Power **I**nformation **C**ollection **A**rchitecture

Final Design Report

Engineering 340: Senior Design 2010-2011

11 April 2011

Team 01: A. Ball, K. Wiersma, N. Jen, A. Sterk

Abstract

The PICA system addresses several problems with power metering through the use of two metering systems, the E meter and smart breakers, and one data collection center, the base station. The E meter measures power qualities relevant to the power companies and uses an MSPGCC4 processor as the main component. The smart breakers measure circuit-by-circuit information relevant to the homeowner in addition to solid state relays to function in place of the traditional electromechanical breakers. The ADE776 metering chip and ATmega328 microprocessor provide most of the functionality of these smart breakers. The base station's full functionality using the LEON3 processor and custom Linux system could not be completed in time for the final project, so a standard desktop computer will perform this function. The team also designed a switch-mode power supply to power the metering devices.

This report explains the design and design choices for the system and subsystems, a business case for the project and recommendations for future work on the project.

Summary of conclusion goes here.

Contents

List of Figures

List of Tables

Listings

1 Acronyms

AC Alternating Current 20
ADC Analog-to-Digital Converter 29
AES Advanced Encryption Standard 19
ANSI American National Standards Institute 17
CAD Computer Aided Design 28
CCS4 Code Composer Studio 4 27
DC Direct Current 13
DHCP Dynamic Host Configuration Protocol 10
DIP Dual Inline Package 31
EM electromagnetic 14
FCC Federal Communications Commission 14
FPGA Field Programable Gate Array 3
FPU Floating Point Unit 55
GCC GNU Compiler Collection 56

GNU GNU's not Unix
GPL GNU General Public License 53
HTTP Hypertext Transfer Protocol 11
IC Integrated Circuit 31
IEC International Electrotechnical Commision 82
ISR Interrupt Service Routine 35
JCI Johnson Controls, Inc. 38
JTAG Joint Test Area Group 6
LAN Local Area Network 8
LCD Liquid Crystal Display 17
LED Light Emitting Diode 9
MCU Master Control Unit 14
NEMA National Electrical Manufacturers Association 82
NILM Non-intrusive Load Monitoring 83
NPC National Power Corporation 15
NTP Network Time Protocol 8

2 Introduction

2.1 About Us

Amy Ball: Amy works as an intern at Johnson Controls, where she works as part of the Systems Engineering Team. She brings good communication skills, circuit-building experience, and presentation skills to the project. Originally her section was the solid-state breakers, but the team decided to use solid-state relays for the breaker purposes. After which her new task was to build all of the power supplies for the project as well as an idea of how to identify which circuit is in use at certain times. She is also working closely with much of the analog hardware involved with the project.

Kendrick Wiersma: Kendrick works as an intern at Raytheon Missile Systems in the Electronics Center, where he performs embedded system design and verification. Kendrick hails from Tucson, Arizona where he was born and raised. He brings real-world project experience and experience working with embedded hardware and software to the team. Kendrick leads the development of the E-meter, which measures whole-building power consumption, reporting data to the power company and the PICA base station.

Nathan Jen: Nathan has worked at Amway on the production floor and has gained involvement with club leadership at Calvin College. He brings leadership experience and a good understanding of how smaller elements of a system fit together as a whole. His section of the project is the monitoring of individual circuits and some of the control logic for the breakers.

Avery Sterk: Avery worked as an intern at the SLAC National Accelerator Laboratory doing CAD design. He brings varied experience with software design and implementation to the project. His section of the project is the base station, especially providing the primary user interface and designing embedded software.

2.2 About the Course

This course takes place over two semesters in the senior design project sequence. For the first semester course (Engineering 339), emphasis is placed on design team formation, project identification, and production of a feasibility study. Students focus on the development of task specifications in light of the norms for design and preliminary validation of the design by means of basic analysis and appropriate prototyping. Lectures focus on integration of the design process with a reformed Christian worldview, team building, and state-of-the-art technical aspects of design. Interdisciplinary projects are encouraged.

Prerequisites: Concurrent registration in the seventh semester of the model program for a particular concentration or permission of the instructors; developing a Christian mind and philosophical foundations. For the second semester in the senior design project sequence (Engineering 340), emphasis is placed on the completion of a major design project initiated in Engineering 339. This project should entail task specifications in light of the norms for design by means of engineering analysis and an appropriate prototype focused on primary functionality. A final presentation is given at the May Senior Design Night Banquet. Lectures continue to focus on integration of the design process with a Christian reformed world-view, team activity, and state-of-the art technical aspects of design.

3 System Overview

3.1 Problem Definition

Standard electric meters were developed decades ago and are still used today, despite many technological advances in the last several years. Along with these technological advances, Americans have become accustomed to having access to large amounts of data, but due to the nature of the standard electric meter, data regarding the usage of power is severely limited. For the power companies, data from the meters is minimal and grid control is limited to manual operation, costing them time and money. As the cost of electricity becomes higher and higher, electricity use in buildings is becoming a bigger concern and people have few cheap or simple ways to monitor this consumption. Of the options available, most only address part of the whole problem, giving some information to the consumer and none to the power company or vice-versa. While there are devices such as breakers and fuses that provide electrical safety for buildings, advances in technology have made it possible to further improve safety but have not been implemented in a cost-effective way.

3.2 Customer Base

The target market for the entire PICA system comprises both electricity producers and electricity consumers, as set forth by the nature of the subsystems. As the power companies supply and own the electricity meters attached to the buildings to which they supply power, the PICA E-meter appeals only to the market of electricity-producing companies. The other two subsystems, the solid-state breakers and base station, target the power-consuming audience, as the devices will assist in monitoring power flow inside the building, where the power company has no presence. As these two markets are essentially exclusive in both membership and interest in the PICA subsystems, the E-meter will be able to function independently of the other consumer-targeted subsystems, and vice versa.

3.2.1 Power Companies

As power companies currently distribute the whole-building metering hardware that determines how much energy their customer used, the E-meter clearly targets power companies. In fact, the power companies own the power-measuring hardware external to the buildings to which they provide power, so only they may replace or upgrade those devices. At present, power companies send trained meter-readers to read the data

from most traditional power meters under their control. The PICA E-meter subsystem aims to improve on this process by automatically sending the measurements to the power company using a means and protocol selected by the particular company. While this will require some hardware customization for each company, the volume of company-specific production should allow the cost to develop the design to spread into a small per-unit cost.

The PICA E-meter subsystem also provides numerous more measures of power than the simple spinning-dial meters. For example, the E-meter will measure the frequency and the RMS voltage of the incoming supply lines, which help indicate the overall quality of power delivered to the customers in the area. This information may also help diagnose any observed issues with power delivery without dispatching a worker to take measurements by hand. In this way, power companies using the PICA E-meter can improve the quality of the service they provide and can save on the labor costs associated with making a site visit.

3.2.2 Power Consumers

Although the power company's customers cannot modify the metering panel installed by the power company, they are free to modify the other power distribution components inside their own buildings. The solid-state breakers fall into this category, and provide previously unavailable measurements regarding power consumption and its location within the building. However, as these breakers will replace the pre-existing breakers inside the building, the consumer must be convinced that using the PICA system is worth the trouble and cost of replacing the mechanical circuit breakers with the more feature-rich PICA breakers. To this effect, the most receptive market for the solid-state breakers includes homeowners and building managers who are curious or concerned about power usage inside their building. That is, the people for whom this information can inspire a meaningful change in practice will likely become the first adopters of the subsystem.

The product may also gain a following as an alternative to mechanical breakers in during the construction of a new home or building. This would likely require that the product already have a proven history of reliability and safety, so the previous group of cost- or environmentally-concerned individuals might have to adopt the produce first. If the PICA solid-state breakers become an alternative during construction, the net cost to the user will be lessened, as the building-to-be will not have any pre-existing breakers to discard or replace.

The base station may apply to either of these two consumer groups, as its primary purpose is to manage and interface with the other systems. It does not specifically require the solid-state breakers or the E-meter, but provides little value in a building without any installed PICA systems. The base station exists solely to manage and collect data from other PICA subsystems, as well as format and display these measurements, so its target audience consists of power consumers whose buildings use the E-meter, the solid-state breakers, or both.

3.3 System Requirements – TBD

To be included...

3.4 System Block Diagram

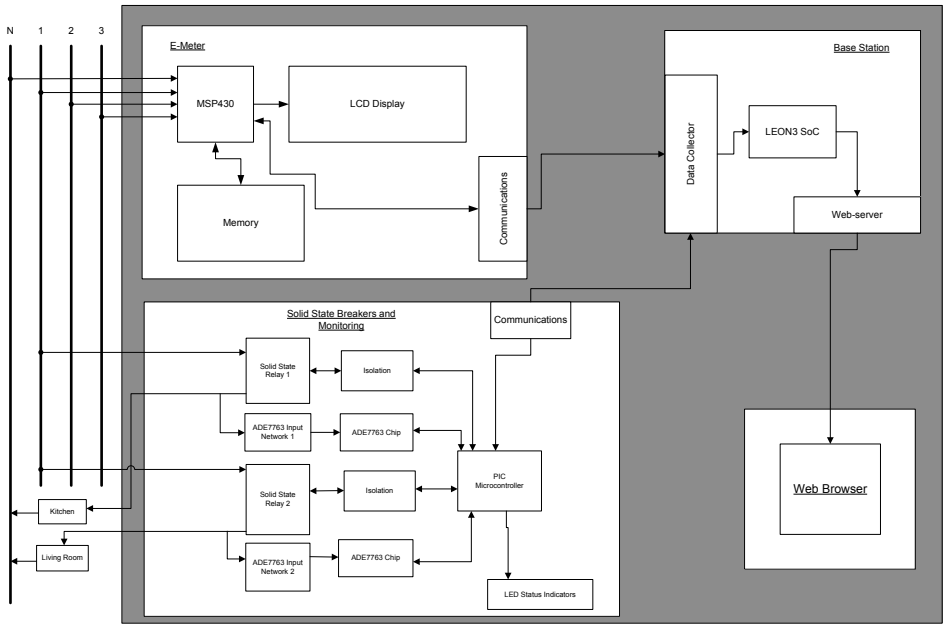

Figure 1: System block diagram

3.5 System Explanation – TBD

3.6 Testing – TBD

3.7 Future Work – TBD

4 Requirements

4.1 Base Station Requirements

All requirements under this heading are to be assumed to be of the base station (“the system”) unless explicitly stated otherwise.

4.1.1 Functional Requirements

1. Shall be capable of upgrading its software and firmware upon administrator requests.
2. Shall be capable of connecting with other PICA sub-systems over a pre-defined and pre-arranged communications protocol.
    (a) Shall receive and store power usage measurements from connected PICA systems.
        i. These measurements will be taken at a frequency that does not exceed the established bandwidth of the chosen protocol.
        ii. These measurements will be summarized or discarded (at the user’s selection) when the storage space of the system nears its capacity.
    (b) Shall receive and store events and alerts from connected PICA systems.
        i. The nature of these events shall be determined by each individual system, but shall be communicated to the base station in a standardized format.
        ii. The system shall organize these events internally and display them to the user.
    (c) Shall function as a Network Time Protocol (NTP) server for connected PICA systems.
    (d) Shall receive and record event log information from connected PICA systems.
3. Shall be capable of connecting to a Local Area Network (LAN).
4. Shall be capable of using settings that the user selects.
    (a) Shall be capable of recognizing an invalid setting.
    (b) Shall be capable of reverting to a default setting when the user setting is not valid.
5. Shall be capable of displaying the most recent measurements from the connected PICA systems.

6. Shall be capable of displaying the status of the connected PICA systems.
    (a) Status shall include whether the connected system is in an error state.
    (b) Status shall include the time since the last observed error state, if applicable.
    (c) Status shall include the nature of the last error, if communicated.
7. Shall be capable of giving an authenticated base-station adminstrator similarly-privileged administrative access to connected PICA systems.
    (a) The extent of this access shall be deteremined by the individual PICA systems.
    (b) The base station shall inform the administrator when the remote administrative access is rejected or unavailable.
8. Shall be capable of distributing system updates for connected PICA systems.
    (a) Shall be capable of sending update data and images to the appropriate subsystems.
    (b) Shall identify whether or not the connected PICA systems require updating.
9. Shall be capable of giving debugging and troubleshooting output.
    (a) Shall display simple status indication using Light Emitting Diodes (LEDs). (See 4.1.5)
    (b) Shall present more detailed debugging and troubleshooting information over a dedicated RS-232 connection.
10. Shall be capable of actively notifying the power-company and consumer.

4.1.2 Behavioral Requirements

1. Shall store user-defined configuration in non-volatile media.
    (a) Shall initialize this configuration using a pre-defined default configuration.
    (b) Shall load the default configuration if the user-defined configuration is unavailable.
        i. This includes the user-defined configuration being absent.
        ii. This includes the user-defined configuration containing invalid data.
2. Shall include a backup firmware in the event of a failed firmware upgrade.

   (a) Firmware upgrade success or failure shall be determined by comparing checksums of the firmware to be written and the firmware present after writing.

   (b) The backup firmware shall be engaged if the system fails to boot from the first firmware.

3. Shall store critical event logs from connected PICA systems in a non-volatile media.
4. Shall host a webpage to display system information when browsed over the LAN connection.
5. Shall store its software in non-volatile media.
6. Shall run an operating system to manage hardware, device drivers, and connections to connected PICA systems.
7. Shall use a defined communication protocol to communicate with connected PICA systems.

4.1.3 Software Requirements

1. Shall include and run an upgradable Operating System (OS).
   (a) The OS shall include the drivers necessary to operate the system hardware.

   (b) The OS shall include the protocols necessary to connect to PICA sub-systems.

2. The OS shall have an administrator-privileged user who may change the configuration of the system and of connected PICA systems.
3. The OS shall give the system owner access this administrator-privileged user account.
   (a) The OS is not required to give such access to connected systems that are not owned by the owner of the base station system (such as those owned by the power company).

   (b) The OS shall authenticate that the user requesting administrator access is authorized to do so using a means configured during system installation.

4. The OS shall include the protocols necessary to connect to an existing LAN.
   (a) The OS shall include a Dynamic Host Configuration Protocol (DHCP) client.

   (b) The OS shall support manual LAN connection configuration.

5. The OS shall control the system connectivity hardware to prevent unwanted devices, such as those owned by other customers, from being connected to the system.

(a) The OS shall distinguish between wanted and unwanted systems as defined by administrator selection.

i. By default, ZigBee connection requests will be rejected until authorized by the administrator.

ii. By default, LAN connection requests will be trusted unless disallows by the administrator.

(b) The OS shall record connection requests from unwanted devices and display them to the administrator.

6. Shall include and run Hypertext Transfer Protocol (HTTP) server software to provide the web interface.

7. Shall include the necessary tools to download and apply software and firmware updates.

4.1.4 Hardware Requirements

1. Shall include a power supply to power the system from line voltage during regular usage conditions.

2. The power supply shall tolerate moderate expected fluctuations in input voltage from a standard power outlet.

(a) The power supply is not required to tolerate incorrect voltage input (240V instead of 120V, etc.)

(b) The power supply is not required to tolerate voltage spikes as may be associated with electrical storms.

3. Shall include an internal power storage device to allow the system to perform necessary data storage immediately after a power failure.

4. Shall have adequate Random-Access Memory (RAM) to contain system software, cache data from other PICA devices, and cache data to send to other PICA devices without requiring a cache flush more than once per minute.

5. Shall have a processor to execute the system software and process data from connected PICA devices.

(a) The processor must have sufficient processing speed to process the incoming data from other PICA devices at a rate greater than the rate of data incoming.

(b) The processor must be able to perform all necessary mathematical operations without a co-processor.

6. Shall have an Ethernet controller for connecting to a LAN.
7. Shall have an RS-232 controller for debugging and troubleshooting the system.
8. Shall have ZigBee connectivity hardware for communication with other PICA systems.
9. Shall have non-volatile storage dedicated to storing system firmware.
   (a) Shall have a re-writable non-volatile storage device to contain boot firmware.
   (b) Shall have a separate non-volatile storage device to contain backup firmware to use in the event of boot failure.
10. Shall have non-volatile storage sufficient to store system software.
11. Shall have non-volatile storage sufficient to store recorded events and short-term consumption history for up to a period of 3 years.

4.1.5 User Interface Requirements

1. Shall provide a web interface for viewing collected data over the LAN.
2. Shall provide a web interface for viewing current and predicted pricing information as provided by the power company.
3. Shall provide a web interface for system administration to an authenticated administrator.
   (a) Shall include an interface for managing updates to the system's firmware and software
      i. Shall include an interface for displaying the version numbers or codes of the firmware and software currently installed on the base station.
      ii. Shall include an interface for indicating the availability of system updates for the base station.
      iii. Shall include an interface by which the administrator may request that the base station apply its available updates and be informed on whether or not the base station must be rebooted after the update is installed.

iv. Shall include an interface for indicating the progress of update installation.

(b) Shall include an interface for managing connections to other PICA systems.

i. Shall include an interface for displaying current connections to other PICA systems.

ii. Shall include an interface for adding, removing, or connections made to other PICA systems.

(c) Shall include an interface for administration of connected PICA systems.

i. Shall include interfaces for managing configurations of connected PICA systems.

ii. Shall include an interface for deploying software/firmware upgrades to connected PICA systems.

4. Shall provide a debugging interface over an RS-232 serial connection.

5. Shall display status indication lights.

(a) System is connected to power.

(b) System is connected to other PICA subsystems.

(c) System is connected to the home network.

(d) System has encountered an error.

i. Error light shall trigger if a connection to another PICA system is lost and cannot be recovered within 30 seconds.

ii. Error light shall trigger when the storage medium contains less than 5% free space.

iii. Error light shall trigger when the storage medium's file system cannot be mounted.

A. This includes the storage medium being physically absent.

B. This includes the storage medium being corrupt.

4.1.6 Power Requirements

1. Shall be powered from a standard 120V wall outlet.

2. Shall have a Direct Current (DC) power supply to power internal components.

3. Shall have a backup power supply to enable the system to save all measurements residing in memory to the non-volatile storage medium.
4. Shall require less than 10W to operate.

4.1.7 Codes and Compliances

1. Shall have a polarized electrical plug if the power supply features an an off-on control switch.
2. Shall restrict electromagnetic (EM) radiation to comply with Federal Communications Commission (FCC) Title 47 Part 15.

4.2 Solid State Breakers and Monitoring Requirements

All requirements are to be assumed to be of the solid state breakers (“the system”) unless explicitly stated otherwise.

The breaker subsystem must be capable of two major functions; it must protect the connected circuit when a specified threshold is exceeded, and it must pass information to and from the user interface, either directly or through another subsystem.

4.2.1 Functional Requirements

1. Shall be capable of completely disconnecting, either physically or virtually, the power delivered to the connected circuit.
2. Shall provide two-way communications to the base station through the Master Control Unit (MCU).
   (a) Shall provide power usage information, including at a minimum, instantaneous voltage and current of the connected circuit.
   (b) Shall be capable of providing on/off status of the breaker.
   (c) Shall be capable of turning breakers on and off when requested by the base station.
3. Shall be capable of detecting power sags, brownout conditions and blackouts.

(a) The National Power Corporation (NPC) defines sag as ”80% to 85% below normal [voltage] for a short period of time,” [3]. For project uses, this is below 100V. The team defines “a short period of time” to be for three cycles to ten cycles, based on information given in [4]. The NPC defines a brownout as “a steady lower voltage state,” [3] Blackouts are defined by the NPC as “as zero-voltage condition that lasts for more than two cycles,”[3].

4. Shall store up to a minute of gathered information for at least five minutes in on-chip memory for transmission to the MCU.
5. Shall package the stored information for transmission to MCU.
6. Shall be capable of turning off circuits individually.
7. Shall stop current flow to a circuit when it exceeds a specified threshold.
8. Shall be capable of being reset when stopped, without requiring new parts such as fuses.

4.2.2 Behavioral Requirements

1. Shall initialize all components by writing from non-volatile storage to all necessary registers when power is restored to the circuit.
2. Shall report all system events that are not part of standard operation to the critical event log.
3. Shall measure voltage levels in the connected circuit.
4. Shall measure current flow in the connected circuit.

4.2.3 Hardware Requirements

1. Shall use non-volatile storage to store data when the system is without power.
2. Shall be capable of managing its own data and functions.

4.2.4 User Interface Requirements

1. Shall have an external interface that is understandable by the average consumer.
2. Shall provide a way to control the breakers independent from the base station.

3. Shall encase all circuitry except user interface controls (buttons, switches) so that they cannot be tampered with without breaking the casing.

4.2.5 Power Requirements

1. Shall be powered by line-voltage.
2. Shall be powered such that interrupting the circuit does not cause the breaker control circuitry to lose power.

4.2.6 Physical Requirements

1. Shall have dimensions less than or equal to 1in x 3in x 4in for a single breaker or 2in x 3in x 4in for a breaker pair.
   (a) Dimensions are determined by the standard size of a mechanical breaker, so that smart breakers may be interchangeable with conventional residential or commercial mechanical breakers.
2. Shall not weigh more than one pound.
   (a) Weight is determined by the average weight of a standard mechanical breaker [5], so that smart breakers may be interchangeable with conventional residential or commercial mechanical breakers.
3. Shall accommodate wire sizes from 14-4 Cu to 12-8 Al.
   (a) Wire size is based on the average size wire used with standard mechanical breakers [5], so that smart breakers may be interchangeable with conventional residential or commercial mechanical breakers.

4.2.7 Safety Requirements

1. Shall protect everything connected to the circuit from current exceeding a specified threshold by providing circuit interruption.
2. Shall have safety hazards clearly marked and visible from outside the system.
3. Shall safely isolate high-voltage areas so that they provide no more threat than a standard wall outlet.

4.2.8 Codes and Compliances

1. Shall be compliant with American National Standards Institute (ANSI) C12.19 [6].
2. Shall be compliant with ANSI C12.21 [7].
3. Shall be compliant with FCC Title 47 Part 15 [8].

4.3 Electric Panel Meter Requirements

All requirements are to be assumed to be of the electric panel meter (“the system”) unless explicitly stated otherwise.

4.3.1 Functional Requirements

1. Shall continuously monitor the power used from either a single-phase or a multi-phase installation.
2. Shall store power usage data locally, to be transmitted back to the base station at regular intervals.
3. Shall by default display instantaneous and historical power usage data on an Liquid Crystal Display (LCD) module integrated into the electric panel.
4. Shall provide two-way communication with the MCU to report usage data.
5. Shall be capable of detecting a brownout condition and storing critical data before shutting down.
6. Shall be capable of restarting and restoring stored data after a brownout condition.
7. Shall be capable of detecting any tampering, such as opening the sealed metering unit, and transmitting a tamper message to both the power-company and the consumer.
8. Shall monitor current flow through the main service lines for automated meter reading.
9. Shall monitor voltage levels on the main service lines for automated meter reading.
10. Shall control the service shutoff switch by receiving and validating a service shutoff message from the power-company.
11. Shall provide a method for controlling the service shutoff switch from a local interface.

12. Shall provide an interface for 3rd party meters, such as gas, water, or other utility meters to report data over the PICA network.
13. Shall support on-demand reports from the power company via the Zigbee network or the user via the PICA web interface of power usage, energy consumption, demand, power quality and system status.
14. Shall support bi-directional metering and calculation of net power usage to support alternative energy generation systems.
15. Shall support automatic meter reads.
16. Shall analyze the voltage flicker, logging a warning when the flicker exceeds 20% of the stated $117V_{RMS}$.
17. Shall meter reactive power consumption, logging data for billing purposes.

4.3.2 Behavioral Requirements

1. Shall, in the event of wireless link loss; attempt to re-establish the wireless link.
2. Shall, in the event of a wireless link loss, revert to stand-alone mode, storing data internally until internal storage is full, at which point the system will begin overwriting the oldest data with the newest data.
3. Shall, in the event of a wireless link loss, notify the user via the LCD display.
4. Shall perform a built-in self-test upon system boot up to verify onboard storage integrity and to verify proper operating software.
5. Shall, in the event of a brownout, save all volatile information to non-volatile storage space.
6. Shall be capable of detecting corrupted data, via parity bits, when brought out of a brownout condition.
7. Shall log all events processed into the following four categories:
   (a) critical: messages requiring immediate attention.
   (b) error: messages requiring attention and may affect system functionality.
   (c) warning: messages that require attention but do no impact system functionality.

(d) note: messages that require no attention but provide verification of proper operation.

8. Shall report all events to the PICA base station.
9. Shall report to the power-company as specified by event criticality.
10. Shall have dedicated non-volatile storage for all critical settings and configuration data.
11. Shall compute the total power used in kilowatt-hours.
12. Shall be capable of receiving messages from the power-company, providing the user with the current cost of a kilowatt-hour.
13. Shall use 128-bit Advanced Encryption Standard (AES) encryption for all messages transmitted outside of the device.
14. Shall report the total amount of outage time to both the power company and the PICA base station.
15. Shall date-stamp all detected outages with the date, time, and duration of the outage.

4.3.3 Software Requirements

1. Shall verify system firmware on boot up.
2. Shall periodically, minimum once per day, perform a system check to verify the health and status of the system.
3. Shall perform an on-demand system health and status check as demanded by the PICA base station or the power-company.
4. Shall contain sufficient non-volatile storage for all system configuration settings.
5. Shall be updateable through the power-company wireless interface.
6. Shall be capable of properly recovering from a failed software update.
7. Shall give authorized access to components of the system configuration as appropriate to the power-company and consumer.
8. Shall notify the power-company and the PICA base station once service has been restored containing the time of restoration and a voltage measurement.
9. Shall have a unique IPv6 address for the power-company mesh network.

10. Shall have a unique IPv4 or IPv6 network address for the local home-area-network.
11. Shall receive an NTP message from the PICA base station to set the hardware clock.
12. Shall synchronize the hardware clock with the base station time once per day.
13. Shall support on-demand hardware clock synchronization via the PICA base station web interface.

4.3.4 Hardware Requirements

1. Shall be completely enclosed in a weatherproof case, tolerant of extreme temperature differences.
2. Shall be completely Alternating Current (AC) coupled against transient AC voltages.
3. Shall be mounted in the same location as a standard power meter.
4. Shall provide non-volatile storage.
5. Shall be grounded.
6. Shall provide a hardware system clock, set by the software and synchronized with the PICA base station.

4.3.5 User Interface Requirements

1. Shall have a 160-segment LCD display module, viewable from outside the electric panel.
2. Shall be capable of interfacing with a web-based application for stand-alone configuration.
3. Shall provide push-buttons for changing the viewing contents on the display module between metering and system status views.

4.3.6 Power Requirements

1. Shall be capable of operating from line-voltage.
2. Shall be powered from before the master breaker, preventing the meter from losing power when the master breaker is switched off.

4.3.7 Safety Requirements

1. Shall meet or exceed safety requirements for devices inside an electric panel.
2. Shall provide a grounding point to ground the system when installed.
3. Shall be protected against the elements. 4. Shall safely isolate high-voltage areas.

4.3.8 Codes and Compliances

1. Shall be compliant with ANSI C12.19.
2. Shall be compliant with ANSI C12.21.
3. Shall be compliant with FCC Title 47 Part 15.

5 Design Goals

5.1 Provide a physical system that accurately monitors power usage

Inaccurate information provides no benefit to either the power company or the consumer, despite any added features. Therefore, the system should be as accurate as or more accurate than monitors currently used.

5.2 Provide manuals for maintenance and general use

The design team recognizes that no system is perfect and will eventually need maintenance and that most consumers do not have extensive knowledge of electrical systems or components. Providing manuals will assist the consumer in understanding their system and getting the most benefit from it. The design team will look to create a manual that will include an overview of how the system works, how to install and configure the system, and how to maintain the system during the course of normal use. This manual will also contain a troubleshooting guide, support information, liabilities, safety information, and any other pertinent information.

5.3 Design the system to be modular

Providing information to both the power company and to the consumer is the main goal of the project, but it is possible that an installation will not include all the subsystems. For example, a consumer may want the consumer-oriented part of the system, while the power company does not want the power-company-oriented subsystem, or vice versa. The modularity goal aims to satisfy all situations without forcing extra costs on any party. To do this, the design team will design the system so that the subsystems providing information to the power company and the subsystems providing information to the consumer do not depend on or require each other.

5.4 Present power usage information in a way that is understandable to an average consumer

The goal of the design team is to present the information in an understandable format for the consumer. Because the average consumer does not have an engineering background, the design team would like to

display the power usage information in dollars per minute, per hour, per day, per week, per month and per yearly. The system will report these costs in addition to more technical information including voltage, current, power factor and more.

5.5 Minimize on-site maintenance as much as possible

In many cases, the consumer calls the power company to fix something as simple as a tripped breaker, and the power companies waste a lot of time just driving to and from the site. The ability to do work remotely allows the power company to minimize this cost. Remotely controlling or monitoring different aspects of the meter also lets the power company quickly assess if a problem still needs attention. Aggravated customers may also become violent towards power-company technicians, so by having remote access, the power company's employees are safer from these threats.

6 E-Meter

The E-Meter, short for electricity meter, fulfills the role of monitoring all power consumed for a given customer. The PICA E-Meter exists as a replacement for the standard electricity meter outside most homes and businesses. As a replacement for an already existing piece of technology the E-Meter must fulfill all the same roles as a traditional power meter as well as providing extended capabilities, such as wireless reading reporting and tamper detection.

Since the average consumer does not actually own the physical metering hardware, this portion of our senior design project targets the electric companies. We hope that by providing more accurate readings, more reliable meters, and more convenience the PICA E-Meter could entice some, if not all of the electric utility providers who still rely on traditional electric meters.

6.1 Original Ideas

Early in the project team PICA decided to use a Texas Instruments (TI) MSP430 low-power microprocessor. The family of MSP430 devices includes several devices that could perform the necessary tasks, however, the MSP430F471xx family of devices are tailored for metrology. After some investigation the team decided to use an MSP430F47197 microprocessor as the core of the E-Meter. A summary of the specifications for this micrprocssor follows in table 1.

Table 1: Specifications for the MSP430F47197[2]

Specification	MSP430F47197
Frequency	16 MHz
Flash	120 kB
RAM	4 kB
GPIO	68
Pin/Package	100LQFP
LCD Segments	160
ADC	7 16-bit Sigma Delta
Other Integrated Peripherals	Comparator, DMA, Hardware Multiply, SVS
End Equipment Optimization	Energy Meter

Continued on next page

Table 1: Continued from previous page

Specification	MSP430F47197
Interfaces	2 USCI_A, 2 USCI_B
Timers	1 Watchdog, 2 16-bit, 2 8-bit

With its included flash, RAM, and onboard peripherals the MSP430 provides a good all-in-one solution on which to base the E-Meter. Along with these features the MSP430 boasted the lowest power consumption of any solution the team examined. A full discussion of the alternatives can be found in Team PICA's Project Proposal and Feasibility Study (PPFS) [9].

In order to be compliant with ANSI codes C12.19 [6] and C12.21 [7] the PICA E-Meter must display the instantaneous power usage on the local metering unit. For this reason, the E-Meter should have an LCD screen displaying the necessary data. After some research, the team found 2 possible display units the SCLCD from SynchroSystems Embedded Computer Design [10] and the Softbaugh SBLCDA4 from SoftBaugh Inc. [11]. Each of these displays would use the MSP430's internal 160-segment LCD driver. As both display units are comparably priced the team chose to use the display from SynchroSystems because they provided a driver written in embedded C along with a backlight kit to make the display more readable. The implementation of this display will be discussed in more detail in the next section.

6.2 Design Work

In order to break down the E-Meter into several subsystems Team PICA began with the block diagram seen in figure 2. This block diagram describes 5 major subsections of the E-Meter, the microcontroller, the LCD screen, the sensors, the data-link, and the debug interface. In order to construct a fully working E-Meter each of these components required some individual proving and then integration into the final design.

6.2.1 Microprocessor Integration

Before beginning work on any of the other components, Team PICA needed to prove that code could be compiled and loaded onto the MSP430 platform. In order to accomplish these two tasks several options exist:

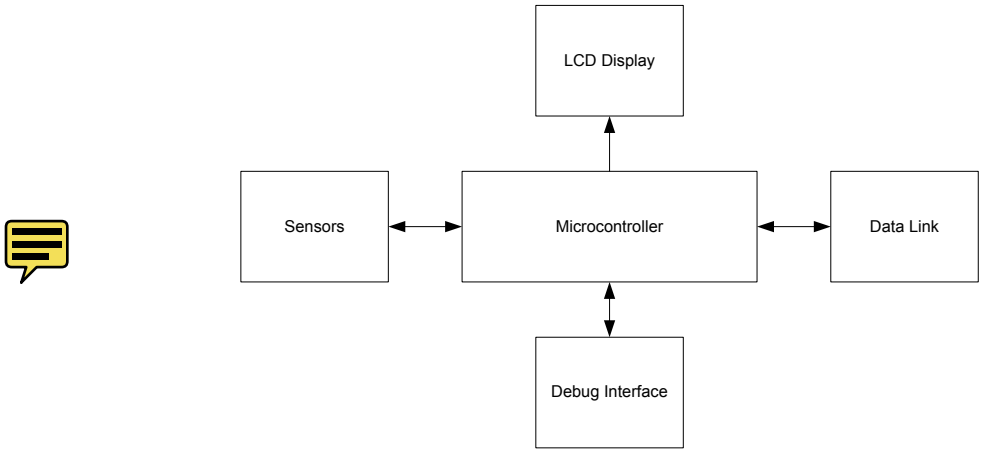

Figure 2: A simple block diagram of the PICA E-Meter.

1. IAR Workbench
2. Code Composer Studio
3. MSPGCC 4.x

The first two options, both provided by TI, are proprietary development environments tailored for compiling and debugging embedded C. However, both of these tools are provided for Microsoft Windows only. That being the case, the team wanted to try and work with the open-source MSPGCC4 project as the compiler and debugger could run on any computing platform. At the time this seemed like a good idea as the two team-members working on the E-Meter did not use Microsoft Windows as their primary computing platform.

After some configuration, the team successfully compiled and installed the MSPGCC compiler. A full guide to repeating this process can be found in appendix A.3. The first test, after setting up the toolchain, consisted of determining if the computer could detect and properly configure the MSP430 programming pod. Figure 3 shows the dmesg output showing successful pod detection and configuration. At this point, the programming pod is recognized as a valid USB device and configured, thus the pod can configure the MSP430 device. However, at this point the team learned that while the MSPGCC4 toolchain supported their device, the debug utility, mspdebug did not support loading code through the team's programming pod. Thus, while MSPGCC4 could compile code, the team would need another tool to load compiled code onto the microprocessor.

```
kendrickwiersma@MINIPC:~$ dmesg | tail
[  464.521313] usbcore: registered new interface driver ti_usb_3410_5052
[  464.521321] ti_usb_3410_5052: v0.10:TI USB 3410/5052 Serial Driver
[  464.521624] usb 4-1: USB disconnect, address 2
[  464.688120] usb 4-1: new full speed USB device using ohci_hcd and address 3
[  464.958526] ti_usb_3410_5052 4-1:1.0: TI USB 3410 1 port adapter converter de
tected
[  464.958594] ti_usb_3410_5052: probe of 4-1:1.0 failed with error -5
[  464.962405] ti_usb_3410_5052 4-1:2.0: TI USB 3410 1 port adapter converter de
tected
[  464.962778] usb 4-1: TI USB 3410 1 port adapter converter now attached to tty
USB0
[  468.041208] audit_printk_skb: 15 callbacks suppressed
[  468.041216] type=1400 audit(1297454335.835:17): apparmor="DENIED" operation="
open" parent=2061 profile="/usr/sbin/cupsd" name="/dev/ttyUSB0" pid=2074 comm="s
erial" requested_mask="w" denied_mask="w" fsuid=0 ouid=0
kendrickwiersma@MINIPC:~$
```

Figure 3: Dmesg output showing Ubuntu Linux correctly detecting and configuring the MSP430 programming pod.

Upon learning that MSPGCC4 would not work, Team PICA turned to the suggested tool for developing MSP430 software, IAR Workbench. However, after several trial runs, the team readily switched to Code Composer Studio 4 (CCS4) for its advanced debugging abilities and familiar interface (Code Composer Studio is based on the Eclipse platform).

Using CCS4 team PICA wrote a “Hello World” program for the MSP430, capable of blinking one of the onboard LEDs. This provided conclusive evidence that the team could successfully write, compile, load, execute, and debug code for the MSP430. See appendix A.2 for a listing of our “Hello World” program.

6.2.2 LCD Integration

Team PICA decided to begin integration of external peripherals with the LCD as SynchroSystems had already provided driver software compatible with the MSP430 family of devices. After some minor tweaking to tailor their provided driver and demo to run on the MSP430F47197, the design team could successfully compile code to work with the LCD screen. Primarily this tailoring consisted of porting their code to use the LCD_A controller (an integrated peripheral on the MSP430F47197) as opposed to the LCD Controller found on some of the other MSP430 devices. However, this did not address the physical hardware connection of adding a screen to the MSP-TS430PZ100A development board donated by TI.

In order to make the physical connection between the SynchroSystems SCLCD and the

MSP-TS430PZ100A (hereafter refered to as MSP430 development board) Team PICA designed a breakout board using the freely available EAGLE schematic and PCB layout software. See figure 4 for a picture of the Computer Aided Design (CAD) drawing of the breaout board. After correspondance with Chuck Cox,

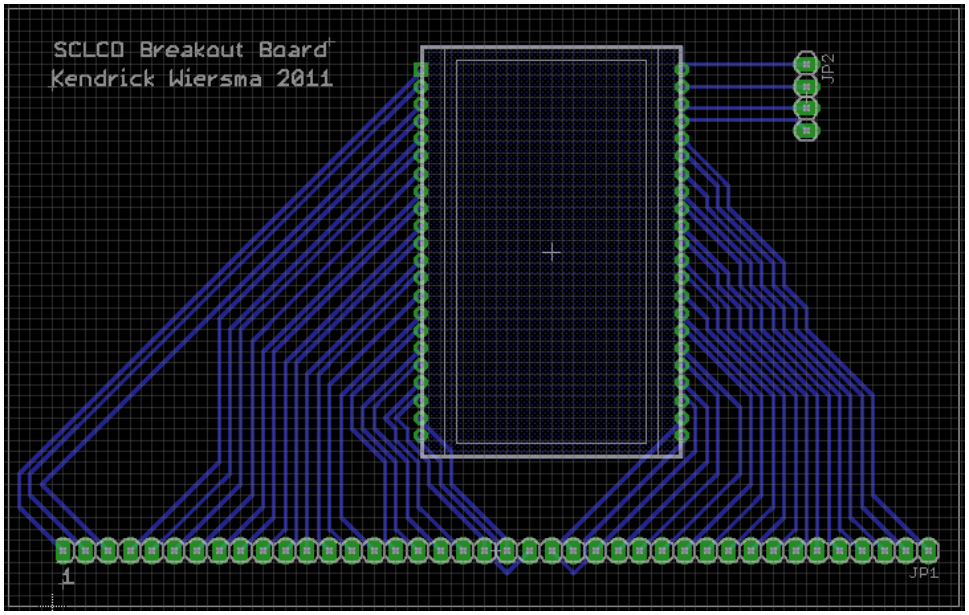

Figure 4: EAGLE CAD drawing of the SCLCD Breakout Board.

owner of SynchroSystems, SynchroSystems provided an EAGLE library containing the CAD drawings of their part, allowing the breakout board board to be produced quicker. At the time of this writting this part library is not available for public release.

In spite of having a fully assembled breakout board and code that compiled properly, Team PICA still struggled to properly display information on the LCD screen. After reading through the user's manual [12], the team learned that the MSP430F47197 microprocessor required an additional clock circuit to drive the peripherals, which operate at a different frequency than the main processor. The clock circuit can be seen in figure 5.

With the addition of this clocking circuit, and a 22uF capacitor on the **LCDREF** (**LCDREF!**) pin, the LCD screen immediately began displaying characters and the modified demo from SynchroSystems ran properly. A video of the LCD screen working can be seen at

`http://youtu.be/9oDglYHUhxo?hd=1`.

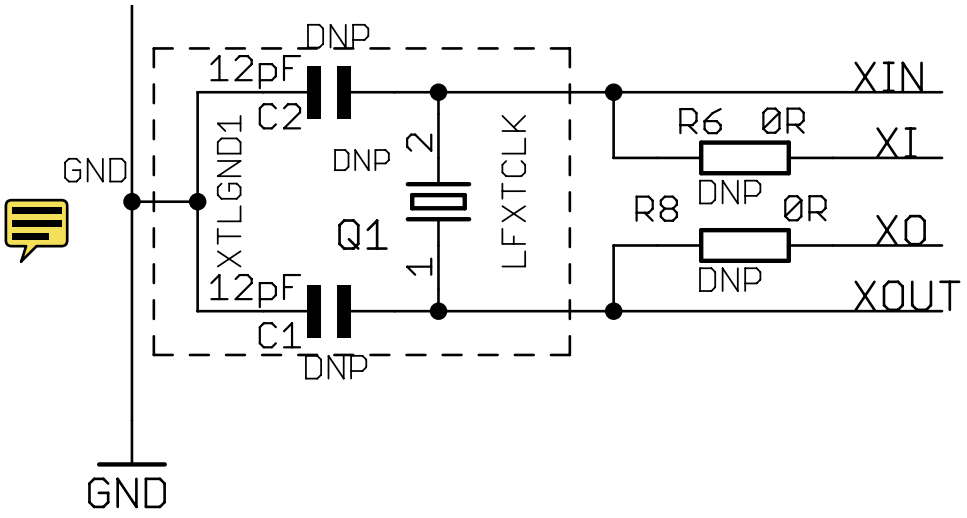

Figure 5: 32.768MHz clock circuit for MSP430 development board.

6.2.3 Sensor Integration

After Team PICA had the ability to display data on a local LCD, the next logical step became to collect data from our sensors. As previously noted, the MSP430F47197 contains 7 16-bit Sigma-Delta (abbreviated SD16) Analog-to-Digital Converter (ADC)s used to convert a voltage into a digital measurement. For our 3-phase configuration, each of SD16 converter requires an input network to ensure that the differential voltage applied to the terminals of the ADC does not exceed 500mV.

Of the 7 available SD16 converters, 2 are dedicated to each phase of the power. The first SD16 for each phase measures the current, using a special input network seen in figure 6. The positive and negative

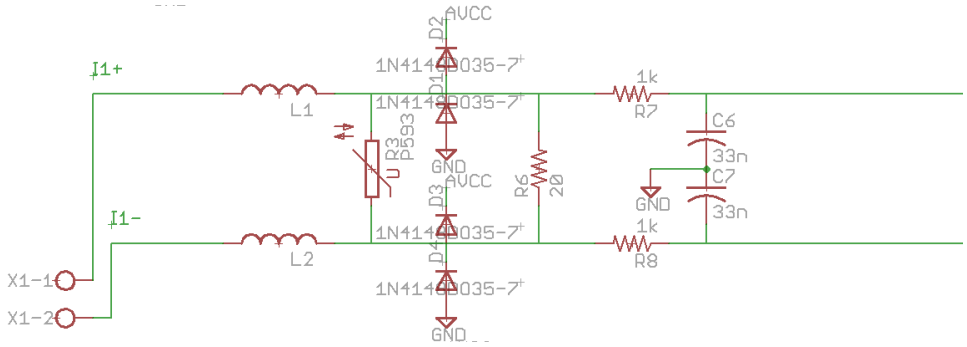

Figure 6: SD16 current sense input network.

terminals of a current transformer attach to points X1-1 and X1-2 in the SD16 input network. A current transformer provides a convenient method for measuring large currents by keeping the wires carrying large currents physically detached from the circuit. Current transformers rely on a process known as induction,

where a wire carrying current is passed through a loop, or many loops, of wire. Those loops of wire then begin to conduct current in response to the EM field produced by the current carrying wire. The input network can sense the wire-loop induced current as a differential voltage, which the SD16 can then convert into a digital measurement. The inductors, L1 and L2 provide protection against any electro-magnetic interference while the varistor R3 and diodes D1-D4 provide protection against large transient spikes should the current transformer induce too large a current. Resistor R6 is known as the burdon resistor for the current transformer, which provides a low impedance path for induced current to flow, and further limiting the amount of induced current in the current transformer, keeping the circuit operating in a stable and predictable manner. The remaining resistors, R7 and R8, along with capacitors C6 and C7, server to further limit and scale the generated differential voltage from the burdon resistor before being read by the SD16 converter. The value of the burdon resistor depends on the turns ratio of the current transformer.

Once the design team added the serial data port, discussed in the next section, the upper 4 bits of the data could be transmitted back to the workstation Personal Computer (PC) for processing in MATLAB. The intent of this test being to verify that readings from the SD16 were periodic in nature. By showing that the SD16 converter could properly capture and read a periodic waveform the team proved that the SD16 converter functions as expected. See figure 7 for the test results. The results really need very little

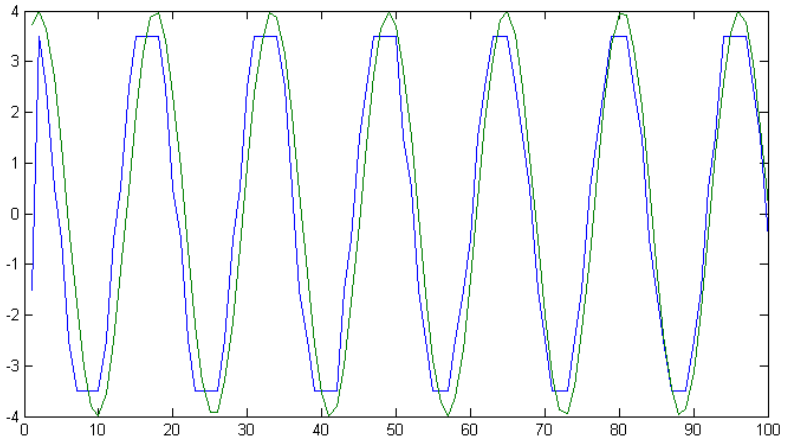

Figure 7: Data captured from the SD16 converter over RS232 (blue). Ideal sinusoid (black).

explanation, figure 7 shows that the output of the SD16 converter, once normalized to zero, can properly read a sinusoid. Given that only the top 4 bits of the result were captured for this test, the loss of the top and bottom of the sinusoid should be resolved with capturing more bits. See appendix A.4 for the code

used in this test.

6.2.4 Serial Data Link Integration

The second to last block of the block diagram in figure 2, data link, describes the method for communicating the measured data to the rest of the world. Ultimately, Team PICA desires to implement an Xbee radio communication system, however, before using the Xbee radio a simple RS232 Universal Asynchronous Receiver/Transmitter (UART) fulfills the data transmission requirement.

In order to implement RS232 communication the design team chose to use the MAX233A Integrated Circuit (IC) which comes in a 20-pin Dual Inline Package (DIP). The MAX233A provides line-level conversion from the 3.0V output of the MSP430 up to the 5.5V required for RS232 communication. Conversely, the chip also converts the receive line from RS232 5.5V down to 3.0V for the MSP430 UART. Other methods exist for performing this line-level conversion, however the MAX233A provides a simple, 1-chip solution without requiring large amounts of board space and without adding significant cost to the final product. See figure 8 for the serial data-link circuit used in the PICA E-Meter.

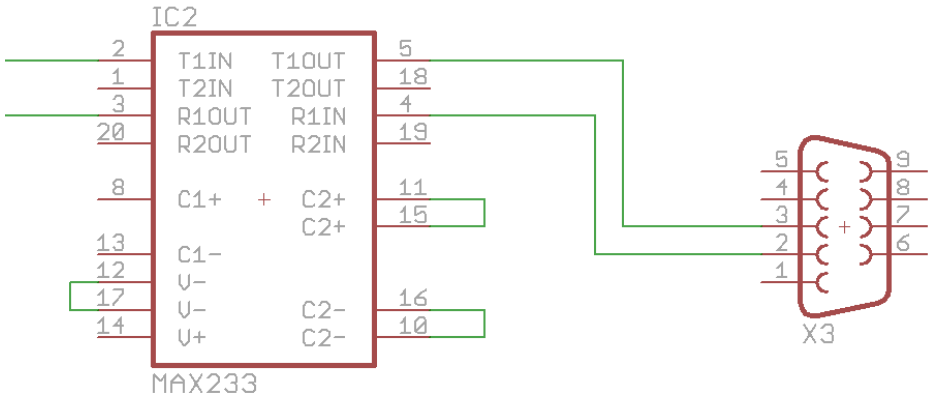

Figure 8: MAX233A RS232 serial data link circuit.

In order to verify the proper operation of the serial data link, the team performed an echo test. In order to perform this test, the team wrote code for the MSP430 to move a received character from the receive buffer to the transmit buffer on a receive interrupt. Characters typed into a serial terminal on the team's workstation PC would then be transmitted from the computer, to the MSP430 before being transmitted back to the PC and displayed in the terminal window. This operation succeeded on the very first attempt

with the results seen in figure 9a and 9b. Even though serial communication can be implemented fairly

(a) Serial UART registers in the MSP430

(b) Serial UART terminal on the workstation PC

Figure 9: RS232 Echo Test

easily, the team desired to implement a wireless interface to the E-M allowin greater distance between the unit and the data recipient. Xbee then becomes the logic choice, as Xbee uses a wireless serial interface. Each Xbee device holds non-volatile configuration data describing the devices function, either an end-device or a router. The Xbee devices, however, require an additional signal, **DTR!** (**!**), to alert the device when a data packet is ready for transmission. The MSP430 software will need to generate this signal when it wants to transmit data.

Meanwhile the team decided to also add support for flow control to the RS232 port. This simply involves adding two additional signals between the D-Sub 9 connector and the MAX233A chip, along with two additional lines to the MSP430. As of right now, software support for flow control does not exist. See figure 10 for a full schematic for the Recommended Standard 232 (RS232) connection.

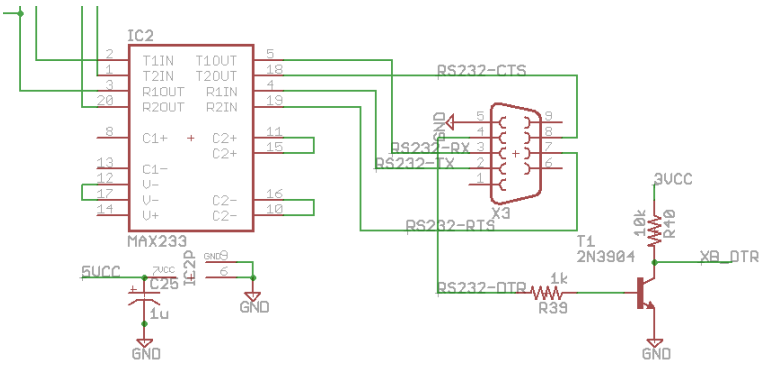

Figure 10: Flow control and DTR lines for RS232 communications

6.2.5 Debug Interface Integration

While the design team continunes to use the MSP-TS430PZ100A development board the design does not require a dedicated JTAG debug port for programming and debugging the micrprocessor. However, once

the team moves to debugging the final Printed Circuit Board (PCB) version of the E-Meter. The JTAG standard defines the implementation of the communications protocol as well as circuitry needed to interface a JTAG programmer and the JTAG port on a device. For the E-Meter, see figure 11 for the JTAG interface circuit. In a final production version the JTAG interface may still remain on the PCB for the initial

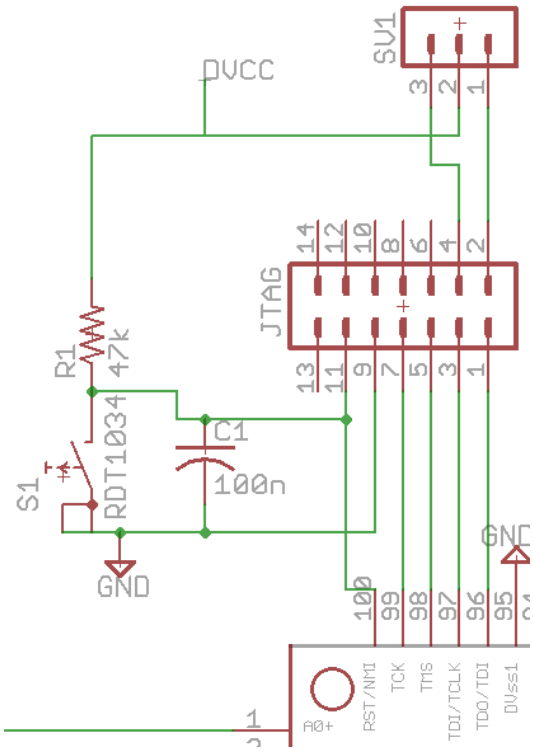

Figure 11: JTAG circuit for the MSP430F47197.

programming of the microprocessor but the advanced functionality of reseting, clearing, and stepping the microprocessor should be disabled.

order for JTAG to properly configure a device, 4 signals must be present, with an optional 5th signal:

1. TDI: Test Data In
2. TDO: Test Data Out
3. TCK: Test Clock
4. TMS: Test Mode Select
5. TRST: Test Reset (optional)

The MSPP430 processor has dedicated ports for each of these signals which arrive through the TI programming pod. In order to add this functionality to the E-Meter PCB, a 14-pin header provides the interface between the programming pod and the microprocessor. The signals can either be hardwired into the microprocessor, or as seen in figure 11 with minimal extra circuitry, allow for a push-button reset switch. Effectively, this switch drops the VCC to ground, killing power to the microprocessor, and pulsing the **TRST!** (/**TRST!**) line, which halts activity. Because the programming pod provides capability to power the microprocessor, and additional header allows for switching between the programmer power and the board power. Capacitor C1 in figure 11 simply provides debouncing on the incoming signal. This prevents cases where the user jiggles the buttons, potentially causing several partial resets.

6.3 Software Design

The typical programming paradigm for the MS430 is to maximize the amount of time the processor spends in the sleep state. To do this, many of the peripherals on the MS430 chip can be configured to generate interrupts that wake the processor and execute code relevant to the triggering events. The software designed for the E-meter follows this paradigm of sleeping and interrupts.

6.3.1 Design

The primary focus of the E-meter is measuring the current load and line voltage of each phase of electricity entering the building. The input networks for the E-meter convert these different elements into voltages that the MSP430 can read on its dedicated ADCs hardware. The E-meter software must allow time for these ADCs to resolve the voltages, but must also output summaries of these measurements. To do this, the software running in the E-meter first loads pre-defined values into the device's configuration registers, then measures the voltages on the ADCs, outputs a summary of the information collected, then repeats the collection and output sections.

The initialization step loads values into the MSP430's configuration registers that enable the desired features of the E-meter. The list of possible values can be found in the MSP430 User Manual, which the design team referenced frequently in determining the values for a suitable configuration. This stage selects the clocking frequency for the ADCs and for the serial UART, selects which of the ADCs to measure, and which pins are used for the LCD display.

As the final task in this preparation stage, the program enters a loop where the main calculations take place. At the beginning of this loop, the processor enables the ADCs and puts the main process to sleep.

When the ADCs have measured the voltage from their attached circuits, any of the devices may raise an interrupt to signal the completion of their task. This interrupt triggers the ADC Interrupt Service Routine (ISR), which determines which of the several ADCs circuits finished its calculation. When it does so, the ISR then turns off the ADCs, reads from the interrupting ADC, sq that 16-bit value, and adds it to a running total of squared measurements, which is stored as a floating-point number. Finally, the ISR increments a counter that indicates how many measurements have been taken since the last report has been made. If that increase does not cause the counter to meet a pre-de ined number, the ISR re-enables the ADC interrupts and exits. If th nber was met, the interrupts will remain disabled and the main loop resumes when the ISR exits.

When the main process wakes up, it resumes execution at the same location in the code where it had gone to sleep. Specifically, it continues executing the code in the loop it had just entered. To compute the Root Mean Square (RMS) voltage and current, the software divides the accumulated totals of the squares of these two elements by the number of measurements, then calculates the square root of these mean squares. Finally, these values are outp to the LCD screen and over the serial UART.

6.3.2 Limitations

As the square-root calculation is sometimes a lengthy one, these calculations are performed with the ADCs turned off, so that these calculations are performed without interruption. This approach does imply that ome data will not be collected, which decreases the accuracy of the measurements, but ensures that the measurements taken are not corrupted when processed.

6.3.3 Current Status

At present, the software collects readings from two of the MSP430's ADC "channels", simulating what an individual "phase" would be measured. As mentioned in the MSP430 user's guide, these two channels are grouped together, so they wait for each other to resolve before triggering the ISR. This approach can be extended to the other channels; with only a few changes in software, all of the six ADCs needed for this application can group together.

Laboratory execution and examination using a stopwatch show that the current software can make approximately 760 measurements per second on both channels when 120 measurements are taken before outputting over RS232.

6.3.4 Future Work

The E-meter software does not currently interface with the external LCD screen. Adding support for this peripheral will be necessary before the device is complete, as it will be included on the final PCB for the device. Additionally, the software must be able to handle the interface buttons to control the display and affect the overall behavior of the device.

The software is currently very single-threaded: when the main loop is active, the ADC functionality is disabled. While tests have shown that this only affects performance speed by about 6%, it may be possible to create a more multi-threaded program that would allow the ISR to do less computation while the main loop runs in the background to process the data while the ADCs read it. Many different approaches can accomplish this goal.

6.3.5 Critical Conditions

It is crucial that the sampling rate of the ADCs not decrease below 120 samples per second. In order to correctly sample a 60Hz wave such as line voltage or current in the U.S., the sampling frequency must be at least twice that rate. This requirement is known as the Nyquist criterion in numerical analysis.

6.4 Testing

In order to verify the performance of the PICA E-Meter, several tests stressing various components of the system are described below. These tests verify both functional and behavioral correctness of the E-Meter in various situations, responding to several different stimuli.

[Testing and verification plan not yet completed. however this section will include both the testing and verification plan and the results of the tests.]

6.5 Printed Circuit Board

Team PICA decided for the purposes of this project that a PCB would provide the best presentation of the final E-Meter. By designing a PCB for the project, the team would gain valuable experience in fabrication, and be able to present a cleaner project. Additionally, a PCB can help prevent parasitic capacitance inherent in terminal connections. For instance, by removing additional capacitance in the connections to the LCD display, the contrast increases and the power consumption decreases.

In order to design the PCB the team used the freely available program Eagle, produced by Cadsoft Computer. Several versions of the software are available, ranging in price and features from a free board-size limited version to an expensive professional no-limits version. As the team budget did not include software licenses, the free boad-size limited version was used to produce the E-Meter PCB.

Originally the E-Meter would be composed of a single PCB containing all of the components except for the power supply. However, as the free version of Eagle only allows for an 80 x 100 mm board, the E-Meter became two linked boards; one for the ADC input networks, and the other for the main E-Meter processing and data transmission components. These two boards can be seen in figures 12 and 13 (larger versions, and complete schematics, can be found in appendix A.6 and A.5 respectively). Each of these boards are

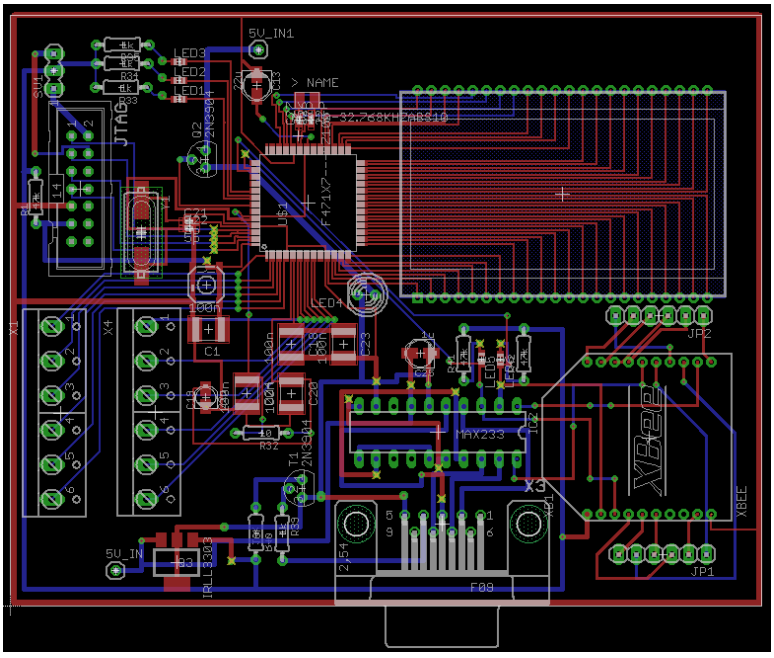

Figure 12: E-Meter main processing and data transmission board.

composed of 2 layers, a front and a back. Signals are routed on either side with vias inbetween the two

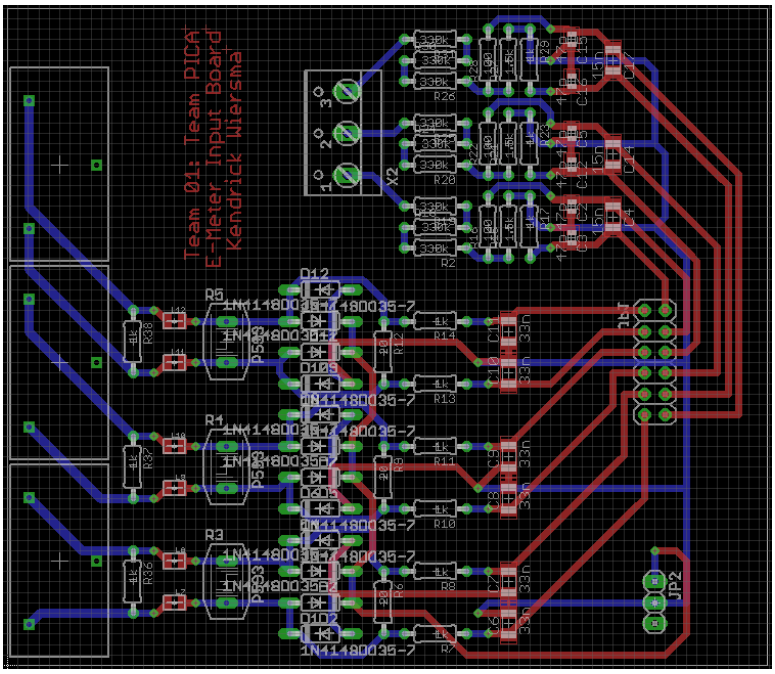

Figure 13: E-Meter input board.

sides. The team chose to use a two layer b after Johnson Controls, Inc. (JCI) graciously offered to allow the team to print PCBs on their rapid prototyping machine. This allows for a quicker turnaround time, and a significant decrease in the required project budget.

The main board brings together the MCU, RS232 transceiver, Xbee radio, LCD, user pushbuttons, and JTAG interface. Along with these components, several MCU peripherals, such as clocks, are also included on this PCB. Spatial locality of the components with respect to the MCU determined which components appeared on this board, and which components were allocated to the secondary board. Peripherals such as clocks and transmission signals that are susceptible to noise need to be close to the MCU to prevent interference from electromagnetic radiation in the environment or the surroundings. In order to further reduce the possibility of electro magnetic radiation, traces carring signals remain isolated (as much as possible) from traces carrying voltage. Where not possible, signal traces were routed between ground traces to provide extra shielding.

Eagle provides part libraries containing CAD schematics for many common parts for use in PCB development. Many companies also provide custom part libraries with their part to aid designers in during PCB layout. For this project, the team used the MSP430 library freely available from TI , the SparkFun library, a library from SynchroSystems, and a custom library in addition to the built-in part libraries. As Cadsoft cannot possibly provide CAD schematics for every part on the market, Eagle allows users to create

custom part CAD files. A tutorial to building a custom Eagle library can be found in appendix ..

6.6 Future Work

[Once the design nears completion the team will identify several points that could use improvement, or additional features that would provide a better overall experience for the user.]

7 Smart Breakers

7.1 Breakers

7.1.1 Breaker Overview

The breaker system is half of the smart breakers system aimed at providing the consumer with better circuit protection in addition to circuit information.

To achieve the goal of providing a system that is user friendly, the team decided to design smart breakers that the user can use in place of their current electro-mechanical breakers. This keeps installation simple and centralizes all parts of the system, making the system safer and more aesthetically pleasing.

7.1.2 Breaker System Diagram

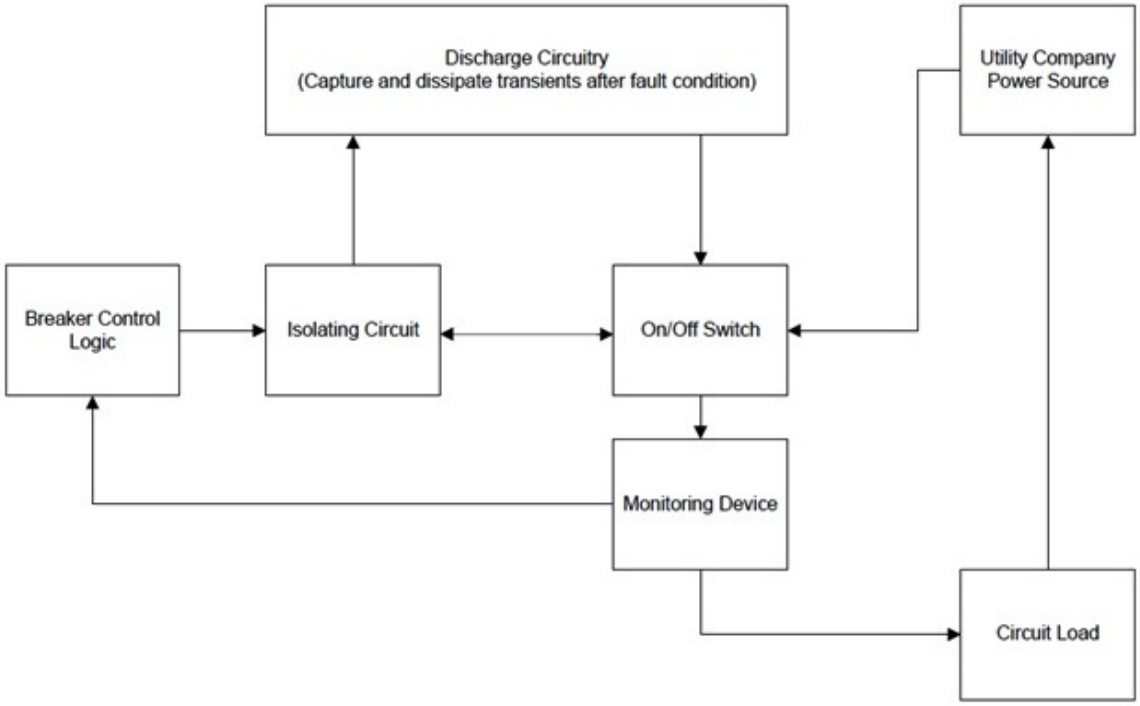

Figure 14: Block diagram of breaker system

The diagram above shows the breaker technology within the outlined box in relation to the other aspects of the smart breaker. The isolating circuit, the discharge circuitry and on/off switch are the three key components needed for a safe and fully functional breaker system.

7.1.3 Component Selection

On/Off Switch The team focused initially on selecting an appropriate switching component and looked at SCRs, triacs, FETs, thyristors and solid state relays as possible options. The table below shows several benefits and drawbacks of each.

Components				
SCR	Triac	FET	Thyristor	Solid State Relay
Benefits				
Work Well With High Current and High Voltage				
		We've Worked With Them Before		Easy to Use
Drawbacks				
Require Heat Sinks				
Require Additional Circuitry				
Require Complex Triggering				Expensive

Figure 15: Table comparing switching components

Ultimately, the team decided to use solid state relays, primarily due to their ease of use and lack of need for additional circuitry. This greatly reduced design time and allowed the team to focus on the primary goal of providing the user with power usage information.

Isola Circuit The isolation circuitry's purpose is to protect the control logic should any line voltage cross over to the control side. The team selected an opto isolator for this purpose based on ava ity, ease of use, and functionality.

Dis e Circuitry At high currents and voltages, suddenly turning the circuit off can cause damaging transients. The discharge circuitry captures those transients before they become a problem and dissipates them later in a safe manner. For a final design, this component is important; however, it is not necessary to demonstrate the smart breakers' ability to read and control power and was not included in this prototype.

7.1.4 Testing

Following selection of the switching component, the team ran a few basic tests and found that at low current levels the SSR provides as good or better functionality than standard airgap breakers. Turn on and turn off control voltages were also consistent, keeping the control logic simple.

7.1.5 Future Work

Solid state relays work well to demonstrate the smart breakers' ability to monitor and control electrical power, but are not a good selection for a consumer device because of high cost. Given more time, the team would like to design breakers that make use of another switching component at a lower cost and that minimize the need for heat sinks. Adding discharge circuitry is another aspect the team would like to include.

7.2 Individual Monitor

7.2.1 Problem Definition

The individual circuit monitors provide the means for the homeowner to know where power is used within the building and makes up the second half of the smart breakers, along with the switching circuitry. They measure voltage and current in the connected home circuit, use it for controlling the breakers and feed the information to the user through the base station. Homeowners can then use the information to adjust their power consumption habits, while the control logic uses the information to detect over-current and over-voltage situations and change the switch's status accordingly.

7.2.2 System Diagram

The above diagram shows the components of the individual monitors and their relationships to each other and the breaker section of the smart breakers.

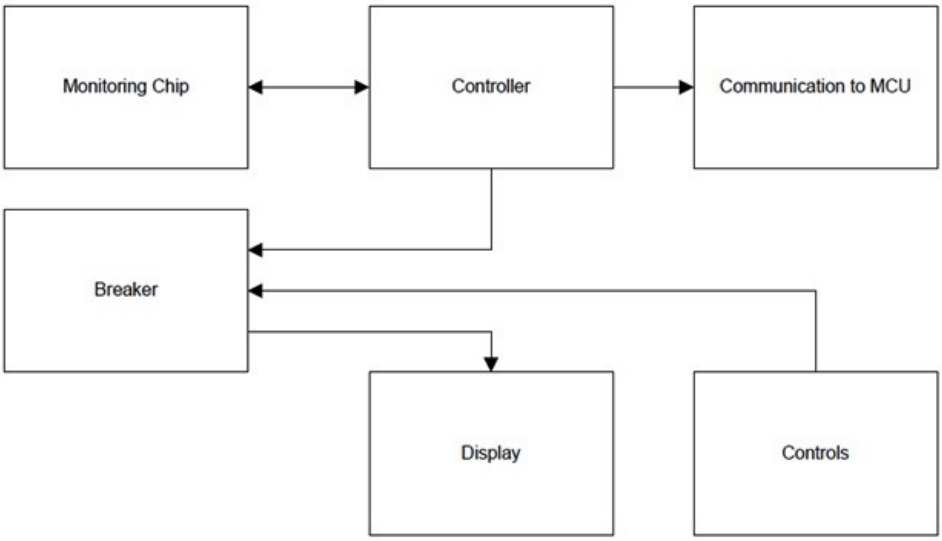

Figure 16: Block diagram of monitioring system

7.2.3 System Components

The purpose of the metering chip and its inputs is to measure instantaneous voltage and current and pass the data to the controller.

At the beginning of the year, the team found ADE7763 metering chips from Analog Devices and obtained free samples. As an IC designed for energy metering [ADE datasheet], the chip has all of the functionality needed for the project. At a minimum, the chip needs to read voltage and current with additional measurements being nice but not necessary. The team also found several other chips that perform similarly, but due to cost, the team decided to use the ADE7763 metering chips.

Input methods for the ADE7763 were selected based on the AN 564 app note, using a current transformer and voltage divider to measure the power line. Both work well for the project because they are simple and accurate. The current transformer is more expensive than the team hoped, but was easily available and the voltage divider is both easily available and low cost. Selected components and values keep the input levels lower than the maximum value of 500mV for the ADE7763 channels using the equations below.

$$V_{secondary} = \frac{(I_{primary} \times R_{burden})}{2511} \tag{1}$$

This gives the voltage across the secondary of the current transformer and comes from the datasheet for the current transformer From this equation, R_{burden} is selected as 60Ω, allowing the current transformer to be

used for up to 20A.

$$V_{measure} = V_{line} \times \frac{R_{measure}}{R_{total}} \tag{2}$$

$$P = \frac{V^2}{R} \tag{3}$$

The first equation is the standard voltage divider equation used to determine values so the maximum input voltage is not exceeded. The second equation is the power equation used to determine resistor values so they don't burn up. To keep $V_{measure}$ under 500mV, $R_{measure}$ is 1kΩ with an R_{total} of 511kΩ.

$$f_{3dB} = \frac{1}{2\pi(R)(C)} \tag{4}$$

This equation is used to determine the 3dB point of the attenuation network in Figure 17 [1]. A 22nF capacitor with the $R_{measure}$ from above gives a cutoff of 7200Hz as suggested by the app note [1].

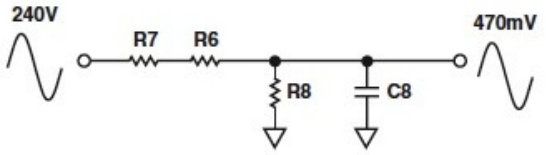

Figure 17: Attenuation network for ADE7763 [1].

The microcontroller's job is to transfer the data stored in the metering chip's registers to the base station and does so through the SPI connection to the chip that's provided. It also provides the logic needed to detect and turn off the breaker in an over-voltage or over-current. Other functions include managing start up and shut down, primarily initializing registers and saving data.

First semester, the team decided to use an FPGA for this purpose as they are very flexible and can easily manage data from the metering chip. However, the design process for the rest of the breakers took longer than anticipated and the team decided to use the Altera Nios II processor available from Calvin. Calvin also had some pic microcontrollers, but the team is not familiar with them and would have had to learn new protocols and language, making the design more complex than the Nios II which had been used in other classes before. After working with the Nios II, the team discovered there is very little documentation for the functions needed, making it very difficult to work with. The team decided to look at possible alternatives

and found an Arduino Uno that uses an ATmega328 microcontroller and has more and clearer documentation.

Figure 18 shows the microcontroller's operating process for normal operation and what it does in both an overvoltage and overcurrent situation. Voltage and current are both necessary to calculate power, so the team focused on reading those values first. Other features of the ADE7763 may be used if the project were given more time, but as they're not critical, the team didn't deal with them since the ADE7763 makes it easy to ignore information it collects.

Figure 19 is an expanded view of the read operation in Figure 18 and includes the write process in addition to the read process for communicating with the ADE7763.

The purpose of the communication component is to transfer data from the smart breakers to the base station and is controlled by the microcontroller.

First semester the team decided to use Ethernet to achieve this, based on compatibility with the base station, low cost and ability to connect several breakers. However, longer than expected design time for other aspects of the project made it necessary to use a simpler method. The team chose XBee radios instead, which use RS232 to connect, lowering the design time. XBee also removes the need to have external wires connecting the breakers and base station, making the product more aesthetically appealing to the customer.

The purpose of the display is to provide the user with basic on/off information relating to the breakers should the system fail and require a manual shutoff. This meant it must be simple and easy to understand, which led to the decision to use well-labeled LEDs. Because LEDs are very easy to include in a schematic, using them instead of a character-based display also reduced design time.

Because it can be critical to shut off power, the team decided to include a physical failsafe method in addition to the system's ability to do so automatically. Because of the way the automatic switching is configured, this backup switch would only allow the user to turn power off, will not be able to override the controls and force an over-voltage or over-current situation.

7.2.4 Final Decisions

To summarize, the individual circuit monitors consist of an Arduino Uno microcontroller and Analog Devices ADE7763 metering chip with input networks using a voltage divider and current transformer. The

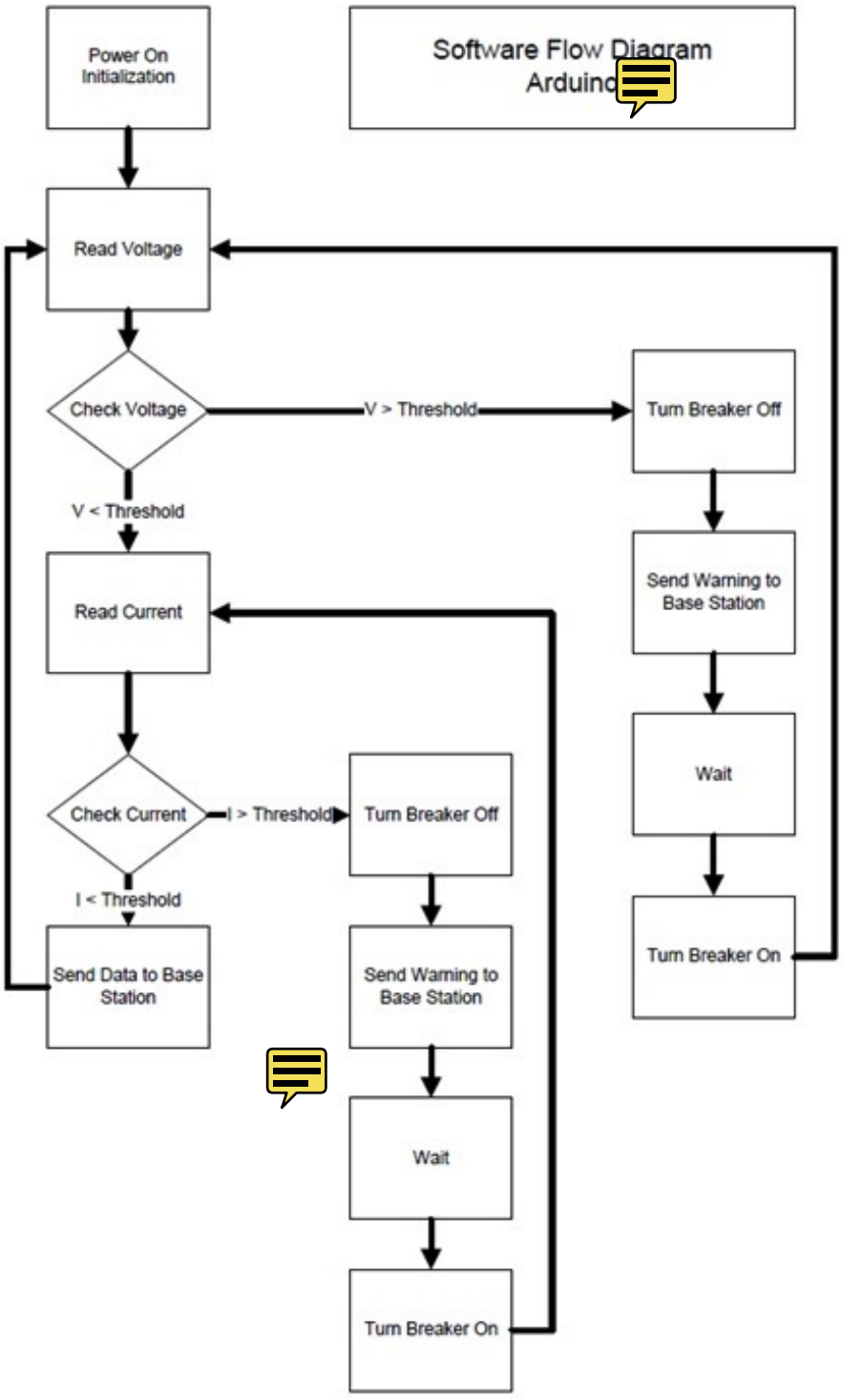

Figure 18: Software flow diagram for Arduino microcontroller

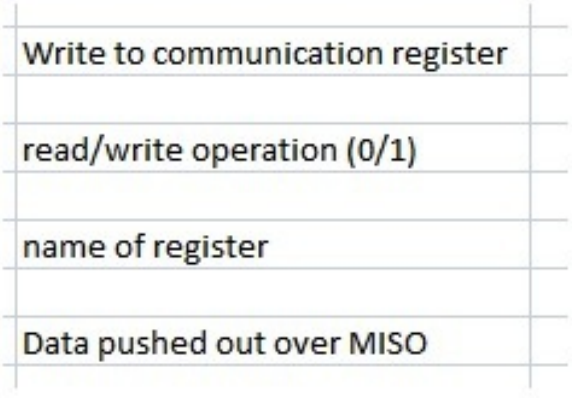

Figure 19: Read/write process for ADE7763

unit will communicate with the base station computer through the Arduino to computer interface directly, using SPI for communication between the ADE7763 and Arduino Uno. Minimal on/off displays and controls including LEDs and a physical switch will be provided to ensure user safety as well. The system will use a solid state relay to demonstrate that the microcontroller is able to control a switching device, but will use a yet to be determined technology for a marketable product.

7.2.5 Printed Circuit Board

7.2.6 Testing

The table and graph below show the results of current transformer testing. During testing, a 6.2kΩ burden resistor was used instead of the 60Ω selected for final design to make it easier to read results using an oscilloscope. Table 1 gives the component values used and the resulting voltage and current measurements, both with percent error from values calculated based on measurements on the primary side.

Table 2: Measured:Primary

Desc.	Voltage(V)	Power (W)	Resistance (Ω)	Current (A)
1 S	117	40.00	342	0.360
2 S	117	20.00	684	0.250
3 S	117	13.33	1027	0.210
4 S	117	10.00	1369	0.200
2 Prll	117	80.00	171	0.690

Table 3: Measured:Secondary with Percent Error

Desc.	Voltage(V)	% Error	Resistance (Ω)	Current (mA)	% Error
1 S	0.840	5.5	6200	0.135	5.8
2 S	0.580	6.0	6200	0.093	6.6
3 S	0.490	5.5	6200	0.084	0.4
4 S	0.440	10.9	6200	0.069	13.4
2 Prll	1.620	4.9	6200	0.265	3.6

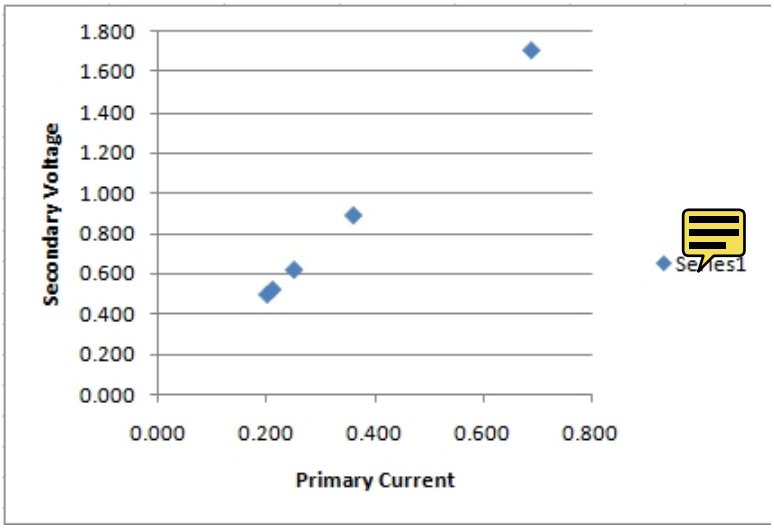

Figure 20: Current-Voltage Response of Current Transformers

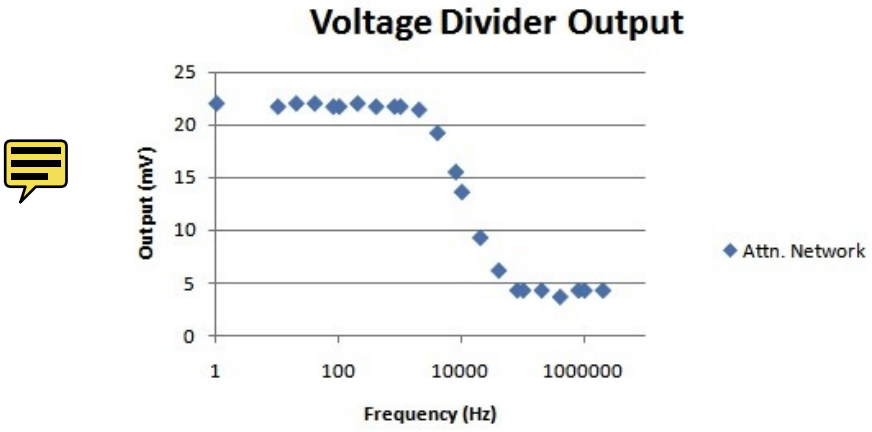

Figure 21: Graph showing results from voltage divider testing

7.3 Final Decisions

To summarize, the individual circuit monitors consist of an Arduino Uno microcontroller and Analog Devices ADE7763 metering chip with input networks using a voltage divider and current transformer. The unit will communicate with the base station computer through the Arduino to computer interface directly, using SPI for communication between the ADE7763 and Arduino Uno. Minimal on/off displays and controls including LEDs and a physical switch will be provided to ensure user safety as well. The system will use a solid state relay to demonstrate that the microcontroller is able to control a switching device, but will use a yet to be determined technology for a marketable product.

7.4 Future Work

While the system appears to work well for a prototype, there is still much that needs to be completed before a final product can be released. Some things specific to the individual circuit monitoring include testing with complex loads, designing communications for several units at the same time, effective solid state technology for switching purposes and selecting/implementing a different metering chip and microcontroller.

Adjusting the design for more complex loads should be easy and will likely include adjusting the input networks to the metering chip for more precise measurements and selecting appropriate parameters for the chip.

Designing a good solid state breaker to fit in the small space provided and within a reasonable price range will be a challenge. Without changing the design of a typical breaker box, the limited access to cooling and the voltage and current thresholds required make this a very difficult step forward.

Right now, the prototype monitor is set up so that a single monitoring chip reports directly to the computer. However, in a typical home, multiple devices are expected to be used simultaneously, requiring a more complicated process for passing data to ensure no data is lost. Assuming a system with minimal data transmission lines, the team recommends the use of a Master Control Unit to manage the monitoring devices prior to transmitting data back to the base station. Another concern is the bandwidth and ability to handle real-time data from over a dozen devices. If the user desires high precision and frequent readings from many monitoring devices, a single ZigBee system (currently the most likely option) will not be sufficient.

For production purposes, implementing a different metering chip is another critical stage. The ADE7763 from Analog Devices works well as a proof of concept, but is not the most cost effective solution. Another possible benefit is that the microcontroller will work better with the chip its designed to work with. This should help utilize more of the available functionality in the metering chips, continuing to provide the customer with more information.

8 Circuit Identifier

8.1 Purpose

This device should allow users to accurately map individual outlets to a specific circuit/breaker, by temporarily using/manipulating power from that outlet, which means that it will not be permanently plugged into the circuit.

8.2 Requirements

1. Shall run off of 120V line
2. Shall use or manipulate power for a time interval such that DSP can recognize the device is “on”.
3. Shall not adversely affect other devices on the circuit.

8.3 Design Criteria

1. Should be user friendly/intuitive
2. Should be easily portable
3. Efficiency is not a concern
4. Cost
5. Overall system should require minimal user input

8.4 Ideas

8.4.1 First Idea Descriptor

The Identifier must have a master controller (or something similar that is back at the service panel) identify the circuit into which it is plugged. Something that might work is an X-10 power line control module or similar. There are some people who have a device that allows you to turn on and off lights remotely. There is a device you plug in between the socket and the lamp that receives the signal. There is also a controller you plug in to any plug (not sure the plug has to be on the same circuit, but that does not matter in this

purpose). The controller sends a high frequency signal down the line that the receiver intercepts and interprets and an on/off/dim signal. Therefore, the control box could be the thing the team is looking for. The control box is basically a box with a few buttons and a 120-VAC cord. You plug it in and press a button. The controller at the panel sees the high frequency signal come through, figures out which circuit it came from, and we have our detection.

8.4.2 Second Idea Descriptor

There are also things that were found online that might work. One of them is called an Extech RT30 Wireless Remote AC Circuit Identifier. It has a clamp that acts as a probe to connect to wires that will have an indication as to if the circuit is being used or not. The data sheet can be found at http://www.globaltestsupply.com/datasheets/RT30data.pdf

8.4.3 Third Idea Descriptor

The other thing that was found online is called an Automatic Circuit Identifier with Digital Receiver. This is a more of a manual approach to see which circuit is on and which is off. It is not the ideal method, but it is an option nonetheless. The information on this can be found at
`http://www.idealindustries.com/media/pdfs/61534_manual.pdf`

8.5 Decision

The control box idea was a great idea, but the team did not believe they had enough time to incorporate the design into their project because it was not ready for use directly out of box. It would require more design work than the team had time for. Therefore the team decided to go with idea number 3, which was the Automatic Circuit Identifier with Digital Receiver. It would work very well out of box, and provide the team with the functionality needed to complete this part of the project in a timely fashion.

9 SPARC-Based Embedded Linux System

The original design for the base station called for a self-contained Linux system to collect and process the measurements from the other PICA systems. To accomplish this goal, the design team decided to use an FPGA development board and program it will the SPARC-compatible LEON3 soft-processor, which is licensed under the GNU General Public License (GPL). While this option showed a great deal of promise, it ultimately lead to more obstacles than the team decided could be overcome without detracting from the other areas of the project. In light of this decision, the role of the base station will be fulfilled by a standard desktop computer using custom software to perform the tasks that the embedded system would have performed.

9.1 Device Selection

As the base station would have collected and formatted the measurements taken from the other PICA devices, it would have required a fair amount of processing power to perform its duties. While a modern desktop computer would provide more than enough computational ability, the design team focused on creating a single-purpose embedded Linux system that would perform the necessary tasks without competing with the processes a user might run on a home computer. In order to build such a device, the design team required hardware that was flexible enough to allow extra devices to attach to it, but would also have features like networking built-in.

9.1.1 Board Selection

In their search for suitable hardware, the design team discovered an FPGA development board that a previous senior design team had ordered for their own project. This development board, a Digilent ML-509 educational board, included a 9-pin RS232 serial port, a standard networking port, a compact-flash reader, and many configurable pins for attaching extra devices. In addition to these peripherals, the board featured a Xilinx Virtex 5 FPGA, which could be configured into a wide variety of different processors or systems for which a standard hardware description could be found. Upon finding that these features met the requirements for the base station, the design team selected the board for making a prototype of the base station.

9.1.2 Processor Selection

Having selected the development board, the design team looked at several different options for soft-processors. The team settled on the SPARC-compatible LEON3 processor, which is licensed under the GPL, as it seemed to provide an outstanding all-in-one package. In addition to the processor itself, the LEON3 package included interfaces for the development board's network, display, and serial peripheral controllers. The LEON3 vendor, Gaisler-Aeroflex, provided a host of cross-compilers and tools to interface with and debug the processor under free or demo licenses. They also advertise a Linux distribution containing a variety of utilities and programs. With all these tools available, the design team felt certain that the LEON3 would provide everything required of the base station.

9.2 Building the LEON3 Processor

In order to use the LEON3 soft-processor, the FPGA required a configuration file generated from the LEON3 hardware description files. As the FPGA was a Xilinx part, the design team downloaded ISE, the Xilinx software suite for creating part-specific configurations from the description files. Although Xilinx provides a "Webpack" edition of ISE free of charge, its functionality is very limited. In fact, the synthesis tools for the specific FPGA on the board, the XC5VLX110t, were explicitly disabled without a license for ISE. To work around this issue, the team requested a thirty-day evaluation license from Xilinx, which unlocked the required synthesis tools. processor on the ML-509 board.

9.2.1 Configuring the Source Files

The LEON3 hardware description source files came with pre-made configurations for many different boards, including one for the ML-509. As such, this configuration could be used directly without requiring further changes, although the automated process took more than an hour to complete. When this task completed, it produced a bitfile that configured the specific model of FPGA to behave like the LEON3 processor. The design team later discovered that Gaisler had a functionally similar bitfile available for direct download.

9.2.2 Programming the FPGA

In order to use the Xilinx tool to program the FPGA, the workstation computer needed access to the development board's configuration. Xilinx provided a programmer pod, a Universal Serial Bus (USB) device to access the FPGA's configuration using the JTAG protocol, for exactly this purpose. Once fitted with the appropriate drivers, the workstation successfully detected the pod. The Xilinx program `impact` wrote the bitfile into the FPGA, transforming it into the LEON3 processor.

9.2.3 Managing the LEON3 Processor

Gaisler-Aeroflex provided an evaluation version of their `grmon` tool to connect to the LEON3 for loading and debugging code on it. As described in its manual, `grmon` could connect to the processor using the same JTAG pod that `impact` used to program the device. This claim held true, and `grmon` promptly gave a summary of the system it found there.

9.3 Extending the LEON3

While Gaisler-Aeroflex provided the core LEON3 and its interfaces to hardware under the GPL, they do not distribute an Floating Point Unit (FPU) under the same license. As such, on the default LEON3 system, any float-point operations would be emulated in software, rather than performed in hardware. As the intended purpose of the base station would likely involve operations with floating-point numbers, including a hardware FPU would likely benefit the performance of the base station.

While the LEON3 configuration tools include a section for adding an FPU to the system, they did not include the description files for such a device. Instead, Gaisler provided its own FPUs as a separate download. By separating them from the GPL-licensed source, Gaisler could distribute the FPU in a form that could be included into the LEON3 system without exposing their source description files. Once these files were added into the project, the LEON3 configuration tool could include them into the next bitfile. The design team created and loaded the bitfile onto the FPGA, and asked `grmon` to display information about the system. Its output is given in the appendix, listing 4.

9.4 Running Gaisler Linux on the LEON3 Processor

As previously mentioned, Gaisler-Aeroflex provided a customized Linux distribution for the LEON3, which is based on of the SnapGear embedded Linux distribution. They also provided a pre-built compiling tool chain based on GNU Compiler Collection (GCC). Using these two together should have produced a small Linux system for the LEON3. The critical components of the system, such as the Linux kernel and the LEON3 bootloader, functioned without much extra tweaking. An example of the messages printed by the Linux kernel over a serial connection may be found in the appendix, listing 5.

Gaisler provided a configuration tool for their distribution that behaved very similarly to their previous configuration tools. This tool not only allowed for configuring the kernel, but also for creating an entire Linux system around it, complete with a shell and a wide range of utilities and applications. Unfortunately, very few of them worked without tinkering. Some options could not be built at all, others could be built with a sufficient amount of tweaking, but even then the configuration files for these programs did not install correctly into the distribution's miniature file-system. The design team tried multiple times to get the system working, but when the distribution very strongly resisted running a DHCP client, the team decided to try a different approach. Message logs with these failure can be found in the appendix, listings 6 and 7.

9.5 Building Linux from Scratch for the LEON3

In order to build a Linux system without dealing with Gaisler's distribution, the design team changed focus to building a Linux distribution from source code. The team referenced the guide at `http://cross-lfs.org/view/clfs-embedded/arm/`, and adapted it for the LEON3 SPARC system. While this process ran quite smoothly, it ultimately failed when trying to boot on the LEON3, where the system stopped at a break point even when instructed to continue past it. The team's modified guide is given in the appendix, section C.

9.6 Final Decision

At the time of this writing, the design team believes that the time required to get a LEON3-powered base station would be better spent on other areas of the project. As the ZigBee radio receivers work well on desktops, the team intends to create a monitoring application to run on a ZigBee-enabled home computer.

While this does not allow for an always-on dedicated device, it provides similar functionality using a device that is familiar to most home-owners.

10 Power Supply

10.1 Requirements

All requirements under this heading are to be assumed to be of the power supply (“the system”) unless explicitly stated otherwise.

1. Shall be 78% efficient.
2. Shall power 2 ADE devices.
3. Shall provide 5VDC +/- 5%.
4. Shall provide 2mA, 3mA (total 5mA) plus a Safety factor.
5. Shall also power an FPGA that will control these chips in a production-scale design at 5VDC $\pm$5%.
6. Shall also power the FPGA at 1.5A.
   (a) While the exact requirement for the FPGA are unknown the current was chosen much higher than probably needed in order to make sure the power supply would not need to be changed to provide more power.
7. Shall have a ripple voltage of 1%.
8. Shall be isolated from mains supply by a high frequency transformer.
9. Shall be small and not exceed 4in^3.

10.2 Design Criteria

1. Size and weight: These could conflict however, if the size is no bigger than 4 cubic inches and all of the parts are surface mount, the heaviest part is the transformer. The power supply will not exceed 3 pounds.
2. Output voltage: The output voltage will need to be sufficient for various loads. It will need to still remain at 5V
3. Efficiency: This is very important. Linear power supplies are not very efficient, which is why the team decided to build a switched-mode power supply. They are generally 30% more efficient than

linear power supplies.

4. Heat: Shall not be hot enough to where heat sinks are required.
5. Power Dissipation: Power dissipation shall be low.
6. Complexity: Switched-mode power supplies are much more complex than linear. The complexity of the design should not get to the point where it would need explanation for an engineer in the field.
7. Electronic noise at input and outputs: The electronic noise at input and outputs shall not get to the point where compensation for noise is needed.

10.3 Alternatives

10.3.1 Buck converter

"The input voltage is converted into a lower output voltage."[13]

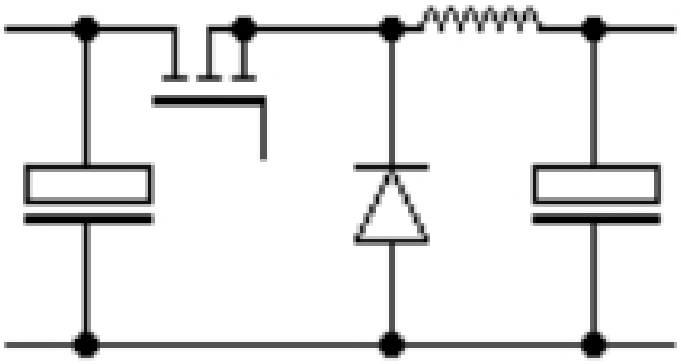

Figure 22: Buck Converter

10.3.2 A boost converter

"The input voltage is converted into a higher output voltage"[13]

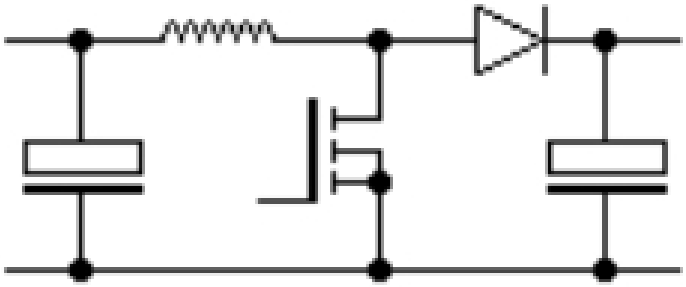

Figure 23: Boost Converter

10.3.3 A buck-boost converter

"The input voltage is converted into a negative voltage"[13]

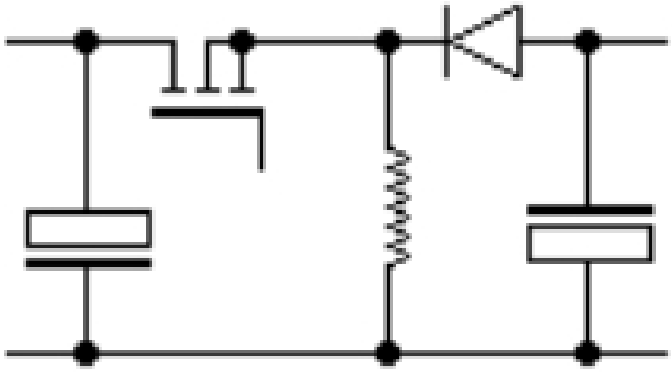

Figure 24: Buck-Boost Converter

10.3.4 A flyback converter

"Several isolated output voltages, up to approximately 250 are possible"[13]

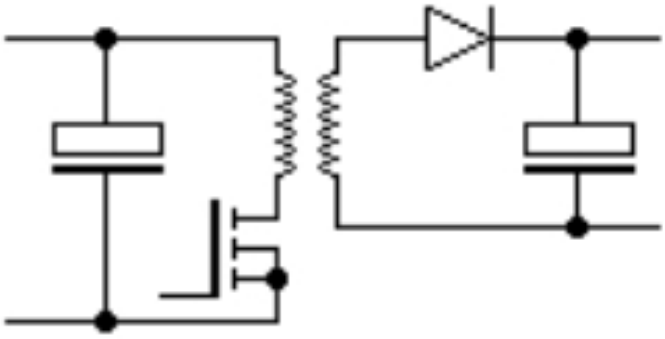

Figure 25: Flyback Converter

10.3.5 Single Transistor Forward Converter

"One electrically isolated voltage, up to approximately 100 Watts."[13]

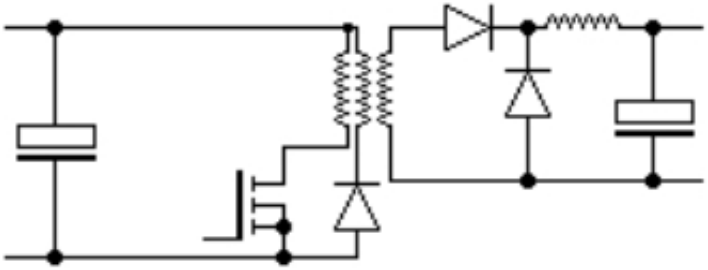

Figure 26: Single Transistor Forward Converter

10.3.6 Two Transistor Forward Converter

"One electrically isolated voltage, up to approximately 1kW."[13]

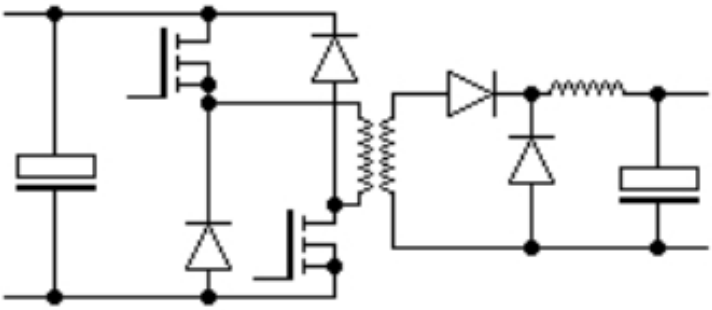

Figure 27: Two-Transistor Forward Converter

10.3.7 Half-Bridge Push-Pull Converter

"One electrically isolated voltage, up to few kilo watts."[13]

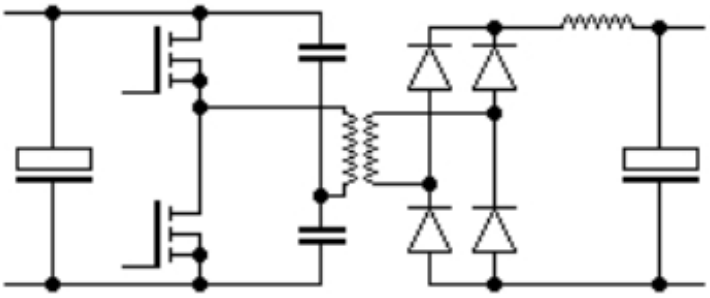

Figure 28: Half-Bridge Push-Pull Converter

10.3.8 Full-Bridge Push-Pull Converter

"One electrically isolated voltage, up to many kilo watts."[13]

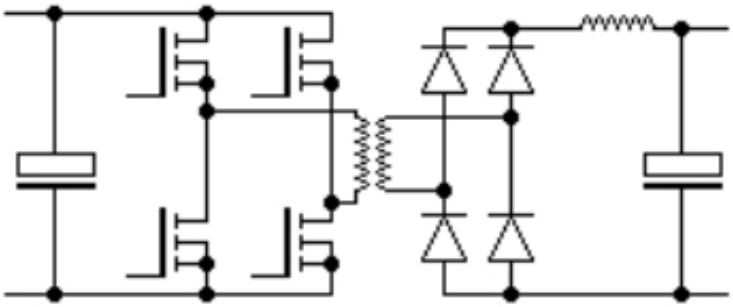

Figure 29: Full-Bridge Push-Pull Converter

10.4 Rating the Alternatives

Once the team decided on a switched-mode supply, we needed to select a topology to use. The choices include a buck converter, a boost converter, a buck-boost converter, a flyback converter, single- and two-transistor forward converters, and half- and full-bridge push-pull converters. Some of these topologies can be eliminated immediately because the design will be too complex than needed to meet the needs of this design. For example, a boost converter takes the input voltage and converts it to a higher output voltage, which does not fit line voltage being converted to 5V DC. Similarly, a buck-boost converter outputs a negative voltage, which is also not fitting for this application. The single-transistor forward converter, two-transistor forward converter, half-bridge push-pull converter, and full-bridge push-pull converter have one electrically isolated voltage up to a varying number of watts. This would work for our design, however they are more complex than needed, other designs will work and be less complex. After these eliminations, the two remaining topologies are the flyback and the buck converters. The buck takes an input DC voltage and converts it into a lower output DC voltage, while a flyback can produce several isolated output voltages from an input AC voltage. Using a transformer and rectifier bridge can adapt an AC input into a DC voltage, allowing the buck topology to work where the flyback topology could also be used. Either design is feasible for the purpose specified. The flyback would use an inductor, while the buck uses a transformer. The team has used transformers for these purposes before, which would make the design easier. A flyback also can have multiple isolated voltages, but this was not needed in the design, so the team did not feel like this design was best. The team decided on a Buck Converter. Additionally, Johnson Controls Incorporated graciously donated parts for a buck converter, as well as expert advice on designing and assembling one. As none of the members of the team have made a SMPS before, it felt very nice that the team could have an expert to go to for advice on a chip and design he has worked with several times.

10.5 Implementation

10.5.1 Scope

This document describes the analysis of the 5V/2A, LM25011-Buck Converter.

10.5.2 Module Overview and Block Diagram

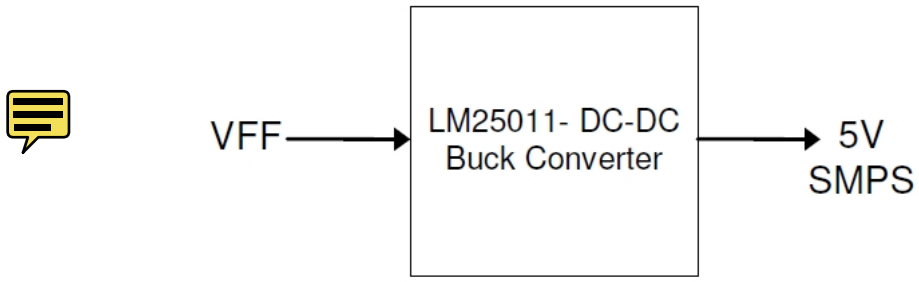

Figure 30: Buck Conversion Block Diagram

10.5.3 Schematics

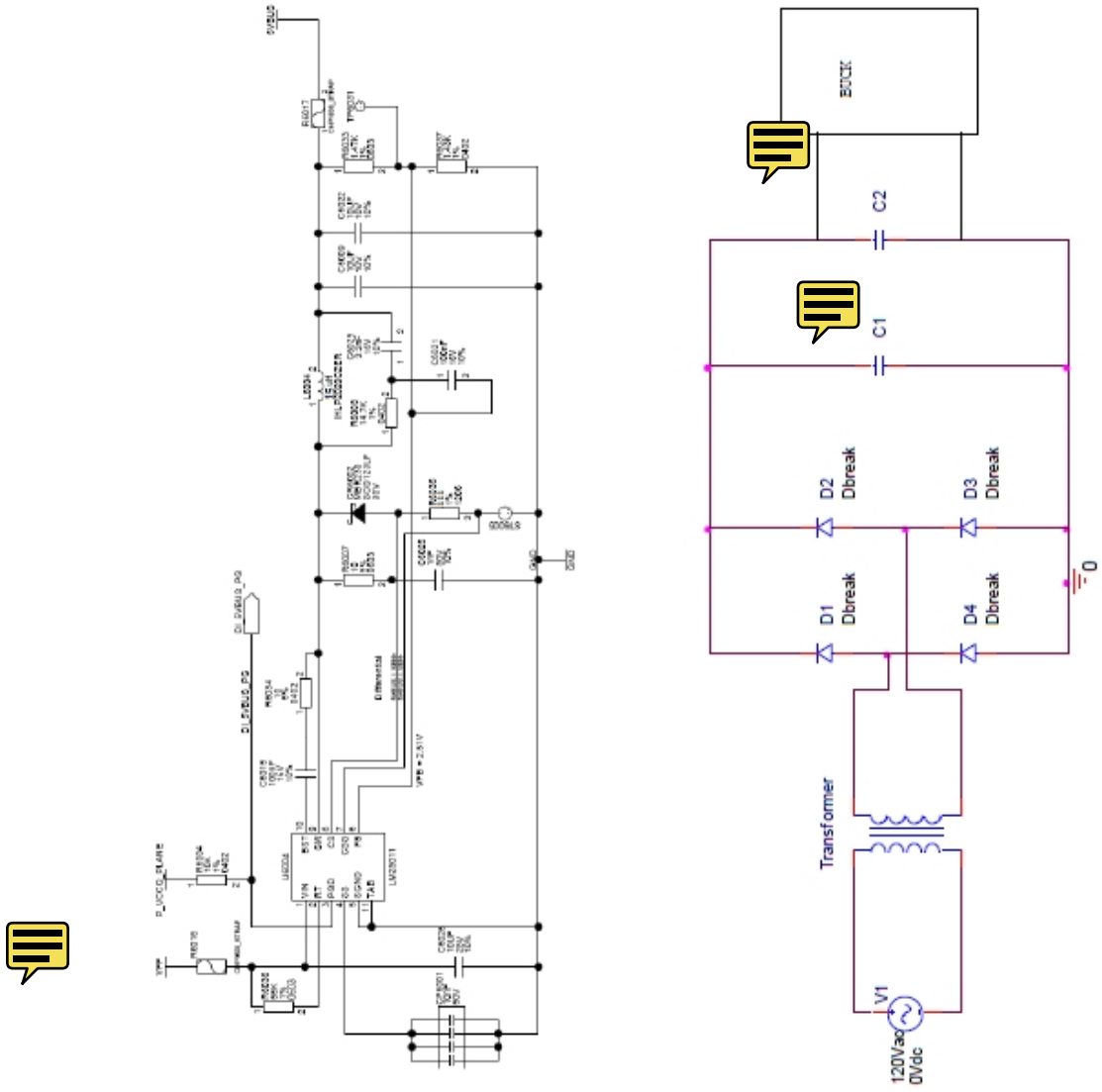

Figure 31: Schematic of Power Supply as Buck Converter

10.5.4 Calculations

This power supply will require a transformer and four diodes as shown in the schematics. These equations are for that part of the schematic.

$$V_{AC} = 18\text{V} \tag{5}$$

This is the voltage the transformer will output from a 120 V_{AC} outlet.

$$Transformer_{peak} = VAC\sqrt{2} \tag{6}$$

Calculating for the rectifier circuit. This comes from the equation for finding V_{rms}, which is

$$V_{rms} = \frac{V_p}{\sqrt{2}} \tag{7}$$

$Current = 2$ This is the desired output current.

$$dt = \frac{1}{120} \tag{8}$$

is the duration between charging cycles and is equal to

$$\frac{1}{2 * Main\ \ Frequency} \tag{9}$$

Therefore since in America the main frequency is 60Hz,

$$dt = \frac{1}{120} \tag{10}$$

$$Overhead = 1.4 \tag{11}$$

$$dv = Transformer_{peak} - Overhead - 8 = 16.056 \tag{12}$$

We need this 8V for the switching regulator, it needs 3V of headroom and since we want 5V out we take the voltage +3V and get 8V, but for a little bit of wattmark.

$$Capacitor = Current\frac{dt}{dv} \tag{13}$$

$$Capacitor = .001\text{F} \tag{14}$$

This simply means that the capacitor(s) for filtering need to be at least 1mF.

Note:This is a minimum value This value could be achieved by one capacitor or multiple as long as the total capacitance is at least 1mF.

The diodes were selected based on the fact that Chuck Howerda gave them to our team and said they would work well for this application and for the diode bridge.

10.5.5 Capability Justification Table

Table 4: Power Supply Capabilities

Capability Parameter	Specification	Analysis Result	Remarks
Input Voltage	Min	7.09V	From Current Draw
Input Voltage	Max	17.10V	From Current Draw
Switching Frequency	Max	3.38MHz	From Current Draw
Output Voltage	Min	4.99V	From Calculation
Output Voltage	Max	5.19V	From Calculation
Output Ripple Voltage	Max	10mV	Assumed
Output Current	Max	2A	From Current Draw
Output Inductor	Typ	2.2µH	From Schematic
Power Dissipation across Inductor	Max	99mW	From Calculation
Power Dissipation across current sense resistor during current limit mode	Max	246mW	From Calculation
Efficiency	Max	86.86%	From Calculation
Softstart Time	Min	8.032ms	From Calculation
Softstart Time	Max	12.048ms	From Calculation
Input Under Voltage Protection Threshold	Min	4.6V	From Datasheet
Thermal Shutdown	Typ	155°C	From Datasheet
Line Regulation	Max	5 +/- 5%	Verify by Testing

Continued on next page

Table 4: Continued from previous page

Capability Parameter	Specification	Analysis Result	Remarks
Load Regulation	Max	5 +/- 5%	Verify by Testing
Gain Margin			Verify by Testing
Phase Margin			Verify by Testing

This table was co-produced with Hilbrand Sybesma, a hardware engineer at JCI. After we were both done working on January 27, 2011, he took the time to work with me on them. As I was creating it, he stopped me if I made any errors and he told me other things to include. He went above and beyond with help.

10.5.6 Theory of Operation

"The LM25011 Constant On-time Step-Down switching regulator is capable of supplying up to 2A of load current. This high voltage regulator contains an N-Channel Buck switch, a startup regulator, current limit detection,and internal ripple control. The constant on-time regulation principle requires no loop compensation, results in fast load transient response, and simplifies circuit implementation.

The operating frequency remains relatively constant with line and load. The adjustable valley current limit detection results in a smooth transition from constant voltage to constant current mode, when current limit is reached, without the use of current limit foldback. The PGD output indicates the output voltage has increased to within 5% of the expected regulation value.

Additional features include: Low output ripple, VIN under-voltage lock-out, adjustable soft-start timing, thermal shutdown, gate drive pre-charge, gate drive under-voltage lock-out, and maximum duty cycle limit."[14]

10.5.7 Design Assumptions

VFF value is considered as min. 7.09V and max. 17.10V from Current draw of the chip in use.

10.5.8 Analysis

1. Input voltage range:

$$P_{VFF} = 7.09, 12.3, 17.1\text{V} \tag{15}$$

   (a) Input Voltage range for the LM25011 is 6V to 42V 12.3V was choosen for a medium between 7.09 and 17.1.

2. Output Voltage

$$P_{5V} = 4.75, 5, 5.25\text{V} \tag{16}$$

   (a) These values are from the 5%tolerance.

3. Load Current

$$I_{out} = 2\text{A} \tag{17}$$

   (a) Note: As per datasheet, LM25011 can deliver up to 2Aload current

10.5.9 Output Voltage Calculation

Calculations from datasheet:

$$V_{ref} := \begin{pmatrix} 2.46 \\ 2.51 \\ 2.56 \end{pmatrix} \cdot V \qquad R_{fb1} := \begin{pmatrix} 1 - 1\% \\ 1 \\ 1 + 1\% \end{pmatrix} \cdot 1.43 \cdot K\Omega$$

$$R_{fb2} := \begin{pmatrix} 1 - 1\% \\ 1 \\ 1 + 1\% \end{pmatrix} \cdot 1.47 \cdot K\Omega$$

$$V_{out} := \overrightarrow{\left(V_{ref} \cdot \frac{R_{fb1} + R_{fb2}}{R_{fb1}} \right)} \qquad V_{out} = \begin{pmatrix} 4.989 \\ 5.090 \\ 5.192 \end{pmatrix} V$$

$$V_{out\_nom} := V_{out_1} \qquad V_{out\_nom} = 5.090\,V$$

Figure 32: Calculation from datasheet

10.5.10 Selection of Switching Frequency

Information from datasheet:

The maximum allowed frequency may be limited by the minimum and maximum input voltage, as described below:

1. Operation at a low input voltage requires a high duty cycle to maintain output regulation. The maximum duty cycle is limited by the minimum off-time of the LM25011 (150ns typ., 208 ns max). The output voltage drops out of regulation at low VIN if the switching frequency does not satisfy the following condition:

$$Fsw < (Vin(min) - Vout)/(Vin(min)x208ns) \tag{18}$$

2. Operation at high input voltage requires a low duty cycle, limited by the minimum on-time of the LM25011 (90 ns). The switching frequency must satisfy the following condition to ensure the on-time is greater than 90 ns:

$$Fsw < Vout/(Vin(max)x90ns) \tag{19}$$

If the selected value of Fsw does not satisfy this condition, the LM25011 maintains regulation, but the switching frequency will be lower than the desired value.

$$f_{s\_max\_con1} := \frac{\min(P\_VFF) - V_{out\_nom}}{\min(P\_VFF)\cdot 208\cdot ns} \qquad f_{s\_max\_con1} = 1.356\cdot MHz$$

$$f_{s\_max\_con2} := \frac{V_{out\_nom}}{\max(P\_VFF)\cdot 90\cdot ns} \qquad f_{s\_max\_con2} = 3.307\cdot MHz$$

Figure 33: Condition from datasheet

From the conditions shown, maximum switching frequency should not exceed 1.356MHz.

The approximate operating frequency is calculated as follows:

$$RT_{used} = \begin{pmatrix} 1-1\% \\ 1 \\ 1+1\% \end{pmatrix} \tag{20}$$

$$V_in = P_{VFF} \tag{21}$$

$$f_s = \frac{V_{out->nominal}}{(V_{in}15ns) + [4.1 * 10^{-11}(RT_{used} + .5\text{k}\Omega)\text{As}]} \tag{22}$$

$$f_s = \begin{pmatrix} 2.121 \\ 2.035 \\ 1.961 \end{pmatrix} \tag{23}$$

$$f_{sMAX} = 2\text{MHz} \quad \text{From Datasheet} \tag{24}$$

For the selected RT value, switching frequency is higher than 2MHz. However after speaking with the contact from National Semiconductor I have confirmed that LM25011 can operate up to 2.4MHz.

10.5.11 Selection of Inductor

As per JCI recommendations as well as National Semiconductor, we opted to use a 15µHinductor since they said it is the best to use because we will want more than the minimum requirement.

10.5.12 Ripple Current

1. Maximum Ripple Current

$$I_{ripple\_max} = (max(V_{in}) - min(V_{out}))(\frac{min(T_{on\_min})}{min(Lrecom)}) \tag{25}$$

   (a) Note: This is acceptable.

$$I_{ripple\_max} = 294.851\text{mA} \tag{26}$$

2. Nominal Ripple Current

$$I_{ripple\_nom} = (V_{in_1} - V_{out})(\frac{min(T_{on\_min})}{min(Lrecom)}) \tag{27}$$

$$I_{ripple\_nom} = 175.525\text{mA} \tag{28}$$

3. Minimum Ripple Current

$$I_{ripple\_min} = (min(V_{in}) - max(V_{out}))(\frac{min(T_{on\_min})}{min(Lrecom)}) \quad (29)$$

$$I_{ripple\_min} = 95.321\text{mA} \quad (30)$$

4. Inductor Peak Current

$$I_{peak} = I_{out\_max} + \frac{I_{ripple\_max}}{2} \quad (31)$$

$$I_{peak} = 2.147\text{A} \quad (32)$$

10.5.13 Minimum Valley Threshold for Current Limit

$$I_{valley\_th} = I_{out\_max} - \frac{I_{ripple\_min}}{2} \quad (33)$$

10.5.14 Current Sense Resistor

As long as we have a current sense resistor that does not exceed RS_{max} (located in the datasheet) it will be okay. The more current we want in our power supply the smaller our current sense resistor will need to be. Which we can vary as we see fit. The proportion to which this varies depends on the ripple voltage and the calculation for which is shown below.

10.5.15 Ripple Voltage

$$RS_{ripple\_min} = min(RS_{used})min(I_{ripple\_min}) \quad (34)$$

$$RS_{ripple\_min} = 10.381\text{mV} \quad (35)$$

As per datasheet, minimum ripple voltage across current sense resistor should be greater than 10mV. In this case it is very close, but that should not be a problem.

10.5.16 Power Dissipation

We set the current sense resistor to get 1.5A out. Also, by the results of the testing we get 1.3A out instead of 1.5A. We still V out. Therefore we can calculate the efficiency.

$$Eff = \frac{P_{out}}{P_{in}} \tag{36}$$

$$Eff = \frac{1.3 \times 5}{1.5 \times 5} \tag{37}$$

$$Eff = 88.7\% +/- 5\% \tag{38}$$

So, therefore $Eff = 84.23\%,\ 88.7\%,\ 93.1\%$

10.5.17 Efficiency

No adequate information is provided in the datasheet about power loss in LM25011, However to identify approximate power loss in LM25011, the **Efficiency of the LM25011 buck converter is simulated using National Semiconductor Webench tool for the same specification.**[15]

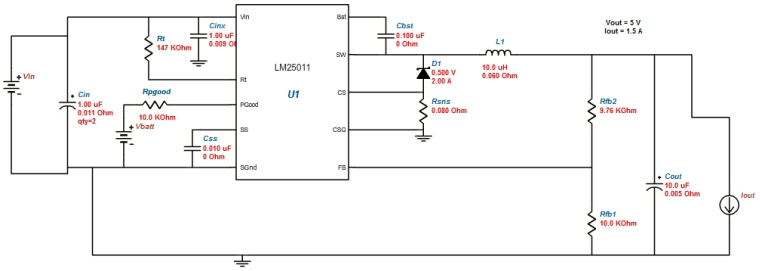

Figure 34: Circuit as shown in Webench

10.5.18 Total Output Power

$$P_{out} = max(V_{out})I_{out\_max} \tag{39}$$

$$P_{out} = 10.383\text{W} \tag{40}$$

$$P_{in} = \frac{P_{out}}{Eff} \tag{41}$$

$$P_{in} = 11.955\text{W} \tag{42}$$

Figure 35: LM25011 Efficiency graph from Webench tool $Eff = 86.85\%$

10.5.19 Approximate power loss in LM25011

$$PD_{LM25011} = P_{in} - PD_{ext} - P_{out} \quad (43)$$

$$PD_{LM25011} = 223.948\text{mW} \quad (44)$$

10.6 Testing

First I tested just the power supply with no load to make sure it gave a 5V output, which was measured with a DMM. That was a success! After which, I put a load of 2.5Ω to see how much current went through. Those values and test setup are shown in the pictures as follows:

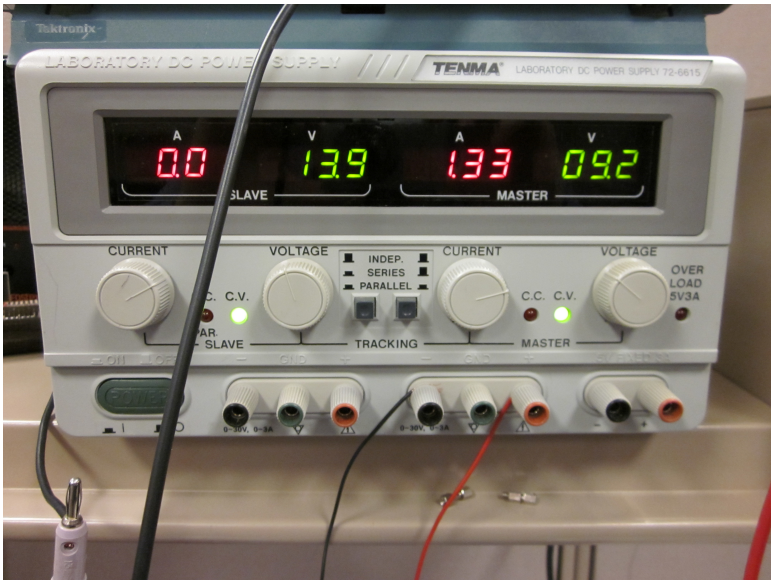

Figure 36: Current and Input Voltage Reading

11 Business Case

11.1 Industry Profile

11.1.1 Overview of the Problem

Standard electric meters were developed decades ago and are still used today, despite many technological advances in the last several years. Along with these technological advances, Americans have become accustomed to having access to large amounts of data, but due to the nature of the standard electric meter, data regarding the usage of power is severely limited. For the power companies, data from the meters is minimal and grid control is limited to manual operation, costing them time and money. As the cost of electricity becomes higher and higher, electricity use in buildings is becoming a bigger concern and people have few cheap or simple ways to monitor this. Of the options available, most only address part of the whole problem, giving some information to the consumer and none to the power company or vice-versa. While there are devices such as breakers and fuses that provide electrical safety for buildings, advances in technology have made it possible to further improve safety but have not been implemented in a cost-effective way or made easily available to an average consumer, which for the purpose of this project shall be defined as a person without a mathematical or scientific education beyond high-school.

11.1.2 Major Customer Groups

The two main customers of the PICA system are power companies and power consumers. The E-meter subsystem will only be sold to power companies, as they must in turn provide metering equipement to their customers. The base station and solid-state breakers will be sold to power consumers who are interested in knowing how much power they use in different regions of their buildings.

11.1.3 Regulatory Requirements

The PICA system must meet certain codes in order to be safe enough for the customer to use, which will also protect from unexpected lawsuits. Underwriters Laboratories (UL) is an independent product safety certification organization, which offers safety certifications to products [16]. In order to gain the confidence of customers, the devices of the PICA system will be UL certifiable. The specific qualifications of UL certification remain unknown to the design team, as the documents regarding the certification requirements

are not publicly available. The system will also restrict EM radiation to comply with FCC Title 47 Part 15. It will also comply with ANSI C12.19 and ANSI C12.21 standards.

While these standards should ensure the general safety of the PICA devices, defects or unforeseen circumstances could imperil users or their property. The PICA system will provide a limited warranty against defects, but cannot be expected to foresee all possible circumstances. To this end, the devices will ship and work with a disclaimer regarding safe operating conditions and the hazards of tampering with the device.

In addition to ensuring the physical safety of the users, the system should also ensure the privacy and security of the users' information. While any wireless link runs the risk of packet interception and capture by a malicious observer, this will only affect the data currently being transferred, and data encryption schemes may greatly hinder these intrusions. The stored data will likely not be encrypted, but will not be actively transmitted: the only means of accessing this data will be through the software controls set in place by the base station or by physically removing the storage medium and removing the data from it. The base station software will use permissions-based file system access and will require a user to authenticate as an administrator before accessing this information. In this way, the user's data will be stored with access controls and will be kept private.

11.1.4 Significant Trends and Growth Rate

Recently, the demand for "smarter" and more informative devices has been increasing with the awareness of resource stewardship and the effects of human activity on the environment. Currently, power companies are investigating smart meters and deploying them to their customers in pilot programs. Power consumers are becoming more energy- and economically-aware, so the time is ripe for providing the products that PICA, LLC. proposes.

11.1.5 Barriers to Entry and Exit

The major barriers to entry is customer recognition. Although power companies may still be testing smart meters before deploying them to their customers, introducing them to a new product might be difficult if they are already near to making a decision. Additionally, power consumers will not be able to buy from PICA, LLC. unless they specifically know of it already.

11.2 Business Strategy

11.2.1 Desired image and position in market

PICA seeks to be known as a leader in making consumers energy aware. This primarily means providing relevant information in an easy-to-understand format that allows users to make informed decisions about their energy usage. Our position in the market will likely be a part of the "green" market, which has been rapidly growing. However, we would like to aim for the consumer oriented part of this market, rather than reaching out to the part of the market dealing with clean power generation.

11.2.2 SWOT analysis

Strengths

One big strength of the product is that in a single system it addresses problems that previously would require multiple, independent systems and products. Another strength is the solid state breakers, which in addition to providing usage information also have a faster response time and include a safe automatic reset option.

Weaknesses

The biggest weakness is the cost of the system. While users want to know the information provided, many are not willing to pay a lot for that information.

Opportunities

The idea of solid state breakers for home usage is relatively new and has not been taken advantage of, so there is a lot of potential growth in that area.

Threats

A big threat is that other companies have already started to establish themselves in similar fields. The PICA system may not provide enough of an advantage over these systems to be successful in an area where consumers have started to use these other systems.

Competitive strategy

As there is a high cost associated with the PICA products, the company will rely less on cost to remain competitive than on the differentiation and focus of the company. The company has already started to differentiate itself primarily through the development of the solid state breakers. The company is also different from many in how we provide solutions to more than one problem the consumer and power company may have. The company hopes to remain competitive by continuing to focus on more aspects of the problems associated with power usage than potential competitors do. Instead of providing a solution relating only to single outlet usage, PICA will provide devices that can all work together, leaving room for easy expansion when the customer decides they want even more information.

11.3 Competitor Analysis

This section analyzes a few products with similar features to various components of the PICA system such as Kill-A-Watt, Cent-a-Meter, The Energy Dectective, and Watts Up? Smart Circuit. The table below gives a overview of these products compared to the PICA system.

11.3.1 Kill-A-Watt

Around the turn of the millennium P3 International introduced the Kill-A-Watt device, which they marketed as a "user-friendly power meter that enables people to calculate the cost of their home appliances," [17]. According to Amazon.com these devices range in price from 52*to*99 Manufacturer Suggested Retail Price (MSRP) depending on features, most notably how many devices can be monitored simultaneously. P3 produces three models of the Kill-A-Watt devices:

1. Kill-A-Watt PS (P4320): A power strip capable of monitoring voltage, line frequency, amperage, KWH, and current leakage for up to eight devices simultaneously and includes built-in surge protection [18].
2. Kill-A-Watt (P4400): The original Kill-A-Watt device, capable of monitoring voltage, amperage, watts used, line frequency, KWH, uptime, power factor, and reactive power for 1 device [19].
3. Kill-A-Watt EZ (P4460): This device is functionally identical to the P4400 series except that it includes one extra feature, it can calculate how much a device costs the consumer, after being programmed with the $/KWH provided by the power company [20].

Table 5: Comparison of PICA Competitors

Product	Monitoring features	Control Features	Cost (Fixed)	Cost (Recurring)	PICA Competitor
Kill-A-Watt	voltage, line frequency, amperage, KWH, and current leakage	N/A	$52 to $99	N/A	Ciruit-by-Circuit Monitors
Cent-A-Meter	Cost of electricity, temperature, humidity, kW of demand kg/hour greenhouse gas emissions	N/A	$140	N/A	E-Panel Meter
The Energy Detective	kW load, $/hour	N/A	$119.95 to $455.80	N/A	E-Panel Meter
Watts Up? Smart Circuit	Current, Voltage, kilowatts used	Remote On/Off	$194.95 /circuit	Free to $50.00 /month	Circuit-by-Circuit monitors
Smart-Watt	voltage, line frequency, amperage, KWH, current leakage, circuit load, on/off cycles	N/A	$169 to $249	N/A	Circuit-by-Circuit Monitors

All of the Kill-A-Watt devices claim to be accurate to within 0.2% of the actual power the monitored device uses [18][19][20]. The Kill-A-Watt devices cannot replace a power meter, but simply provide a method of supplying a consumer with additional data about their power consumption.

11.4 Cent-a-Meter

The Australian company, Clipsal produces the Cent-a-meter also known as the Electrisave or the Owl in the UK. Clipsal only produces one version of the Cent-a-meter which displays the cost of the electricity used in the home along with the temperature and humidity [21]. The device can also measure kW of demand, and kg/hour of greenhouse gas emissions [22]. Unlike the Kill-A-Watt, the centimeter does not accumulate any data, just displays instantaneous data on a receiver unit mounted in the home [21]. Clipsal does not list an MSRP for the Cent-a-meter, however SmartHome USA sells Cent-a-meter devices for $140 [22].

11.4.1 The Energy Detective (TED)

Energy Inc., a division of 3M, recently introduced its TED (The Energy Detective) power monitor. Functionally, TED operates exactly as the meter on the exterior of a consumer's home or business but the display resides indoors in a more convenient viewing location. Energy Inc. currently produces two series of the TED device:

1. TED1000 series: The TED1000 devices monitor current energy consumption in killowatts, and current energy cost in $/hour, and log this data for 13 months to predict energy use for the current billing cycle. TED1000 devices can integrate with a proprietary software package, Footprints, provided by Energy Inc. to visually display usage data [23]. TED1000 series devices range in price from $119.95 to $229.95 depending on the amp-rating of the service installation [24].
2. TED5000 series: The TED5000 sought to improve upon the TED1000 series by extending the functionality of the TED devices. The largest selling point for the TED5000 is integration with the Google Power service to track power usage data on the web [25]. TED5000 series units range in price from $239.95 to $455.80 depending on from how many measurement units the device gathers data [26].

11.4.2 Watts Up?

In 1997 Electronic Educational Devices Inc. introduced the Watts Up? product line to the education market. The product immediately became a hit, and soon utility companies across the United States began to take notice [27]. EED markets the Smart Circuit devices as a replacement for traditional circuit breaker devices for 100V to 250V, 20 amp 50/60Hz circuits [28]. Each Smart Circuit contains a built in web-server that allows for aggregation of collected data at a maximum rate of once per second [28]. These Smart Circuits are typically installed into a standard panel enclosure box, similar to standard circuit breakers, mounting directly to the industry-standard DIN rail inside the enclosure[28]. Alternatively, if needed at one local outlet, the Smart Circuit can be housed in a standard double gang electrical box [28]. Each Smart Circuit device can turn itself on or off when it receives a certain remote-control signal or when it detects one of many programmable stimuli. This self-waking feature makes these devices ideal for home-automation projects [28].

A single Smart Circuit, capable of controlling one circuit, costs $194.95, with enclosures for one, five, or ten Smart Circuits devices going for $325.95, $1495.95, and $2495.95 respectively [28]. A basic account, to view aggregated data and control the devices is free for residential use, but data rates, historical data and devices rules are limited [29]. A top-tier account, featuring the fastest update time, 1 second, up to 25 meters, 1 year of archival data, and 25 rules costs $50.00 a month [29].

11.4.3 Smart-Watt

The Smart-Watt device from Smartworks Inc. takes a similar approach to the Kill-A-Watt device in metering a single device at a time, but monitors much more information including circuit load over any period of time, and number of on/off cycles the attached device undergoes [30]. The biggest advantage to the Smart-Watt devices comes from the proprietary network Smartworks has developed for their devices. Each device attaches to a local network where a central server collects and collates all the data [30]. The Smart-Watt comes in two versions, one for International Electrotechnical Commision (IEC) plugs and receptacles and one for National Electrical Manufacturers Association (NEMA) plugs and receptacles. Both devices are similarly priced ranging from $169 to $249 depending on the current rating [30].

11.4.4 Standard Power Meter

Most homes or businesses attached to the electric grid are metered using a standard analogue power meter. This device provided by the power company, measures the amount of electrical energy consumed over a period of time. Typically, a power meter records in billing units, such as KWH. Each meter requires periodic readings based on the billing cycle of the power company; it is safe to assume that meters are read approximately once per month. In order to read the meter, an employee of the power company will physically go out to the meter and record usage data.

11.4.5 Nonintrusive Appliance Load Monitor

All of the products discussed here use a technique known as Non-intrusive Load Monitoring (NILM) to monitor power consumption without affecting the load on the circuit [31]. However, some more sophisticated products in this area use NILM to estimate the number of individual loads on the circuit [32]. If the research in this field proves that NILM provides accurate and useful data, devices based on the NILM technology would have a large advantage over other single-device power monitors, as such a device could be inserted into the feeder lines from the utility company and monitor all devices in the entire installation from a single point. Research turned up no significant products that claim to be capable of monitoring multiple loads on a circuit from a single point on the circuit. Thus this section is included to provide information, but does not represent a viable competitor in the market just yet.

11.4.6 PICA Competitors Comparison

In order to better understand the competition in the marketplace table 5 reproduces the information laid out above as a comparative table. The column on the far right side describes the PICA component that most directly competes with the product listed in the left column.

11.5 Project Finances

11.5.1 Bill of Materials

Table 6: Materials and Cost for a Single Breaker

Item	Quantity	Unit Price	Single System	Manufacturer Part #	Vendor
Solid State Relays	2	18.42	36.84	STH24D25	Mouser
5pF SMD Cap	2	0.03	0.06	C0603C0G1E050C	Mouser
Opto Isolator	2	3.03	6.06	OPI1264C	Mouser
3.579545MHz Crystal	1	0.63	0.63	ABLSG-3.579545MHZ-D-2-Y-T	Mouser
Total:	7		43.59		

Table 7: Materials and Cost for 1000 Breakers

Item	Quantity	Price per 1000	For 1000 Systems	Manufacturer Part #	Vendor
Solid State Relays	2	13.00	26000.00	STH24D25	Mouser
5pF SMD Cap	2	0.005	10.00	C0603C0G1E050C	Mouser
Opto Isolator	2	1.76	3520.00	OPI1264C	Mouser
3.579545MHz Crystal	1	0.35	350.00	ABLSG-3.579545MHZ-D-2-Y-T	Mouser
Total:	7.00		29880.00		

Table 8: Materials and Cost for a Single E-Meter

Item	Quantity	Part Price	Single System	Manufacturer Part #	Vendor
LCD Display	1	22.75	22.75	SCLBC	SynchroSystems

Continued on next page

Table 8: Continued from previous page

Item	Quantity	Part Price	Single System	Manufacturer Part #	Vendor
SSOP to DIP Adapter 20-Pin	2	3.95	7.90	BOB-00499	SparkFun
Break Away Headers – Straight	1	2.5	2.50	PRT-00116	SparkFun
1N4148 Diode	24	0.09	2.16	1N4148	Mouser
2500 Series Box Header	2	1.41	2.82	N2514-6002RB	Mouser
AVE Series Electrolytic Cap 1uF	12	0.07	0.84	AVE105M50B12T-F	Mouser
Kehmet .033uF SMD Cap	12	0.59	7.08	C1206C333KARACTU	Mouser
Kehmet .015uF SMD Cap	6	0.34	2.04	C1206C153KARACTU	Mouser
Kehmet 47pF SMD Cap	12	0.07	0.84	C0603C470J5GACTU	Mouser
Murata Inductor EFI	12	0.11	1.32	BLM21BD102SN1D	Mouser
Maxim RS233a RS232 Driver	1	8.7	8.70	MAX233CPP+G36	Mouser
D-Sub 9 Connector	1	2.32	2.32	56F404-001	Mouser
Xbee Connector 2mm pitch	2	1.6	3.20	M22-7131042	Mouser

Continued on next page

Table 8: Continued from previous page

Item	Quantity	Part Price	Single System	Manufacturer Part #	Vendor
16MHz Crystal	1	0.41	0.41	ABLS-16.000MHZ-B4-T	Mouser
32.768kHz Crystal	1	0.99	0.99	ABS10-32.768KHZ-7-T	Mouser
Tyco 6P Terminal Block	2	2.33	4.66	796949-6	Mouser
Tyco 3P Terminal Block	2	0.83	1.66	796949-3	Mouser
Varistor	3	0.26	0.78	V8ZA05P	Mouser
Current Transformer	3	6.11	18.33	CST-1020	Mouser
Linear Regulator	3	1.54	4.62	ZSR300GTA	Mouser
100nF SMD Cap	3	0.7	2.10	CB037D0104JBA	Mouser
22uF SMD Cap	4	0.54	2.16	EEE-1CA220SR	Mouser
18pF SMD Cap	2	0.09	0.18	CGA2B2C0G1H180J	Mouser
Red SMD LED	1	0.4	0.40	HSMC-C190	Mouser
Green SMD LED	1	0.88	0.88	HSMQ-C120	Mouser
Amber SMD LED	1	0.4	0.40	HSMA-C190	Mouser
7pF SMD Cap	2	0.03	0.06	C0603C0G1E070C	Mouser
Tact Switch	2	0.47	0.94	B3W-1052	Mouser
Total:	119		103.04		

Table 9: Materials and Cost for 1000 E-Meters

Item	Quantity	Unit Price	Single System	Manufacturer Part #	Vendor
Solid State Relays	2	18.42	36.84	STH24D25	Mouser
5pF SMD Cap	2	0.03	0.06	C0603C0G1E050C	Mouser
Opto Isolator	2	3.03	6.06	OPI1264C	Mouser
3.579545MHz Crystal	1	0.63	0.63	ABLSG-3.579545MHZ-D-2-Y-T	Mouser
Total:	7		43.59		

Table 10: Materials and Cost for a Single Power Supply

Item	Quantity	Part Price	Single System	Manufacturer Part #	Vendor
1000uF Cap	1	3.18	3.18	TVX1J102MCD	Mouser
10u Cap	3	0.3	0.90	C1206C106K9PACTU	Mouser
100n Cap	2	0.03	0.06	C1608Y5V1H104Z	Mouser
47n Cap	1	0.2	0.20	C1608X7S2A473K	Mouser
2.2n Cap	1	0.16	0.16	06035C222K4Z2A	Mouser
1n Cap	1	0.03	0.03	C1608X7R1H102M	Mouser
2A 30V Schottky Diode	1	0.55	0.55	RB060M-30TR	Mouser
4.7u Inductor	1	0.72	0.72	SRR0604-4R7ML	Mouser
56K Resistor	1	1.6	1.60	RG1608N-563-B-T1	Mouser
15K Resistor	1	0.1	0.10	TNPW060315K0DEEA	Mouser
10K Resistor	1	1.1	1.10	PAT0603E1002BST1	Mouser
1.5K Resistor	2	0.79	1.58	TNPW06031K50BEEN	Mouser

Continued on next page

Table 10: Continued from previous page

Item	Quantity	Part Price	Single System	Manufacturer Part #	Vendor
10 Resistor	2	0.62	1.24	TNPW060310R0BEEA	Mouser
0.1 Resistor	1	0.3	0.30	CRL1206-FW-R100ELF	Mouser
LM25011	1	4.86	4.86	LM25011MY/NOPB	Digikey
Transformer	1	23.56	23.56	166M18	Mouser
Diode	4	0.23	0.92	1N4004G	Mouser
Total:	25		41.06		

Table 11: Materials and Cost for 1000 Power Supplies

Part	Quantity	Price per 1000	For 1000 Systems	Manufacturer Part #	Vendor
1000uF Cap	1	2.02	2020	TVX1J102MCD	Mouser
10u Cap	3	0.055	165	C1206C106K9PACTU	Mouser
100n Cap	2	0.007	14	C1608Y5V1H104Z	Mouser
47n Cap	1	0.036	36	C1608X7S2A473K	Mouser
2.2n Cap	1	0.08	80	06035C222K4Z2A	Mouser
1n Cap	1	0.006	6	C1608X7R1H102M	Mouser
2A 30V Schottky Diode	1	0.169	169	RB060M-30TR	Mouser
4.7u Inductor	1	0.34	340	SRR0604-4R7ML	Mouser
56K Resistor	1	0.37	370	RG1608N-563-B-T1	Mouser
15K Resistor	1	0.072	72	TNPW060315K0DEEA	Mouser
10K Resistor	1	0.472	472	PAT0603E1002BST1	Mouser
1.5K Resistor	2	0.47	940	TNPW06031K50BEEN	Mouser
10 Resistor	2	0.37	740	TNPW060310R0BEEA	Mouser

Continued on next page

Table 11: Continued from previous page

Part	Quantity	Price per 1000	For 1000 Systems	Manufacturer Part #	Vendor
0.1 Resistor	1	0.121	121	CRL1206-FW-R100ELF	Mouser
LM25011	1	2.835	2835	LM25011MY/NOPB	Digikey
Transformer	1	17.28	17280	166M18	Mouser
Diode	4	0.042	168	1N4004G	Mouser
Total:	25		25828.00		

11.5.2 Board Cost

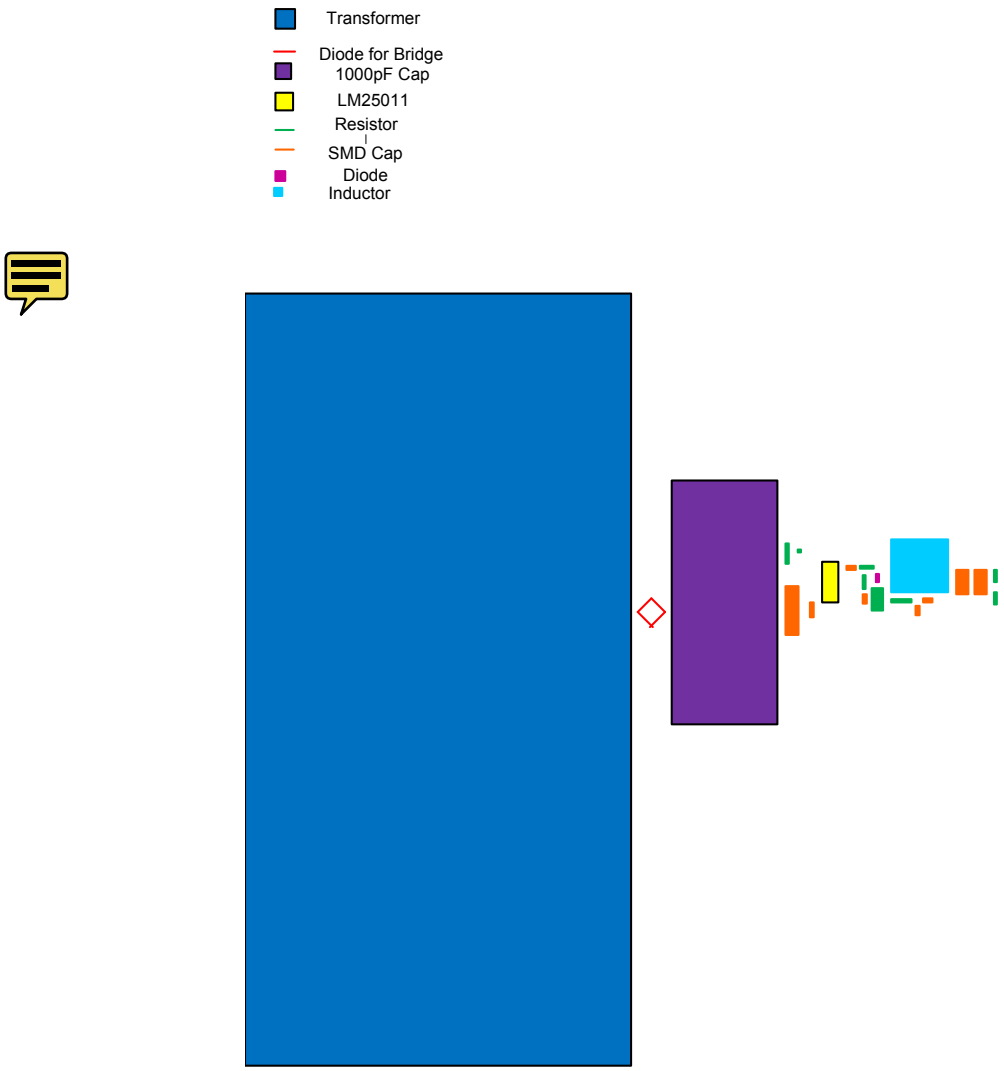

Figure 37: Power supply board mockup, 92.5 x 95 mm.

12 Acknowledgements

Team PICA would like to acknowledge and thank the following people for their help in the course of the project.

- Professor VanderLeest for his work as the Team Advisor, providing constructive feedback and insights, and for pushing us to always improve.
- Professor Ribeiro for his help in Engr. 315 and insights on power systems.
- Consumer's Energy, for allowing us to visit their Smart Grid Learning Center, and Brian Bushey, Mark Luehmann and Tim Voss specifically for answering many questions about their work as a power company.
- Tim Theriault for his time as the team's Industrial Consultant.
- Mark Michmerhuizen for his work providing feedback on team documents.
- Chuck Cox, and John Lupien of SynchroSystems for their assistance in debugging the E-Meter LCD.
- The Business team from Bus. 396 for helping us understand business aspects of our project.
- Professor Medema for helping understand business aspects of our project.
- Bob DeKraker, Chuck Holwerda and Phil Jasperse for their help as shop and lab technicians.
- Glenn Remelts for assisting with team research.
- Texas Instruments for generously providing us with a MSP430 development kit.
- All of our friends and family for supporting, encouraging and putting up with us.

References

[1] S. English and D. Smith, “A power meter reference design based on ade7756,” Electronic, 2001.

[2] T. Instruments, “Msp430f471x3, msp430f471x6, msp430f471x7 mixed signal microcontroller,” March 2011.

[3] National power corporation. [Online]. Available: www.natpow.com

[4] M. Hill, “What are voltage sags?”

[5] The home depot. [Online]. Available: www.homedepot.com

[6] ANSI, “American national standard for utility industry end device data tables,” National Electrical Manufacturers Association, Tech. Rep., February 2009.

[7] ——, “Ansi c12.21-2006 american national standard protocol specification for telephone modem communication,” National Electrical Manufacturers Association, Tech. Rep., May 2006.

[8] FCC. Title 47: Telecommunication. [Online]. Available: http://ecfr.gpoaccess.gov/cgi/t/text/text-idx?c=ecfr&sid=72eb0650720fb4c17ca9a1f782cdb360&rgn=div5&view=text&node=47:1.0.1.1.14&idno=47

[9] A. Ball, A. Sterk, K. Wiersma, and N. Jen, “Team 01: Team pica project proposal and feasability study,” December 2010.

[10] (2011, April). [Online]. Available: www.synchros.com

[11] (2011, April). [Online]. Available: http://www.softbaugh.com/ProductPage.cfm?strPartNo=SBLCDA4

[12] T. Instruments, “slac316a.”

[13] (2011, April). [Online]. Available: http://schmidt-walter.eit.h-da.de/smps_e/smps_e.html

[14] (2011, April). [Online]. Available: http://www.national.com/pf/LM/LM25011.html#Overview

[15] (2011, April). [Online]. Available: http://www.national.com/analog/webench/power

[16] U. Laboratories. (2010) About ul. [Online]. Available: http://www.ul.com/global/eng/pages/corporate/aboutul/?set-cookie=true

[17] P. International, “About p3,” 2008. [Online]. Available: http://www.p3international.com/about.html

[18] ——, “Kill a watt ps electricity usage monitoring power strip,” Electronic Data Sheet, 2007.

[19] ——, “Kill a watt electricity usage monitor,” Electronic Data Sheet, 2007.

[20] ——, “Kill a watt ez,” Electronic Data Sheet, 2007.

[21] Clipsal. (2010) Cent-a-meter sales. [Online]. Available: http://www.clipsal.com/homeowner/products/smart_home_technology/cent-a-meter

[22] (2010, November) Smarthomeusa webshop. [Online]. Available: http://www.smarthomeusa.com/ShopByManufacturer/eco-response/Item/CM113A/specifications/

[23] (2010) Ted 1000 features. [Online]. Available: http://www.theenergydetective.com/ted-1000/features

[24] (2010) Ted1000 store. [Online]. Available: http://www.theenergydetective.com/store/ted-1000

[25] “Ted 5000 features,” 2010. [Online]. Available: http://www.theenergydetective.com/ted-5000/features

[26] “Ted 5000 store,” 2010. [Online]. Available: http://www.theenergydetective.com/store/ted-5000?limit=25

[27] (2010) About watts up? [Online]. Available: https://www.wattsupmeters.com/secure/about.php

[28] (2010) Smart circuit 20. [Online]. Available: https://www.wattsupmeters.com/secure/products.php?pn=20&wai=256&more=2

[29] (2010) Watts up? services overview. [Online]. Available: https://www.wattsupmeters.com/secure/products.php?pn=2

[30] SmartWorks. (2010) Inline high-resolution watt-hour meter. [Online]. Available: http://www.smart-works.com/pdf/SmartWattBrochureInline.pdf

[31] G. W. Hart, E. C. Kern, Jr., and F. C. Schweppe, “Non-intrusive appliance monitor apparatus,” U.S. Patent 4 858 141, August 15, 1989.

[32] G. W. Hart. (1992) Nonintrusive appliance load monitoring. [Online]. Available: http://www.georgehart.com/research/nalm.html

A E-Meter Appendix

A.1 Device Pinout

[This section will eventually contain a table showing how each of the 100 pins on the MSP430 are used.]

A.2 Hello World Program

Listing 1: Embedded C MSP430 Hello World program.

```
/*************************************************************
 * MSP430F47197 Hello World Demo
 * Description: Toggle LED located at P5.1 on MSP430 Dev Board
 * Hardware Setup: See diagram below, requires only JTAG clk.
 *
 *                   MSP430x471x7
 *              -----------------
 *          /|\|                 |
 *           | |                 |
 *           --|RST              |
 *             |                 |
 *             |             P5.1|-->LED
 * Author: Kendrick Wiersma -- kendrick.g.wiersma@gmail.com
 */

#include "msp430x471x7.h"

void main(void) {
  volatile unsigned int i;

  WDTCTL = WDTPW + WDTHOLD; // Ask the watchdog timer to please hold for us
  P5DIR |= BIT1;            // Set P5.1 to be an output

  while(1) {
    P5OUT ^= BIT1;
    for(i=50000; i > 0; i--); //delay for a bit
  }
}
```

A.3 Compiling the MSPGCC4 Toolchain for Ubuntu Linux

The design team used Ubuntu Linux 10.10 32-bit to compile the MSPGCC4 toolchain. The MSPGCC4 toolchain depends on several packages readily available in the Ubuntu repository:

- subversion
- texinfo
- gcc-4.4
- patch

Listing 2: Install MSPGCC4 dependencies

```
sudo apt-get install subversion gcc-4.4 texinfo patch libncurses5-dev zlibc zlib1g-dev libx11-dev
     libusb-dev libreadline6-dev
```

- libncurses5-dev
- zlibc
- zlib1g-dev
- libx11-dev
- libusb-dev
- libreadline6-dev

Install all the following packages by executing the following command in Terminal:

Next, checkout the MSPGCC4 source from the online subversion repository using the command line version of subversion.

Once all the code has been downloaded, switch to the new directory and execute the buildgcc.sh script. Accept all the default options and wait for a while until the compilation completes.

Now navigate to `http://mspdebug.sourceforge.net/download.html` and download the latest release. Extract the files from the downloaded archive. Change directories to the newly extracted code and issue a make command. Once the compilation completes, install mspdebug using make install.

This completes all the essential tools for compiling code using MSPGCC4. Some additional tools may be required to work with the programming pod.

In order to load code onto the MSP430, gdb and HAL need to be configured to properly talk to the programming pod. Change to the install directory of the MSPGCC4 (usually /opt/msp430-gcc-4.4.3/bin/msp430-gdb-proxy/) and issue the command:

```
sudo wget http://www.soft-switch.org/downloads/mspgcc/msp430-gdbproxy
```

once the file transfer completes issue the command:

```
chmod 777 msp430-gdbproxy
```

to allow the file to execute.

Now switch to /usr/lib and download the libHil.so file using wget:

```
sudo wget http://www.soft-switch.org/downloads/mspgcc/libHIL.so
```

Repeat to download the MSP430.so driver file:

```
sudo wget http://www.soft-switch.org/downloads/mspgcc/MSP430.so
```

A.4 SD16 to RS232 UART Test

The following code transmits results from the second SD16 converter to the RS232 UART

Listing 3: Embedded C MSP430 code to transmit date from the SD16 readings to the RS232 UART.

```
//******************************************************************************
//  MSP430x471xx Demo - SD16, Single Conversion on a Single Channel Using ISR
//
//                 MSP430x471xx
//             -----------------
//          /|\|                 XIN|-
//           | |                    | 32kHz
//           --|RST             XOUT|-
//             |                    |
//     Vin+ -->|A2.0+               |
//     Vin- -->|A2.0-               |
//             |                    |
//             |              VREF  |---+
//             |                    |   |
//             |                    |  -+- 100nF
//             |                    |  -+-
//             |                    |   |
//             |              AVss  |---+
//             |                    |
// Based on TI demo code
// Modified by Avery Sterk April 2011
//******************************************************************************
#include  <msp430x471x7.h>

unsigned int result;

void main(void)
{
  volatile unsigned int i;                  // Use volatile to prevent removal
                                            // by compiler optimization

  WDTCTL = WDTPW + WDTHOLD;                 // Stop WDT
  FLL_CTL0 |= XCAP14PF;                     // Configure load caps

  do
  {
    IFG1 &= ~OFIFG;                         // Clear OSCFault flag
    for (i = 0x47FF; i > 0; i--);           // Time for flag to set
  }
  while ((IFG1 & OFIFG));                   // OSCFault flag still set?

  P2SEL |= BIT4+BIT5;                       // P2.4,5 = USCI_A0 RXD/TXD
  UCA0CTL1 |= UCSSEL_1;                     // CLK = ACLK
  UCA0BR0 = 0x03;                           // 32k/9600 - 3.41
  UCA0BR1 = 0x00;                           //
  UCA0MCTL = 0x06;                          // Modulation
  UCA0CTL1 &= ~UCSWRST;                     // **Initialize USCI state machine**
  IE2 |= UCA0RXIE;                          // Enable USCI_A0 RX interrupt

  SD16CTL = SD16REFON+SD16SSEL0;            // 1.2V ref, SMCLK
  SD16CCTL2 |= SD16SNGL+SD16IE ;            // Single conv, enable interrupt
  for (i = 0; i < 0x3600; i++);             // Delay for 1.2V ref startup
```

```
  while (1)
  {
    _NOP();                                 // SET BREAKPOINT HERE
    SD16CCTL2 |= SD16SC;                    // Set bit to start conversion
    __bis_SR_register(LPM0_bits + GIE);     // Enter LPM0 w/ interrupts
  }
}

#pragma vector=SD16A_VECTOR
__interrupt void SD16AISR(void)
{
  switch (SD16IV)
  {
  case 2:                                   // SD16MEM Overflow
    break;
  case 4:                                   // SD16MEM0 IFG
    break;
  case 6:                                   // SD16MEM1 IFG
    break;
  case 8:                                   // SD16MEM2 IFG
    result = SD16MEM2;                      // Save CH2 results (clears IFG)
//    UCA0TXBUF = (result >> 9); // bitshift result down by 9 bits, so it's at least ASCII
    UCA0TXBUF = (SD16MEM2 >> 12) + 0x41; // map top 4 bits of 16-bit measurement to 'A'-'P
        '.
    break;
  }

  __bic_SR_register_on_exit(LPM0_bits);        // Exit LPM0
}
```

A.5 E-Meter Schematic

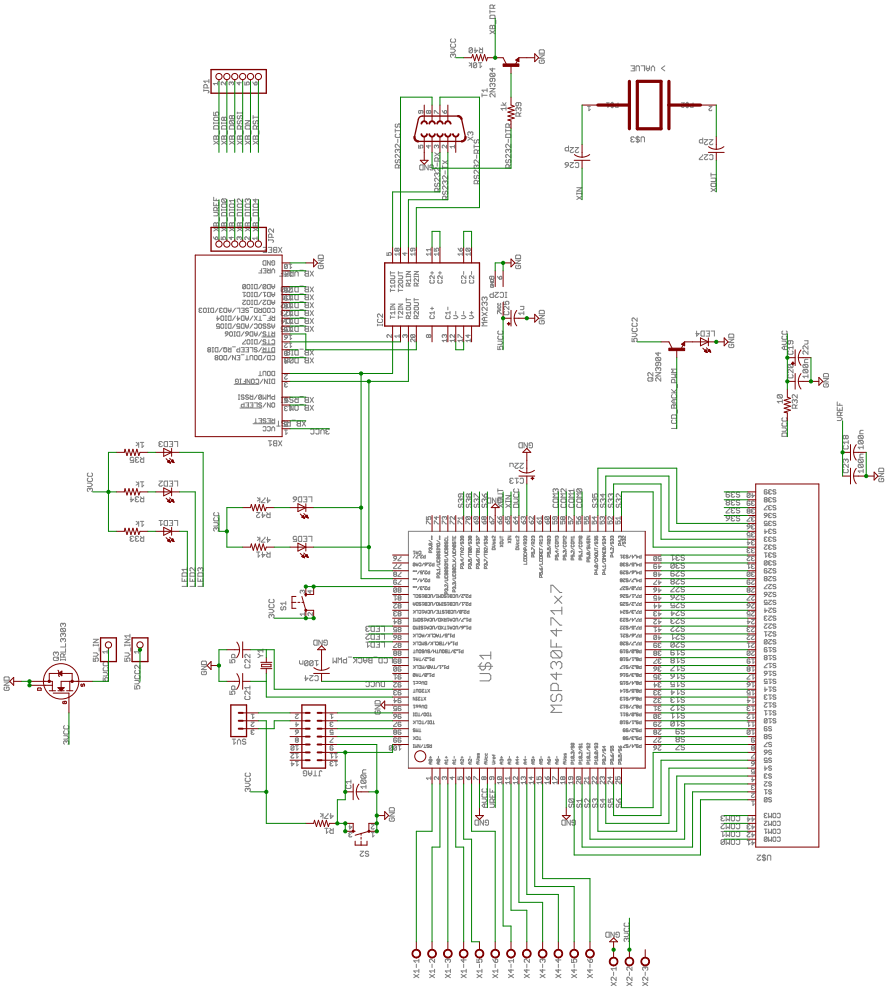

Figure 38: E-Meter Main Board Schematic

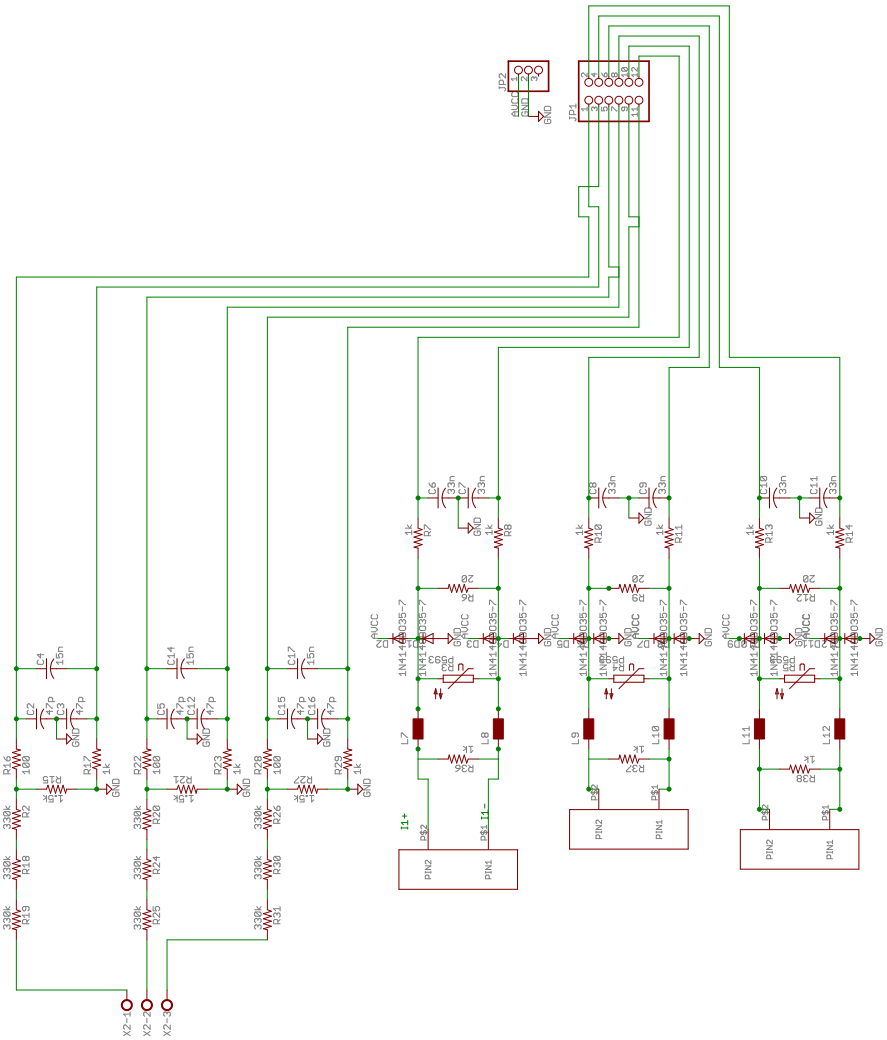

Figure 39: E-Meter input board schematic.

A.6 E-Meter Printed Circuit Board

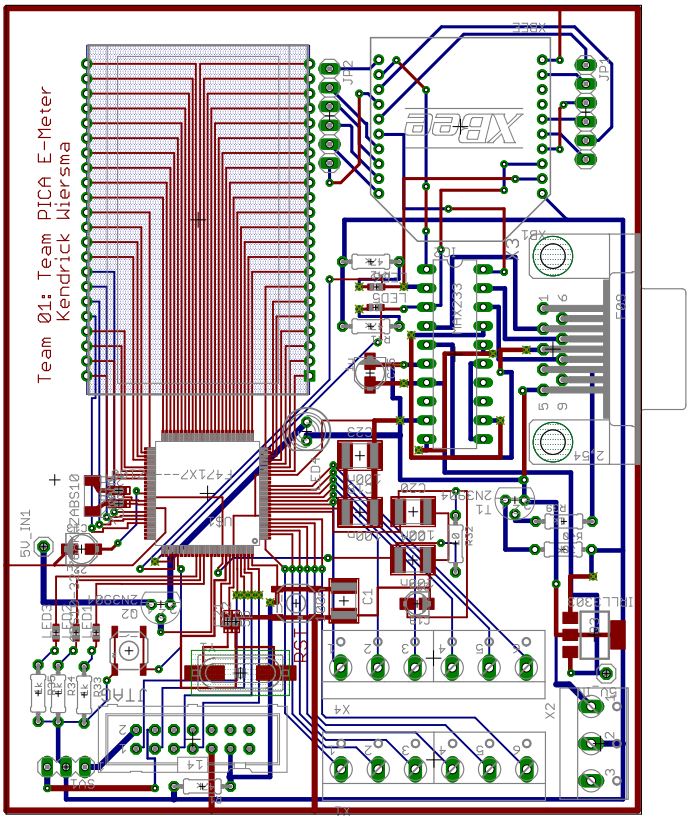

Figure 40: E-Meter main board.

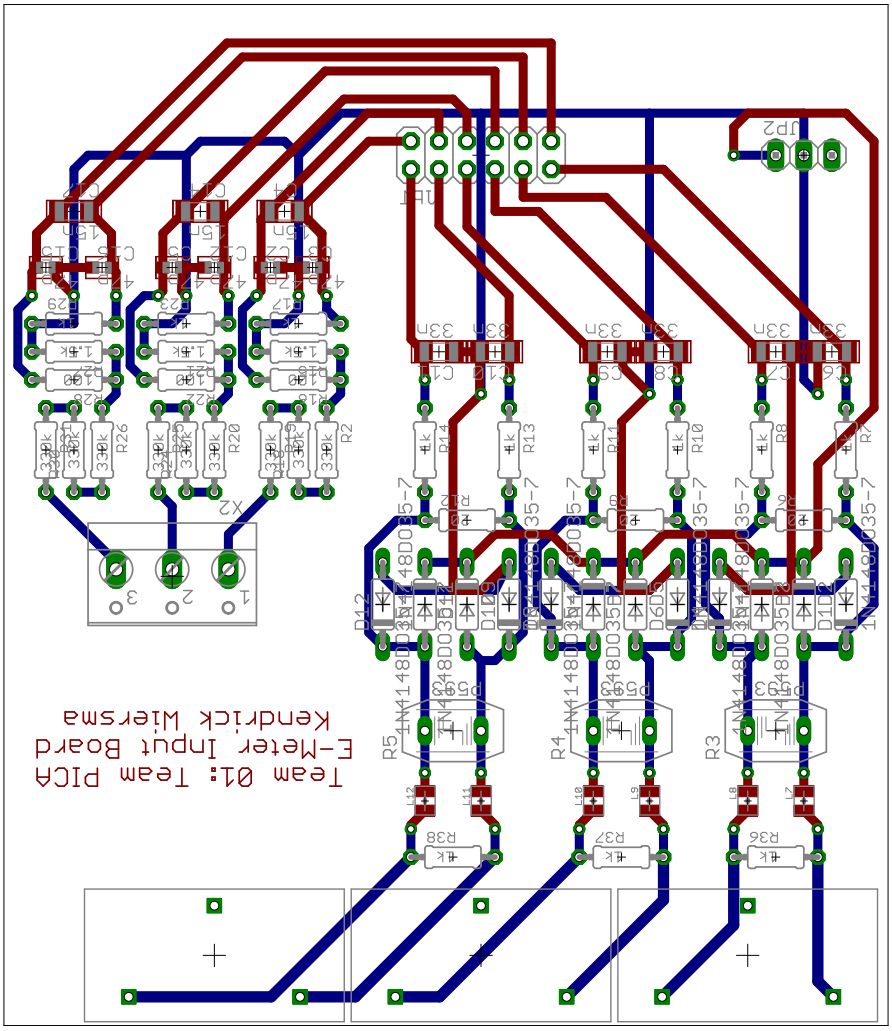

Figure 41: E-Meter input board.

A.7 E-Meter Code Listing

B Base Station Appendix

B.1 LEON3 Hardware Linux

Listing 4: GRMON Hardware Listing

```
GRMON LEON debug monitor v1.1.46 evaluation version

Copyright (C) 2004,2005 Gaisler Research - all rights reserved.
For latest updates, go to http://www.gaisler.com/
Comments or bug-reports to support@gaisler.com

This evaluation version will expire on 28/9/2011
Xilinx cable: Cable type/rev : 0x3
JTAG chain: xc5vlx110t xccace xc95144xl xcf32p xcf32p

Device ID: : 0x509
GRLIB build version: 4104

initialising
detected frequency:  80 MHz

Component                            Vendor
LEON3 SPARC V8 Processor             Gaisler Research
AHB Debug UART                       Gaisler Research
AHB Debug JTAG TAP                   Gaisler Research
SVGA Controller                      Gaisler Research
GR Ethernet MAC                      Gaisler Research
DDR2 Controller                      Gaisler Research
AHB/APB Bridge                       Gaisler Research
LEON3 Debug Support Unit             Gaisler Research
LEON2 Memory Controller              European Space Agency
System ACE I/F Controller            Gaisler Research
Generic APB UART                     Gaisler Research
Multi-processor Interrupt Ctrl       Gaisler Research
Modular Timer Unit                   Gaisler Research
PS/2 interface                       Gaisler Research
PS/2 interface                       Gaisler Research
General purpose I/O port             Gaisler Research
AMBA Wrapper for OC I2C-master       Gaisler Research
AMBA Wrapper for OC I2C-master       Gaisler Research
AHB status register                  Gaisler Research

Use command 'info sys' to print a detailed report of attached cores


Grmon> info sys
00.01:003   Gaisler Research  LEON3 SPARC V8 Processor (ver 0x0)
             ahb master 0
01.01:007   Gaisler Research  AHB Debug UART (ver 0x0)
             ahb master 1
             apb: 80000700 - 80000800
             baud rate 115200, ahb frequency 80.00
02.01:01c   Gaisler Research  AHB Debug JTAG TAP (ver 0x1)
             ahb master 2
```

```
03.01:063   Gaisler Research  SVGA Controller (ver 0x0)
             ahb master 3
             apb: 80000600 - 80000700
             clk0: 25.00 MHz  clk1: 25.00 MHz  clk2: 40.00 MHz  clk3: 65.00 MHz
04.01:01d   Gaisler Research  GR Ethernet MAC (ver 0x0)
             ahb master 4, irq 12
             apb: 80000b00 - 80000c00
             Device index: dev0
             edcl ip 192.168.0.52, buffer 2 kbyte
00.01:02e   Gaisler Research  DDR2 Controller (ver 0x0)
             ahb: 40000000 - 60000000
             ahb: fff00100 - fff00200
             64-bit DDR2 : 1 * 256 Mbyte @ 0x40000000, 4 internal banks
                           190 MHz, col 10, ref 7.8 us, trfc 131 ns
01.01:006   Gaisler Research  AHB/APB Bridge (ver 0x0)
             ahb: 80000000 - 80100000
02.01:004   Gaisler Research  LEON3 Debug Support Unit (ver 0x1)
             ahb: 90000000 - a0000000
             AHB trace 128 lines, 32-bit bus, stack pointer 0x4ffffff0
             CPU#0 win 8, hwbp 2, itrace 128, V8 mul/div, srmmu, lddel 1, GRFPU-lite
                   icache 2 * 8 kbyte, 32 byte/line lru
                   dcache 1 * 8 kbyte, 16 byte/line
03.04:00f   European Space Agency  LEON2 Memory Controller (ver 0x1)
             ahb: 00000000 - 20000000
             ahb: 20000000 - 40000000
             ahb: c0000000 - c2000000
             apb: 80000000 - 80000100
             16-bit prom @ 0x00000000
04.01:067   Gaisler Research  System ACE I/F Controller (ver 0x0)
             irq 13
             ahb: fff00200 - fff00300
01.01:00c   Gaisler Research  Generic APB UART (ver 0x1)
             irq 2
             apb: 80000100 - 80000200
             baud rate 38461, DSU mode (FIFO debug)
02.01:00d   Gaisler Research  Multi-processor Interrupt Ctrl (ver 0x3)
             apb: 80000200 - 80000300
03.01:011   Gaisler Research  Modular Timer Unit (ver 0x0)
             irq 8
             apb: 80000300 - 80000400
             8-bit scaler, 2 * 32-bit timers, divisor 80
04.01:060   Gaisler Research  PS/2 interface (ver 0x2)
             irq 4
             apb: 80000400 - 80000500
05.01:060   Gaisler Research  PS/2 interface (ver 0x2)
             irq 5
             apb: 80000500 - 80000600
08.01:01a   Gaisler Research  General purpose I/O port (ver 0x1)
             apb: 80000800 - 80000900
09.01:028   Gaisler Research  AMBA Wrapper for OC I2C-master (ver 0x2)
             irq 14
             apb: 80000900 - 80000a00
           Controller index for use in GRMON: 1
0c.01:028   Gaisler Research  AMBA Wrapper for OC I2C-master (ver 0x2)
             irq 11
             apb: 80000c00 - 80000d00
           Controller index for use in GRMON: 2
0f.01:052   Gaisler Research  AHB status register (ver 0x0)
             irq 7
```

```
            apb: 80000f00 - 80001000

Grmon>
```

B.2 Linux Log

Listing 5: Linux Log

```
=~=~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2011.02.27 23:59:45 =~=~=~=~=~=~=~=~=~=~=~=~=~=
Booting Linux
Booting Linux...
PROMLIB: Sun Boot Prom Version 0 Revision 0

Linux version 2.6.21.1 (avery@atlantis) (gcc version 3.4.4) #2 Sun Feb 27 23:56:21 EST
    2011

ARCH: LEON

TYPE: Leon2/3 System-on-a-Chip

Ethernet address: 0:0:0:0:0:0

CACHE: direct mapped cache, set size 8k

Boot time fixup v1.6. 4/Mar/98 Jakub Jelinek (jj@ultra.linux.cz). Patching kernel for
    srmmu[Leon2]/iommu

64MB HIGHMEM available.

Nocache: 0xfc000000-0xfc400000, 1024 pages [128-1280]

node 2: /cpu00 (type:cpu) (props:.node device_type mid mmu-nctx clock-frequency
    uart1_baud uart2_baud )

node 3: /a: (type:serial) (props:.node device_type name )

node 4: /ambapp0 (type:ambapp) (props:.node device_type name )

PROM: Built device tree from rootnode 1 with 1478 bytes of memory.

DEBUG: psr.impl = 0xf  fsr.vers = 0x7

Built 1 zonelists.  Total pages: 64133

Kernel command line: console=ttyS0,38400 rdinit=/sbin/init

PID hash table entries: 1024 (order: 10, 4096 bytes)

Todo: init master_l10_counter

Attaching grlib apbuart serial drivers (clk:80hz):

Console: colour dummy device 80x25

Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)

Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)

pkbase: 0xfc800000 pkend: 0xfcc00000 fixstart 0xfcff1000
Memory: 252096k/262144k available (1516k kernel code, 9920k reserved, 168k data, 1732k
    init, 65536k highmem)

Mount-cache hash table entries: 512
```

```
NET: Registered protocol family 16
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
leon: power management initialized
highmem bounce pool size: 64 pages
io scheduler noop registered
io scheduler cfq registered (default)
grlib apbuart: 1 serial driver(s) at [0x80000100(irq 2)]
grlib apbuart: system frequency: 80000 khz, baud rates: 38400 38400
ttyS0 at MMIO 0x80000100 (irq = 2) is a Leon
Testing fifo size for UART port 0: got 4 bytes.
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
xsysace xsysace.0: Xilinx SystemACE revision 1.0.12
xsysace xsysace.0: capacity: 3906560 sectors
 xsa: xsa1 xsa2
Xilinx SystemACE device driver, major=254
Probing GRETH Ethernet Core at 0x80000b00
Detected MARVELL 88EE1111 Revision 2
10/100 GRETH Ethermac at [0x80000b00] irq 12. Running 100 Mbps full duplex
TCP cubic registered
NET: Registered protocol family 10
IPv6 over IPv4 tunneling driver
Freeing unused kernel memory: 1732k freed

init started: BusyBox v1.8.2 (2011-02-27 23:11:31 EST)
```

```
starting pid 14, tty '': '/etc/init.d/rcS'
mount: mounting tmpfs on /var/tmp failed: Invalid argument

starting pid 25, tty '': '/bin/sh'
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:01:D2:66
          inet addr:192.168.0.80  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe01:d266/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Base address:0xb00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # echo "Now I"m pl      'm plugging in the E ethernet"
Now I'm plugging in the ethernet
/ #
/ # echo "Now I'm plugging in the ethernet"
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:01:D2:66
          inet addr:192.168.0.80  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe01:d266/64 Scope:Link
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:6 errors:0 dropped:0 overruns:0 frame:0
          TX packets:6 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Base address:0xb00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # ifconfig eht0   th0 down
/ #
/ # ifconfig eth0 down    up
/ #
/ # ifconfig eth0 up
/ # ifconfig eth0 down
/ # ifconfig
eth0      Link encap:Ethernet  HWaddr 00:0A:35:01:D2:66
          inet addr:192.168.0.80  Bcast:192.168.0.255  Mask:255.255.255.0
          inet6 addr: fe80::20a:35ff:fe01:d266/64 Scope:Link
```

```
          UP BROADCAST RUNNING  MTU:1500  Metric:1
          RX packets:9 errors:0 dropped:0 overruns:0 frame:0
          TX packets:10 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)
          Base address:0xb00

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

/ # dh
/ # dh
dhcprelay  dhrystone

/ # dh  exit

process '/bin/sh' (pid 25) exited. Scheduling it for restart.

starting pid 33, tty '': '/bin/sh'
/ #
```

B.3 DHClient with Busybox Log

Listing 6: DHClient with Busybox

```
=~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2011.03.08 13:45:09 =~=~=~=~=~=~=~=~=~=~=~=~=
Booting Linux
Booting Linux...
PROMLIB: Sun Boot Prom Version 0 Revision 0

Linux version 2.6.21.1 (avery@atlantis) (gcc version 3.4.4) #13 Mon Mar 7 17:44:18 EST
    2011

ARCH: LEON

Attaching textvga drivers [0xf01fcf34 s:0x1fe0]: ##! error GAISLER_VGA(0x61) not found

TYPE: Leon2/3 System-on-a-Chip

Ethernet address: 0:0:0:0:0:0

CACHE: direct mapped cache, set size 8k

Boot time fixup v1.6. 4/Mar/98 Jakub Jelinek (jj@ultra.linux.cz). Patching kernel for
    srmmu[Leon2]/iommu

64MB HIGHMEM available.

Nocache: 0xfc000000-0xfc400000, 1024 pages [128-1280]

node 2: /cpu00 (type:cpu) (props:.node device_type mid mmu-nctx clock-frequency
    uart1_baud uart2_baud )

node 3: /a: (type:serial) (props:.node device_type name )

node 4: /ambapp0 (type:ambapp) (props:.node device_type name )

PROM: Built device tree from rootnode 1 with 1478 bytes of memory.

DEBUG: psr.impl = 0xf  fsr.vers = 0x7

Built 1 zonelists.  Total pages: 64485

Kernel command line: console=ttyS0,38400 root=254:1 rw init=/sbin/init

PID hash table entries: 1024 (order: 10, 4096 bytes)

Todo: init master_l10_counter

Attaching grlib apbuart serial drivers (clk:80hz):

Console: colour dummy device 80x25

Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)

Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)

pkbase: 0xfc800000 pkend: 0xfcc00000 fixstart 0xfcff1000
Memory: 253504k/262144k available (1656k kernel code, 8512k reserved, 204k data, 136k
    init, 65536k highmem)
```

```
Mount-cache hash table entries: 512
NET: Registered protocol family 16
Generic PHY: Registered new driver
SCSI subsystem initialized
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
leon: power management initialized
highmem bounce pool size: 64 pages
io scheduler noop registered
io scheduler cfq registered (default)
GRVGA: Defaulting to 640x480 8-bit resolution
Framebuffer address from node : 0xf0980000
Framebuffer address from videomemory : 0x40980000
fb0: Gaisler frame buffer device, using 300K of video memory
grlib apbuart: 1 serial driver(s) at [0x80000100(irq 2)]
grlib apbuart: system frequency: 80000 khz, baud rates: 38400 38400
ttyS0 at MMIO 0x80000100 (irq = 2) is a Leon
Testing fifo size for UART port 0: got 4 bytes.
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
xsysace xsysace.0: Xilinx SystemACE revision 1.0.12
xsysace xsysace.0: capacity: 3906560 sectors
 xsa: xsa1 xsa2
Xilinx SystemACE device driver, major=254
Marvell 88E1101: Registered new driver
```

```
Marvell 88E1111: Registered new driver
Marvell 88E1145: Registered new driver
Probing GRETH Ethernet Core at 0x80000b00
Detected MARVELL 88EE1111 Revision 2
10/100 GRETH Ethermac at [0x80000b00] irq 12. Running 100 Mbps full duplex
gr_udc-gr_probe(2154): could not find amba slave with id's 00000001 00000021
Attaching grlib ps2 mouse drivers at 0x80000400, irq: 4
Attaching grlib ps2 keyboard drivers at 0x80000500, irq: 5
Netfilter messages via NETLINK v0.30.
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP cubic registered
NET: Registered protocol family 17
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 136k freed

init started: BusyBox v1.8.2 (2011-03-06 14:47:37 EST)
starting pid 15, tty '': '/etc/init.d/rcS'
starting pid 26, tty '': '/bin/sh'
/ # dhcline  ent eth0
Internet Software Consortium DHCP Client 2.0pl5
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

sh: cannot exec 'if': No such file or directory
sh: cannot exec 'then': No such file or directory
sh: cannot open '/etc/dhclient-exit-hooks'
sh: cannot exec 'fi': No such file or directory
sh: cannot exec '-r'': No such file or directory
sh: cannot exec ':': No such file or directory
sed: bad option in substitution expression
sed: bad option in substitution expression
[: 2: unknown operand
[: 2: unknown operand
sh: cannot exec 'exit_with_hooks': No such file or directory
sh: syntax error
Listening on LPF/eth0/00:0a:35:01:d2:66
Sending on   LPF/eth0/00:0a:35:01:d2:66
Sending on   Socket/fallback/fallback-net
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
```

```
sh: cannot exec 'if': No such file or directory
sh: cannot exec 'then': No such file or directory
sh: cannot open '/etc/dhclient-exit-hooks'
sh: cannot exec 'fi': No such file or directory
sh: cannot exec '-r``': No such file or directory
sh: cannot exec ':': No such file or directory
sed: bad option in substitution expression
sed: bad option in substitution expression
sh: syntax error
bound to 192.168.1.118 -- renewal in 43200 seconds.
```

B.4 DHClient with BASH Log

Listing 7: DHClient with BASH

```
=~=~=~=~=~=~=~=~=~=~=~=~=~= PuTTY log 2011.03.08 15:31:48 =~=~=~=~=~=~=~=~=~=~=~=~=~=
Booting Linux
Booting Linux...
PROMLIB: Sun Boot Prom Version 0 Revision 0

Linux version 2.6.21.1 (avery@atlantis) (gcc version 3.4.4) #13 Mon Mar 7 17:44:18 EST
    2011

ARCH: LEON

Attaching textvga drivers [0xf01fcf34 s:0x1fe0]: ##! error GAISLER_VGA(0x61) not found

TYPE: Leon2/3 System-on-a-Chip

Ethernet address: 0:0:0:0:0:0

CACHE: direct mapped cache, set size 8k

Boot time fixup v1.6. 4/Mar/98 Jakub Jelinek (jj@ultra.linux.cz). Patching kernel for
    srmmu[Leon2]/iommu

64MB HIGHMEM available.

Nocache: 0xfc000000-0xfc400000, 1024 pages [128-1280]

node 2: /cpu00 (type:cpu) (props:.node device_type mid mmu-nctx clock-frequency
    uart1_baud uart2_baud )

node 3: /a: (type:serial) (props:.node device_type name )

node 4: /ambapp0 (type:ambapp) (props:.node device_type name )

PROM: Built device tree from rootnode 1 with 1478 bytes of memory.

DEBUG: psr.impl = 0xf  fsr.vers = 0x7

Built 1 zonelists.  Total pages: 64485

Kernel command line: console=ttyS0,38400 root=254:1 rw init=/sbin/init

PID hash table entries: 1024 (order: 10, 4096 bytes)

Todo: init master_l10_counter

Attaching grlib apbuart serial drivers (clk:80hz):

Console: colour dummy device 80x25

Dentry cache hash table entries: 32768 (order: 5, 131072 bytes)

Inode-cache hash table entries: 16384 (order: 4, 65536 bytes)

pkbase: 0xfc800000 pkend: 0xfcc00000 fixstart 0xfcff1000
Memory: 253504k/262144k available (1656k kernel code, 8512k reserved, 204k data, 136k
    init, 65536k highmem)
```

```
Mount-cache hash table entries: 512
NET: Registered protocol family 16
Generic PHY: Registered new driver
SCSI subsystem initialized
NET: Registered protocol family 2
IP route cache hash table entries: 2048 (order: 1, 8192 bytes)
TCP established hash table entries: 8192 (order: 4, 65536 bytes)
TCP bind hash table entries: 8192 (order: 3, 32768 bytes)
TCP: Hash tables configured (established 8192 bind 8192)
TCP reno registered
leon: power management initialized
highmem bounce pool size: 64 pages
io scheduler noop registered
io scheduler cfq registered (default)
GRVGA: Defaulting to 640x480 8-bit resolution
Framebuffer address from node : 0xf0980000
Framebuffer address from videomemory : 0x40980000
fb0: Gaisler frame buffer device, using 300K of video memory
grlib apbuart: 1 serial driver(s) at [0x80000100(irq 2)]
grlib apbuart: system frequency: 80000 khz, baud rates: 38400 38400
ttyS0 at MMIO 0x80000100 (irq = 2) is a Leon
Testing fifo size for UART port 0: got 4 bytes.
RAMDISK driver initialized: 16 RAM disks of 4096K size 1024 blocksize
loop: loaded (max 8 devices)
xsysace xsysace.0: Xilinx SystemACE revision 1.0.12
xsysace xsysace.0: capacity: 3906560 sectors
 xsa: xsa1 xsa2
Xilinx SystemACE device driver, major=254
Marvell 88E1101: Registered new driver
```

```
Marvell 88E1111: Registered new driver
Marvell 88E1145: Registered new driver
Probing GRETH Ethernet Core at 0x80000b00
Detected MARVELL 88EE1111 Revision 2
10/100 GRETH Ethermac at [0x80000b00] irq 12. Running 100 Mbps full duplex
gr_udc-gr_probe(2154): could not find amba slave with id's 00000001 00000021
Attaching grlib ps2 mouse drivers at 0x80000400, irq: 4
Attaching grlib ps2 keyboard drivers at 0x80000500, irq: 5
Netfilter messages via NETLINK v0.30.
ip_tables: (C) 2000-2006 Netfilter Core Team
TCP cubic registered
NET: Registered protocol family 17
VFS: Mounted root (ext2 filesystem).
Freeing unused kernel memory: 136k freed

init started: BusyBox v1.8.2 (2011-03-06 14:47:37 EST)
starting pid 15, tty '': '/etc/init.d/rcS'
starting pid 26, tty '': '/bin/sh'
sh-3.1# echo $SHELL
/bin/sh
sh-3.1# which bash
/bin/bash
sh-3.1# dhclient eth0
Internet Software Consortium DHCP Client 2.0pl5
Copyright 1995, 1996, 1997, 1998, 1999 The Internet Software Consortium.
All rights reserved.

Please contribute if you find this software useful.
For info, please visit http://www.isc.org/dhcp-contrib.html

/bin/sh: error while loading shared libraries: libdl.so.2: cannot open shared object
    file: No such file or directory
Listening on LPF/eth0/00:0a:35:01:d2:66
Sending on   LPF/eth0/00:0a:35:01:d2:66
Sending on   Socket/fallback/fallback-net
DHCPREQUEST on eth0 to 255.255.255.255 port 67
DHCPACK from 192.168.1.1
/bin/sh: error while loading shared libraries: libdl.so.2: cannot open shared object
    file: No such file or directory
bound to 192.168.1.118 -- renewal in 43200 seconds.
sh-3.1# halt

The system is going down NOW!
```

```
Sending SIGTERM to all processes
Terminated
sh-3.1#
Sending SIGKILL to all processes

Requesting system halt
System halted.

Halt
```

C Embedded Linux from Scratch for the LEON3 SPARC

This guide is adapted from `http://cross-lfs.org/view/clfs-embedded/arm` to work on the LEON3 Sparc-V8-compatible hardware. It assumes that the SPARC processor has an FPU built into it.

This guide was originally tested and performed on Ubuntu 10.04 LTS. Hence, `/media` for the base of the mountpoint directory.

C.1 Setting up the build environment

C.1.1 Make a new partition

Use the program “GParted” to make an Ext3 partition of 2GB or more, and label it “CLFS”. Non-Ubuntu systems will probably have their own particular way of doing this that depends on where disks are mounted.

C.1.2 Mount the new partition

- Mount the partition: see your distribution for details. On Ubuntu, click "Places > CLFS" in the GNOME Panel.
- Set up an environment variable that points to the partition

```
export CLFS=/media/CLFS
```

- Create the source directories (as root)

```
sudo mkdir -v $CLFS/sources
sudo chmod -v a+wt $CLFS/sources
```

C.1.3 Download the CLFS-embedded sources

- Enter the sources directory

```
cd $CLFS/sources
```

- Make a list of all the sources

```
cat > embedded_list.txt << EOF
http://ftp.gnu.org/gnu/binutils/binutils-2.21.tar.bz2
http://busybox.net/downloads/busybox-1.17.3.tar.bz2
http://cross-lfs.org/files/packages/embedded-0.0.1/clfs-embedded-
    bootscripts-1.0-pre5.tar.bz2
http://downloads.sourceforge.net/e2fsprogs/e2fsprogs-1.41.14.tar.gz
ftp://gcc.gnu.org/pub/gcc/releases/gcc-4.5.2/gcc-4.5.2.tar.bz2
http://ftp.gnu.org/gnu/gmp/gmp-5.0.1.tar.bz2
http://cross-lfs.org/files/packages/embedded-0.0.1/iana-etc-2.30.tar.bz2
http://www.kernel.org/pub/linux/kernel/v2.6/linux-2.6.36.3.tar.bz2
http://www.multiprecision.org/mpc/download/mpc-0.8.2.tar.gz
http://gforge.inria.fr/frs/download.php/27105/mpfr-3.0.0.tar.bz2
http://www.uclibc.org/downloads/uClibc-0.9.31.tar.bz2
http://downloads.sourceforge.net/libpng/zlib-1.2.3.tar.gz EOF
```

- Download these packages: (`-no-clobber` lets you skip what you already have)

```
wget --no-clobber -i embedded_list.txt
```

- Make a list of the Leon3-specific files:

```
cat > leon3_list.txt << EOF
ftp://gaisler.com/gaisler.com/linux/linux-2.6/kernel/mklinuximg
    -2.6.36-1.0.8.tar.bz2
ftp://gaisler.com/gaisler.com/linux/linux-2.6/kernel/leon-linux
    -2.6-2.6.36.3-1.0.1.tar.bz2
ftp://gaisler.com/gaisler.com/linux/linux-2.6/kernel/grlib-linux-drvpkg
    -1.0.0.tar.bz2
EOF
```

- Download these files, too.

```
wget --no-clobber -i leon3_list.txt
```

- Make a list of all patches from CLFS:

```
cat > patch_list.txt << EOF
http://patches.cross-lfs.org/embedded-dev/busybox-1.17.3-config-1.patch
```

```
http://patches.cross-lfs.org/embedded-dev/busybox-1.17.3-fixes-1.patch
http://patches.cross-lfs.org/embedded-dev/iana-etc-2.30-update-1.patch
http://patches.cross-lfs.org/embedded-dev/uClibc-0.9.31-configs-2.patch
EOF
```

- Download the patches

```
wget --no-clobber -i patch_list.txt
```

C.1.4 Add the CLFS user

- Become root to administer users, and keep CLFS set

```
sudo -s CLFS=$CLFS
```

- (As root) Create the group and user "clfs"

```
groupadd clfs
useradd -s /bin/bash -g clfs -m -k /dev/null clfs
passwd clfs
```

- (As root) Give the clfs user access to $CLFS – Note: $CLFS should still be set. This should not error.

```
chown -Rv clfs ${CLFS}
```

- Surrender root privileges

```
exit
```

- Become the clfs user

```
su - clfs
```

C.1.5 Configuring the clfs user

- Bash profile (when in a login shell)

```
cat > ~/.bash_profile << "EOF"
exec env -i HOME=${HOME} TERM=${TERM} PS1='\u:\w\$ ' /bin/bash
EOF
```

- Bash rc (for non-login shells)

```
cat > ~/.bashrc << "EOF"
set +h
umask 022
CLFS=/media/CLFS
LC_ALL=POSIX
PATH=${CLFS}/cross-tools/bin:/bin:/usr/bin
export CLFS LC_ALL PATH
EOF
```

- Load the new bash profile

```
~/.bash_profile
```

C.1.6 Preparing the filesystem

- Create directories necessary for Linux to work

```
mkdir -pv ${CLFS}/{bin,boot,dev,{etc/,}opt,home,lib/{firmware,modules},
    mnt}
mkdir -pv ${CLFS}/{proc,media/{floppy,cdrom},sbin,srv,sys}
mkdir -pv ${CLFS}/var/{lock,log,mail,run,spool}
mkdir -pv ${CLFS}/var/{opt,cache,lib/{misc,locate},local}
install -dv -m 0750 ${CLFS}/root
install -dv -m 1777 ${CLFS}{/var,}/tmp
mkdir -pv ${CLFS}/usr/{,local/}{bin,include,lib,sbin,src}
mkdir -pv ${CLFS}/usr/{,local/}share/{doc,info,locale,man}
mkdir -pv ${CLFS}/usr/{,local/}share/{misc,terminfo,zoneinfo}
mkdir -pv ${CLFS}/usr/{,local/}share/man/man{1,2,3,4,5,6,7,8}
for dir in ${CLFS}/usr{,/local}; do
  ln -sv share/{man,doc,info} ${dir}
done
```

- Make the cross-tools directories

```
install -dv $CLFS/cross-tools{,/bin}
```

- Set up the necessary files for Linux

  Mtab symlink (list of mounted file systems)

```
ln -svf ../proc/mounts ${CLFS}/etc/mtab
```

  Passwd (list of users)

```
cat > ${CLFS}/etc/passwd << "EOF"
root::0:0:root:/root:/bin/ash
daemon:x:2:6:daemon:/sbin:/bin/false
EOF
```

  Group (list of groups)

```
cat > ${CLFS}/etc/group << "EOF"
root:x:0:
bin:x:1:
sys:x:2:
kmem:x:3:
tty:x:4:
tape:x:5:
daemon:x:6:
floppy:x:7:
disk:x:8:
lp:x:9:
dialout:x:10:
audio:x:11:
video:x:12:
utmp:x:13:
usb:x:14:
cdrom:x:15:
EOF
```

  Log files for login, agetty, and init

```
touch ${CLFS}/var/run/utmp ${CLFS}/var/log/{btmp,lastlog,wtmp}
chmod -v 664 ${CLFS}/var/run/utmp ${CLFS}/var/log/lastlog
```

C.2 Making the Cross-Compile Tools

C.2.1 Setting CFLAGS

```
unset CFLAGS
unset CXXFLAGS
echo unset CFLAGS >> ~/.bashrc
echo unset CXXFLAGS >> ~/.bashrc
```

C.2.2 Setting build settings

Note: the CPU setting below was taken from the post at

http://gcc.gnu.org/ml/gcc/2010-11/msg00440.html

- Temporarily

```
export CLFS_HOST=$(echo ${MACHTYPE} | sed "s/-[^-]*/-cross/")
export CLFS_TARGET="sparc-embedded-linux-gnu"
export CLFS_ARCH="sparc"
export CLFS_CPU="sparchfleonv8"
export CLFS_FLOAT="hard"
```

- Persistently (you can log out and this will re-load on login)

```
echo export CLFS_HOST=\""$CLFS_HOST\"" >> ~/.bashrc
echo export CLFS_TARGET=\""$CLFS_TARGET\"" >> ~/.bashrc
echo export CLFS_ARCH=\""$CLFS_ARCH\"" >> ~/.bashrc
echo export CLFS_CPU=\""$CLFS_CPU\"" >> ~/.bashrc
echo export CLFS_FLOAT=\""$CLFS_FLOAT\"" >> ~/.bashrc
```

C.2.3 Install the Linux headers

- Unpack

```
cd $CLFS/sources
tar xf linux-2.6.36.3.tar.bz2
tar xf leon-linux-2.6-2.6.36.3-1.0.1.tar.bz2
tar xf grlib-linux-drvpkg-1.0.0.tar.bz2
```

- Apply Leon patches

```
cd linux-2.6.36.3
for file in `ls ../leon-linux-2.6-2.6.36.3-1.0.1/patches/*.patch`; do
  echo $file; patch -Np1 < $file; sleep 0.5; done
cp -r ../grlib-linux-drvpkg-1.0.0/kernel/drivers .
patch -p1 < drivers/grlib/add_grlib_tree.patch
if test $? -ne 0; then
  echo "obj-\$(CONFIG_SPARC_LEON) += grlib/"; fi >> drivers/Makefile
```

- Install headers

```
make mrproper
make ARCH=${CLFS_ARCH} headers_check
make ARCH=${CLFS_ARCH} INSTALL_HDR_PATH=dest headers_install
cp -rv dest/include/* ${CLFS}/usr/include
```

- Cleanup

```
cd $CLFS/sources
rm -rf linux-2.6.36.3
rm -rf leon-linux-2.6-2.6.36.3-1.0.1
rm -rf grlib-linux-drvpkg-1.0.0
```

C.2.4 Install GMP-5.0.1

```
cd $CLFS/sources
```

- Unpack

```
tar xf gmp-5.0.1.tar.bz2
```

- Configure

```
cd gmp-5.0.1
CPPFLAGS=-fexceptions ./configure --prefix=${CLFS}/cross-tools
```

- Make, time the result (2m11s for me.)

```
time make
```

- (Optional) Check that it's correct (2m46s for me)

```
time make check
```

- Install

```
make install
```

- Cleanup

```
cd $CLFS/sources
rm -rf gmp-5.0.1
```

C.2.5 MPFR-3.0.0

- Unpack

```
tar xf mpfr-3.0.0.tar.bz2
cd mpfr-3.0.0
```

- Configure

```
LDFLAGS="-Wl,-rpath,${CLFS}/cross-tools/lib" \
./configure --prefix=${CLFS}/cross-tools --enable-shared \
  --with-gmp=${CLFS}/cross-tools
```

- Make and install (1m29s for me)

```
time { make && make install; }
```

- Cleanup

```
cd $CLFS/sources
rm -rf mpfr-3.0.0
```

C.2.6 MPC-0.8.2

- Unpack

```
tar xf mpc-0.8.2.tar.gz
cd mpc-0.8.2
```

- Configure

```
LDFLAGS="-Wl,-rpath,${CLFS}/cross-tools/lib" \
  ./configure --prefix=${CLFS}/cross-tools \
  --with-gmp=${CLFS}/cross-tools \
  --with-mpfr=${CLFS}/cross-tools
```

- Build and install (About 21 seconds)

```
time { make && make install; }
```

- Cleanup

```
cd $CLFS/sources rm -rf mpc-0.8.2
```

C.2.7 Cross Binutils-2.21

- Extract

```
cd $CLFS/sources
tar xf binutils-2.21.tar.bz2
```

- Make a temporary directory

```
mkdir binutils-cross
cd binutils-cross
```

- Configure

```
../binutils-2.21/configure --prefix=${CLFS}/cross-tools \
  --target=${CLFS_TARGET} --with-sysroot=${CLFS} --disable-nls \
  --enable-shared --disable-multilib
```

- Build and install (5m11s)

```
time { make configure-host && make && make install; }
```

- Post-install maintenance

```
cp -v ../binutils-2.21/include/libiberty.h ${CLFS}/usr/include
```

- Cleanup

```
cd $CLFS/sources && rm -rf binutils-2.21 binutils-cross
```

C.2.8 Cross GCC - 4.5.2 – Static

- Extract

```
cd $CLFS/sources tar xf gcc-4.5.2.tar.bz2
```

- Make a temporary directory

```
mkdir -v gcc-cross-static cd gcc-cross-static
```

- Configure

```
AR=ar LDFLAGS="-Wl,-rpath,${CLFS}/cross-tools/lib" \
  ../gcc-4.5.2/configure --prefix=${CLFS}/cross-tools \
  --build=${CLFS_HOST} --host=${CLFS_HOST} --target=${CLFS_TARGET} \
  --with-sysroot=${CLFS} --disable-nls --disable-shared \
  --with-mpfr=${CLFS}/cross-tools --with-gmp=${CLFS}/cross-tools \
  --with-mpc=${CLFS}/cross-tools --without-headers --with-newlib \
```

```
  --disable-decimal-float --disable-libgomp --disable-libmudflap \
  --disable-libssp --disable-threads --enable-languages=c \
  --disable-multilib --with-float=${CLFS_FLOAT}
```

- Build (13m4s)

```
time make all-gcc all-target-libgcc
```

- Install (4 seconds) time

```
make install-gcc install-target-libgcc
```

- Cleanup

```
cd $CLFS/sources && rm -rf gcc-4.5.2 gcc-cross-static
```

C.2.9 uClibc-0.9.31

- Extract

```
tar xf uClibc-0.9.31.tar.bz2
```

- Patch cd uClibc-0.9.31

```
patch -Np1 -i ../uClibc-0.9.31-configs-2.patch
cp -v clfs/config.arm.big .config
```

- Configure Set arch=sparc (v8 will be selected automatically), and set FPU to be what you have in the Leon (Y/N)

```
make menuconfig
```

- Build (1m39s)

```
time make
```

- Install

```
make PREFIX=${CLFS} install
```

- Cleanup

```
cd $CLFS/sources && rm -rf uClibc-0.9.31
```

C.2.10 GCC-4.5.2 – Full

- Extract

```
cd $CLFS/sources tar xf gcc-4.5.2.tar.bz2
```

- Make build directories

```
mkdir gcc-cross-full cd gcc-cross-full
```

- Configure

```
AR=ar LDFLAGS="-Wl,-rpath,${CLFS}/cross-tools/lib" \
  ../gcc-4.5.2/configure --prefix=${CLFS}/cross-tools \
  --build=${CLFS_HOST} --target=${CLFS_TARGET} --host=${CLFS_HOST} \
  --with-sysroot=${CLFS} --disable-nls --enable-shared \
  --enable-languages=c,c++ --enable-c99 --enable-long-long \
  --with-mpfr=${CLFS}/cross-tools --with-gmp=${CLFS}/cross-tools \
  --with-mpc=${CLFS}/cross-tools --disable-multilib \
  --with-float=${CLFS_FLOAT}
```

- Build (20m02s)

```
time make
```

- Install

```
make install
```

- Cleanup

```
cd $CLFS/sources && rm -rf gcc-cross-full gcc-4.5.2
```

C.3 Building the System

C.3.1 Set up cross-compiling variables for convenience

```
echo export CC=\""${CLFS_TARGET}-gcc\"" >> ~/.bashrc
echo export CXX=\""${CLFS_TARGET}-gcc\"" >> ~/.bashrc
echo export AR=\""${CLFS_TARGET}-ar\"" >> ~/.bashrc
echo exportAS=\""${CLFS_TARGET}-as\"" >> ~/.bashrc
echo export LD=\""${CLFS_TARGET}-ld\"" >> ~/.bashrc
echo export RANLIB=\""${CLFS_TARGET}-ranlib\"" >> ~/.bashrc
echo export READELF=\""${CLFS_TARGET}-readelf\"" >> ~/.bashrc
echo export STRIP=\""${CLFS_TARGET}-strip\"" >> ~/.bashrc
source ~/.bashrc
```

C.3.2 busybox-1.17.3

-
- Extract

```
cd $CLFS/sources tar xf busybox-1.17.3.tar.bz2
```

- Patch

```
cd busybox-1.17.3
patch -Np1 -i ../busybox-1.17.3-fixes-1.patch
patch -Np1 -i ../busybox-1.17.3-config-1.patch
```

- Configuration Check Should not require input, just checks that the configuration is acceptable

```
cp -v clfs/config .config make oldconfig
```

- Build

```
make CROSS_COMPILE="${CLFS_TARGET}-"
```

- Install

```
make CROSS_COMPILE="${CLFS_TARGET}-" CONFIG_PREFIX="${CLFS}" install
```

- Cleanup

```
cd $CLFS/sources && rm -rf busybox-1.17.3
```

C.3.3 e2fsprogs-1.41.14

- Extract

```
tar xf e2fsprogs-1.41.14.tar.gz
cd e2fsprogs-1.41.14
```

- Make an internal build directory

```
mkdir -v build
cd build
```

- Configure

```
CC="${CC} -Os" ../configure --prefix=/usr --with-root-prefix="" \
  --host=${CLFS_TARGET} --disable-tls --disable-debugfs \
  --disable-e2initrd-helper --disable-nls \
```

- Build (53 seconds)

```
time make
```

- Install

```
make DESTDIR=${CLFS} install install-libs
```

- Cleanup

```
cd $CLFS/sources && rm -rf e2fsprogs-1.41.14
```

C.3.4 iana-etc-2.30

- Extract

```
tar xf iana-etc-2.30.tar.bz2
cd iana-etc-2.30
```

- Patch

```
patch -Np1 -i ../iana-etc-2.30-update-1.patch
```

- Make information up-to-date

```
make get
```

- Build (<1 second)

```
time make
```

- Install

```
make DESTDIR=${CLFS} install
```

- Cleanup

```
cd $CLFS/sources && rm -rf iana-etc-2.30
```

C.3.5 zlib-1.2.3

- Extract

```
tar xf zlib-1.2.3.tar.gz
cd zlib-1.2.3
```

- Patch

```
cp configure{,.orig}
sed -e 's/-O3/-Os/g' configure.orig > configure
```

- Configure

```
./configure --prefix=/usr --shared
```

- Build (<5 seconds)

```
time make
```

- Install

```
make prefix=${CLFS}/usr install
mv -v ${CLFS}/usr/lib/libz.so.* ${CLFS}/lib
ln -svf ../../lib/libz.so.1 ${CLFS}/usr/lib/libz.so
```

- Cleanup

```
cd $CLFS/sources && rm -rf zlib-1.2.3
```

C.3.6 Create the fstab file

```
cat > ${CLFS}/etc/fstab << "EOF"
# Begin /etc/fstab
# This assumes that your CF card has Ext3 on partition 1
#     and swap on partition 2.
# file system mount-point type    options          dump  fsck
#                                                       order
/dev/sda1     /            ext3    defaults         1     1
/dev/sda2     swap         swap    pri=1            0     0
proc          /proc        proc    defaults         0     0
sysfs         /sys         sysfs   defaults         0     0
devpts        /dev/pts     devpts  gid=4,mode=620   0     0
shm           /dev/shm     tmpfs   defaults         0     0
# End /etc/fstab
EOF
```

C.3.7 Compile the Linux kernel

- Extract

```
cd $CLFS/sources
tar xf linux-2.6.36.3.tar.bz2
tar xf leon-linux-2.6-2.6.36.3-1.0.1.tar.bz2
tar xf grlib-linux-drvpkg-1.0.0.tar.bz2
tar xf mklinuximg-2.6.36-1.0.8.tar.bz2
```

- Apply Leon patches

```
cd linux-2.6.36.3
for file in `ls ../leon-linux-2.6-2.6.36.3-1.0.1/patches/*.patch`; do
  echo $file; patch -Np1 < $file; done
cp -r ../grlib-linux-drvpkg-1.0.0/kernel/drivers .
patch -p1 < drivers/grlib/add_grlib_tree.patch
if test $? -ne 0; then
  echo "obj-\$(CONFIG_SPARC_LEON) += grlib/"; fi >> drivers/Makefile
```

- Prepare source

```
make mrproper
```

- Configure – don't make modules.

```
make ARCH=${CLFS_ARCH} CROSS_COMPILE="${CLFS_TARGET}-" menuconfig
```

- Build

```
time make ARCH=${CLFS_ARCH} CROSS_COMPILE=${CLFS_TARGET}- zImage
```

- Make the GRMON image Look up the MAC address of your device's ethernet port, then substitute it for the 001122334455 below.

```
cd ${CLFS}/sources/mklinuximg-2.6.36-1.0.8
sed -i "s@sparc-linux-@$CLFS_TARGET-@g" mklinuximg
./mklinuximg ../linux-2.6.36.3/arch/${CLFS_ARCH}/boot/image image.dsu \
  -cmdline "console=ttyS0,38400 root=/dev/sda1" -ethmac 001122334455
cp image.dsu ${CLFS}/boot/
cd ../linux-2.6.36.3
```

- Install the kernel

```
cp arch/${CLFS_ARCH}/boot/uImage ${CLFS}/boot/clfskernel-2.6.36.3
cp System.map ${CLFS}/boot/System.map-2.6.36.3
cp .config ${CLFS}/boot/config-2.6.36.3
```

- Cleanup

```
cd $CLFS/sources
rm -rf linux-2.6.36.3 leon-linux-2.6-2.6.36.3-1.0.1 grlib-linux-drvpkg
   -1.0.0
```

C.3.8 Giving control to root

- Become root

```
su -s CLFS=\$CLFS"
```

- Change the ownership of all of $CLFS

```
chown -Rv root:root ${CLFS}
```

- Switch a few to another group

```
chgrp -v utmp ${CLFS}/var/run/utmp ${CLFS}/var/log/lastlog
```

- Create the two most important nodes

```
mknod -m 0666 ${CLFS}/dev/null c 1 3
mknod -m 0600 ${CLFS}/dev/console c 5 1
```

C.3.9 CLFS-Bootscripts-1.0-pre5

- Extract

```
cd ${CLFS}/sources
tar xf clfs-embedded-bootscripts-1.0-pre5.tar.bz2
cd clfs-embedded-bootscripts-1.0-pre5
```

item Install

```
mkdir ${CLFS}/etc/rc.d
mkdir ${CLFS}/etc/rc.d/{init.d,start,stop}
make DESTDIR=${CLFS} install-bootscripts
```

- Cleanup

```
cd ${CLFS}/sources && rm -rf clfs-embedded-bootscripts-1.0-pre5
```

C.3.10 mdev

- Fill the file

```
cat > ${CLFS}/etc/mdev.conf << "EOF"
# /etc/mdev/conf
# Symlinks:
# Syntax: %s -> %s

MAKEDEV -> ../sbin/MAKEDEV
/proc/core -> kcore
fd -> /proc/self/fd
mcdx -> mcdx0
radio -> radio0
ram -> ram1
sbpcd -> sbpcd0
sr0 -> scd0
sr1 -> scd1
sr10 -> scd10
sr11 -> scd11
sr12 -> scd12
sr13 -> scd13
sr14 -> scd14
sr15 -> scd15
sr16 -> scd16
sr2 -> scd2
sr3 -> scd3
sr4 -> scd4
sr5 -> scd5
sr6 -> scd6
sr7 -> scd7
sr8 -> scd8
sr9 -> scd9
```

```
stderr -> fd/2
stdin -> fd/0
stdout -> fd/1

# Remove these devices, if using a headless system
# You will see an error mdev: Bad line 35
vbi -> vbi0
vcs -> vcs0
vcsa -> vcsa0
video -> video0
# Stop Remove for headless system

# Devices:
# Syntax: %s %d:%d %s
# devices user:group mode

null 0:0 777
zero 0:0 666

urandom 0:0 444

console 0:5 0600
fd0 0:11 0660
hdc 0:6 0660
kmem 0:9 000
mem 0:9 0640
port 0:9 0640
ptmx 0:5 0660
tun[0-9]* 0:0 0640 =net/

sda* 0:6 0660
sdb* 0:6 0660
hda* 0:6 0660
hdb* 0:6 0660

tty 0:5 0660
```

```
tty0* 0:5 0660
tty1* 0:5 0660
tty2* 0:5 0660
tty3* 0:5 0660
tty4* 0:5 0660
tty5* 0:5 0660
tty6* 0:5 0660

ttyS* 0:20 640
EOF
```

C.3.11 Profile

- Create the file

```
cat > ${CLFS}/etc/profile << "EOF"
# /etc/profile

# Set the initial path
export PATH=/bin:/usr/bin

if [ `id -u` -eq 0 ] ; then
        PATH=/bin:/sbin:/usr/bin:/usr/sbin
        unset HISTFILE
fi

# Setup some environment variables.
export USER=`id -un`
export LOGNAME=$USER
export HOSTNAME=`/bin/hostname`
export HISTSIZE=1000
export HISTFILESIZE=1000
export PAGER='/bin/more '
export EDITOR='/bin/vi'

# End /etc/profile
```

```
EOF
```

C.3.12 Inittab

```
cat > ${CLFS}/etc/inittab << "EOF"
# /etc/inittab

::sysinit:/etc/rc.d/startup

tty1::respawn:/sbin/getty 38400 tty1
tty2::respawn:/sbin/getty 38400 tty2
tty3::respawn:/sbin/getty 38400 tty3
tty4::respawn:/sbin/getty 38400 tty4
tty5::respawn:/sbin/getty 38400 tty5
tty6::respawn:/sbin/getty 38400 tty6

# Put a getty on the serial line (for a terminal)
# uncomment this line if your using a serial console
#::respawn:/sbin/getty -L ttyS0 115200 vt100

::shutdown:/etc/rc.d/shutdown
::ctrlaltdel:/sbin/reboot
EOF
```

C.3.13 Hostname

- Pick a hostname. Set it to `CLFS_HOST` (replace localhost below)

```
CLFS_HOST="localhost"
```

- Make it into a file

```
echo "$CLFS_HOST" > ${CLFS}/etc/HOSTNAME
```

C.3.14 Hosts file

- Fill file. Replace `CLFS_HOSTNAME` with your name, as cat doesn't evaluate it.

```
cat > ${CLFS}/etc/hosts << "EOF"
# Begin /etc/hosts (network card version)

127.0.0.1 localhost
127.0.1.1 CLFS_HOSTNAME

# End /etc/hosts (network card version)
EOF
```

C.3.15 Network configuration

```
cat > ${CLFS}/etc/network.conf << "EOF"
# /etc/network.conf
# Global Networking Configuration
# interface configuration is in /etc/network.d/

# set to yes to enable networking
NETWORKING=yes

# set to yes to set default route to gateway
USE_GATEWAY=no

# set to gateway IP address
GATEWAY=192.168.0.1

# Interfaces to add to br0 bridge
# Leave commented to not setup a network bridge
# Substitute br0 for eth0 in the interface.eth0 sample below to bring up br0
# instead
# bcm47xx with vlans:
#BRIDGE_INTERFACES="eth0.0 eth0.1 wlan0"
# Other access point with a wired eth0 and a wireless wlan0 interface:
```

```
#BRIDGE_INTERFACES="eth0 wlan0"

EOF

mkdir ${CLFS}/etc/network.d
cat > ${CLFS}/etc/network.d/interface.eth0 << "EOF"
# Network Interface Configuration

# network device name
INTERFACE=eth0

# set to yes to use DHCP instead of the settings below
DHCP=no

# IP address
IPADDRESS=192.168.1.2

# netmask
NETMASK=255.255.255.0

# broadcast address
BROADCAST=192.168.1.255
EOF

cat > ${CLFS}/etc/udhcpc.conf << "EOF"
#!/bin/sh
# udhcpc Interface Configuration
# Based on http://lists.debian.org/debian-boot/2002/11/msg00500.html
# udhcpc script edited by Tim Riker <Tim@Rikers.org>

[ -z "$1" ] && echo "Error: should be called from udhcpc" && exit 1

RESOLV_CONF="/etc/resolv.conf"
RESOLV_BAK="/etc/resolv.bak"

[ -n "$broadcast" ] && BROADCAST="broadcast $broadcast"
```

```
[ -n "$subnet" ] && NETMASK="netmask $subnet"

case "$1" in
        deconfig)
                if [ -f "$RESOLV_BAK" ]; then
                        mv "$RESOLV_BAK" "$RESOLV_CONF"
                fi
                /sbin/ifconfig $interface 0.0.0.0
                ;;

        renew|bound)
                /sbin/ifconfig $interface $ip $BROADCAST $NETMASK

                if [ -n "$router" ] ; then
                        while route del default gw 0.0.0.0 dev $interface ; do
                                true
                        done

                        for i in $router ; do
                                route add default gw $i dev $interface
                        done
                fi

                if [ ! -f "$RESOLV_BAK" ] && [ -f "$RESOLV_CONF" ]; then
                        mv "$RESOLV_CONF" "$RESOLV_BAK"
                fi

                echo -n > $RESOLV_CONF
                [ -n "$domain" ] && echo search $domain >> $RESOLV_CONF
                for i in $dns ; do
                        echo nameserver $i >> $RESOLV_CONF
                done
                ;;
esac

exit 0
```

```
EOF

chmod +x ${CLFS}/etc/udhcpc.conf
cat > ${CLFS}/etc/resolv.conf << "EOF"
# Begin /etc/resolv.conf

domain example.org
nameserver 192.168.0.1
nameserver 192.168.1.1

# End /etc/resolv.conf
EOF
```

C.4 Finishing Up

C.4.1 Make a final directory

- Copy to a new directory

```
install -dv ${CLFS}-final
cp -arv ${CLFS}/* ${CLFS}-final/
```

- Remove the unnecessary things

```
rm -rfv ${CLFS}-final/cross-tools
rm -rfv ${CLFS}-final/usr/src/*
rm -rfv ${CLFS}-final/usr/include
rm -rfv ${CLFS}-final/usr/man
rm -rfv ${CLFS}-final/usr/share/man
rm -rfv ${CLFS}-final/sources
FILES="$(ls ${CLFS}-final/lib/*.a ${CLFS}-final/usr/lib/*.a)"
for file in $FILES; do rm -fv $file; done
```

C.4.2 Create a tarball

```
install -dv ${CLFS}/build
cd ${CLFS}-final
tar jcfv ${CLFS}/build/clfs-leon3-embedded.tar.bz2 *
```

D Power Supply Photographs

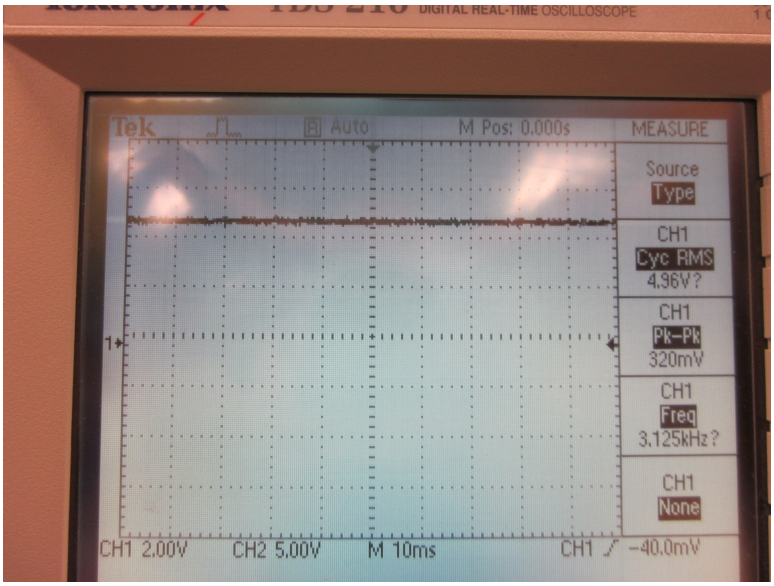

Figure 42: Oscilloscope Reading-Shows about 4.96V

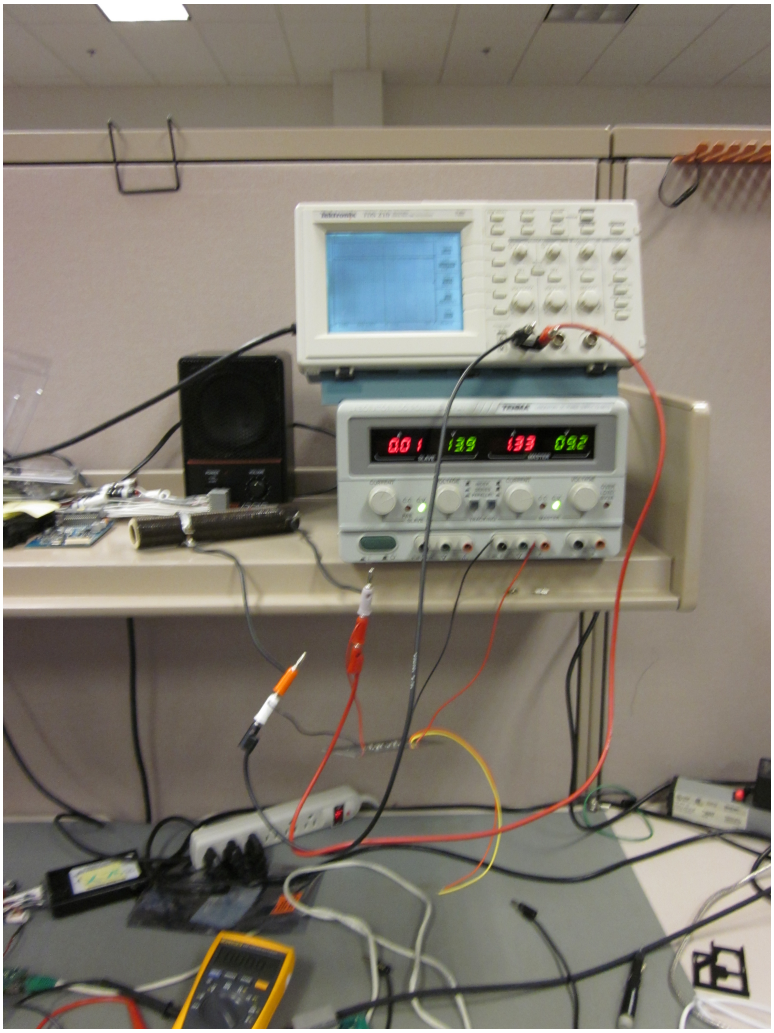

Figure 43: General Test Setup

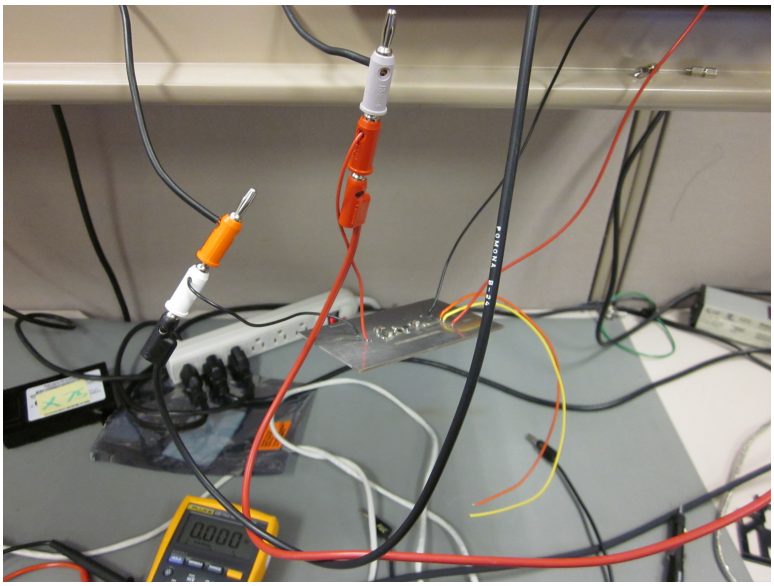

Figure 44: Power Supply Being Tested

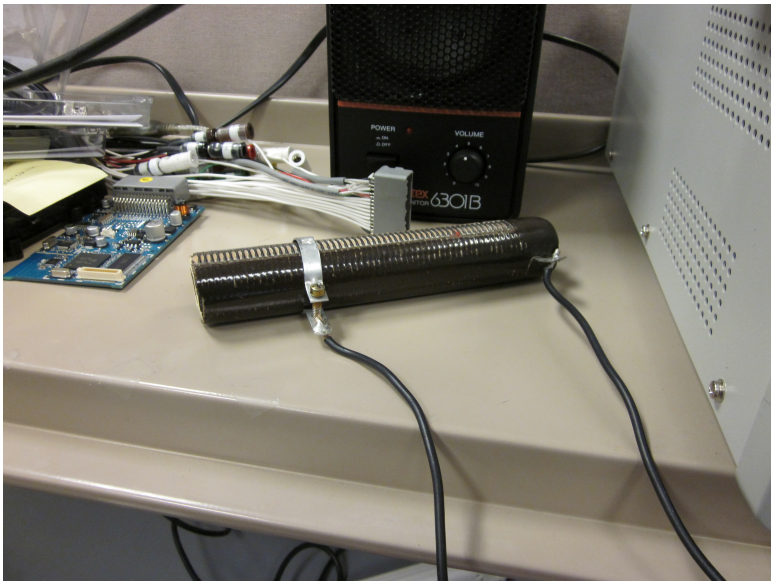

Figure 45: 2.5Ω Load