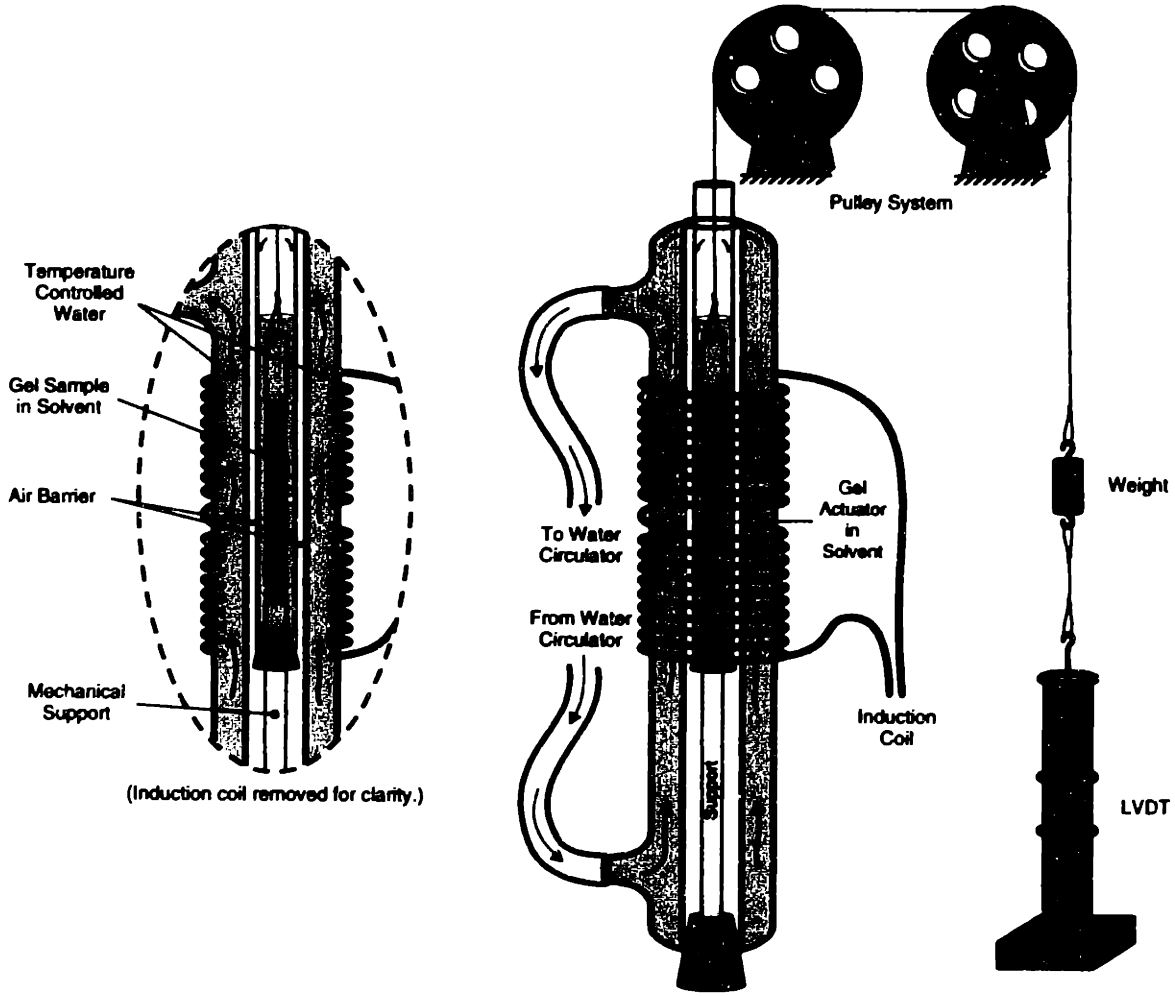



**Figure 10.8:** Experimental setup showing the Liebig condenser and suspended gel.

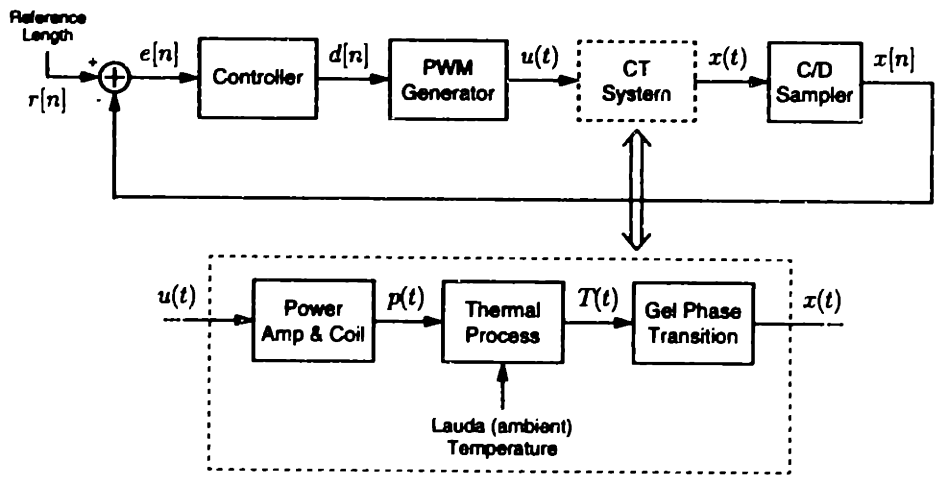



**Figure 10.9:** Block diagram of the closed-loop PWM position controller.

### 10.5.2 Open-Loop Results

A magnetic field was applied to both the nickel-seeded gel and the ferrofluid gel to induce a shrinking transition. In each case, the gels were mechanically loaded with a mass of 5.1 g. Responses for both gels are shown in Figure 10.10. Note that the vertical axes in the figure indicates the length of the steel rod in the bore of the LVDT and not the absolute length of the gel. During the "field on" portion of each plot, the magnetic field was applied to the gels in periodic two-second bursts. A delay of two seconds was inserted between bursts so that the position could be measured without interference from high-frequency noise generated by the power electronics. The dynamic profiles of both gels exhibit two distinct time constants. This is qualitatively consistent with the observations made in [104].

Figure 10.11 plots the open-loop dynamic responses of the nickel-seeded gel under different loading conditions. Loads of 5.1 g and 10.19 g were used. The gel's elasticity causes its equilibrium volume or length to vary with load. As illustrated in Figure 10.11, the gel is longer in both the shrunken and expanded states when it is more heavily loaded. These experiments demonstrate that the shape and rate of the dynamic response of the gel can change with the mechanical load. This change appears more pronounced in the shrinking transition. It also complicates the control development.

### 10.5.3 Closed-Loop Results

A closed-loop feedback system was implemented to provide control of the gel position. The feedback system is illustrated in Figure 10.9. The length of the gel $x[n]$ is sampled every $T=4$ seconds by the MATLAB-based data-acquisition system. The measured error $e[n]$ between the measurement $x[n]$ and a reference $r[n]$ is used to modify a duty-ratio command $d[n]$ which enables the class-E power electronics. The duty ratio can vary from $0-50\%$ allowing for PWM control over the "on" time of the magnetic field. By increasing $d[n]$ more heat is delivered to the gel over one switch period, and its temperature rises. On the other hand, decreasing $d[n]$ permits the ambient cooling from the circulating water in the outer jacket to lower the temperature of the gel. The closed-loop controller can, therefore, modulate the value of $d[n]$ to drive the gel position in either direction to minimize the position error.

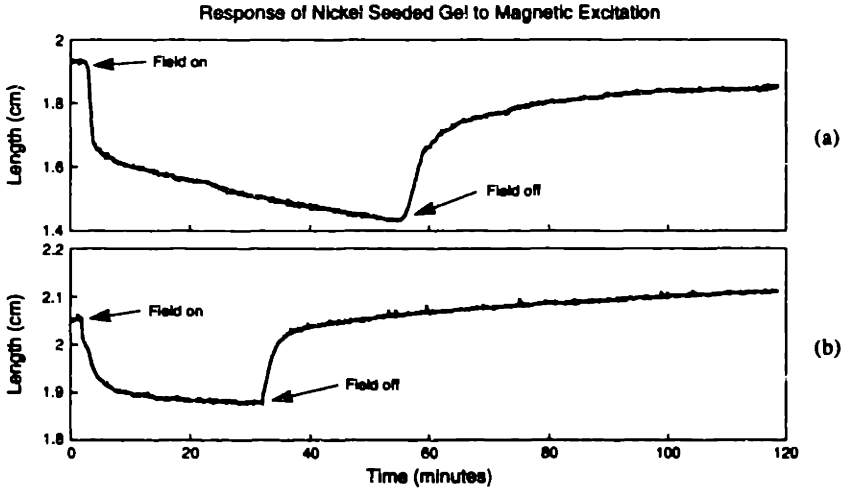



Figure 10.10: Open-loop response. (a) Nickel seeded gel. (b) Ferrofluid gel.

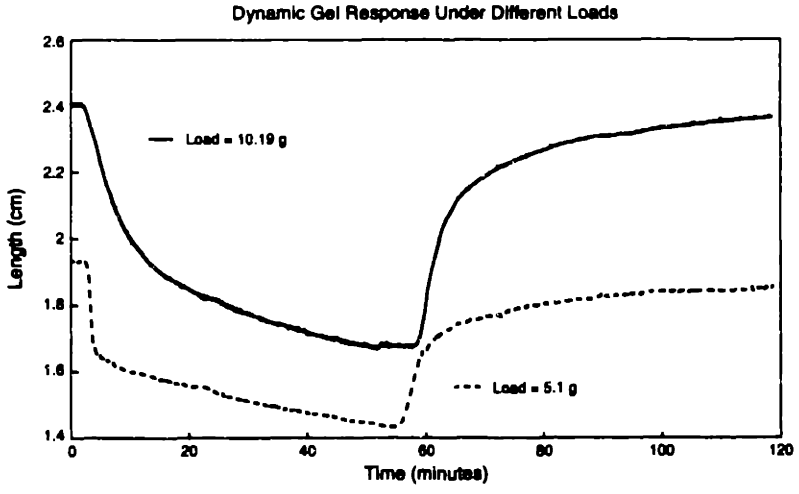



Figure 10.11: Magnetic gel response under different loading conditions.

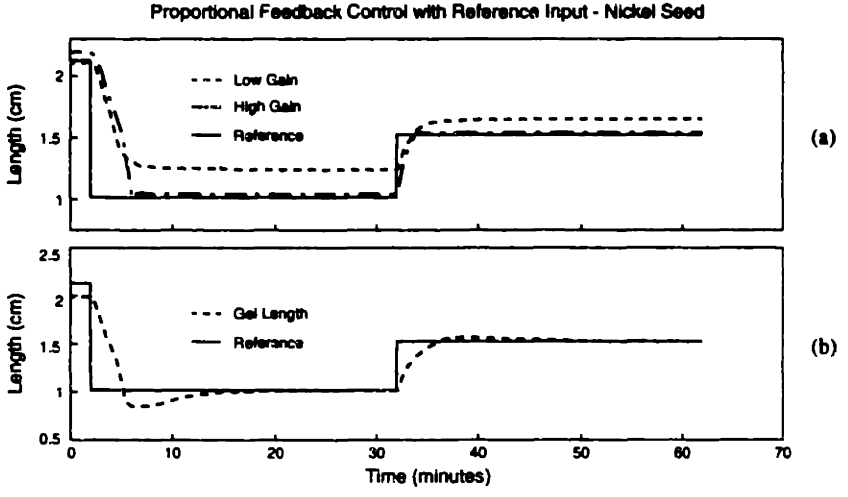



Figure 10.12: Closed-loop response of the nickel-seeded gel. (a) Proportional feedback. (b) Proportional-accumulator feedback.

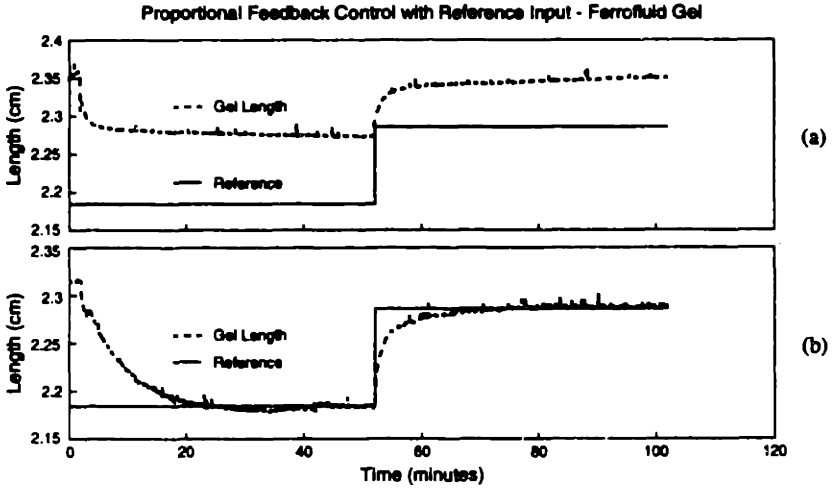



Figure 10.13: Closed-loop response of the ferrofluid gel. (a) Proportional feedback. (b) Proportional-accumulator feedback.

A second-order transfer function from the gel temperature $T(t)$ to position $x(t)$ has been shown to be a reasonable model for the kinetics of small changes in the volume of cylindrical NIPA/PVA gels under constant mechanical load [75, 76]. The model has two real poles and one zero as follows:

$$\frac{X(s)}{T(s)} = \frac{b_1 s + b_2}{(s + a_1)(s + a_2)}. \tag{10.14}$$

If the effects of the nickel particles and the ferrofluid solvent on the dynamics of the gel are ignored, this model can be used to describe the NIPA/PVA mag-gels. This assumption is justified by qualitative analysis of the open-loop experiments, which indicate that the general two-time constant behavior of the gel is unchanged by the ferromagnetic seed particles. Therefore, for the purpose of control design, we assumed the transfer function in (10.14) as the form of a model for the mag-gels. Methods for identifying the coefficients of this model are discussed in [75].

The dynamics of both the power electronics and the temperature diffusion from the seed targets to the gel were ignored because these dynamics are faster than the dominant gel kinetics. This approximation is valid only because of the large size and accompanying slow response of the polymer gels used here. With smaller faster gels, the thermal dynamics should not be ignored. Models that incorporate all of the relevant dynamics for different time-scale separations are under development [75].

Two discrete-time controllers were implemented and tested in MATLAB: a proportional controller and a proportional-accumulator or proportional-"integral" controller. The compensation was designed to yield a stable closed-loop system with good tracking performance. The results are illustrated in Figures 10.12 and 10.13. Rising and falling step references were provided as trajectories to the closed-loop system. Figure 10.12 plots the closed-loop responses of the nickel-seeded gel, and Figure 10.13 plots the responses of the ferrofluid gel. In both cases a proportional-accumulator controller yields reasonable transient and tracking performance with zero steady-state error. However, the results here are preliminary. Naturally, any fixed control strategy must be robust so that it can tolerate the effects of mechanical loading on the gel model. Future work is expected to improve the closed-loop performance, perhaps by incorporating the adaptive control techniques dis-

cussed in Part I of this thesis. A further discussion of control design for polymer gel systems can be found in [75].

## 10.6 Discussion

Polymer gels are a unique material with a tremendous range of potential applications. Inductive coupling greatly extends this range by providing a non-contact means to supply energy to phase-transition polymer gels. The resulting magnetically-activated polymer gels respond to heat induced within the gel through induction heating. The variety of seeding methods provides the opportunity to optimize gel performance, chemical composition, and form factor for specific applications.

Gels seeded with Methods 1 and 2 are being considered for application as *in vivo* drug delivery systems, actuators (synthetic muscles), and multi-part industrial chemical delivery systems (where the gel is loaded with an appropriate chemical or medicinal solvent). Gels seeded with Method 3 are being considered for the creation of fluids that exhibit a variable viscosity in response to an applied magnetic field. For example, we have loaded ferrofluids with millimeter- to micron-sized gel beads. When the beads are small, they occupy a small fraction of the solution volume, and the viscosity of the solution is essentially that of the surrounding ferrofluid. When the beads swell, they occupy a substantial part of the solution volume, and the overall solution exhibits a higher viscosity. With magnetic triggering, these fluids could be used to create remotely activated clutching mechanisms, vibration dampers, and molding systems.

## 10.7 Summary

Part III of this thesis has described the development of inductively-coupled polymer-gel actuators. Three different seeding methods have been described for fabricating mag-gels, and three prototype power-electronic drives were built and used to demonstrate the techniques. The prototypes utilized high-frequency circuit topologies, which were specifically adapted to the poor power factor and large apparent power required for the application. Test apparatus used to characterize the performance of the mag-gels was described, and measured results were presented. Finally, closed-loop position control was described to demonstrate the gels potential as a controllable linear actuator.

# Chapter 11

# Conclusions

## 11.1 Summary

This thesis has investigated inductive coupling as a general approach to power transfer for electromechanical systems. The focus has been on three different applications: single-stage inductively-coupled power transfer, multi-stage inductively-coupled power transfer, and an inductively-coupled polymer-gel actuator. These applications span a spectrum of engineering challenges relevant to inductively-coupled power transfer. Their presentation has illustrated a range of power-electronic drives and control strategies appropriate for inductively-coupled electromechanical systems. The inductively-coupled systems that have been considered are capable of transferring power from the milliwatt to the kilowatt range, and control strategies have been developed which operate reliably and with demonstrable stability in the face of large-signal changes in the operating parameters.

The thesis presentation was divided into three parts. Part I developed in detail a single-stage inductively-coupled system for electric-vehicle battery charging. Flexible power electronics have been designed, constructed, and assembled into two prototype systems: a 1.5-kW unidirectional system and a 600-W bidirectional system. The prototype systems have demonstrated a number of desirable features such as unity power factor, bidirectional power flow, active ripple cancellation, and line-frequency inverter operation. In addition, efficiency gains are evident upon direct comparison with existing systems. The 1.5-kW unidirectional prototype charger has demonstrated an overall efficiency of 89% at 1000 W. In comparison, the portable MAGNE-CHARGE inductive charger has an 85% efficiency at 1.2 kW [15], and an alternative 1.7-kW design in [12] has an 83% efficiency at 800 W. The $4-6\%$ gain in efficiency represents a significant reduction in conversion losses. This

reduction can be attributed to topological enhancements in design and control. An interleaved PFC stage has been presented with the potential to cut conversion losses in half. Also, by moving the control action to the PFC stage, the DC/DC stage can be operated at a fixed frequency and unity duty ratio, substantially reducing conduction and rectifier turn-off losses.

Both hardware prototypes in Part I were designed for microprocessor control. This allowed flexible digital control algorithms to be developed. Part I has presented digital control algorithms for the EV battery charging application as well as large-signal linear control algorithms, which form the basis for a general multirate control technique. Furthermore, adaptive digital control and estimation algorithms have been described and demonstrated experimentally. These "self-tuning" control techniques combined with the advantages of non-contact inductive power transfer greatly expand the range of potential real-world applications for inductively-coupled systems.

Part II of this thesis has described potential architectures for multi-stage inductively-coupled systems. These systems are capable of supplying power along an inductively-coupled AC bus, where "pickups" or power taps can be placed at any point along the length of the system. Two possible architectures have been studied: voltage-fed power transfer and current-fed power transfer. Through mathematical analysis and computer simulation, relevant design issues and trade-offs for both architectures have been revealed. The results should serve as design guidelines for future experimental work in the area. The context of segmented robotic manipulators was considered as a potential application. Inductively-coupled joints for this application, in particular those with two degrees of freedom, have also been presented.

Part III of this thesis has introduced a new inductively-coupled linear actuator based on polymer gels. This actuator directly combines electrical, thermal, and mechanical interactions in a single inductively-coupled system. New techniques have been presented for supplying activation energy to phase-transition polymer gels. Energy is supplied through the induction heating of a seed material within the gel. Three different gel seeding methods have been described, and three prototype power-electronic drives have been built and used to demonstrate the techniques. The circuit topologies selected were specifically

adapted to the poor coupling and large apparent power requirements of the application. The largest prototype power-electronic drive has demonstrated an apparent power capability of 15.3 kVA while dissipating only 135.2 W of real power.

## 11.2 Future Work

Part I of this thesis has presented a number of prototype circuit topologies. However, this work has not taken advantage of some of the newest power semiconductor technologies. For example, recent advances in high-performance IGBT switches and integrated power semiconductors—where as many as six switching devices are combined in a single package—offer the potential to simultaneously improve the performance and reduce the parts count of interleaved converters. The newest IGBT power devices demonstrate lower losses than traditional MOSFETs when operated at moderate frequencies (less than approximately 50 kHz). Interleaved conversion techniques guarantee moderate switching frequencies within individual switching cells without sacrificing the net converter performance. Future experimental work should capitalize on advantages IGBTs offer.

The discussion of multi-stage inductively-coupled systems in Part II provided general conclusions based on mathematical analysis and computer simulation. The results were intended only as design aids. Future work should refine these results and include experimental verification of the findings. Furthermore, the discussion in Part II suggested that power *and* information signals could be passed across an inductive coupling, completely eliminating the need for flexible wiring in a system. Although this thesis focused only on power transfer, techniques for the simultaneous transmission of power and information have been suggested in the literature [23, 53]. However, the results are preliminary, and significant problems that arise with multi-stage inductively-coupled architectures have yet to be solved satisfactorily. In particular, the information signal is typically absorbed by the load after crossing a single stage. An inelegant solution to this problem is to repeat the transmission at each stage. We suspect that an appropriate encoding technique might bypass the problem and allow direct communication between any two points on a multi-stage inductively-coupled bus. Simultaneous communications is another area for future work.

The inductively-coupled polymer gel actuators described in Part III are still an active area of development. Future work needs to focus on further improving the efficiency of the power-electronic drives so that their power consumption and overall size may be reduced. Substantial work is also needed to improve gel fabrication techniques so that the strength and speed of the gel are maximized. In addition, more complex actuator configurations need to be considered. The use of multiple gels in a push-pull configuration has been suggested in order to speed the dynamic response of the combined system. This technique might help alleviate the effects of performance bottlenecks such as the available ambient cooling rate.

# Appendix A

# Design Equations

This appendix provides specific equations and analyses that were used during the design of the prototype inductively-coupled hardware. This appendix supplements the discussions of the prototype hardware provided in Chapters 3 and 4.

## A.1 Boost Converter Design Equations

A high-power-factor boost pre-regulator performs two functions. First, it provides input power-factor correction so that the supply operates with near unity power factor (UPF). Second, it rectifies the AC utility voltage to produce a high-voltage DC bus. A schematic of the basic boost converter architecture was provided in Figure 3.5. To achieve UPF at the input, the input voltage and current must be in phase and of the same shape. Assuming the utility voltage is a sinusoid at 60 Hz, and ignoring current ripple at the switching frequency, the input voltage and current for UPF operation are

$$v_{ac}(t) = V_{ac}\sin(2\pi f_L t) \qquad i_{ac}(t) = I_{ac}\sin(2\pi f_L t)\,. \tag{A.1}$$

where $V_{ac}$ and $I_{ac}$ are the peak input voltage and current, and $f_L$ is the line frequency.

The relationships in (A.1) for UPF operation are achieved in practice by appropriately modulating the duty ratio $d(t)$ of the boost converter switch (or switches in the interleaved case). Formulas for $d(t)$ during UPF operation are provided in [79]. For a single-cell boost converter operating in CCM, the switch duty ratio must be

$$\text{(CCM)} \qquad d(t) = 1 - \frac{V_{ac}}{V_{bst}}\left|\sin(2\pi f_L t)\right| \tag{A.2}$$

where $V_{bst}$ is the DC level of the output voltage. The absolute value sign is a result of the input rectification. A similar, slightly more complex, relationship, for DCM operation is:

$$\text{(DCM)} \qquad d(t) = \sqrt{\frac{2L_1 f_s I_{ac}}{V_{ac}}\left(1 - \frac{V_{ac}}{V_{bst}}\left|\sin(2\pi f_L t)\right|\right)}. \tag{A.3}$$

In (A.3), the duty ratio depends on the boost inductance $L_1$, the switch frequency $f_s$, and the peak input current $I_{ac}$. Equation (A.3) is easily extended to the interleaved case, as follows:

$$(N\text{-cell DCM}) \qquad d(t) = \sqrt{\frac{2L_1 f_s I_{ac}}{NV_{ac}}\left(1 - \frac{V_{ac}}{V_{bst}}\left|\sin(2\pi f_L t)\right|\right)} \tag{A.4}$$

where $N$ is the number of interleaved switching cells.

It is important to note that the CCM duty ratio in (A.1) is independent of $I_{ac}$. This results from the fact that, for CCM operation, the transfer function relating $d(t)$ to the input current effectively contains an integrator (see Section A.2 for additional discussion). Therefore, any error in the duty ratio will cause the inductor current to deviate with an exponentially increasing error. This exponential growth makes it difficult to balance current equally among CCM switching cells in an interleaved converter. For example, if the same duty-ratio command $d(t)$ is used to generate the PWM patterns for a number of parallel stages, the slightest mismatch between cell inductances will cause the currents to unbalance. A solution to this problem is to employ multiple feedback controllers, which supply independent duty-ratio commands to the individual cells. This solution is costly.

If the converter is operated in DCM, the inductor current returns to zero before the end of each switching cycle. Thus, the inductor current each cycle is independent of its value during the previous cycle, and errors that result from cell mismatches do not accumulate. This eliminates the integral relationship between $d(t)$ and the inductor current in favor of the functional relationship in (A.4). A single duty-ratio command can now be used to drive parallel interleaved switching cells in DCM, and current sharing between the cells is guaranteed. The matching between cell currents will be roughly proportional to the matching between cell inductances.

Discontinuous conduction is advantageous for the reasons just stated and because it can improve the efficiency of the converter as described in Chapter 3. Thus, it is important to know the boundaries between CCM and DCM operation. These boundaries were described analytically in [79]. To ensure DCM operation, the cell inductance $L_1$ must satisfy two constraints:

$$L_{1,max1} < \frac{NV_{ac,min}^2}{4\langle P_{ac,max}\rangle f_s}\left(1 - \frac{V_{ac,min}}{V_{bst,min}}\right) \tag{A.5}$$

and

$$L_{1,max2} < \frac{NV_{ac,max}^2}{4\langle P_{ac,max}\rangle f_s}\left(1 - \frac{V_{ac,max}}{V_{bst,min}}\right) \tag{A.6}$$

where $\langle P_{ac}\rangle$ denotes the time-averaged input power, and the subscripts $min$ and $max$ denote minimum and maximum values, respectively. Equations (A.5) and (A.6) can be used to find $L_1$ provided that min and max values of $V_{bst}$, $V_{ac}$, and $P_{ac}$ are known. Alternatively, when $L_1$ is known, DCM operation is ensured if $V_{bst}$ satisfies (A.7) over the full range of $V_{ac}$.

$$V_{bst} > \frac{NV_{ac}^3}{NV_{ac}^2 - 4L_1\langle P_{ac,max}\rangle f_s} \tag{A.7}$$

Equation (A.7) can be used to determine the DCM operating range for the eight-cell interleaved boost converter described in Chapter 3. The appropriate values for the prototype are $N=8$, $L_1=550\ \mu$H, $P_{ac}=1.5$ kW, and $f_s=25$ kHz. The range for $V_{ac}$ is nominally $100-140\ V_{rms}$ ($141-198$ V peak). Substituting into (A.7) tells us that $V_{bst}$ must be greater than 292 V in order to guarantee DCM operation. This does not imply that CCM operation will result if (A.7) is not satisfied. However, mixed-mode operation is likely, with CCM operation beginning near the peak input voltage and current.

## A.2 Boost/Buck Averaged Current-Mode Control

Inner-current-loop compensation of the first-stage PFC circuits was accomplished using averaged current-mode control. This technique has proven to provide precise control over the inductor current during the boost and buck operating modes of both the unidirectional

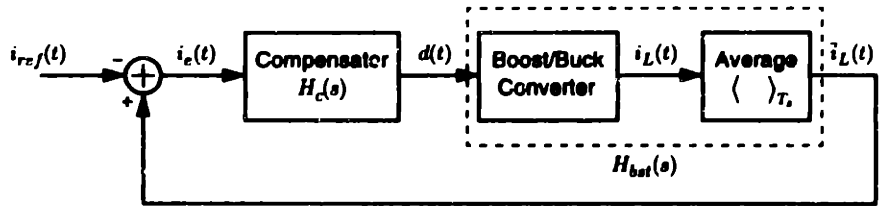



**Figure A.1:** Block diagram of the inner-current-loop controller.

and bidirectional prototypes. The analog circuit that performs this compensation was described briefly in Chapter 3, subsection 3.2.3. A block diagram of the control structure appears in Figure A.1. Feedback forces the time-averaged inductor current $\bar{i}_L(t)$ to track a desired reference $i_{ref}(t)$. For UPF operation this reference is set to $i_{ref}(t) = k(t)v_{rect}(t)$, where $v_{rect}(t) = |v_{ac}(t)|$.

For the purpose of developing the current-loop compensator, an averaged-circuit model is used to derive a transfer function relationship between $\bar{i}_L(t)$ and $d(t)$ for the boost converter in CCM. The method is described in [21]. The derivation relies on an assumption that the output voltage of a well-designed system exhibits small variations. So $\bar{v}_{bst}(t) = V_{bst} + \hat{v}_{bst}(t)$, where $|\hat{v}_{bst}(t)| \ll |V_{bst}|$. Although, this assumption is technically violated by large-signal changes in $\bar{v}_{bst}(t)$, if these changes are slow with respect to the inner-loop bandwidth, their effect is insignificant. Figure A.2 shows the averaged-circuit model of a boost cell operating in CCM. The variables $\bar{v}_{rect}(t)$, $\bar{i}_L(t)$, and $\bar{v}_{bst}(t)$ are time-averaged over one switch period $T_s$, and $d'(t) = (1 - d(t))$. A difference equation relating the time-averaged variables is obtained using Kirchhoff's voltage law:

$$L\frac{d}{dt}\bar{i}_L(t) = \bar{v}_{rect}(t) - d'(t)\bar{v}_{bst}(t)\,. \tag{A.8}$$

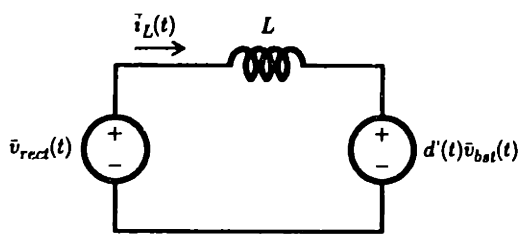



**Figure A.2:** Boost converter switched-circuit average model.

Linearizing the output voltage $\bar{v}_{bst}(t)$ in (A.8) and cancelling the small-signal terms yields:

$$L\frac{d}{dt}\bar{i}_L(t) = \bar{v}_{rect}(t) - d(t)V_{bst}. \tag{A.9}$$

The time-averaged control-to-current transfer function for CCM operation is found by setting the independent inputs other than $d(t)$ to zero, and then solving for $\bar{i}_L(t)$:

$$H_{bst}(s) = \frac{I_L(s)}{D(s)} = \frac{V_{bst}}{sL}. \tag{A.10}$$

Although this result substantially simplifies the true time-varying non-linear behavior of the boost converter, it is commonly used as a basis for developing inner-current-loop compensation [111].

If the boost cell operates in DCM, the inductor current resets each switching cycle and $d(t)$ and $\bar{i}_L(t)$ are relating by

$$\bar{i}_L(t) = d^2(t)\frac{T_s}{2L}\frac{\bar{v}_{rect}(t)}{(1 - \bar{v}_{rect}(t)/V_{bst})}, \tag{A.11}$$

which does not contain dynamics. Although (A.11) is non-linear, its may be interpreted as a piecewise linear gain that varies on the slow, line-frequency, time scale of $\bar{v}_{rect}(t)$. Since the CCM case is more difficult from a control perspective, it is used for the purpose of designing a compensator. The low-frequency gain of the resulting compensator will be more than adequate to handle the simpler DCM case. It is important to note that, when the power flow is reversed, the resulting buck-mode relationships are nearly identical. However, $\bar{i}_L(t)$ flows in the opposite direction so the roles of $V_{bst}$ and $\bar{v}_{rect}(t)$ are reversed. Since $V_{bst} \geq \bar{v}_{rect}(t)$, a controller designed for the boost mode is also suitable for the buck mode.

A transfer function $H_c(s)$ for the inner-loop compensator is given by

$$H_c(s) = \frac{D(s)}{I_e(s)} = -\frac{g_c(s+\beta)}{s(s+\alpha)} \tag{A.12}$$

where

$$g_c = \frac{C_1 + C_2}{C_1 C_2 R_3}, \qquad \beta = \frac{1}{R_1(C_1 + C_2)}, \qquad \text{and} \qquad \alpha = \frac{1}{R_1 C_1}. \tag{A.13}$$

The variables $C_1$, $C_2$, $R_1$, and $R_3$ refer to the circuit elements in Figure 3.8. If (A.12) is combined with (A.10), the forward-loop transfer function for the inner-current loop becomes

$$H_{lp}(s) = -\left(\frac{g_c V_{bst}}{L}\right)\frac{(s+\beta)}{s^2(s+\alpha)}, \tag{A.14}$$

and the resulting closed-loop transfer function is given by

$$H_{cl}(s) = \frac{k_{lp}(s+\beta)}{s^3 + \alpha s^2 + k_{lp} s + k_{lp}\beta} \tag{A.15}$$

where $k_{lp} = g_c V_{bst}/L$.

To complete the compensator design, we must select $\alpha$, $\beta$, and $g_c$ for the compensator. This is easier if we first pick a target for the loop-gain crossover frequency $f_c$. The magnitude of the loop gain is critical. If it is too high, it can cause subharmonic oscillations in the PWM controller as described in [111]. The solution is to constrain the loop gain so that the downslope of the inductor current does not exceed the downslope of the ramp used to generate the PWM waveform. The appropriate constraint is given by

$$G_{max} = \frac{f_s L}{V_{bst}} \tag{A.16}$$

where $f_s$ is the cell switching frequency. The crossover frequency is now found by multiplying (A.16) by (A.10), evaluated at $f_c$. The result is set to unity and solved for $f_c$, as follows:

$$G_{max} H_{bst}(2\pi f_c) = \frac{f_s L}{V_{bst}} \frac{V_{bst}}{2\pi f_c L} = \frac{f_s}{2\pi f_c} = 1 \tag{A.17}$$

Equation (A.17) yields $f_c = f_s/2\pi$.

The pole $\alpha$ and zero $\beta$ are selected above and below $f_c$, respectively, to increase the phase margin at crossover and to provide high-frequency roll-off for filtering out switch-frequency noise. The frequencies were selected as follows:

$$\frac{\alpha}{2\pi} = 3f_c \approx \frac{f_s}{2}, \qquad \text{and} \qquad \frac{\beta}{2\pi} = \frac{f_c}{2} \approx \frac{f_s}{12}. \tag{A.18}$$

A large loop gain at low frequency is guaranteed by the integrator in $H_c(s)$, even if the converter operates DCM. Finally, the compensator gain $g_c$ is found according to

$$g_c = G_{max} \frac{1}{R_1(C_1 \| C_2)}, \tag{A.19}$$

which is equivalent to setting the ratio of $R_1$ and $R_3$ in the compensator to

$$\frac{R_1}{R_3} = G_{max}. \tag{A.20}$$

The above procedure was used to compute component values for the experimental prototypes. The design parameters for the experimental 1.5-kW interleaved prototype were $f_s = 25$ kHz, $L = 550$ $\mu$H, and $V_{bst} = 390$ V (in the worst case). The compensator is designed for the parameters of a single switching cell despite the fact that the converter is interleaved. The final component values for this prototype were $C_1 = 470$ pF, $C_2 = 2.2$ nF, and $R_1 = 30$ k$\Omega$. These values correspond to $C_{62}$, $C_{61}$, and $R_{39}$, respectively, on the schematic in Section C.2. The component $R_3$ from Figure 3.8 does not have a direct counterpart in the prototype circuit. Instead, its effective value is

$$R_{3\ \text{effective}} = \frac{R_{41} V_{pwm}}{G_{sns}} \tag{A.21}$$

where $V_{pwm}$ is the amplitude of the PWM ramp generator and $G_{sns}$ is the gain of the inductor current sense circuitry. For the 1.5-kW interleaved prototype the corresponding values are $V_{pwm} = 12$ V and $G_{sns} = 0.425$. The effective $R_3$ is approximately 1081 k$\Omega$. The resulting forward-loop pole/zero locations are

$$\text{(pole)} \quad \frac{\alpha}{2\pi} \approx 12 \text{ kHz} \qquad \text{and} \qquad \text{(zero)} \quad \frac{\beta}{2\pi} \approx 2 \text{ kHz}. \tag{A.22}$$

Similarly, for the 600-W bidirectional prototype $f_s = 25$ kHz, $L = 2.0$ mH, and $V_{bst} = 390$ V. The resulting component values for the inner-loop compensator are $C_{62} = 47$ pF, $C_{61} = 470$ pF, $R_{39} = 150$ k$\Omega$, and $R_{41} = 38.3$ k$\Omega$ ($R_{3\ \text{effective}} = 1081$ k$\Omega$). The effectiveness of both compensators for PFC was demonstrated in Chapters 3 and 4.

## A.3 Interleaved Ripple Cancellation

The amplitude of the input-current ripple to an interleaved converter is non-trivially related to the number of switching cells, input voltage, input current, duty ratio, and output voltage of the converter. In order to evaluate the performance of the interleaved prototype, it was necessary to estimate the expected effects of interleaving. Several approximate relationships, which use both frequency-domain and time-domain techniques, have been described in the literature. The focus of these techniques is to provide insight into the net ripple behavior as well as to reduce the computational complexity. A "composite interleaving function," described in [79], is one such approximate relation.

For the purpose of designing and evaluating the prototype, it was decided that direct time-domain simulation provided more accurate results. A MATLAB script was written for this purpose, and its source-code listing (`RIPPLE.M`) is provided in Appendix E. The script constructs the idealized current profile for an individual switching cell and then replicates this pattern, shifted in time, for the remaining switching cells. The individual cell currents are summed to yield the net input current. The peak-to-peak ripple amplitude is then estimated from the composite waveform. This process is repeated over a range of input voltage and current levels to simulate the behavior of the converter during PFC.

Figure A.3 (a) depicts simulated ripple waveforms for an eight-cell interleaved boost converter. The simulation parameters were set to reflect those of the 1.5-kW prototype converter. The figure illustrates the cell current in each of the eight interleaved stages. Notice that each cell operates in DCM as desired. The peak current within the $a^{\text{th}}$ switching cell can be shown to be constrained by

$$I_{La} \le \frac{V_{bst}}{4 f_s L_{1a}}. \tag{A.23}$$

Surprisingly, the same bound also constrains the peak-to-peak current ripple in a CCM switching cell [79]. The maximum peak-to-peak ripple at the input of an interleaved converter can be shown to be constrained by

$$I_{rip} \le \frac{V_{bst}}{4 N f_s L_1} \tag{A.24}$$

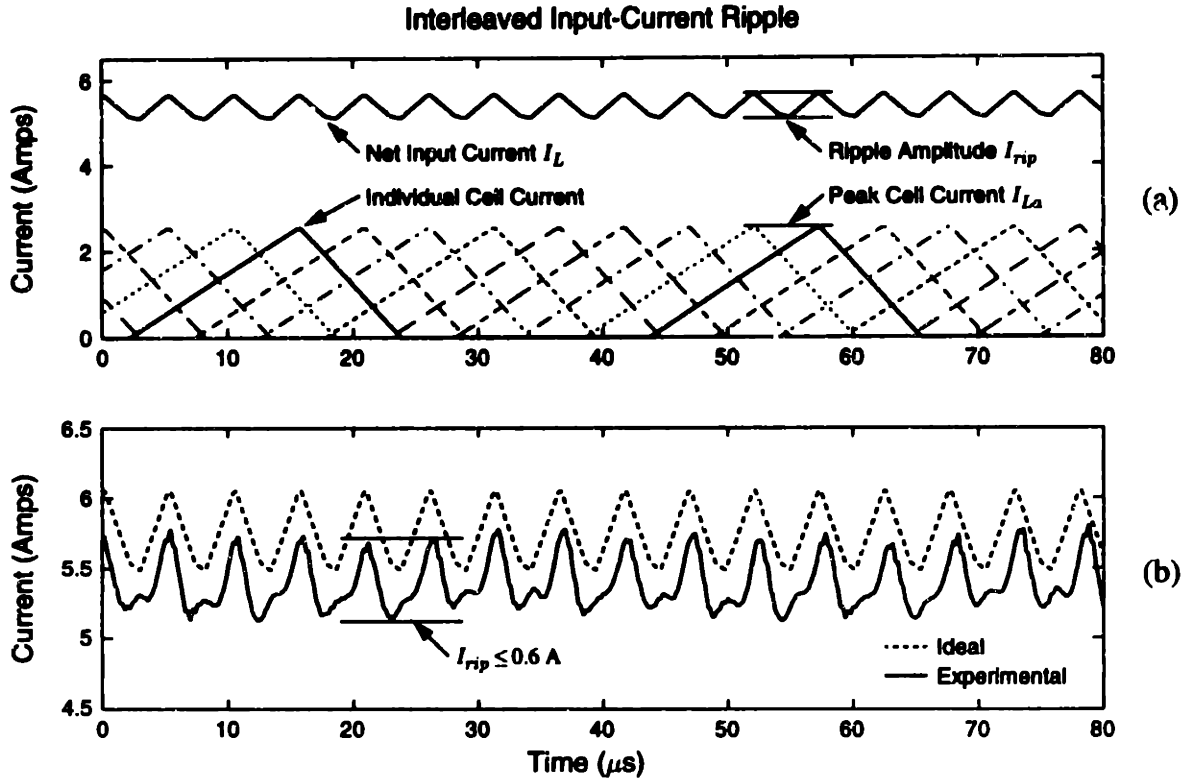



**Figure A.3:** (a) Idealized waveforms showing the net and individual cell currents. (b) Experimental input-current ripple. ($V_{ac}$= 100 $V_{DC}$, $I_{ac}$=5.4 A and $V_{bst}$=265 V.)

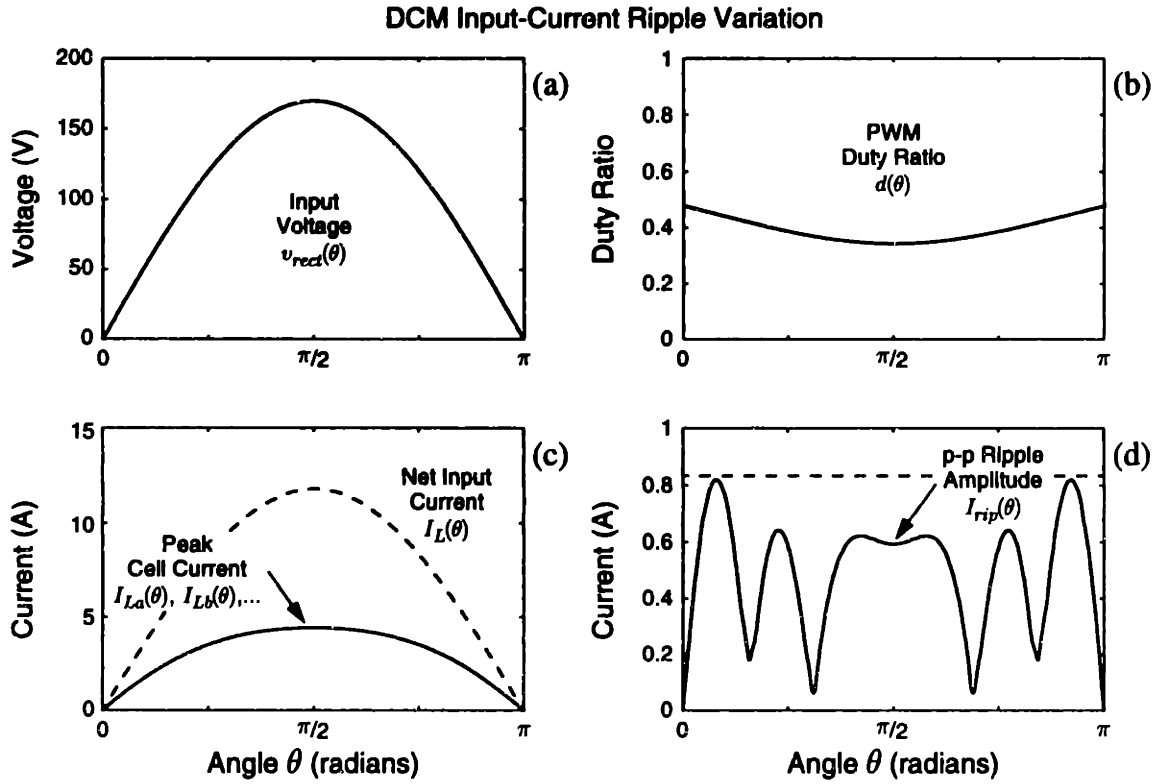



**Figure A.4:** Simulated current-ripple characteristics for the prototype interleaved boost converter. ($V_{ac}$=120 $V_{rms}$, $V_{bst}$=350 V, $P_{ac}$=1 kW and $N$=8.)

where $L_l$ is the inductance of an individual switching cell. Plugging in the operating parameters from Figure A.3 (a), the maximum peak-to-peak ripple in the net input current is approximately $I_{rip} = 0.6$ A. This value is compared against experimental data in Figure A.3 (b). The input EMI shunt filter was removed from the 1.5-kW prototype so that the ripple current could be observed. The dashed line in Figure A.3 (b) is simulated and the solid line is experimental. The experimental and simulated amplitudes match well. The irregular appearance of the experimental data can be attributed to filtering effects from parasitic circuit capacitances.

Figure A.4 shows the graphical output of `RIPPLE.M`. The $x$-axis in each plot is the phase angle $\theta$, in radians, of the input voltage $v_{rect}$. Graphs (a) and (b) plot the input voltage $v_{rect}$ and the duty-ratio command $d$, respectively. The variation in $d$ over the span of a half line-cycle produces the input current $I_L$ graphed in (c). Since $I_L$ is proportional to $v_{rect}$, the input power factor is unity. Also in (c) are the peak cell currents $I_{La,\ldots,h}$. The peak cell current is substantially smaller than $I_L$, however, the reduction is always less than the factor $N$. The peak-to-peak input-current ripple $I_{rip}$ is plotted in (d). The ripple amplitude exhibits a peculiar variation with the input phase angle. Nevertheless, the maximum amplitude is always constrained by the bound in (A.24). The bound is drawn as a dashed line in (d).

## A.4 AC-Inverter Switching Sequences

Chapter 4 briefly discussed a technique that uses binary switching sequences to synthesizing sinewaves in an AC inverter. The switching sequences are not subject to the fixed-frequency constraints of PWM waveforms. Consequently, it is possible to create low distortion sinewave approximations that use fewer switch transitions per cycle than a comparable PWM pattern. The switching sequences are broadly applicable in a number of systems, including AC inverters, motor drives, and synchronous detectors. In order to apply these switching sequences effectively, a development tool is needed so the designer can quickly and easily satisfy requirements such as the number of transitions, harmonic content, and amplitude (i.e., the ratio of ones to zeros).

The example 30-bit sequence shown earlier in Figure 4.10 appears simple; however, there are approximately 3,000 30-bit sequences with the same ratio of ones to zeros. Naturally, only a few of these sequences approximate a sinusoid with any accuracy, but finding them may be a challenge. One technique is to generate each and every sequence, compute their fast-Fourier-transforms (FFTs), and select only those with the lowest harmonic distortion. This technique has been used in published literature. For example, the 384-bit sequence shown in Figure A.5 was published in [62], where the term 'magic sinewave' was coined to describe the switching sequences. Figure A.5 plots the experimental output of the prototype boost-buck-inverter driving a 1-HP shop vacuum with a 384-bit magic sinewave. The inverter voltage and current appear in traces (a) and (b), respectively. The harmonic content of the voltage is plotted in Figure A.6. The magnitudes of 12 of the first 13 harmonics are less than 1% of the fundamental. While this reduction is significant, the output would still require substantial low-pass filtering in order to reduce the higher-harmonic distortion seen by the load.

The number of insignificant harmonics can be increased by developing longer magic sinewave sequence. However, as the length $N$ is increased, the number of possible sequences quickly gets out of hand. The number of $N$-point sequences with $E$ ones is given by

$$\eta = \frac{N!}{(N-E)!E!}. \tag{A.25}$$

Taking full advantage of the symmetry of a sinewave, it possible to completely characterize a magic sinewave of length $4N$ with only $N$ bits. Nonetheless, the number of combinations for the 384-bit sequence with 144 ones is still $\eta = 3.2\times10^{26}$ ($N=96$ and $E=36$ in (A.25)). If the length is increased to 1024 bits and 384 ones, the number of combinations increases dramatically to $\eta = 1.83\times10^{72}$. The number continues to grows exponentially with $N$ as the length increases. The direct FFT approach, is therefore computationally infeasible.

A colleague, Steve Shaw, suggested a "simulated annealing" algorithm in [96] to approximate the solution of this problem in sub-exponential time. A variant on his algorithm was coded into MATLAB, and the source listing for this code appears in

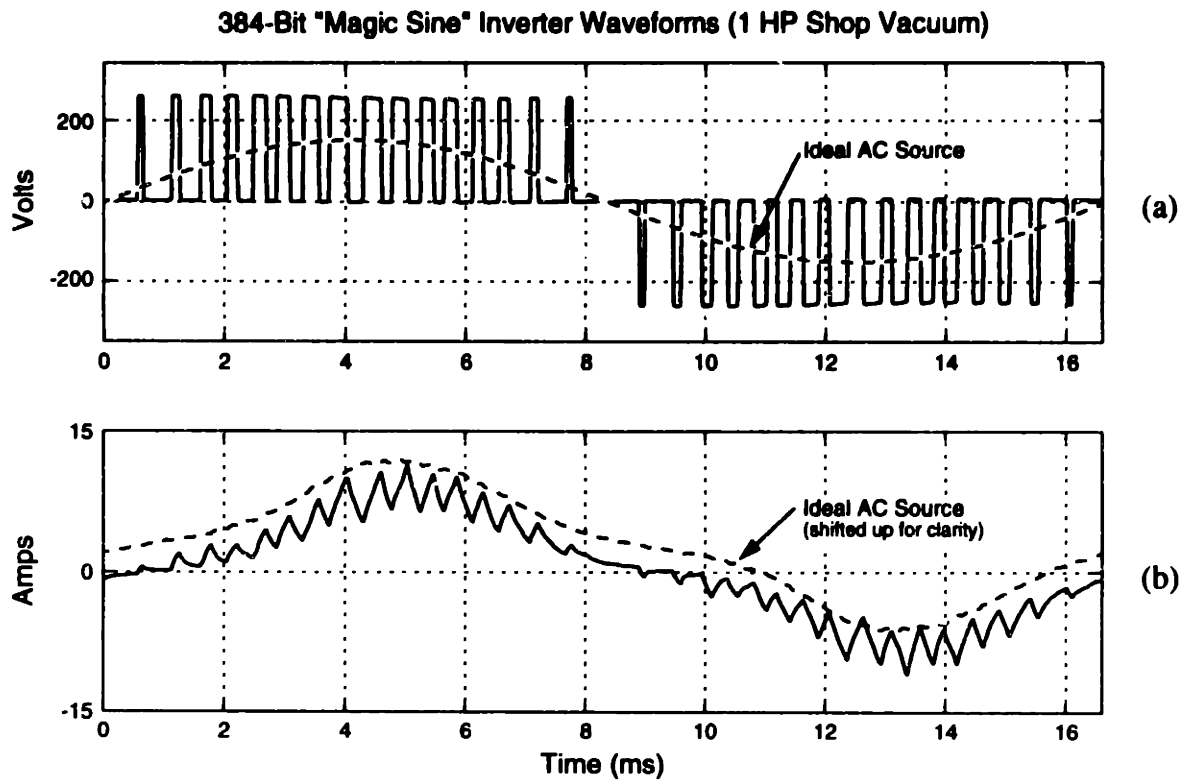



**Figure A.5:** Measured waveforms with the converter driving a 1-HP shop vacuum in inverter mode. (a) Inverter voltage. (b) Inverter current.

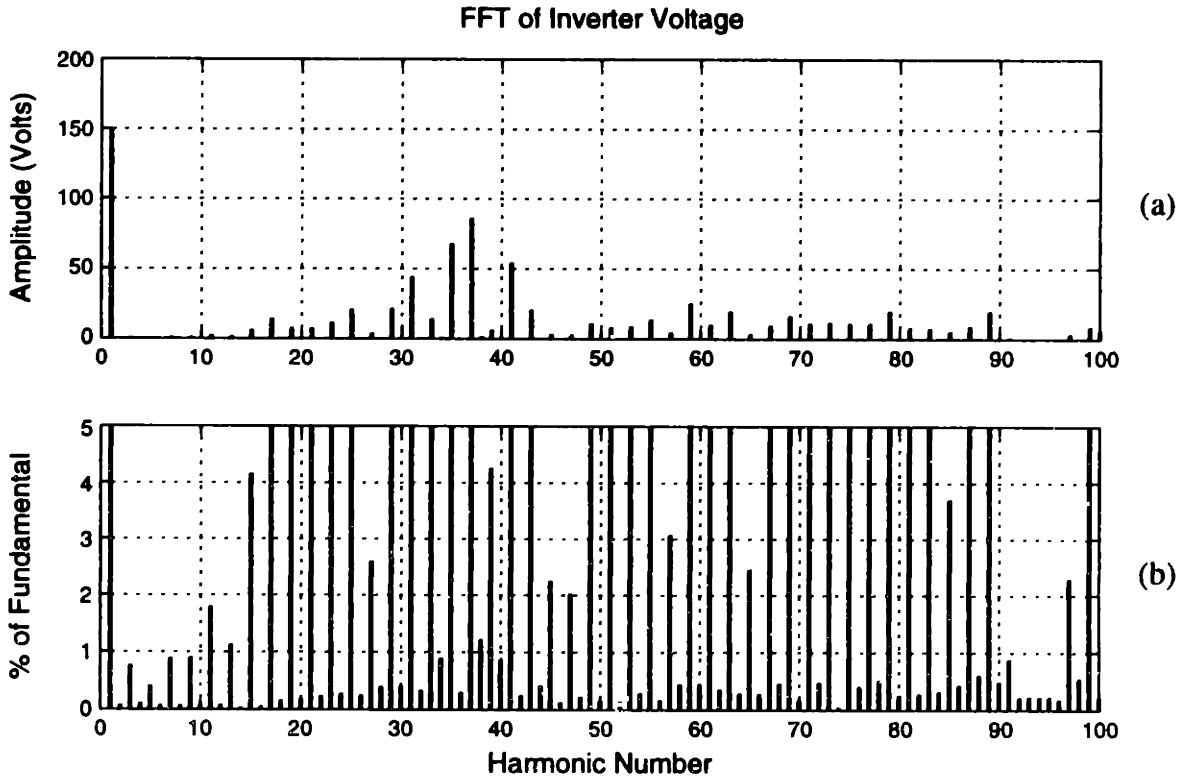



**Figure A.6:** Fourier transform of the inverter voltage. (a) FFT with amplitude in volts. (b) An expanded view showing harmonics as a percent of the fundamental.

Appendix E (see ANNEAL1.M). The algorithm attempts to minimize a loss function that penalizes both the harmonic content and the number of transitions in the switching sequence. The minimization is accomplished by randomly swapping zeros and ones in the sequence. After each swap, the new configuration is accepted or rejected using a probabilistic Metropolis criterion, which depends on a "temperature" variable $T$. The process begins by raising the temperature, increasing the number of random perturbations, until the movement in the loss function is largely random. The temperature is then cooled, favoring downhill movements in the loss function toward a minimum. The algorithm repeatedly mixes and then refines the solution by raising and lowering the temperature. The technique receives its name from the analogous physical process of annealing metals.

The script ANNEAL1.M was used to generate the 1024-bit sequence plotted in Figures 4.11 and 4.12. The script was configured to minimize the low-frequency harmonic content of the sequence, and the result was obtained in approximately 10 minutes on a Pentium PC. The time required to converge on a suitable result depends heavily on the bound applied to the harmonic content. Relaxing this bound will generally improve the convergence rate. The 1024-bit sequence in Figures 4.11 and 4.12 demonstrates significant harmonic reduction through the first 40 voltage harmonics. In a real-world application, sequences as long as 8192 bits may be practical. This could yield sequences with substantial harmonic reduction through approximately the first 300 harmonics. In general, the number of harmonics that can be effectively eliminated varies proportionally with the length of the sequence and the number of edge transitions.

# Appendix B

# AMP/M.I.T. Prototype Documentation

## B.1 Operating Instructions

# Operating the
# AMP/M.I.T.
# Prototype Inductively-Coupled Charger

Deron K. Jackson
S/W Rev. 5.1
11/15/96

### B.1.1. Description

The AMP/M.I.T. inductively-coupled charger is a prototype. This unit is designed to demonstrate power electronics and control techniques developed at M.I.T. for use in an inductively-coupled electric-vehicle battery charger. The *CHARGER* (silver case) contains a unity-power-factor boost converter, a high-frequency inverter, and digital control electronics. The simulated *VEHICLE* (blue case) contains a rectifier, a filter, and some communication electronics. In place of a battery, three 300-W halogen bulbs are used as a demonstration load. The lamps are conveniently mounted on a custom reflector inside the top of the *VEHICLE* case.

The prototype is controlled by an internal microcontroller. The user operates the unit from a control panel located on the *CHARGER*. The control panel is illustrated in Figure B.1. A series of menus allow the user to select various operating modes, control

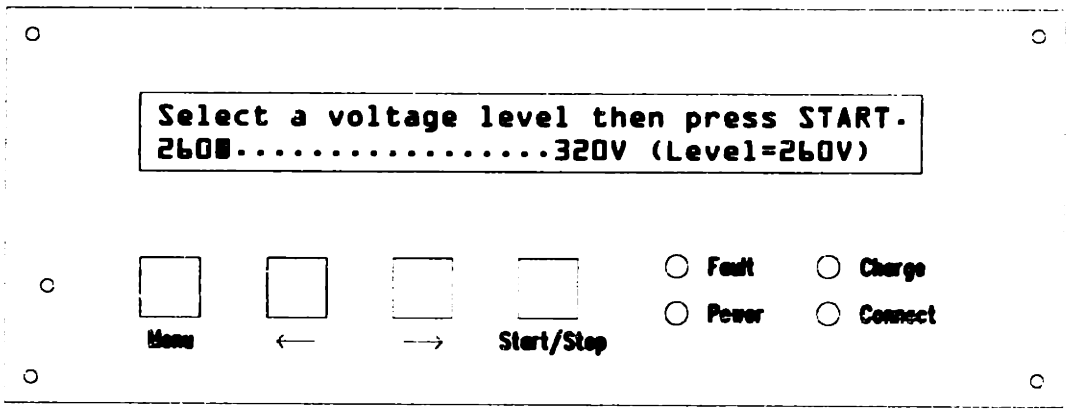



**Figure B.1:** Control panel for the prototype charger.

options, and voltage or current levels. Table B.1 lists the operating modes available with the current software revision. Additional or specific operating modes can be added quite easily through software. All control modes can be selected to act on the "boost output" or the "DC/DC output." Due to certain limitations of the prototype construction the "boost output" option yields better performance.

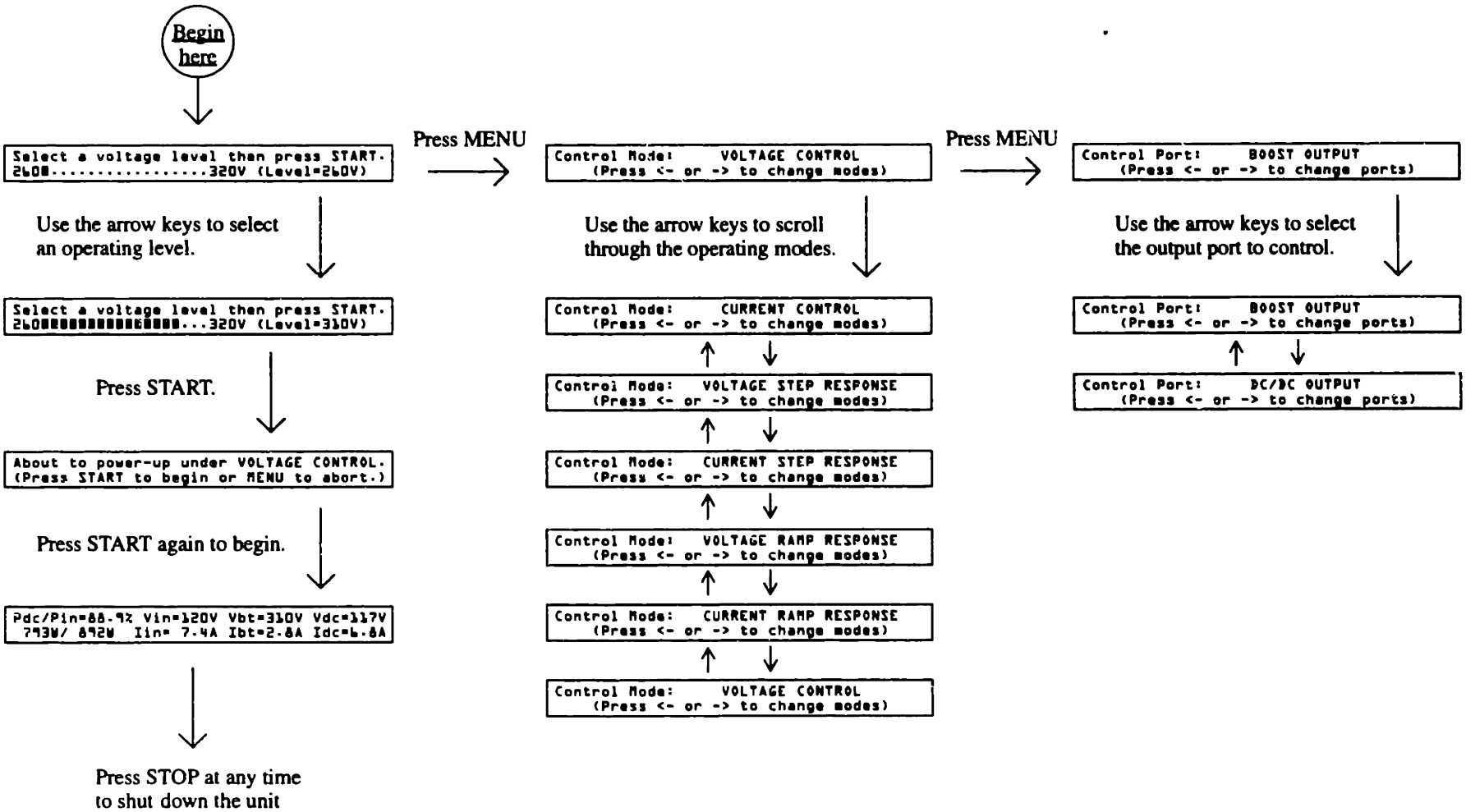
| Control Modes | Description |
|---|---|
| • Voltage Control | Constant Output Voltage |
| • Voltage Step Response | Voltage Square-Wave Output (Period = 5 seconds) |
| • Voltage Ramp Response | Voltage Sawtooth Output (Period = 5 seconds) |
| • Current Control | Constant Output Current |
| • Current Step Response | Current Square-Wave Output (Period = 5 seconds) |
| • Current Ramp Response | Current Sawtooth Output (Period = 5 seconds) |

**Table B.1:** Available operating modes.

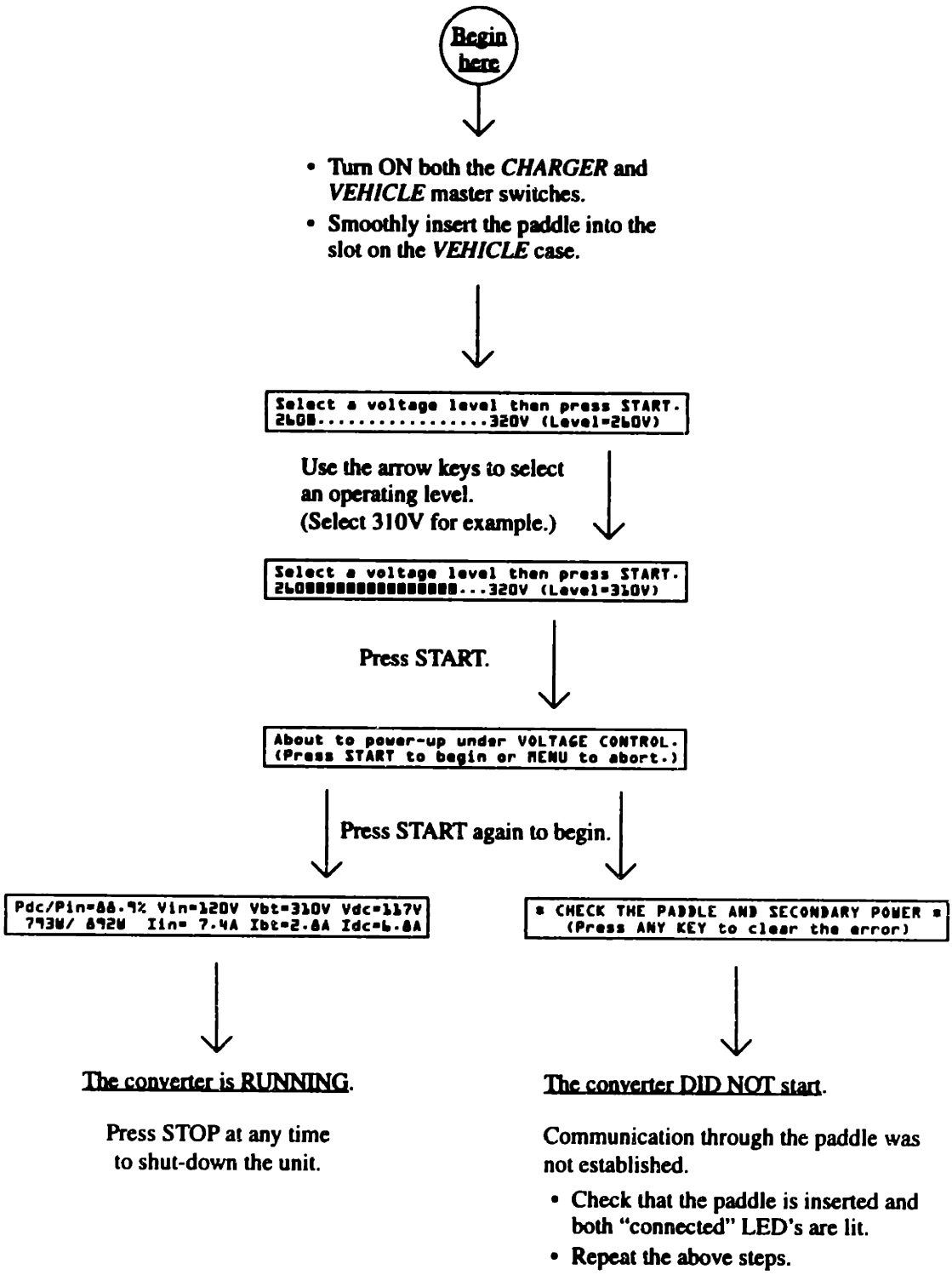
### B.1.2. Operation

The LCD screen on the prototype uses a series of menus to guide the user through the operating procedures. Therefore, detailed step-by-step instruction are not necessary. Instead, the following pages contain three flow charts which detail the most common operating steps. A fourth chart outlines the steps to be taken when the user is presented with an error screen.

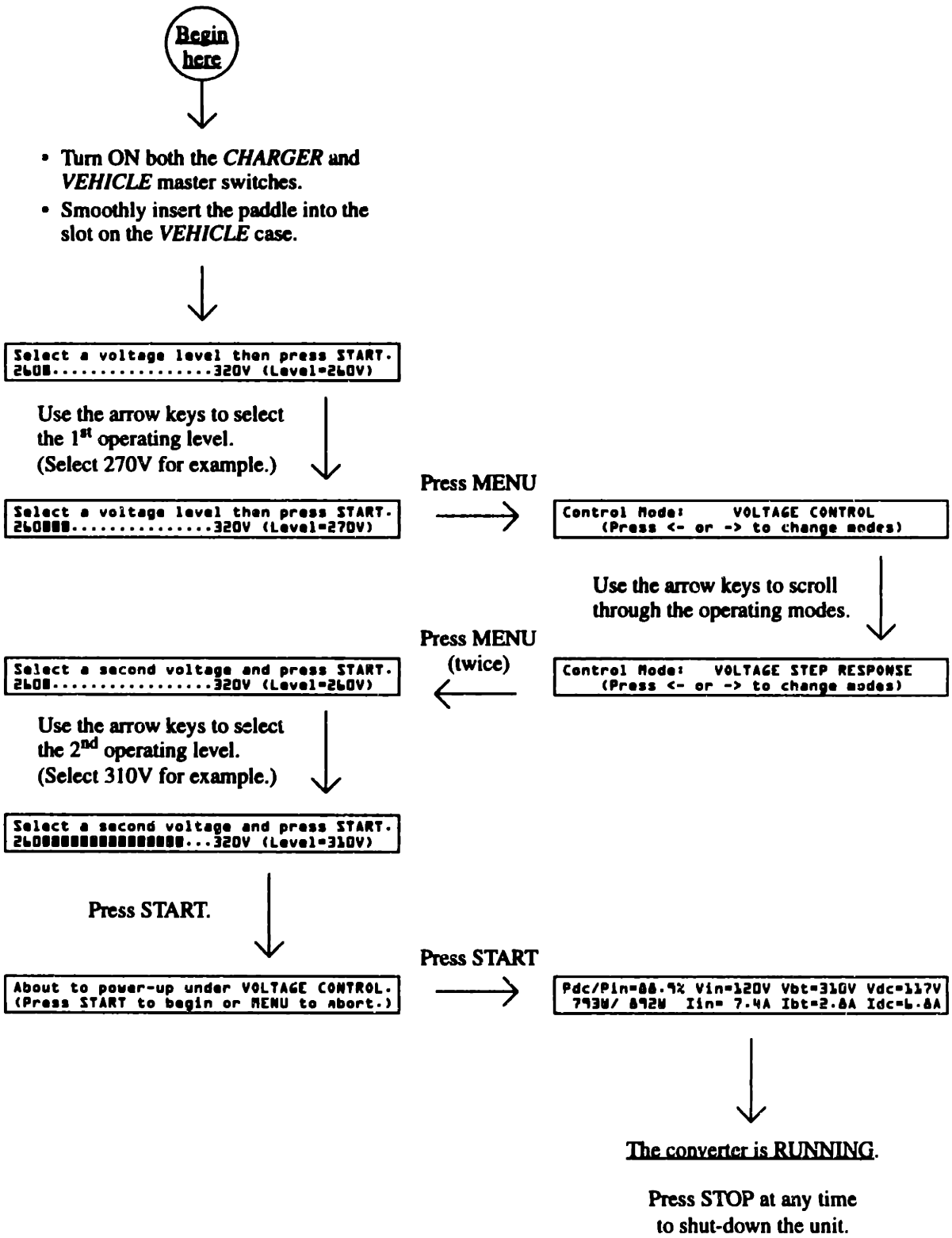
# Menu System Overview



# Typical Start-Up Procedure



# Voltage-Step Start-Up Procedure



# What to do in case of ERROR!

| ERROR Message | Problem/Solution |
|---|---|
| * CHECK THE PADDLE AND SECONDARY POWER *<br>(Press ANY KEY to clear the error) | PROBLEM: Communication through the paddle was not established.<br>SOLUTIONS: Try one of the following.<br>1. Check that the paddle is inserted and both "connected" LED's are lit.<br>2. Wipe clean the exposed fiber-optic ends on the paddle and vehicle receptacle.<br>3. The batteries in the vehicle (blue) case are low. Charge them using the supplied 48V charger. |
| **** ERROR #11: PLEASE CHECK PADDLE ****<br>(Press ANY KEY to clear the error) | PROBLEM: Communication through the paddle was unexpectedly disconnected.<br>SOLUTIONS: Try one of the following.<br>1. The paddle was removed during operation. Re-insert the paddle.<br>2. Wipe clean the exposed fiber-optic ends on the paddle and vehicle receptacle. |
| **** ERROR #1: Vin too HIGH (###V) *****<br>(Press ANY KEY to clear the error)<br>**** ERROR #2: Vin too LOW (###V) ******<br>(Press ANY KEY to clear the error)<br>**** ERROR #3: Iin too HIGH (#.#A) *****<br>(Press ANY KEY to clear the error)<br>*** ERROR #9: NO INPUT (Cycles = 0) ****<br>(Press ANY KEY to clear the error) | PROBLEM: AC input power has exceeded specified limits.<br>SOLUTION:<br>1. Move the unit to another outlet. |
| **** ERROR #4: Vbt too HIGH (###V) *****<br>(Press ANY KEY to clear the error)<br>**** ERROR #5: Ibt too HIGH (#.#A) *****<br>(Press ANY KEY to clear the error)<br>** ERROR #10: Ibt > Iset @ Vdc = ##V ***<br>(Press ANY KEY to clear the error) | PROBLEM: A voltage or current setting has been requested that exceeds preset operating limits.<br>SOLUTION:<br>1. Change the desired voltage/current setting and retry. |

## B.2 Optical Communications Link

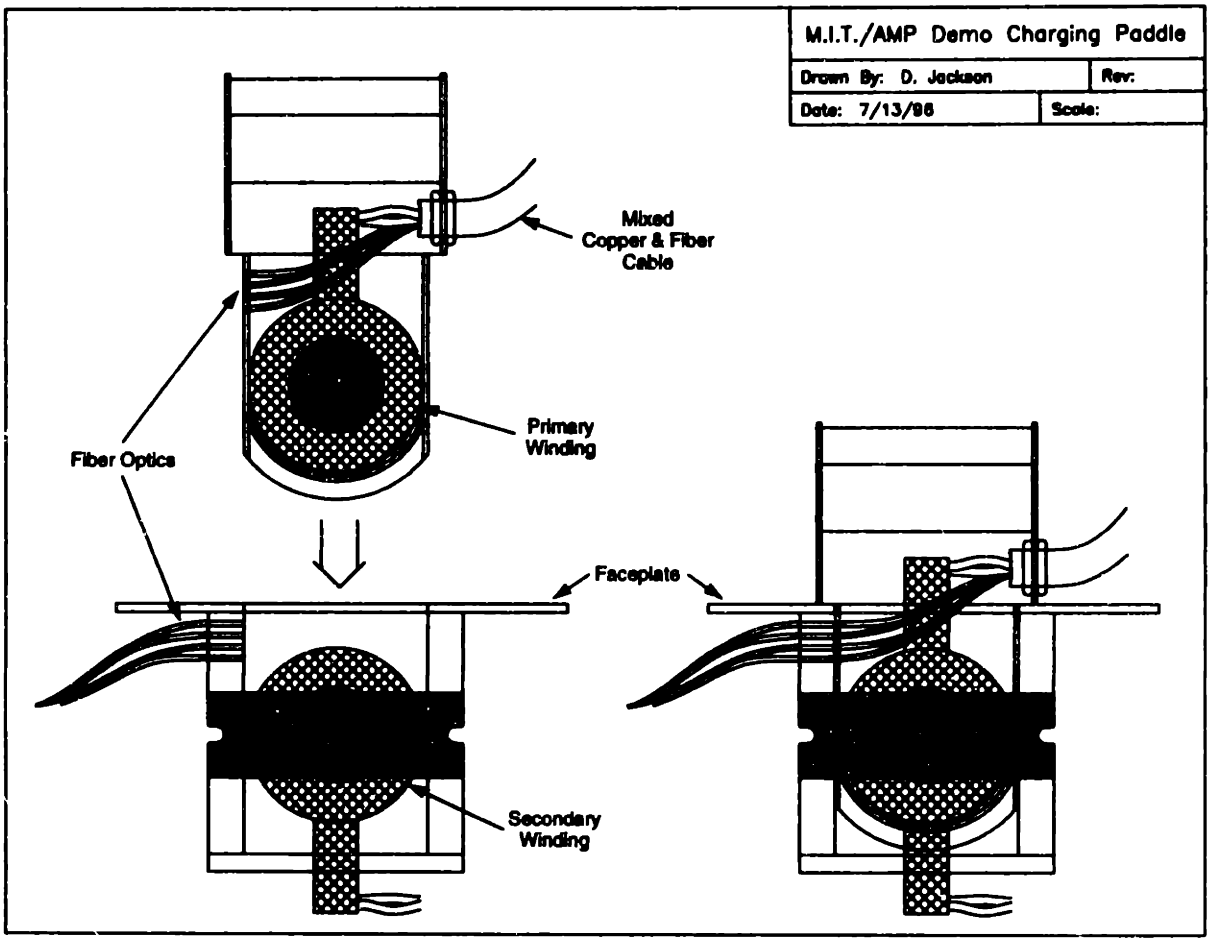



**Figure B.2:** Diagram illustrating the inductively-coupled paddle and inlet. Note that the optical fibers align only when the paddle is fully inserted.

The AMP/M.I.T. prototype includes an optical communications system to relay information between the *CHARGER* and *VEHICLE* sides of the system. The optical link was established using four optical fibers as shown in Figure B.2. The fibers originate from the DC/DC controller board and run down the length of a mixed copper/fiber cable. A multimode flexible plastic fiber was used. The fibers terminate at the edge of the inductive coupling where their ends are polished and aligned precisely as shown in Figures B.2 and B.3. The precise alignment ensures that an optical connection is possible only when the paddle is fully inserted into the inlet. This allows the microcontroller to ascertain the status of the coupler by polling the communication system. Additional detail is provided in Figures 3.3 and 3.4 in the text.

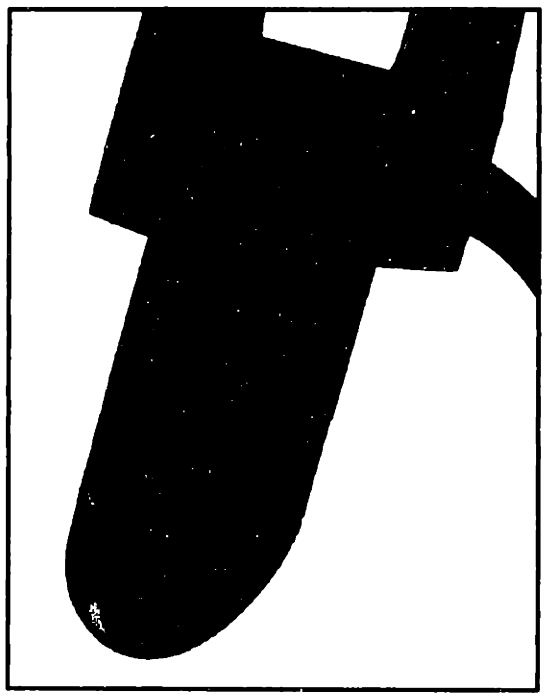


**Figure B.3:** The AMP/M.I.T. prototype paddle. (Photograph is courtesy of AMP.)

The communications link operates bidirectionally. Digital data is relayed simultaneously in both directions at a rate of 62.5 kbits/second. The link operates synchronously, with clock, sync, and data signals originating from the *CHARGER* side of the system, and a second data signal returning information from the *VEHICLE* side. Sequencing for the communication signals is controlled by a finite-state machine. High-output red LED emitters drive the optical fibers directly from buffered TTL data. On the receiving end, fiber-optic photodetectors from Honeywell yield TTL compatible outputs. The system relays measurements of the charger-side output voltage and current across the coupling. Analog data is encoded using two 12-bit serial-output A/D converters. An additional eight bits of binary data is exchanged bidirectionally. This data is used to monitor and control the DC/DC converter operation.

The same communications link was used for both the unidirectional and bidirectional inductively-coupled prototypes. When the system was bench tested, and the pot-core transformer was used in place of the paddle/inlet coupling, the optical fiber system was joined directly using a short fiber segment. Complete schematics of the communications hardware are provided in Appendix C. Relevant source code for the programmable logic devices is provided in Appendix D.

# Appendix C

# Schematic Diagrams

This appendix contains schematic diagrams for the hardware prototypes as well as overviews of the PCB layouts and photographs of the completed circuit boards. Many of the schematics make use of programmable logic devices. Source code for these devices is provided in Appendix D.

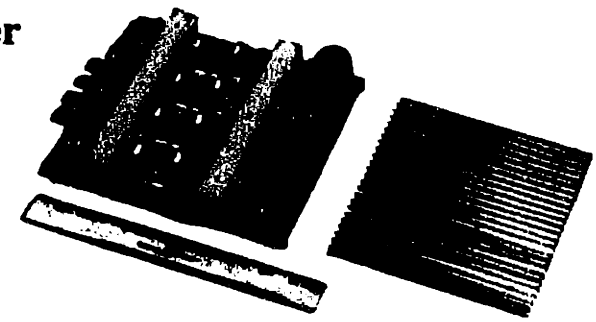
- Interleaved boost converter — Section 3.2
- Digital microcontroller board — Section 3.1, 5.1
- Half-bridge DC/DC converter and control boards — Section 3.3
- Bidirectional boost-buck-inverter — Section 4.2
- Full-bridge DC/DC converter and control boards — Section 4.3
- 3.0-MHz class-E induction heater — Section 10.3

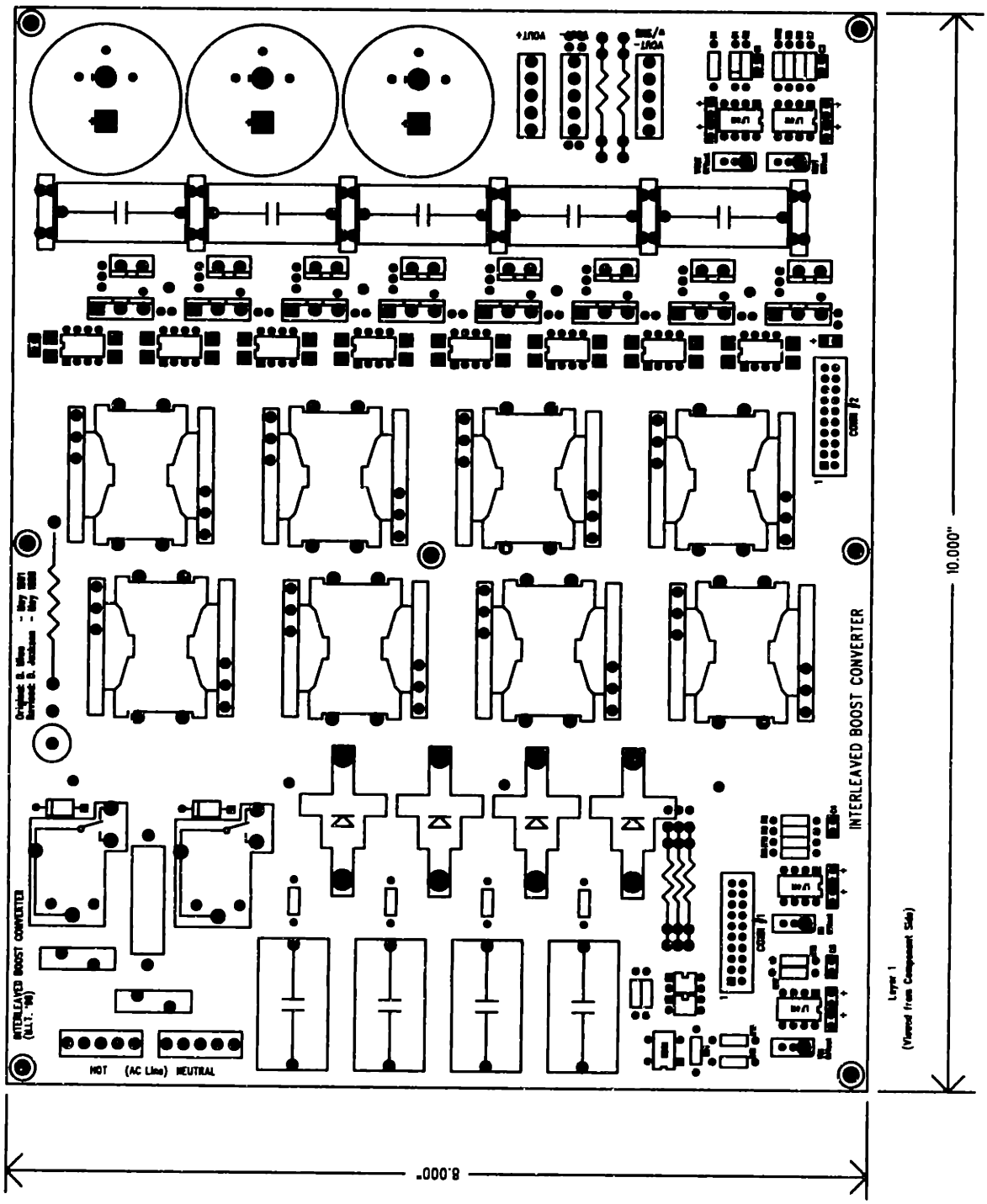
The following were early prototypes that were hand-wired using vector-board construction. Full schematics are not available, however, photographs of the completed units are provided.

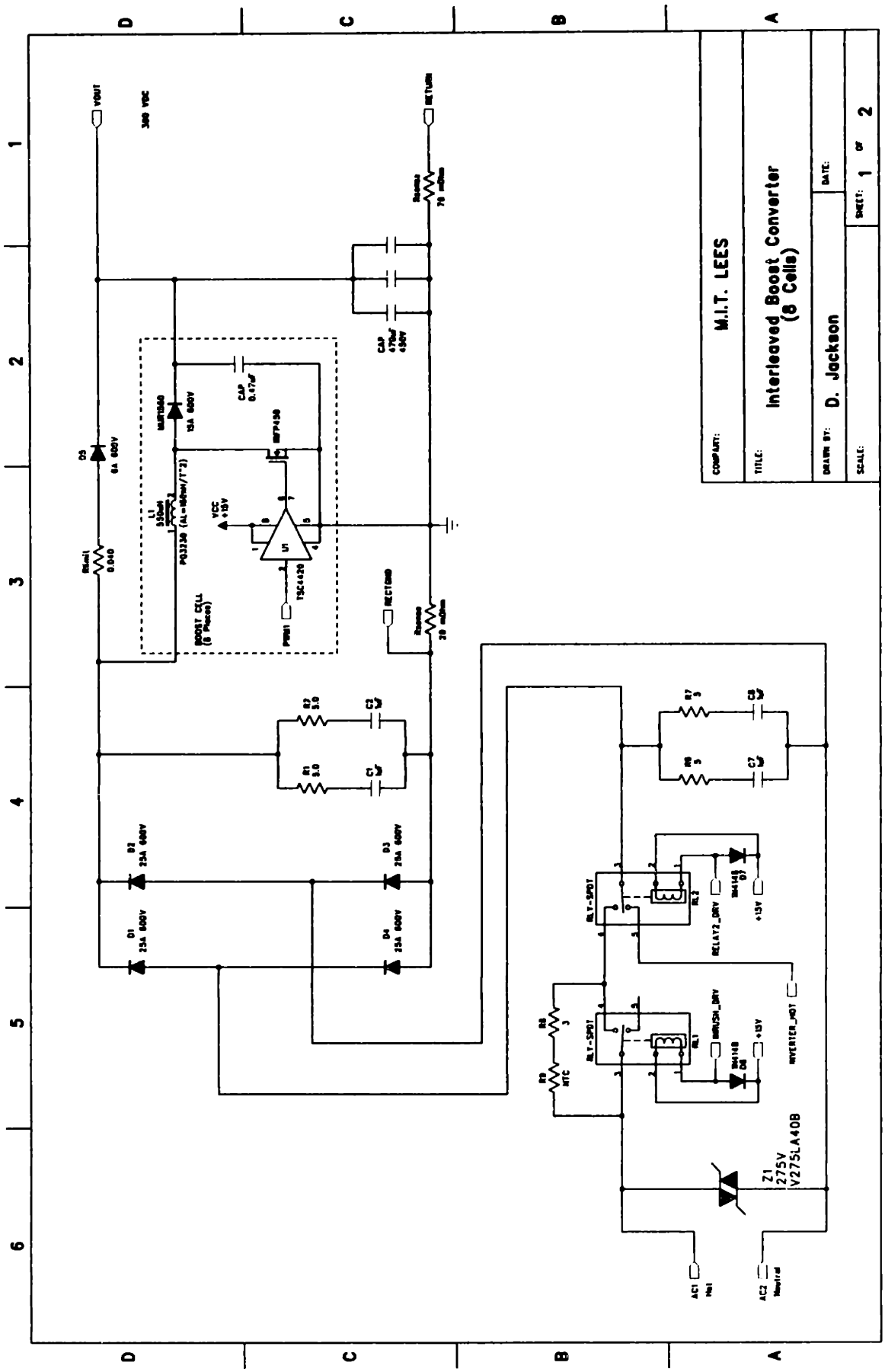
- Full-bridge induction heater — Section 10.2
- Portable 500-kHz class-E Induction Heater — Section 10.3

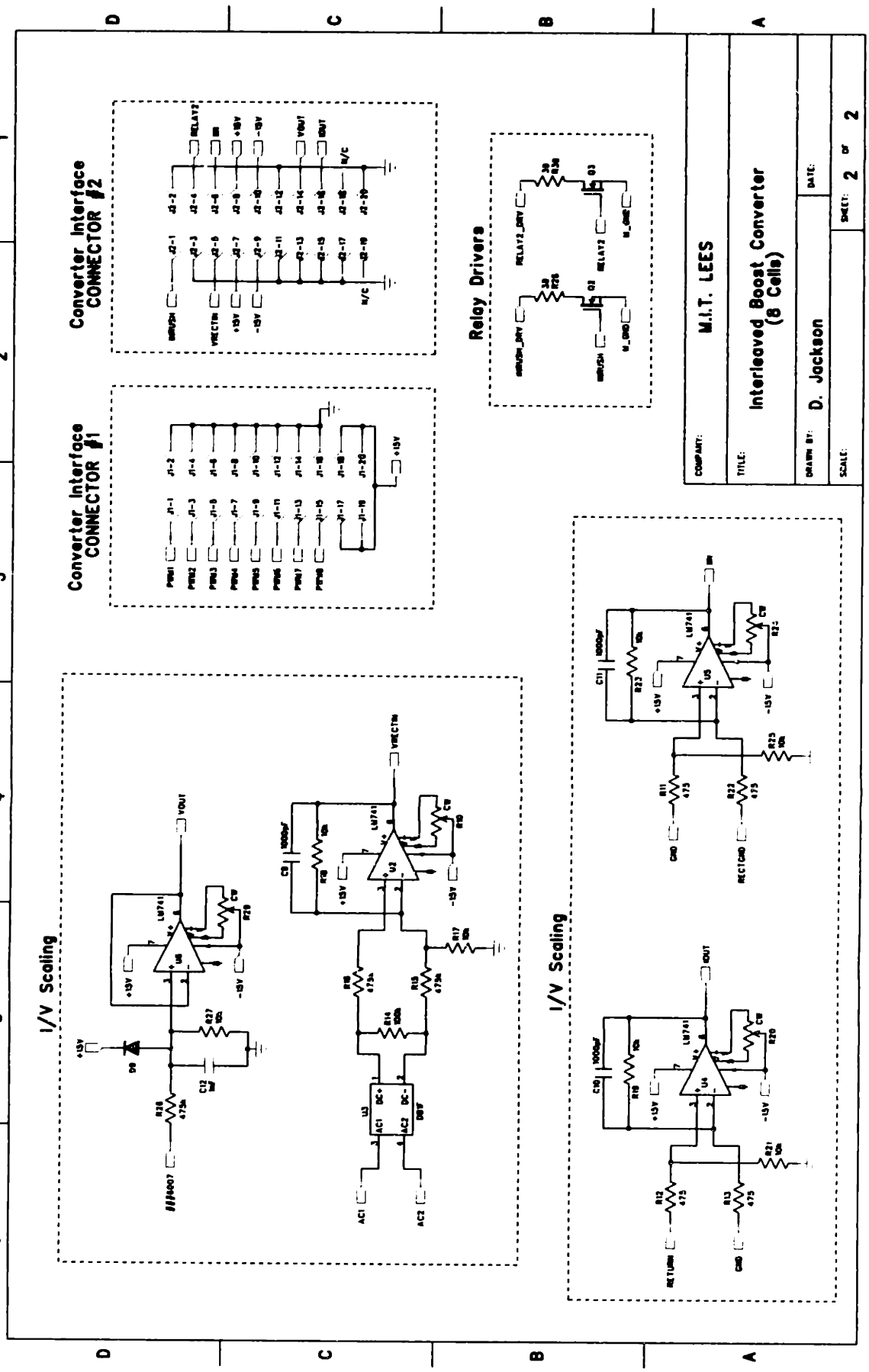
## C.1 Interleaved Boost Converter

- Photograph
- PCB Layout (4 Layer)
- Schematic Drawings



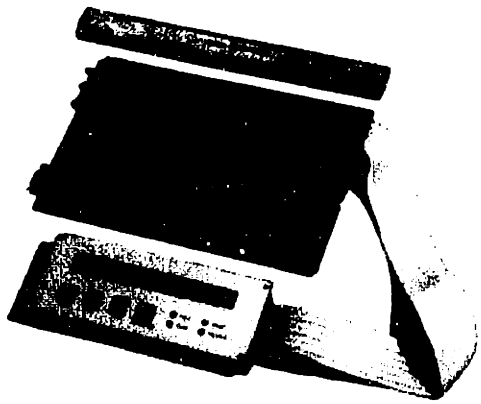


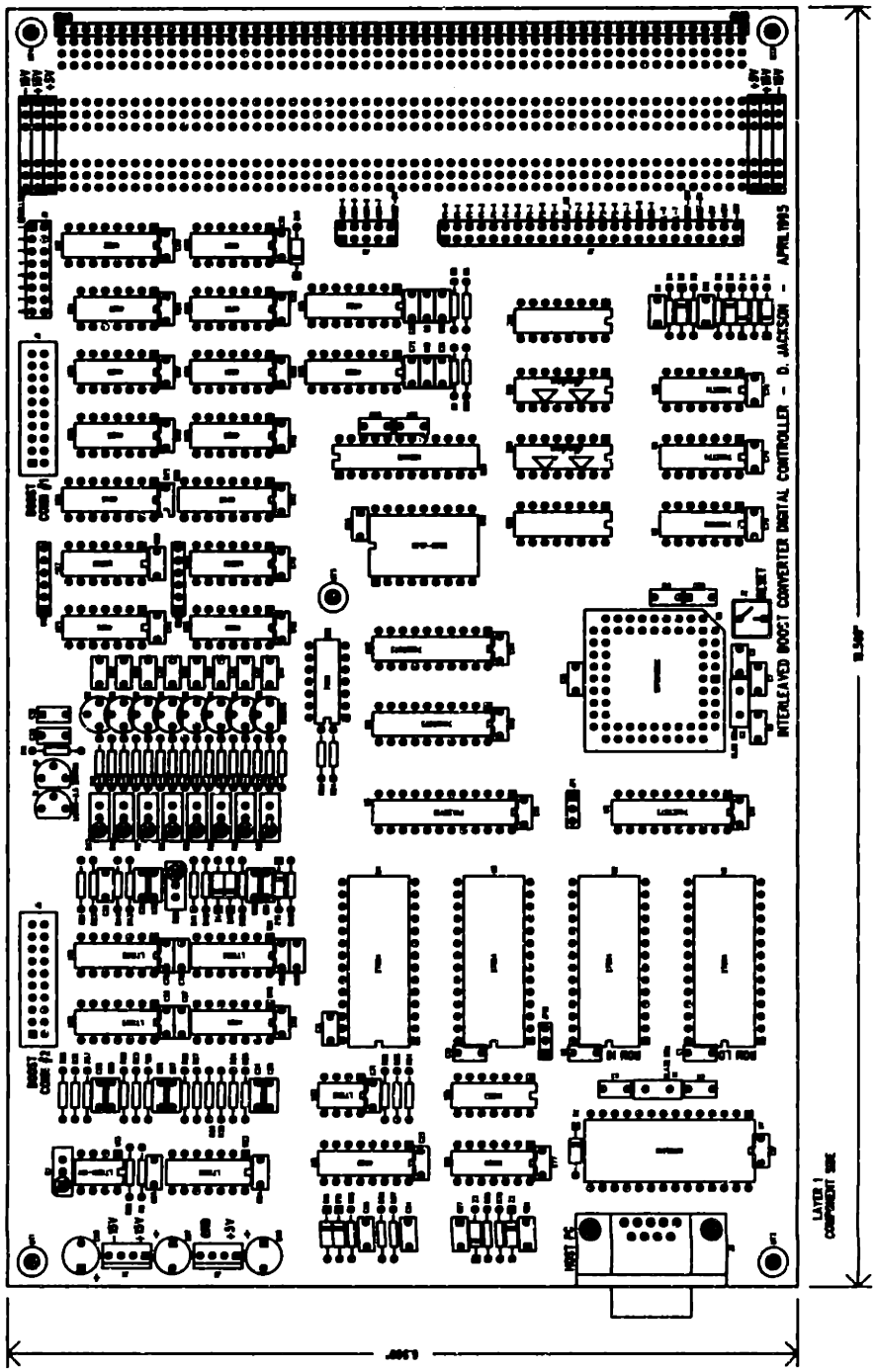

COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter (8 Cells)
DRAWN BY: D. Jackson
DATE:
SCALE:
SHEET: 1 OF 2
VOUT
300 VDC
RETURN
CAP 470uF 450V
CAP 0.47uF
MUR1560 15A 600V
IRF450
D5 6A 600V
L1 350uH PQ3230 (AL=182nH/T^2)
VCC +15V
U1
TSC4420
PWM1
BOOST CELL (8 Places)
Rsense 10 mOhm
D1 25A 600V
D2 25A 600V
D3 25A 600V
D4 25A 600V
R1 5.0
R2 5.0
C1
C2
R7 3
R8 3
C7
C8
RLY-SPDT
RL1
RL2
1N4148 D6
1N4148 D7
+15V
BRUSH_DRV
RELAY2_DRV
INVERTER_HOT
R9 3
R10 NTC
Z1 275V V275LA40B
AC1 Hot
AC2 Neutral

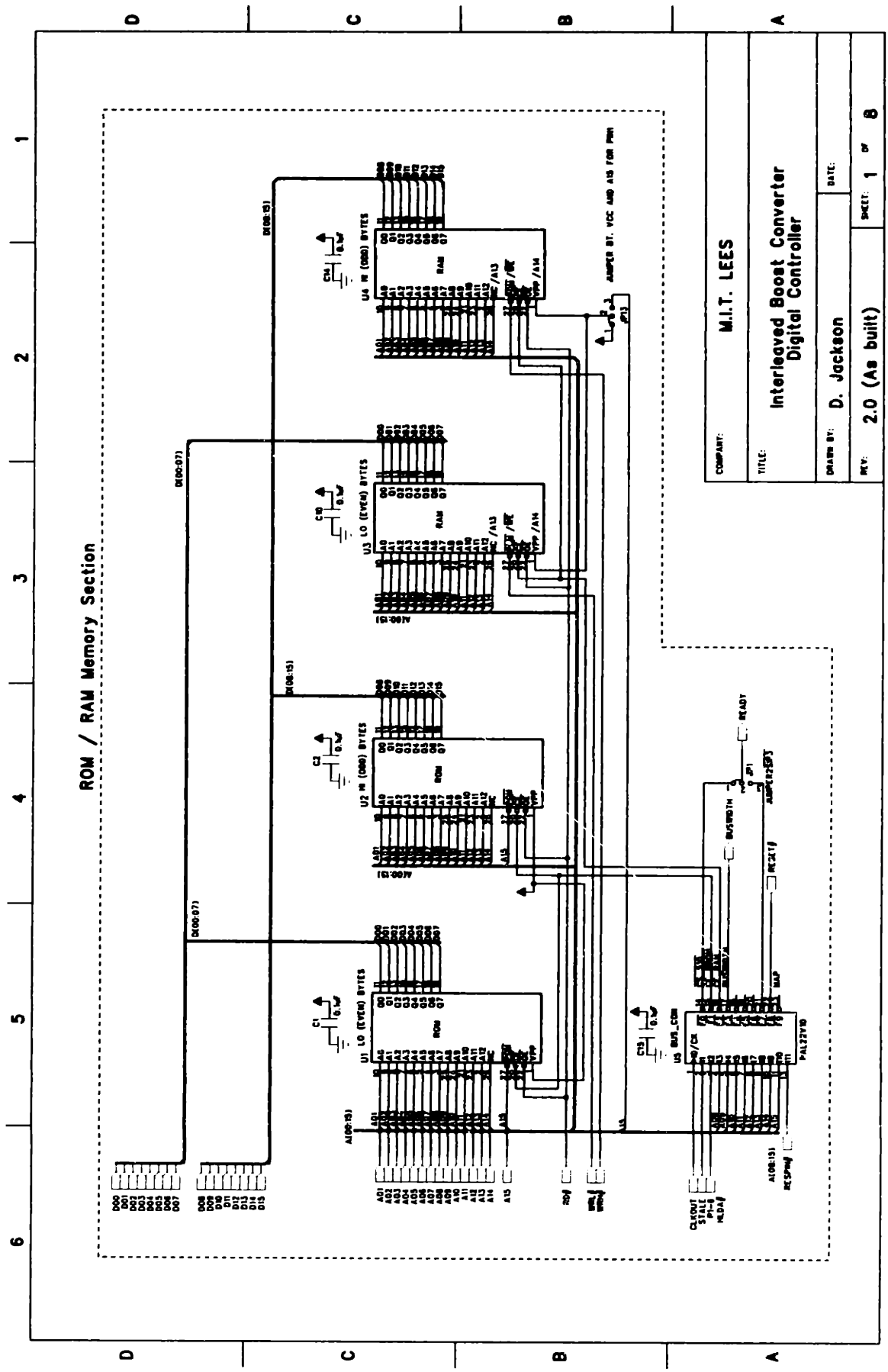

Converter Interface
CONNECTOR #2
Converter Interface
CONNECTOR #1
Relay Drivers
I/V Scaling
I/V Scaling
COMPANY:
M.I.T. LEES
TITLE:
Interleaved Boost Converter
(8 Cells)
DRAWN BY:
D. Jackson
DATE:
SCALE:
SHEET: 2 OF 2

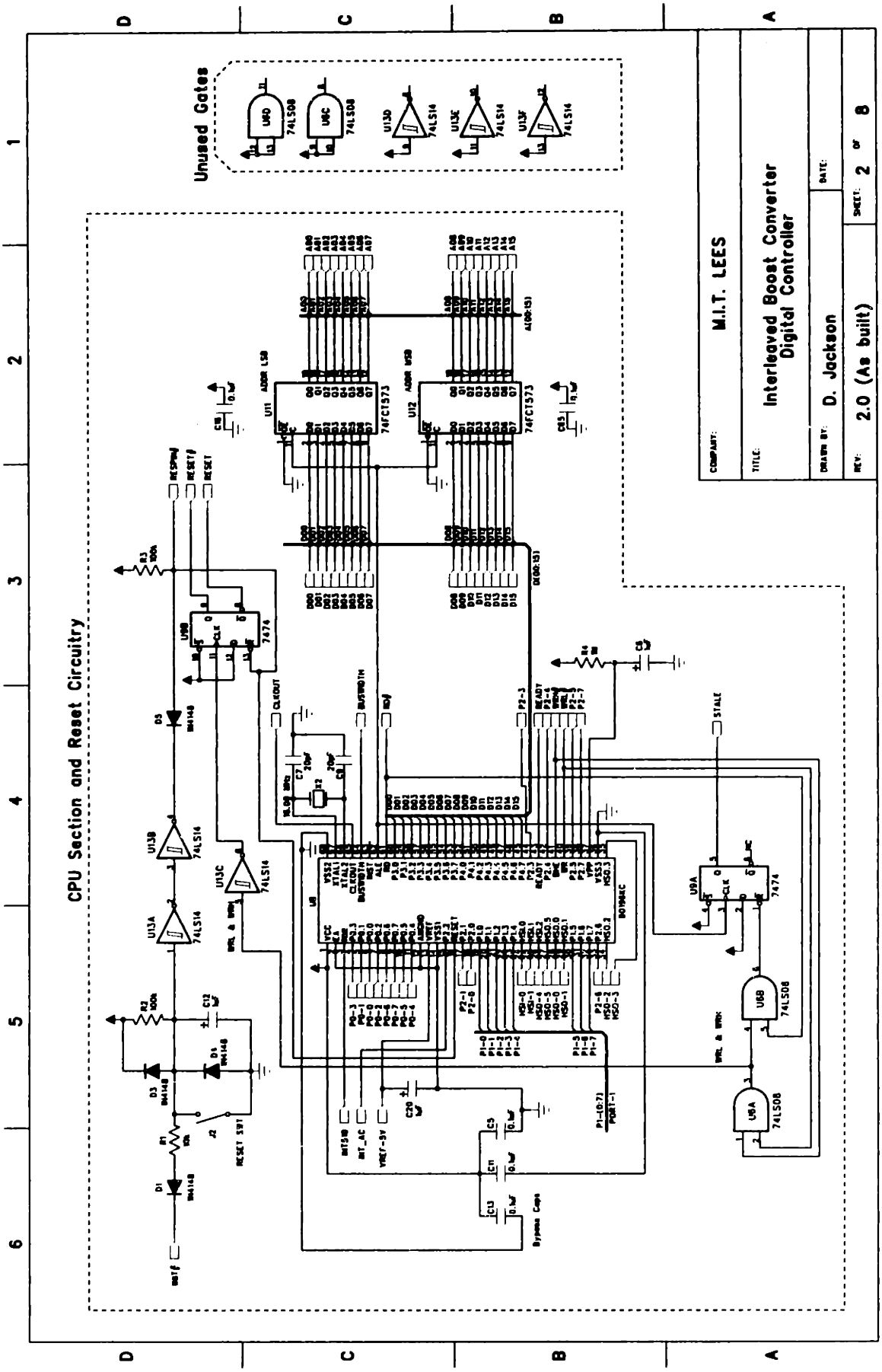
## C.2 Digital Microcontroller Board

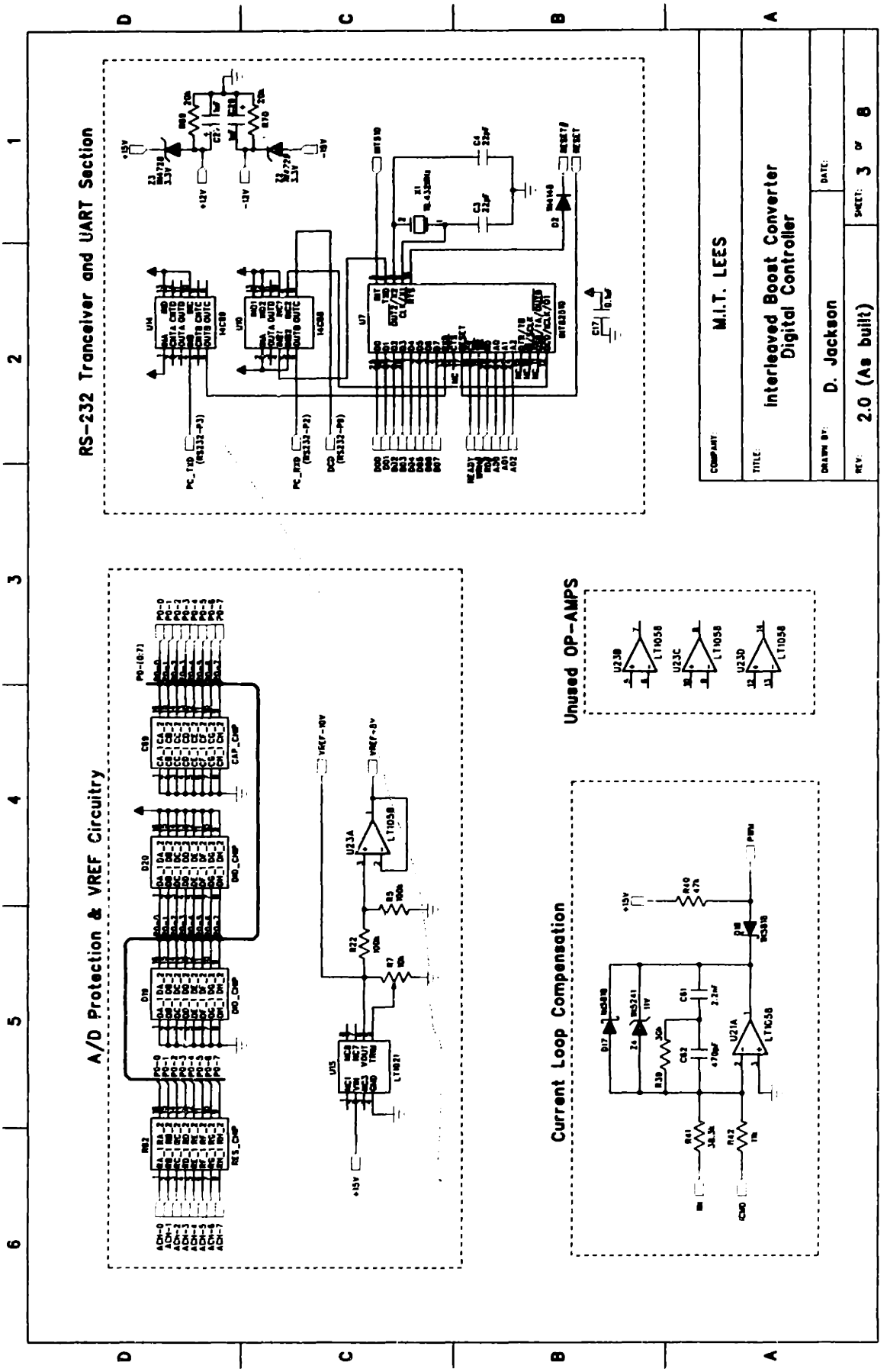
- Photograph
- PCB Layout (4 Layers)
- Schematic Drawings

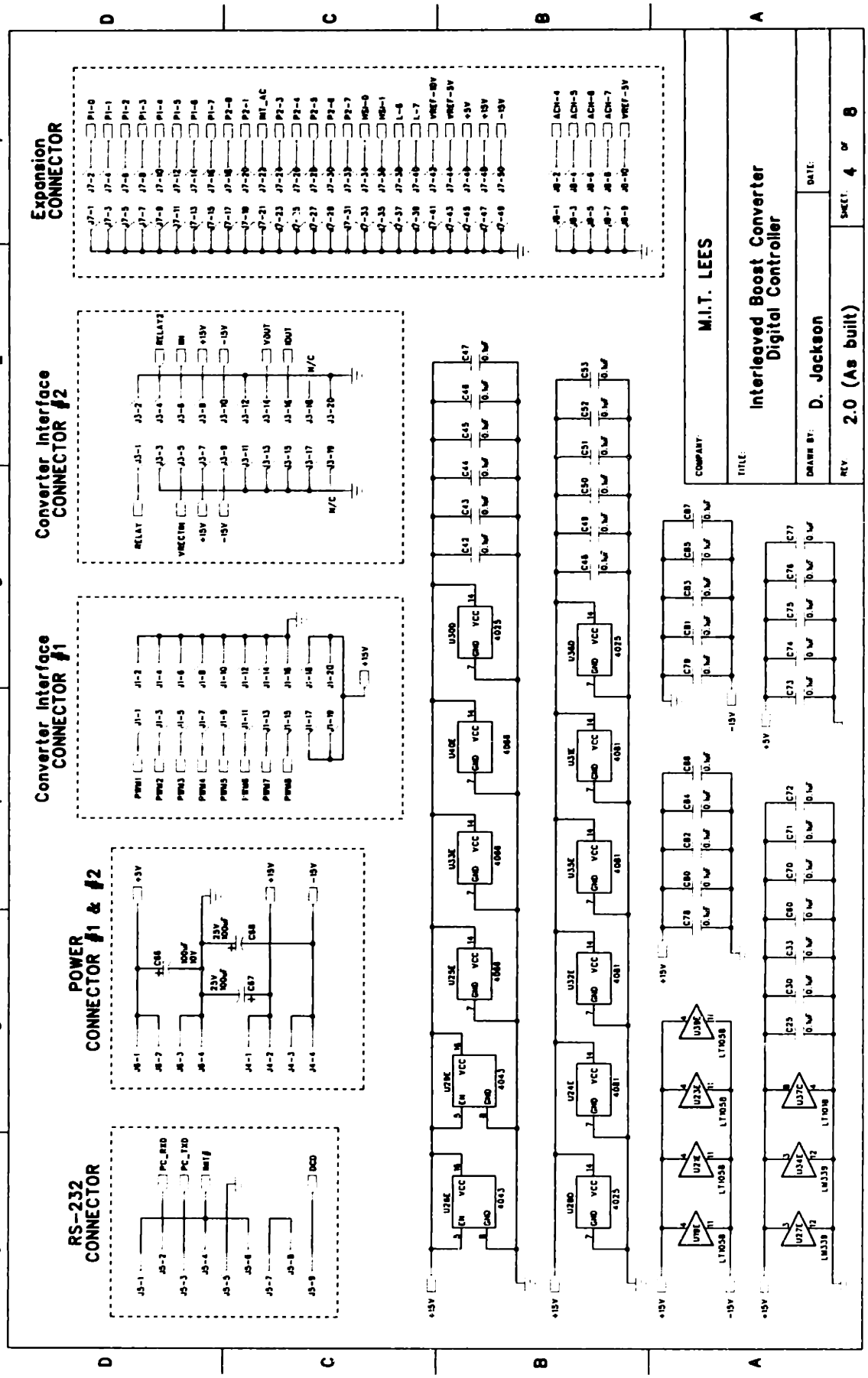


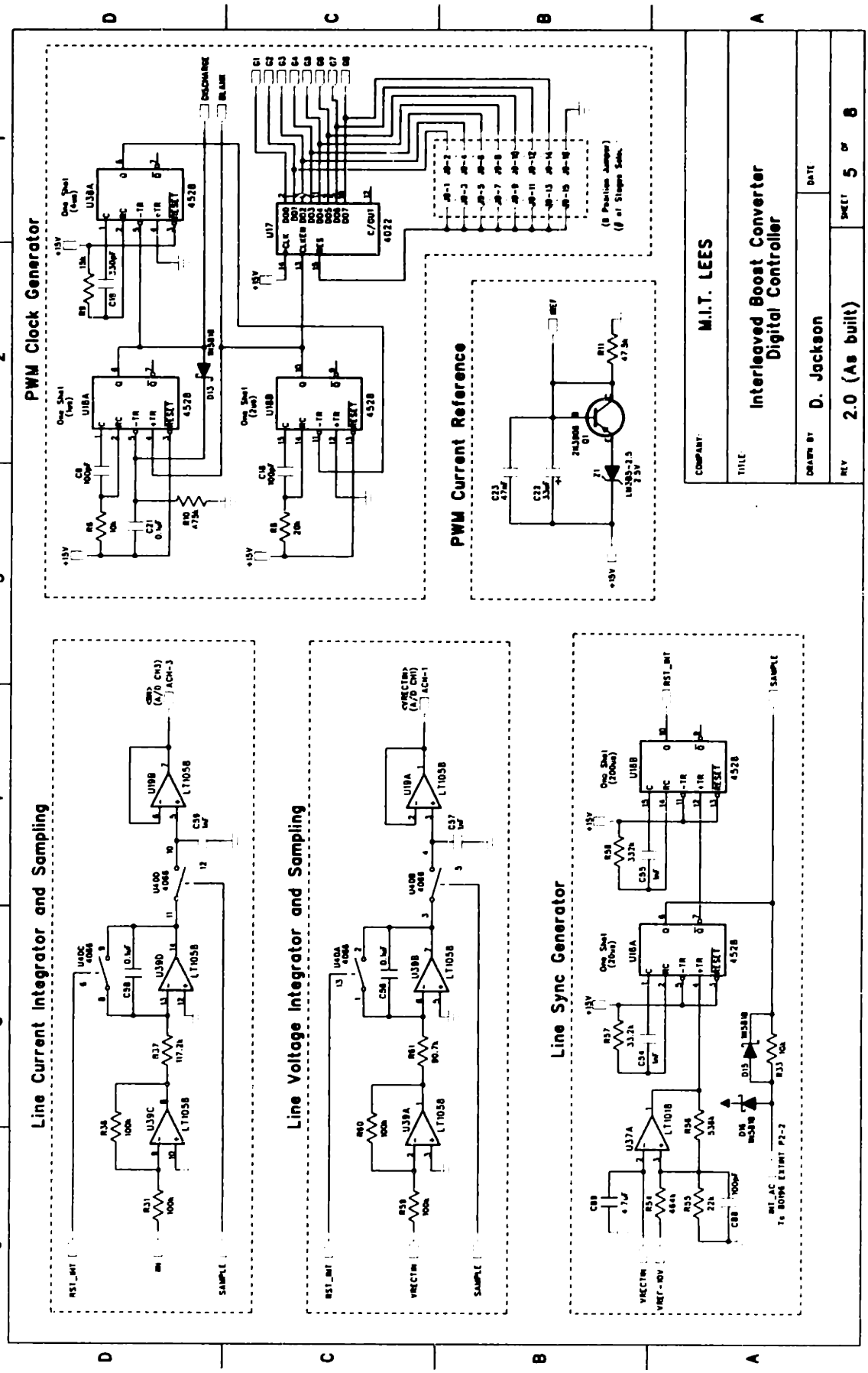


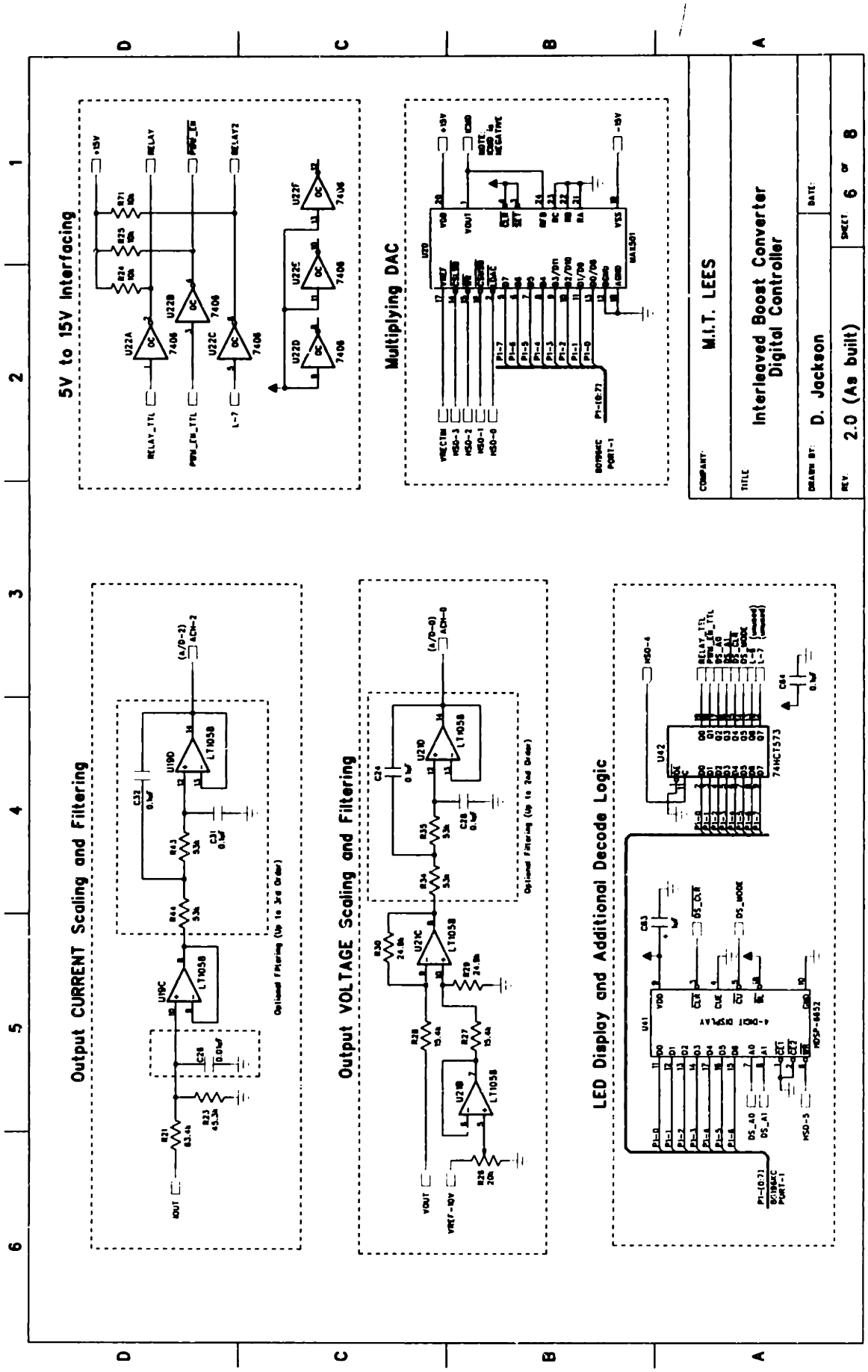

ROM / RAM Memory Section
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 1 OF 8

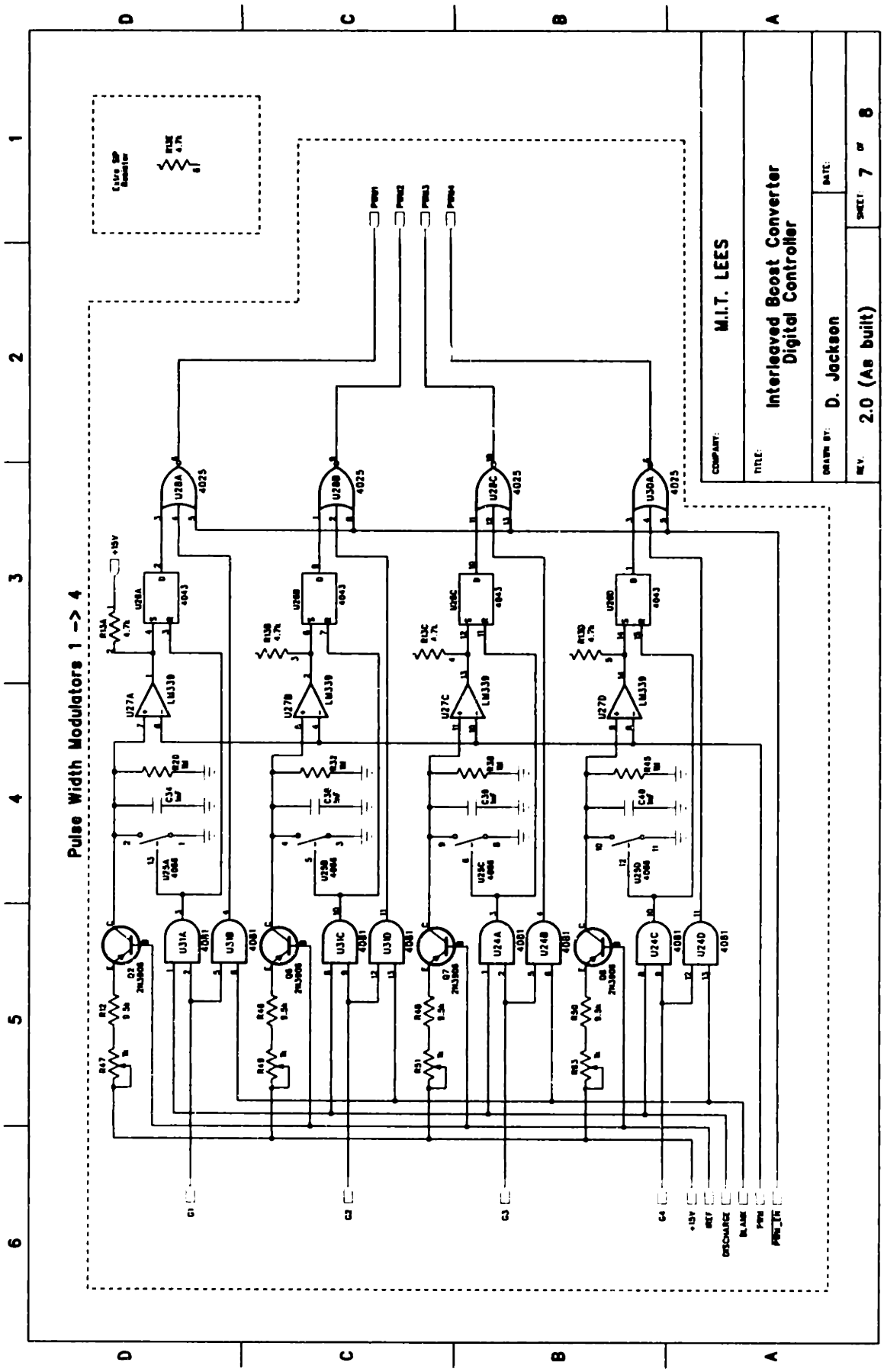

Unused Gates
U9D 74LS08
U9C 74LS08
U13D 74LS14
U13E 74LS14
U13F 74LS14
CPU Section and Reset Circuitry
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 2 OF 8

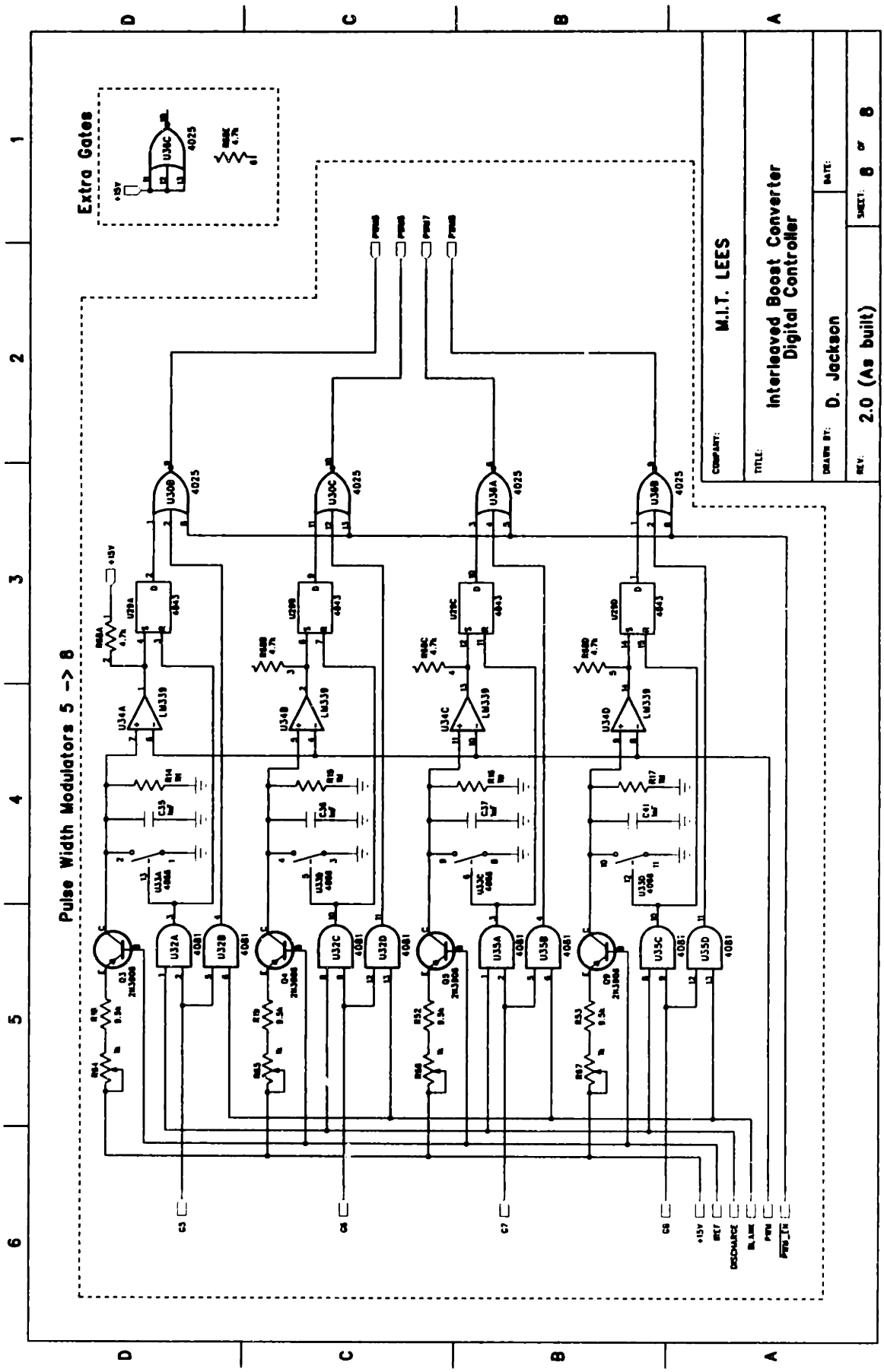

RS-232 Tranceiver and UART Section
A/D Protection & VREF Circuitry
Unused OP-AMPS
Current Loop Compensation
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 3 OF 8

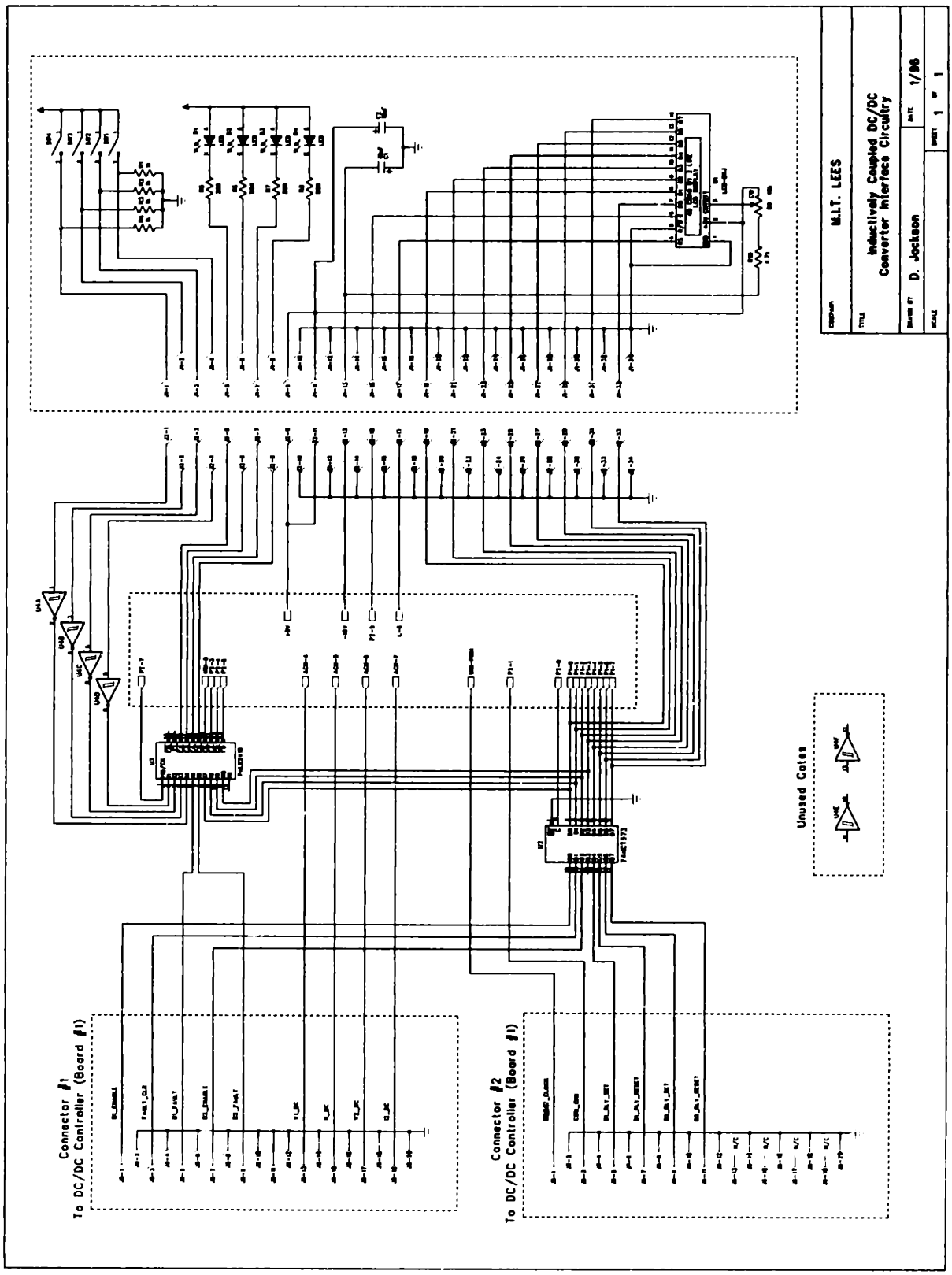

Expansion CONNECTOR
Converter Interface CONNECTOR #2
Converter Interface CONNECTOR #1
POWER CONNECTOR #1 & #2
RS-232 CONNECTOR
M.I.T. LEES
Interleaved Boost Converter
Digital Controller
D. Jackson
2.0 (As built)
4 OF 8


PWM Clock Generator
PWM Current Reference
Line Current Integrator and Sampling
Line Voltage Integrator and Sampling
Line Sync Generator
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET 5 OF 8


5V to 15V Interfacing
Multiplying DAC
Output CURRENT Scaling and Filtering
Output VOLTAGE Scaling and Filtering
LED Display and Additional Decode Logic
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 6 OF 8

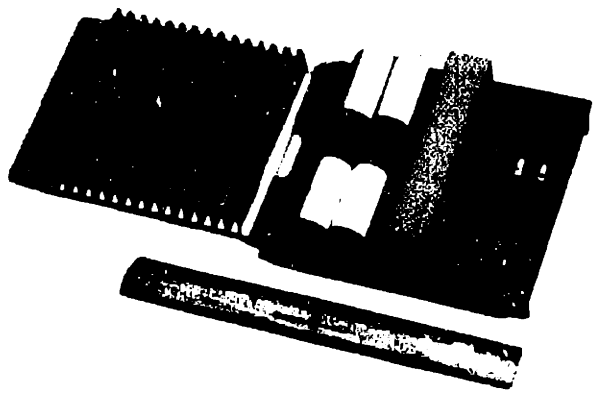

Pulse Width Modulators 1 -> 4
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 7 of 8

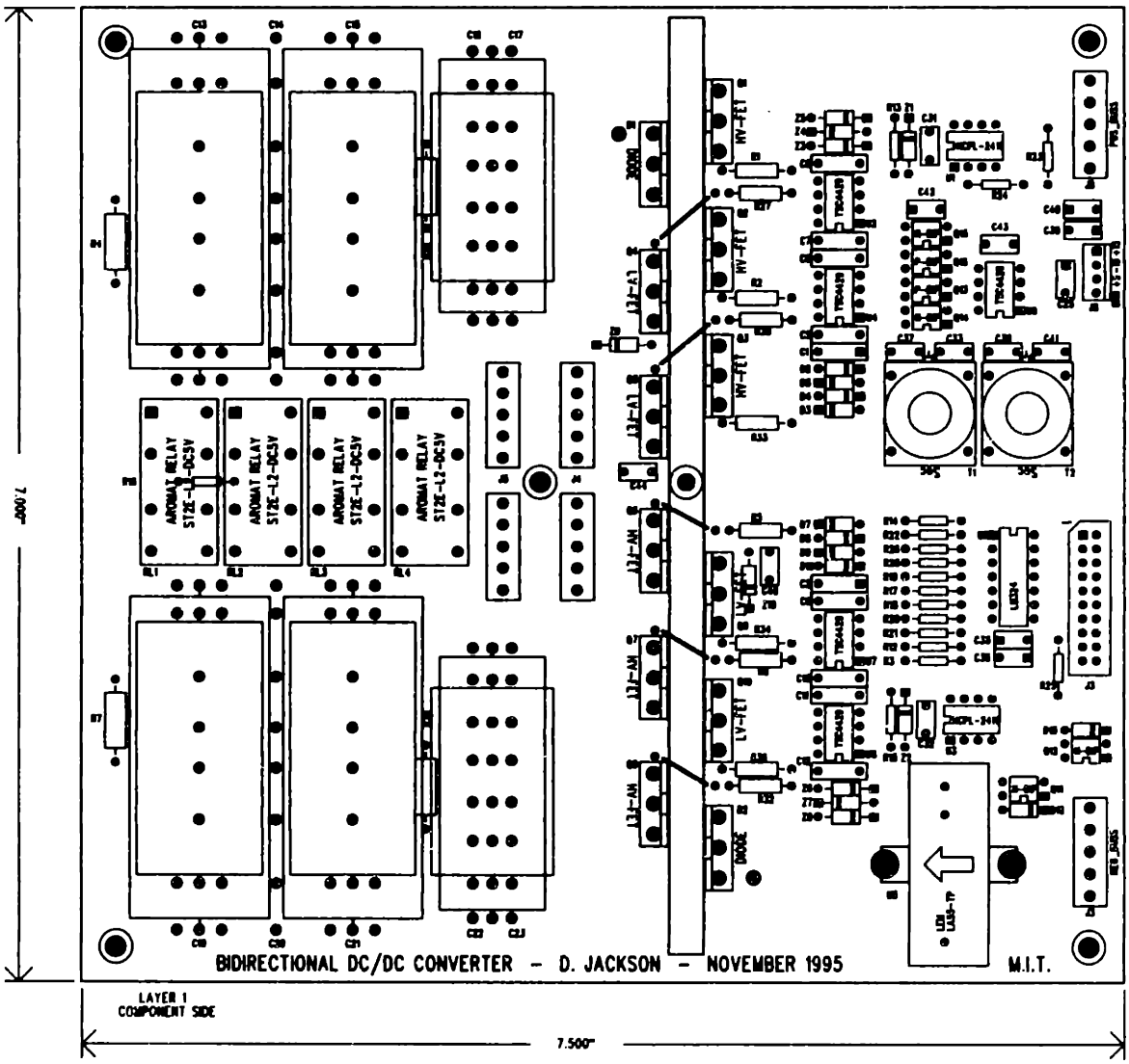

Extra Gates
Pulse Width Modulators 5 -> 8
COMPANY: M.I.T. LEES
TITLE: Interleaved Boost Converter Digital Controller
DRAWN BY: D. Jackson
DATE:
REV: 2.0 (As built)
SHEET: 8 of 8

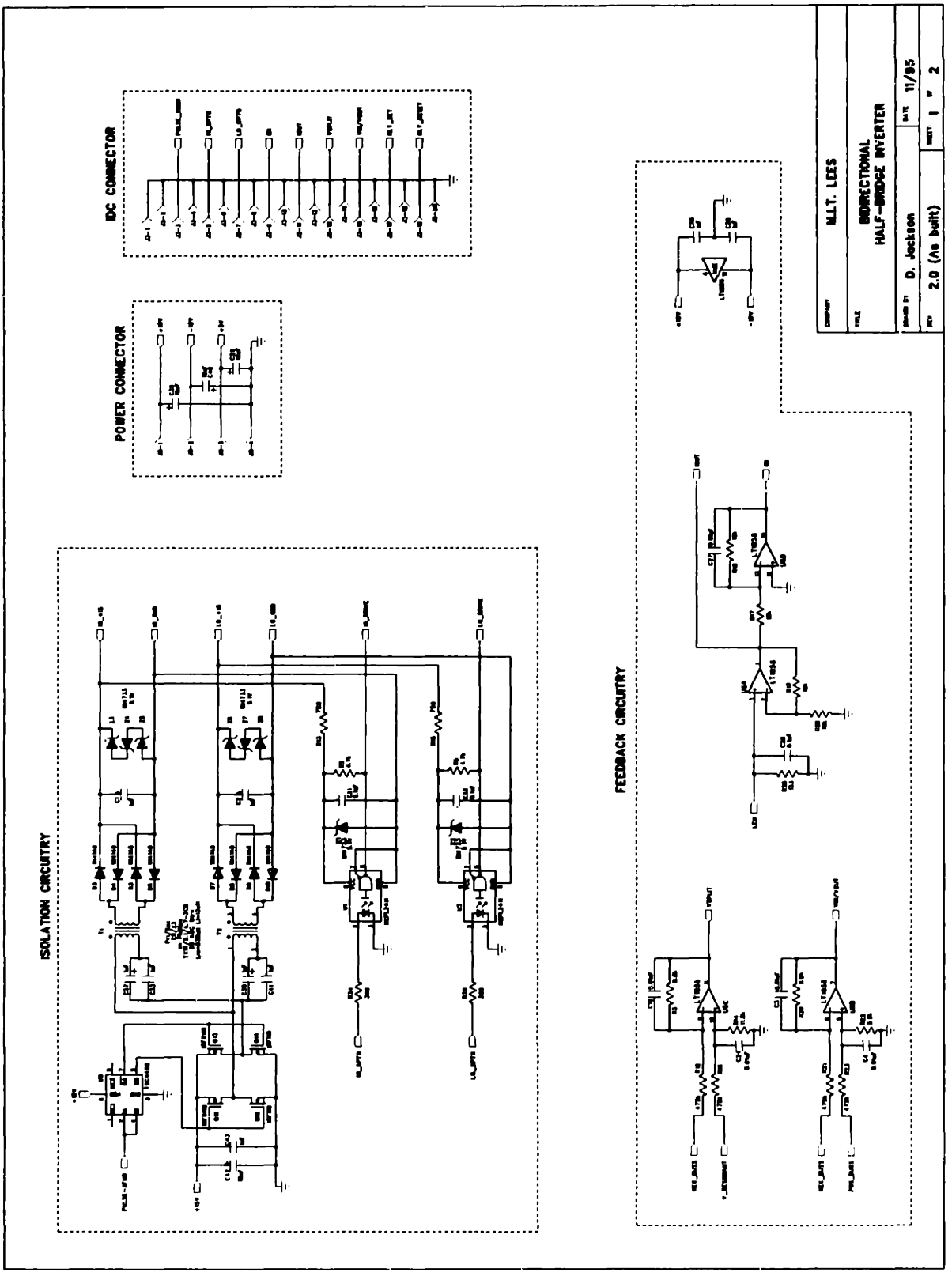

M.I.T. LEES
Inductively Coupled DC/DC Converter Interface Circuitry
D. Jackson
1/96
Unused Gates
Connector #1
To DC/DC Controller (Board #1)
Connector #2
To DC/DC Controller (Board #1)

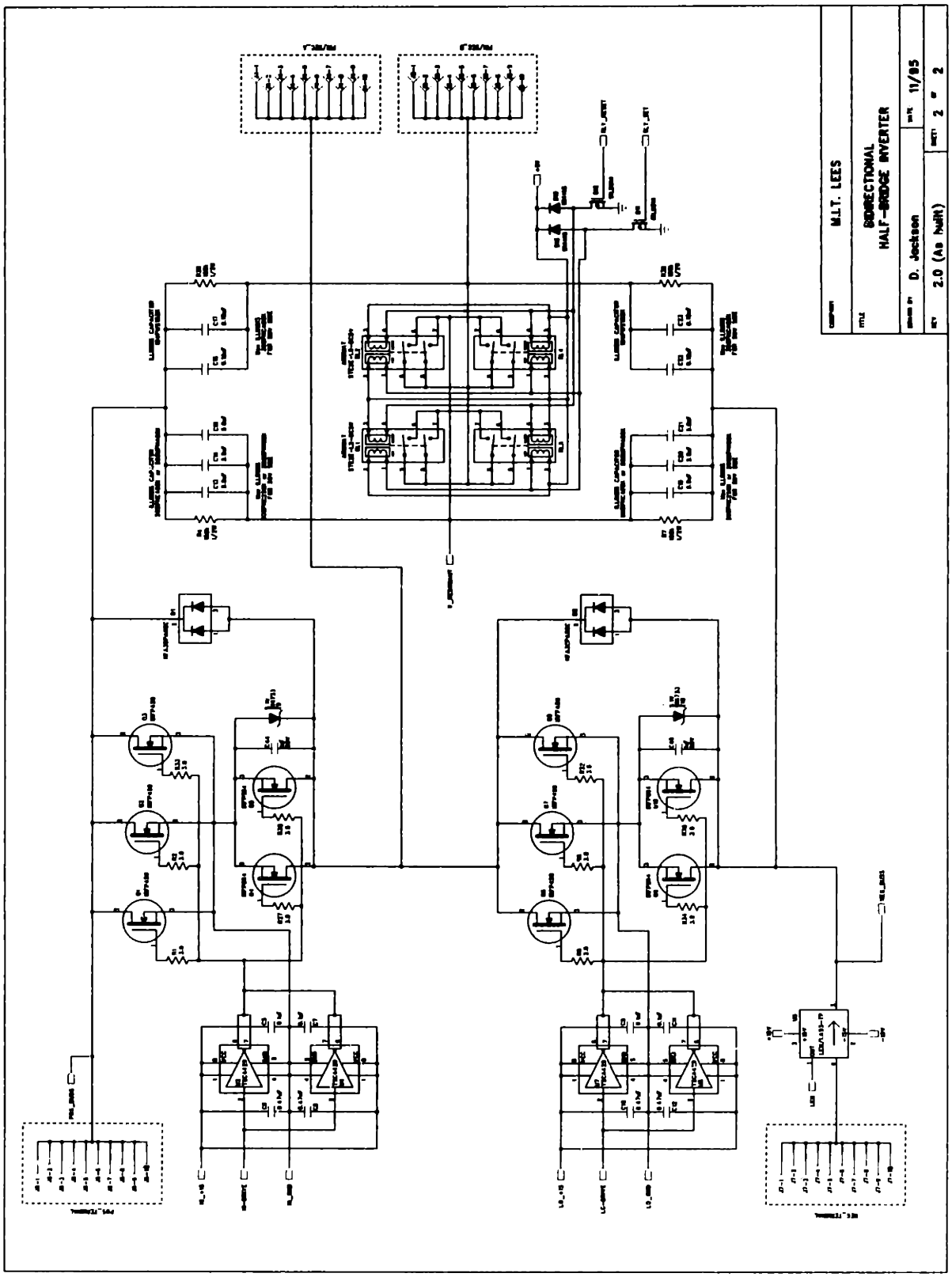
# C.3 Half-Bridge DC/DC Converter

- Photograph
- PCB Layout (2 Layer)
- Schematic Drawings



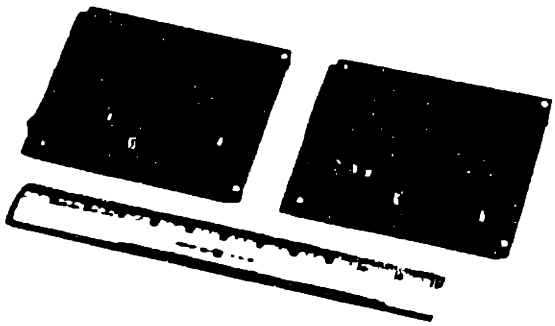


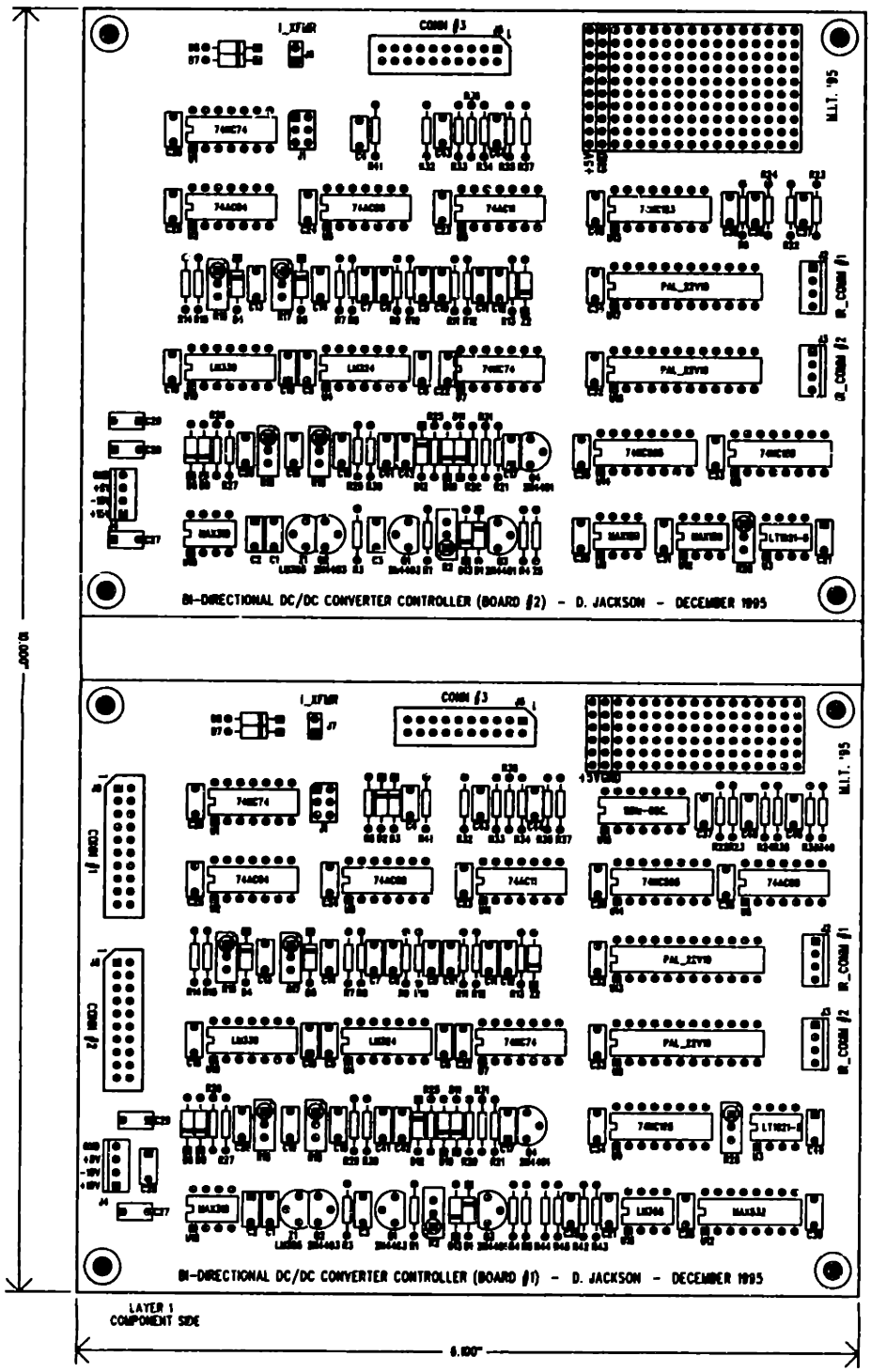

I/O CONNECTOR
POWER CONNECTOR
ISOLATION CIRCUITRY
FEEDBACK CIRCUITRY
M.I.T. LEES
BIDIRECTIONAL
HALF-BRIDGE INVERTER
D. Jackson
11/95
2.0 (As built)

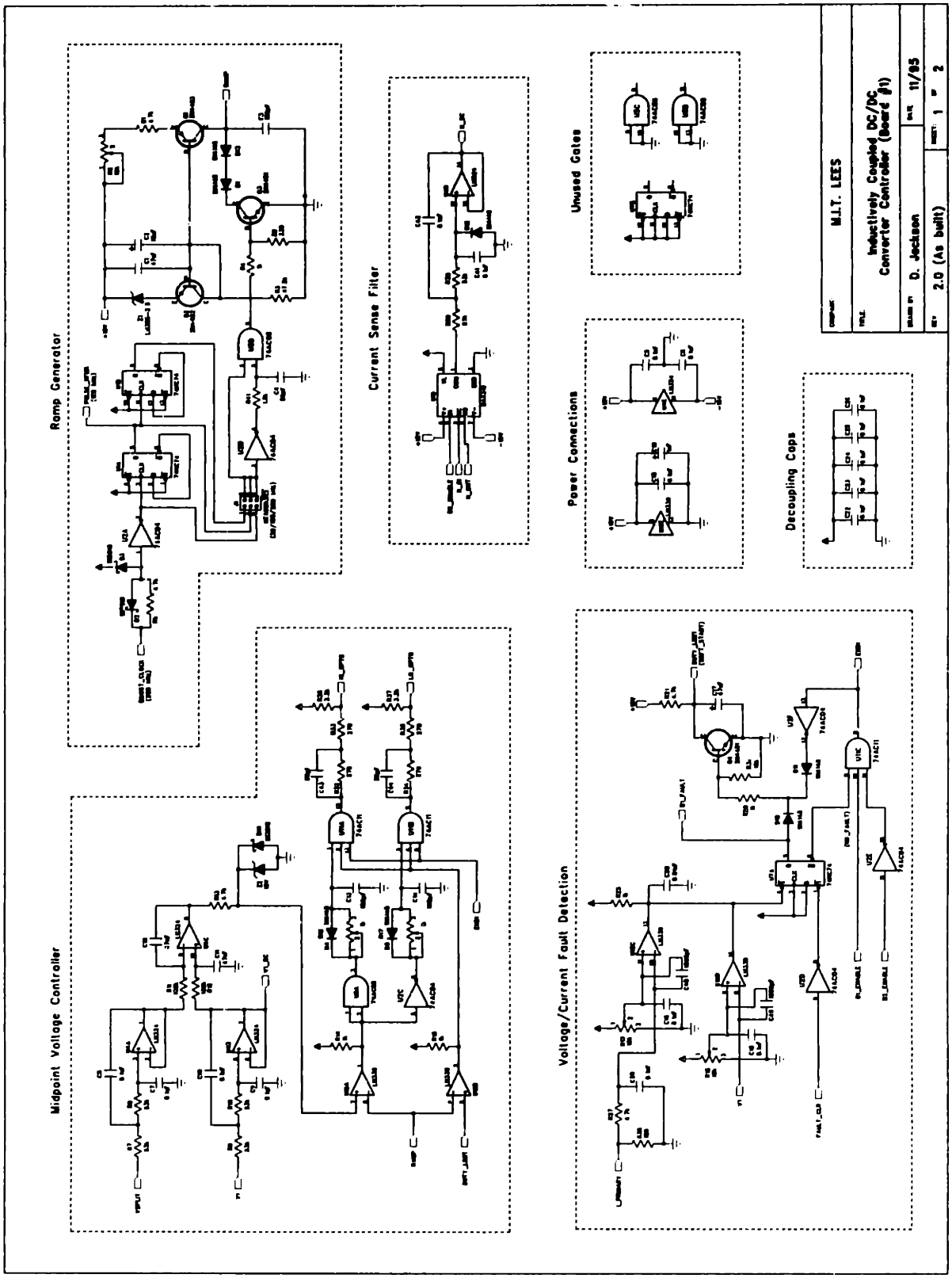

M.I.T. LEES
BIDIRECTIONAL
HALF-BRIDGE INVERTER
D. Jackson
11/95
2.0 (As Built)
2 of 2

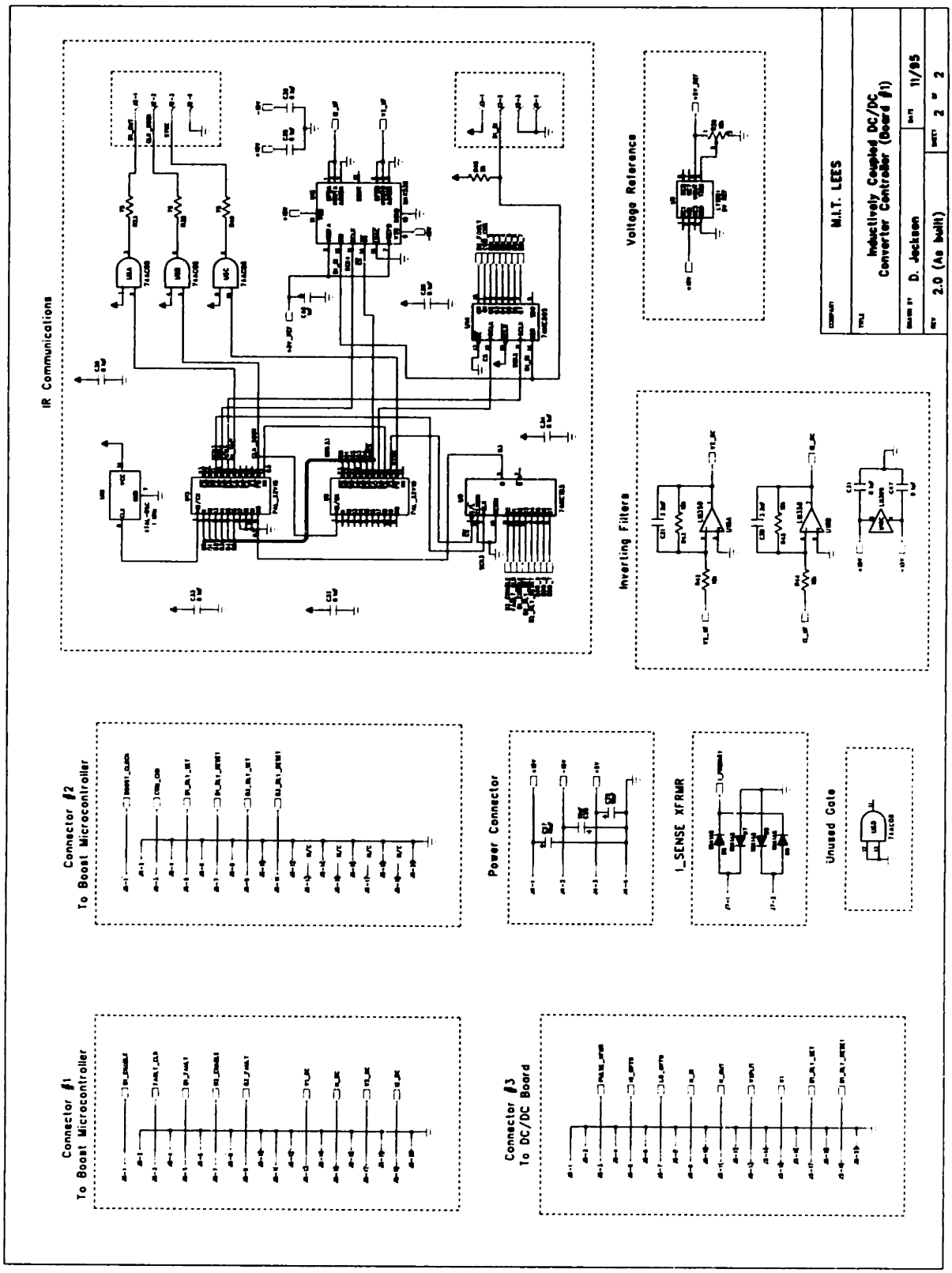
## C.4 Half-Bridge Controllers

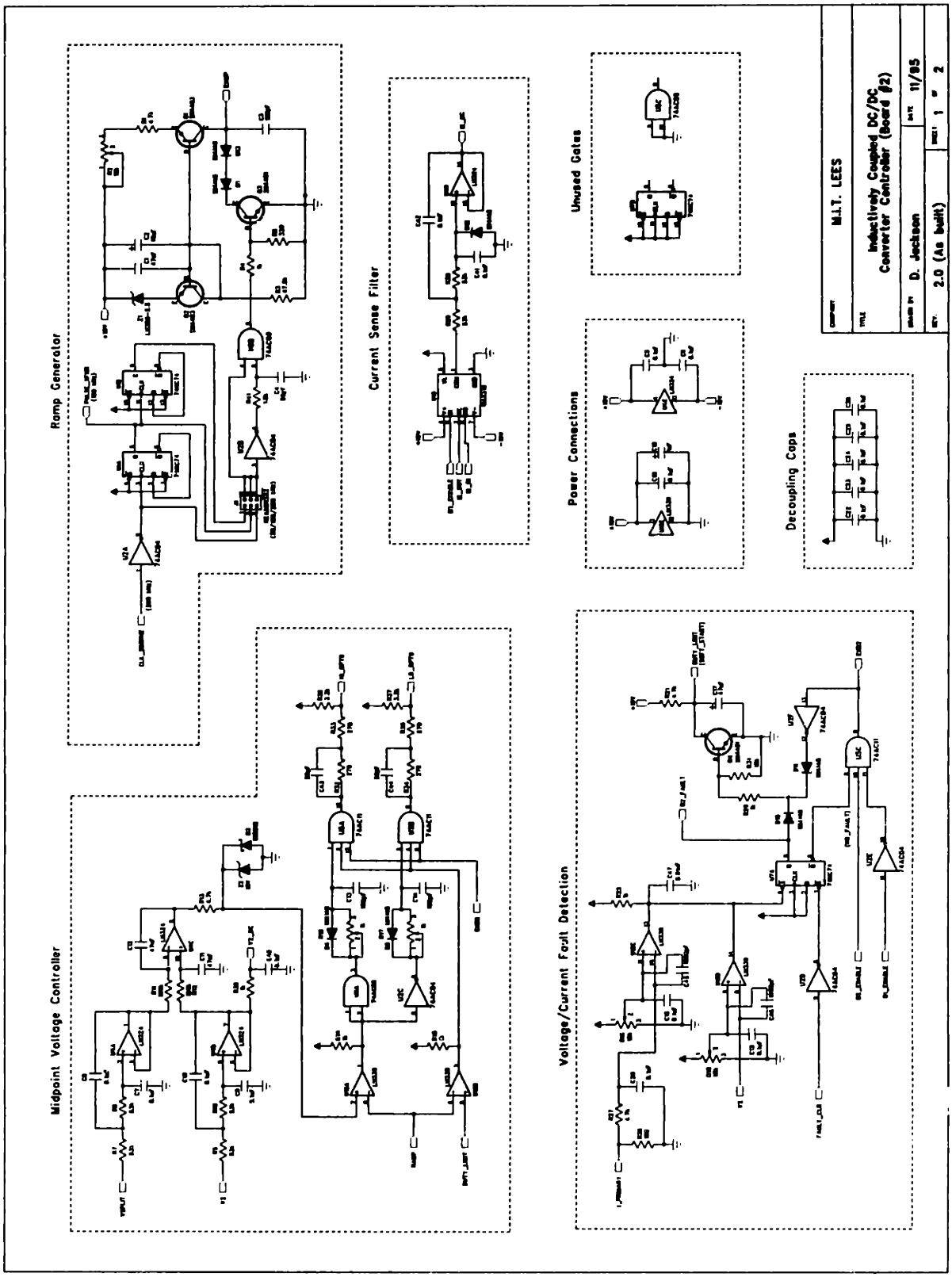
- Photograph
- PCB Layout (2 Layer)
- Schematic Drawings

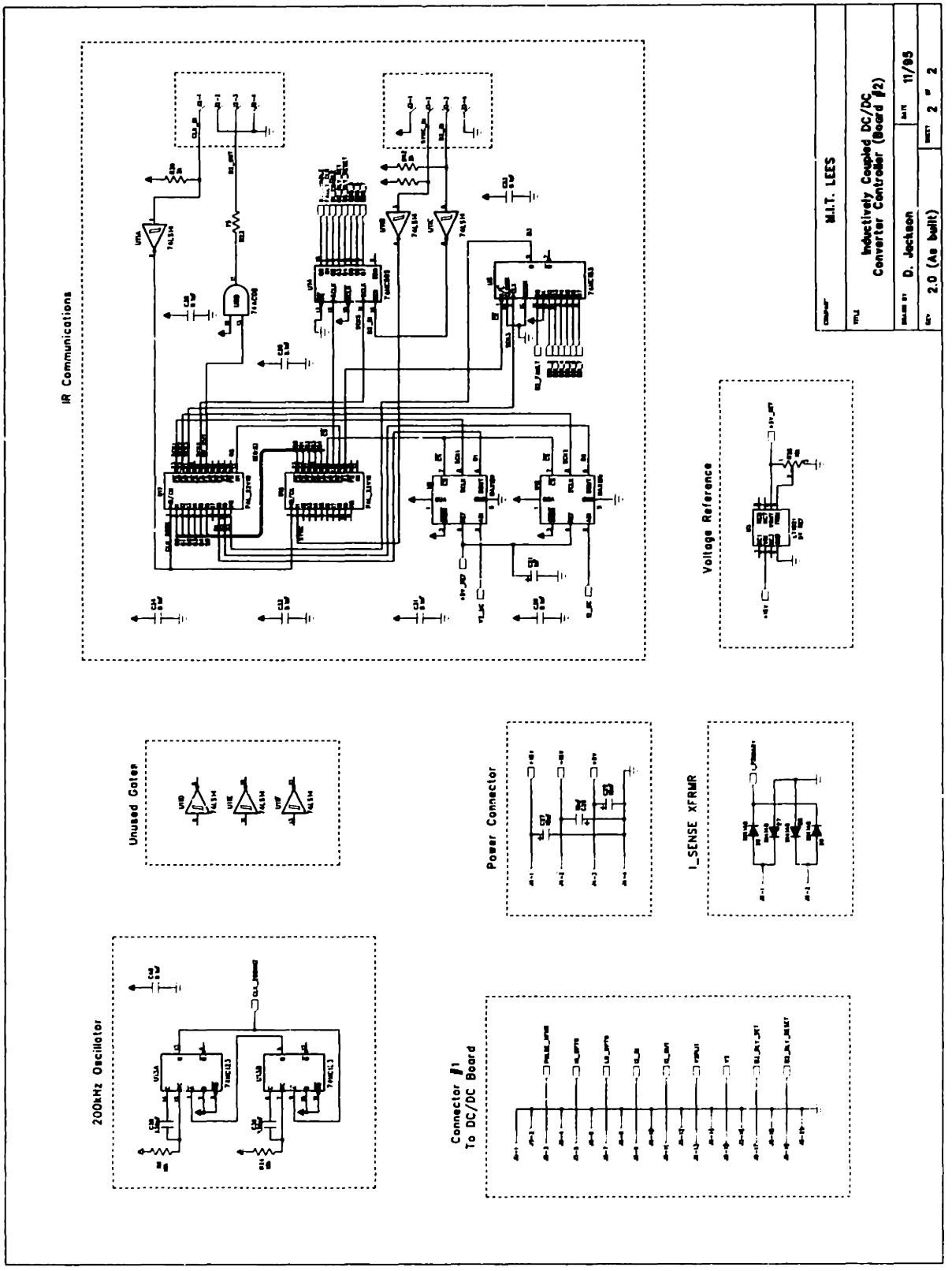





Ramp Generator
Current Sense Filter
Unused Gates
Power Connections
Decoupling Caps
Midpoint Voltage Controller
Voltage/Current Fault Detection
M.I.T. LEES
Inductively Coupled DC/DC Converter Controller (Board #1)
D. Jackson
11/95
2.0 (As built)

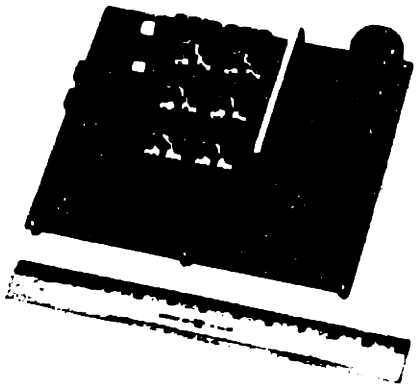

IR Communications
Voltage Reference
Inverting Filters
Connector #2
To Boost Microcontroller
Power Connector
I_SENSE XFRMR
Unused Gate
Connector #1
To Boost Microcontroller
Connector #3
To DC/DC Board
M.I.T. LEES
Inductively Coupled DC/DC
Converter Controller (Board #1)
D. Jackson
11/95
2.0 (As built)
2 of 2

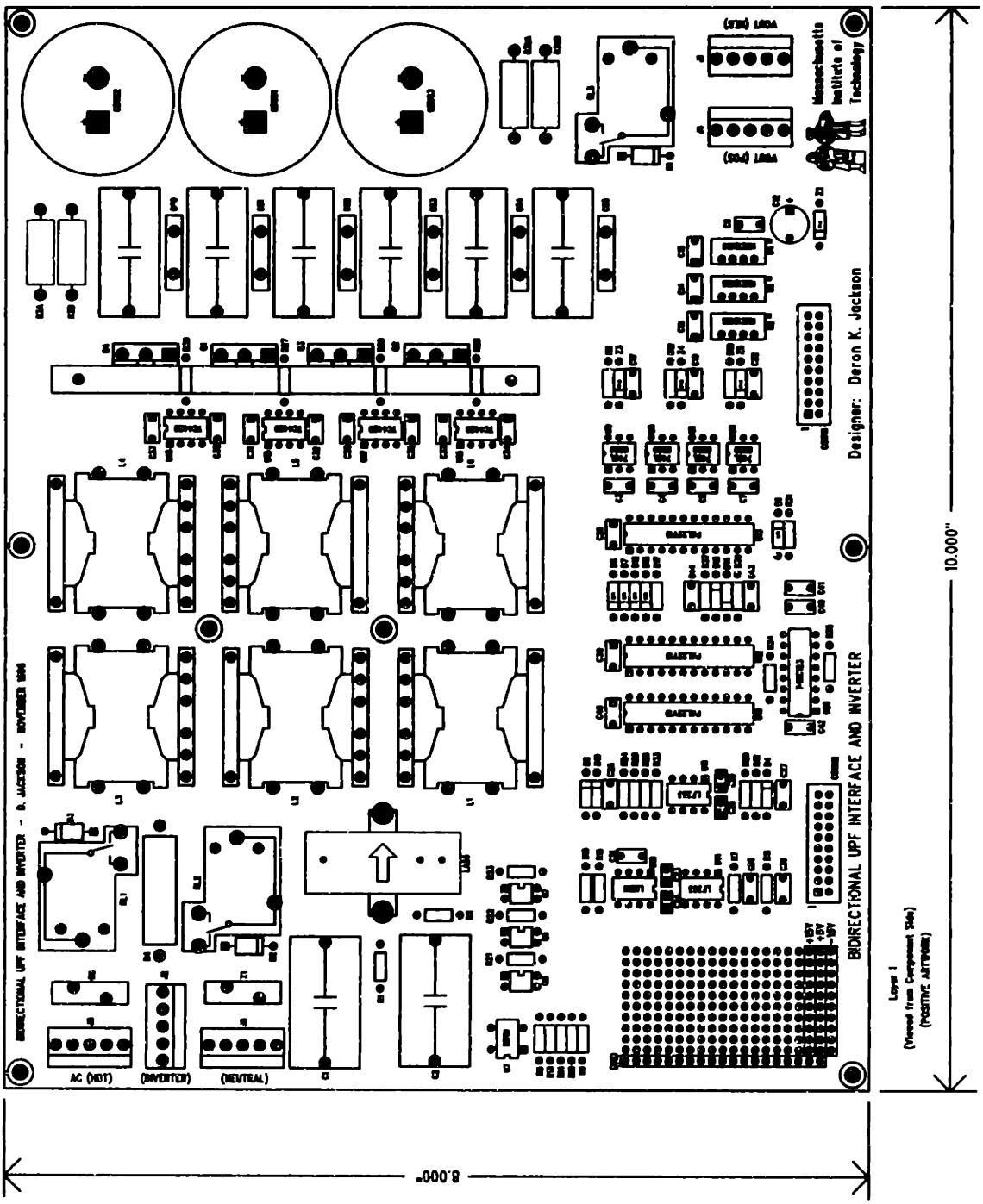

Ramp Generator
Current Sense Filter
Unused Gates
Power Connections
Decoupling Caps
Midpoint Voltage Controller
Voltage/Current Fault Detection
M.I.T. LEES
Inductively Coupled DC/DC Converter Controller (Board #2)
D. Jackson
11/95
2.0 (As built)

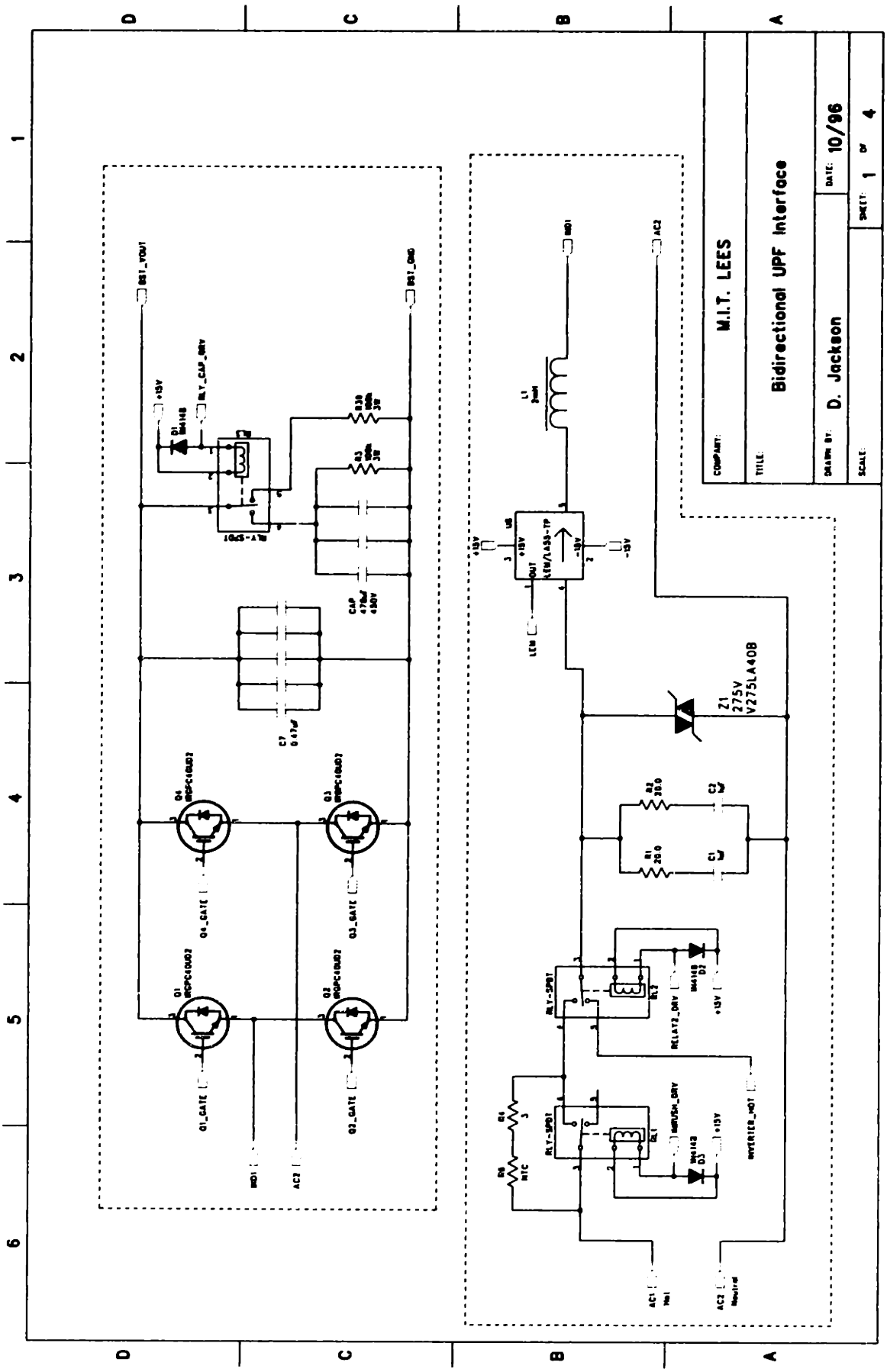

IR Communications
Voltage Reference
Unused Gates
Power Connector
I_SENSE XFRMR
200kHz Oscillator
Connector #1
To DC/DC Board
M.I.T. LEES
Inductively Coupled DC/DC
Converter Controller (Board #2)
D. Jackson
11/95
2.0 (As built)
2 OF 2

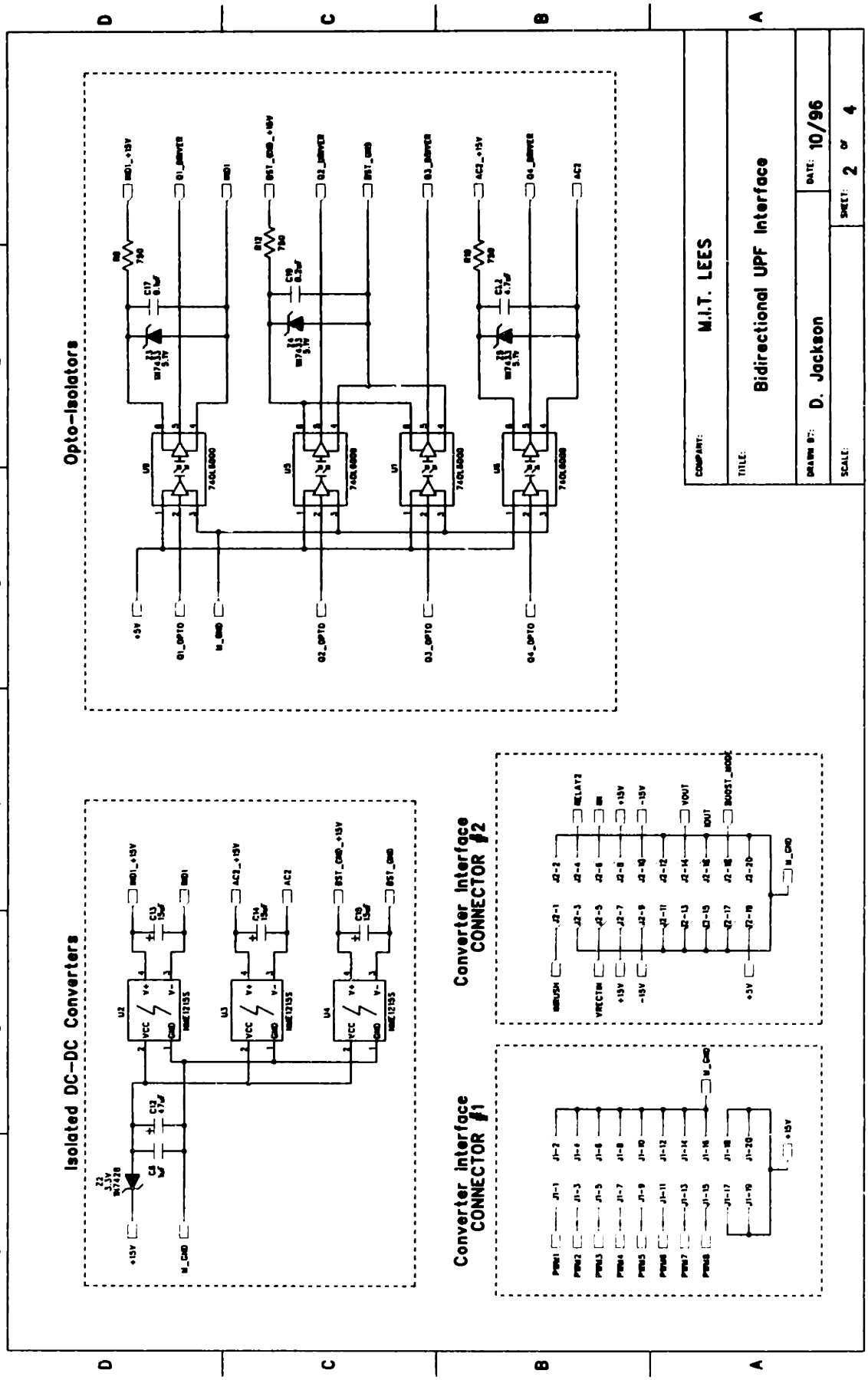
## C.5 Bidirectional Boost-Buck-Inverter

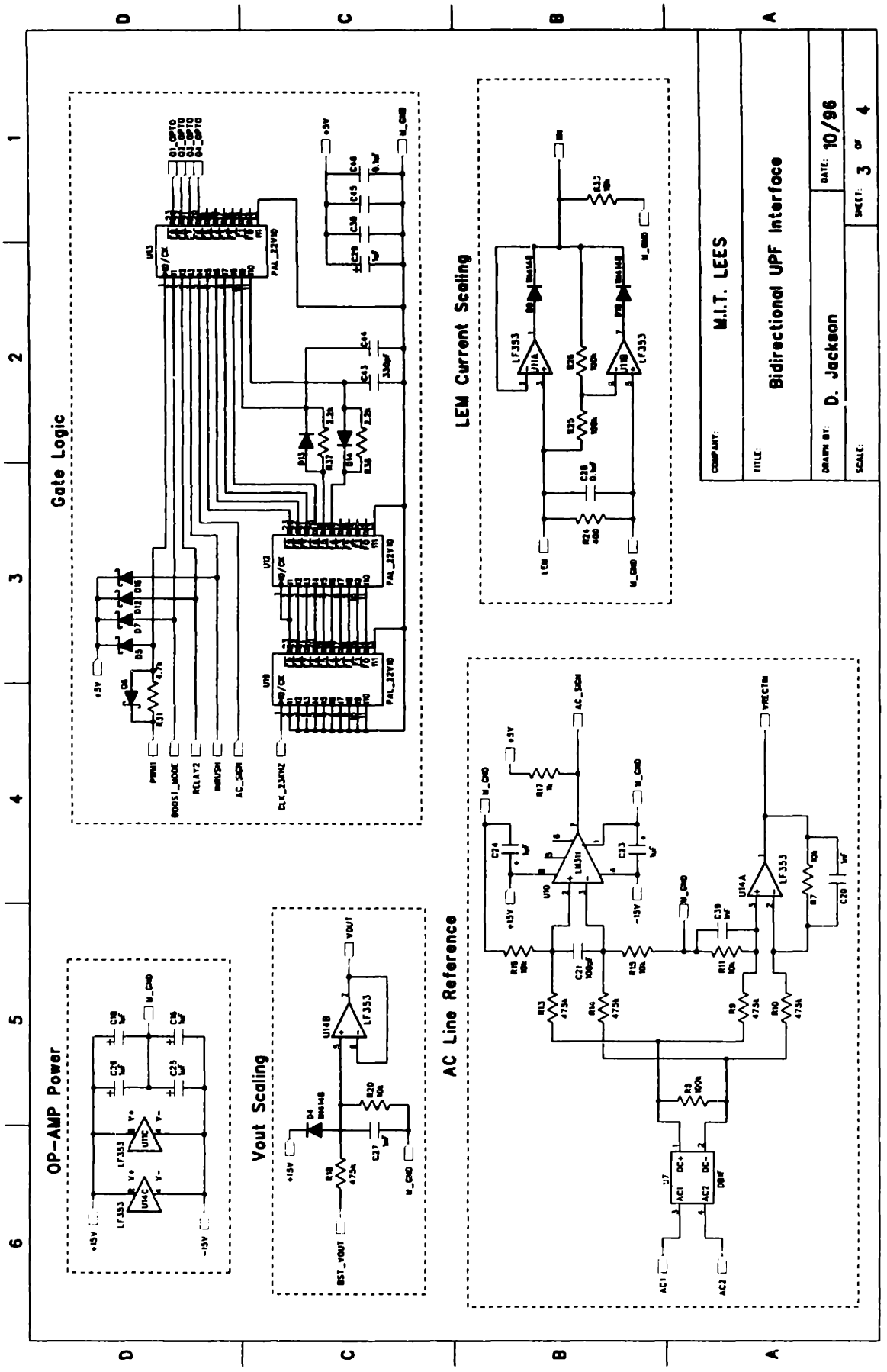
- Photograph
- PCB Layout (4 Layer)
- Schematic Drawings

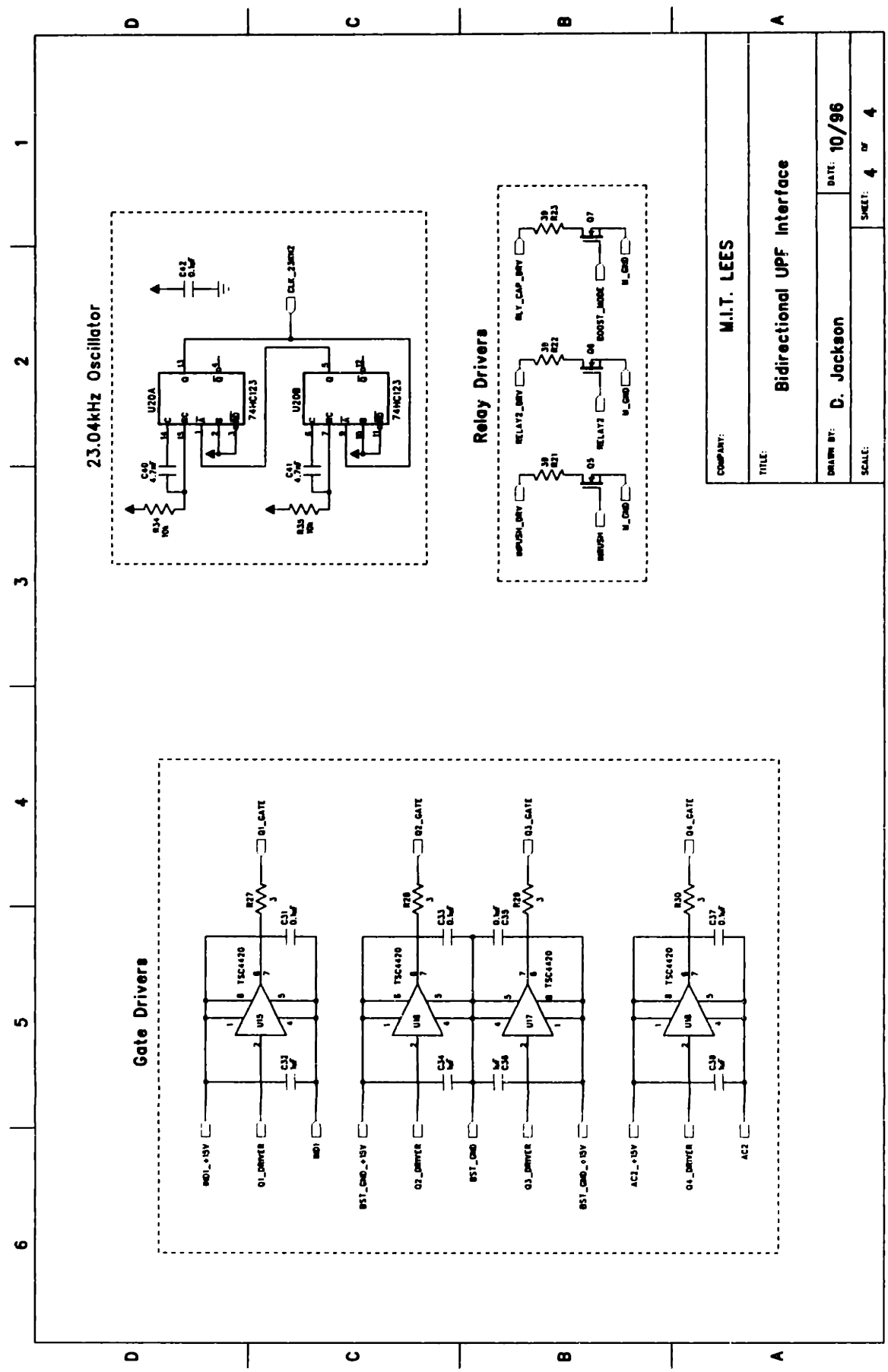





COMPANY: M.I.T. LEES
TITLE: Bidirectional UPF Interface
DRAWN BY: D. Jackson
DATE: 10/96
SCALE:
SHEET: 1 OF 4
Z1
275V
V275LA40B

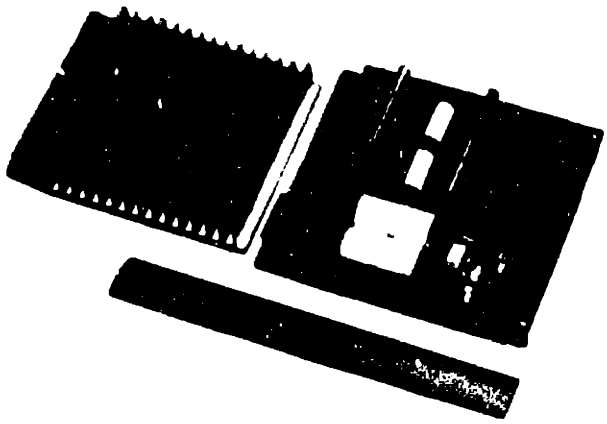

Opto-Isolators
Isolated DC-DC Converters
Converter Interface CONNECTOR #2
Converter Interface CONNECTOR #1
COMPANY: M.I.T. LEES
TITLE: Bidirectional UPF Interface
DRAWN BY: D. Jackson
DATE: 10/96
SCALE:
SHEET: 2 of 4

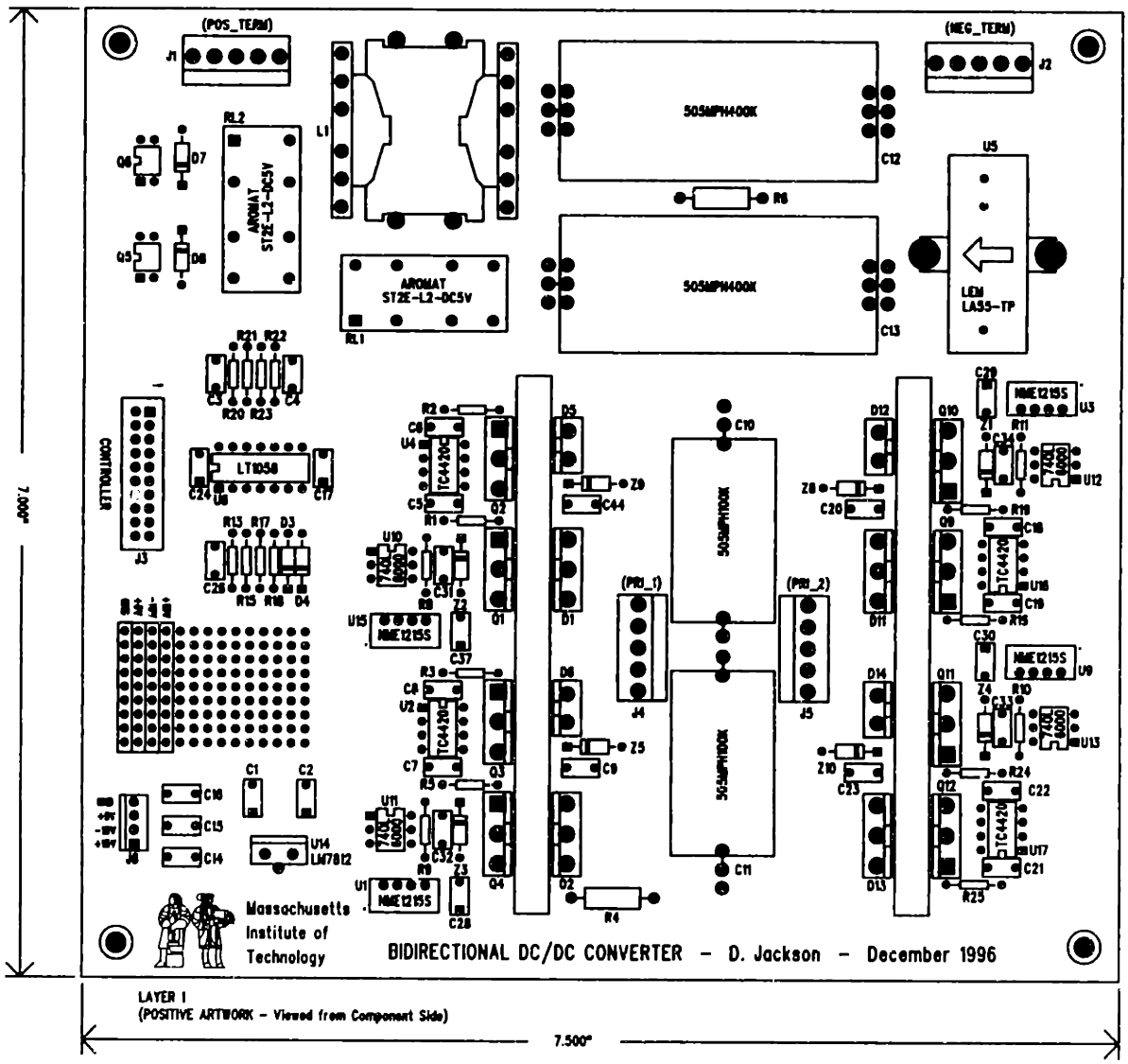

Gate Logic
LEM Current Scaling
OP-AMP Power
Vout Scaling
AC Line Reference
COMPANY: M.I.T. LEES
TITLE: Bidirectional UPF Interface
DRAWN BY: D. Jackson
DATE: 10/96
SCALE:
SHEET: 3 OF 4

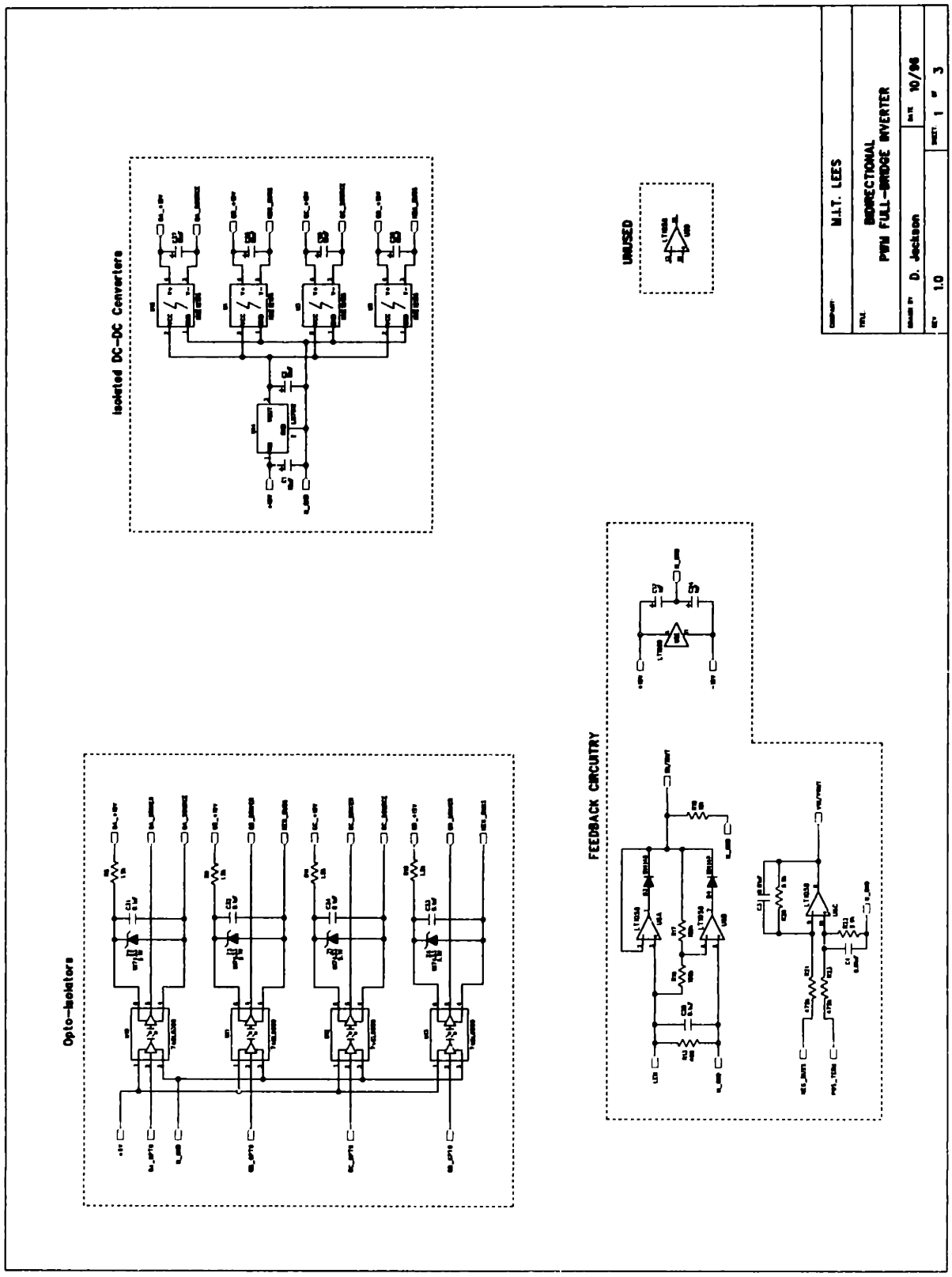

Gate Drivers
23.04kHz Oscillator
Relay Drivers
COMPANY: M.I.T. LEES
TITLE: Bidirectional UPF Interface
DRAWN BY: D. Jackson
DATE: 10/96
SCALE:
SHEET: 4 OF 4

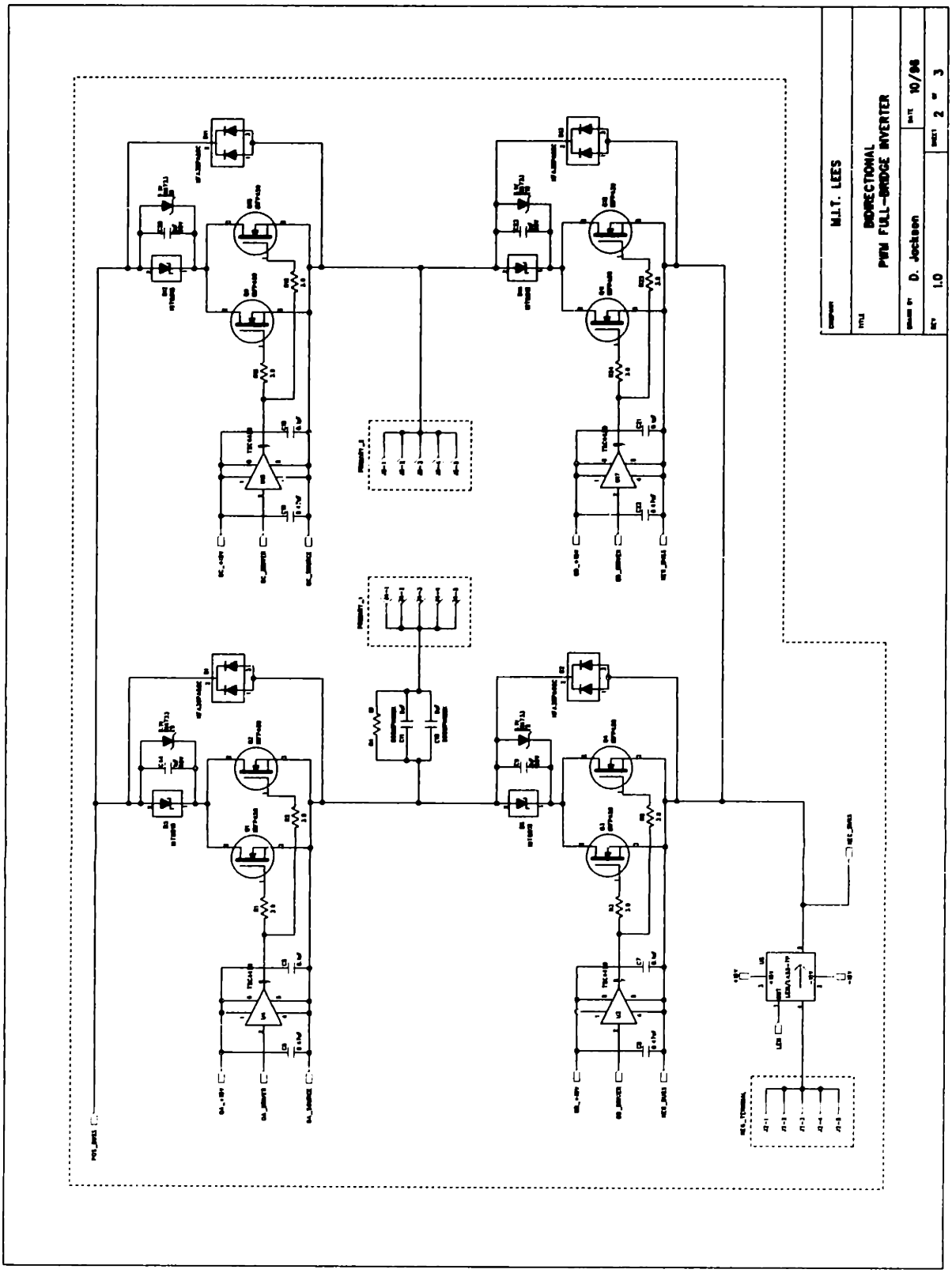
# C.6 Full-Bridge DC/DC Converter

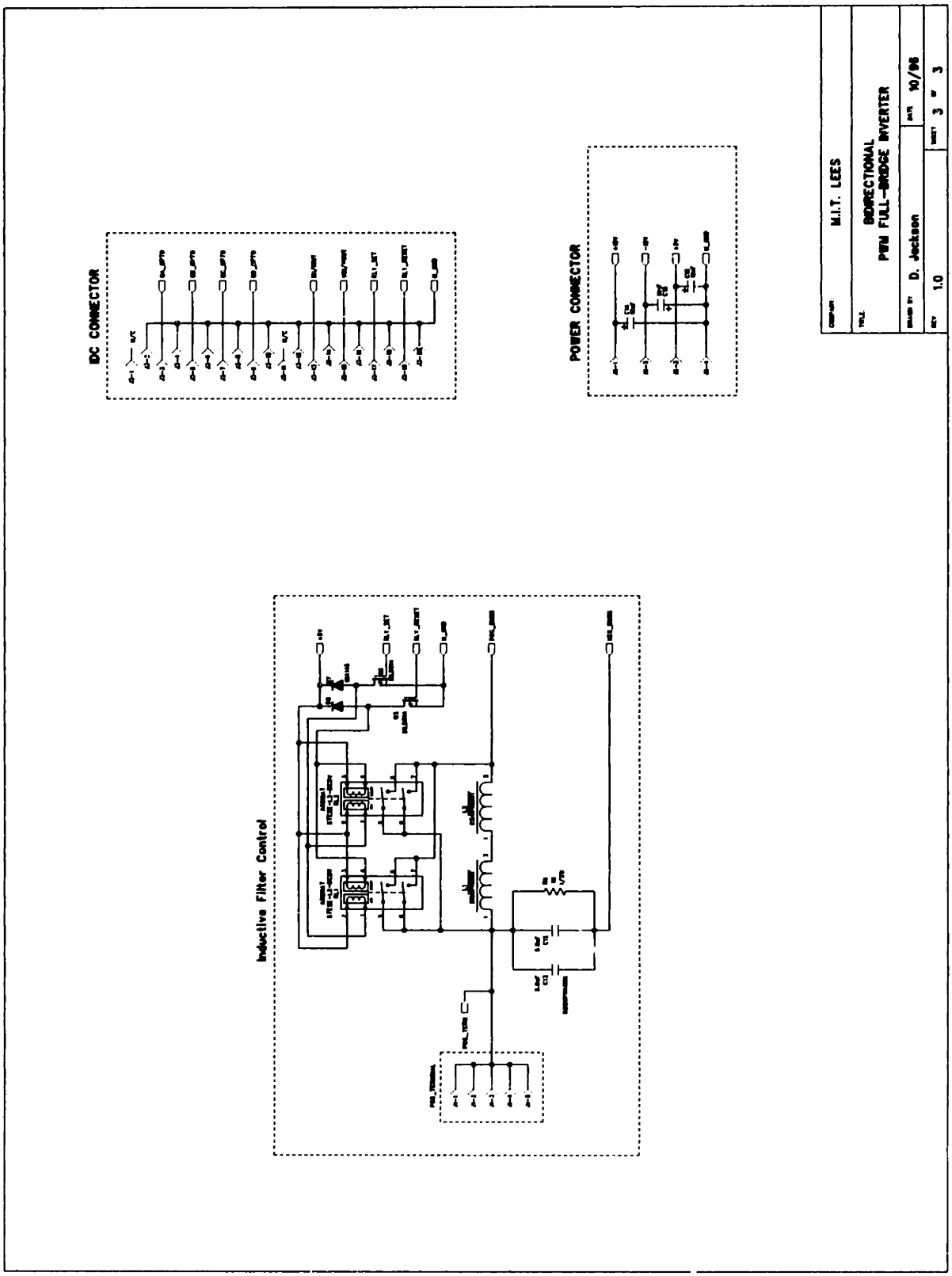
- Photograph
- PCB Layout (4 Layer)
- Schematic Drawings





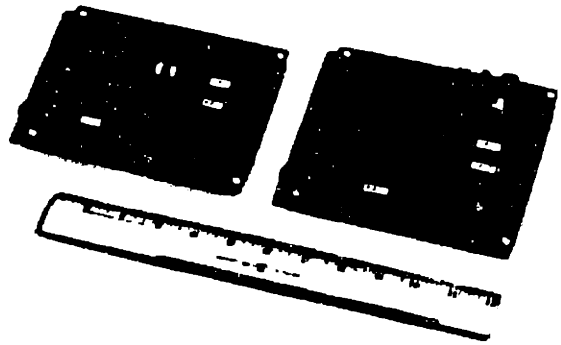

Isolated DC-DC Converters
UNUSED
FEEDBACK CIRCUITRY
Opto-Isolators
M.I.T. LEES
BIDIRECTIONAL
PWM FULL-BRIDGE INVERTER
D. Jackson
10/96
1.0
1 of 3

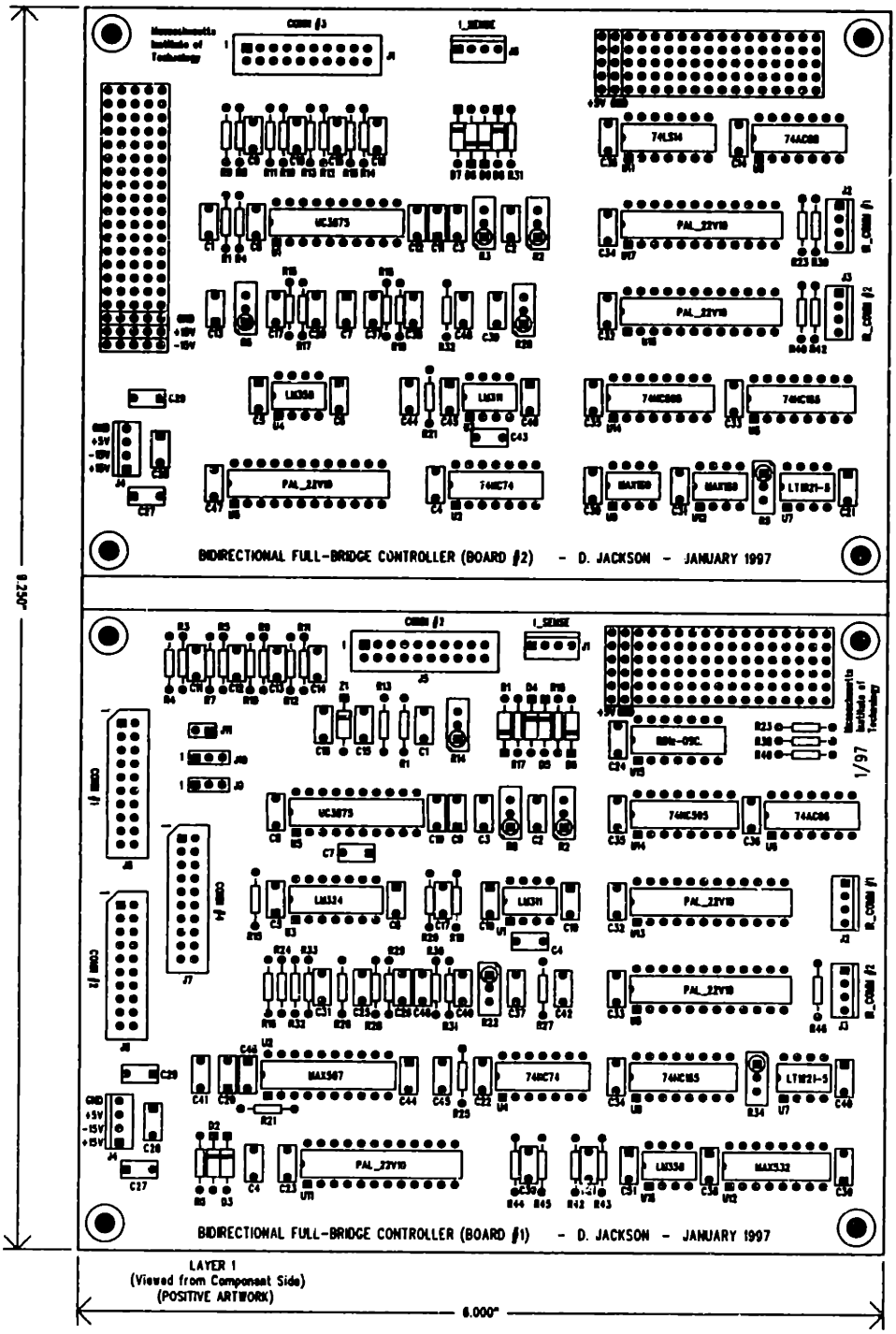

M.I.T. LEES
BIDIRECTIONAL
PWM FULL-BRIDGE INVERTER
D. Jackson
10/96
1.0
2 of 3

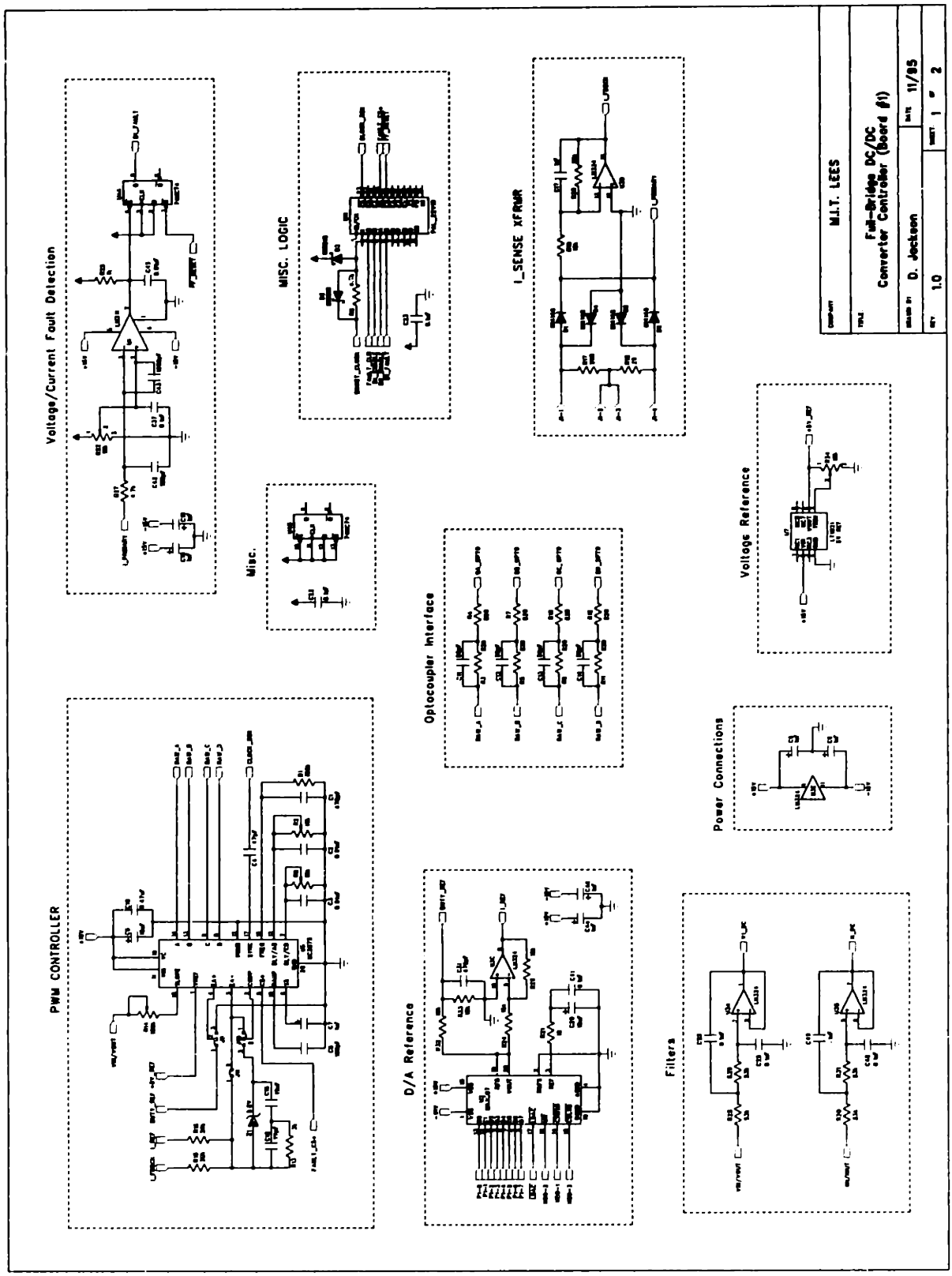

DC CONNECTOR
POWER CONNECTOR
Inductive Filter Control
M.I.T. LEES
BIDIRECTIONAL
PWM FULL-BRIDGE INVERTER
D. Jackson
10/96
1.0
3 of 3

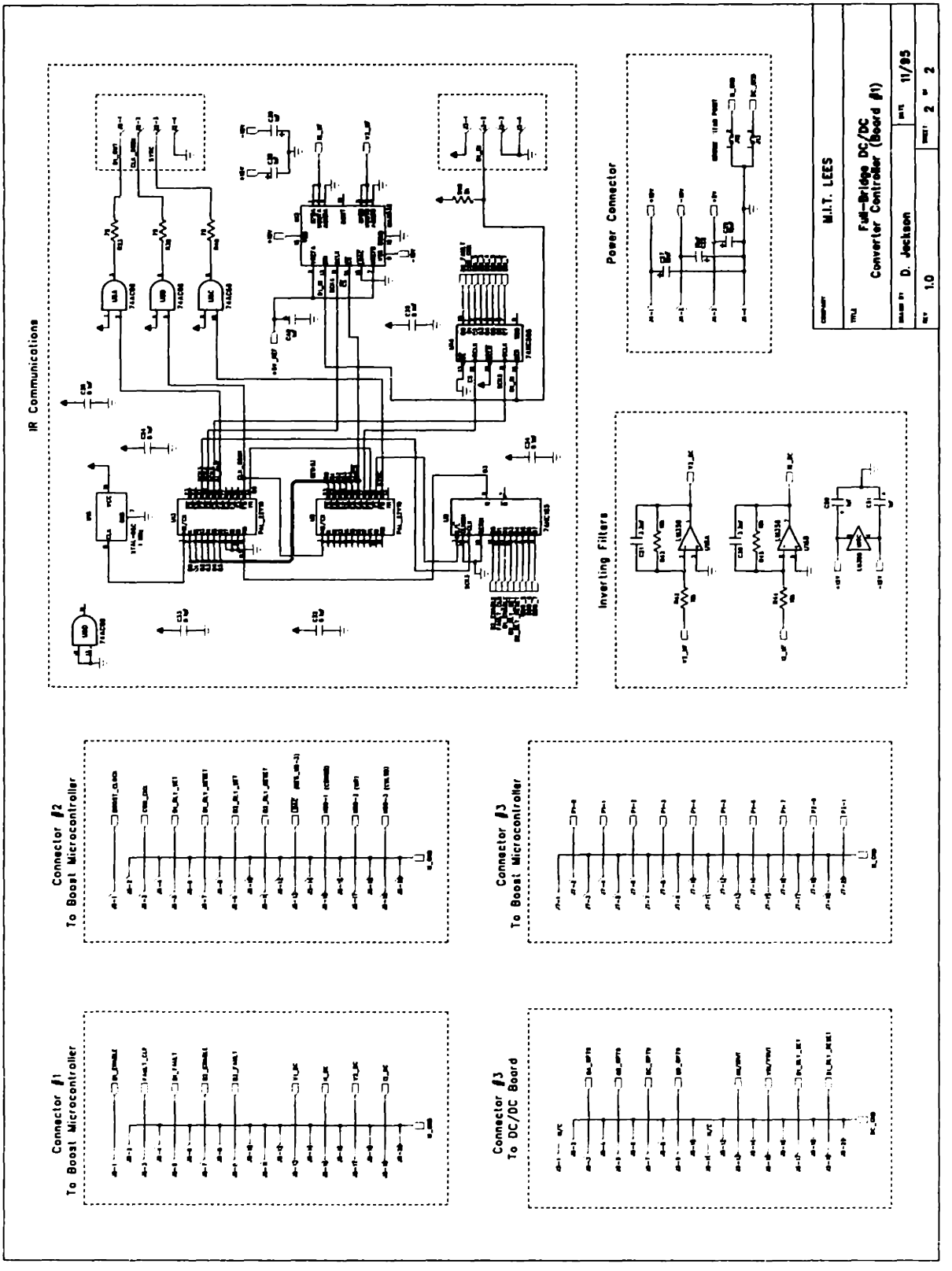
# C.7 Full-Bridge Controllers

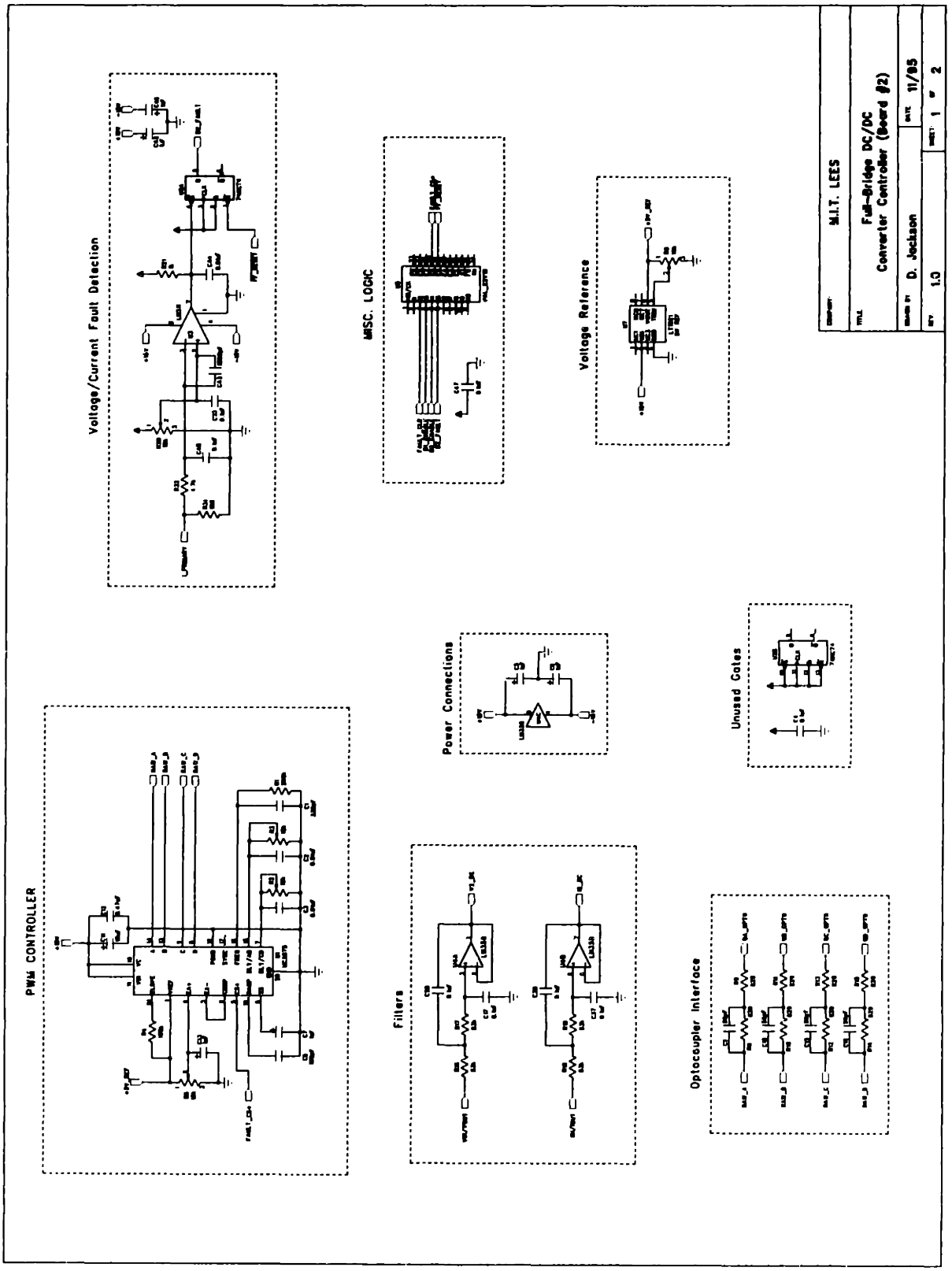
- Photograph
- PCB Layout (2 Layer)
- Schematic Drawings

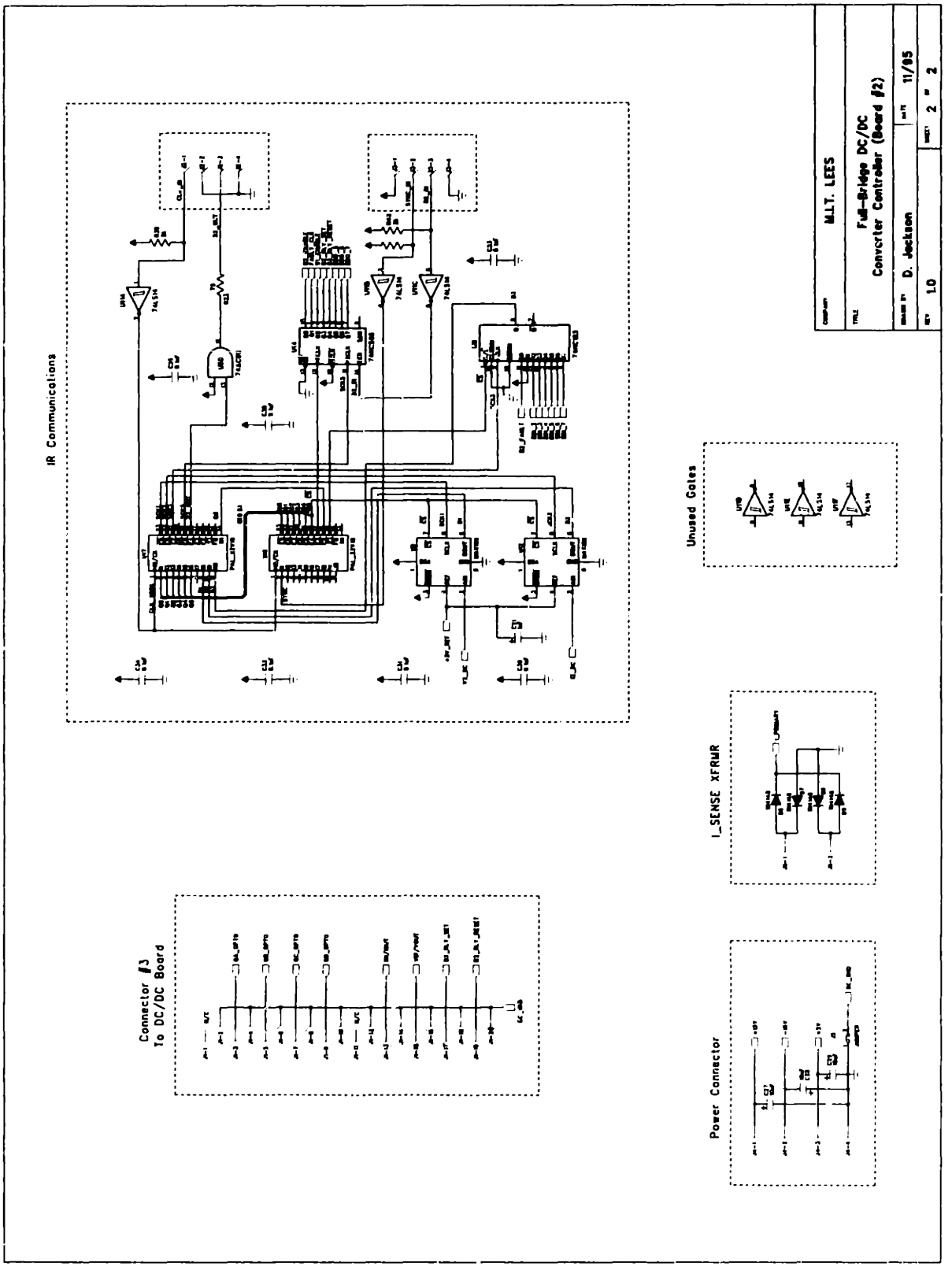





Voltage/Current Fault Detection
MISC. LOGIC
I_SENSE XFRMR
Misc.
Optocoupler Interface
Voltage Reference
Power Connections
PWM CONTROLLER
D/A Reference
Filters
M.I.T. LEES
Full-Bridge DC/DC
Converter Controller (Board #1)
D. Jackson
11/95
1.0
1
2

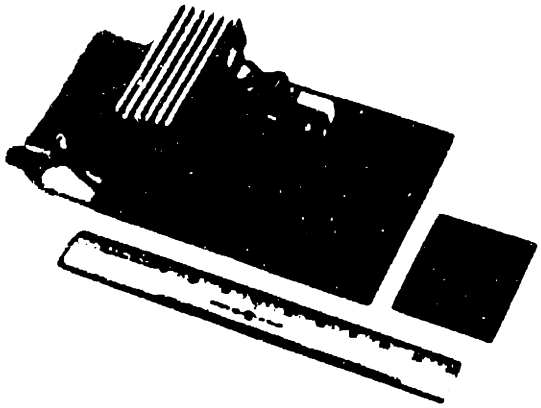

IR Communications
Power Connector
Inverting Filters
Connector #2
To Boost Microcontroller
Connector #3
To Boost Microcontroller
Connector #1
To Boost Microcontroller
Connector #3
To DC/DC Board
M.I.T. LEES
Full-Bridge DC/DC
Converter Controller (Board #1)
D. Jackson
11/95
1.0
2 of 2

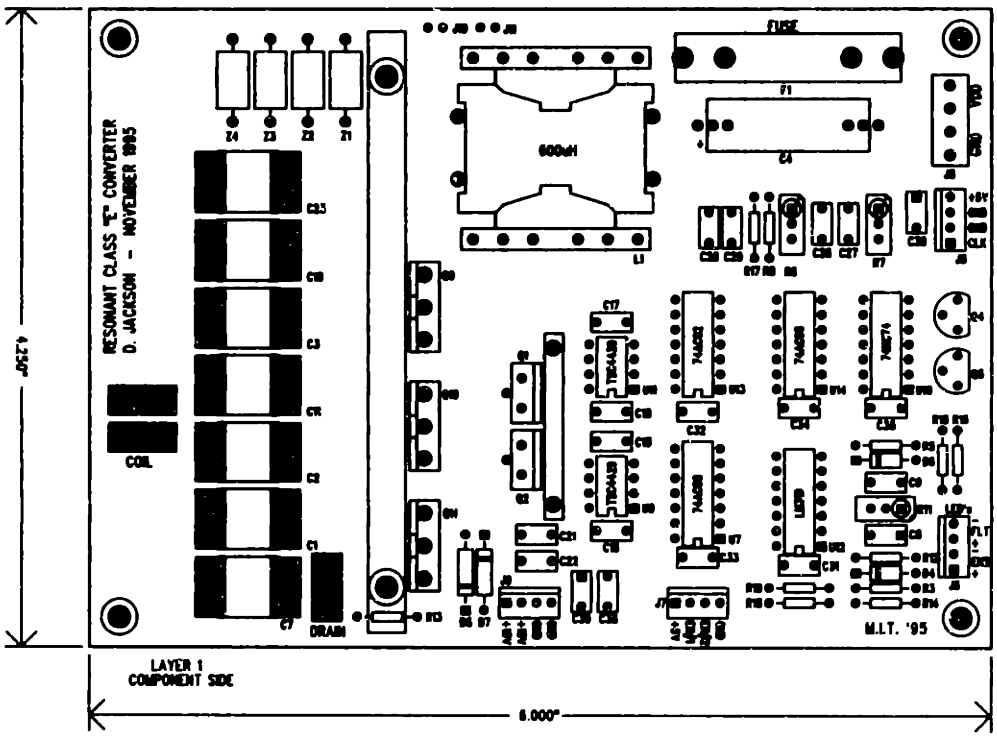

Voltage/Current Fault Detection
MISC. LOGIC
Voltage Reference
PWM CONTROLLER
Power Connections
Unused Gates
Filters
Optocoupler Interface
M.I.T. LEES
Full-Bridge DC/DC
Converter Controller (Board #2)
D. Jackson
11/95
1.0
1
2

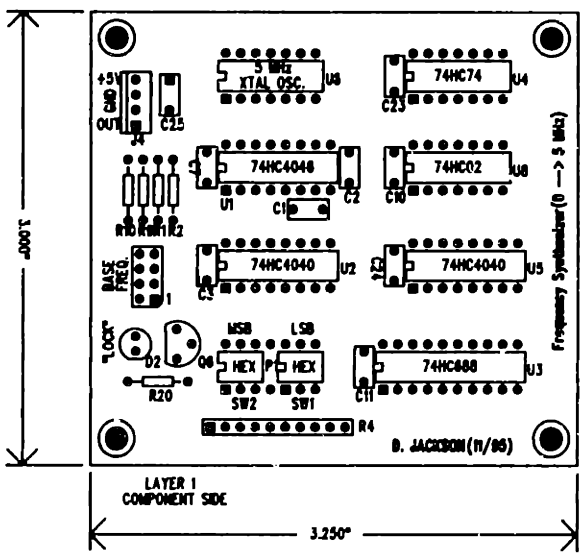

IR Communications
Unused Gates
I_SENSE XFRMR
Power Connector
Connector J3
To DC/DC Board
M.I.T. LEES
Full-Bridge DC/DC
Converter Controller (Board #2)
D. Jackson
11/95
1.0
2 of 2

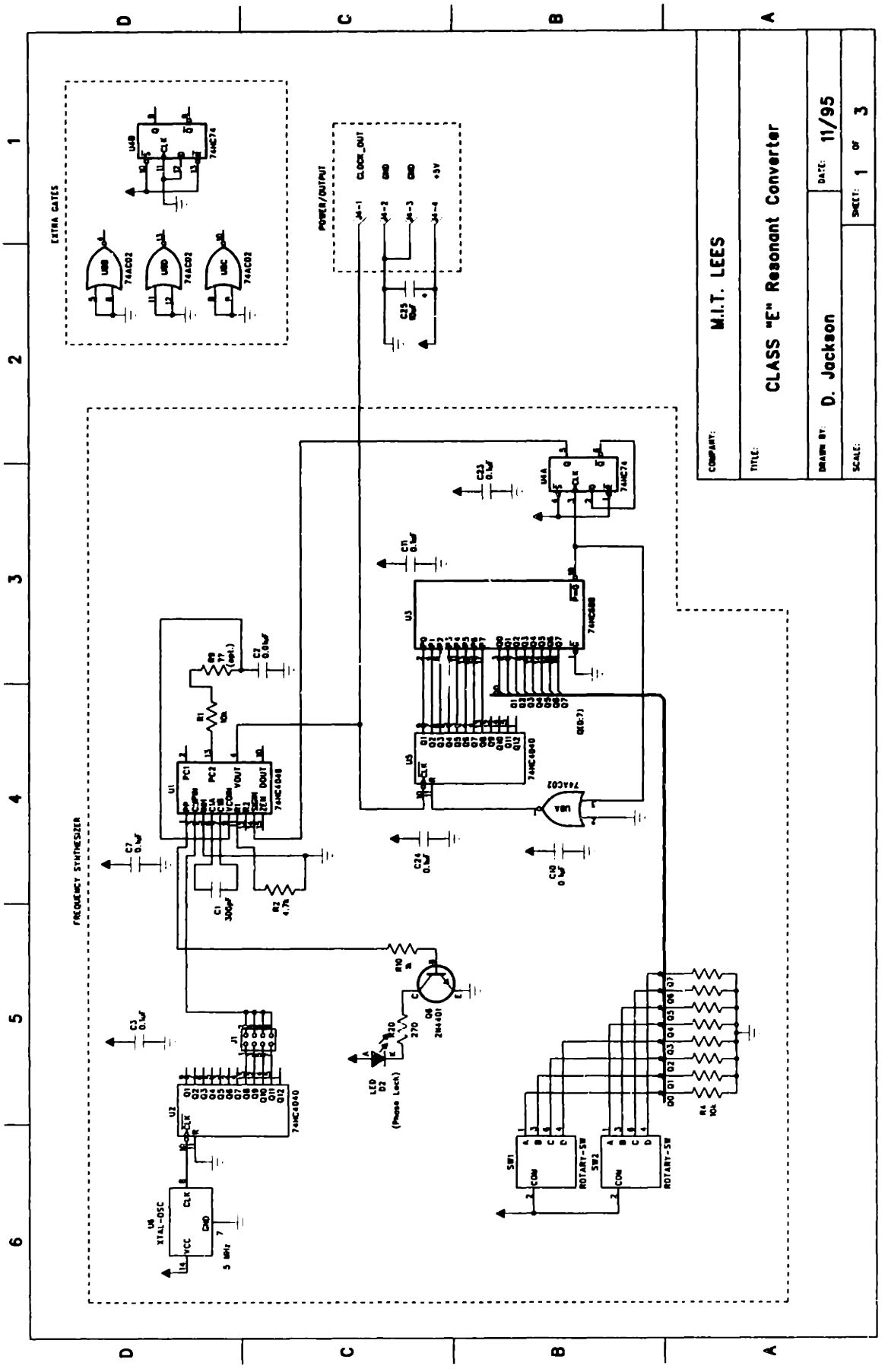
## C.8 3.0-MHz Class-E Induction Heater

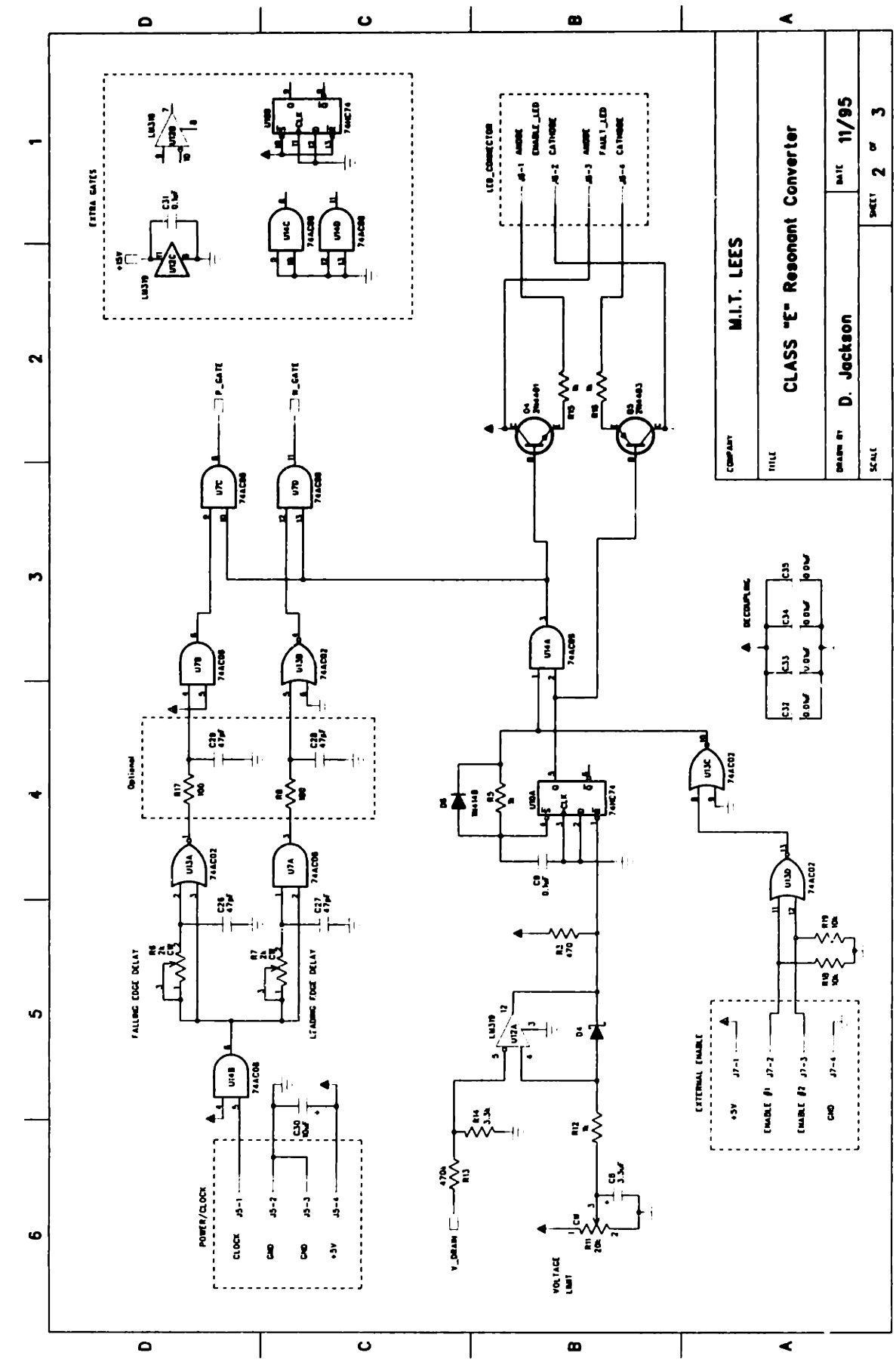
- Photograph
- PCB Layout (2 Layer)
- Schematic Drawings

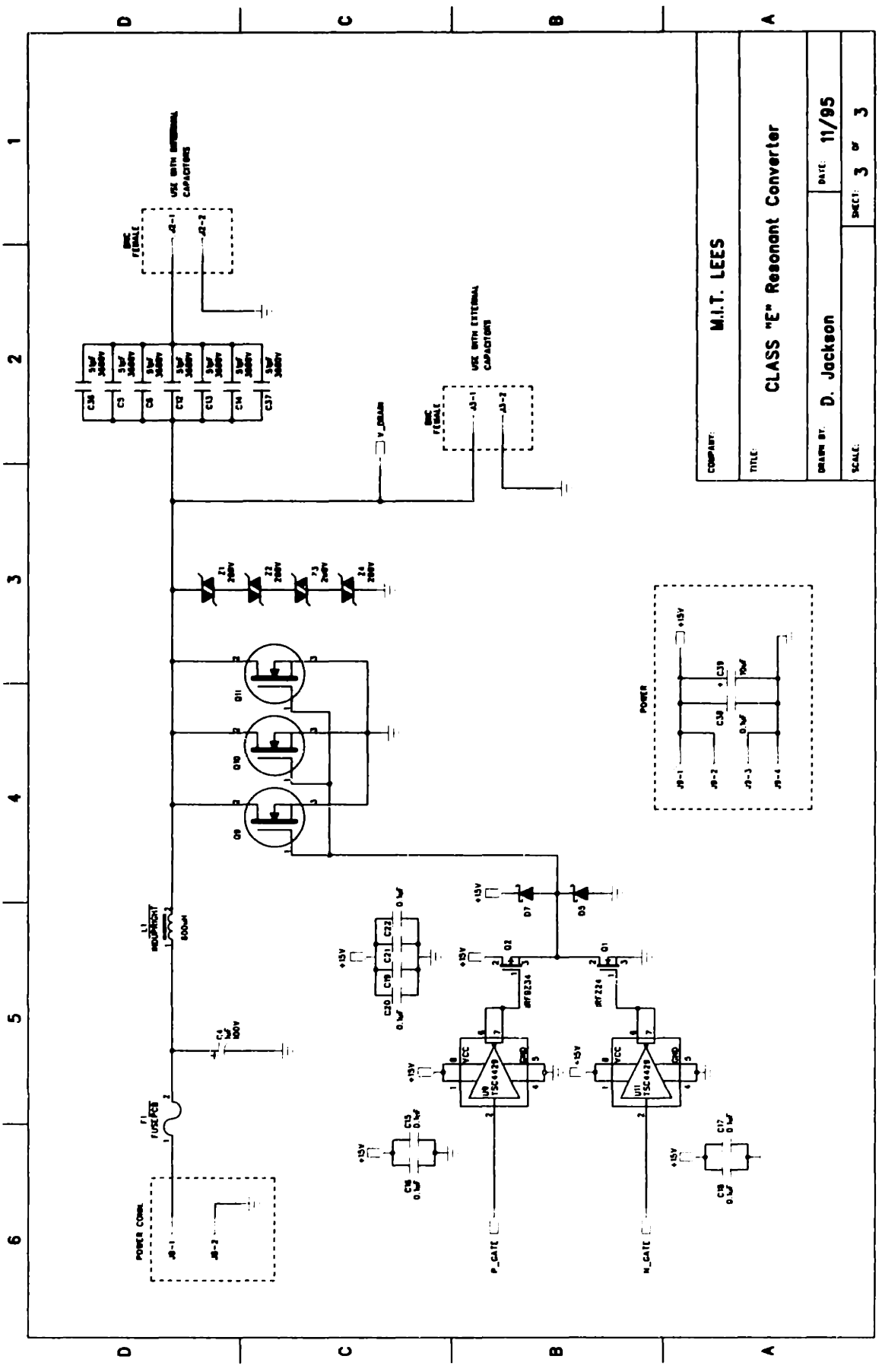




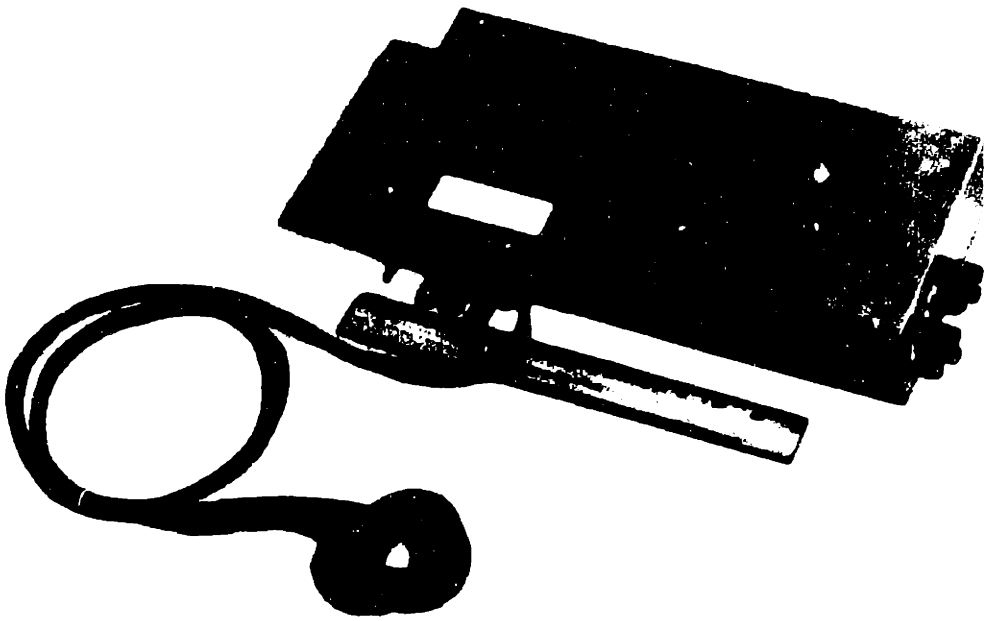

COMPANY:
M.I.T. LEES
TITLE:
CLASS "E" Resonant Converter
DRAWN BY:
D. Jackson
DATE:
11/95
SCALE:
SHEET:
1 OF 3
EXTRA GATES
POWER/OUTPUT
CLOCK_OUT
FREQUENCY SYNTHESIZER
XTAL-OSC
ROTARY-SW
LED
(Phone Lock)


M.I.T. LEES
CLASS "E" Resonant Converter
D. Jackson
11/95
2 of 3
EXTRA GATES
LED_CONNECTOR
ENABLE_LED
FAULT_LED
CATHODE
P_GATE
N_GATE
Optional
DECOUPLING
FALLING EDGE DELAY
LEADING EDGE DELAY
POWER/CLOCK
CLOCK
GND
+5V
EXTERNAL ENABLE
ENABLE #1
ENABLE #2
VOLTAGE LIMIT
V_DRAIN


COMPANY: M.I.T. LEES
TITLE: CLASS "E" Resonant Converter
DRAWN BY: D. Jackson
DATE: 11/95
SCALE:
SHEET: 3 of 3
POWER
P_GATE
N_GATE
POWER COND.

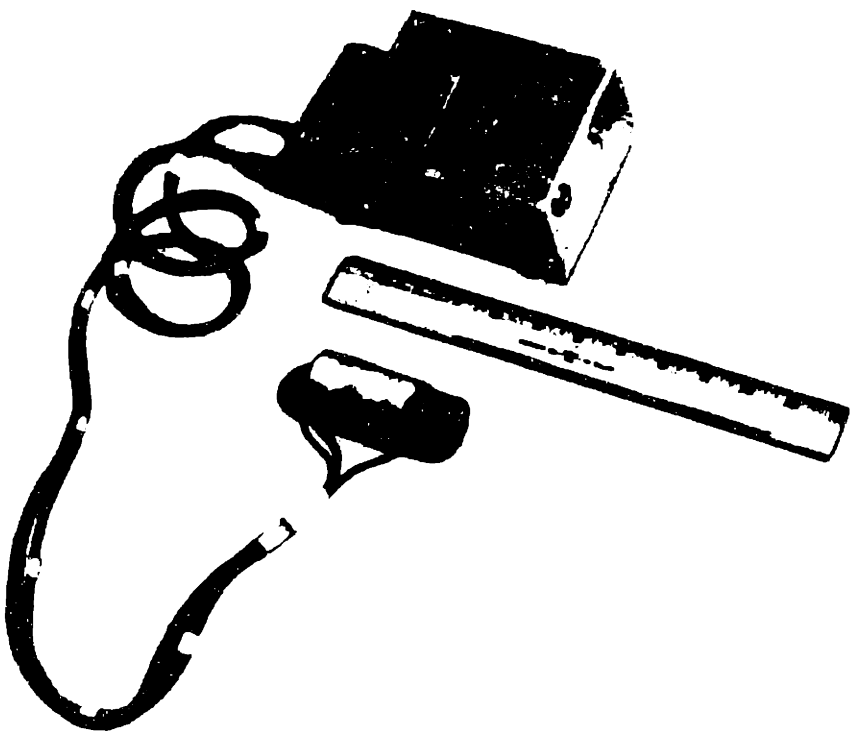
## C.9 Full-Bridge Induction Heater

- **Hand-wired protoboard construction.**



## C.10 Portable 500-kHz Class-E Induction Heater

- **Hand-wired protoboard construction.**



# Appendix D

# Programmable Array Logic

This appendix contains source listings for the programmable-array-logic (PAL) devices used in the hardware prototypes. PALs perform a variety of critical logic functions. In particular, they generate finite-state-machines for event sequencing and they simplify the generation of complex logic equations. The source code in this appendix was developed using the programmable logic compiler CUPL from Logical Devices.

## D.1 PAL Code Listings

The listing DKJ_MEM1 below is for the device U5 in the digital microcontroller schematic. Please see page 248 in Section C.2. This PAL performs memory address decoding for the external EPROM and RAM connected to the 80C196KC.

**PAL Code - DKJ_MEM1.PLD**

```
Name            DKJ_MEM1;
Partno          None;
Date            3/17/96;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;
                                                                        10
/*************************************************************************/
/* This PAL performs some simple bus decode logic for the 80C196       */
/* Boost-Converter microcontroller.  This PAL is to be used when       */
/* the desired program is located on ROM.                              */
/*************************************************************************/

/** INPUT PINS **/

Pin  1          =  clk          ;       /*          */
Pin  2          =  stale        ;       /*          */                 20
Pin  3          =  !hlda        ;       /*          */
Pin  [4..11]    =  [a8..a15]    ;       /*          */
Pin 13          =  !reset       ;       /* RESPIN   */
Pin 22          =  !reset2      ;       /*          */

/** OUTPUT PINS **/

Pin  14         =  !cs510       ;
Pin  15         =  !ce_prom     ;
Pin  16         =  !ce_ram      ;                                       30
```

```
Pin  17       =  !buswidth    ;

/** Logic Equations **/

FIELD memaddr = [a15..8];

eprom = memaddr:[2000..7fff];
ram   = memaddr:[8000..ffff];

cs510    = 'b'0;                                                    40
buswidth = cs510;

ce_prom = eprom;
ce_ram  = ram;
```

The listing BTTN_PAL below is for the device U3 in the DC/DC interface schematic. Please see page 256 in Section C.2. This PAL performs decode and multiplexing functions for the control panel buttons, LEDs, and the DC/DC fault circuitry.

## PAL Code - BTTN_PAL.PLD

```
Name            BTTN_PAL
Partno          None;
Date            3/17/96;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;
                                                                    10
/*****************************************************************/
/* This PAL performs some simple interface logic in order to read the */
/* value of 4 buttons and to trigger an INT on the HSI line.  It also */
/* triggers the HSI on DC/DC fault conditions and latches the values  */
/* of P1.0 -> P1.3 for use as LED drives.                             */
/*                                                                    */
/* Flt1 Flt2 B1 B2 B3 B4 | OUT1 OUT2 CUT3                            */
/* ----------------------+---------------                             */
/*   0    1   x  x  x  x |   0    0    1                              */
/*   1    0   x  x  x  x |   0    1    0                              */   20
/*   1    1   x  x  x  x |   0    1    1                              */
/*   0    0   0  0  0  1 |   1    0    0                              */
/*   0    0   0  0  1  0 |   1    0    1                              */
/*   0    0   0  1  0  0 |   1    1    0                              */
/*   0    0   1  0  0  0 |   1    1    1                              */
/*                       |                                          */
/*       all others      |   0    0    0                              */
/*                       |                                          */
/*****************************************************************/
                                                                    30
/** INPUT PINS **/

pin 1   =   LATCH;      /* This is P2.7 used to latch L1-L4   */
pin 2   =   B1;         /* Button #1                          */
pin 3   =   B2;         /* Button #2                          */
pin 4   =   B3;         /* Button #3                          */
pin 5   =   B4;         /* Button #4                          */
pin 6   =   F1;         /* Fault Line #1                      */
pin 7   =   F2;         /* Fault Line #2                      */
pin 8   =   P10;        /* Port 1-0                           */   40
pin 9   =   P11;        /* Port 1-1                           */
pin 10  =   P12;        /* Port 1-2                           */
pin 11  =   P13;        /* Port 1-3                           */

/** OUTPUT PINS **/

pin 23  =   OUT1;       /* OUTPUT #1   (P2-6)                 */
pin 22  =   OUT2;       /* OUTPUT #2   (P2-4)                 */
pin 21  =   OUT3;       /* OUTPUT #3   (P2-3)                 */
pin 20  =   INT1;       /* Interrupt Output Line              */   50
pin 19  =   L1;         /* LED #1 Output                      */
pin 18  =   L2;         /* LED #2 Output                      */
pin 17  =   L3;         /* LED #3 Output                      */
pin 16  =   L4;         /* LED #4 Output                      */
```

```
/** LOGIC EQUATIONS **/

OUT1 = !B4 & !B3 & B2 & !B1 & !F2 & !F1 #
       !B4 & B3 & !B2 & !B1 & !F2 & !F1 #
       B4 & !B3 & !B2 & !B1 & !F2 & !F1 #
       !B4 & !B3 & !B2 & B1 & !F2 & !F1;

OUT2 = !B4 & !B3 & B2 & !B1 & !F2 #
       F1 #
       !B4 & !B3 & !B2 & B1 & !F2;

OUT3 = !B4 & !B3 & !B2 & B1 & !F1 #
       !B4 & B3 & !B2 & !B1 & !F1#
       F2;

INT1 = F1 #
       F2 #
       B1 #
       B2 #
       B3 #
       B4;

L1.ar  =   'b'0;
L1.sp  =   'b'0;
L1.d   =   !P10;

L2.ar  =   'b'0;
L2.sp  =   'b'0;
L2.d   =   !P11;

L3.ar  =   'b'0;
L3.sp  =   'b'0;
L3.d   =   !P12;

L4.ar  =   'b'0;
L4.sp  =   'b'0;
L4.d   =   !P13;
```

The listings IR_TRNA and IR_TRNB below are for the devices U13 and U8, respectively, on the DC/DC controller boards. Please see the schematics on page 262 in Section C.4 and page 276 in Section C.7. The "IR Transmitter" PAL set controls the optical communications on the charger-side of the prototype inductive coupling.

## PAL Code - IR_TRNA.PLD

```
Name            IR_TRNA;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;

/*************************************************************************/
/* This is PAL U13 on the DC/DC controller board.                        */
/*                                                                       */
/* This PAL set (IR_TRNA/B) generates the various signals to transmit    */
/* and receive 2 12-bit analog channels and 1 8-bit digital channel.     */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK_1MHZ;   /* External 1 MHz Clock                */
pin 2   =   Q0;         /* State output from PAL-B             */
pin 3   =   Q1;         /* State output from PAL-B             */
pin 4   =   Q2;         /* State output from PAL-B             */
pin 5   =   Q3;         /* State output from PAL-B             */
pin 6   =   Q4;         /* State output from PAL-B             */
pin 7   =   CS_NOT;
pin 9   =   D1;         /* Serial DATA from MAX189 A/D         */
pin 10  =   D2;         /* Serial DATA from MAX189 A/D         */
pin 11  =   D3;         /* Serial DATA from 74HC165 Shift-Reg  */
```

```
pin 13  =   Q5;          /* State output from PAL-B                */

/** OUTPUT PINS **/

pin 14  =   NOT_CLK_OUT;
pin 15  =   CLK_500K;
pin 16  =   CLK_250K;
pin 17  =   CLK_125K;

pin 23  =   CLK_OUT;    /* Register for Clock/16  */
pin 22  =   SCK2;
pin 21  =   SCK3;
pin 20  =   SCK4;
pin 19  =   SCK5;
pin 18  =   NOT_D_OUT;

/** LOGIC EQUATIONS **/

CLK_500K.ar  =    'b'0;
CLK_500K.sp  =    'b'0;
CLK_500K.d   =    !CLK_500K;

CLK_250K.ar  =    'b'0;
CLK_250K.sp  =    'b'0;
CLK_250K.d   =    CLK_250K $ CLK_500K;

CLK_125K.ar  =    'b'0;
CLK_125K.sp  =    'b'0;
CLK_125K.d   =    CLK_125K $ CLK_250K;

CLK_OUT.ar   =    'b'0;
CLK_OUT.sp   =    'b'0;
CLK_OUT.d    =    CLK_OUT $ CLK_125K;

D_MSK1 =          !Q3 & Q4 & !Q5 #
                  Q1 & !Q4 & !Q5 #
                  Q2 & !Q4 & !Q5 #
                  Q3 & !Q4 & !Q5 #
                  !Q1 & !Q2 & Q4 & !Q5;

D_MSK2 =          !Q4 & Q5 #
                  Q1 & Q3 & Q4 & !Q5 #
                  Q2 & Q3 & Q4 & !Q5 #
                  !Q1 & !Q2 & !Q3 & Q5;

D_MSK3 =          !Q1 & Q3 & Q4 & Q5 #
                  !Q2 & Q3 & Q4 & Q5 #
                  Q1 & !Q3 & Q4 & Q5 #
                  Q2 & !Q3 & Q4 & Q5 #
                  !Q0 & Q3 & Q4 & Q5;

SCK1 =            Q0 & !Q3 & !Q5 #
                  Q0 & !Q4 & !Q5 #
                  Q0 & !Q1 & !Q2 & !Q5;

SCK2 =            Q0 & !Q4 & Q5 #
                  Q0 & Q3 & Q4 & !Q5 #
                  Q0 & !Q1 & !Q2 & !Q3 & Q5;

SCK3 =            !Q0 & !Q2 & Q4 & Q5 #
                  !Q0 & !Q3 & Q4 & Q5 #
                  !Q0 & !Q1 & Q4 & Q5;

SCK4 =            Q0 & !Q4 & Q5 #
                  Q0 & Q1 & !Q5 #
                  Q0 & Q2 & !Q5 #
                  Q0 & Q3 & !Q5 #
                  Q0 & Q4 & !Q5 #
                  Q0 & !Q1 & !Q2 & !Q3 & Q5;

SCK5 =            Q0 & !Q2 & Q4 & Q5 #
                  Q0 & !Q3 & Q4 & Q5 #
                  Q0 & Q1 & Q2 & Q3 & !Q4 & Q5 #
                  Q0 & !Q1 & Q4 & Q5;

D_OUT =           D1 & D_MSK1 #
                  D2 & D_MSK2 #
                  D3 & D_MSK3;

NOT_D_OUT = !D_OUT;

NOT_CLK_OUT = !CLK_OUT;
```

# PAL Code - IR_TRNB.PLD

```
Name            IR_TRNB;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;
                                                                        10
/*************************************************************************/
/* This is PAL U8 on the DC/DC controller board.                         */
/*                                                                       */
/* This PAL set (IR_TRNA/B) generates the various signals to transmit    */
/* and receive 2 12-bit analog channels and 1 8-bit digital channel.     */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK;        /* External 500 kHz Clock              */       20

/** OUTPUT PINS **/

pin 23  =   Q0;         /* Register Output */
pin 22  =   Q1;         /* Register Output */
pin 21  =   Q2;         /* Register Output */
pin 20  =   Q3;         /* Register Output */
pin 19  =   Q4;         /* Register Output */
pin 18  =   CS_NOT;
pin 17  =   RCLK;                                                       30
pin 16  =   Q5;         /* Register Output */
pin 15  =   SL;
pin 14  =   NOT_SYNC;

/** LOGIC EQUATIONS **/

Q0.ar   =   'b'0;
Q0.sp   =   'b'0;
Q0.d    =   !Q0;
                                                                        40
Q1.ar   =   'b'0;
Q1.sp   =   'b'0;
Q1.d    =   Q0 & !Q1 #
            !Q0 & Q1;

Q2.ar   =   'b'0;
Q2.sp   =   'b'0;
Q2.d    =   !Q1 & Q2 #
            Q0 & Q1 & !Q2 #
            !Q0 & Q2;                                                   50

Q3.ar   =   'b'0;
Q3.sp   =   'b'0;
Q3.d    =   !Q1 & Q3 #
            !Q2 & Q3 #
            Q0 & Q1 & Q2 & !Q3 #
            !Q0 & Q3;

Q4.ar   =   'b'0;
Q4.sp   =   'b'0;                                                       60
Q4.d    =   !Q1 & Q4 #
            !Q2 & Q4 #
            !Q3 & Q4 #
            Q0 & Q1 & Q2 & Q3 & !Q4 #
            !Q0 & Q4;

Q5.ar   =   'b'0;
Q5.sp   =   'b'0;
Q5.d    =   !Q1 & Q5 #
            !Q2 & Q5 #                                                  70
            !Q3 & Q5 #
            !Q4 & Q5 #
            Q0 & Q1 & Q2 & Q3 & Q4 & !Q5 #
            !Q0 & Q5;

CS_NOT  =   !Q1 & Q3 & Q4 & Q5 #
            !Q2 & Q3 & Q4 & Q5 #
            Q0 & Q1 & !Q3 & Q4 & Q5 #
            Q2 & !Q3 & Q4 & Q5 #
            !Q0 & Q3 & Q4 & Q5;                                         80

SL =        !Q1 & Q4 & Q5 #
```

```
          !Q2 & Q4 & Q5 #
          !Q3 & Q4 & Q5 #
          Q1 & Q2 & Q3 & !Q4 & Q5 #
          !Q0 & Q4 & Q5;

RCLK =    Q0 & Q1 & Q2 & Q3 & Q4 & Q5;

SYNC =    Q0 & Q1 & Q2 & Q3 & Q4 & Q5;

NOT_SYNC = !SYNC;
```

The listings IR_RCVA and IR_RCVB below are for the devices U17 and U16, respectively, on the DC/DC controller boards. Please see the schematic on page 264 in Section C.4 and the schematic on page 278 in Section C.7. The "IR Receiver" PAL set controls the optical communications on the vehicle-side of the prototype inductive coupling.

## PAL Code - IR_RCVA.PLD

```
Name            IR_RCVA;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;

/*************************************************************************/
/* This is PAL U17 on the DC/DC controller board.                        */
/*                                                                       */
/* This PAL set (IR_RCVA/B) generates the various signals to transmit  */
/* and receive 2 12-bit analog channels and 1 8-bit digital channel.   */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK;        /* External 125 kHz Clock            */
pin 2   =   Q0;         /* State output from PAL-B           */
pin 3   =   Q1;         /* State output from PAL-B           */
pin 4   =   Q2;         /* State output from PAL-B           */
pin 5   =   Q3;         /* State output from PAL-B           */
pin 6   =   Q4;         /* State output from PAL-B           */
pin 7   =   CS_NOT;
pin 9   =   D1;         /* Serial DATA from MAX189 A/D       */
pin 10  =   D2;         /* Serial DATA from MAX189 A/D       */
pin 11  =   D3;         /* Serial DATA from 74HC165 Shift-Reg */
pin 13  =   Q5;         /* State output from PAL-B           */

/** OUTPUT PINS **/

pin 23  =   SCK1;
pin 22  =   SCK2;
pin 21  =   SCK3;
pin 20  =   SCK4;
pin 19  =   SCK5;
pin 18  =   D_OUT;

/** LOGIC EQUATIONS **/

D_MSK1 =        !Q3 & Q4 & !Q5 #
                Q1 & !Q4 & !Q5 #
                Q2 & !Q4 & !Q5 #
                Q3 & !Q4 & !Q5 #
                !Q1 & !Q2 & Q4 & !Q5;

D_MSK2 =        !Q4 & Q5 #
                Q1 & Q3 & Q4 & !Q5 #
                Q2 & Q3 & Q4 & !Q5 #
                !Q1 & !Q2 & !Q3 & Q5;
```

```
D_MSK3 =        !Q1 & Q3 & Q4 & Q5 #
                !Q2 & Q3 & Q4 & Q5 #
                Q1 & !Q3 & Q4 & Q5 #
                Q2 & !Q3 & Q4 & Q5 #
                !Q0 & Q3 & Q4 & Q5;

SCK1 =          Q0 & !Q3 & !Q5 #
                Q0 & !Q4 & !Q5 #
                Q0 & !Q1 & !Q2 & !Q5;

SCK2 =          Q0 & !Q4 & Q5 #
                Q0 & Q3 & Q4 & !Q5 #
                Q0 & !Q1 & !Q2 & !Q3 & Q5;

SCK3 =          !Q0 & !Q2 & Q4 & Q5 #
                !Q0 & !Q3 & Q4 & Q5 #
                !Q0 & !Q1 & Q4 & Q5;

SCK4 =          Q0 & !Q4 & Q5 #
                Q0 & Q1 & !Q5 #
                Q0 & Q2 & !Q5 #
                Q0 & Q3 & !Q5 #
                Q0 & Q4 & !Q5 #
                Q0 & !Q1 & !Q2 & !Q3 & Q5;

SCK5 =          Q0 & !Q2 & Q4 & Q5 #
                Q0 & !Q3 & Q4 & Q5 #
                Q0 & Q1 & Q2 & Q3 & !Q4 & Q5 #
                Q0 & !Q1 & Q4 & Q5;

D_OUT =         D1 & D_MSK1 #
                D2 & D_MSK2 #
                D3 & D_MSK3;
```

## PAL Code - IR_RCVB.PLD

```
Name            IR_RCVB;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;

/*************************************************************************/
/* This is PAL U16 on the DC/DC controller board.                        */
/*                                                                       */
/* This PAL set (IR_RCVA/B) generates the various signals to transmit  */
/* and receive 2 12-bit analog channels and 1 8-bit digital channel.   */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK;        /* External 500 kHz Clock  */
pin 2   =   SYNC;       /* External SYNC signal    */

/** OUTPUT PINS **/

pin 23  =   Q0;       /* Register Output */
pin 22  =   Q1;       /* Register Output */
pin 21  =   Q2;       /* Register Output */
pin 20  =   Q3;       /* Register Output */
pin 19  =   Q4;       /* Register Output */
pin 18  =   CS_NOT;
pin 17  =   RCLK;
pin 16  =   Q5;       /* Register Output */
pin 15  =   SL;

/** LOGIC EQUATIONS **/

Q0.ar   =   SYNC;
Q0.sp   =   'b'0;
Q0.d    =   !Q0;

Q1.ar   =   SYNC;
Q1.sp   =   'b'0;
Q1.d    =   Q0 & !Q1 #
            !Q0 & Q1;
```

```
Q2.ar  =   SYNC;
Q2.sp  =   'b'0;
Q2.d   =   !Q1 & Q2 #
           Q0 & Q1 & !Q2 #
           !Q0 & Q2;

Q3.ar  =   SYNC;
Q3.sp  =   'b'0;
Q3.d   =   !Q1 & Q3 #
           !Q2 & Q3 #
           Q0 & Q1 & Q2 & !Q3 #
           !Q0 & Q3;

Q4.ar  =   SYNC;
Q4.sp  =   'b'0;
Q4.d   =   !Q1 & Q4 #
           !Q2 & Q4 #
           !Q3 & Q4 #
           Q0 & Q1 & Q2 & Q3 & !Q4 #
           !Q0 & Q4;

Q5.ar  =   SYNC;
Q5.sp  =   'b'0;
Q5.d   =   !Q1 & Q5 #
           !Q2 & Q5 #
           !Q3 & Q5 #
           !Q4 & Q5 #
           Q0 & Q1 & Q2 & Q3 & Q4 & !Q5 #
           !Q0 & Q5;

CS_NOT =   !Q1 & Q3 & Q4 & Q5 #
           !Q2 & Q3 & Q4 & Q5 #
           Q0 & Q1 & !Q3 & Q4 & Q5 #
           Q2 & !Q3 & Q4 & Q5 #
           !Q0 & Q3 & Q4 & Q5;

SL =       !Q1 & Q4 & Q5 #
           !Q2 & Q4 & Q5 #
           !Q3 & Q4 & Q5 #
           Q1 & Q2 & Q3 & !Q4 & Q5 #
           !Q0 & Q4 & Q5;

RCLK =     SYNC;
```

The listings FB_CTR1 and FB_CTR2 below are for the devices U11 and U5, respectively, on the full-bridge DC/DC controller boards. Please see the schematics on pages 275 and 277 in Section C.7. These PALs perform miscellaneous logic functions.

## PAL Code - FB_CTR1.PLD

```
Name            FB_CTR1;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;

/*************************************************************************/
/* This PAL performs misc logic on the Full-Bridge Controller board #1.  */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   BOOST_CLOCK;    /* Clock from boost board ~200kHz */
pin 3   =   FAULT_CLR;
pin 4   =   B1_ENABLE;
pin 5   =   B2_ENABLE;
pin 6   =   B1_FAULT;

/** OUTPUT PINS **/

pin 23  =   CLOCK_OUT;
```

```
pin 20  =   FAULT_CS;
pin 19  =   NOT_FF_RESET;

/** LOGIC EQUATIONS **/

/** CLOCK_OUT.ar = 'b'0;          **/
/** CLOCK_OUT.sp = 'b'0;          **/
/** CLOCK_OUT.d  = !CLOCK_100K;   **/

CLOCK_OUT = BOOST_CLOCK;

FAULT_CS = B1_FAULT #
           B2_ENABLE #
           !B1_ENABLE;

NOT_FF_RESET = !FAULT_CLR;
```

## PAL Code - FB_CTR2.PLD

```
Name            FB_CTR2;
Partno          None;
Date            6/3/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;

/*************************************************************************/
/* This PAL performs misc logic on the Full-Bridge Controller board #2.  */
/*************************************************************************/

/** INPUT PINS **/

pin 3   =   FAULT_CLR;
pin 4   =   B1_ENABLE;
pin 5   =   B2_ENABLE;
pin 6   =   B2_FAULT;

/** OUTPUT PINS **/

pin 20  =   FAULT_CS;
pin 19  =   NOT_FF_RESET;

/** LOGIC EQUATIONS **/

FAULT_CS = B2_FAULT #
           B1_ENABLE #
           !B2_ENABLE;

NOT_FF_RESET = !FAULT_CLR;
```

The listings BIDIR1L and BIDIR2L below are for the devices U19 and U12, respectively, on the bidirectional boost-buck-inverter board. Please see the schematic on page 268 in Section C.5. These PALs make up a finite-state-machine, which generates the 1024-bit "magic sinewave" sequence used during AC-inverter operation.

## PAL Code - BIDIR1L.PLD

```
Name            BIDIR1L;
Partno          None;
Date            10/14/96;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
```

```
Device          P22V10;
                                                                              10
/*****************************************************************************/
/* This pal is part U19 on the bidirectional boost/buck/inverter board.  */
/* PALS BIDIR1L and BIDIR2L generate the 1024-bit Magic Sinewave.        */
/*****************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK;        /* External 23.04 kHz Clock  */

/** OUTPUT PINS **/                                                          20

pin 23  =   TOGGLE;
pin 22  =   Q8;
pin 21  =   Q7;
pin 20  =   Q6;
pin 19  =   Q5;
pin 18  =   Q4;
pin 17  =   Q3;
pin 16  =   Q2;
pin 15  =   Q1;                                                              30
pin 14  =   Q0;

/** LOGIC EQUATIONS **/

Q0.ar   =   'b'0;
Q0.sp   =   'b'0;
Q0.d    =   !Q0;

Q1.ar   =   'b'0;
Q1.sp   =   'b'0;                                                            40
Q1.d    =   Q0 & !Q1 #
            !Q0 & Q1;

Q2.ar   =   'b'0;
Q2.sp   =   'b'0;
Q2.d    =   !Q1 & Q2 #
            Q0 & Q1 & !Q2 #
            !Q0 & Q2;

Q3.ar   =   'b'0;                                                            50
Q3.sp   =   'b'0;
Q3.d    =   !Q1 & Q3 #
            !Q2 & Q3 #
            Q0 & Q1 & Q2 & !Q3 #
            !Q0 & Q3;

Q4.ar   =   'b'0;
Q4.sp   =   'b'0;
Q4.d    =   !Q1 & Q4 #
            !Q2 & Q4 #                                                       60
            !Q3 & Q4 #
            Q0 & Q1 & Q2 & Q3 & !Q4 #
            !Q0 & Q4;

Q5.ar   =   'b'0;
Q5.sp   =   'b'0;
Q5.d    =   !Q1 & Q5 #
            !Q2 & Q5 #
            !Q3 & Q5 #
            !Q4 & Q5 #                                                       70
            Q0 & Q1 & Q2 & Q3 & Q4 & !Q5 #
            !Q0 & Q5;

Q6.ar   =   'b'0;
Q6.sp   =   'b'0;
Q6.d    =   !Q1 & Q6 #
            !Q2 & Q6 #
            !Q3 & Q6 #
            !Q4 & Q6 #
            !Q5 & Q6 #                                                       80
            Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & !Q6 #
            !Q0 & Q6;

Q7.ar   =   'b'0;
Q7.sp   =   'b'0;
Q7.d    =   !Q1 & Q7 #
            !Q2 & Q7 #
            !Q3 & Q7 #
            !Q4 & Q7 #
            !Q5 & Q7 #                                                       90
            !Q6 & Q7 #
            Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6 & !Q7 #
            !Q0 & Q7;
```

```
Q8.ar  =    'b'0;
Q8.sp  =    'b'0;
Q8.d   =    !Q1 & Q8 #
            !Q2 & Q8 #
            !Q3 & Q8 #
            !Q4 & Q8 #                                                    100
            !Q5 & Q8 #
            !Q6 & Q8 #
            !Q7 & Q8 #
            Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & Q6 & Q7 & !Q8 #
            !Q0 & Q8;

TOGGLE.ar = 'b'0;
TOGGLE.sp = 'b'0;
TOGGLE.d =  !Q0 & !Q1 & !Q2 & !Q3 & !Q4 & !Q5 & !Q6 & !Q7 & !Q8;
```

## PAL Code - BIDIR2L.PLD

```
Name            BIDIR2L;
Partno          None;
Date            10/14/96;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;                                                   10

/*************************************************************************/
/* This pal is part U12 on the bidirectional boost/buck/inverter board.  */
/* PALS BIDIR1L and BIDIR2L generate the 1024-bit Magic Sinewave.        */
/*************************************************************************/

/** INPUT PINS **/

pin 1   =   CLK;        /* Tied to TOGGLE  */
pin 2   =   TOGGLE;
pin 3   =   Q8;                                                           20
pin 4   =   Q7;
pin 5   =   Q6;
pin 6   =   Q5;
pin 7   =   Q4;
pin 8   =   Q3;
pin 9   =   Q2;
pin 10  =   Q1;
pin 11  =   Q0;

/** OUTPUT PINS **/                                                       30

pin 23  =   SIGN;
pin 22  =   PREGATE5;
pin 21  =   PREGATE6;
pin 20  =   GATE;
pin 19  =   GATE2;
pin 18  =   PREGATE7;
pin 17  =   PREGATE1;
pin 16  =   PREGATE2;
pin 15  =   PREGATE3;                                                     40
pin 14  =   PREGATE4;

/** LOGIC EQUATIONS **/

SIGN.ar  =    'b'0;     /* Toggle sign each cycle */
SIGN.sp  =    'b'0;
SIGN.d   =    !SIGN;

PREGATE1 =      !Q0 & !Q1 & !Q2 & !Q3 & !Q4 & Q5 & Q7 & Q8 #
                !Q0 & Q1 & Q2 & !Q3 & !Q4 & Q5 & Q6 & Q7 & Q8 #            50
                !Q0 & Q1 & !Q2 & !Q3 & Q4 & !Q5 & Q8 #
                Q1 & Q2 & Q3 & !Q4 & !Q5 & Q6 & Q7 #
                Q0 & !Q1 & !Q2 & !Q3 & !Q4 & !Q5 & Q6 & Q7 & Q8 #
                !Q0 & !Q1 & !Q2 & Q3 & Q5 & !Q6 & Q7 #
                !Q0 & !Q1 & !Q2 & Q3 & !Q4 & !Q6 & Q7 & Q8 #
                Q1 & !Q2 & !Q3 & Q4 & Q5 & !Q6 & Q8 #
                Q1 & Q2 & Q3 & Q4 & Q5 & !Q6 & Q7 & Q8 #
                !Q1 & !Q2 & Q3 & !Q4 & Q5 & !Q6 & Q7 #
                !Q0 & !Q1 & !Q3 & !Q4 & Q5 & !Q6 & Q7 & Q8 #
                Q0 & Q1 & Q2 & !Q3 & !Q4 & !Q6 & Q7 & Q8 #                60
                !Q0 & !Q1 & Q2 & !Q3 & !Q4 & !Q5 & !Q6 & Q8 #
                Q0 & !Q1 & !Q3 & Q4 & !Q5 & !Q6 & Q7 & Q8;
```

```
PREGATE2 =      Q1 & Q2 & !Q3 & !Q5 & !Q6 & Q7 & Q8 #
                Q0 & !Q1 & Q2 & Q4 & !Q5 & !Q6 & Q7 & Q8 #
                Q0 & Q1 & !Q2 & !Q3 & !Q4 & !Q5 & !Q6 #
                Q1 & !Q2 & !Q3 & !Q4 & !Q5 & !Q6 & Q8 #
                Q0 & Q1 & !Q2 & !Q3 & !Q4 & !Q7 & Q8 #
                Q1 & !Q2 & Q4 & Q5 & Q6 & !Q7 & Q8 #
                !Q1 & !Q2 & !Q3 & !Q4 & !Q5 & !Q6 & !Q7 & Q8 #                  70
                !Q1 & !Q2 & !Q3 & Q4 & Q5 & !Q6 & !Q7 #
                !Q0 & Q2 & Q3 & !Q4 & Q5 & Q6 & !Q7 & Q8 #
                !Q1 & Q2 & !Q4 & Q5 & Q6 & !Q7 & Q8 #
                Q2 & !Q3 & !Q4 & !Q5 & Q6 & !Q7 & Q8 #
                Q0 & Q1 & Q2 & !Q3 & !Q5 & Q6 & !Q7;

PREGATE3 =      Q1 & Q2 & !Q3 & Q4 & !Q5 & Q6 & !Q7 #
                !Q0 & Q1 & !Q2 & Q4 & !Q5 & !Q7 & Q8 #
                !Q0 & Q3 & Q4 & !Q5 & Q6 & !Q7 & Q8 #
                !Q0 & !Q1 & !Q2 & !Q5 & Q6 & !Q7 & Q8 #                        80
                !Q0 & !Q2 & Q3 & !Q5 & Q6 & !Q7 & Q8 #
                !Q1 & Q2 & Q3 & Q4 & !Q5 & !Q7 & Q8 #
                Q0 & Q2 & Q3 & Q4 & !Q6 & !Q7 & Q8 #
                !Q1 & Q2 & Q4 & Q5 & !Q6 & !Q7 & Q8 #
                Q0 & !Q1 & Q2 & Q5 & !Q6 & !Q7 & Q8 #
                !Q0 & !Q2 & !Q3 & !Q6 & !Q7 & Q8;

PREGATE4 =      !Q0 & !Q1 & Q2 & !Q5 & !Q6 & !Q7 & Q8 #
                !Q0 & Q1 & Q3 & !Q4 & Q5 & !Q6 & !Q7 & Q8 #
                Q1 & !Q2 & !Q4 & Q5 & !Q6 & !Q7 & Q8 #                         90
                Q2 & Q3 & !Q5 & !Q6 & !Q7 & Q8 #
                !Q2 & !Q3 & !Q5 & !Q6 & !Q7 & Q8 #
                Q0 & !Q1 & !Q2 & !Q3 & Q4 & !Q5 & Q7 & !Q8 #
                Q1 & !Q2 & !Q3 & !Q4 & Q5 & Q7 & !Q8 #
                !Q2 & !Q3 & Q5 & Q6 & Q7 & !Q8;

PREGATE5 =      !Q1 & Q2 & Q3 & !Q4 & !Q5 & Q6 & !Q8 #
                Q1 & Q2 & Q3 & !Q4 & Q7 & !Q8 #
                Q0 & Q1 & Q3 & Q4 & Q5 & Q7 & !Q8 #
                Q0 & Q2 & Q3 & Q6 & Q7 & !Q8 #                                 100
                Q2 & Q3 & Q5 & Q6 & Q7 & !Q8 #
                Q0 & Q1 & !Q2 & !Q4 & Q6 & Q7 & !Q8 #
                !Q0 & !Q1 & !Q2 & !Q3 & !Q4 & !Q5 & Q6 & !Q8 #
                Q0 & !Q1 & Q2 & Q3 & !Q4 & Q5 & !Q8 #
                Q0 & !Q1 & !Q3 & !Q4 & Q5 & Q7 & !Q8 #
                !Q0 & Q1 & !Q2 & !Q5 & Q6 & Q7 & !Q8;

PREGATE6 =      !Q0 & !Q1 & Q2 & Q3 & !Q5 & Q7 & !Q8 #
                !Q1 & Q2 & Q4 & !Q5 & Q6 & Q7 & !Q8 #
                Q1 & !Q2 & Q4 & !Q5 & !Q6 & Q7 & !Q8 #                         110
                !Q1 & Q2 & !Q4 & !Q5 & !Q6 & Q7 & !Q8 #
                Q0 & Q2 & !Q3 & Q5 & !Q6 & Q7 & !Q8 #
                !Q0 & !Q1 & Q3 & Q4 & Q5 & !Q6 & Q7 & !Q8 #
                !Q2 & Q3 & Q4 & Q5 & !Q6 & Q7 & !Q8 #
                !Q0 & !Q1 & !Q2 & Q3 & Q4 & Q6 & !Q7 & !Q8 #
                !Q0 & Q1 & !Q2 & !Q4 & Q5 & Q6 & !Q7 & !Q8 #
                !Q0 & Q1 & Q3 & !Q4 & Q5 & Q6 & !Q7 & !Q8 #
                Q0 & Q1 & Q2 & !Q3 & Q4 & Q6 & !Q7 & !Q8 #
                Q0 & Q1 & Q2 & Q3 & Q4 & !Q5 & !Q7 & !Q8;
                                                                               120
PREGATE7 =      Q0 & Q1 & Q3 & Q4 & !Q5 & Q6 & !Q7 & !Q8 #
                !Q1 & !Q2 & !Q3 & !Q4 & !Q5 & Q6 & !Q7 & !Q8 #
                !Q1 & Q3 & Q4 & Q5 & Q6 & !Q7 & !Q8 #
                !Q1 & !Q2 & Q3 & Q5 & Q6 & !Q7 & !Q8 #
                Q0 & !Q2 & Q3 & Q4 & Q5 & Q6 & !Q7 & !Q8 #
                !Q0 & Q1 & Q2 & Q3 & Q4 & Q5 & !Q6 & !Q7 & !Q8 #
                Q0 & !Q1 & !Q2 & Q3 & Q4 & !Q5 & !Q6 & !Q7 & !Q8 #
                !Q0 & !Q1 & Q2 & Q3 & Q4 & Q5 & Q6;

GATE2 =         PREGATE1 #                                                     130
                PREGATE2 #
                PREGATE3 #
                PREGATE4 #
                PREGATE5 #
                PREGATE6 #
                PREGATE7;

GATE =          GATE2;
```

The code BIDIR3 below is for the device U13 on the bidirectional boost-buck-inverter board. Please see page 268 in Section C.5. This PAL performs logic operations that multiplex the MOSFET gate-drive signals.

# PAL Code - BIDIR3.PLD

```
Name            BIDIR3;
Partno          None;
Date            5/17/97;
Revision        01;
Designer        D. Jackson;
Company         MIT;
Assembly        None;
Location        None;
Device          P22V10;                                                    10

/*************************************************************************/
/* This pal is part U13 on the bidirectional boost/buck/inverter board.  */
/*                                                                       */
/* The switching logic truth table is below:                             */
/*    +------+---------+---------+---------+----------+                  */
/*    | SIGN | Switch  |  Boost  |  Buck   | Inverter |                  */
/*    +------+---------+---------+---------+----------+                  */
/*    |  0   |   Q1    |   PWM   |   OFF   | !GATE_F  |                  */
/*    |  0   |   Q2    |   OFF   |   PWM   |  GATE_R  |                  */
/*    |  0   |   Q3    |   OFF   |   OFF   |   OFF    |                  */
/*    |  0   |   Q4    |   ON    |   ON    |   ON     |                  */
/*    |  1   |   Q1    |   OFF   |   PWM   |  GATE_R  |                  */
/*    |  1   |   Q2    |   PWM   |   OFF   | !GATE_F  |                  */
/*    |  1   |   Q3    |   ON    |   ON    |   ON     |                  */
/*    |  1   |   Q4    |   OFF   |   OFF   |   OFF    |                  */
/*    +------+---------+---------+---------+----------+                  */
/*                                                                       */
/*************************************************************************/

/** INPUT PINS **/                                                        30

pin 1    =    PWM1;             /* PWM line from the Microcontroller        */
pin 2    =    BOOST_MODE;       /* Boost mode select from the Microcontroller*/
pin 3    =    RELAY2;           /* Relay control line from Microcontroller  */
pin 4    =    INRUSH;           /* Relay control line from Microcontroller  */
pin 5    =    AC_SIGN;          /* AC_SIGN from the line monitor            */
pin 6    =    SIGN;             /* SIGN used for INVERTER operation         */
pin 7    =    NC1;              /* No function */
pin 8    =    NC2;              /* No function */
pin 9    =    GATE;             /* GATE for INVERTER operation              */   40
pin 10   =    GATE_FALL_DLY;    /* GATE w/fall delay for INVERTER operation */
pin 11   =    GATE_RISE_DLY;    /* GATE w/rise delay for INVERTER operation */

/** OUTPUT PINS **/

pin 23   =    Q1_OPTO;
pin 22   =    Q2_OPTO;
pin 21   =    Q3_OPTO;
pin 20   =    Q4_OPTO;
pin 19   =    LEM_SIGN;                                                  50

/** LOGIC EQUATIONS **/

INVERTER   = INRUSH & !RELAY2 & !BOOST_MODE;    /* High in INVERTER MODE */

BUCKBOOST = INRUSH & RELAY2;                    /* High in BUCK or BOOST modes */

/***********************************/
/* BUCKBOOST MODE switching logic */
/***********************************/                                      60
Q1_BUCKBOOST = PWM1 & !BOOST_MODE & AC_SIGN #    /* BUCK  -> PWM when AC_SIGN=1 */
               PWM1 & BOOST_MODE & !AC_SIGN;     /* BOOST -> PWM when AC_SIGN=0 */
Q2_BUCKBOOST = PWM1 & !BOOST_MODE & !AC_SIGN #   /* BUCK  -> PWM when AC_SIGN=0 */
               PWM1 & BOOST_MODE & AC_SIGN;      /* BOOST -> PWM when AC_SIGN=1 */
Q3_BUCKBOOST = AC_SIGN;
Q4_BUCKBOOST = !AC_SIGN;

/*********************************/
/* INVERTER MODE switching logic */
/*********************************/                                      70
Q1_INVERTER  = !GATE_FALL_DLY & !SIGN #
               GATE_RISE_DLY & SIGN;
Q2_INVERTER  = GATE_RISE_DLY & !SIGN #
               !GATE_FALL_DLY & SIGN;
Q3_INVERTER  = SIGN;
Q4_INVERTER  = !SIGN;

/****************/
/* Output logic */
/****************/                                                        80
Q1_OPTO = Q1_BUCKBOOST & BUCKBOOST #
          Q1_INVERTER & INVERTER;
```

```
Q2_OPTO = Q2_BUCKBOOST & BUCKBOOST #
          Q2_INVERTER & INVERTER;
Q3_OPTO = Q3_BUCKBOOST & BUCKBOOST #
          Q3_INVERTER & INVERTER;
Q4_OPTO = Q4_BUCKBOOST & BUCKBOOST #
          Q4_INVERTER & INVERTER;
```

# Appendix E

# MATLAB Code

This appendix contains source-code listings for MATLAB scripts and functions which were used to create some of the results in this thesis. The files below were written for MATLAB version 4.2c for Windows. A number of the simulations were also performed using the MATLAB's Simulink extension. However, the graphical nature of the Simulink environment precludes printed source-code listings.

## E.1 M-File Listings

The script `RIPPLE.M` and its support functions `RIP_PFC.M` and `RIP_AMP.M` were used to generate the results in Figure A.4. The script calculates the net- and cell-current ripples for an $N$-cell interleaved boost converter. The simulation takes into account the fact that the amplitude of the ripple varies with the phase of the 60-Hz utility voltage. Results are obtained by direct time-domain simulation of the idealized cell currents, and the output is presented in a graphical form. This script is useful for the design and performance evaluation of interleaved boost converters.

**MATLAB Script - RIPPLE.M**

```
% MATLAB Script: RIPPLE.M
% ------------------------
% This script calculates and plots the input current ripple
% amplitude for an N-cell interleaved boost converter.
%
Vrms=120;          % Input AC voltage
Vout=350;          % Output DC voltage
Power=1000;        % Input/Output Power
Points=150;        % Number of points per half-line cycle
N=8;               % Number of interleaved cells                    10
L=525e-6;          % Cell inductance
T=40e-6;           % Cell switching peroid

% Calculate ripple amplitude for one half-line cycle.
[theta,stage,total,duty,avg]=rip_pfc(Vrms,Vout,Power,N,L,T,Points);

% Find the ripple amplitude limit
r_limit=Vout*T/(4*N*L);
```

```
% Plot the results
subplot(221)
plot(theta*2/pi,120*sqrt(2)*sin(theta));
axis([0 2.001 0 200]);
ylabel('Voltage (V)');

subplot(222)
plot(theta*2/pi,duty);
axis([0 2.001 0 1]);
ylabel('Duty Cycle');

subplot(223)
plot(theta*2/pi,stage,theta*2/pi,avg)
axis([0 2.001 0 15]);
xlabel('Angle (radians)');
ylabel('Current (A)');

subplot(224)
plot(theta*2/pi,total,[0 2],r_limit*[1 1])
axis([0 2.001 0 1]);
xlabel('Angle (radians)');
ylabel('Current (A)');
```

## MATLAB Function - RIP_PFC.M

```
function [theta,stage,total,duty,avg]=rip_pfc(Vrms,Vout,Power,N,L,T,PTS)
%
% This function is used by RIPPLE.M.

theta=linspace(0,pi,PTS*2);

Vpeak=Vrms*sqrt(2);
vin=Vpeak*sin(theta);
Irms=Power/Vrms;
k=Irms/Vrms;

duty=sqrt(2*L*k/(N*T)*(Vout-vin)/Vout);

for i=1:PTS;
   [stage(i),total(i),avg(i)]=rip_amp(vin(i),Vcut,duty(i),N,L,T);
end

stage([1:PTS]+PTS)=stage(PTS:-1:1);
total([1:PTS]+PTS)=total(PTS:-1:1);
avg([1:PTS]+PTS)=avg(PTS:-1:1);

disp(['Power=' num2str(Power) ' Watts']);
disp(['Peak Stage=' num2str(max(stage)) ' Amps']);
disp(['Peak Ripple=' num2str(max(total)) ' Amps']);
```

## MATLAB Function - RIP_AMP.M

```
function [stage,total,avg]=rip_amp(Vin,Vout,duty,N,L,T)
%
% This function estimates the peak-peak ripple current for an N-cell
% interleaved boost converter operating in DCM.
%
%    Vin   = Input Voltage (DC)          stage = Peak cell current
%    Vout  = Output Voltage (DC)         total = Net input current
%    duty  = Switch duty ratio           avg   = Mean input current
%    L     = Cell inductance
%    T     = Cell switching period

base=500;
period=base*N;
d_rise=ceil(duty*period);

peak=Vin/L*duty*T;
fall_slope=(Vout-Vin)/L;
fall_time=peak/fall_slope;
d_fall=ceil(fall_time/T*period+eps);
d_off=period-d_rise-d_fall;

% If d_off is less than zero, then CCM operation will occur
if(d_off<0)
   disp('CCM\n');
```

```
end

% Generate the interleaved pattern
pulse=[linspace(0,peak,d_rise) linspace(peak,0,d_fall) linspace(0,0,d_off)];
pulse_sum=zeros(size(pulse));
pulse_shift=pulse;
index=[[(base+1):period] [1:base]];
for i=1:N
   pulse_shift=pulse_shift(index);
   pulse_sum=pulse_sum+pulse_shift;
end

% Measure peak values
total=max(pulse_sum)-min(pulse_sum);
avg=mean(pulse_sum);
stage=peak;
```

The script VDROOP.M below was used to generate the results in Figures 3.20 and 3.21. The script calculates the DC transfer characteristic of the half-bridge DC/DC converter using the switching model (Mode 1 and 2) described in Chapter 3, subsection 3.3.4. This script is useful for evaluating DC/DC converter performance as well as potential inductive coupling designs.

## MATLAB Script - VDROOP.M

```
% MATLAB Script: VDROOP.M
% ----------------------
% This script calculates the voltage droop for a half-bridge DC/DC
% converter using the Mode 1/2 model.

Lp=18.77e-6;          % Primary-side leakage inductance
Ls=4.33e-6;           % Secondary-side leakage inductance
Lm=199.3e-6;          % Magnetizing inductance
N=12/25;              % Turns ratio

T=10.0e-6;            % Switching frequency
C=350e-12;            % Output rectifier capacitance
Vbuss=380;            % Input bus voltage
Cout=(15e-6)/2;       % Output bus capacitance

kd=1.15*T/(2*Cout); % Output capacitance correction factor
ddrop=1.7;            % Output rectifier voltage drop

N2=N*(Lm/(Lm+Lp));
Lls=N^2*(Ls+Lp*(Lm/(Lm+Lp)));
Io1=[logspace(log10(0.01),log10(1),50) linspace(1,35,150)];
Vina=Vbuss*N2;

% Mode 1 calculation
a=1;
b=-T*Io1/(4*C);
c=T*Vina*Io1/(8*C);
x1a=(-b-sqrt(b.^2-4*a.*c))./(2*a);
tmp=(-b+sqrt(b.^2-4*a.*c))./(2*a);
wa=4*x1a*sqrt(Lls*C)./(2*x1a-Vina);
x1a=x1a+kd*Io1-ddrop;

% Mode 2 calculation
P=T/2;
a=16*C*Lls;
b=4*P*(P*Vina-2*Io1*Lls-4*Vina*sqrt(Lls*C));
c=12*Io1*P*Lls*Vina-2*P^2*Vina^2;
x2a=(-b+sqrt(b.^2-4*a.*c))./(2*a);
x2a=x2a+kd*Io1-ddrop;

% Find real solutions
P1a=Io1.*real(x1a);
x1a(abs(imag(x1a))>0)=nan*ones(size(x1a(abs(imag(x1a))>0)));
v1a=real(x1a*2);
P2a=Io1.*real(x2a);
v2a=real(x2a*2);
```

```
% Find mode boundry and combine mode 1 & 2 results
transa=length(wa)-sum(wa>T/2);
cva=[v1a(1:transa-1) v2a(transa:length(wa))];
cpa=[P1a(1:transa-1) P2a(transa:length(wa))];

% Plot voltage droop versus output power
plot(cpa,cva);axis([0 1200 0 200])
grid
title('Droop Characterestic Using Switching Model');
xlabel('Power (Watts)');
ylabel('Output Voltage (Volts)');
```

The script ANNEAL1.M below was used to generate the 1024-bit "magic sinewave" sequence shown in Figures 4.11 and 4.12. This script is based on a the simulated annealing algorithm developed in [96]. A brief description of magic sinewaves and the simulated annealing algorithm was provided in Section A.4.

## MATLAB Script - ANNEAL1.M

```
% MATLAB Script: ANNEAL1.M
% ------------------------
% This script uses random bit jittering and the metropolis algorithm
% to hunt for Magic Sinewaves.
%
% The simulated annealing process minimizes a loss function,
% which has two parts:
%   1) The user supplies a bound on the harmonic magnitudes over
%      a range of odd frequencies.  The difference between this
%      bound and the actual bound is linearly weighted and then
%      summed.
%   2) The user supplies a target for the # of transitions.  The
%      difference between the actual & target is then linearly
%      weighted.

% Magin sinewave parameters
N=256;                          % Magic SIN quarter_length.
Amp=0.5911;                     % Amplitude of the fundamental

% Loss function properties
bound=[linspace(0.1,0.5,4) linspace(0.5,1,15) linspace(2,10,4)];
linear_weight=10;               % Linear weighting factor
tran_target=300;                % Target # of transitions
tran_weight=100;                % Linear weighting factor on #trans-target

% Simulated annealing parameters
Tinitial=3;                     % Initial temperature
Tcool=0.90;                     % Cooling factor
Theat=3;                        % Heating factor
hit_perc=1;                     % Hit percentage for Temp change
Titerations=200;                % MAX Number of temperature cycles
double_jitters=0;               % A one causes double bit jitters

shuffles=N/2;                 % Set the numbers of shuffles or
if shuffles < 200             % bit jitters and a given temperature
   shuffles=200;
end
if shuffles > 250
   shuffles=250;
end

% Calculations
%-------------
sin_length=4*N;
half_length=2*N;
N_ones=round(Amp*2*N/pi);   % Amplitude in # of 1's
N_zeros=N-N_ones;

% Generate an initial guess for the Magic SIN
magic_sin=[zeros(1,N-N_ones) ones(1,N_ones)];

index=1:N;
normalize_bound=mean(bound);          % Use to normalize bound
harms=length(bound);                  % Number of harmonics to cancel
harm_indexes=[3:2:(2*(harms-1)+3)];  % Harmonic indexes
```

```
% Initialize annealing variables
T=Tinitial;
best=1e6;
last_loss=1e6;                                                    60
reheats=0;

% Main annealing LOOP
for j=1:Titerations
   hits=0;
   Qhits=0;
   for i=1:shuffles
      new_magic_sin=magic_sin;

      z=floor(rand(1)*(N_zeros-1))+2;      % Select a random z and o to change   70
      o=floor(rand(1)*N_ones)+1;
      i_ones=index(new_magic_sin);         % Locate z's
      one_location=i_ones(o);
      i_zeros=index(1-new_magic_sin);      % Locate o's
      zero_location=i_zeros(z);
      new_magic_sin(one_location)=0;       % Swap bits
      new_magic_sin(zero_location)=1;

      if(j<double_jitters)
         z=floor(rand(1)*(N_zeros-1))+2;      % Select a random z and o to change   80
         o=floor(rand(1)*N_ones)+1;
         i_ones=index(new_magic_sin);         % Locate z's
         one_location=i_ones(o);
         i_zeros=index(1-new_magic_sin);      % Locate o's
         zero_location=i_zeros(z);
         new_magic_sin(one_location)=0;       % Swap bits
         new_magic_sin(zero_location)=1;
      end

      % Assemble the full magic SIN                                 90
      half_sin=[new_magic_sin new_magic_sin(N:-1:1)];
      full_sin=[half_sin -half_sin];

      % Find loss factor due to # of transitions
      num_trans=sum(abs(diff(full_sin)));
      trans_error=(num_trans-tran_target)/tran_target;
      if trans_error<0
         trans_factor=0;
      else
         trans_factor=trans_error*tran_weight;                    100
      end

      % Find loss due to bounded harmonics
      temp=2*abs(fft(full_sin)/sin_length);
      fundamental=max(temp);
      temp=temp/fundamental*100;
      trial_spectrum=temp(harm_indexes+1);
      % Find harmonic error
      harm_error=((trial_spectrum-bound)./bound);
      % Attempt to normalize                                       110
      harm_error=harm_error/length(bound)*normalize_bound;
      % Linearly Weight & sum
      harm_error=harm_error*linear_weight;
      harm_error([harm_error<0])=zeros(1,length(harm_error([harm_error<0])));
      harm_factor=sum(harm_error);

      % Add loss components
      loss=harm_factor + trans_factor;

      % Compare to last loss and save new SIN if loss improves     120
      change=loss-last_loss;
      if (change < 0)
         last_loss=loss;
         magic_sin=new_magic_sin;
         hits=hits+1;
      elseif (rand(1) < exp(-change/T))
         last_loss=loss;
         magic_sin=new_magic_sin;
         Qhits=Qhits+1;
         hits=hits+1;                                            130
      end

      % Keep a copy of the best SIN
      if loss < best
         best_sin=new_magic_sin;
         best=loss;
         best_harm_factor=harm_factor;
         best_tran_factor=trans_factor;
         best_trans=num_trans;
      end                                                        140
   end
```

```
    magic_sin=best_sin;

    % Cool or heat the "temperature" depending on the # of hits
    if (hits > (hit_perc/100 * shuffles))
       T=T*Tcool;
       temp_string=['Cooling'];
    else
       if ((hits==0) & (T==Tinitial))
          disp(['                          No hits at T=Tinitial.....finished.']);
          break;
       end
       T=T*Theat;
       temp_string=['Heating'];
       reheats=reheats+1;
    end

    % Display progress
    string1=sprintf('Tcyc=%2.0f Loss=%1.4f (H=%1.4f T=%1.4f) #Trans=%4.0f  ', ...
                    j,best,best_harm_factor,best_tran_factor,best_trans);
    string2=sprintf('Hits=%4.1f%% Q%%=%4.1f%% -> %s T=%1.4f', ...
                    hits/shuffles*100,Qhits/(hits+100*eps)*100,temp_string,T);
    disp([string1 string2]);

    % Plot the intermediate result
    full_sin=[best_sin best_sin(N:-1:1) -best_sin -best_sin(N:-1:1)];
    spect=2*abs(fft(full_sin)/sin_length);
    fundamental=max(spect);
    spect=spect/fundamental*100;

    figure(1)
    subplot(311)
    stairs(1:sin_length,full_sin);
    axis([0 sin_length -1.1 1.1]);
    xlabel('Index (n)');
    ylabel('Amplitude');
    string=['Bit Pattern for a Single Period After ' num2str(j) ' Temp. Cycles'];
    title(string);
    text(0.25*N,-0.5,['Length=' num2str(4*N) '   #Trans=' num2str(best_trans)]);
    string=sprintf('Fundamental=%4.3f',fundamental);
    text(2.25*N,0.5,string);

    subplot(312)
    bar(0:sin_length-1,spect);
    hold on
    [x,y]=stairs(harm_indexes-1,bound);
    plot(x,y,'b');
    hold off
    axis([0 min([length(bound)*10 2*N]) 0 100]);
    xlabel('Harmonic Number');
    ylabel('Magnitude');
    title('Wide View of Harmonic Content');

    subplot(313)
    bar(0:sin_length-1,spect);
    hold on
    [x,y]=stairs(harm_indexes-1,bound);
    plot(x,y,'b');
    hold off
    axis([0 min([length(bound)*3 2*N]) 0 max(bound)]);
    xlabel('Harmonic Number');
    ylabel('Magnitude');
    title('Zoomed View of Harmonic Content');

    figure(gcf)
end

% Display summary of final result
num_trans=sum(abs(diff(full_sin)));
disp('-----------');
string=sprintf('FINAL Rel Lost = %1.4f  #Trans=%4.0f   #Reheats=%1.0f', ...
                best,num_trans,reheats);
disp(string);
```

# Appendix F

# Microcontroller Code

All digital control algorithms were developed in the C programming language on the 80C196KC microcontroller. This appendix briefly discusses how the 80C196KC is programmed. The software structure is overviewed and representative source-code listings are provided.

## F.1 Programming the 80C196KC

The Intel 80C196KC is a low-cost 16-bit integer-math microcontroller, which runs at 16 MHz. The processor executes instructions stored in EPROM on the microcontroller board. As configured in Appendix C, the system has 24 kbytes of EPROM and 32 kbytes of RAM available for user programs. Since the 80C196KC addresses external memory using 16-bit odd/even addressing, two 32 k by 8-bit EPROMs (or 16 k by 8-bit) are used, one for the HI (odd) bytes and one for the LO (even) bytes. User programs can be changed readily by swapping EPROMs.

Source code was developed in "C" on a Pentium PC. It was compiled using cross-compilers from Intel and Hi-Tech. The Hi-Tech compiler was later used exclusively because of its macro support and complete library of 32-bit floating-point math routines, which greatly simplify algorithm implementation. The Hi-Tech compiler also provides an intermediate output in assembly, which aids in debugging. Compiled code is linked to produce a binary file suitable for EPROM programming. A batch file `MAKEROM.BAT` at the end of this appendix performs this operation. Programs were "burned" into the HI/LO EPROM pairs using a universal EPROM programmer from Microchip Technologies.

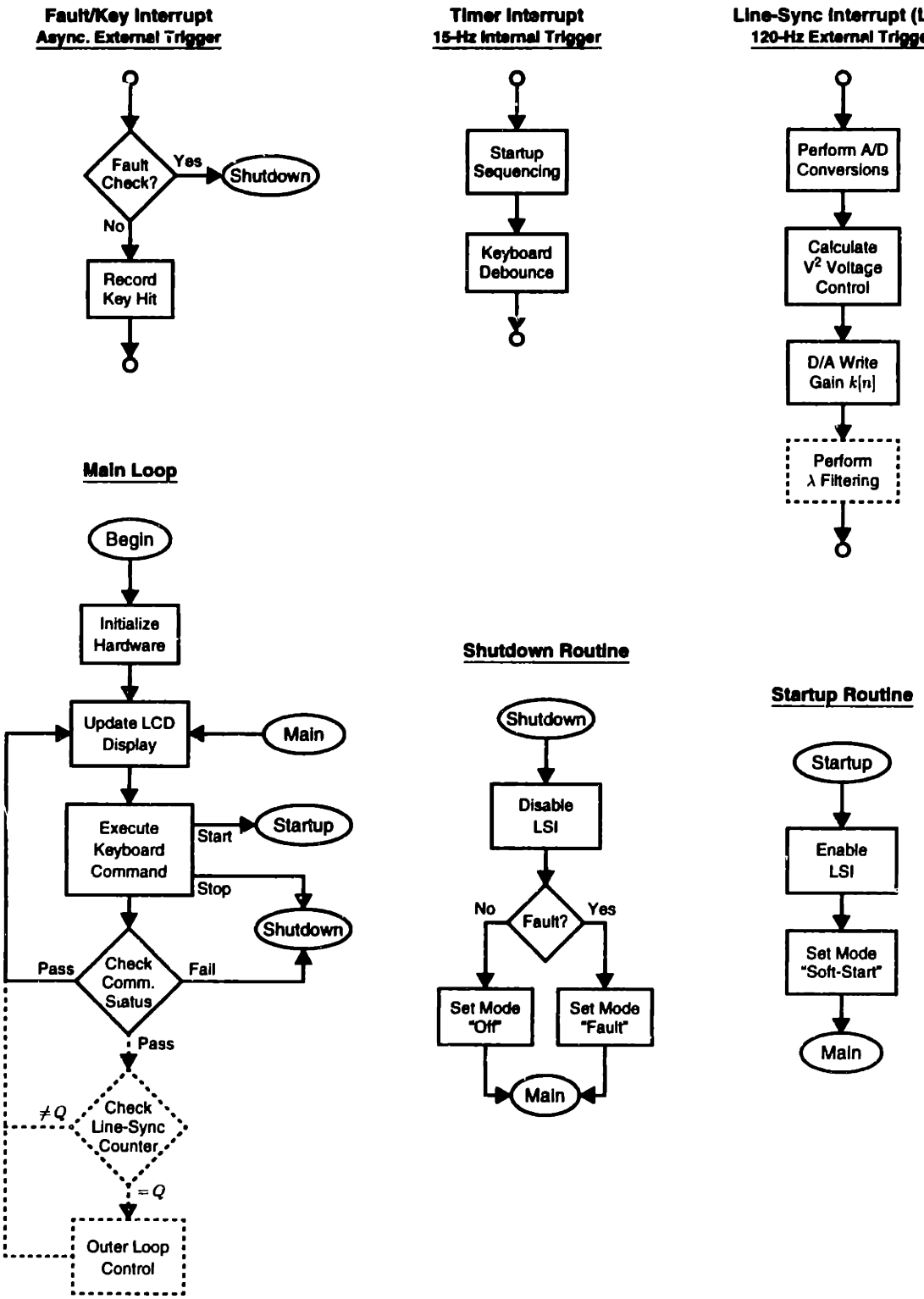



**Figure F.1:** Microcontroller software flowchart. Critical operations, such as control and fault detection, are performed by interrupt driven routines.

## F.2 Code Structure

Figure F.1 flowcharts the general structure of the microcontroller software. The three flows at the top of Figure F.1 are interrupt driven. This means that normal execution is interrupted so that these routines may execute at specific instances. These interrupt-driven routines may be switched on and off so that their execution can be controlled. The three flows at the bottom of Figure F.1 are lower priority. These routines execute only during the interim cycles between interrupts.

Program execution begins at the top of the main loop. A brief initialization sets up the external hardware, and the main loop then begins to an endless cycle. Two of the interrupt routines, the timer interrupt and the fault/keyboard interrupt, always remain active so that the main loop can be interrupted when necessary. The timer interrupt is internally triggered, and it executes periodically at a rate of 15 Hz. When called, this routine debounces input from the external keyboard and sequences the converter soft start. The fault/keyboard interrupt is triggered externally by a TTL signal. This signal is generated when a fault is detected by the DC/DC control circuitry or by a key hit on one of the four control panel buttons. A fault causes the converter to be shutdown, otherwise the key hit is recorded (for later execution) and the interrupt ends.

Pressing the "Start" or "Stop" keys will momentarily branch the main loop to the startup or shutdown routines. The startup routine enables the line-sync interrupt (LSI) and notifies the soft-start sequencer to begin turning on the converter. The LSI executes in sync with the zero crossings of the AC utility voltage (nominally 120 Hz), and it performs critical operations for the digital control algorithm. When called, the LSI samples all eight A/D converter channels, computes a new $V^2$ voltage-loop command $k[n]$, and writes this command (via a D/A converter) to the analog inner current-loop controller. The LSI may also perform $\lambda$ filtering for the adaptive LM-RLS algorithm.

When the LSI is enabled, the $V^2$ voltage-loop controller executes automatically. Therefore, an outer-loop controller, as in the case of adaptive or general multirate control, can be implemented as part of the main loop. The dashed elements of the main loop in Figure F.1 show the program flow including the outer-loop control. The query block

ensures that the outer-loop calculation occurs once per $Q$ cycles of the LSI. This maintains the $n = QN$ relationship between multirate time indexes. Although it is not clear from the diagram, the line-sync counter is queried not once, but many times during a single cycle of the main loop. This minimizes the potential delay between the instant $n = QN$ and the time the outer-loop control calculation begins.

The interrupt-driven software structure ensures that time-critical tasks, such as voltage control and fault detection, receive high priority. Interim processor cycles, when available, are devoted to low priority tasks such as updating the LCD display and responding to stored keyboard input. In practice, this structure has been proven to work well. The 80C196KC has more than enough processing power to tackle complex control tasks.

## F.3 Source-Code Listings

The large amount of experimental work in this thesis resulted in many different versions of the microcontroller software. Since it would be unreasonable to include all of them in this appendix, two representative versions were selected. The files `BID_FL5` and `MOTOR1`, listed in the subsections that follow, were designed for the bidirectional and unidirectional prototypes, respectively. Although both programs share the basic structure of Figure F.1, hardware differences between the two prototypes necessitates complete software listings of both. Included at the end of this section are batch files, which were used to compile and link the "C" code, and a MathCAD spreadsheet, which was used to generate a number of the hardware specific #DEFINE constants.

### F.3.1. Bidirectional Battery Charge/Discharge

The source code `BID_FL5` below was developed for use with the 600-W bidirectional prototype. This software implements the digital battery-current control algorithms described in Section 5.3, and it was used to generate the data for Figures 4.5 – 4.8, 4.11, 4.17, 4.18 and 5.14 – 5.16 in the text. The software assumes that a 120-V, 2-kWh lead acid battery pack is connected to the secondary side of the full-bridge DC/DC converter. Menu options allow the user to select between "charge," "discharge," or "inverter" operating modes. In addition, the user may select between constant, square wave, or sawtooth current profiles for the charge and discharge modes.

# "C" Source Code - BID_FL5.C

```
/*********************************************************************/
/* FILE: BID_FL5.C                                  REVISION: 5.0    */
/* BY: Deron Jackson                                DATE: 8/1/97     */
/*                                                                   */
/* HARDWARE:  BIDIRECTIONAL BOOST/BUCK                               */
/*            FULL-BRIDGE DC/DC                                      */
/*                                                                   */
/* This version does:                                                */
/*      - BATTERY Charge/Discharge under current control             */
/*      - Floating-point math                                        */
/*      - Selectable safeties                                        */
/*      - Buck operation (constant, step, ramp)                      */
/*      - Boost operation (constant, step, ramp)                     */
/*      - Inverter mode                                              */
/*                                                                   */
/* NOTES:                                                            */
/*    This code is configured for the HITECH 80196 C-Compiler.       */
/*    The code must be linked together with the DKJSTART.OBJ and     */
/*    DKJROM.OBJ files.                                              */
/*********************************************************************/

#include<80C196kc.h>
#include<intrpt.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<float.h>

#include "bid_fl5.h"

  /*****************************/
  /** Setup the IRQ routines **/
  /*****************************/
interrupt void software_timer(void);
interrupt void line_sync_interrupt(void);
interrupt void hsi_event(void);
interrupt void analog_conversion_done(void);

  /*************************/
  /** Function Prototypes **/
  /*************************/
void wait_fun(uint cntsize);
void MDAC_write(void);
void FBDAC_write(void);
void write_lcd_line(uchar line_num);
void write_hp_display(void);
void halt_converter(void);
int check_com(void);
void boost_current_control(void);
void buck_current_control(void);
void step_ramp_control(void);
int int_part(float fp_value);
int dec_part(float fp_value);
int round_fp(float fp_value);
void make_bar_graph(float level,float min_lev,float max_lev);
void start_converter(void);
void select_control_mode(void);
void select_control_direction(void);
void select_safety_mode(void);
void spare_select_menu(void);
void spare_select_menu2(void);
void spare_select_menu3(void);
void spare_select_menu4(void);
void spare_select_menu5(void);
void lcd_menu_running(void);
void lcd_menu_halt(void);
void lcd_menu_stop(void);
void lcd_menu_start(void);
void initialize_globals(void);
void initialize_hardware(void);


/*********************************************************************/
/** WAIT_FUN()  This is a simple wait loop.                         **/
/*********************************************************************/
void wait_fun(uint cntsize)
{
   uint t1=0;
   uint t2=0;
```

```
   uint t3=0;

   while(t1 < cntsize)
   {
      t1++;
      while(t2 < cntsize)
      {
         t2++;
         while(t3 < cntsize)
            t3++;
      }
   }
}


/*********************************************************************/
/** MDAC_WRITE()  This procedure writes a 10 bit word to the MAX501 **/
/** multiplaying DAC.                                               **/
/*********************************************************************/
void MDAC_write(void)
{
   ioport1 = (uchar) (int_icmd >> 8);
   HSO_clr(1);              /* Set /CSMSB LOW  (select the MSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the MSB) */
   HSO_set(1);              /* Set /CSMSB HIGH                    */
                            /* Set IOPORT1 with the LSB 8 bits    */
   ioport1 = (uchar) (int_icmd & 0x00ff);
   HSO_clr(3);              /* Set /CSLSB LOW  (select the LSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the LSB) */
   HSO_set(3);              /* Set /CSLSB HIGH                    */

   HSO_clr(0);              /* Set /LDAC  LOW  (transfer 12 bits) */
   HSO_set(0);              /* Set /LDAC  HIGH                    */
}


/*********************************************************************/
/** FBDAC_WRITE()  This procedure writes a 10 bit word to the MAX508 **/
/** DAC on the Full-Bridge controller board.                        **/
/*********************************************************************/
void FBDAC_write(void)
{
   ioport1 = (uchar) (int_fbcmd >> 8);
   HSO_clr(1);              /* Set /CSMSB LOW  (select the MSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the MSB) */
   HSO_set(1);              /* Set /CSMSB HIGH                    */
                            /* Set IOPORT1 with the LSB 8 bits    */
                            /* Set IOPORT1 with the LSB 8 bits    */
   ioport1 = (uchar) (int_fbcmd & 0x00ff);
   HSO_clr(3);              /* Set /CSLSB LOW  (select the LSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the LSB) */
   HSO_set(3);              /* Set /CSLSB HIGH                    */

   REG_U2_clrb(3);          /* REG_U2 bit 3 (the /LDAC pin)       */
   REG_U2_setb(3);
}


/*********************************************************************/
/** WRITE_LCD_LINE()  This procedure writes a 40 character line to  **/
/** the LCD display.                                                **/
/*********************************************************************/
void write_lcd_line(uchar line_num)
{
   uchar pos;

   di();
   REG_U42_clrb(6);             /* Set L-6 = 0 ---- LCD Inst. mode */
   if(line_num==1)
      ioport1 = 0x80;           /* Set Address for line 1          */
   else
      ioport1 = 0xC0;           /* Set Address for line 2          */
   P2_setb(5);                  /* Strobe P2-5 to latch LCD data   */
   P2_clrb(5);
   REG_U42_setb(6);             /* Set L-6 = 1 ---- LCD Data mode  */
   ei();
   wait_fun(LCD_DELAY);

   for(pos=0;pos<40;pos++)
   {
      di();
      if(line_num==1)
```

```
      ioport1 = lcd_line1[pos];
   else
      ioport1 = lcd_line2[pos];
   P2_setb(5);                /* Strobe P2-5 to latch LCD data   */
   P2_clrb(5);
   ei();
   wait_fun(LCD_DELAY);
  }
}

/*********************************************************************/
/** WRITE_HP_DISPLAY()  This procedure writes 4 characters to the HP**/
/** display.                                                        **/
/*********************************************************************/
void write_hp_display(void)
{
  uchar hp_digit, hp_char, pulse;

  pulse=(strobe >> 2);
  for(hp_digit=0;hp_digit<4;hp_digit++)
  {
            /*********************************************/
            /******* NOTE:  BITS 6 and 4 are reversed ******/
            /******* ----   to correct a hardware bug. *****/
            /*********************************************/
    hp_char = (display[hp_digit] & 0x2F)
                + ((display[hp_digit] & 0x10) << 2)
                   + ((display[hp_digit] & 0x40) >> 2);
    switch(hp_digit)
    {
      case 0: set_bit(reg_U42,2);set_bit(reg_U42,3);        /* Select the 1st DIGIT */
              if(pulse==0) hp_char=42; break;
      case 1: set_bit(reg_U42,2);clr_bit(reg_U42,3);        /* Select the 2nd DIGIT */
              if(pulse==1 || pulse==5) hp_char=42; break;
      case 2: clr_bit(reg_U42,2);set_bit(reg_U42,3);        /* Select the 3rd DIGIT */
              if(pulse==2 || pulse==4) hp_char=42; break;
      case 3: clr_bit(reg_U42,2);clr_bit(reg_U42,3);        /* Select the 4th DIGIT */
              if(pulse==3) hp_char=42; break;
      default: break;
    }
    SET_reg_U42(reg_U42);
    di();
    ioport1 = hp_char;
    HSO_clr(5);                 /* Strobe the /WR pin on the DISPLAY */
    HSO_set(5);
    ei();
    wait_fun(LCD_DELAY);
  }
}

/*********************************************************************/
/** HALT_CONVERTER()  This procedure is called in order to shutdown **/
/** the converter.                                                  **/
/*********************************************************************/
void halt_converter(void)
{
  di();                     /* DISABLE all the interrupts    */
  int_pending = 0;          /* Clear any pending interrupts  */
  int_mask = 0;             /* Mask all IRQ's unless we hit  */
                            /* an ei() on the way to main.   */
  reset_ints=TRUE;

  int_icmd = 0;             /* Zero the current command */
  MDAC_write();

  if(ctrl_mode==INVERTER_MODE)    /* Inverter shutdown */
  {
    reg_U2=0x52;            /* Set B1_ENABLE     = 0   DISABLE B1     */
                            /* Set FAULT_CLR     = 1   CLEAR FAULTS   */
                            /* Set B2_ENABLE     = 0   DISABLE B2     */
                            /* Set FB_LDAC       = 0                  */
                            /* Set B1_RLY_SET    = 1   SET B1 RELAYS  */
                            /* Set B1_RLY_RESET  = 0                  */
                            /* Set B2_RLY_SET    = 1   SET B2 RELAYS  */
                            /* Set B2_RLY_RESET  = 0                  */
    SET_reg_U2(reg_U2);
  }
  else if(ctrl_dir==REVERSE)     /* Buck shutdown */
  {
    reg_U42=0x70;           /* Set RELAY_TTL     = 0   CLOSE RELAY_1  */
                            /* Set PWM_EN_TTL    = 0   DISABLE the PWM */
                            /* Set DS_A0         = 0                  */
                            /* Set DS_A1         = 0                  */
                            /* Set L-4           = 1                  */
```

```
                        /* Set L-5 (B_MDE)= 1     BUCK MODE (1)    */
                        /* Set L-6 (RS)   = 1     LCD Data Mode    */
                        /* Set L-7 (RLY2) = 0     CLOSE RELAY_2    */
      SET_reg_U42(reg_U42);
      reg_U2=0x52;      /* Set B1_ENABLE    = 0   DISABLE B1     */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS   */                    260
                        /* Set B2_ENABLE    = 0   DISABLE B2     */
                        /* Set FB_LDAC      = 0                  */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS  */
                        /* Set B1_RLY_RESET = 0                  */
                        /* Set B2_RLY_SET   = 1   SET B2 RELAYS  */
                        /* Set B2_RLY_RESET = 0                  */
      SET_reg_U2(reg_U2);
      wait_fun(10000);   /* Wait for I to settle */
   }
   else if(ctrl_dir==FORWARD)     /* Boost shutdown */                               270
   {
      reg_U42=0xD1;     /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM  */
                        /* Set DS_A0      = 0                      */
                        /* Set DS_A1      = 0                      */
                        /* Set L-4        = 1                      */
                        /* Set L-5 (B_MDE)= 0     BOOST_MODE (0)   */
                        /* Set L-6 (RS)   = 1     LCD Data Mode    */
                        /* Set L-7 (RLY2) = 1     OPEN RELAY_2     */
      SET_reg_U42(reg_U42);                                                         280
      reg_U2=0x92;      /* Set B1_ENABLE    = 0   DISABLE B1     */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS   */
                        /* Set B2_ENABLE    = 0   DISABLE B2     */
                        /* Set FB_LDAC      = 0                  */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS  */
                        /* Set B1_RLY_RESET = 0                  */
                        /* Set B2_RLY_SET   = 0   CLR B2 RELAYS  */
                        /* Set B2_RLY_RESET = 1                  */
      SET_reg_U2(reg_U2);
      wait_fun(1000);   /* Wait for I to settle */                                  290
   }

   reg_U42=0xD1;        /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM  */
                        /* Set DS_A0      = 0                      */
                        /* Set DS_A1      = 0                      */
                        /* Set L-4        = 1                      */
                        /* Set L-5 (B_MDE)= 0     BOOST_MODE (0)   */
                        /* Set L-6 (RS)   = 1     LCD Data Mode    */
                        /* Set L-7 (RLY2) = 1     OPEN RELAY_2     */                300
   SET_reg_U42(reg_U42);

   if(err_num!=0)
   {
      leds=LED_PF;                  /* Turn ON the POWER & FAULT LEDs only */
      menu_mode=HALT_ERROR_MENU;    /* Set the ERROR MENU */
   }
   else
   {
      leds=LED_P;                   /* Turn ON the POWER LED only */              310
      menu_mode=CONVERTER_STOPPED; /* Set MAIN MENU */
   }

   mode=OFF;                    /* Set the converter mode to OFF */
   button_id=0;
   button_db=0;
}


/*********************************************************************/            320
/** HSI_EVENT()  This procedure executes each time an HSI.0 event   **/
/** occurs.  This may be a button press or a fault in the DC-DC     **/
/** converter.                                                      **/
/*********************************************************************/
interrupt void hsi_event(void)
{
   uchar hold_wsr, hold_ioport1;
   uchar hsi_temp;

   IRQ_ENTRY;                                                                       330

   hsi_temp = ioport2 & 0x58;    /* Read the status of P2-3,P2-4,P2-6 */

   if(hsi_temp)                 /* If any button was pressed reset the LED's */
   {                            /* This helps to fix an apparent glitch.     */
      SET_leds(leds);           /* Update the 4 LED's */
   }

   if((hsi_temp==0x08 || hsi_temp==0x10 || hsi_temp==0x18) && FAULT_CHECKS
                                                  && (safety_mode==ALL))            340
```

```
   {
      if(menu_mode!=HALT_ERROR_MENU)   /* Check for an existing halt state  */
      {
         if(hsi_temp==0x08)
            err_num=7;                /* Fault on DC-DC Board #1 */
         else if(hsi_temp==0x10)
            err_num=6;                /* Fault on DC-DC Board #2 */
         else
            err_num=8;                /* Fault on both boards    */
         halt_converter();                                                    350
         IRQ_EXIT;
         return;
      }
   }

   if(hsi_temp==MENU_ID || hsi_temp==LEFT_ID || hsi_temp==RIGHT_ID || hsi_temp==START_ID)
   {
      if(button_db>BUTTON_DB)       /* Debounce the button press by N/20th's of a sec */
         button_id=hsi_temp;
   }                                                                          360
   IRQ_EXIT;
}


/*********************************************************************/
/** CHECK_COM()  This procedure checks the com line to the DC/DC     **/
/** converter.  If low it means the puck is not inserted.           **/
/*********************************************************************/
int check_com(void)
{                                                                           370
   if(check_bit(ioport2,1)==0)           /* Read the COM_CHK line P2.1 */
   {
      wait_fun(25);
      if(check_bit(ioport2,1)==0)        /* If failed, double check    */
      {
         wait_fun(25);
         if(check_bit(ioport2,1)==0)     /* If failed, triple check    */
            return(1);                   /* Finally return 1 if failed */
      }
   }                                                                        380
   return(0);                            /* Otherwise return a 0 */
}


/*********************************************************************/
/** SOFTWARE_TIMER()  This procedure is set to execute 20 times a   **/
/** second.                                                         **/
/*********************************************************************/
interrupt void software_timer(void)
{                                                                           390
   uchar hold_wsr, hold_ioport1;

   IRQ_ENTRY;
   line_cycs=cyc_count;             /* Record the # of line cycles in */
                                    /* the last 1/15.3th of a second. */
   cyc_count=0;                     /* Reset the counter.             */
   swt_count++;                     /* Increment the swt_counter      */
   strobe++;                        /* Increment strobe counter       */
   if((strobe >> 2) > 5)
      strobe=0;                                                             400

   if(button_db < 255)              /* Increment the button debouncer */
      button_db++;

   if(swt_count > swt_delay)
   {
      switch(mode)
      {
         case INVERTER_STARTUP_1:
               REG_U2_clrb(1);      /* Clear FAULT_CLR -- READY FAULTS  */   410
               REG_U2_setb(2);      /* Set   B2_ENABLE -- ENABLE B2     */
               mode=INVERTER_STARTUP_2;
               swt_count=0;                 /* Reset the swt_count */
               swt_delay=BUCK_DC_DELAY;     /* Set a new delay time*/
               break;
         case INVERTER_STARTUP_2:
               REG_U42_clrb(0);     /* Clear RELAY_TTL -- CLOSE RELAY_1 */
               mode=INVERTER_RUNNING;
               break;
         case BUCK_STARTUP_1:                                               420
               REG_U42_clrb(0);     /* Clear RELAY_TTL -- CLOSE RELAY 1 */
               REG_U2_clrb(1);      /* Clear FAULT_CLR -- READY FAULTS  */
               REG_U2_setb(2);      /* Set   B2_ENABLE -- ENABLE B2     */
               mode=BUCK_STARTUP_2;
               swt_count=0;                 /* Reset the swt_count */
               swt_delay=BUCK_DC_DELAY;     /* Set a new delay time*/
```

```
                break;
        case BUCK_STARTUP_2:
                REG_U42_setb(1);     /* Set  PWM_EN_TTL -- ENABLE PWM    */
                mode=BUCK_RUNNING;                                          430
                break;
        case BOOST_STARTUP_1:
                REG_U42_clrb(0);     /* Clear RELAY_TTL -- CLOSE RELAY 1 */
                REG_U2_clrb(1);      /* Clear FAULT_CLR -- READY FAULTS  */
                REG_U2_setb(0);      /* Set   B1_ENABLE -- ENABLE B1     */
                mode=BOOST_STARTUP_2;
                swt_count=0;                    /* Reset the swt_count */
                swt_delay=DC_DC_DELAY;          /* Set a new delay time*/
                break;
        case BOOST_STARTUP_2:                                               440
                REG_U42_setb(1);     /* Set  PWM_EN_TTL -- ENABLE PWM    */
                mode=BOOST_SOFT_START;
                swt_count=0;                    /* Reset the swt_count */
                swt_delay=SOFT_START_DELAY;     /* Set a new delay time*/
                break;
        case BOOST_SOFT_START:
                REG_U42_setb(1);     /* Set  PWM_EN_TTL -- ENABLE PWM    */
                mode=BOOST_RUNNING;
                break;
        default: break;                                                     450
     }
  }

        /* Since the line-sync interrupt does not run in */
        /* inverter mode, do some extra stuff.           */
  if((ctrl_mode==INVERTER_MODE) && (mode!=OFF))
  {
     if(safety_mode!=NONE)
     {
        if(check_com())                                                     460
        {
           err_num=11;
           halt_converter();
           IRQ_EXIT;
           return;
        }
     }
     ad_command =  8;          /* Begin a Channel 0 A/D conversion */
                               /* ACH0 is the output voltage VBST  */
  }
  IRQ_EXIT;                                                                 470
}


/*********************************************************************/
/** LINE_SYNC_INTERRUPT()  This procedure is triggered at the 120 Hz **/
/** frequency generated by the line sync hardware.  The line sync    **/
/** triggers an EXTINT on P2.2.                                      **/
/*********************************************************************/
interrupt void line_sync_interrupt(void)                                    480
{
   uchar hold_wsr, hold_ioport1;

   IRQ_ENTRY;
   cyc_count++;              /* Increment the line-cycle counter */

   /*** Read the COM_CHK line to check that the puck is inserted ***/
   if((mode!=OFF) && (safety_mode!=NONE))   /* Skip while the DC/DC starts, */
   {                                        /* or safety=NONE.              */
      if(check_com())                                                       490
      {
         err_num=11;
         halt_converter();
         IRQ_EXIT;
         return;
      }
   }

   if(update_asap==0)
      MDAC_write();             /* Write delayd icmd to MDAC if update_asap==0 */   500

   ad_command =  8;             /* Begin a Channel 0 A/D conversion */
                                /* ACH0 is the output voltage VBST  */
   IRQ_EXIT;
}


/*************************************************************/
/** BOOST_CURRENT_CONTROL()  This procedure contains all the **/
/** processing for boost mode charging-current control.      **/                510
/*************************************************************/
void boost_current_control(void)
```

```
{
                /************************************/
                /* Here are the control calculations */
                /************************************/
   fp_power = fp_vbst * fp_ibst;
   fp_x = fp_vbst * fp_vbst;
   fp_cmd_c = (fp_iref - fp_ibst) * fp_rseries + fp_vbst;
   fp_cmd_c = fp_cmd_c * fp_cmd_c;                          /* Calc command c[n]*/      520

   fp_delta_k = H1_FP * (fp_cmd_c - fp_x);                  /* Calc the H1 term */

                /************************************/
                /* Select ramp-tracking version or not. */
                /************************************/
   if(ramp_tracking)
      fp_delta_k = fp_delta_k + H2_FP * fp_error_last;      /* Calc the H2 term */
   else
      fp_delta_k = fp_delta_k + H2_FP * (fp_cmd_c - fp_x_last);  /* Calc the H2 term */  530

   fp_delta_k = fp_delta_k + HP_FP * (fp_power - fp_power_last); /* Power FF term    */
   fp_delta_k = fp_delta_k / (fp_vac * fp_vac);             /* Divide by vac^2  */

      /* Clip the delta_k command in STEP and RAMP modes. */
      /* This to prevents overshoot.                      */
   if(ctrl_mode==CURRENT_STEP || ctrl_mode==CURRENT_RAMP)
   {
      if(fp_delta_k > fp_delta_k_clip)
         fp_delta_k = fp_delta_k_clip;                                                  540
      else if(fp_delta_k < -fp_delta_k_clip)
         fp_delta_k = -fp_delta_k_clip;
   }

   fp_icmd = fp_icmd_last + fp_delta_k;

   if(mode==BOOST_SOFT_START)                   /* Cap the kmax command during the  */
      fp_kmax = fp_kmax_ss;                     /* soft-start period.               */
   else
      fp_kmax = fp_kmax_const / fp_vac;                                               550

   if(fp_icmd > fp_kmax)                        /* In case of too high a command.   */
      fp_icmd = fp_kmax;                        /* Cap the int_icmd at kmax         */
   if(fp_icmd > MAX_ICMD_FP)                    /* Is icmd valid?                   */
      fp_icmd = MAX_ICMD_FP;
   if(fp_icmd < 0)                              /* Is icmd negative?                */
      fp_icmd = 0;                              /* Then send a icmd = 0.            */

   int_icmd = (uint) fp_icmd;
                                                                                      560
}


/*************************************************************/
/** BUCK_CURRENT_CONTROL()  This procedure contains all the    **/
/** processing for buck mode discharge current control.       **/
/*************************************************************/
void buck_current_control()
{
                /************************************/                             570
                /* Here are the control calculations */
                /************************************/
   if(buck_loop==2)             /* Smoothed K command */
   {
      fp_key1 = (280.0 * fp_iref) + 60.0;
      fp_icmd = (0.90 * fp_icmd_last) + (0.10 * fp_key1);
   }
   else
   {
      if(buck_loop==0)          /* Power error control */                           580
         fp_delta_k = BUCK_GAIN_FP * (fp_iref - fp_ibst) * fp_vbst;/* Calc. new command */
      else if(buck_loop==1)     /* Current error control */
         fp_delta_k = BUCK_GAIN_FP * (fp_iref - fp_ibst) * 310.0;  /* Calc. new command */

      fp_delta_k = fp_delta_k / (fp_vac * fp_vac);              /* Divide by vac^2   */
      fp_icmd = fp_icmd_last + fp_delta_k;
   }

   fp_kmax = fp_kmax_const / fp_vac;
   if(fp_icmd > fp_kmax)                        /* In case of too high a command.   */  590
      fp_icmd = fp_kmax;                        /* Cap the int_icmd at kmax         */
   if(fp_icmd > MAX_ICMD_FP)                    /* Is icmd valid?                   */
      fp_icmd = MAX_ICMD_FP;
   if(fp_icmd < 0)                              /* Is icmd negative?                */
      fp_icmd = 0;                              /* Then send a icmd = 0.            */

   int_icmd = (uint) fp_icmd;
}
```

```
/*************************************************************/
/** POST_CONTROL_CALCS()  This procedure contains non-time- **/
/** critical control calcs.                                 **/
/*************************************************************/
void post_control_calcs()
{
        /*****************************************/
        /* These are non-time-critical control calcs */
        /*****************************************/
   fp_icmd_last = fp_icmd;
   fp_power_last = fp_power;
   fp_x_last = fp_x;
   fp_error_last = fp_cmd_c - fp_x;

          /*********************************/
          /* Here's the STEP/RAMP control  */
          /*********************************/
   step_dly++;
   if (ctrl_mode==CURRENT_STEP)
   {
      if(step_dly == 300)             /* Switch levels every 2.5 secs */
         fp_iref = fp_iset2;
      if(step_dly == 600)
         fp_iref = fp_iset;
      if(step_dly >= 900)
         step_dly = 299;
   }
   else if(ctrl_mode==CURRENT_RAMP)
   {
      if(fp_iset2 >= fp_iset)
      {
         fp_iref = fp_iref + ((fp_iset2 - fp_iset) / 300.0);
         if(fp_iref > fp_iset2)
         {
            fp_iref = fp_iset;
            step_dly = 0;
         }
      }
      if(fp_iset2 < fp_iset)
      {
         fp_iref = fp_iref - ((fp_iset - fp_iset2) / 300.0);
         if(fp_iref < fp_iset2)
         {
            fp_iref = fp_iset;
            step_dly = 0;
         }
      }
   }
   else
      fp_iref = fp_iset;
}


/*************************************************************************/
/** ANALOG_CONVERSION_DONE()  This procedure is executed upon the   **/
/** completion of any A/D conversion.  All four A/D's are executed  **/
/** in sucession and then the voltage-loop control code is called.  **/
/*************************************************************************/
interrupt void analog_conversion_done(void)
{
   uchar hold_wsr, hold_ioport1, ad_channel;

   di();                          /* Disable IRQ's since A/D is high priority */
   IRQ_ENTRY;
   int_ad_hi = (uint) ad_result_hi;              /* Read the HI and LO parts */
   int_ad_lo = (uint) ad_result_lo;              /* of the 10 bit value.     */
   a_d_value = (int_ad_lo >> 6) + (int_ad_hi << 2);    /* Store 10-bit value. */
   ad_channel = ad_result_lo & 7;

   switch(ad_channel)
   {
              /*************************************/
              /* A/D Channel 0  --->  OUTPUT VOLTAGE */
              /*************************************/
      case 0:
         int_key1 = 1023 - a_d_value;      /* To measure VBST, we must first subtract */
                                           /* a_d_value from 1023.  This corrects for */
                                           /* a minor bug in the hardware.            */
         fp_vbst = ((float) int_key1 * VBST_TO_FP) + VBST_OFFSET_FP; /* Calc FP value */
         if (fp_vbst > VBST_HIGH_FP)             /* Check if VBST is over VBST_HIGH.  */
         {
            err_num=4;
            halt_converter();
            IRQ_EXIT;
```

```
      return;
   }
   ad_command =  9;                          /* Begin a Channel 1 A/D conversion */
   break;
         /*************************************/
         /* A/D Channel 1  --->  INPUT VOLTAGE */                                    690
case 1:  /*************************************/
   fp_vac = (float) a_d_value * VAC_TO_FP;     /* Set VAC equal to the A/D value */
      /** Check the VAC levels when in BOOST or BUCK modes.   **/
      /** Ignore VAC levels during startup, since they spike. **/
      /** Clear VAC in INVERTER MODE since its inactive.      **/
   if(ctrl_mode==INVERTER_MODE)
      fp_vac = 0;
   else
   {
      if((mode!=BOOST_STARTUP_1) && (mode!=BUCK_STARTUP_1) && (mode!=OFF)          700
                                                          && (safety_mode==ALL))
      {
         if (fp_vac < VAC_LOW_FP)                  /* Check if VAC is too LOW */
         {
            err_num=2;
            halt_converter();
            IRQ_EXIT;
            return;
         }
         else if (fp_vac > VAC_HIGH_FP)            /* Check if VAC is too HIGH */   710
         {
            err_num=1;
            halt_converter();
            IRQ_EXIT;
            return;
         }
      }
   }
   ad_command = 13;                          /* Begin a Channel 5 A/D conversion */
   break;                                                                          720
            /*********************************/
            /* A/D Channel 5  --->  DC-DC I1 */
case 5:     /*********************************/
   if(ctrl_dir==FORWARD)
   {
      if(DC_I1_OFST > a_d_value)
         fp_ibst = 0;
      else
      {
         int_key1 = a_d_value - DC_I1_OFST;                                       730
         fp_ibst = (float) int_key1 * DC_I1_TO_FP;
      }
   }
   else
   {
      int_key1 = a_d   lue + DC_I1_OFST;
      fp_ibst = (flc c) int_key1 * DC_I1_TO_FP;
   }

   ad_command = 12;                          /* Begin a Channel 4 A/D conversion */  740
   break;
            /*********************************/
            /* A/D Channel 4  --->  DC-DC V1 */
case 4:     /*********************************/
   fp_key1 = (float) a_d_value * DC_V1_TO_FP;

   if(fp_key1 < VBST_OFFSET_FP)                /* Read the DC_V1 voltage and use */
      fp_vbst = fp_key1;                       /* it when vbst < VBST_OFFSET.    */
         /*********************************************/
         /* Now that all REQUIRED A/D's have been read */                          750
         /* Let's go see about some control.           */
         /*********************************************/
   if(ctrl_mode==DEBUG_CONST_K)
   {
      int_icmd = (uint) fp_const_k;
      MDAC_write();
   }
   else
   {
      if(mode==BOOST_SOFT_START || mode==BOOST_RUNNING)                            760
         boost_current_control();              /* Calc icmd for boost mode control */
      else if(mode==BUCK_RUNNING)
         buck_current_control();
      else
         int_icmd = 0;                         /* Otherwise send a zero command    */
      if(update_asap)
         MDAC_write();                         /* Write the result to the MDAC     */
      post_control_calcs();                    /* Go do extra calcs                */
   }
   ad_command = 11;                          /* Begin a Channel 3 A/D conversion */  770
```

```
      break;
            /*************************************/
            /* A/D Channel 3  --->  INPUT CURRENT  */
      case 3:   /*************************************/
         fp_iac = (float) a_d_value * IAC_TO_FP;     /* Set IAC equal to the A/D value */
            /** Check the IAC levels when in BOOST or BUCK modes.    **/
            /** Clear IAC in INVERTER MODE since its inactive.       **/
         if(ctrl_mode==INVERTER_MODE)
            fp_iac=0;
         else                                                                    780
         {                                                  /* Check if IAC is to high.*/
            if ((mode!=OFF) && (ip_iac > IAC_HIGH_FP) && (safety_mode==ALL))
            {
               err_num=3;
               halt_converter();
               IRQ_EXIT;
               return;
            }
         }
         ad_command = 14;                        /* Begin a Channel 6 A/D conversion */     790
         break;
                  /*************************************/
                  /* A/D Channel 6  --->  DC-DC V2 */
      case 6:     /*************************************/
         fp_dc_v2 = (float) a_d_value * DC_V2_TO_FP;

         ad_command = 15;                        /* Begin a Channel 7 A/D conversion */
                                                 /* ACH2 is the DC-DC I2             */
         break;
                  /*************************************/                          800
                  /* A/D Channel 7  --->  DC-DC I2 */
      case 7:     /*************************************/
         if(DC_I2_OFST > a_d_value)
            fp_dc_i2 = 0;
         else
         {
            int_key1 = a_d_value - DC_I2_OFST;
            fp_dc_i2 = (float) int_key1 * DC_I2_TO_FP;
         }
         break;                                                                  810
      default: break;
   }
   IRQ_EXIT;
   ei();
} /* End of analog_conversion_done ISR.  */


/*********************************************************************/
/** INT_PART()  Return the rounded integer part of a float.         **/
/*********************************************************************/                820
int int_part(float fp_value)
{
   fp_value = fp_value + 0.05;
   return((int) fp_value);
}

/*********************************************************************/
/** DEC_PART()  Return the rounded 1st decimal place of a float.    **/
/*********************************************************************/
int dec_part(float fp_value)                                                       830
{
   fp_value = fp_value + 0.05;
   return((int) ((fp_value - (int) fp_value) * 10));
}

/*********************************************************************/
/** ROUND_FP()  Return the rounded integer value of a float.        **/
/*********************************************************************/
int round_fp(float fp_value)
{                                                                                  840
   fp_value = fp_value + 0.5;
   return((int) fp_value);
}


/*********************************************************************/
/** MAKE_BAR_GRAPH()  This procedure constructs a neat little bar-  **/
/** graph for the LCD display.                                      **/
/*********************************************************************/
void make_bar_graph(float level,float min_lev,float max_lev)                       850
{
   int i,j;

   sprintf(lcd_line2,"%d\0",round_fp(min_lev));
   fp_temp1 = (level - min_lev) / (max_lev - min_lev);
   fp_temp1 = fp_temp1 * 18;
```

```
   j= (int) fp_temp1;
   for(i=0;i<j;i++)
      strcat(lcd_line2,"\377\0");
   strcat(lcd_line2,"\377\0");
   for(i=0;i<(18-j);i++)
      strcat(lcd_line2,"\245\0");
   sprintf(lcd_line2,"%s%d\0",lcd_line2,round_fp(max_lev));
}


/*****************************************  **********************/
/** START_CONVERTER()  This procedure starts the converter.          **/
/*************************************************************************/
void start_converter(void)
{
   if(safety_mode!=NONE)   /* Skip comm check if safety=NONE */
   {
      if(check_com())
      {
         err_num=12;
         halt_converter();
         return;
      }
   }

   menu_mode=CONVERTER_RUNNING;
   debug_lev=0;
   di();                   /* Disable interrupts */
   fp_ibat=0;              /* Clear the sampled variables */
   fp_vbat=0;
   fp_iac=0;
   fp_vac=0;
   fp_dc_v2=0;
   fp_dc_i2=0;
   pin_filtered=0;
   pdc_filtered=0;
   step_dly=0;
   line_cycs=6;
   fp_icmd=0;
   int_icmd=0;
   fp_iref=fp_iset;
   fp_delta_k_clip=DELTA_K_CLIP_FP*fabs(fp_iset2-fp_iset);

   if(ctrl_mode==INVERTER_MODE)
   {
                  /*************************/
                  /*** Start INVERTER MODE **/
                  /*************************/
      leds=LED_PD;          /* Turn ON the POWER & DISCHARGE LEDs */
      reg_U2=0x52;          /* Set B1_ENABLE     = 0   DISABLE B1     */
                            /* Set FAULT_CLR     = 1   CLEAR FAULTS   */
                            /* Set B2_ENABLE     = 0   DISABLE B2     */
                            /* Set FB_LDAC       = 0                  */
                            /* Set B1_RLY_SET    = 1   SET B1 RELAYS  */
                            /* Set B1_RLY_RESET  = 0                  */
                            /* Set B2_RLY_SET    = 1   SET B2 RELAYS  */
                            /* Set B2_RLY_RESET  = 0                  */
      SET_reg_U2(reg_U2);
      wait_fun(1000);
      reg_U42=0xF1;         /* Set RELAY_TTL   = 1     OPEN RELAY_1     */
                            /* Set PWM_EN_TTL  = 0     DISABLE the PWM  */
                            /* Set DS_A0       = 0                      */
                            /* Set DS_A1       = 0                      */
                            /* Set L-4         = 1                      */
                            /* Set L-5 (B_MDE) = 1     BUCK_MODE (1)    */
                            /* Set L-6 (RS)    = 1     LCD Data Mode    */
                            /* Set L-7 (RLY2)  = 1     OPEN RELAY_2     */
      SET_reg_U42(reg_U42);
      mode=INVERTER_STARTUP_1;
   }
   else if(ctrl_dir==FORWARD)
   {
                  /*************************/
                  /**** Start BOOST MODE ****/
                  /*************************/
      leds = LED_PC;        /* Turn ON the POWER & CHARGE LEDs */
      reg_U2 = 0x92;        /* Set B1_ENABLE     = 0   DISABLE B1     */
                            /* Set FAULT_CLR     = 1   CLEAR FAULTS   */
                            /* Set B2_ENABLE     = 0   DISABLE B2     */
                            /* Set FB_LDAC       = 0                  */
                            /* Set B1_RLY_SET    = 1   SET B1 RELAYS  */
                            /* Set B1_RLY_RESET  = 0                  */
                            /* Set B2_RLY_SET    = 0   CLR B2 RELAYS  */
                            /* Set B2_RLY_RESET  = 1                  */
      SET_reg_U2(reg_U2);
      wait_fun(1000);
```

```
      reg_U42 = 0x51;      /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                           /* Set PWM_EN_TTL = 0     DISABLE the PWM */
                           /* Set DS_A0      = 0                      */
                           /* Set DS_A1      = 0                      */
                           /* Set L-4        = 1                      */
                           /* Set L-5 (B_MDE)= 0     BOOST_MODE (0)   */
                           /* Set L-6 (RS)   = 1     LCD Data Mode    */
                           /* Set L-7 (RLY2) = 0     CLOSE RELAY_2    */                    950
      SET_reg_U42(reg_U42);
      int_fbcmd = 4000;
      FBDAC_write();     /* Write duty-cycle command to the FB controller */
      mode=BOOST_STARTUP_1;
   }
   else if(ctrl_dir==REVERSE)
   {
                  /*************************/
                  /**** Start BUCK MODE ****/
                  /*************************/                                              960
      leds=LED_PD;         /* Turn ON the POWER & DISCHARGE LEDs */
      reg_U2=0x52;         /* Set B1_ENABLE     = 0   DISABLE B1    */
                           /* Set FAULT_CLR     = 1   CLEAR FAULTS  */
                           /* Set B2_ENABLE     = 0   DISABLE B2    */
                           /* Set FB_LDAC       = 0                 */
                           /* Set B1_RLY_SET    = 1   SET B1 RELAYS */
                           /* Set B1_RLY_RESET  = 0                 */
                           /* Set B2_RLY_SET    = 1   SET B2 RELAYS */
                           /* Set B2_RLY_RESET  = 0                 */
      SET_reg_U2(reg_U2);                                                                  970
      wait_fun(1000);
      reg_U42=0x71;        /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                           /* Set PWM_EN_TTL = 0     DISABLE the PWM */
                           /* Set DS_A0      = 0                      */
                           /* Set DS_A1      = 0                      */
                           /* Set L-4        = 1                      */
                           /* Set L-5 (B_MDE)= 1     BUCK_MODE (1)    */
                           /* Set L-6 (RS)   = 1     LCD Data Mode    */
                           /* Set L-7 (RLY2) = 0     CLOSE RELAY_2    */
      SET_reg_U42(reg_U42);                                                                980
      mode=BUCK_STARTUP_1;
   }
   wait_fun(5000);              /* Delay here for a split sec and   */

   swt_count=0;                 /* Reset the swt_count */
   swt_delay=DC_DC_DELAY;       /* Set a new delay time*/
   int_pending = 0;
   if(ctrl_mode==INVERTER_MODE)
      int_mask = 0x13;          /* Unmask TIMER_MASK, HSI0_MASK, AD_MASK. */
   else                                                                                    990
      int_mask = 0x93;          /* Unmask ALL interrupts: TIMER_MASK, */
                                /* EXTINT_MASK, HSI0_MASK, AD_MASK.   */
   ei();                        /* Re-enable interrupts */

   wait_fun(5000);              /* Delay here for a split sec and   */
   button_id=0;                 /* flush the key buffer.            */

   REG_U42_setb(0);             /* Set   RELAY_TTL -- OPEN RELAY 1  */
}                                                                                         1000

/*************************************************************************/
/** SELECT_CONTROL_MODE()  This procedure controls the MENU & DISPLAY **/
/*************************************************************************/
void select_control_mode(void)
{
   switch(ctrl_mode)
   {
   case 1: sprintf(lcd_line1,"Control Mode:   CURRENT STEP RESPONSE   ");
           break;                                                                        1010
   case 2: sprintf(lcd_line1,"Control Mode:   CURRENT RAMP RESPONSE   ");
           break;
   case 3: sprintf(lcd_line1,"Control Mode:      INVERTER MODE        ");
           break;
   case 4: sprintf(lcd_line1,"Control Mode:    DEBUG:  CONSTANT K     ");
           break;
   default: sprintf(lcd_line1,"Control Mode:     CURRENT CONTROL       ");
            break;
   }
   sprintf(lcd_line2,"    (Press <- or -> to change modes)    ");                        1020
   if(button_id!=0)
   {
      switch(button_id)
      {
      case MENU_ID: if(ctrl_mode==INVERTER_MODE)
                       menu_mode=SELECT_SAFETY_MODE;      /* MENU */
                    else
                       menu_mode=SELECT_CONTROL_DIR;      /* MENU */
```

```
                    break;
      case LEFT_ID: if(ctrl_mode==CURRENT_CONTROL)     /* LEFT ARROW KEY  */
                       ctrl_mode=DEBUG_CONST_K;
                    else
                       ctrl_mode--;
                    break;
      case RIGHT_ID: if(ctrl_mode==DEBUG_CONST_K)      /* RIGHT ARROW KEY */
                       ctrl_mode=CURRENT_CONTROL;
                     else
                       ctrl_mode++;
                     break;
      default: break;                                  /* ALL OTHER KEYS  */
      }
      button_id=0;
      button_db=0;
   }
}


/*************************************************************************/
/** SELECT_CONTROL_DIRECTION()  This procedure controls the MENU & DISPLAY **/
/*************************************************************************/
void select_control_direction(void)
{
   switch(ctrl_dir)
   {
      case 1: sprintf(lcd_line1,"Power Flow:    Forward (Charging)       \0");
              break;
      default: sprintf(lcd_line1,"Power Flow:    Reverse (Discharging)    \0");
               break;
   }
   sprintf(lcd_line2,"  (Press <- or -> to change direction)  \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SELECT_SAFETY_MODE;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
                       if(ctrl_dir==REVERSE)
                          ctrl_dir=FORWARD;
                       else
                          ctrl_dir--;
                       break;
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                        if(ctrl_dir==FORWARD)
                           ctrl_dir=REVERSE;
                        else
                           ctrl_dir++;
                        break;
         default: break;
      }
      button_id=0;
      button_db=0;
   }
}


/*************************************************************************/
/** SELECT_SAFETY_MODE()  This procedure controls the MENU & DISPLAY **/
/*************************************************************************/
void select_safety_mode(void)
{
   switch(safety_mode)
   {
      case SOME: sprintf(lcd_line1,"Safety Checks:       SOME ENABLED        \0");
                 break;
      case NONE: sprintf(lcd_line1,"Safety Checks:    NONE - Be Careful!     \0");
                 break;
      default: sprintf(lcd_line1,"Safety Checks:       ALL ENABLED         \0");
               break;
   }
   sprintf(lcd_line2,"       (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
                       if(safety_mode==ALL)
                          safety_mode=NONE;
                       else
                          safety_mode--;
```

```
                    break;
        case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                    if(safety_mode==NONE)
                       safety_mode=ALL;
                    else
                       safety_mode++;                                    1120
                    break;
        default: break;
      }
      button_id=0;
      button_db=0;
   }
}

/*********************************************************************/   1130
/** SPARE_SELECT_MENU()  This procedure controls the MENU & DISPLAY **/
/*********************************************************************/
void spare_select_menu(void)
{
   if((ctrl_dir==REVERSE) || (ctrl_mode==INVERTER_MODE) || (ctrl_mode==DEBUG_CONST_K))
   {
      menu_mode=SPARE_SELECT_MENU2;
      return;
   }
   if(ramp_tracking)                                                    1140
      sprintf(lcd_line1,"Boost Control Loop:  RAMP TRACKING            \0");
   else
      sprintf(lcd_line1,"Boost Control Loop:  NON-RAMP TRACKING        \0");
   sprintf(lcd_line2,"       (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU2;                    1150
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                       if(ramp_tracking)
                          ramp_tracking=0;
                       else
                          ramp_tracking=1;
                       break;
         default: break;
      }                                                                 1160
      button_id=0;
      button_db=0;
   }
}

/*********************************************************************/
/** SPARE_SELECT_MENU2() This procedure controls the MENU & DISPLAY **/
/*********************************************************************/
void spare_select_menu2(void)                                           1170
{
   if(ctrl_mode==INVERTER_MODE)
   {
      menu_mode=SPARE_SELECT_MENU3;
      return;
   }
   sprintf(lcd_line1,"Current Kmax @ 120Vac:  %d                    \0",
                                              (int) (fp_kmax_const/120.0));
   sprintf(lcd_line2,"       (Press <- or -> to change)        \0");
   if(button_id!=0)                                                     1180
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU3;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
                       fp_kmax_const-=1200.0;
                       break;
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/                      1190
                       fp_kmax_const+=1200.0;
                       break;
         default: break;
      }
      button_id=0;
      button_db=0;
   }
}

                                                                        1200
```

```
/*****************************************************************/
/** SPARE_SELECT_MENU3() This procedure controls the MENU & DISPLAY **/
/*****************************************************************/
void spare_select_menu3(void)
{
   if((ctrl_dir==REVERSE) || (ctrl_mode==INVERTER_MODE) || (ctrl_mode==DEBUG_CONST_K))
   {
      menu_mode=SPARE_SELECT_MENU4;
      return;
   }                                                                          1210
   sprintf(lcd_line1,"Current Soft-Start K value: %d                  \0",
                                                          (int) fp_kmax_ss);
   sprintf(lcd_line2,"      (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU4;
                       break;                                               1220
         case LEFT_ID: /*** LEFT ARROW KEY ***/
                       fp_kmax_ss-=2.0;
                       break;
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                       fp_kmax_ss+=2.0;
                       break;
         default: break;
      }
      button_id=0;
      button_db=0;                                                          1230
   }
}


/*****************************************************************/
/** SPARE_SELECT_MENU4()  This procedure controls the MENU & DISPLAY **/
/*****************************************************************/
void spare_select_menu4(void)
{
   if((ctrl_mode==INVERTER_MODE) || (ctrl_mode==DEBUG_CONST_K))           1240
   {
      menu_mode=SPARE_SELECT_MENU5;
      return;
   }
   if(update_asap)
      sprintf(lcd_line1,"Control loop updates:  ASAP                  \0");
   else
      sprintf(lcd_line1,"Control loop updates:  1 Unit Delayed        \0");
   sprintf(lcd_line2,"      (Press <- or -> to change)        \0");
   if(button_id!=0)                                                         1250
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU5;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                       if(update_asap)
                          update_asap=0;                                    1260
                       else
                          update_asap=1;
                       break;
         default: break;
      }
      button_id=0;
      button_db=0;
   }
}
                                                                            1270

/*****************************************************************/
/** SPARE_SELECT_MENU5()  This procedure controls the MENU & DISPLAY **/
/*****************************************************************/
void spare_select_menu5(void)
{
   if((ctrl_dir==FORWARD) || (ctrl_mode==INVERTER_MODE) || (ctrl_mode==DEBUG_CONST_K))
   {
      menu_mode=CONVERTER_STOPPED;
      return;                                                               1280
   }
   switch(buck_loop)
   {
      case 0: sprintf(lcd_line1,"Buck Control Loop:  Power Error          \0");
              break;
      case 1: sprintf(lcd_line1,"Buck Control Loop:  Current Error        \0");
```

```
            break;
      case 2: sprintf(lcd_line1,"Buck Control Loop:  Smoothed K              \0");
            break;
      default: break;                                                                  1290
   }
   sprintf(lcd_line2,"      (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=CONVERTER_STOPPED;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/                                       1300
                       if(buck_loop==0)
                          buck_loop=2;
                       else
                          buck_loop--;
                       break;
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                        if(buck_loop==2)
                           buck_loop=0;
                        else
                           buck_loop++;                                               1310
                        break;
         default: break;
      }
      button_id=0;
      button_db=0;
   }
}


/*********************************************************************/                1320
/** LCD_MENU_RUNNING()  This procedure controls the MENU & DISPLAY **/
/*********************************************************************/
void lcd_menu_running(void)
{
                /*** LP-Filter Pin & Pdc ***/
   pin_filtered = (pin_filtered * 0.8) + (fp_vac * fp_iac * 0.2);
   pdc_filtered = (pdc_filtered * 0.8) + (fp_dc_v2 * fp_dc_i2 * 0.2);

   if(debug_lev==0)
   {                                                                                   1330
      if(ctrl_dir==FORWARD)
      {
            /*** Calculate forward Effeciency ***/
         fp_temp1 = 100.0 * pdc_filtered / pin_filtered;
         if(fp_temp1 < 100.0)
            sprintf(lcd_line1,"Pdc/Pin=%2d.%1d%% \0",int_part(fp_temp1),dec_part(fp_temp1));
         else
            sprintf(lcd_line1,"Pdc/Pin=**.*%% \0");
         sprintf(lcd_line2,"%4dW/%4dW\0",round_fp(pdc_filtered), round_fp(pin_filtered));
      }                                                                                1340
      else
      {
            /*** Calculate reverse Effeciency ***/
         fp_temp1 = 100.0 * pin_filtered / pdc_filtered;
         if(fp_temp1 < 100.0)
           sprintf(lcd_line1,"Pin/Pdc=%2d.%1d%% \0", int_part(fp_temp1), dec_part(fp_temp1));
         else
            sprintf(lcd_line1,"Pin/Pdc=**.*%% \0");
         sprintf(lcd_line2,"%4dW/%4dW\0", round_fp(pin_filtered), round_fp(pdc_filtered));
      }                                                                                1350

         /*** Now finish laying out the LCD display ***/
      sprintf(lcd_line1,"%sVac=%3dV Vbt=%3dV Vdc=%3dV ", lcd_line1, round_fp(fp_vac),
              round_fp(fp_vbst), round_fp(fp_dc_v2));
      sprintf(lcd_line2,"%s  Iac=%2d.%1dA Ibt=%1d.%1dA Idc=%1d.%1dA ", lcd_line2,
              int_part(fp_iac), dec_part(fp_iac), int_part(fp_ibst), dec_part(fp_ibst),
                 int_part(fp_dc_i2), dec_part(fp_dc_i2));
   }
   else if(debug_lev==1)
   {                                                                                   1360
      sprintf(lcd_line1,"DB: VAC=%3d.%1dV   VBT=%3d.%1dV   VDC=%3d.%1dV   ",
              int_part(fp_vac), dec_part(fp_vac), int_part(fp_vbst), dec_part(fp_vbst),
                 int_part(fp_dc_v2), dec_part(fp_dc_v2));
      sprintf(lcd_line2,"#1: IAC=%3d.%1dV   IBT=%3d.%1dV   IDC=%3d.%1dV   ",
              int_part(fp_iac), dec_part(fp_iac), int_part(fp_ibst), dec_part(fp_ibst),
                 int_part(fp_dc_i2), dec_part(fp_dc_i2));
   }
   else if(debug_lev==2)
   {
      sprintf(lcd_line1,"DB: K=%4d   Kmax=%4d                                  ",   1370
              int_icmd, (uint) fp_kmax);
      sprintf(lcd_line2,"#2:                                                    ");
```

```
    }
    else if(debug_lev==3)
    {
       if(mode==BOOST_RUNNING)        /* If boosting allow feedforward ON/OFF */
       {
          if(check_bit(reg_U2,4))
             sprintf(lcd_line1,"    Feedforward Ripple Cancel: ON       ");
          else                                                                    1380
             sprintf(lcd_line1,"    Feedforward Ripple Cancel: OFF      ");
          sprintf(lcd_line2,"     ( OFF <-- arrow keys --> ON )       ");
       }
       else
          debug_lev=0;
    }

    if(button_id!=0)
    {
       switch(button_id)                                                      1390
       {
          case MENU_ID: /*** MENU KEU ***/
                        debug_lev++;              /* Switch to the debug screens  */
                        if(debug_lev > 3)
                           debug_lev = 0;
                        break;
          case LEFT_ID: /*** LEFT ARROW KEY ***/
                        if(debug_lev==3)
                        {
                           REG_U2_clrb(4);                /* Toggle U2-4 */       1400
                        }
                        else         /* If we are running then allow a ref change */
                        {
                           if((mode==BOOST_RUNNING) || (mode==BUCK_RUNNING))
                           {
                              if(fp_iset > (CUR_MIN_FP + CUR_STEP_FP))
                                 fp_iset = fp_iset - CUR_STEP_FP;
                              else
                                 fp_iset = CUR_MIN_FP;
                           }                                                      1410
                        }
                        break;
          case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                         if(debug_lev==3)
                         {
                            REG_U2_setb(4);              /* Toggle U2-4 */
                         }
                         else         /* If we are running then allow a ref change */
                         {
                            if((mode==BOOST_RUNNING) || (mode==BUCK_RUNNING))    1420
                            {
                               if(fp_iset < (CUR_MAX_FP - CUR_STEP_FP))
                                  fp_iset = fp_iset + CUR_STEP_FP;
                               else
                                  fp_iset = CUR_MAX_FP;
                            }
                         }
                         break;
          case START_ID: /*** START BUTTON ***/
                         err_num=0;                  /* Shutdown the converter    */
                         halt_converter();           /* without an error flag.    */  1430
                         break;
          default: break;
       }
       button_id=0;
       button_db=0;
    }
}


                                                                               1440
/*********************************************************************/
/** LCD_MENU_HALT()  This procedure controls the MENU & DISPLAY     **/
/*********************************************************************/
void lcd_menu_halt(void)
{
   switch(err_num)
   {
   case 1: sprintf(lcd_line1,"**** ERROR #%d: Vac too HIGH (%dV) ********",err_num,
                   round_fp(fp_vac));
           break;                                                              1450
   case 2: sprintf(lcd_line1,"***** ERROR #%d: Vac too LOW (%dV) ********",err_num,
                   round_fp(fp_vac));
           break;
   case 3: sprintf(lcd_line1,"**** ERROR #%d: Iac too HIGH (%d.%dA) ********",err_num,
                int_part(fp_iac), dec_part(fp_iac));
           break;
   case 4: sprintf(lcd_line1,"***** ERROR #%d: Vbt too HIGH (%dV) ********",err_num,
                   round_fp(fp_vbst));
```

```
            break;
   case 5: sprintf(lcd_line1,"**** ERROR #%d: Ibt too HIGH (%d.%dA) ********",err_num,
                  int_part(fp_ibst), dec_part(fp_ibst));
            break;
   case 6:
   case 8: sprintf(lcd_line1,"ERROR #%d: DC#1 Flt (Vbt=%dV Ibt=%d.%dA)      ",err_num,
                     round_fp(fp_vbst), int_part(fp_ibst), dec_part(fp_ibst));
            break;
   case 7: sprintf(lcd_line1,"ERROR #%d: DC#2 Flt (Vdc=%uV Idc=%u.%uA)      ",err_num,
                     round_fp(fp_dc_v2), int_part(fp_dc_i2), dec_part(fp_dc_i2));
            break;
   case 9: sprintf(lcd_line1,"**** ERROR #%d: NO INPUT (Cycles = %u) *******",err_num,
                     line_cycs);
            break;
   case 10: sprintf(lcd_line1,"*** ERROR #%d: Ibt > Iset @ Vdc = %dV ******",err_num,
                     round_fp(fp_dc_v2));
            break;
   case 11: sprintf(lcd_line1,"***** ERROR #%d: PLEASE CHECK PADDLE *******",err_num);
            break;
   case 12: sprintf(lcd_line1,"** CHECK THE PADDLE AND SECONDARY POWER **",err_num);
            break;
   default: sprintf(lcd_line1,"***** A fault has occured: ERROR #%u ******",err_num);
            break;
   }
   sprintf(lcd_line2,"    (Press ANY KEY to clear the error)    ");

   if(button_id!=0)                    /* If ANY KEY is pressed then...  */
   {
      menu_mode=CONVERTER_STOPPED;     /* Return to MAIN MENU            */
      leds=LED_P;                      /* Turn ON the POWER LED          */

      button_id=0;
      button_db=0;
   }
}


/*********************************************************************/
/** LCD_MENU_STOP()  This procedure controls the MENU & DISPLAY     **/
/*********************************************************************/
void lcd_menu_stop(void)
{
   switch(ctrl_mode)
   {
   case 1: /* Current STEP */
   case 2: /* Current RAMP */
           sprintf(lcd_line1,"Select a second current and press START.");
           make_bar_graph(fp_iset2,CUR_MIN_FP,CUR_MAX_FP);
           sprintf(lcd_line2,"%sA  (Level=%d.%dA)        ", lcd_line2, int_part(fp_iset2),
                     dec_part(fp_iset2));
           break;
   case 3: /* Inverter MODE */
           sprintf(lcd_line1,"Inverter mode - no options. Press START.");
           sprintf(lcd_line2,"                                        ");
           break;
   case 4: /* DEBUG: Constant K */
           sprintf(lcd_line1,"Select a value for K and press START.   ");
           make_bar_graph(fp_const_k,CK_MIN_FP,CK_MAX_FP);
           sprintf(lcd_line2,"%sA  (K=%4d)        ", lcd_line2, (uint) fp_const_k);
           break;
   default: /* Current Control */
            sprintf(lcd_line1,"Select a current level then press START.");
            make_bar_graph(fp_iset,CUR_MIN_FP,CUR_MAX_FP);
            sprintf(lcd_line2,"%sA  (Level=%d.%dA)        ", lcd_line2, int_part(fp_iset),
                     dec_part(fp_iset));
            break;
   }

   if(button_id!=0)           /***  Now interpret the menu KEYS ***/
   {
      switch(button_id)
      {
      case MENU_ID: /*** MENU KEU ***/
                    menu_mode = SELECT_CONTROL_MODE;
                    break;
      case LEFT_ID: /*** LEFT ARROW KEY ***/
                    switch(ctrl_mode)
                    {
                       case 1: /* Current STEP */
                       case 2: /* Current RAMP */
                               if(fp_iset2 > (CUR_MIN_FP + CUR_STEP_FP))
                                  fp_iset2 = fp_iset2 - CUR_STEP_FP;
                               else
                                  fp_iset2 = CUR_MIN_FP;
                               break;
                       case 3: /* Inverter mode */
```

```
                        break;
                    case 4: /* Constant K */
                            if(fp_const_k > (CK_MIN_FP + CK_STEP_FP))
                               fp_const_k = fp_const_k - CK_STEP_FP;
                            else
                               fp_const_k = CK_MIN_FP;
                            break;
                    default: /* Current Control */
                            if(fp_iset > (CUR_MIN_FP + CUR_STEP_FP))
                               fp_iset = fp_iset - CUR_STEP_FP;
                            else
                               fp_iset = CUR_MIN_FP;
                            break;
                  }
                  break;
    case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                  switch(ctrl_mode)
                  {
                     case 1: /* Current STEP */
                     case 2: /* Current RAMP */
                            if(fp_iset2 < (CUR_MAX_FP - CUR_STEP_FP))
                               fp_iset2 = fp_iset2 + CUR_STEP_FP;
                            else
                               fp_iset2 = CUR_MAX_FP;
                            break;
                     case 3: /* Inverter mode */
                             break;
                     case 4: /* Constant K */
                            if(fp_const_k < (CK_MAX_FP - CK_STEP_FP))
                               fp_const_k = fp_const_k + CK_STEP_FP;
                            else
                               fp_const_k = CK_MAX_FP;
                            break;
                     default: /* Current Control */
                             if(fp_iset < (CUR_MAX_FP - CUR_STEP_FP))
                                fp_iset = fp_iset + CUR_STEP_FP;
                             else
                                fp_iset = CUR_MAX_FP;
                             break;
                  }
                  break;
    case START_ID: /*** START BUTTON ***/
                  menu_mode = PREPARE_TO_START;    /* Goto the about to start menu */
                  break;
    default: break;
    }
    button_id=0;
    button_db=0;
  }
}


/*********************************************************************/
/** LCD_MENU_START()  This procedure controls the MENU & DISPLAY    **/
/*********************************************************************/
void lcd_menu_start(void)
{
  if(ctrl_mode==INVERTER_MODE)
    sprintf(lcd_line1,"About to begin INVERTER MODE.           \0");
  else if(ctrl_mode==DEBUG_CONST_K)
    sprintf(lcd_line1,"About to begin const K.  USE CAUTION!!  \0");
  else
  {
    if(ctrl_dir==FORWARD)
      sprintf(lcd_line1,"About to begin CHARGING.     ");
    else
      sprintf(lcd_line1,"About to begin DISCHARGING   ");

    if(ctrl_mode==CURRENT_CONTROL)
      sprintf(lcd_line1,"%s(Fixed I)   \0",lcd_line1);
    else if(ctrl_mode==CURRENT_STEP)
      sprintf(lcd_line1,"%s(Step I)    \0",lcd_line1);
    else
      sprintf(lcd_line1,"%s(Ramp I)    \0",lcd_line1);
  }
  sprintf(lcd_line2,"(Press START to begin or MENU to abort.) \0");

  if(button_id!=0)           /***  Now interpret the menu KEYS ***/
  {
    switch(button_id)
    {
    case MENU_ID: /*** MENU KEY ***/
    case LEFT_ID: /*** LEFT ARROW KEY ***/
    case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                  menu_mode = CONVERTER_STOPPED;     /* Return to the Main menu */
                  break;
```

```
      case START_ID: /*** START BUTTON ***/
                     start_converter();   /* Start with safety ON */
                     break;
      default: break;
      }
      button_id=0;
      button_db=0;
   }
}
                                                                          1640

/*************************************************************************/
/** INITIALIZE_GLOBALS()  This procedure inits various global variables **/
/*************************************************************************/
void initialize_globals(void)
{
   hold_iop2=0x40;                 /* Initialize ioport2                 */
   hold_ios0=0x00;                 /* Initialize ios0                    */
   debug_lev=0;                    /* Zero the debug level.              */
   safety_mode=ALL;                /* Zero the safety level.             */   1650
   line_cycs=6;                    /* Preset the line-cycle counter      */
   step_dly=0;                     /* Zero the step delay counter.       */
   mode=OFF;                       /* Set the startup mode to OFF.       */
   int_icmd=0;                     /* Set the initial current command to zero.  */
   fp_iref=0;                      /* Set an initial value for iref.     */
   fp_iset=CUR_MIN_FP;             /* Initialize the current setting.    */
   fp_iset2=CUR_MIN_FP;            /* Initialize the current setting.    */
   fp_const_k = CK_MIN_FP;         /* Initialize the k setting.          */
   menu_mode=CONVERTER_STOPPED;    /* Initialize the menu mode.          */
   ctrl_mode=CURRENT_CONTROL;      /* Initialize CONSTANT CURRENT control.  */   1660
   ctrl_dir=FORWARD;               /* Set control direction.             */
   leds=LED_P;                     /* Set only the POWER LED for starters.  */
   strobe=0;                       /* Clear strobe.                      */
   button_id=0;                    /* Clear the button_id.               */
   button_db=0;                    /* Clear the button debounce counter  */
   reset_ints=TRUE;                /* Set the reset_ints flag.           */

   fp_vbst=0;                      /* Clear floating point a/d values    */
   fp_vac=0;
   fp_ibst=0;                                                             1670
   fp_iac=0;
   fp_dc_v2=0;
   fp_dc_i2=0;
   fp_rseries=RSERIES_FP;
   fp_kmax_const=KMAX_CONST_FP;
   fp_kmax_ss=SS_KMAX_FP;
   ramp_tracking=1;
   update_asap=0;
   buck_loop=2;
}                                                                         1680

/*************************************************************************/
/** INITIALIZE_HARDWARE()  This procedure inits the hardware setup.  **/
/*************************************************************************/
void initialize_hardware(void)
{
   wsr = 0;
   ioc1 = 0x54;          /* Enable HSO.4, HSO.5 and set P2.2 as EXTINT */
   ioport2 = hold_iop2; /* Preset ioport2 */                              1690

   HSO_set(0);          /* Set the /LDAC  pin */
   HSO_set(1);          /* Set the /CSMSB pin */
   HSO_set(2);          /* Set the /WR    pin */
   HSO_set(3);          /* Set the /CSLSB pin */
   HSO_clr(4);          /* Set the reg_U42 LATCH LOW */
   HSO_set(5);          /* Set the /WR line on the DISPLAY to HIGH */

   MDAC_write();        /* Write a starting int_icmd = 0 */
                        /* to the DAC.                   */              1700

   reg_U42=0x91;        /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM  */
                        /* Set DS_A0      = 0                      */
                        /* Set DS_A1      = 0                      */
                        /* Set L-4        = 1                      */
                        /* Set L-5 (B_MDE)= 0     BOOST_MODE (0)   */
                        /* Set L-6 (RS)   = 0     LCD Inst. Mode   */
                        /* Set L-7 (RLY2) = 1     OPEN RELAY_2     */
   SET_reg_U42(reg_U42);                                                  1710

   reg_U2=0x92;         /* Set B1_ENABLE    = 0   DISABLE B1     */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS   */
                        /* Set B2_ENABLE    = 0   DISABLE B2     */
                        /* Set FB_LDAC      = 0                  */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS  */
```

```
                    /* Set B1_RLY_RESET = 0                */
                    /* Set B2_RLY_SET   = 0   CLR B2 RELAYS */
                    /* Set B2_RLY_RESET = 1                */
   SET_reg_U2(reg_U2);                                                    1720

   strcpy(display,"INIT");
   write_hp_display();          /* Direct write "INIT" to the screen */

   wait_fun(10000);             /* Wait for the LCD to become ready */
                                /* after a powerup.                 */
   ioport1 = 0x3C;              /* Init the LCD FUNCTION-SET Instruction */
   P2_setb(5);                  /* Strobe P2-5 to latch LCD data */
   P2_clrb(5);
   wait_fun(100);                                                         1730

   ioport1 = 0x0C;              /* Set LCD display control to ON */
   P2_setb(5);                  /* Strobe P2-5 to latch LCD data */
   P2_clrb(5);
   wait_fun(100);

   ioport1 = 0x01;              /* Clear the LCD */
   P2_setb(5);                  /* Strobe P2-5 to latch LCD data */
   P2_clrb(5);
   wait_fun(30000);             /* Wait a few milliseconds for the clear */   1740

   SET_leds(leds);              /* Turn on the POWER LED only */
}


/*****************************************************************/
/** MAIN()  This is the main procedure.  Most of the processing  **/
/** is really done in the ISR routines.  The main routine calls  **/
/** a number of procedure that maintain the menu system.         **/
/*****************************************************************/                                                   1750
main(void)
{
          /************************************/
          /* These macros drop in some in-line */
          /* assembly to define the IRQ table. */
          /************************************/
   set_vector(2000h,software_timer);
   set_vector(2002h,analog_conversion_done);
   set_vector(2008h,hsi_event);
   set_vector(200ah,line_sync_interrupt);                                 1760

                        /*****************************/
   initialize_globals();    /* INITIALIZE VARIABLES     */
   initialize_hardware();   /* INITIALIZE HARDWARE      */
                            /*****************************/

    /*************************************************************************/
    /** MAIN LOOP -- All the menu processing and display control is performed **/
    /** here.  The boost control functions are interrupt driven.            **/
    /*************************************************************************/                                        1770
   while(1)
   {
      switch(menu_mode)
      {
         case 1: /*** Select CONTROL MODE ***/
                 select_control_mode();          /* Display/select control mode on LCD */
                 break;
         case 2: /*** Select CONTROL DIRECTION ***/
                 select_control_direction();     /* Dsiplay/select control dir on LCD */
                 break;                                                     1780
         case 3: /*** Select SAFETY MODE ***/
                 select_safety_mode();           /* Dsiplay/select safety mode on LCD */
                 break;
         case 4: /*** Select spare MENU ***/
                 spare_select_menu();            /* Dsiplay/select spare menu on LCD  */
                 break;
         case 5: /*** Converter Running ***/
                 strcpy(display,"RUN ");
                 lcd_menu_running();             /* Display RUN info on the LCD    */
                 break;                                                     1790
         case 6: /*** Converter Error Halt ***/
                 strcpy(display,"HALT");
                 lcd_menu_halt();                /* Display HALT info on the LCD   */
                 break;
         case 7: /*** About to start menu ***/
                 lcd_menu_start();
                 break;
         case 8: /*** Select spare MENU2 ***/
                 spare_select_menu2();           /* Dsiplay/select spare menu on LCD  */
                 break;                                                     1800
         case 9: /*** Select spare MENU3 ***/
                 spare_select_menu3();           /* Dsiplay/select spare menu on LCD  */
```

```
                break;
        case 10: /*** Select spare MENU4 ***/
                 spare_select_menu4();              /* Dsiplay/select spare menu on LCD  */
                 break;
        case 11: /*** Select spare MENU4 ***/
                 spare_select_menu5();              /* Dsiplay/select spare menu on LCD  */
                 break;
        default: /*** Converter Stopped ***/                                              1810
                 strcpy(display,"STOP");
                 lcd_menu_stop();                  /* Display/select STOP info on the LCD */
                 break;
      }
      write_lcd_line(1);          /* Write line #1 on the LCD    */
      write_lcd_line(2);          /* Write line #2 on the LCD    */
      write_hp_display();         /* Write 4 chars to HP display */
      wait_fun(LCD_DELAY);
      SET_leds(leds);             /* Update the 4 LED's          */
                                                                                        1820
        /*** Check line-cycle count for loss of Input Voltage ***/
        /*** Only check when !INVERTER, !OFF, or !STARTUP_1   ***/
      if((ctrl_mode!=INVERTER_MODE) && (ctrl_mode!=DEBUG_CONST_K))
      {
         if((mode!=BOOST_STARTUP_1) && (mode!=BUCK_STARTUP_1) && (mode!=OFF))
         {
            if((line_cycs > 8) || (line_cycs < 5))
            {
               err_num=9;
               halt_converter();                                                        1830
            }
         }
      }
      if(reset_ints==TRUE)
      {
         reset_ints=FALSE;
         int_pending = 0;            /* Clear any pending interrupts          */
         int_mask = 0x11;            /* Unmask only the SWT_MASK and HSI0_MASK */
         ei();                       /* Enable interrupts                     */
         wait_fun(30000);            /* Force a small delay then continue     */       1840
         button_id=0;
      }
   }
}  /* End main */
```

## "C" Header File - BID_FL5.H

```
/*********************************************************************/
/* FILE: BID_FL5.H                                 REVISION: 5.0     */
/* BY: Deron Jackson                               DATE: 8/1/97      */
/*                                                                   */
/* This is a HEADER FILE for BID_FL5.C                               */
/*********************************************************************/


   /*************************/                                                        10
   /** Constant Definitions **/
   /*************************/
#define DEBUG        1       /* Set this to 1 for DEBUG messages.       */
#define FAULT_CHECKS 1       /* Set this to 1 to enable DC/DC faults.   */
#define TRUE         1
#define FALSE        0

#define uchar unsigned char
#define uint unsigned int
                                                                                    20

/*****************************************/
/* These are some convenient macros for    */
/* setting 80c196 hardware ports.  They    */
/* ensure that WSR is set correctly.       */
/*****************************************/
#define HSO_set(bit) \
             wsr=15; \
             hold_ios0 |= (0x01 << (bit)); \
             ios0 = hold_ios0; \                                                    30
             wsr=0
#define HSO_clr(bit) \
             wsr=15; \
             hold_ios0 &= ~(0x01 << (bit)); \
             ios0 = hold_ios0; \
             wsr=0
```

```
#define P2_setb(bit) \
            hold_iop2 |= (0x01 << (bit)); \
            ioport2 = hold_iop2
#define P2_clrb(bit) \
            hold_iop2 &= ~(0x01 << (bit)); \
            ioport2 = hold_iop2
#define REG_U2_setb(bit) \
            reg_U2 |= (0x01 << (bit)); \
            ioport1=(unsigned char) reg_U2; \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define REG_U2_clrb(bit) \
            reg_U2 &= ~(0x01 << (bit)); \
            ioport1=(unsigned char) reg_U2; \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define SET_reg_U2(val) \
            ioport1=(unsigned char) (val); \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define REG_U42_setb(bit) \
            reg_U42 |= (0x01 << (bit)); \
            ioport1=(unsigned char) reg_U42; \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define REG_U42_clrb(bit) \
            reg_U42 &= ~(0x01 << (bit)); \
            ioport1=(unsigned char) reg_U42; \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define SET_reg_U42(val) \
            ioport1=(unsigned char) (val); \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define SET_leds(val) \
            ioport1=(unsigned char) (val); \
            hold_iop2 |= 0x80; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x80; \
            ioport2 = hold_iop2
#define IRQ_ENTRY \
            hold_wsr=wsr; \
            wsr=0; \
            hold_ioport1=ioport1
#define IRQ_EXIT \
            ioport1=hold_ioport1; \
            wsr=hold_wsr


/*****************************************/
/* The following constant definitions are  */
/* hardware specific.  The specific values */
/* must be calibrated for each board set.  */
/* (See the MATHCAD file BID_CAL.MCD)      */
/*****************************************/
#define RSERIES_FP          15.90   /* Estimated value for Rseries          */
#define H1_FP             5.64197e2  /* Boost current-loop gain H1           */
#define H2_FP            -5.21883e2  /* Boost current-loop gain H2           */
#define HP_FP             2.22300e4  /* Boost current-loop power FF gain HP  */

#define VBST_TO_FP        1.46201e-1 /* Conversion ratio for Vbst Int-to-Float */
#define VAC_TO_FP         2.91730e-1 /* Conversion ratio for Vac Int-to-Float  */
#define IAC_TO_FP         2.13560e-2 /* Conversion ratio for Iac Int-to-Float  */
#define DC_V1_TO_FP       3.91007e-1 /* Conversion ratio for V1 Int-to-Float   */
#define DC_I1_TO_FP       9.30596e-3 /* Conversion ratio for I1 Int-to-Float   */
#define DC_V2_TO_FP       3.94405e-1 /* Conversion ratio for V2 Int-to-Float   */
#define DC_I2_TO_FP       1.82593e-2 /* Conversion ratio for I2 Int-to-Float   */
```

```
#define VBST_HIGH_FP       390.0    /* Output voltage upper limit.              */
#define IBST_HIGH_FP         4.0    /* Output current upper limit.              */
#define IAC_HIGH_FP          8.0    /* Input current upper limit.               */
#define VAC_HIGH_FP        150.0    /* Input voltage upper limit.               */
#define VAC_LOW_FP          90.0    /* Boost shuts down for low voltages.       */
#define VBST_OFFSET_FP     250.0    /* This is an offset value added to the VBST*/
#define CUR_MIN_FP           0.0    /* This is the MENU current minimum.        */
#define CUR_STEP_FP          0.1    /* This is the MENU step size.              */
#define CUR_MAX_FP           3.0    /* This is the MENU current maximum.        */
#define CK_MIN_FP            0.0    /* This is the MENU K minimum.              */
#define CK_STEP_FP          50.0    /* This is the MENU K step size.            */
#define CK_MAX_FP         1000.0    /* This is the MENU K maximum.              */

#define BUCK_GAIN_FP     2.47000e3  /* Buck current loop gain                   */
#define KMAX_CONST_FP    1.11150e5  /* This is usad to calculate the MAX i_cmd  */

#define MAX_ICMD_FP       4095.0    /* This is the absolute MAX value of i_cmd  */
#define DELTA_K_CLIP_FP    100.0    /* This is multiplied by (i_set-i_set2)     */
#define SS_KMAX_FP          75.0    /* This is MAX k during a boost soft-start  */

#define DC_I1_OFST            15    /* This is an offset for DC/DC I1           */
#define DC_I2_OFST             0    /* This is an offset for DC/DC I2           */

#define MENU_ID         0x40  /* MENU button ID          */
#define LEFT_ID         0x48  /* LEFT ARROW button ID    */
#define RIGHT_ID        0x50  /* RIGHT ARROW button ID   */
#define START_ID        0x58  /* START-STOP button ID    */

#define CURRENT_CONTROL   0  /* Controller mode setting */
#define CURRENT_STEP      1  /* Controller mode setting */
#define CURRENT_RAMP      2  /* Controller mode setting */
#define INVERTER_MODE     3  /* Controller mode setting */
#define DEBUG_CONST_K     4  /* Controller mode setting */

#define REVERSE           0  /* Controller direction setting */
#define FORWARD           1  /* Controller direction setting */

#define CONVERTER_STOPPED     0  /* Display mode setting */
#define SELECT_CONTROL_MODE   1  /* Display mode setting */
#define SELECT_CONTROL_DIR    2  /* Display mode setting */
#define SELECT_SAFETY_MODE    3  /* Display mode setting */
#define SPARE_SELECT_MENU     4  /* Display mode setting */
#define CONVERTER_RUNNING     5  /* Display mode setting */
#define HALT_ERROR_MENU       6  /* Display mode setting */
#define PREPARE_TO_START      7  /* Display mode setting */
#define SPARE_SELECT_MENU2    8  /* Display mode setting */
#define SPARE_SELECT_MENU3    9  /* Display mode setting */
#define SPARE_SELECT_MENU4   10  /* Display mode setting */
#define SPARE_SELECT_MENU5   11  /* Display mode setting */

#define ALL                   0  /* Safety mode: ALL                          */
#define SOME                  1  /* Safety mode: Just COMM, Vbst HIGH, Line Cycs */
#define NONE                  2  /* Safety mode: Just Vbst HIGH, Line Cycs    */

#define OFF                   0  /* mode: OFF                                 */
#define BOOST_STARTUP_1       1  /* mode: Wait cap pre-charge, close INRSH,   */
                                 /*       enable PWM, enable DC/DC Board 1.   */
#define BOOST_STARTUP_2       2  /* mode: Open-loop Ramp                      */
#define BOOST_SOFT_START      3  /* mode: BOOST soft-start                    */
#define BOOST_RUNNING         4  /* mode: BOOST is running                    */

#define BUCK_STARTUP_1       11  /* mode: Wait then close RLY1 & RLY2         */
                                 /*       and enable DC/DC Board 2.           */
#define BUCK_STARTUP_2       12  /* mode: Wait for DC stable -> Enable PWM    */
#define BUCK_RUNNING         13  /* mode: BUCK is running                     */

#define INVERTER_STARTUP_1   21  /* mode: Enable DC/DC Board 2                */
#define INVERTER_STARTUP_2   22  /* mode: Wait for DC stable -> close INRUSH  */
#define INVERTER_RUNNING     23  /* mode: INVERTER is running                 */

#define LCD_DELAY          50  /* This is a delay time between LCD updates  */
#define BUTTON_DB           2  /* This is a delay for button debouncing     */
#define PRE_CHARGE_TIME    15  /* This sets the time for cap pre-charge.    */
                               /* The value is in 15ths of a second.        */
                               /* (NOTE: This number must be > 5.)          */
#define DC_DC_DELAY        15  /* This sets the time delay after DC-DC      */
                               /* turn-on.  The value is in 15ths of a sec. */
#define BUCK_DC_DELAY       3  /* This sets the time delay after DC-DC      */
                               /* turn-on.  The value is in 15ths of a sec. */
#define SOFT_START_DELAY   50  /* This is the time delay for the soft-start
*/                               /* turn-on.  The value is in 15ths of a sec. */
                               /* 76 is approximately 5 seconds.            */
#define LED_P           2      /* Byte to turn on LED's */
#define LED_PC          6      /* Byte to turn on LED's */
#define LED_PD         10      /* Byte to turn on LED's */
#define LED_PF          3      /* Byte to turn on LED's */
```

```
/*****************************/                                          210
/** Variable Declarations **/
/*****************************/

        /*** DISPLAY CONTROL VARIABLES ***/
near unsigned char reg_U2;          /* This is the value of register U2   */
near unsigned char reg_U42;         /* This is the value of register U42  */
near unsigned char hold_iop2;       /* This hold the value of ioport2     */
near unsigned char hold_ios0;       /* This hold the value of ios0        */
near unsigned char leds;            /* This holds the LED status          */
near unsigned char debug_lev;       /* This holds a setting used for debug*/   220
near unsigned char safety_mode;     /* This holds the safety menu setting */
near unsigned char ramp_tracking;   /* This holds the boost ctrl mode     */
near unsigned char buck_loop;       /* This holds the buck ctrl mode      */
near unsigned char update_asap;     /* Flag for loop update delay         */
near float pin_filtered;            /* Makes a Pin running avg. filter    */
near float pdc_filtered;            /* Makes a Pdc running avg. filter    */
near char display[4];               /* Holds the 4 digit display          */
char lcd_line1[256];                /* String LCD line1 text              */
char lcd_line2[256];                /* String LCD line2 text              */
                                                                           230
        /*** VOLTAGE-LOOP CONTROL VARIABLES ***/
near unsigned int a_d_value;        /* This holds the 10 bit A/D value    */
near unsigned int int_ad_lo;        /* This holds the lo A/D value        */
near unsigned int int_ad_hi;        /* This holds the hi A/D value        */
near unsigned int int_icmd;         /* The current-loop command value (k) */
near unsigned int int_fbcmd;        /* The full-bridge command value      */
near float fp_kmax;                 /* This holds the max f-point i_cmd   */
near float fp_icmd;                 /* The current-loop command k in fp   */
near float fp_icmd_last;            /* The last value --> i_cmd[n-1]      */
near float fp_error_last;           /* The value of c[n-1]-x[n-1]         */   240
near float fp_delta_k;              /* The change in k                    */
near float fp_delta_k_clip;         /* Clipping value for delta_k         */
near float fp_vbst;                 /* The output voltage VBST.           */
near float fp_ibst;                 /* The output current                 */
near float fp_vac;                  /* The peak input voltage             */
near float fp_iac;                  /* The peak input current             */
near float fp_power;                /* The value of vbst * ibst           */
near float fp_power_last;           /* Last value of vbst * ibst          */
near float fp_x;                    /* The value of vbst * vbst           */
near float fp_x_last;               /* Last value of vbst * vbst          */   250
near float fp_dc_v2;                /* This is V2 on Board 2 of the DC-DC */
near float fp_dc_i2;                /* This is I2 on Board 2 of the DC-DC */
near float fp_cmd_c;                /* Holds the command (c) in fp        */
near float fp_kmax_const;           /* Holds the value of the kmax const  */
near float fp_kmax_ss;              /* Holds the value of the kmax const  */
near float fp_rseries;              /* This holds the value of rseries    */

        /*** CURRENT-LOOP CONTROL VARIABLES ***/
near float fp_iref;                 /* The output current reference       */
                                                                           260
        /*** MISC TEMP VARIABLES ***/
near float fp_temp1;                /* Holds a temp floating point value  */

    /*** KEY TEMP VARIABLES (ONLY for IRQ ROUTINES) ***/
near unsigned int int_key1;         /* Temp variable for IRQ routines     */
near float fp_key1;                 /* Temp variable for IRQ routines     */

        /*** GENERAL OPERATING VARIABLES ***/
near unsigned char swt_count;       /* IRQ counter for the software timer */
near unsigned char swt_delay;       /* IRQ counter for the software timer */   270
near unsigned char strobe;          /* IRQ counter for strobe indicator   */
near unsigned char cyc_count;       /* Hold line cycle count in progress  */
near unsigned char line_cycs;       /* Counts the # of cycles between     */
                                    /* software IRQ's                     */
near unsigned int step_dly;         /* This keeps count of the delay used */
                                    /* to implement the step response.    */
near float fp_iset;                 /* Current setting for control loop   */
near float fp_iset2;                /* Current setting for control loop   */
near float fp_const_k;              /* K setting for debug mode.          */
                                                                           280
near unsigned char mode;            /* This indicates the converter mode. */
                                    /* mode = 0 --> OFF                   */
                                    /* mode = 1 --> Cap Pre-charge Delay  */
                                    /* mode = 2 --> DC-DC Startup Delay   */
                                    /* mode = 3 --> Open-loop soft-start  */
                                    /* mode = 4 --> Closed-loop soft-start*/
                                    /* mode = 5 --> Closed-loop           */
near unsigned char ctrl_dir;        /* Indicates the control direction    */
                                    /* mode = 0 --> Reverse BUCK discharge*/
                                    /* mode = 1 --> Forward BOOST charging*/   290
near unsigned char menu_mode;       /* This indicates the menu on the LCD */
                                    /* mode = 0 --> Power Up - Set Level  */
                                    /* mode = 1 --> Select Control Mode   */
```

```
                                /* mode = 2 --> Select Control Port   */
                                /* mode = 5 --> Running - Charging    */
                                /* mode = 6 --> Fault condition       */
near unsigned char ctrl_mode;   /* This indicates the control strategy*/
                                /* mode = 0 --> Constant Boost Voltage*/
                                /* mode = 1 --> Constant Boost Current*/
                                /* mode = 2 --> Voltage Boost Step    */
                                /* mode = 3 --> Current Boost Step    */
                                /* mode = 4 --> Constant DC_DC Voltage*/
near unsigned char err_num;     /* This indicates the type of error   */
                                /* err_num = 0 --> No error           */
                                /* err_num = 1 --> Vac too high       */
                                /* err_num = 2 --> Vac too low        */
                                /* err_num = 3 --> Iac too high       */
                                /* err_num = 4 --> Vbst too high      */
                                /* err_num = 5 --> Ibst too high      */
                                /* err_num = 6 --> DC-DC #1 Fault     */
                                /* err_num = 7 --> DC-DC #2 Fault     */
                                /* err_num = 8 --> DC-DC #1&2 Fault   */
                                /* err_num = 9 --> Line-Cycle error   */
                                /* err_num =10 --> I-Set exceeded     */
                                /* err_num =11 --> Comm. Disconnected */
near unsigned char button_id;   /* Stores ID for the start/stop bttns */
near unsigned char button_db;   /* Stores counter to debounce bttns   */
near unsigned char reset_ints;  /* Reset the INT_MASK when high       */
```

## F.3.2. Adaptive Motor-Speed Control

The source code MOTOR1 below was developed for use with the 1.5-kW unidirectional prototype. This software implements the PP voltage-control algorithm described in Section 5.2, as well as the LM-RLS adaptive motor-speed control algorithm described in Section 6.4. It was used to generate the data for Figures 5.7 – 5.9 and 6.14 in the text. Menu options allow the user to select between voltage control, fixed motor-speed control, and adaptive motor-seed control.

### "C" Source Code - MOTOR1.C

```
/*****************************************************************/
/* FILE: MOTOR1.C                              REVISION: 1.0     */
/* BY: Deron Jackson                           DATE: 8/19/97     */
/*                                                               */
/* HARDWARE:  INTERLEAVED BOOST CONVERTER                        */
/*            HALF-BRIDGE DC/DC                                  */
/*                                                               */
/* This version does:                                            */
/*      - Voltage control into a R-load (constant, step, ramp)   */
/*      - FIXED Motor Speed Control                              */
/*      - ADAPTIVE LM-RLS Motor Speed Control                    */
/*                                                               */
/* NOTES:                                                        */
/*    This code is configured for the HITECH 80196 C-Compiler.   */
/*    The code must be linked together with the DKJSTART.OBJ and */
/*    DKJROM.OBJ files.                                          */
/*****************************************************************/

#include<80C196kc.h>
#include<intrpt.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<math.h>
#include<float.h>

#include "motor1.h"

  /*****************************/
```

```
  /** Setup the IRQ routines **/
  /*****************************/
interrupt void software_timer(void);
interrupt void line_sync_interrupt(void);
interrupt void hsi_event(void);
interrupt void analog_conversion_done(void);

  /*************************/
  /** Function Prototypes **/
  /*************************/
void wait_fun(uint cntsize);
void MDAC_write(void);
void write_lcd_line(uchar line_num);
void write_hp_display(void);
void halt_converter(void);
int check_com(void);
void boost_voltage_control(void);
void step_ramp_calcs(void);
void speed_control(void);
int int_part(float fp_value);
int dec_part(float fp_value);
int round_fp(float fp_value);
void make_bar_graph(float level,float min_lev,float max_lev);
void start_converter(void);
void select_control_mode(void);
void select_safety_mode(void);
void spare_select_menu(void);
void lcd_menu_running(void);
void lcd_menu_halt(void);
void lcd_menu_stop(void);
void lcd_menu_start(void);
void initialize_globals(void);
void initialize_hardware(void);


/*********************************************************************/
/** WAIT_FUN()  This is a simple wait loop.                          **/
/*********************************************************************/
void wait_fun(uint cntsize)
{
   uint t1=0;
   uint t2=0;
   uint t3=0;

   while(t1 < cntsize)
   {
      t1++;
      while(t2 < cntsize)
      {
         t2++;
         while(t3 < cntsize)
            t3++;
      }
   }
      /*** Check speed-cyc-count for a 1 sec interval ***/
      /*** then go do the speed control calcs.        ***/
   if((ctrl_mode >= SPEED_CONTROL) && (speed_cyc_count >= 120))
   {
      speed_cyc_count=0;
      REG_U2_setb(3);
      speed_control();
      REG_U2_clrb(3);
   }
}


/*********************************************************************/
/** MDAC_WRITE()  This procedure writes a 10 bit word to the MAX501 **/
/** multiplaying DAC.                                               **/
/*********************************************************************/
void MDAC_write(void)
{
   ioport1 = (uchar) (int_icmd >> 8);
   HSO_clr(1);              /* Set /CSMSB LOW  (select the MSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the MSB) */
   HSO_set(1);              /* Set /CSMSB HIGH                    */
                            /* Set IOPORT1 with the LSB 8 bits    */
   ioport1 = (uchar) (int_icmd & 0x00ff);
   HSO_clr(3);              /* Set /CSLSB LOW  (select the LSB)   */
   HSO_clr(2);              /* Set /WR    LOW                     */
   HSO_set(2);              /* Set /WR    HIGH (latching the LSB) */
   HSO_set(3);              /* Set /CSLSB HIGH                    */

   HSO_clr(0);              /* Set /LDAC  LOW  (transfer 12 bits) */
```

```
   HSO_set(0);              /* Set /LDAC  HIGH                    */
}

/*********************************************************************/
/** WRITE_LCD_LINE()  This procedure writes a 40 character line to  **/
/** the LCD display.                                                **/
/*********************************************************************/
void write_lcd_line(uchar line_num)
{
   uchar pos;

   di();
   REG_U42_clrb(6);         /* Set L-6 = 0 ---- LCD Inst. mode */
   if(line_num==1)
      ioport1 = 0x80;       /* Set Address for line 1          */
   else
      ioport1 = 0xC0;       /* Set Address for line 2          */
   P2_setb(5);              /* Strobe P2-5 to latch LCD data   */
   P2_clrb(5);
   REG_U42_setb(6);         /* Set L-6 = 1 ---- LCD Data mode  */
   ei();
   wait_fun(LCD_DELAY);

   for(pos=0;pos<40;pos++)
   {
      di();
      if(line_num==1)
         ioport1 = lcd_line1[pos];
      else
         ioport1 = lcd_line2[pos];
      P2_setb(5);           /* Strobe P2-5 to latch LCD data   */
      P2_clrb(5);
      ei();
      wait_fun(LCD_DELAY);
   }
}


/*********************************************************************/
/** WRITE_HP_DISPLAY()  This procedure writes 4 characters to the HP**/
/** display.                                                        **/
/*********************************************************************/
void write_hp_display(void)
{
   uchar hp_digit, hp_char, pulse;

   pulse=(strobe >> 2);
   for(hp_digit=0;hp_digit<4;hp_digit++)
   {
            /*************************************************/
            /******* NOTE:  BITS 6 and 4 are reversed ******/
            /******* ----   to correct a hardware bug. *****/
            /*************************************************/
      hp_char = (display[hp_digit] & 0x2F)
                + ((display[hp_digit] & 0x10) << 2)
                     +((display[hp_digit] & 0x40) >> 2);
      di();
      switch(hp_digit)
      {
         case 0: set_bit(reg_U42,2);set_bit(reg_U42,3);      /* Select the 1st DIGIT */
                 if(pulse==0) hp_char=42; break;
         case 1: set_bit(reg_U42,2);clr_bit(reg_U42,3);      /* Select the 2nd DIGIT */
                 if(pulse==1 || pulse==5) hp_char=42; break;
         case 2: clr_bit(reg_U42,2);set_bit(reg_U42,3);      /* Select the 3rd DIGIT */
                 if(pulse==2 || pulse==4) hp_char=42; break;
         case 3: clr_bit(reg_U42,2);clr_bit(reg_U42,3);      /* Select the 4th DIGIT */
                 if(pulse==3) hp_char=42; break;
         default: break;
      }
      ei();
      wait_fun(LCD_DELAY);
      SET_reg_U42(reg_U42);
      di();
      ioport1 = hp_char;
      HSO_clr(5);                  /* Strobe the /WR pin on the DISPLAY */
      HSO_set(5);
      ei();
      wait_fun(LCD_DELAY);
   }
}


/*********************************************************************/
/** HALT_CONVERTER()  This procedure is called in order to shutdown **/
/** the converter.                                                  **/
```

```
/*********************************************************************/
void halt_converter(void)
{
   di();                    /* DISABLE all the interrupts    */
   int_pending = 0;         /* Clear any pending interrupts  */
   int_mask = 0;            /* Mask all IRQ's unless we hit  */
                            /* an ei() on the way to main.   */
   reset_ints=TRUE;                                                    210

   int_icmd = 0;            /* Zero the current command */
   MDAC_write();

   reg_U42=0xF1;        /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM  */
                        /* Set DS_A0      = 0                      */
                        /* Set DS_A1      = 0                      */
                        /* Set L-4        = 1                      */
                        /* Set L-5 (B_MDE)= 1                      */   220
                        /* Set L-6 (RS)   = 1     LCD Data Mode    */
                        /* Set L-7 (RLY2) = 1     OPEN RELAY_2     */
   SET_reg_U42(reg_U42);
   reg_U2=0x52;         /* Set B1_ENABLE    = 0   DISABLE B1     */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS   */
                        /* Set B2_ENABLE    = 0   DISABLE B2     */
                        /* Set FB_LDAC      = 0                    */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS  */
                        /* Set B1_RLY_RESET = 0                    */
                        /* Set B2_RLY_SET   = 1   SET B2 RELAYS  */   230
                        /* Set B2_RLY_RESET = 0                    */
   SET_reg_U2(reg_U2);

   if(err_num!=0)
   {
      leds=LED_PF;                  /* Turn ON the POWER & FAULT LEDs only */
      menu_mode=HALT_ERROR_MENU;    /* Set the ERROR MENU */
   }
   else
   {                                                                   240
      leds=LED_P;                   /* Turn ON the POWER LED only */
      menu_mode=CONVERTER_STOPPED; /* Set MAIN MENU */
   }

   mode=OFF;                    /* Set the converter mode to OFF */
   button_id=0;
   button_db=0;
}

                                                                       250
/*********************************************************************/
/** HSI_EVENT()  This procedure executes each time an HSI.0 event   **/
/** occurs.  This may be a button press or a fault in the DC-DC     **/
/** converter.                                                      **/
/*********************************************************************/
interrupt void hsi_event(void)
{
   uchar hold_wsr, hold_ioport1;
   uchar hsi_temp;

   IRQ_ENTRY;                                                          260

   hsi_temp = ioport2 & 0x58;     /* Read the status of P2-3,P2-4,P2-6 */

   if(hsi_temp)                 /* If any button was pressed reset the LED's */
   {                            /* This helps to fix an apparent glitch.     */
      SET_leds(leds);           /* Update the 4 LED's */
   }

   if(hsi_temp==MENU_ID || hsi_temp==LEFT_ID || hsi_temp==RIGHT_ID || hsi_temp==START_ID)   270
   {
      if(button_db>BUTTON_DB)       /* Debounce the button press by N/20th's of a sec */
         button_id=hsi_temp;
   }
   IRQ_EXIT;
}


/*********************************************************************/
/** CHECK_COM()  This procedure checks the com line to the DC/DC    **/   280
/** converter.  If low it means the puck is not inserted.           **/
/*********************************************************************/
int check_com(void)
{
   if(check_bit(ioport2,1)==0)          /* Read the COM_CHK line P2.1 */
   {
      wait_fun(25);
      if(check_bit(ioport2,1)==0)       /* If failed, double check    */
```

```
      {
         wait_fun(25);                                                    290
         if(check_bit(ioport2,1)==0)      /* If failed, triple check    */
            return(1);                    /* Finally return 1 if failed */
      }
   }
   return(0);                             /* Otherwise return a 0 */
}


/*********************************************************************/
/** SOFTWARE_TIMER()  This procedure is set to execute 20 times a   **/   300
/** second.                                                         **/
/*********************************************************************/
interrupt void software_timer(void)
{
   uchar hold_wsr, hold_ioport1;

   IRQ_ENTRY;
   line_cycs=cyc_count;             /* Record the # of line cycles in */
                                    /* the last 1/15.3th of a second. */
   cyc_count=0;                     /* Reset the counter.             */   310
   swt_count++;                     /* Increment the swt_counter      */
   strobe++;                        /* Increment strobe counter       */
   if((strobe >> 2) > 5)
      strobe=0;

   if(button_db < 255)              /* Increment the button debouncer */
      button_db++;

   if(swt_count > swt_delay)
   {                                                                    320
      switch(mode)
      {
         case BOOST_STARTUP_1:
                REG_U42_clrb(0);    /* Clear RELAY_TTL -- CLOSE RELAY 1 */
                mode=BOOST_STARTUP_2;
                swt_count=0;                    /* Reset the swt_count */
                swt_delay=MINI_DELAY;           /* Set a new delay time*/
                break;
         case BOOST_STARTUP_2:
                REG_U42_setb(1);    /* Set  PWM_EN_TTL -- ENABLE PWM    */   330
                REG_U2_clrb(1);     /* Clear FAULT_CLR -- READY FAULTS  */
                REG_U2_setb(0);     /* Set   B1_ENABLE -- ENABLE B1     */
                mode=BOOST_STARTUP_3;
                swt_count=0;                    /* Reset the swt_count */
                swt_delay=DC_DC_DELAY;          /* Set a new delay time*/
                break;
         case BOOST_STARTUP_3:
                mode=BOOST_OL_SOFT_START;    /* Jump to open_loop SS after some delay */
                break;
         default: break;                                                340
      }
   }

   if(ctrl_mode==DEBUG_CONST_K)
   {
      ad_command =  8;              /* Begin a Channel 0 A/D conversion */
                                    /* ACH0 is the output voltage VBST  */
   }
   IRQ_EXIT;
}                                                                       350


/*********************************************************************/
/** LINE_SYNC_INTERRUPT()  This procedure is triggered at the 120 Hz **/
/** frequency generated by the line sync hardware.  The line sync   **/
/** triggers an EXTINT on P2.2.                                     **/
/*********************************************************************/
interrupt void line_sync_interrupt(void)
{
   uchar hold_wsr, hold_ioport1;                                        360

   IRQ_ENTRY;
   cyc_count++;              /* Increment the line-cycle counter */
   speed_cyc_count++;        /* Increment the counter for the speed control loop */

   /*** Read the COM_CHK line to check that the puck is inserted ***/
   if((mode!=OFF) && (safety_mode!=NONE))   /* Skip while the DC/DC starts, */
   {                                        /* or safety=NONE.              */
      if(check_com())
      {                                                                 370
         err_num=11;
         halt_converter();
         IRQ_EXIT;
         return;
```

```
      }
   }

   if(update_asap==0)
      MDAC_write();              /* Write delayd icmd to MDAC if update_asap==0 */

   ad_command =  8;              /* Begin a Channel 0 A/D conversion */
                                 /* ACH0 is the output voltage VBST  */

   IRQ_EXIT;
}


/*************************************************************/
/** BOOST_VOLTAGE_CONTROL()  This procedure contains all the   **/
/** processing for boost mode voltage control.                 **/
/*************************************************************/
void boost_voltage_control(void)
{
                /***********************************/
                /* Here is the soft-start control */
                /***********************************/
   if(mode==BOOST_OL_SOFT_START)
   {
      if(fp_vbst < ENDOFSOFTSTART_FP)
      {
         fp_icmd += ICMD_INC_FP;
         int_icmd = (uint) fp_icmd;
         return;
      }
      else
      {
          fp_vref = fp_vset;
          mode=BOOST_RUNNING;
          step_dly=0;
          tstep_dly=0;
          speed_cyc_count=0;
      }
   }
                /*************************************/
                /* Here are the control calculations */
                /*************************************/
   fp_cmd_c = fp_vref * fp_vref;                                  /* Calc command c[n]*/

   fp_delta_k = H1_FP * (fp_cmd_c - fp_x);                        /* Calc the H1 term */
   fp_delta_k = fp_delta_k + H2_FP * (fp_cmd_c - fp_x_last);      /* Calc the H2 term */

   fp_delta_k = fp_delta_k + HP_FP * (fp_power - fp_power_last); /* Power FF term    */
   fp_delta_k = fp_delta_k / (fp_vac * fp_vac);                   /* Divide by vac^2  */

   fp_icmd = fp_icmd_last + fp_delta_k;

   fp_kmax = fp_kmax_const / fp_vac;

   if(fp_icmd > fp_kmax)                        /* In case of too high a command.   */
      fp_icmd = fp_kmax;                        /* Cap the int_icmd at kmax         */
   if(fp_icmd > MAX_ICMD_FP)                    /* Is icmd valid?                   */
      fp_icmd = MAX_ICMD_FP;
   if(fp_icmd < 0)                              /* Is icmd negative?                */
      fp_icmd = 0;                              /* Then send a icmd = 0.            */

   int_icmd = (uint) fp_icmd;
}


/*************************************************************/
/** STEP_RAMP_CALCS()  This procedure contains non-time-     **/
/** critical control calcs.                                  **/
/*************************************************************/
void step_ramp_calcs(void)
{
             /***********************************/
             /* Here's the STEP/RAMP control   */
             /***********************************/
   step_dly++;
   if (ctrl_mode==VOLTAGE_STEP)
   {
      if(step_dly == 300)             /* Switch levels every 2.5 secs */
         fp_vref = fp_vset2;
      if(step_dly == 600)
         fp_vref = fp_vset;
      if(step_dly >= 900)
         step_dly = 299;
   }
   else if(ctrl_mode==VOLTAGE_RAMP)
   {
      if(fp_vset2 >= fp_vset)
```

```
      {
         fp_vref = fp_vref + ((fp_vset2 - fp_vset) / 300.0);
         if(fp_vref > fp_vset2)
         {
            fp_vref = fp_vset;
            step_dly = 0;
         }
      }
      if(fp_vset2 < fp_vset)
      {                                                                    470
         fp_vref = fp_vref - ((fp_vset - fp_vset2) / 300.0);
         if(fp_vref < fp_vset2)
         {
            fp_vref = fp_vset;
            step_dly = 0;
         }
      }
   }
   else if(ctrl_mode==VOLTAGE_CONTROL)
      fp_vref = fp_vset;                                                   480
}


/*********************************************************************/
/** ANALOG_CONVERSION_DONE()  This procedure is executed upon the   **/
/** completion of any A/D conversion.  All four A/D's are executed  **/
/** in sucession and then the voltage-loop control code is called.  **/
/*********************************************************************/
interrupt void analog_conversion_done(void)
{                                                                      490
   uchar hold_wsr, hold_ioport1, ad_channel;

   di();                          /* Disable IRQ's since A/D is high priority */
   IRQ_ENTRY;
   int_ad_hi = (uint) ad_result_hi;              /* Read the HI and LO parts */
   int_ad_lo = (uint) ad_result_lo;              /* of the 10 bit value.     */
   a_d_value = (int_ad_lo >> 6) + (int_ad_hi << 2);    /* Store 10-bit value. */
   ad_channel = ad_result_lo & 7;

   switch(ad_channel)                                                  500
   {
            /*************************************/
            /* A/D Channel 0  --->  OUTPUT VOLTAGE */
      case 0:   /*************************************/
         int_key1 = 1023 - a_d_value;      /* To measure VBST, we must first subtract */
                                           /* a_d_value from 1023.  This corrects for */
                                           /* a minor bug in the hardware.            */
         fp_vbst = ((float) int_key1 * VBST_TO_FP) + VBST_OFFSET_FP; /* Calc FP value */
         if (fp_vbst > VBST_HIGH_FP)             /* Check if VBST is over VBST_HIGH.  */
         {                                                             510
            err_num=4;
            halt_converter();
            IRQ_EXIT;
            return;
         }
         ad_command =  9;                          /* Begin a Channel 1 A/D conversion */
         break;
            /*************************************/
            /* A/D Channel 1  --->  INPUT VOLTAGE */
      case 1:   /*************************************/                  520
         fp_vac = (float) a_d_value * VAC_TO_FP;    /* Set VAC equal to the A/D value */

         if((mode!=BOOST_STARTUP_1) && (mode!=OFF) && (safety_mode==ALL))
         {
            if (fp_vac < VAC_LOW_FP)                    /* Check if VAC is too LOW */
            {
               err_num=2;
               halt_converter();
               IRQ_EXIT;
               return;                                                 530
            }
            else if (fp_vac > VAC_HIGH_FP)             /* Check if VAC is too HIGH */
            {
               err_num=1;
               halt_converter();
               IRQ_EXIT;
               return;
            }
         }
         ad_command = 10;                          /* Begin a Channel 2 A/D conversion */   540
         break;
                  /*************************************/
                  /* A/D Channel 2  --->  OUTPUT CURRENT */
      case 2:     /*************************************/
         fp_ibst = (float) a_d_value * IBST_TO_FP;    /* Set Ibst equal to the A/D value */
```

```
      if (fp_ibst > IBST_HIGH_FP)              /* Check if VBST is over VBST_HIGH.  */
      {
         err_num=5;                                                              550
         halt_converter();
         IRQ_EXIT;
         return;
      }
      ad_command = 12;                          /* Begin a Channel 4 A/D conversion */
      break;
               /*********************************/
               /* A/D Channel 4  --->  DC-DC V1 */
   case 4:     /*********************************/
      fp_key1 = (float) a_d_value * DC_V1_TO_FP;                               560
      if(fp_key1 < VBST_OFFSET_FP)              /* Read the DC_V1 voltage and use */
         fp_vbst = fp_key1;                     /* it when vbst < VBST_OFFSET.    */
            /*********************************************/
            /* Now that all REQUIRED A/D's have been read */
            /* Let's go see about some control.           */
            /*********************************************/


      fp_power = fp_vbst * fp_ibst;            /***********************************/
      fp_x = fp_vbst * fp_vbst;                /* Here are some needed pre-calcs */  570
                                               /***********************************/
      if(ctrl_mode==DEBUG_CONST_K)
      {
         int_icmd = (uint) fp_const_k;
         MDAC_write();
      }
      else
      {
         if(mode==BOOST_OL_SOFT_START || mode==BOOST_RUNNING)
            boost_voltage_control();           /* Calc icmd for boost mode control */  580
         else
            int_icmd = 0;                      /* Otherwise send a zero command    */
         if(update_asap)
            MDAC_write();                      /* Write the result to the MDAC     */
         if(mode==BOOST_RUNNING)
            step_ramp_calcs();                 /* Go do step/ramp stuff            */
      }
      fp_icmd_last = fp_icmd;                  /***********************************/
      fp_power_last = fp_power;                /* Here are some needed post-calcs */
      fp_x_last = fp_x;                        /***********************************/  590

      ad_command = 11;                         /* Begin a Channel 3 A/D conversion */
      break;
            /*************************************/
            /* A/D Channel 3  --->  INPUT CURRENT */
   case 3:  /*************************************/
      fp_iac = (float) a_d_value * IAC_TO_FP;     /* Set IAC equal to the A/D value */
         /** Check the IAC levels when in BOOST or BUCK modes.    **/
         /** Clear IAC in INVERTER MODE since its inactive.       **/
      if ((mode!=OFF) && (fp_iac > IAC_HIGH_FP) && (safety_mode==ALL))          600
      {
         err_num=3;
         halt_converter();
         IRQ_EXIT;
         return;
      }
      ad_command = 15;                         /* Begin a Channel 7 A/D conversion */
      break;
               /*****************************/
               /* A/D Channel 7  --->  Speed */                                 610
   case 7:     /*****************************/
      fp_key1 = (float) a_d_value * SPEED_TO_FP;
      /** LPF the speed measurements with and approx 1/5 sec filter **/
      fp_speed = (0.9592 * fp_speed) + (0.0408 * fp_key1);
      /** LPF the speed measurements with a 10 sec filter **/
      fp_speed_filt1 = (9.9916701e-1 * fp_speed_filt1) + (8.3299e-4 * fp_key1);
      /** LPF again with a 10 sec filter **/
      fp_speed_filt2 = (9.9916701e-1 * fp_speed_filt2) + (8.3299e-4 * fp_speed_filt1);
      break;
   default: break;                                                             620
  }
  IRQ_EXIT;
  ei();
} /* End of analog_conversion_done ISR.  */


/*************************************************************/
/** SPEED_CONTROL()  Perform speed control calcs.           **/
/*************************************************************/
void speed_control(void)                                                       630
{
```

```
if(ctrl_mode==SPEED_STEP_PP)
{
      /* Used FIXED gains for non-adaptive PP controller */
   fp_tn1 = N1_FP;
   fp_tn2 = N2_FP;
   fp_tn3 = N3_FP;
   fp_td1 = D1_FP;
   fp_td2 = D2_FP;
}
else
{
         /** Generate filtered voltage command **/
         /** Filter is set for 10 sec period   **/
   fp_pwr_filt1 = (9.0484e-1 * fp_pwr_filt1) + (9.5160e-2 * fp_pwr_cmd);
   fp_pwr_filt2 = (9.0484e-1 * fp_pwr_filt2) + (9.5160e-2 * fp_pwr_filt1);

   /*********************************************/
   /**** RLS parameter estimator calculations ****/
   /*********************************************/
   if(((fp_speed_filt1 - fp_speed_filt2) > 0.002) ||
         ((fp_speed_filt2 - fp_speed_filt1) > 0.002))
   {
         /* Generate filtered regressor vector */
      X[0] = fp_speed - 2*fp_speed_filt1 + fp_speed_filt2;
      X[1] = TAU_FP * (fp_speed_filt1 - fp_speed_filt2);
      X[2] = fp_speed_filt2 * TAU_SQ_FP;
         /* Calculate denominator */
      rls_den = X[0]*X[0]*P[0] + X[0]*X[1]*P[1] +
                  X[0]*X[2]*P[2] + X[1]*X[0]*P[3] +
                     X[1]*X[1]*P[4] + X[1]*X[2]*P[5] +
                        X[2]*X[0]*P[6] + X[2]*X[1]*P[7] +
                           X[2]*X[2]*P[8] + LAMBDA1;
         /* Calculate RLS gain vector */
      K[0] = (X[0]*P[0] + X[1]*P[3] + X[2]*P[6])/rls_den;
      K[1] = (X[0]*P[1] + X[1]*P[4] + X[2]*P[7])/rls_den;
      K[2] = (X[0]*P[2] + X[1]*P[5] + X[2]*P[8])/rls_den;
         /* Compute estimation error */
      rls_error = fp_pwr_filt2*TAU_SQ_FP -
                     X[0]*theta[0] - X[1]*theta[1] - X[2]*theta[2];
         /* Update parameter estimates */
      theta[0] = theta[0] + K[0] * rls_error;
      theta[1] = theta[1] + K[1] * rls_error;
      theta[2] = theta[2] + K[2] * rls_error;
         /* Update P matrix */
      P_new[0] = (P[0] - K[0]*X[0]*P[0] - K[0]*X[1]*P[1] - K[0]*X[2]*P[2])/LAMBDA1;
      P_new[1] = (P[1] - K[1]*X[0]*P[0] - K[1]*X[1]*P[1] - K[1]*X[2]*P[2])/LAMBDA1;
      P_new[2] = (P[2] - K[2]*X[0]*P[0] - K[2]*X[1]*P[1] - K[2]*X[2]*P[2])/LAMBDA1;
      P_new[3] = (P[3] - K[0]*X[0]*P[3] - K[0]*X[1]*P[4] - K[0]*X[2]*P[5])/LAMBDA1;
      P_new[4] = (P[4] - K[1]*X[0]*P[3] - K[1]*X[1]*P[4] - K[1]*X[2]*P[5])/LAMBDA1;
      P_new[5] = (P[5] - K[2]*X[0]*P[3] - K[2]*X[1]*P[4] - K[2]*X[2]*P[5])/LAMBDA1;
      P_new[6] = (P[6] - K[0]*X[0]*P[6] - K[0]*X[1]*P[7] - K[0]*X[2]*P[8])/LAMBDA1;
      P_new[7] = (P[7] - K[1]*X[0]*P[6] - K[1]*X[1]*P[7] - K[1]*X[2]*P[8])/LAMBDA1;
      P_new[8] = (P[8] - K[2]*X[0]*P[6] - K[2]*X[1]*P[7] - K[2]*X[2]*P[8])/LAMBDA1;
      P[0] = P_new[0];
      P[1] = P_new[1];
      P[2] = P_new[2];
      P[3] = P_new[3];
      P[4] = P_new[4];
      P[5] = P_new[5];
      P[6] = P_new[6];
      P[7] = P_new[7];
      P[8] = P_new[8];
   }
   if(tstep_dly < 90)
   {
      fp_tn1 = N1_FP;
      fp_tn2 = N2_FP;
      fp_tn3 = N3_FP;
      fp_td1 = D1_FP;
      fp_td2 = D2_FP;
   }
   else
   {
      fp_A = 1.0 / theta[2];                    /* Estimated CT system gain */
      fp_t1 = theta[1]*theta[1] - 4.0*theta[2]*theta[0];
      if(fp_t1 > 0)
         fp_t1 = sqrt(fp_t1);
      else
         fp_t1 = 0;

      fp_rc1 = (theta[1]+fp_t1)/(2*theta[2]);     /* These are the estimated */
      fp_rc2 = (theta[1]-fp_t1)/(2*theta[2]);     /* CT system poles         */

      if(fp_rc1 < 5.0)                    /* Clip the pole locations if they */
         fp_rc1 = 5.0;                    /* seem insane.                    */
      if(fp_rc2 < 5.0)
```

```
            fp_rc2 = 5.0;

         fp_r1 = exp(-1.0/fp_rc1);                  /* These are the equivalent */
         fp_r2 = exp(-1.0/fp_rc2);                  /* DT system poles          */

         fp_c = - fp_r1 - fp_r2;                    /* Map poles/gain into coeffs */
         fp_d = fp_r1 * fp_r2;                      /* and generate controller    */
         fp_b = fp_A * (1.0 + fp_c + fp_d);

               /* Calculate GAINS for adaptive PP controller */
         fp_td2 = fp_c - 1.0 - CL1_DENA_FP;
         fp_tn3 = -(fp_d*fp_td2 - CL1_DEND_FP)/fp_b;
         fp_tn2 = -(-fp_d - fp_d*fp_td2 + fp_c*fp_td2 - CL1_DENC_FP)/fp_b;
         fp_tn1 = -(fp_d - fp_c*fp_td2 - fp_c + fp_td2 - CL1_DENB_FP)/fp_b;
         fp_td1 = -fp_td2 - 1.0;
      }
   }
         /* Here's the PP control command */
   fp_pwr_cmd_last2 = fp_pwr_cmd_last;    /* Store two-back command */
   fp_pwr_cmd_last = fp_pwr_cmd;          /* Store last command */
   fp_pwr_cmd = - (fp_td1 * fp_pwr_cmd_last) - (fp_td2 * fp_pwr_cmd_last2)
                + (fp_tn1 * (fp_tref - fp_speed))
                   + (fp_tn2 * (fp_tref - fp_speed_last))
                      + (fp_tn3 * (fp_tref - fp_speed_last2));
      /* Convert the requested power into a voltage command */
   if(fp_pwr_cmd > PWR_MAX)
      fp_pwr_cmd = PWR_MAX;
   if(fp_pwr_cmd < PWR_MIN)
      fp_pwr_cmd = PWR_MIN;

   fp_v_cmd = ((fp_pwr_cmd + BIAS_VOLTAGE_FP) + 19.982)*2.60;

   if(fp_v_cmd > (VBST_HIGH_FP - 5.0))
      fp_v_cmd = (VBST_HIGH_FP - 5.0);

   fp_vref = fp_v_cmd;              /* Now go with it */

   fp_speed_last2 = fp_speed_last;              /* Store two-back value   */
   fp_speed_last = fp_speed;                    /* Store last bucket temp */

            /*****************************/
            /* Here's the STEP control   */
            /*****************************/
   tstep_dly++;

   if(tstep_dly == 90)              /* Switch levels every 90 secs */
      fp_tref = fp_tset2;
   if(tstep_dly == 180)
      fp_tref = fp_tset;
   if(tstep_dly >= 270)
      tstep_dly = 89;
}

/*****************************************************************/
/** INT_PART()  Return the rounded integer part of a float.     **/
/*****************************************************************/
int int_part(float fp_value)
{
   fp_value = fp_value + 0.05;
   return((int) fp_value);
}

/*****************************************************************/
/** DEC_PART()  Return the rounded 1st decimal place of a float.**/
/*****************************************************************/
int dec_part(float fp_value)
{
   fp_value = fp_value + 0.05;
   return((int) ((fp_value - (int) fp_value) * 10));
}

/*****************************************************************/
/** ROUND_FP()  Return the rounded integer value of a float.    **/
/*****************************************************************/
int round_fp(float fp_value)
{
   fp_value = fp_value + 0.5;
   return((int) fp_value);
}

/*****************************************************************/
/** MAKE_BAR_GRAPH()  This procedure constructs a neat little bar- **/
/** graph for the LCD display.                                  **/
/*****************************************************************/
```

```
void make_bar_graph(float level,float min_lev,float max_lev)
{
   int i,j;

   sprintf(lcd_line2,"%d\0",round_fp(min_lev));
   fp_temp1 = (level - min_lev) / (max_lev - min_lev);                    810
   fp_temp1 = fp_temp1 * 18;
   j= (int) fp_temp1;
   for(i=0;i<j;i++)
      strcat(lcd_line2,"\377\0");
   strcat(lcd_line2,"\377\0");
   for(i=0;i<(18-j);i++)
      strcat(lcd_line2,"\245\0");
   sprintf(lcd_line2,"%s%d\0",lcd_line2,round_fp(max_lev));
}                                                                         820

/*************************************************************************/
/** START_CONVERTER()  This procedure starts the converter.             **/
/*************************************************************************/
void start_converter(void)
{
   if(safety_mode!=NONE)    /* Skip comm check if safety=NONE */
   {
      if(check_com())
      {                                                                   830
         err_num=12;
         halt_converter();
         return;
      }
   }

   menu_mode=CONVERTER_RUNNING;
   debug_lev=0;
   di();                    /* Disable interrupts */
   fp_ibat=0;               /* Clear the sampled variables */             840
   fp_vbat=0;
   fp_iac=0;
   fp_vac=0;
   fp_speed=0;
   fp_speed_last=0;
   fp_speed_last2=0;
   fp_speed_filt1=0;
   fp_speed_filt2=0;
   pin_filtered=0;
   step_dly=0;                                                            850
   line_cycs=6;
   fp_icmd=0;
   int_icmd=0;
   fp_vref=fp_vset;
   fp_tref=fp_tset;

   P[0]=1000.0;                  /* Initialize some RLS variables */
   P[1]=0.0;
   P[2]=0.0;
   P[3]=0.0;                                                              860
   P[4]=1000.0;
   P[5]=0.0;
   P[6]=0.0;
   P[7]=0.0;
   P[8]=1000.0;
   theta[0]=0;
   theta[1]=0;
   theta[2]=0;
   fp_pwr_cmd=PWR_MIN;
   fp_pwr_cmd_last=PWR_MIN;                                               870
   fp_pwr_cmd_last2=PWR_MIN;
   fp_pwr_filt1=PWR_MIN;
   fp_pwr_filt2=PWR_MIN;

                  /*************************/
                  /**** Start BOOST MODE ****/
                  /*************************/
   leds = LED_PC;       /* Turn ON the POWER & CHARGE LEDs */
   reg_U42 = 0x71;      /* Set RELAY_TTL  = 1     OPEN RELAY_1     */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM  */     880
                        /* Set DS_A0      = 0                      */
                        /* Set DS_A1      = 0                      */
                        /* Set L-4        = 1                      */
                        /* Set L-5 (B_MDE)= 1                      */
                        /* Set L-6 (RS)   = 1     LCD Data Mode    */
                        /* Set L-7 (RLY2) = 0     CLOSE RELAY_2    */
   SET_reg_U42(reg_U42);

   mode=BOOST_STARTUP_1;
                                                                          890
```

```
   swt_count=0;                  /* Reset the swt_count */
   swt_delay=PRE_CHARGE_TIME;    /* Set a new delay time*/
   int_pending = 0;
   if(ctrl_mode==DEBUG_CONST_K)
      int_mask = 0x13;           /* Unmask TIMER_MASK, HSI0_MASK, AD_MASK. */
   else
      int_mask = 0x93;           /* Unmask ALL interrupts: TIMER_MASK, */
                                 /* EXTINT_MASK, HSI0_MASK, AD_MASK.   */
   ei();                         /* Re-enable interrupts */
   wait_fun(5000);               /* Delay here for a split sec and   */
   button_id=0;                  /* flush the key buffer.            */
}

/*************************************************************************/
/** SELECT_CONTROL_MODE()  This procedure controls the MENU & DISPLAY **/
/*************************************************************************/
void select_control_mode(void)
{
   switch(ctrl_mode)
   {
   case VOLTAGE_STEP: sprintf(lcd_line1,"Control Mode:   VOLTAGE STEP RESPONSE   ");
          break;
   case VOLTAGE_RAMP: sprintf(lcd_line1,"Control Mode:   VOLTAGE RAMP RESPONSE   ");
          break;
   case DEBUG_CONST_K: sprintf(lcd_line1,"Control Mode:    DEBUG:  CONSTANT K     ");
          break;
   case SPEED_CONTROL: sprintf(lcd_line1,"Control Mode:     SPEED CONTROL        ");
          break;
   case SPEED_STEP_PP: sprintf(lcd_line1,"Control Mode:    PP SPEED STEPS (N=1)     ");
          break;
   case SPEED_STEP_ADP: sprintf(lcd_line1,"Control Mode:    ADAPTIVE SPEED (N=1)     ");
          break;
   default: sprintf(lcd_line1,"Control Mode:     VOLTAGE CONTROL       ");
           break;
   }
   sprintf(lcd_line2,"    (Press <- or -> to change modes)    ");
   if(button_id!=0)
   {
      switch(button_id)
      {
      case MENU_ID: menu_mode=SELECT_SAFETY_MODE;     /* MENU */
                    break;
      case LEFT_ID: if(ctrl_mode==VOLTAGE_CONTROL)    /* LEFT ARROW KEY  */
                       ctrl_mode=SPEED_STEP_ADP;
                    else
                       ctrl_mode--;
                    break;
      case RIGHT_ID: if(ctrl_mode==SPEED_STEP_ADP)     /* RIGHT ARROW KEY */
                        ctrl_mode=VOLTAGE_CONTROL;
                     else
                        ctrl_mode++;
                     break;
      default: break;                                 /* ALL OTHER KEYS  */
      }
      button_id=0;
      button_db=0;
   }
}

/*************************************************************************/
/** SELECT_SAFETY_MODE()  This procedure controls the MENU & DISPLAY **/
/*************************************************************************/
void select_safety_mode(void)
{
   switch(safety_mode)
   {
      case SOME: sprintf(lcd_line1,"Safety Checks:      SOME ENABLED        \0");
                 break;
      case NONE: sprintf(lcd_line1,"Safety Checks:    NONE - Be Careful!    \0");
                 break;
      default: sprintf(lcd_line1,"Safety Checks:       ALL ENABLED        \0");
               break;
   }
   sprintf(lcd_line2,"      (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                       menu_mode=SPARE_SELECT_MENU;
                       break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
                       if(safety_mode==ALL)
```

```
                        safety_mode=NONE;
                     else
                        safety_mode--;
                     break;                                                    980
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                     if(safety_mode==NONE)
                        safety_mode=ALL;
                     else
                        safety_mode++;
                     break;
         default: break;
      }
      button_id=0;
      button_db=0;                                                             990
   }
}


/*********************************************************************/
/** SPARE_SELECT_MENU()  This procedure controls the MENU & DISPLAY  **/
/*********************************************************************/
void spare_select_menu(void)
{
   if(update_asap)                                                           1000
      sprintf(lcd_line1,"Control loop updates:  ASAP                  \0");
   else
      sprintf(lcd_line1,"Control loop updates:  1 Unit Delayed        \0");
   sprintf(lcd_line2,"      (Press <- or -> to change)        \0");
   if(button_id!=0)
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEY ***/
                     menu_mode=CONVERTER_STOPPED;                            1010
                     break;
         case LEFT_ID: /*** LEFT ARROW KEY ***/
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                     if(update_asap)
                        update_asap=0;
                     else
                        update_asap=1;
                     break;
         default: break;
      }                                                                        1020
      button_id=0;
      button_db=0;
   }
}


/*********************************************************************/
/** LCD_MENU_RUNNING()  This procedure controls the MENU & DISPLAY **/
/*********************************************************************/
void lcd_menu_running(void)                                                 1030
{
                  /*** LP-Filter Pin & Pdc ***/
   pin_filtered = (pin_filtered * 0.8) + (fp_vac * fp_iac * 0.2);

   if(debug_lev==0)
   {
      sprintf(lcd_line1,"Pdc/Pin=??.?%% \0");
      sprintf(lcd_line2,"????W/%4dW\0",round_fp(pin_filtered));

            /*** Now finish laying out the LCD display ***/                  1040
      sprintf(lcd_line1,"%sVac=%3dV Vbt=%3dV Spd=%1d.%1d      ", lcd_line1, round_fp(fp_vac),
                round_fp(fp_vbst), int_part(fp_speed), dec_part(fp_speed));
      sprintf(lcd_line2,"%s  Iac=%2d.%1dA Ibt=%1d.%1dA                ", lcd_line2,
                int_part(fp_iac), dec_part(fp_iac), int_part(fp_ibst), dec_part(fp_ibst));
   }
   else if(debug_lev==1)
   {
      sprintf(lcd_line1,"DB: VAC=%3d.%1dV   VBT=%3d.%1dV   Spd=%2d.%1dV   ",
                int_part(fp_vac), dec_part(fp_vac), int_part(fp_vbst), dec_part(fp_vbst),
                   int_part(fp_speed), dec_part(fp_speed));                  1050
      sprintf(lcd_line2,"#1: IAC=%3d.%1dV   IBT=%3d.%1dV                  ",
                int_part(fp_iac), dec_part(fp_iac), int_part(fp_ibst), dec_part(fp_ibst));
   }
   else if(debug_lev==2)
   {
      sprintf(lcd_line1,"DB: K=%4d  Kmax=%4d  PWR_C=%3d                   ",
                int_icmd, (uint) fp_kmax, round_fp(fp_pwr_cmd));
      sprintf(lcd_line2,"#2: N1=%3d N2=%3d R1=%3d R2=%3d                  ",
                round_fp(fp_tn1), round_fp(fp_tn2), round_fp(fp_rc1), round_fp(fp_rc2));
   }                                                                        1060

   if(button_id!=0)
```

```
    {
       switch(button_id)
       {
          case MENU_ID: /*** MENU KEU ***/
                        debug_lev++;                    /* Switch to the debug screens  */
                        if(debug_lev > 2)
                           debug_lev = 0;
                        break;                                                          1070
          case LEFT_ID: /*** LEFT ARROW KEY ***/
                        if(ctrl_mode ==VOLTAGE_CONTROL && mode==BOOST_RUNNING)
                        {
                           if(fp_vset > (VOL_MIN_FP + VOL_STEP_FP))
                              fp_vset -= VOL_STEP_FP;
                           else
                              fp_vset = VOL_MIN_FP;
                        }
                        else if(ctrl_mode==DEBUG_CONST_K)
                        {                                                               1080
                           if(fp_const_k > (CK_MIN_FP + CK_STEP_FP))
                              fp_const_k -= CK_STEP_FP;
                           else
                              fp_const_k = CK_MIN_FP;
                        }
                        break;
          case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                         if(ctrl_mode==VOLTAGE_CONTROL && mode==BOOST_RUNNING)
                         {
                            if(fp_vset < (VOL_MAX_FP - VOL_STEP_FP))                    1090
                               fp_vset += VOL_STEP_FP;
                            else
                               fp_vset = VOL_MAX_FP;
                         }
                         else if(ctrl_mode==DEBUG_CONST_K)
                         {
                            if(fp_const_k < (CK_MAX_FP - CK_STEP_FP))
                               fp_const_k += CK_STEP_FP;
                            else
                               fp_const_k = CK_MAX_FP;                              1100
                            break;
                         }
                         break;
          case START_ID: /*** START BUTTON ***/
                         err_num=0;                     /* Shutdown the converter    */
                         halt_converter();              /* without an error flag.    */
                         break;
          default: break;
       }
       button_id=0;                                                                     1110
       button_db=0;
    }
}


/*********************************************************************/
/** LCD_MENU_HALT()  This procedure controls the MENU & DISPLAY     **/
/*********************************************************************/
void lcd_menu_halt(void)
{                                                                                       1120
   switch(err_num)
   {
   case 1: sprintf(lcd_line1,"**** ERROR #%d: Vac too HIGH (%dV) *******",err_num,
                   round_fp(fp_vac));
           break;
   case 2: sprintf(lcd_line1,"***** ERROR #%d: Vac too LOW (%dV) *******",err_num,
                   round_fp(fp_vac));
           break;
   case 3: sprintf(lcd_line1,"**** ERROR #%d: Iac too HIGH (%d.%dA) *******",err_num,
                  int_part(fp_iac), dec_part(fp_iac));                              1130
           break;
   case 4: sprintf(lcd_line1,"**** ERROR #%d: Vbt too HIGH (%dV) *******",err_num,
                   round_fp(fp_vbst));
           break;
   case 5: sprintf(lcd_line1,"**** ERROR #%d: Ibt too HIGH (%d.%dA) *******",err_num,
                  int_part(fp_ibst), dec_part(fp_ibst));
           break;
   case 9: sprintf(lcd_line1,"*** ERROR #%d: NO INPUT (Cycles = %u) ******",err_num,
                   line_cycs);
           break;                                                                   1140
   case 11: sprintf(lcd_line1,"**** ERROR #%d: PLEASE CHECK PADDLE *******",err_num);
            break;
   case 12: sprintf(lcd_line1,"* CHECK THE PADDLE AND SECONDARY POWER *",err_num);
            break;
   default: sprintf(lcd_line1,"**** A fault has occured: ERROR #%u ******",err_num);
            break;
   }
   sprintf(lcd_line2,"   (Press ANY KEY to clear the error)   ");
```

```
   if(button_id!=0)                    /* If ANY KEY is pressed then...  */
   {
      menu_mode=CONVERTER_STOPPED;     /* Return to MAIN MENU            */
      leds=LED_P;                      /* Turn ON the POWER LED          */

      button_id=0;
      button_db=0;
   }
}

/*********************************************************************/
/** LCD_MENU_STOP()  This procedure controls the MENU & DISPLAY     **/
/*********************************************************************/
void lcd_menu_stop(void)
{
   switch(ctrl_mode)
   {
   case VOLTAGE_STEP: /* Voltage STEP */
   case VOLTAGE_RAMP: /* Voltage RAMP */
         sprintf(lcd_line1,"Select a second voltage and press START.");
         make_bar_graph(fp_vset2,VOL_MIN_FP,VOL_MAX_FP);
         sprintf(lcd_line2,"%sA  (Level=%3dV)        ", lcd_line2, round_fp(fp_vset2));
         break;
   case DEBUG_CONST_K: /* DEBUG: Constant K */
         sprintf(lcd_line1,"Select a value for K and press START.   ");
         make_bar_graph(fp_const_k,CK_MIN_FP,CK_MAX_FP);
         sprintf(lcd_line2,"%sA  (K=%4d)         ", lcd_line2, (uint) fp_const_k);
         break;
   case SPEED_CONTROL: /* SPEED Control */
         sprintf(lcd_line1,"Select a speed and then press START.    ");
         make_bar_graph(fp_tset,SPEED_MIN_FP,SPEED_MAX_FP);
         sprintf(lcd_line2,"%sV (Level=%1d.%1d)        ", lcd_line2,
                 int_part(fp_tset),dec_part(fp_tset));
         break;
   case SPEED_STEP_PP:  /* SPEED STEP */
   case SPEED_STEP_ADP:  /* SPEED STEP */
         sprintf(lcd_line1,"Select a second speed then press START.  ");
         make_bar_graph(fp_tset2,SPEED_MIN_FP,SPEED_MAX_FP);
         sprintf(lcd_line2,"%sV (Level=%1d.%1d)        ", lcd_line2,
                 int_part(fp_tset2),dec_part(fp_tset2));
         break;
   default: /* Voltage Control */
          sprintf(lcd_line1,"Select a voltage level then press START.");
          make_bar_graph(fp_vset,VOL_MIN_FP,VOL_MAX_FP);
          sprintf(lcd_line2,"%sA  (Level=%3d)         ", lcd_line2, round_fp(fp_vset));
          break;
   }

   if(button_id!=0)           /***  Now interpret the menu KEYS ***/
   {
      switch(button_id)
      {
      case MENU_ID: /*** MENU KEU ***/
                    menu_mode = SELECT_CONTROL_MODE;
                    break;
      case LEFT_ID: /*** LEFT ARROW KEY ***/
                    switch(ctrl_mode)
                    {
                       case VOLTAGE_STEP: /* Voltage STEP */
                       case VOLTAGE_RAMP: /* Voltage RAMP */
                              if(fp_vset2 > (VOL_MIN_FP + VOL_STEP_FP))
                                 fp_vset2 -= VOL_STEP_FP;
                              else
                                 fp_vset2 = VOL_MIN_FP;
                              break;
                       case DEBUG_CONST_K: /* Constant K */
                               if(fp_const_k > (CK_MIN_FP + CK_STEP_FP))
                                  fp_const_k -= CK_STEP_FP;
                               else
                                  fp_const_k = CK_MIN_FP;
                               break;
                       case SPEED_CONTROL: /* Speed Control */
                               if(fp_tset > (SPEED_MIN_FP + SPEED_STEP_FP))
                                  fp_tset -= SPEED_STEP_FP;
                               else
                                  fp_tset = SPEED_MIN_FP;
                               break;
                       case SPEED_STEP_PP: /* Speed Control */
                       case SPEED_STEP_ADP: /* Speed Control */
                               if(fp_tset2 > (SPEED_MIN_FP + SPEED_STEP_FP))
                                  fp_tset2 -= SPEED_STEP_FP;
                               else
                                  fp_tset2 = SPEED_MIN_FP;
                               break;
```

```
                    default: /* Voltage Control */
                             if(fp_vset > (VOL_MIN_FP + VOL_STEP_FP))
                                fp_vset -= VOL_STEP_FP;
                             else
                                fp_vset = VOL_MIN_FP;
                             break;                                          1240
                    }
                   break;
      case RIGHT_ID: /*** RIGHT ARROW KEY ***/
                   switch(ctrl_mode)
                   {
                      case VOLTAGE_STEP: /* Voltage STEP */
                      case VOLTAGE_RAMP: /* Voltage RAMP */
                             if(fp_vset2 < (VOL_MAX_FP - VOL_STEP_FP))
                                fp_vset2 += VOL_STEP_FP;
                             else                                            1250
                                fp_vset2 = VOL_MAX_FP;
                             break;
                      case DEBUG_CONST_K: /* Constant K */
                             if(fp_const_k < (CK_MAX_FP - CK_STEP_FP))
                                fp_const_k += CK_STEP_FP;
                             else
                                fp_const_k = CK_MAX_FP;
                             break;
                      case SPEED_CONTROL: /* Speed Control */
                             if(fp_tset < (SPEED_MAX_FP - SPEED_STEP_FP))    1260
                                fp_tset += SPEED_STEP_FP;
                             else
                                fp_tset = SPEED_MAX_FP;
                             break;
                      case SPEED_STEP_PP: /* Speed Control */
                      case SPEED_STEP_ADP: /* Speed Control */
                             if(fp_tset2 < (SPEED_MAX_FP - SPEED_STEP_FP))
                                fp_tset2 += SPEED_STEP_FP;
                             else
                                fp_tset2 = SPEED_MAX_FP;                     1270
                             break;
                      default: /* Voltage Control */
                              if(fp_vset < (VOL_MAX_FP - VOL_STEP_FP))
                                 fp_vset += VOL_STEP_FP;
                              else
                                 fp_vset = VOL_MAX_FP;
                              break;
                   }
                   break;
      case START_ID: /*** START BUTTON ***/                                 1280
                   menu_mode = PREPARE_TO_START;    /* Goto the about to start menu */
                   break;
      default: break;
      }
      button_id=0;
      button_db=0;
   }
}

                                                                           1290
/*********************************************************************/
/** LCD_MENU_START()  This procedure controls the MENU & DISPLAY    **/
/*********************************************************************/
void lcd_menu_start(void)
{
   if(ctrl_mode==DEBUG_CONST_K)
      sprintf(lcd_line1,"About to begin const K.  USE CAUTION!!     ");
   else
   {
      sprintf(lcd_line1,"About to begin CL \0");                           1300
      if(ctrl_mode==VOLTAGE_CONTROL)
         sprintf(lcd_line1,"%sVOLTAGE control.              \0",lcd_line1);
      if(ctrl_mode==VOLTAGE_STEP)
         sprintf(lcd_line1,"%sVOLTAGE steps.                \0",lcd_line1);
      if(ctrl_mode==VOLTAGE_RAMP)
         sprintf(lcd_line1,"%sVOLTAGE ramps.                \0",lcd_line1);
      if(ctrl_mode==SPEED_CONTROL)
         sprintf(lcd_line1,"%sSPEED control.                \0",lcd_line1);
      if(ctrl_mode==SPEED_STEP_PP || ctrl_mode==SPEED_STEP_ADP)
         sprintf(lcd_line1,"%sSPEED steps.                  \0",lcd_line1);  1310
   }
   sprintf(lcd_line2,"(Press START to begin or MENU to abort.) \0");

   if(button_id!=0)           /***  Now interpret the menu KEYS ***/
   {
      switch(button_id)
      {
         case MENU_ID: /*** MENU KEU ***/
         case LEFT_ID: /*** LEFT ARROW KEY ***/
         case RIGHT_ID: /*** RIGHT ARROW KEY ***/                          1320
```

```
                    menu_mode = CONVERTER_STOPPED;      /* Return to the Main menu */
                    break;
        case START_ID: /*** START BUTTON ***/
                    start_converter();
                    break;
        default: break;
      }
      button_id=0;
      button_db=0;
   }                                                                          1330
}


/*************************************************************************/
/** INITIALIZE_GLOBALS()  This procedure inits various global variables **/
/*************************************************************************/
void initialize_globals(void)
{
   hold_iop2=0x40;                 /* Initialize ioport2                    */
   hold_ios0=0x00;                 /* Initialize ios0                       */   1340
   debug_lev=0;                    /* Zero the debug level.                 */
   safety_mode=ALL;                /* Zero the safety level.                */
   line_cycs=6;                    /* Preset the line-cycle counter         */
   step_dly=0;                     /* Zero the step delay counter.          */
   mode=OFF;                       /* Set the startup mode to OFF.          */
   int_icmd=0;                     /* Set the initial current command to zero.*/
   fp_vref=0;                      /* Set an initial value for iref.        */
   fp_vset=VOL_MIN_FP;             /* Initialize the volatge setting.       */
   fp_vset2=VOL_MIN_FP;            /* Initialize the voltage setting.       */
   fp_tref=0;                      /* Set an initial value for iref.        */   1350
   fp_tset=SPEED_MIN_FP;           /* Initialize the speed setting.         */
   fp_tset2=SPEED_MIN_FP;           /* Initialize the speed setting.        */
   fp_const_k = CK_MIN_FP;         /* Initialize the k setting.             */
   menu_mode=CONVERTER_STOPPED;    /* Initialize the menu mode.             */
   ctrl_mode=VOLTAGE_CONTROL;      /* Initialize CONSTANT VOLTAGE control.  */
   leds=LED_P;                     /* Set only the POWER LED for starters.  */
   strobe=0;                       /* Clear strobe.                         */
   button_id=0;                    /* Clear the button_id.                  */
   button_db=0;                    /* Clear the button debounce counter     */
   reset_ints=TRUE;                /* Set the reset_ints flag.              */   1360

   fp_vbst=0;                      /* Clear floating point a/d values       */
   fp_vac=0;
   fp_ibst=0;
   fp_iac=0;
   fp_speed=0;
   fp_kmax_const=KMAX_CONST_FP;
   update_asap=0;
}
                                                                              1370

/*************************************************************************/
/** INITIALIZE_HARDWARE()  This procedure inits the hardware setup.  **/
/*************************************************************************/
void initialize_hardware(void)
{
   wsr = 0;
   ioc1 = 0x54;          /* Enable HSO.4, HSO.5 and set P2.2 as EXTINT */
   ioport2 = hold_iop2; /* Preset ioport2 */
                                                                              1380
   HSO_set(0);          /* Set the /LDAC  pin */
   HSO_set(1);          /* Set the /CSMSB pin */
   HSO_set(2);          /* Set the /WR    pin */
   HSO_set(3);          /* Set the /CSLSB pin */
   HSO_clr(4);          /* Set the reg_U42 LATCH LOW */
   HSO_set(5);          /* Set the /WR line on the DISPLAY to HIGH */

   MDAC_write();        /* Write a starting int_icmd = 0 */
                        /* to the DAC.                   */
                                                                              1390
   reg_U42=0xB1;        /* Set RELAY_TTL  = 1     OPEN RELAY_1    */
                        /* Set PWM_EN_TTL = 0     DISABLE the PWM */
                        /* Set DS_A0      = 0                     */
                        /* Set DS_A1      = 0                     */
                        /* Set L-4        = 1                     */
                        /* Set L-5 (B_MDE)= 1                     */
                        /* Set L-6 (RS)   = 0     LCD Inst. Mode  */
                        /* Set L-7 (RLY2) = 1     OPEN RELAY_2    */
   SET_reg_U42(reg_U42);
                                                                              1400
   reg_U2=0x52;         /* Set B1_ENABLE    = 0   DISABLE B1    */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS  */
                        /* Set B2_ENABLE    = 0   DISABLE B2    */
                        /* Set FB_LDAC      = 0                 */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS */
                        /* Set B1_RLY_RESET = 0                 */
```

```
                    /* Set B2_RLY_SET   = 1   SET B2 RELAYS */
                    /* Set B2_RLY_RESET = 0                 */
  SET_reg_U2(reg_U2);                                                    1410

  strcpy(display,"INIT");
  write_hp_display();           /* Direct write "INIT" to the screen */

  wait_fun(10000);              /* Wait for the LCD to become ready */
                                /* after a powerup.                 */
  ioport1 = 0x3C;               /* Init the LCD FUNCTION-SET Instruction */
  P2_setb(5);                   /* Strobe P2-5 to latch LCD data */
  P2_clrb(5);
  wait_fun(100);
                                                                         1420
  ioport1 = 0x0C;               /* Set LCD display control to ON */
  P2_setb(5);                   /* Strobe P2-5 to latch LCD data */
  P2_clrb(5);
  wait_fun(100);

  ioport1 = 0x01;               /* Clear the LCD */
  P2_setb(5);                   /* Strobe P2-5 to latch LCD data */
  P2_clrb(5);
  wait_fun(30000);              /* Wait a few milliseconds for the clear */
                                                                         1430
  SET_leds(leds);               /* Turn on the POWER LED only */

  reg_U2=0x52;          /* Set B1_ENABLE    = 0   DISABLE B1    */
                        /* Set FAULT_CLR    = 1   CLEAR FAULTS  */
                        /* Set B2_ENABLE    = 0   DISABLE B2    */
                        /* Set FB_LDAC      = 0                 */
                        /* Set B1_RLY_SET   = 1   SET B1 RELAYS */
                        /* Set B1_RLY_RESET = 0                 */
                        /* Set B2_RLY_SET   = 1   SET B2 RELAYS */
                        /* Set B2_RLY_RESET = 0                 */   1440
  SET_reg_U2(reg_U2);
}


/*********************************************************************/
/** MAIN()  This is the main procedure.  Most of the processing  **/
/** is really done in the ISR routines.  The main routine calls  **/
/** a number of procedure that maintain the menu system.         **/
/*********************************************************************/
main(void)                                                               1450
{
          /*************************************/
          /* These macros drop in some in-line */
          /* assembly to define the IRQ table. */
          /*************************************/
  set_vector(2000h,software_timer);
  set_vector(2002h,analog_conversion_done);
  set_vector(2008h,hsi_event);
  set_vector(200eh,line_sync_interrupt);
                                                                         1460
                        /*****************************/
  initialize_globals();     /* INITIALIZE VARIABLES      */
  initialize_hardware();    /* INITIALIZE HARDWARE       */
                        /*****************************/

   /*************************************************************************/
   /** MAIN LOOP -- All the menu processing and display control is performed **/
   /** here.  The boost control functions are interrupt driven.            **/
   /*************************************************************************/
  while(1)                                                               1470
  {
    switch(menu_mode)
    {
      case 1: /*** Select CONTROL MODE ***/
              select_control_mode();        /* Display/select control mode on LCD */
              break;
      case 3: /*** Select SAFETY MODE ***/
              select_safety_mode();         /* Dsiplay/select safety mode on LCD */
              break;
      case 4: /*** Select spare MENU ***/                                 1480
              spare_select_menu();          /* Dsiplay/select spare menu on LCD  */
              break;
      case 5: /*** Converter Running ***/
              strcpy(display,"RUN ");
              lcd_menu_running();           /* Display RUN info on the LCD    */
              break;
      case 6: /*** Converter Error Halt ***/
              strcpy(display,"HALT");
              lcd_menu_halt();              /* Display HALT info on the LCD   */
              break;                                                     1490
      case 7: /*** About to start menu ***/
              lcd_menu_start();
```

```
                break;
        default: /*** Converter Stopped ***/
                 strcpy(display,"STOP");
                 lcd_menu_stop();              /* Display/select STOP info on the LCD */
                 break;
      }
      write_lcd_line(1);         /* Write line #1 on the LCD     */
      write_lcd_line(2);         /* Write line #2 on the LCD     */
      write_hp_display();        /* Write 4 chars to HP display */
      wait_fun(LCD_DELAY);
      SET_leds(leds);            /* Update the 4 LED's           */

        /*** Check line-cycle count for loss of Input Voltage ***/
        /*** Only check when !OFF, or !STARTUP_1               ***/
      if(ctrl_mode!=DEBUG_CONST_K)
      {
         if((mode!=BOOST_STARTUP_1) && (mode!=OFF))
         {
            if((line_cycs > 8) || (line_cycs < 5))
            {
               err_num=9;
               halt_converter();
            }
         }
      }
      if(reset_ints==TRUE)
      {
         reset_ints=FALSE;
         int_pending = 0;            /* Clear any pending interrupts          */
         int_mask = 0x11;            /* Unmask only the SWT_MASK and HSI0_MASK */
         ei();                       /* Enable interrupts                     */
         wait_fun(30000);            /* Force a small delay then continue     */
         button_id=0;
      }
   }
}  /* End main */
```

## "C" Header File - MOTOR1.H

```
/*********************************************************************/
/* FILE: MOTOR1.H                                  REVISION: 1.0     */
/* BY: Deron Jackson                               DATE: 9/2/97      */
/*                                                                   */
/* This is a HEADER FILE for MOTOR1.C                                */
/*********************************************************************/


   /**************************/                                          10
   /** Constant Definitions **/
   /**************************/
#define DEBUG        1       /* Set this to 1 for DEBUG messages.      */
#define TRUE         1
#define FALSE        0

#define uchar unsigned char
#define uint unsigned int

                                                                        20
/*****************************************/
/* These are some convenient macros for    */
/* setting 80c196 hardware ports.  They    */
/* ensure that WSR is set correctly.       */
/*****************************************/
#define HSO_set(bit) \
             wsr=15; \
             hold_ios0 |= (0x01 << (bit)); \
             ios0 = hold_ios0; \
             wsr=0                                                      30
#define HSO_clr(bit) \
             wsr=15; \
             hold_ios0 &= ~(0x01 << (bit)); \
             ios0 = hold_ios0; \
             wsr=0
#define P2_setb(bit) \
             hold_iop2 |= (0x01 << (bit)); \
             ioport2 = hold_iop2
#define P2_clrb(bit) \
             hold_iop2 &= ~(0x01 << (bit)); \                           40
             ioport2 = hold_iop2
```

```
#define REG_U2_setb(bit) \
            reg_U2 |= (0x01 << (bit)); \
            ioport1=(unsigned char) reg_U2; \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define REG_U2_clrb(bit) \
            reg_U2 &= ~(0x01 << (bit)); \
            ioport1=(unsigned char) reg_U2; \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define SET_reg_U2(val) \
            ioport1=(unsigned char) (val); \
            hold_iop2 |= 0x01; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x01; \
            ioport2 = hold_iop2
#define REG_U42_setb(bit) \
            reg_U42 |= (0x01 << (bit)); \
            ioport1=(unsigned char) reg_U42; \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define REG_U42_clrb(bit) \
            reg_U42 &= ~(0x01 << (bit)); \
            ioport1=(unsigned char) reg_U42; \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define SET_reg_U42(val) \
            ioport1=(unsigned char) (val); \
            wsr=15; \
            hold_ios0 |= 0x10; \
            ios0 = hold_ios0; \
            hold_ios0 &= ~0x10; \
            ios0 = hold_ios0; \
            wsr=0
#define SET_leds(val) \
            ioport1=(unsigned char) (val); \
            hold_iop2 |= 0x80; \
            ioport2 = hold_iop2; \
            hold_iop2 &= ~0x80; \
            ioport2 = hold_iop2
#define IRQ_ENTRY \
            hold_wsr=wsr; \
            wsr=0; \
            hold_ioport1=ioport1
#define IRQ_EXIT \
            ioport1=hold_ioport1; \
            wsr=hold_wsr

/*****************************************/
/* The following constant definitions are  */
/* hardware specific.  The specific values */
/* must be calibrated for each board set.  */
/* (See the MATHCAD file BST_LP1.MCD)      */
/*****************************************/
#define H1_FP            5.99460e2  /* Boost current-loop gain H1              */
#define H2_FP           -5.54500e2  /* Boost current-loop gain H2              */
#define HP_FP            2.36194e4  /* Boost current-loop power FF gain HP     */

   /* Desired closed-loop poles are at 0.7,...0.7 */
#define N1_FP            7.20760e2  /* PP Num control gain                     */
#define N2_FP           -1.23600e3  /* PP Num control gain                     */
#define N3_FP            5.31930e2  /* PP Num control gain                     */
#define D1_FP           -9.77380e-1 /* PP Den control gain                     */
#define D2_FP           -2.26150e-2 /* PP Den control gain                     */

#define LAMBDA1          9.70000e-1 /* Lambda for RLS1 loop                    */
#define CL1_DENA_FP     -2.80000e0  /* Denominator coeff for RLS1              */
#define CL1_DENB_FP      2.94000e0  /* Denominator cceff for RLS1              */
#define CL1_DENC_FP     -1.37200e0  /* Denominator coeff for RLS1              */
#define CL1_DEND_FP      2.40100e-1 /* Denominator coeff for RLS1              */
#define TAU_FP             10.0     /* Filter time constant for lambda method  */
#define TAU_SQ_FP         100.0     /* Filter time constant for lambda method  */
```

```
#define VBST_TO_FP       1.47263e-1 /* Conversion ratio for Vbst Int-to-Float   */
#define IBST_TO_FP       7.91994e-3 /* Conversion ratio for Ibst Int-to-Float   */
#define VAC_TO_FP        3.06026e-1 /* Conversion ratio for Vac Int-to-Float    */
#define IAC_TO_FP        1.91220e-2 /* Conversion ratio for Iac Int-to-Float    */
#define DC_V1_TO_FP      3.91007e-1 /* Conversion ratio for V1 Int-to-Float     */
#define SPEED_TO_FP      4.88759e-3 /* Conversion ratio for SPEED Int-to-Float   */

#define PWR_MAX            71.0     /* Output POWER maximum value.              */
#define PWR_MIN            25.0     /* Output POWER minimum value.              */
#define VBST_HIGH_FP      395.0     /* Output voltage upper limit.              */
#define IBST_HIGH_FP        4.0     /* Output current upper limit.              */
#define IAC_HIGH_FP        15.0     /* Input current upper limit.               */
#define VAC_HIGH_FP       150.0     /* Input voltage upper limit.               */
#define VAC_LOW_FP         90.0     /* Boost shuts down for low voltages.       */
#define VBST_OFFSET_FP    250.0     /* This is an offset value added to the VBST*/
#define BIAS_VOLTAGE_FP    55.0
#define ENDOFSOFTSTART_FP 260.0     /* Vbst voltage at end of soft_start.       */
#define ICMD_INC_FP         5.0     /* Open loop soft_start increment.          */
#define VREF_INC_FP         2.5     /* Closed-loop soft-start increment.        */
#define VOL_MIN_FP        260.0     /* This is the MENU current minimum.        */
#define VOL_STEP_FP         5.0     /* This is the MENU step size.              */
#define VOL_MAX_FP        390.0     /* This is the MENU current maximum.        */
#define SPEED_MIN_FP        1.0
#define SPEED_STEP_FP       0.1
#define SPEED_MAX_FP        6.0
#define CK_MIN_FP           0.0     /* This is the MENU K minimum.              */
#define CK_STEP_FP         50.0     /* This is the MENU K step size.            */
#define CK_MAX_FP        2000.0     /* This is the MENU K maximum.              */

#define KMAX_CONST_FP    2.83433e5  /* This is used to calculate the MAX i_cmd  */

#define MAX_ICMD_FP       4095.0    /* This is the absolute MAX value of i_cmd  */

#define MENU_ID         0x40  /* MENU button ID          */
#define LEFT_ID         0x48  /* LEFT ARROW button ID    */
#define RIGHT_ID        0x50  /* RIGHT ARROW button ID   */
#define START_ID        0x58  /* START-STOP button ID    */

#define VOLTAGE_CONTROL   0  /* Controller mode setting */
#define VOLTAGE_STEP      1  /* Controller mode setting */
#define VOLTAGE_RAMP      2  /* Controller mode setting */
#define DEBUG_CONST_K     3  /* Controller mode setting */
#define SPEED_CONTROL     4  /* Controller mode setting */
#define SPEED_STEP_PP     5  /* Controller mode setting */
#define SPEED_STEP_ADP    6  /* Controller mode setting */

#define REVERSE           0  /* Controller direction setting */
#define FORWARD           1  /* Controller direction setting */

#define CONVERTER_STOPPED     0  /* Display mode setting */
#define SELECT_CONTROL_MODE   1  /* Display mode setting */
#define SELECT_SAFETY_MODE    3  /* Display mode setting */
#define SPARE_SELECT_MENU     4  /* Display mode setting */
#define CONVERTER_RUNNING     5  /* Display mode setting */
#define HALT_ERROR_MENU       6  /* Display mode setting */
#define PREPARE_TO_START      7  /* Display mode setting */

#define ALL                   0  /* Safety mode: ALL                          */
#define SOME                  1  /* Safety mode: Just COMM, Vbst HIGH, Line Cycs */
#define NONE                  2  /* Safety mode: Just Vbst HIGH, Line Cycs    */

#define OFF                   0  /* mode: OFF                                 */
#define BOOST_STARTUP_1       1  /* mode: Wait cap pre-charge, close INRSH.   */
#define BOOST_STARTUP_2       2  /* mode: Enable PWM, enable DC/DC Board 1.   */
#define BOOST_STARTUP_3       3  /* mode: Open-loop Ramp                      */
#define BOOST_OL_SOFT_START   4  /* mode: BOOST soft-start                    */
#define BOOST_CL_SOFT_START   5  /* mode: BOOST soft-start                    */
#define BOOST_RUNNING         6  /* mode: BOOST is running                    */

#define LCD_DELAY            50  /* This is a delay time between LCD updates  */
#define BUTTON_DB             2  /* This is a delay for button debouncing     */
#define PRE_CHARGE_TIME      15  /* This sets the time for cap pre-charge.    */
                                 /* The value is in 15ths of a second.        */
                                 /* (NOTE: This number must be > 5.)          */
#define DC_DC_DELAY          15  /* This sets the time delay after DC-DC      */
                                 /* turn-on.  The value is in 15ths of a sec. */
#define MINI_DELAY            3  /* This is a mini time delay.                */
#define SOFT_START_DELAY     50  /* This is the time delay for the soft-start */
                                 /* turn-on.  The value is in 15ths of a sec. */
                                 /* 76 is approximately 5 seconds.            */
#define LED_P          2         /* Byte to turn on LED's */
#define LED_PC         6         /* Byte to turn on LED's */
#define LED_PD        10         /* Byte to turn on LED's */
#define LED_PF         3         /* Byte to turn on LED's */
```

```
/**************************/
/** Variable Declarations **/
/**************************/

        /*** DISPLAY CONTROL VARIABLES ***/
near unsigned char reg_U2;          /* This is the value of register U2   */
near unsigned char reg_U42;         /* This is the value of register U42  */
near unsigned char hold_iop2;       /* This hold the value of ioport2     */
near unsigned char hold_ios0;       /* This hold the value of ios0        */
near unsigned char leds;            /* This holds the LED status          */
near unsigned char debug_lev;       /* This holds a setting used for debug*/
near unsigned char safety_mode;     /* This holds the safety menu setting */
near unsigned char update_asap;     /* Flag for loop update delay         */
near float pin_filtered;            /* Makes a Pin running avg. filter    */
near char display[4];               /* Holds the 4 digit display          */
char lcd_line1[256];                /* String LCD line1 text              */
char lcd_line2[256];                /* String LCD line2 text              */

        /*** VOLTAGE-LOOP CONTROL VARIABLES ***/
near unsigned int a_d_value;        /* This holds the 10 bit A/D value    */
near unsigned int int_ad_lo;        /* This holds the lo A/D value        */
near unsigned int int_ad_hi;        /* This holds the hi A/D value        */
near unsigned int int_icmd;         /* The current-loop command value (k) */
near unsigned int int_fbcmd;        /* The full-bridge command value      */
near float fp_kmax;                 /* This holds the max f-point i_cmd   */
near float fp_icmd;                 /* The current-loop command k in fp   */
near float fp_icmd_last;            /* The last value --> i_cmd[n-1]      */
near float fp_delta_k;              /* The change in k                    */
near float fp_vbst;                 /* The output voltage VBST.           */
near float fp_ibst;                 /* The output current                 */
near float fp_vac;                  /* The peak input voltage             */
near float fp_iac;                  /* The peak input current             */
near float fp_power;                /* The value of vbst * ibst           */
near float fp_power_last;           /* Last value of vbst * ibst          */
near float fp_x;                    /* The value of vbst * vbst           */
near float fp_x_last;               /* Last value of vbst * vbst          */
near float fp_cmd_c;                /* Holds the command (c) in fp        */
near float fp_kmax_const;           /* Holds the value of the kmax const  */

            /*** VOLTAGE-LOOP CONTROL VARIABLES ***/
near float fp_vref;                 /* The output voltage reference       */
near float fp_tref;                 /* The output speed reference         */

                /*** MISC TEMP VARIABLES ***/
near float fp_temp1;                /* Holds a temp floating point value  */

        /*** KEY TEMP VARIABLES (ONLY for IRQ ROUTINES) ***/
near unsigned int int_key1;         /* Temp variable for IRQ routines     */
near float fp_key1;                 /* Temp variable for IRQ routines     */

                /*** SPEED CONTROL VARIABLES ***/
near float fp_speed;
near float fp_speed_filt1;          /* Lambda filtered temp for estimator */
near float fp_speed_filt2;          /* Lmabda filtered temp for estimator */
near float fp_tset;                 /* Speed setting for control loop     */
near float fp_tset2;                /* Speed setting for control loop     */

unsigned int speed_cyc_count;       /* Counts cycles for the speed loop   */
unsigned int tstep_dly;             /* Delay for speed step control       */
float fp_pwr_cmd;                   /* Commanded voltage from speed loop  */
float fp_pwr_cmd_last;              /* Last commanded voltage             */
float fp_pwr_cmd_last2;             /* Two-back commanded power           */
float fp_pwr_filt1;                 /* Filtered power for lambda method   */
float fp_pwr_filt2;                 /* Filtered power for lambda method   */
float fp_speed_last;                /* Last value of the bucket temp.     */
float fp_speed_last2;               /* Two-back value of the bucket temp. */
float fp_tn1;                       /* Gain for PP speed controller.      */
float fp_tn2;                       /* Gain for PP speed controller.      */
float fp_tn3;                       /* Gain for PP speed controller.      */
float fp_td1;                       /* Gain for PP speed controller.      */
float fp_td2;                       /* Gain for PP speed controller.      */
float fp_v_cmd;                     /* Holding variable for temp command. */
float fp_A;                         /* Variable for RLS calcs.            */
float fp_r1;                        /* Variable for RLS calcs.            */
float fp_r2;                        /* Variable for RLS calcs.            */
float fp_rc1;                       /* Variable for RLS calcs.            */
float fp_rc2;                       /* Variable for RLS calcs.            */
float fp_t1;                        /* Variable for RLS calcs.            */
float fp_b;                         /* Variable for RLS calcs.            */
float fp_c;                         /* Variable for RLS calcs.            */
float fp_d;                         /* Variable for RLS calcs.            */

                /*** RLS ESTIMATOR VARIABLES ***/
float X[3];                         /* RLS state variables.               */
float K[3];                         /* RLS gain vector.                   */
float theta[3];                     /* RLS parameter estimates.           */
```

```
float P[9];                          /* RLS P matrix.                    */
float P_new[9];                      /* RLS P matrix.                    */
float rls_den;                       /* Denominator used in RLS calc.    */
float rls_error;                     /* Error used in RLS calc.          */

                /*** GENERAL OPERATING VARIABLES ***/
near unsigned char swt_count;        /* IRQ counter for the software timer */
near unsigned char swt_delay;        /* IRQ counter for the software timer */
near unsigned char strobe;           /* IRQ counter for strobe indicator   */
near unsigned char cyc_count;        /* Hold line cycle count in progress  */
near unsigned char line_cycs;        /* Counts the # of cycles between     */
                                     /* software IRQ's                     */
near unsigned int step_dly;          /* This keeps count of the delay used */
                                     /* to implement the step response.    */
near float fp_vset;                  /* Voltage setting for control loop   */
near float fp_vset2;                 /* Voltage setting for control loop   */
near float fp_const_k;               /* K setting for debug mode.          */

near unsigned char mode;             /* This indicates the converter mode. */
near unsigned char menu_mode;        /* This indicates the menu on the LCD */
near unsigned char ctrl_mode;        /* This indicates the control strategy*/
near unsigned char err_num;          /* This indicates the type of error   */
near unsigned char button_id;        /* Stores ID for the start/stop bttns */
near unsigned char button_db;        /* Stores counter to debounce bttns   */
near unsigned char reset_ints;       /* Reset the INT_MASK when high       */
```

## F.3.3. Hi-Tech Compiler Batch Files

### Batch File - HTCC.BAT

```
@echo off
echo Running HITECH C-Compiler on %1.c
echo 1st pass...
c:\hitech\cpp.exe -DHI_TECH_C -Di80196 -IC:\hitech\ %1.c %1.cpp
echo 2nd pass...
c:\hitech\p1.exe %1.cpp %1.p1
echo generating assembly code...
c:\hitech\cg96.exe %1.p1 %1.cg
echo optimizing code...
c:\hitech\opt96.exe %1.cg %1.as
echo assembling...
c:\hitech\as96.exe -o%1.obj %1.as
if NOT EXIST %1.as goto errmsg
del %1.cpp
del %1.p1
del %1.cg
rem del %1.as
goto end
:errmsg
echo Error - %1.as does not exist!
:end
```

### Batch File - HTLINK.BAT

```
@echo off
echo -z -M%1.map \> link.dat
echo -ptext=02080h,data,bss=08000h,stack=0f800h,creg=01ch,regbss=03ah \>> link.dat
echo -o1.obj c:\hitech\dkjstart.obj c:\hitech\dkjrom.obj %1.obj \>> link.dat
echo c:\hitech\96libc.lib c:\hitech\96libf.lib >> link.dat

echo linking...
link < link.dat
echo generating HEX...
objtohex.exe -f 1.obj %1.hex
echo generating BIN...
hexbin.exe %1.hex %1.bin I

@echo off
@del link.dat
@del 1.obj
```

## Batch File - MAKEROM.BAT

```
@echo off
call htcc %1
if NOT EXIST %1.obj goto errmsg
call htlink %1
goto end
:errmsg
echo Compiler Error - %1.obj does not exist!
:end
```

## F.3.4. MathCAD Spreadsheet for #DEFINE Constants

### MathCAD Spreadsheet - BID_CAL.MCD

#### Enter Calibrated Hardware Values Here

| Boost Converter | DC/DC Converters | A/D Converter |
|---|---|---|
| rms $Vac_{rms} := 298.44$ | $V1_{pk} := 400.0$ | $AD_{range} := 1023$ |
| rms $Iac_{rms} := 21.8472$ | $I1_{pk} := 9.52$ | $f_s := 100 \cdot 10^3$ |
| $Vbst_{pk} := 399.5833$ | $V2_{pk} := 403.476$ | $L_{leak} := 25.3 \cdot 10^{-6}$ |
| $Vbst_{ofst} := 250.0$ | $I2_{pk} := 18.6793$ | $RSERIES\_FP := 2 \cdot \pi \cdot f_s \cdot L_{leak}$ |

#### Enter Desired Limits Here

| AC-Side Limits | DC-Side Limits | Menu Limits |
|---|---|---|
| rms VAC_HIGH_FP := 150 | VBST_HIGH_FP := 390 | CUR_MIN_FP := 0 |
| rms VAC_LOW_FP := 90 | IBST_HIGH_FP := 4 | CUR_STEP_FP := 0.1 |
| rms IAC_HIGH_FP := 8 | | CUR_MAX_FP := 3 |
| rms $IAC_{max} := 5.0$ | | |

#### A/D Conversion Factors

$$VBST\_TO\_FP := \frac{Vbst_{pk} - Vbst_{ofst}}{AD_{range}} \qquad VAC\_TO\_FP := \frac{Vac_{rms}}{AD_{range}}$$

$$DC\_V1\_TO\_FP := \frac{V1_{pk}}{AD_{range}} \qquad IAC\_TO\_FP := \frac{Iac_{rms}}{AD_{range}}$$

$$DC\_I1\_TO\_FP := \frac{I1_{pk}}{AD_{range}} \qquad DC\_V2\_TO\_FP := \frac{V2_{pk}}{AD_{range}}$$

$$DC\_I2\_TO\_FP := \frac{I2_{pk}}{AD_{range}} \qquad VBST\_OFFSET\_FP := Vbst_{ofst}$$

#### Enter Measurement Gains

$$G_{vac} := \frac{10}{475} \qquad G_{iac} := \frac{400}{1000} \qquad G_{icmd} := 3.5 \qquad MAX\_ICMD := 4095$$

#### KMAX Constant

$$KMAX\_CONST\_FP := \frac{IAC_{max} \cdot G_{iac} \cdot MAX\_ICMD}{G_{vac} \cdot G_{icmd}}$$

## Boost Current-Loop Calculations

SET THE CURRENT-LOOP POLES HERE ==> poles := 0.85

$G_1 := 2 - 2 \cdot poles \qquad G_2 := \frac{G_1^2}{4} - G_1 \qquad C := \left(470 \cdot 10^{-6}\right) \cdot 3 \qquad T_L := \frac{1}{120}$

DIGITAL GAINS

$$H1\_FP := \frac{G_1 \cdot C \cdot MAX\_ICMD \cdot G_{iac}}{2 \cdot T_L \cdot G_{icmd} \cdot G_{vac}} \qquad H1\_FP = 564.20$$

$$H2\_FP := \frac{G_2 \cdot C \cdot MAX\_ICMD \cdot G_{iac}}{2 \cdot T_L \cdot G_{icmd} \cdot G_{vac}} \qquad H2\_FP = -521.88$$

$$HP\_FP := \frac{MAX\_ICMD \cdot G_{iac}}{G_{icmd} \cdot G_{vac}} \qquad HP\_FP = 22230.0$$

---

## Buck Current-Loop Calculations

SET THE CURRENT-LOOP POLE HERE ==> pole := 0.90

$$alpha := \frac{G_{iac} \cdot MAX\_ICMD}{G_{vac} \cdot G_{icmd}} \qquad BUCK\_GAIN\_FP := \frac{alpha \cdot (1 - pole)}{pole}$$

---

## #DEFINE Constants

RSERIES_FP = 15.90

H1_FP = $5.64197 \cdot 10^{2}$

H2_FP = $-5.21883 \cdot 10^{2}$

HP_FP = $2.22300 \cdot 10^{4}$

VBST_TO_FP = $1.46201 \cdot 10^{-1}$

VAC_TO_FP = $2.91730 \cdot 10^{-1}$

IAC_TO_FP = $2.13560 \cdot 10^{-2}$

DC_V1_TO_FP = $3.91007 \cdot 10^{-1}$

DC_I1_TO_FP = $9.30596 \cdot 10^{-3}$

DC_V2_TO_FP = $3.94405 \cdot 10^{-1}$

DC_I2_TO_FP = $1.82593 \cdot 10^{-2}$

VBST_HIGH_FP = 390.0

IBST_HIGH_FP = 4.0

IAC_HIGH_FP = 8.0

VAC_HIGH_FP = 150.0

VAC_LOW_FP = 90.0

VBST_OFFSET_FP = 250.0

CUR_MIN_FP = 0.0

CUR_STEP_FP = 0.1

CUR_MAX_FP = 3.0

BUCK_GAIN_FP = $2.47000 \cdot 10^{3}$

KMAX_CONST_FP = $1.11150 \cdot 10^{5}$

# Bibliography


[1] Abiomed, Inc., *Total Artificial Heart*, Cerry Hill Drive, Danvers, MA 01923.

[2] Advantech, Genie Data Acquisition Systems, 750 East Arques Avenue, Sunnyvale, CA 94086.

[3] P. Agarwal, "Eddy-Current Losses in Solid and Laminated Iron," *AIEE Transactions*, Vol. 78, 1959, pp. 169–179.

[4] J. Appelbaum and R. Weiss, "An Electrical Model of the Lead-Acid Battery," *International Telecommunications Energy Conference*, October 1982, pp. 304–307.

[5] K. Åström and B. Wittenmark, *Computer Controlled Systems*, Prentice-Hall, Inc., 1984.

[6] J. Barnard, J. Ferreira, and J. van Wyk, "Optimising Sliding Transformers for Contactless Power Transmission Systems," *IEEE Power Electronics Specialists Conference*, June 1995, pp. 245–251.

[7] J. Bolger, C. Haslund, and R. Risser, "Inductive Charging of Electric Vehicles: Testing and Evaluation of an Automated System," *11th International Electric Vehicle Symposium*, September 1992, pp. 1–12.

[8] R. Bozorth, *Ferromagnetism*, IEEE Press, Piscataway, NJ, 1993.

[9] D. Brock, W. Lee, D. Segalman, and W. Witkowski, "A Dynamic Model of a Linear Actuator based on Polymer Hydrogel," *Journal of Intelligent Materials and Structures*, Vol. 5, No. 6, November 1994, pp. 764–771.

[10] Coherent Power, Ltd., *HFXC-5000 Onboard Battery Charger for Electric Vehicles*, 768 Brittain Lane, Santa Rosa, CA 95407.

[11] D. Collins, S. Hinchliffe, and L. Hobson, "Computer Control of a Class-E Amplifier," *International Journal of Electronics*, Vol. 64, No. 3, March 1988, pp. 493–506.

[12] N. Cox, "A Universal Power Converter for Emergency Charging of Electric Vehicle Batteries," *IEEE Applied Power Electronics Conference*, March 1995, pp. 965–969.

[13] E. Dahl, "Induction Charging System," *U.S. Patent #3,938,018*, February 10, 1976.

[14] E. Dede, V. Esteve, J. Garcia, A. Navarro, and J. Carrasco, "On the Design of High Frequency Series Resonant Converters for Induction Heating Applications," *IEEE International Conference on Industrial Electronics*, November 1993, pp. 1303–1307.

[15] Delco Propulsion Systems, "Magne Charge Resonant Design Inductive Chargers," *Product Brochure*, 7901 E. 88th Street, Suite 200, Indianapolis, IN 46256.

[16] J. Dickson, "Design and Testing of a Ball and Socket Type Joint for Use with Polymer Gel Muscles," *B.S. Thesis*, Massachusetts Institute of Technology, May 1996.

[17] R. Dorf, *Modern Control Systems*, Addison-Wesley, Reading, MA, 1990.

[18] P. Eliams and A. Mansell, "Simulated Annealing in the Analysis of Resonant DC Link Inverters," *IEE Proceedings on Electric Power Applications*, Vol. 141, No. 3, pp. 163–168.

[19] G. Elliott, J. Boys, and A. Green, "Magnetically Coupled Systems for Power Transfer to Electric Vehicles," *Proceedings of the 1995 International Conference on Power Electronics and Drive Systems*, February 1995, pp. 797–801.

[20] P. Enjeti, P. Ziogas, and J. Lindsay, "Programmed PWM Techniques to Eliminate Harmonics: A Critical Evaluation," *IEEE Transactions on Industry Applications*, Vol. 26, No. 2, March/April 1990, pp. 302–316.

[21] R. Erickson, *Fundamentals of Power Electronics*, Chapman & Hall, New York, 1997.

[22] A. Esser and H. Skudelny, "A New Approach to Power Supplies for Robots," *IEEE Transactions on Industry Applications*, Vol. 27, No. 5, September 1991, pp. 872–875.

[23] A. Esser, "Contactless Charging and Communication System for Electric Vehicles," *IEEE Industry Applications Magazine*, November/December 1995, pp. 4–11.

[24] Everest Interscience Inc., 1625 West Ina Road, Suite 123, Tucson, AZ 85704.

[25] Ferrofluidics Corporation, 40 Simon Street, Nashua, NH, 03061.

[26] A. Fitzgerald, C. Kingsley, Jr., S. Umans, *Electric Machinery*, McGraw-Hill, Inc., New York, NY, 1990.

[27] A. Ghahary and B. Cho, "Design of a Transcutaneous Energy Transmission System Using a Series Resonant Converter," *IEEE Transactions on Power Electronics*, Vol. 7, No. 2, April 1992, pp. 261–269.

[28] D. Giuliani, R. McMahon, and D. Engel, "Pacing Toothbrush," *U.S. Patent #5,544,382*, August 13, 1996.

[29] G. Goodwin and K. Sin, *Adaptive Filtering Prediction and Control*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1984.

[30] A. Green and J. Boys, "10kHz Inductively Coupled Power Transfer - Concept and Control," *IEE Conference on Power Electronics and Variable-Speed Drives*, October 1994, pp. 694–699.

[31] S. Haider, T. Cetas, J. Wait, and J. Chen, "Power Absorption in Ferromagnetic Implants from Radio Frequency Magnetic Fields and the Problem of Optimization," *IEEE Transactions on Microwave Theory and Techniques*, Vol. 39, No. 11, 1991, pp 1817–1827.

[32] H. Haus and J. Melcher, *Electromagnetic Fields and Energy*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1989.

[33] J. Hayes, S. Schultz, J. Hall, M. Egan, and J. Murphy, "Wide Load Range Resonant Converter Supplying the SAE J-1773 Inductive Charging Interface," *IEEE Industry Applications Society Annual Meeting*, October 1996, pp. 1065–1071.

[34] B. Heeres, D. Novotny, D. Divan, and R. Lorenz, "Contactless Underwater Power Delivery," *IEEE Power Electronics Specialist Conference*, June 1994, pp. 418–423.

[35] S. Hinchliffe and L. Hobson, "High Efficiency DC-AC Converters Suitable for High Frequency Induction Process Heating," *IEEE Power Electronics Specialists Conference*, April 1988, pp. 1228–1235.

[36] D. Hind, "Inductively Coupled Battery Charging System," *Seventh Annual Battery Conference*, April 1992. pp. 1–13.

[37] Y. Hiraga, J. Hirai, A. Kawamura, K. Ishioka, Y. Kaku, and Y. Nitta, "Decentralized Control of Machines with the Use of Inductive Transmission of Power and Signal," *IEEE Industry Applications Society Annual Meeting*, January 1994, pp. 875–881.

[38] S. Hirotsu, Y. Hirokawa, and T. Tanaka, "Volume-Phase Transitions of Ionized N-isopropylacrylamide Gels," *Journal of Chemical Physics*, Vol. 87, No. 2, July 1987, pp. 1392–1395.

[39] Hi-Tech Software LLC, Suite 105, 7830 Ellis Road, Melbourne, FL 39204

[40] IEEE Magnetics Society, *IEEE Standard for Pulse Transformers*, 1987.

[41] Illinois Capacitor, Inc., *Illinois Capacitor Engineering Guide*, 1995, Lincolnwood, IL.

[42] P. Imbertson and N. Mohan, "Asymmetrical Duty Cycle Permits Zero Switching Loss in PWM Circuits with No Conduction Loss Penalty," *IEEE Transactions on Industry Applications*, Vol. 29, No. 1, Jun./Feb. 1993, pp. 121–125.

[43] Intel Corporation, *EV80C196KC Microcontroller Evaluation Board User's Manual*, 1992.

[44] D. Jackson, S. Leeb, A. Mitwalli, P. Narvaez, D. Fusco, and E. Lupton, "Power Electronic Drives for Magnetically Triggered Gels," *IEEE Transactions on Industrial Electronics*, April 1997, pp. 217–225.

[45] D. Jackson, S. Leeb, A. Mitwalli, P. Narvaez, D. Fusco, and E. Lupton, "Power Electronic Drives for Magnetically Triggered Phase Transition Gels," *IEEE Power Electronics Specialists Conference*, June 1996, pp. 302–309.

[46] D. Jackson, A. Schultz, S. Leeb, A. Mitwalli, G. Verghese, S. Shaw, "A Multirate Digital Controller for a 1.5-kW Electric Vehicle Battery Charger," *IEEE Transactions on Power Electronics*, Vol. 12, No. 6, November 1997, pp. 1000–1006.

[47] D. Jackson, S. Leeb, A. Schultz, and A. Mitwalli, "A Comparison of Multirate Digital Compensators for a Battery Charger," *IEEE 5th Workshop on Computers in Power Electronics*, August 1996, pp. 58–65.

[48] R. Johansson, *System Modeling and Identification*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1993.

[49] J. Kassakian, M. Schlecht, and G. Verghese, *Principles of Power Electronics*, Addison-Wesley, Reading, MA, 1991.

[50] P. Kassakian, "Design and Testing of a Mechanical Arm for Use with Polymer Gel Muscles," *B.S. Thesis*, Massachusetts Institute of Technology, May 1995.

[51] M. Kazimierczuk and D. Czarkowski, *Resonant Power Converters*, John Wiley & Sons, Inc., New York, NY, 1995.

[52] M. Kazimierczuk and K. Puczko, "Class E Tuned Power Amplifier with Antiparallel Diode or Series Diode at Switch, with Any Loaded Q and Switch Duty Cycle," *IEEE Transactions on Circuits and Systems*, Vol. 36, No. 9, September 1989, pp. 1201–1209.

[53] A. Kawamura, K. Ishioka, and J. Hirai, "Wireless Transmission of Power and Information Through One High-Frequency Resonant AC Link Inverter for Robot Manipulator Applications," *IEEE Transactions on Industry Applications*, Vol. 32, No. 3, May 1996, pp. 503–508.

[54] A. Kelley and W. Owens, "Connectorless Power Supply for an Aircraft-Passenger Entertainment System," *IEEE Transactions on Power Electronics*, Vol. 4, No.3, July 1989, pp. 348–354.

[55] I. Kimura and T. Katsuki, "VLF Induction Heating for Clinical Hyperthermia," *IEEE Transactions on Magnetics*, Vol. MAG-22, No. 6, November 1986, pp. 1897–1900.

[56] K. Klontz, D. Divan, D. Novotny, and R. Lorenz, "Contactless Battery Charging System," *U.S. Patent #5,157,319*, October 1992.

[57] K. Klontz, D. Divan, D. Novotny, and R. Lorenz, "Contactless Power Delivery System for Mining Applications," *IEEE Industry Applications Society Annual Meeting*, October 1991, pp. 1263–1269.

[58] K. Klontz, A. Esser, P. Wolfs, and D. Divan, "Converter Selection for Electric Vehicle Charger Systems with a High-Frequency High-Power Link," *IEEE Power Electronics Specialists Conference*, June 1993, pp. 855–861.

[59] N. Kutkut, D. Divan, D. Novotny, and R. Marion, "Design Considerations and Topology Selection for a 120 kW IGBT Converter for EV Fast Charging," *IEEE Power Electronics Specialists Conference*, June 1995, pp. 238–244.

[60] N. Kutkut, K. Klontz, "Design Considerations for Power Converters Supplying the SAE J-1773 Electric Vehicle Inductive Coupler," *IEEE Applied Power Electronics Conference*, February 1997, pp. 841–847.

[61] Laboratory of Integrated Systems, University of São Paolo, Brazil.

[62] D. Lancaster, "Tech Musings", *Electronics Now*, Vol. 66, No. 5, May 1995.

[63] T. Liang, R. O'Connell, and R. Hoft, "Inverter Harmonic Reduction Using Walsh Function Harmonic Elimination Method," *IEEE Transactions on Power Electronics*, Vol. 12, No. 6, November 1997, pp. 971–982.

[64] D. Linden, *Handbook of Batteries - Second Edition*, McGraw-Hill, Inc., New York, NY, 1995.

[65] Linear Technology, "LTC 1043 Data Sheet," *Linear Databook*, 1990, pp. 11-5–11-30.

[66] Magnetics, *Ferrite Core Databook*, 1995, Butler, PA.

[67] K. Mahabir, G. Verghese, V. Thottuvelil, and A. Heyman, "Linear Averaged and Sampled Data Models for Large Signal Control of High Power Factor AC-DC Converters," *IEEE Power Electronics Specialists Conference*, June 1990, pp. 291–299.

[68] Master Publishing, Inc., *Enercell Battery Guidebook*, 14 Canyon Creek Village, MS31, Richardson, TX 75280.

[69] Mathworks, *MATLAB Student Version 5.0*, 24 Prime Park Way, Natick, MA 01760.

[70] N. Mohan, T. Undeland, and W. Robbins, *Power Electronics, Converters, Applications, and Design*, John Wiley & Sons, Inc., New York, NY, 1995.

[71] H. Martin Jr., "Topology for Miniature Power Supply with Low Voltage and Low Ripple Requirements," *U.S. Patent #4,618,919*, October 1986.

[72] R. Martinez and P. Enjeti, "A High-Performance Single-Phase Rectifier with Input Power Factor Correction," *IEEE Transactions on Power Electronics*, Vol. 11, No. 2, March 1996, pp. 311–317.

[73] H. Matsuki, K. Murakami, T. Satoh, and T. Hoshino, "An optimum Design of a Soft Heating System for Local Hyperthermia," *IEEE Transactions on Magnetics*, Vol. MAG-23, No. 5, September 1987, pp. 2440–2442.

[74] H. Matsuki, M. Shiiki, K. Murakami, K. Nadehara, and T. Yamamoto, "Flexible Transcutaneous Transformer for Artificial Heart System," *IEEE Transactions on Magnetics*, Vol. 26, No. 5, September 1990, pp. 1548–1550.

[75] A. Mitwalli, *Ph. D. Thesis*, Massachusetts Institute of Technology, expected May 1998.

[76] A. Mitwalli, T. Denison, D. Jackson, and S. Leeb, "Closed-Loop Feedback Control of Magnetically-Activated Gels," *Journal of Intelligent Material Systems and Structures*, Vol. 8, No. 7, July 1997, pp. 596–604.

[77] A. Mitwalli, S. Leeb, T. Tanaka, and U. Sinha, "Polymer Gel Actuators – Status Report," *Universities Power Engineering Conference*, September 1994, pp. 871–874.

[78] A. Mitwalli, S. Leeb, G. Verghese, and V. Thottuvelil, "An Adaptive Digital Controller for a Unity Power Factor Converter," *IEEE Transactions on Power Electronics*, Vol. 11, No. 2, March 1996, pp. 374–382.

[79] B. Miwa, "Interleaved Conversion Techniques for High Density Power Supplies," *Ph. D. Thesis*, Massachusetts Institute of Technology, May 1992.

[80] B. Miwa, D. Otten, and M. Schlecht, "High Efficiency Power Factor Correction Using Interleaving Techniques," *IEEE Applied Power Electronics Conference*, February 1992, pp. 557–568.

[81] L. Mweene, "The Design of Front-End DC-DC Converters of Distributed Power Supply Systems with Improved Efficiency and Stability," *Ph. D. Thesis*, Massachusetts Institute of Technology, September 1992.

[82] L. Mweene, D. Otten, and M. Schlecht, "A High-Efficiency 1.5 kW, 390-50 V Half-Bridge Converter Operated at 100% Duty-Ratio," *IEEE Applied Power Electronics Conference*, February 1992, pp. 723–730.

[83] L. Mweene, C. Wright, and M. Schlecht, "A 1 kW, 500 kHz Front-End Converter for a Distributed Power Supply System," *IEEE Applied Power Electronics Conference*, March 1989, pp. 423–432.

[84] Novamet Specialty Products Corporation, 10 Lawlins Park, Wyckoff, NJ 07481.

[85] H. Patel and R. Hoft, "Generalized Technique of Harmonics Elimination and Voltage Control in Thyristor Inverters: Part I Harmonic Elimination," *IEEE Transactions on Industry Applications*, Vol. IA-9, No. 3, May/June 1973, pp. 310–317.

[86] O. Patterson and D. Divan, "Pseudo-Resonant Full Bridge DC/DC Converter," *IEEE Power Electronics Specialists Conference*, June 1987, pp. 424–430.

[87] F. Peng, H. Akagi, A. Nabae, and S. Sugawara, "High-Frequency Current-Source Inverters Using SI Thyristors for Induction Heating Applications," *IEEE Transactions on Industry Applications*, Vol. 25, No. 1, January 1989, pp. 172–180.

[88] F. Raab, "Idealized Operation of the Class E Tuned Power Amplifier," *IEEE Transactions on Circuits and Systems*, Vol. CAS-24, No. 12, 1977, pp. 725–735.

[89] S. Ramo, J. Whinnery, and T. Van Duzer, *Fields and Waves in Communication Electronics*, John-Wiley & Sons, Inc., New York, NY, 1984.

[90] Raychem Corporation, "Induction Heating of Loaded Materials," *U.S. Patent #5,378,879*, January 1995.

[91] S. Rhodes, "The Effects of Separable Cores on High Power Transformer Design," *M. Eng. Thesis*, Massachusetts Institute of Technology, August 1996.

[92] L. Roszyk and L. Barnas, "Hand Held Battery Operated Device and Charging Means Therefor," *U.S. Patent #3,840,795*, October 1974.

[93] F. Sato, J. Murakami, H. Matsuki, S. Kikuchi, K. Harakawa, and T. Satoh, "Stable Energy Transmission to Moving Loads Utilizing New CLPS," *IEEE Transactions on Magnetics*, Vol. 32, No. 5, September 1996, pp. 5034–5036.

[94] A. Schultz, S. Leeb, A. Mitwalli, D. Jackson, G. Verghese, S. Shaw, "A Multirate Digital Controller for an Electric Vehicle Battery Charger," *IEEE Power Electronics Specialists Conference*, June 1996, pp. 1919–1925.

[95] S. Shaw, "High Bandwidth Connectorless Communication in High Frequency Magnetics," *Independent Laboratory Project*, Massachusetts Institute of Technology, January 1994.

[96] S. Shaw, D. Jackson, T. Denison, and S. Leeb, "Computer-Aided Design and Application of Sinusoidal Switching Patterns," *IEEE 7th Workshop on Computers in Power Electronics*, August 1998.

[97] W. Siebert, *Circuits, Signals, and Systems*, McGraw-Hill, Inc., New York, NY, 1986.

[98] Society of Automotive Engineering, *SAE J-1772 Electric Vehicle Conductive Coupling Recommended Practice - DRAFT Version 3.3*, Warrendale, PA, October 1994.

[99] Society of Automotive Engineering, *SAE J-1773 Electric Vehicle Inductive Coupling Recommended Practice*, Warrendale, PA, February 1995.

[100]N. Sokal and A. Sokal, "Class E - A New Class of High Efficiency Tuned Single-Ended Switching Power Amplifier," *IEEE Journal of Solid-State Circuits*, Vol. SC-13, No. 3, 1975, pp. 168–176.

[101]F. Storm, R. Elliott, W. Harrison, and D. Morton, "Clinical RF Hyperthermia by Magnetic-Loop Induction: A New Approach to Human Cancer Therapy," *IEEE Transactions on Microwave Theory and Techniques*, Vol. MTT-30, No. 8, August 1982, pp. 1149–1157.

[102]A. Suzuki and T. Tanaka, "Phase Transition in Polymer Gels Induced by Visible Light," *Nature*, Vol. 346, No. 6282, July 1990, pp. 345–347.

[103]T. Tanaka, "Gels," *Scientific American*, Vol. 244, No. 1, January 1981, pp. 124–138.

[104]T. Tanaka, "Kinetics of Phase Transition in Polymer Gels," *Physica A*, Vol. 140A, No. 1-2, pp. 261–268.

[105]T. Tanaka, I. Nishio, S. Sun, and S. Ueno-Nisho, "Collapse of Gels in an Electric Field," *Science*, Vol. 218, No. 29, October 1982, pp. 467–469.

[106]TDK, *Ferrite Core Selection Guide*, 1995, Tokyo, Japan.

[107]D.Temkin, M. McVey, and U. Carlsson, "A Spacecraft Electrical Battery Simulator," *Intersociety Energy Conversion Engineering Conference*, August 1990, pp. 19–26.

[108]M. Thompson, "Magnetically-Induced Heating Effects in Ferromagnetic Fluids," *EECS Department Area Examination Report*, Massachusetts Institute of Technology, 1996.

[109]Unitrode Integrated Circuits, "UC3854 Controlled Power Factor Correction Circuit Design," *Application Note U-134*, 1993.

[110]Unitrode Integrated Circuits, "Phase-Shifted, Zero Voltage Transition Design Considerations and the UC3875 PWM Controller", *Application Note U-136*, 1993.

[111]Unitrode Integrated Circuits, "Average Current Model Control of Switching Power Supplies," *Application Note U-140*, 1993.

[112]P. Wellstead and M. Zarrop, *Self-Tuning Systems*, John Wiley & Sons, Inc., New York, NY, 1991.

[113]T. Wilson, "The evolution of power electronics," *IEEE International Symposium on Industrial Electronics*, May 1992, pp. 1–9.

[114]M. Zahn, "Power Dissipation and Magnetic Forces on MAGLEV Rebars," *IEEE Transactions on Magnetics*, Vol. 33, No. 2, March 1997, pp. 1021–1036.