Optimized fast response power tracking for solar powered vehicles

by

Paul John Dorweiler

A Thesis Submitted to the

Graduate Faculty in Partial Fulfillment of the

Requirements for the Degree of

MASTER OF SCIENCE

Department: Electrical Engineering and Computer Engineering
Major: Computer Engineering

Approved:

Signatures have been redacted for privacy

Iowa State University
Ames, Iowa
1991

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

This report presents a study of the existing solar power system of PrISUm (Iowa State University's solar powered race car which competed in General Motors' Sunrayce USA the summer of 1990), an analysis of the weaknesses in that system, a redesign and implementation of a new solar power system, and an analysis of the performance of the new system.

In Chapter 2, the existing electrical system for PrISUm is presented along with a design history for this system. A description of the performance and reliability problems of this system encountered during Sunrayce follows.

Chapter 3 contains a re-analysis of the differences in solar maximum power tracking between a stationary terrestrial solar panel and one positioned on a moving vehicle. This analysis will show the drastic differences in performance necessary for a power tracking system between the two cases and where the previous PrISUm system weaknesses seriously affected power performance. Also included is a look at the harsh noise environment contained in a solar powered vehicle. Finally, based upon this analysis the desired goals for a new solar power system are defined.

Chapter 4 details the preliminary design of a new solar maximum power tracker whose performance is designed to operate effectively for a moving solar panel. All components of a power tracker are looked at to determine which critical modules must be redesigned.

Chapter 5 presents the implementation of this new solar power system using an Intel 87C196KC microcontroller, along with all of the interface circuitry necessary for this chip. Design flexibility in the performance, features, and operation of this new system are discussed as well. Additional features added to increase the reliability and operational flexibility of the power tracker are shown along with the justification for their inclusion in the new system.

Chapter 6 describes the testing conducted on a prototype of this new solar power system along with a detailed analysis of how well the system meets the performance goals laid out in

Chapter 3. Various methods of maximum power tracking are studied along with their implementation in software on the microcontroller. Additional items helpful for the final production-level system are presented and all relevant noise-reduction techniques to be used in the production version of this system are described in some detail.

Descriptions of the DC-DC boost converter, silicon solar cells, and the 87C196KC microcontroller used are presented in the Appendix, along with a listing of the control code used on the microcontroller for optimum response power tracking on a moving vehicle.

CHAPTER 2. ORIGINAL PrISUm SYSTEM

Introduction

The existing electrical power system of PrISUm was the result of a design conducted by primarily three students, with the goal of centralized control to achieve a high level of system innerconnectivity. With all of the data collection and control functions in one unit, telemetry would be easy to handle and control of all units with one program would be possible. This was particularly desirable in order to integrate the solar power tracking system and the motor controller. The ultimate aim of this integration was to implement a cruise control system with minimum power demand. A secondary computer system was also included which handled the telemetry interaction with the strategy computer and also sampled various solar cell and motor temperatures around the car. Unfortunately, due to problems encountered in the motor and controller subcontracted, computer control of the motor was never achieved.

Another primary function of the control system was to perform a constant numeric integration to determine the amount of energy, rather than charge, flowing into or out of the silver-zinc battery pack by means of a Hall-effect sensor. This data is key to all strategy decisions for a solar race car. A solar car must, in essence, be driven "by its battery". With all of this data and various control functions made available to the driver by means of a rather complex dash display, such mundane car functions as turn signals, brake lights, the horn, and a speedometer were also given to the control computer to keep extraneous power consumption to an absolute minimum. The computer also had control of power provided to various modules so units could be turned off when not needed.

The Actual System

Given enough time and work, the full electrical system would have been implemented for PrISUm. However, time grew short prior to the race and so various parts of the system were

effectively eliminated. The final system only controlled all five of the maximum power point tracker (MPPT) units, the brake lights, turn signals, horn, and power to each of these units. Some minimal battery monitoring was also performed. A conceptual diagram of the full system is shown in Figure 2.1.

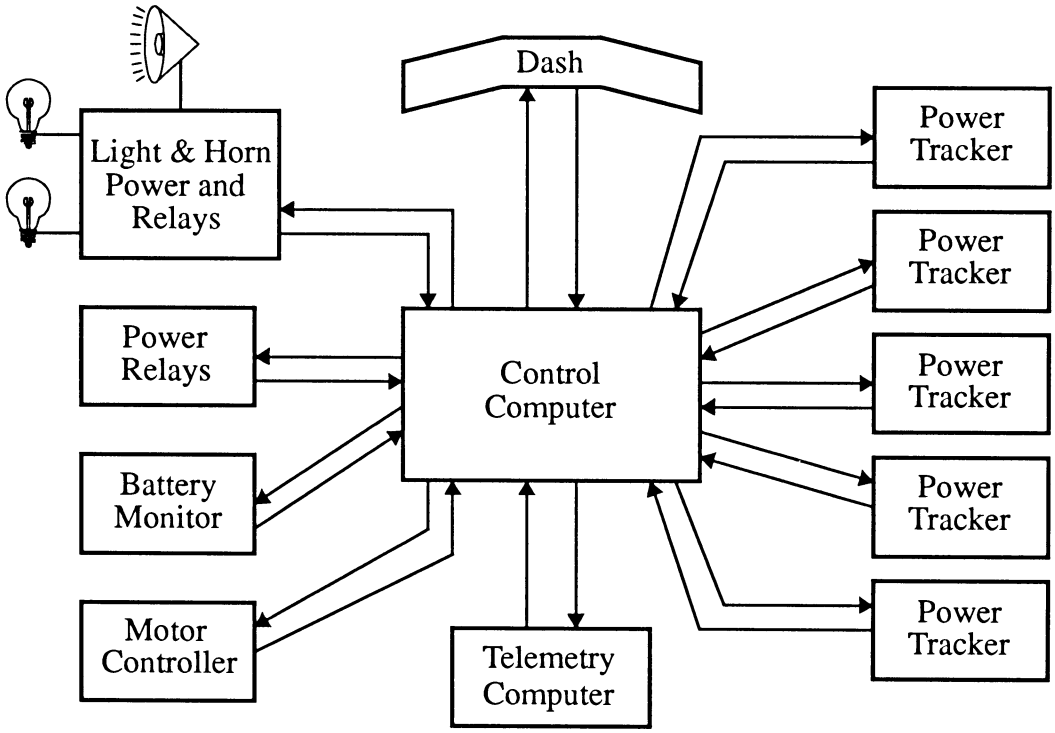

Figure 2.1: Conceptual view of PrISUm electronics

The main control computer for PrISUm consisted of a multiple-board system manufactured by Winsystems, Inc. Six separate boards were included in the cardcage for this computer:

1. the processor board
2. a multichannel 12-bit analog to digital (A/D) converter board
3. a multichannel 12-bit digital to analog (D/A) converter board
4. a general-purpose analog input-output (I/O) board
5. a multichannel TTL I/O board
6. a generic interface board

Each power tracker required two A/D channels for data sampling and one D/A channel for

computer control. The battery monitor used two A/D channels to sample the battery voltage and current flowing into or out of the battery. The TTL lines were used for control of the lights and horn, some of the display lights on the dash, and the power relays. The analog I/O board was to be used for control of the motor using a digital-feedback control algorithm. The generic interface board was used for a custom low-power keyboard encoder and line buffering for the interface to the driver dash unit. An RS-422 serial interface from the main control computer to the telemetry computer was available but never used during Sunrayce.

A power tracker consisted of two primary parts: a DC to DC boost converter to allow the array to run at a different voltage than the battery pack for maximum power, and a Signetics NE5562 analog switched-mode power controller chip. This chip varied the duty cycle of the PWM signal used to drive the power MOSFETs used in the DC-DC converter to perform the voltage boost function. More specific details of a DC-DC boost converter are given in Appendix A. The control computer monitored and adjusted each power tracker in turn. A pictorial of one power tracker and its associated computer control is given in Figure 2.2.

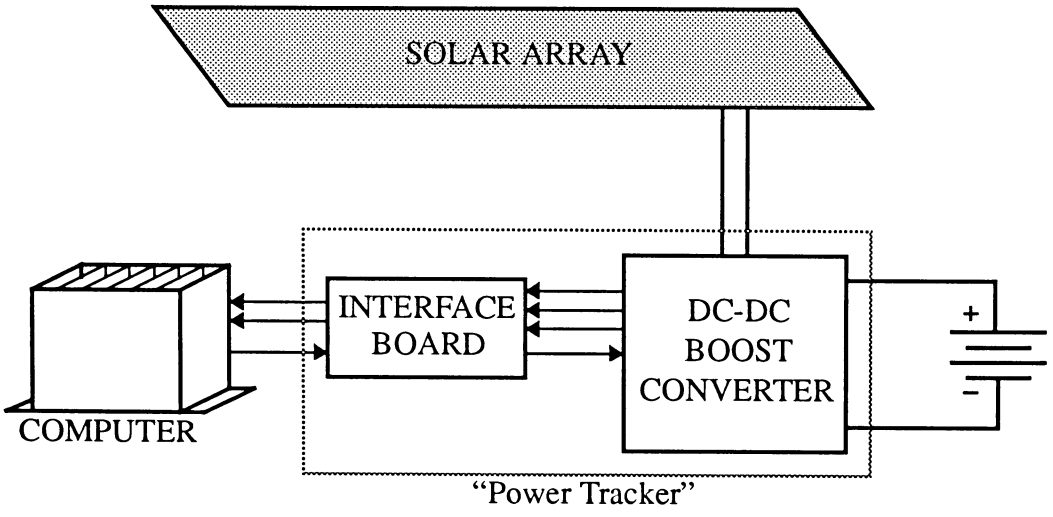

Figure 2.2: Individual power tracker with computer control

Control of each power tracker by computer was accomplished by sampling the array voltage and current flowing from the array, multiplying these values to obtain the power coming from the array at that point in time, comparing this power to previous power values observed, and adjusting a voltage input to the NE5562 chip (thus varying the duty cycle applied to the MOSFET gates). Due to the lack of time prior to the race, little time was spent performing a detailed optimization of the software used to control the power trackers in anything other than a static (non-moving) environment.

Some mention must be made of the complexity of the control software and its impact on the performance of each task delegated to the central control computer. Since each power tracker required two A/D samples per adjustment iteration and other tasks also required A/D conversions for their iterations, the software employed had to "time slice" the A/D converter unit in order to get all of the required voltage samples for each task before calling the task control algorithm. This is very analogous to the sharing of devices between processes in UNIX, and a similar software approach was applied in the control software for PrISUm. The overhead to schedule and perform all of this device and task scheduling was non-negligible and reduced the speed at which each task could be performed. In addition to this scheduling overhead, human-speed interaction (mostly keypress and LCD display activity) also had to be performed by the control computer at irregular intervals.

Sunrayce System Performance and Problems

Solar power collection during the race consisted of three distinct periods during the day, these dictated by the race rules established by GM: a non-moving charge period from 7 to 9AM, the actual race to that day's destination from 9AM to 6:30PM, and another non-moving charge period from 6:30 to 8:30PM. Only solar power from the car's array was allowed to power the car and charge the battery at any time during the race. Any work performed on the car was observed by a GM official and no major performance enhancements were allowed once the race

had started (thankfully, software changes were not included in this category).

Performance of the solar power system of PrISUm changed from day to day due to the fact the system continually evolved from a primitive state toward the original design goal. Computer control of brake lights, turn signals and horn existed from the start of the race, but control of the power trackers by computer did not occur until the third day of the race. Prior to that each tracker was placed in a fixed duty-cycle mode and manually adjusted at stops along the side of the road during the race day. This obviously was not producing maximum solar power from the car's array and much effort was concentrated on getting computer control of the power trackers as quickly as possible.

The first problem initially encountered was an inconsistency of power readings between an analog power meter used to manually adjust the trackers and the power readings from the computer shown on the dash LCD display when the control software was placed in a monitoring-only mode. These power readings also suffered sharp fluctuations with heavy loading on the UNIQ motor and use of the two-way radio by the driver to transmit to the race team. No analysis of this problem was possible at the time, but this was later attributed to the many noise sources present in the car and several design decisions which lead to a lack of noise immunity.

The next problem discovered was the ultimate reason for this work. Once the tracker control algorithms were placed in the control computer's memory they were slowly integrated into the system. Initial algorithms designed to work effectively for a moving panel were tried on a single tracker during the morning and evening charge periods, but proved to be difficult to keep stable. Somewhere in the control chain was a slow system response which made abrupt control signal adjustments to fast-occurring solar changes relatively ineffective. This negated attempts to run control algorithms designed for fast response, and so slower response algorithms were adopted for the remainder of the race. These algorithms sampled and adjusted each tracker two times per second, using previous power values for that panel to determine

which direction to adjust the duty cycle for that panel's DC-DC converter.

At first this algorithm produced an immediate relief of the necessity to stop the car occasionally and adjust the trackers. However, during the remainder of the race I spent quite a bit of time observing the performance of the system while driving the car -- not an easy task, but necessary to give valuable information on car performance for future races. The system proved to be very slow in responding to sharp input changes (sharp cloud or building shadows), taking anywhere from six to ten seconds to find the new maximum power point for each panel. In addition, broken shadows cast from trees and light clouds would cause serious instability in even this simple control code. The inaccuracy of the power readings from the computer also caused serious concern as to whether or not the system was indeed close to maximum power at any given time (later analysis showed the noise affected levels but the general trends were not seriously affected and so close to maximum power was being drawn from each panel most of the time). As will be shown later, the slow system response and data inaccuracy could and were, however, leading to situations where power was not close to maximum and in some circumstances were causing power drain.

The last problem encountered proved ultimately to be the most deadly to the system. With a car came many other mechanical problems which sometimes needed repair. On one occasion it was necessary to re-weld part of the car's aluminum frame. This welding was performed using an electric welder; having over twenty amps AC flowing through the car frame proved to be disastrous to the digital circuitry in the control system. Although the initial design had called for the frame to be electrically isolated to prevent the chance of someone shocking (and possibly killing) themselves while working near the battery, the final packaging of the control system inadvertently lead to a ground leak. As a result, the welding current partially damaged some of the control circuitry, causing intermittent computer "crashes" and other strange occurrences -- on one day late in the race, the left turn signal would begin blinking at random intervals and shut itself off without driver input. More disastrous was the almost

complete failure of the power tracking control system, leading the team to revert to the manual fixed duty cycle mode of the power trackers for the final two days.

In summary, the power data inaccuracy, slow control response time, and unreliability of the centralized computer control system of PrISUm proved to be one of the car's major crutches, rather than the enhancement it was intended to be. Some of this problem was attributable to the lack of time prior to the race to build and test the system, still more to the complexity of the system itself, but most was due to the lack of prior design experience in building a control system for a solar car, which contains many design hazards similar to systems built for commercial aviation or military work.

CHAPTER 3. POWER TRACKING IN A MOVING VEHICLE

Differences Between a Stationary and Moving Array

An analysis of the types of shadows encountered and speed of shadow approach will illustrate the differences in maximum power tracking between stationary and ground-moving arrays. Consider a stationary array placed on the roof of a building, as pictured in Figure 3.1.

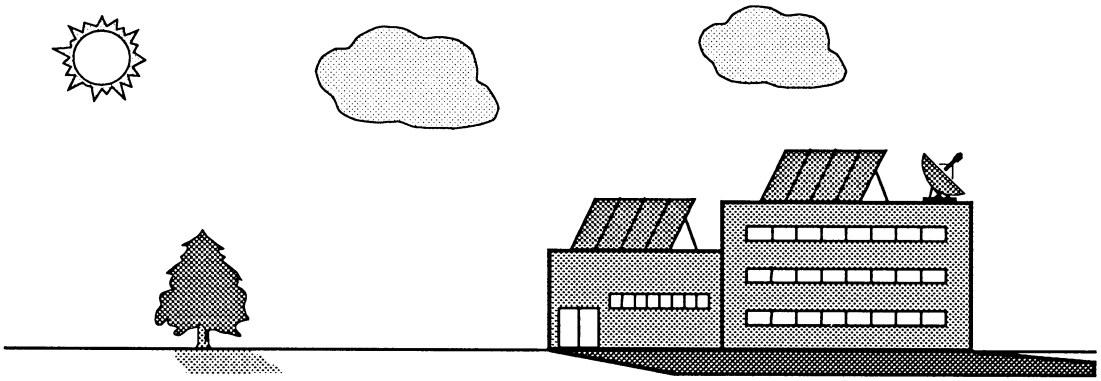

Figure 3.1: Stationary terrestrial solar array

An array placed as depicted will encounter only cloud shadows (unless, of course, the placement of the array puts it directly in the shadow of another, taller building) of varying degrees of solar obscurement. These shadows will overtake the array at the speed with which the cloud is moving overhead. Due to the fact that most clouds move at less than twenty miles an hour and are not sharply defined, the cloud edge (and hence the shadow edge) will overtake the array panel over a period of a few seconds (possibly much longer if the array group is quite large). The slowness of this approach means that a maximum power tracker does not need to shift the voltage of the array (by adjusting the duty cycle of the DC-DC converter switch) that quickly. Thus, a power tracking system with a response time of a second or so per adjustment should do quite well and yet be inexpensive to manufacture.

Next consider a six foot long horizontal panel moving along the ground at twenty-five miles an hour (a fair speed for a solar powered vehicle), as depicted in Figure 3.2.

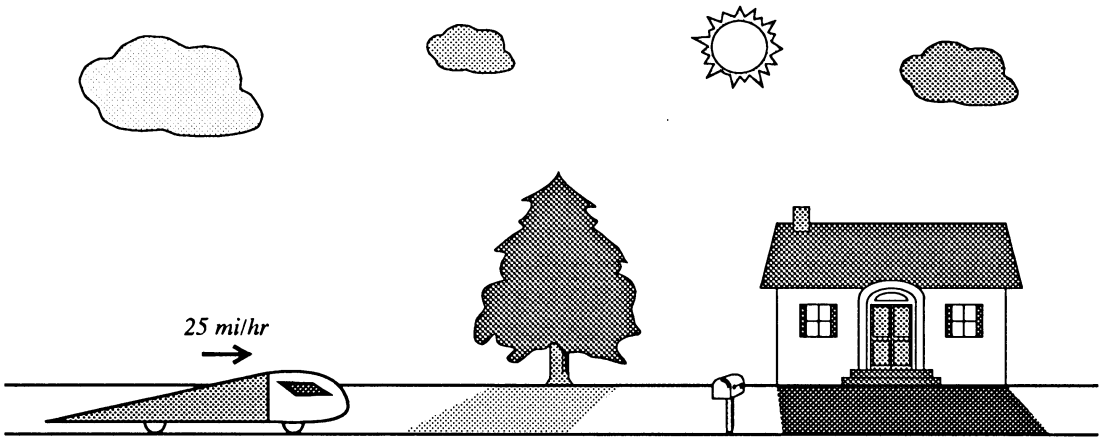

Figure 3.2: Earthbound moving solar panel

This is an entirely different case from the stationary panel, which can be seen by analysis of two types of shadows encountered by the panel: sharp shadows cast from buildings and opaque clouds, and scattered shadows cast from leafy trees, power lines, and the like. Looking at the time for a panel to fully enter a sharp shadow and the time to cross a scattered shadow shows quite drastically the differences between moving and stationary arrays.

First, find the time required for a six foot long array on a car moving at 25 mi/hr to enter a sharp shadow cast from a building:

$$\text{time} = \frac{6\text{ft}}{36.66\frac{\text{ft}}{\text{sec}}} = 0.1636\text{sec}$$

In much less than a second the shadow has fully overtaken the panel. By using a tracker which adjusts the power point only once per second, the panel will be well away from maximum power while in the shadow. If the array is designed poorly or steps are not taken to prevent it, the panel could actually begin to *consume* power rather than produce it.

Next, find the time for a six foot long array on a car moving at 25 mi/hr to pass through a fifteen foot long scattered shadow:

$$\text{time} = \frac{6\text{ft} + 15\text{ft}}{36.66\frac{\text{ft}}{\text{sec}}} = 0.5727\text{sec}$$

In this case the array moves into and out of the scattered shadow in a little over half a second. This is also analogous to the shadow from a light pole, etc., crossing the panel in the same amount of time. Here the effect of the shadow may be minor or major, depending on the amount of scatter placed on the panel. Also, since cells are almost continually moving into and out of sunlight, "maximum" power tracking is really not possible. Simply waiting out the scatter and quickly returning to maximum power in full sun is the desired action of a tracker under this shadow type.

Situations also arise where the panel will move into and out of many shadow types in the course of a minute. All of these shadow combinations point out the differences in attempting to power track between a stationary panel and a ground-moving panel. Using a tracker with a slow adjustment rate (as was the case for the control code used on PrISUm during Sunrayce) will not provide effective maximum power tracking for such quickly changing sunlight conditions. A power tracker must be able to respond within a fifth of a second when a drastic change in sunlight occurs, yet remain stable for the duration of a scattered shadow and quickly return to maximum power in stable sunlight.

Noise Environment of a Solar Powered Vehicle

A solar vehicle exhibits just about every difficult noise problem a designer of an electrical circuit (in this case a power tracker) can encounter. These noise types are:

- Magnetic and transient inductive noise
- Radio-frequency interference (RFI)
- Ground loops and ground noise

Each of these noise sources is itself difficult to design immunity to; when all of these noise sources are present at the same time the task becomes almost impossible.

The magnetic noise present is actually due to the power tracker itself: a large inductor is necessary to handle the amount of power flowing in a switched-mode DC-DC converter. The pulsating current during normal operation of the converter creates a varying magnetic field around the inductor. When multiple trackers are placed in close proximity while operating, a wideband varying magnetic field is produced which can cause induced voltages at many points in a circuit, depending upon location and wiring. Related to this but not the same is the large inductive noise output by the phase leads between the motor and controller. To obtain the best possible power efficiency from a DC motor a brushless, permanent-magnet motor is used with a switched motor controller that drives current through each of the phase leads to turn the rotor. Under heavy load (such as heading up a hill) the motor may need as much as forty amps to provide the necessary torque to keep the rotor spinning. Switching forty amps through large wires yields a sizable induced voltage problem for practically all electronic circuitry in the car, regardless of location.

Radio-frequency interference is caused by the presence of two radios in a typical solar race car arrangement: one radio for telemetry and one for the driver to communicate with the race team. For moderate ranges of errorless reception a watt or more of transmitter power is necessary. Broadcasting a signal of high frequency (megahertz range) at such a power level makes every wire in the car a miniature antenna which picks up a small induced voltage. Most current sensing in the car is done by means of very small (0.5 ohm or less) resistors, the voltage across each being amplified to a larger level for an A/D converter to sample. Since even a small (less than 0.1 volt) rise in voltage is quite substantial to this pre-amplified signal, erroneous readings and possibly erroneous reactions of any control system not properly immunized to such noise can occur. This happened in the power trackers of PrISUm during the race -- keying the transmit button of the driver radio caused the system to falsely sense the power from the arrays and cut the power produced by the arrays a considerable amount.

The last, and potentially most deadly "noise" source present in a solar powered vehicle

is a phenomenon termed a ground loop. This occurs whenever fairly high amounts of current are flowing in wires whose resistance is non-negligible. The result is that ground between different modules of a system may not be at the same electrical potential. If these modules are interconnected (for example, a computer sampling a voltage of another module) a "loop" is formed and excess current will flow back through the connection, possibly causing inductive type noise due to ground loop currents. Related to this is the problem of ground noise: with large currents flowing through wires from switched-mode power sources, repeating surges of current cause a voltage fluctuation on the ground lead. If another module is susceptible to this voltage change, erroneous readings could occur, and once again bad control actions may result.

These three noise problems are manageable individually, but become quite a problem when all occur simultaneously. Techniques do exist, however, to reduce the effect these noise sources have on a circuit [1]. Obviously since accurate telemetry data and immunity to noise are necessary for the power tracking system of a solar race car, the effects of all of these noise sources must be minimized.

Goals for Effective Power Tracking in a Moving Vehicle

From the analysis presented in the previous two sections, several goals for an optimum fast response power tracker for a solar powered vehicle have been derived. They are listed as follows:

- Relative immunity to all noise sources
- Accurate power data to be used for race strategy
- Quick response to sharp solar changes
- Tolerance of moderate shadow conditions
- Good system reliability

The last goal, system reliability, stems from the desire to keep the car moving at all times during a race with solar power continuously available to assist and/or charge the battery.

CHAPTER 4. DESIGN OF A FAST RESPONSE POWER TRACKER

Necessary Elements

A solar power tracking unit is primarily a switched-mode DC to DC converter used to convert one DC voltage to another in as efficient a way as possible. Three types of converters are possible: boost, buck, and buck/boost. A boost converter takes an input voltage and current and steps this up to a higher voltage and smaller current at its output. A buck converter does the reverse, and a buck/boost converter can perform both functions, but at a cost of higher complexity and an inverse ground problem when the converter switches from boost to buck and vice versa.

The type of switched-mode converter used depends on the array and battery arrangement chosen for a particular solar application. For the arrays on PrISUm the maximum power output voltage level is less than 80 volts. Thus, since the battery voltage was chosen to be 100 volts in order to increase the efficiency of the motor, a boost converter was used to power track the solar panels on PrISUm.

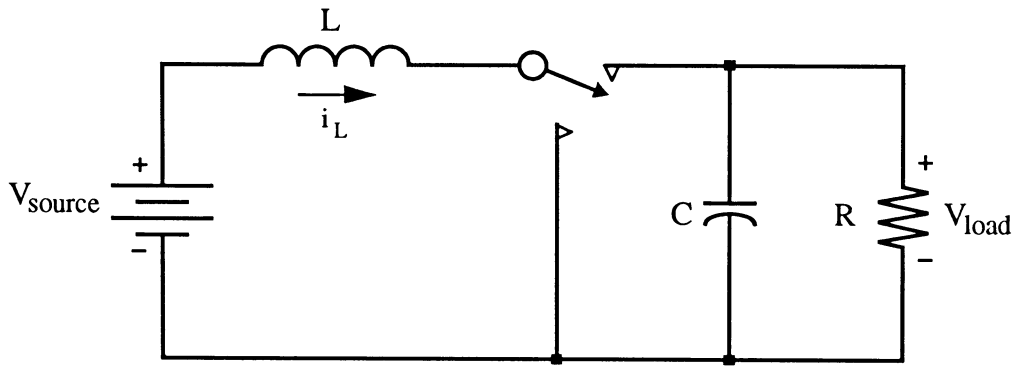

Figure 4.1: Idealized boost converter

A more detailed description of a boost converter is given in Appendix A. In brief, a boost converter consists of an energy storage device (in this case an inductor) and a means to

selectively switch the inductor between charging from the power source to delivering charge to the load. A schematic of an idealized boost converter is shown in Figure 4.1. By varying the duty cycle of the pulse-width modulated signal used to drive the switch, the amount of boost can be adjusted [2]. When one side of the boost converter is connected to a fixed voltage source (such as a battery), the other side of the converter can effectively be "forced" to a voltage by choosing an appropriate duty cycle (dependent on the characteristics of the voltage source on that side). In the case of PrISUm, the load side of each converter is connected to the system bus, which has both the battery and motor controller connected in parallel. With the other side connected to a solar array, varying the duty cycle varies the voltage forced upon the array and hence the voltage across each solar cell. Since a solar cell has a very distinct voltage/current transfer function which varies with voltage, sunlight, and temperature, varying the voltage across each cell varies the amount of current (and therefore, power) delivered by that cell (see Figure 4.2). This is the means used to track the maximum amount of power each cell can deliver [3]. For more information on solar cells, see Appendix B.

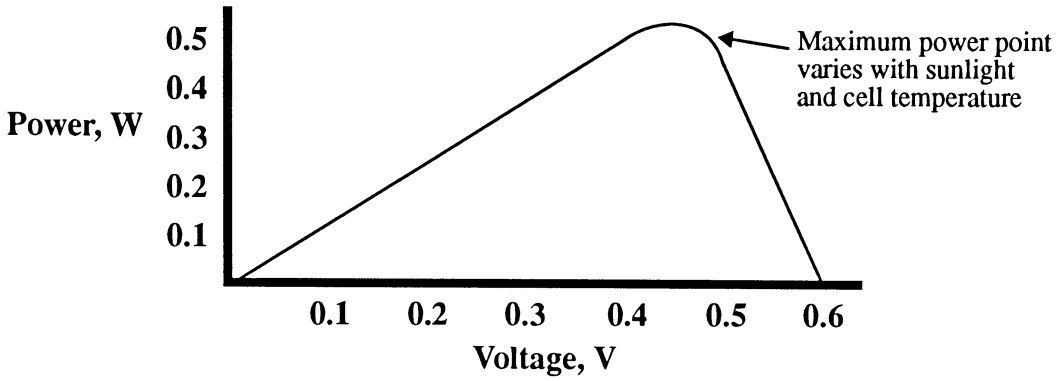

Figure 4.2: Power versus voltage of a silicon solar cell

In addition to the boost converter, some means of producing a pulse-width modulated signal at a given frequency is necessary. The frequency used is dictated by the values of the converter components, the voltages to be used, and the desire to run the converter at maximum

efficiency. The frequency is chosen such that the current through the inductor never drops to zero (referred to as *continuous mode*). For a bus voltage of 100 volts, maximum array voltage of roughly 80 volts, and an inductor of 10 millihenries the best switching frequency to use for the converter is about 25 kHz.

The last element necessary for the maximum power tracking function of this DC-DC converter is a control element. This element must be able to sense the voltage the array is operating at, the current being delivered by the array at that voltage, and the voltage of the load (in this case the motor/battery combination load). Some feedback is implied in a controller able to compare old power with that currently being delivered. The controller should be designed to detect when the load voltage drops below the source voltage and shut off the converter. Lastly, the controller should also shut off the converter if the load voltage increases rapidly (indicating the load has been disconnected and the capacitor across the output of the converter is charging and might blow). A conceptual diagram of a general DC-DC boost converter system capable of power tracking a solar array to a battery load is shown in Figure 4.3.

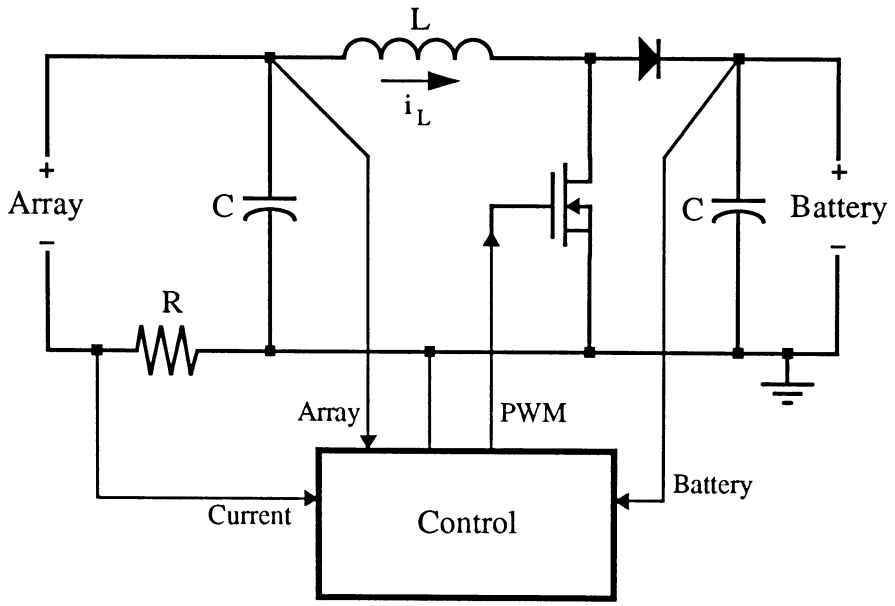

Figure 4.3: Conceptual DC-DC boost converter system

This conceptual boost converter does not include any function to broadcast power information by some means to another device.

Centralized versus Distributed Control

For a system with multiple solar panels, a single battery, and an individual tracker for each panel, two types of controllers are possible. The first is the centralized controller used previously in PrISUm, which places control of all five power trackers in one device. The second controller configuration places a separate controller on each tracker which operates independently of all other trackers. This method of control was not chosen for the original PrISUm electrical system due to the design complexity foreseen; however, this approach increases the overall reliability of the system since all subsystems are not dependent on a single control authority. The individually controlled module scheme is the one adopted for the redesign of the power trackers presented here.

Using this scheme for all of the other modules necessary for the operation of a solar car (battery monitor, motor controller, light/horn controller, telemetry, etc.) yields a scheme where each module is operationally independent of all others. The exception to this is the telemetry system, which must communicate with all other modules to collect data for transmission to the strategy computer. Software requirements for each module decrease drastically in complexity since the "shared resource problem" exhibited with the centralized control system no longer exists. Each module, if properly designed and implemented, has all it needs to perform its control and monitoring functions. It is also much easier to determine if a unit has failed, and enough complexity can be added into each module's control software to detect sub-module failures and notify the telemetry system of this. This capability is difficult to include in a centralized approach.

The only notable disadvantage of the distributed control scheme is that each unit becomes more complex and more difficult to design, since all of the noise problems described

previously must be dealt with in each unit and in communication between units. Also, in order to isolate each system (preventing ground loops), power distribution to each unit must be reduced or eliminated, resulting in the requirement that each unit be self-powering or powered from the battery. Less efficient, but smaller, DC-DC converters or voltage regulators are necessary in each unit to provide whatever voltages the circuitry requires. With these there will likely be an increase in power consumption of the overall system. Considering the increase in system intelligence, performance, and reliability, this may be a desirable trade-off. Figure 4.4 shows a representation of the entire distributed control system.

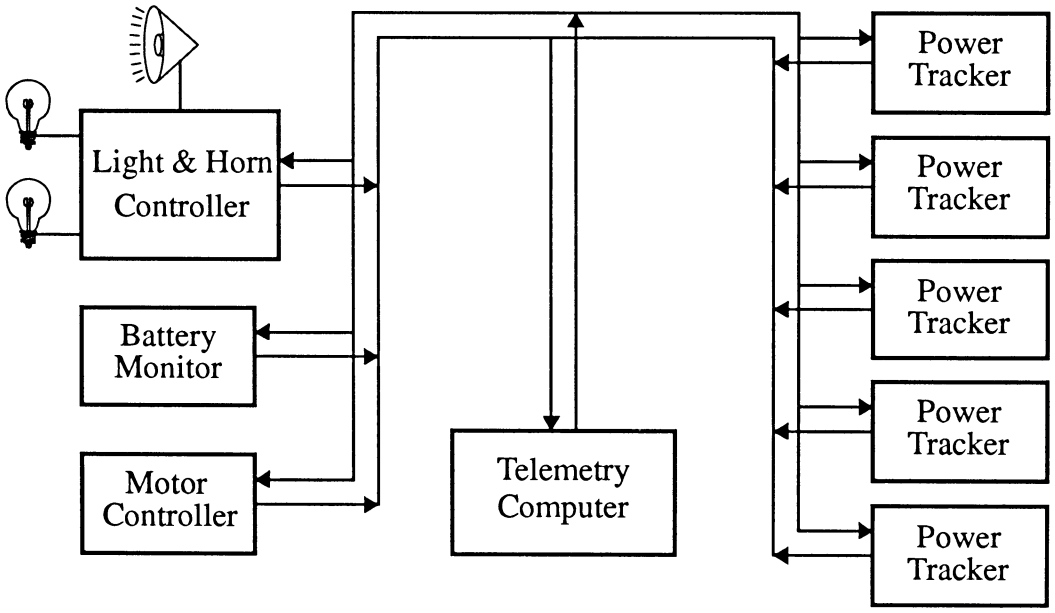

Figure 4.4: Distributed control system for PrISUm

With the overall system now conceptually designed, the final design criteria necessary for each individual power tracker is available. Each tracker should be self-powering (however the possibility of using a more efficient power supply source should be considered) and operate independently of all other systems, including the telemetry system. The tracker should also be ready to deliver the amount of power produced by its panel to the telemetry computer when

requested. With these specifications an actual control system can now be designed.

Analysis of Existing System for Redesign

The existing power tracker units for PrISUm contained three parts: the boost converter, an interface board with a Signetics NE5562 chip to perform the PWM feedback control, and the centralized control computer. Of these three parts, only the boost converter is potentially usable under the new control scheme -- most of slow response time in the old system was attributed to the NE5562 chip. Some thought was put into changing from a boost to a buck/ boost converter to handle the possibility of the battery voltage dropping below the array voltage (this occurred several times during Sunrayce). Ultimately, due to lack of time and the fact that the present boost converters functioned without a breakdown during the race, the decision was made to use the existing boost converter without modification. The only problem foreseen was the weight and construction of the inductor used: this inductor amounts to a small transformer with quite a bit of mass, capable of easily cracking a printed-circuit board under considerable vibration. A future enhancement to the system might be that of finding a less massive inductor capable of providing the same efficiency.

By removing the other two parts of the old tracker, their function must be replaced (and hopefully enhanced) in the new tracker. The new control unit must sample the array power and produce a PWM signal for the converter switch. This control function could be implemented using either analog or digital circuitry. While analog circuitry might yield a less complex solution, it was very doubtful the response performance necessary could be accomplished using such a circuit. Also, since at some point data must be collected and delivered to the telemetry computer, a digital solution is better suited for the task.

In order to keep the power consumption as low as possible, use of a microcontroller for the signal sampling and PWM control functions seemed the best choice. The alternative was to use a pre-made controller board, but the power consumption and packaging of such a board

would have proved too costly to be feasible in a solar powered vehicle.

From all of the specifications derived previously, the basic requirements for a microcontroller are as follows:

- Analog to digital converter with at least three channels
- Pulse-width modulated signal output capable of operation at 25 kHz
- Serial port for communication with telemetry computer
- On-chip RAM and EPROM for low power consumption
- Good noise immunity built into the chip

Also necessary for the control unit are filters to keep all other noise out of the sampled values (array voltage, array current, and battery voltage), and internal generation of all supply voltages necessary to operate the control unit. Care must also be taken to prevent ground loops in the design of the controller and the interface to the telemetry computer.

CHAPTER 5. IMPLEMENTATION OF A FAST RESPONSE POWER TRACKER

Microcontroller Unit

In order to choose a specific manufacturer's microcontroller to use, some additional criteria must be established. The number of bits of resolution in the analog to digital converter, operating speed of the chip, amount of RAM and EPROM available, hardware units necessary on-chip to implement a control algorithm, and additional items necessary for other modules in the PrISUm control system must all be specified. The last requirement stems from a desire to standardize all of the control modules in the new PrISUm system to aid development and support of the entire package. With one type of microcontroller used in each module wherever needed, the same software development package and hardware interfacing techniques can be used throughout the system.

To determine the resolution required for the microcontroller's A/D converter, some knowledge about the signals to be sampled and the desired accuracy is needed. For the power tracker three signals are sampled: array voltage, array current (across a small shunt resistor), and battery voltage. Since the sampled voltages cannot be higher than five volts for any microcontroller-based A/D converter, voltage dividers for the battery and array voltages will be needed. The best resistors available are 1% precision metal-film resistors, and these will be used for the voltage dividers. The current sensing voltage will need to be amplified rather than divided, and again the same precision of components will be used. Thus, the best accuracy possible using a "perfect" A/D converter is one percent. The least significant bit of an eight-bit A/D converter (with full scale of five volts) is:

$$\text{LSB} = \frac{5\text{V}}{2^8} = 0.0195\text{volts}$$

which is less than one percent of five volts. At first glance eight bits would appear to be sufficient for sampling; however, using such a converter would require throwing out the least

significant bit before computing values based upon this data. This truncation adds to the 1% error already present in the voltage dividers. In order to increase the accuracy of the A/D converter (and shrink the error of the overall sampling system), adding additional bits of resolution is necessary. Using a ten-bit A/D converter yields a least significant bit of:

$$\text{LSB} = \frac{5\text{V}}{2^{10}} = 0.0049\text{volts}$$

Taking all ten bits of resolution to sample a signal will obviously increase the accuracy of the A/D converter, but some bits must still be ignored when calculations are performed using this data. However, by rounding from the ninth bit and using the upper eight bits, the voltage data obtained by sampling will be at least one more bit accurate than the eight-bit converter, if not more accurate since additional comparisons are made to determine bits nine and ten. The end result of this is that the sampled values should be accurate close to 1% of their true values.

Determining the operating speed of the microcontroller is a bit more of a subjective decision, since only some knowledge of the final control algorithm is known before the unit is built. Determining whether to use an eight or sixteen-bit microcontroller is fairly easy, however: in order to compute power from the array, two eight-bit values must be multiplied together, yielding a sixteen bit value. Because several previous power values will need to be stored to implement a feedback control algorithm, it is desirable to fetch these values from memory in one access. This allows the algorithm to run more quickly in an interrupt service routine. Also, since it is never really wrong to have too *much* processor power (keep in mind here that the same microcontroller will be used for many different applications in the car), choosing a sixteen-bit microcontroller is a good decision. Most sixteen-bit chips run from eight to sixteen megahertz, easily sufficient speed for a slow control algorithm. As the response time (and hence, complexity) of the control algorithm is increased to improve the reaction to sharp input changes, this extra processor speed may prove quite beneficial.

Along the lines of processor speed is the amount of RAM and EPROM available on-chip.

Off-chip memory could be used, but this would increase the power consumption of the system since current would constantly be shuttled around on the system bus to perform memory accesses. Keeping memory demands on-chip is an effective means to increase the reliability of the unit (fewer printed circuit board traces capable of breaking) and lower power consumption. Determining the amount of memory needed for an application before actually writing it is at best black magic. Here again, "you can never have enough of a good thing" is an appropriate adage.

A helpful unit for control algorithm implementation is a fairly flexible software timer, useful for creating interrupts on a fixed-time basis to perform various control functions (such as starting an A/D conversion or adjusting the pulse width modulated output). There are other hardware items which can helpful, but not necessary, for a tracker; their inclusion may be of great help to designers of the other PrISUm control modules.

With all of this knowledge, time was spent poring over data books and finally deciding on the Intel 87C196KC microcontroller. A short description of this chip is available in Appendix C. In addition to eight analog channels of ten-bit resolution, the chip features 16K bytes of EPROM and 512 bytes of RAM, three pulse-width modulated outputs, several high-speed inputs and outputs, and a very flexible timer unit. While not the "perfect" microcontroller for this application, it has broad applicability and is designed for harsh noise environments.

Interface Circuitry

Three analog inputs, one analog output (the PWM signal), and the serial port are to be used on this microcontroller chip. In addition, since all memory accesses will be internal, the address/data lines are available for use as a parallel port. Connecting DIP switches to these lines and using the switches to set software values (operating modes, threshold values, etc.) will provide additional flexibility in the unit, allowing one program to act in many different ways. In this way the tracker can perform better for different operating environments.

The easiest interface circuitry to design is the voltage dividers for the battery and array voltages. Since these are mostly DC signals with some high-frequency noise added, low-pass filters with very low (less than 500 hertz) cutoff frequencies should eliminate most of the noise. In order to allow room for various array and battery sizes, full-scale voltage is set higher than the actual expected voltages on both voltage dividers. The circuitry is shown in Figure 5.1.

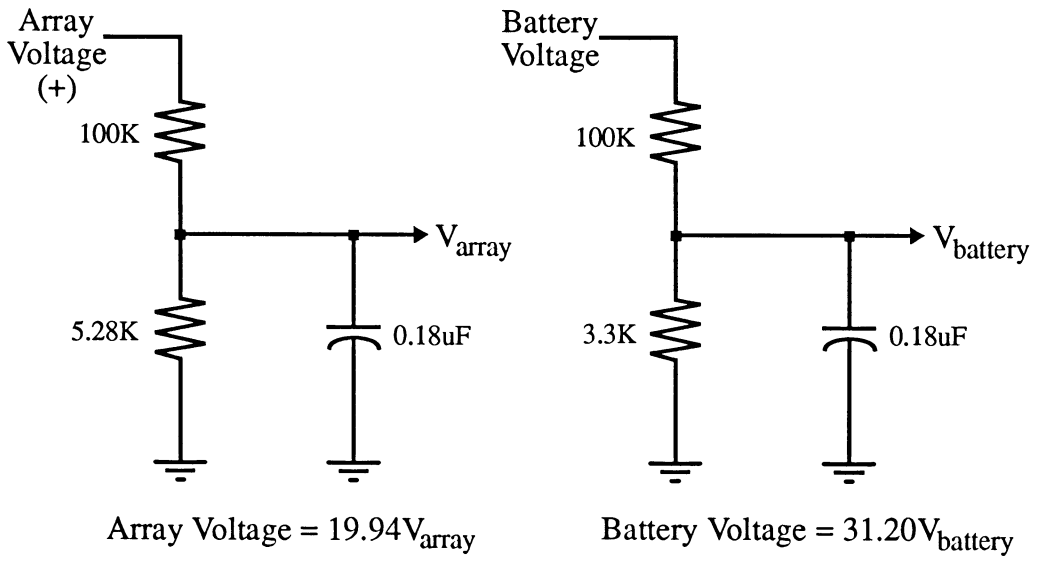

Figure 5.1: Array and battery voltage dividers

One more input signal must be sampled: the voltage across the small shunt resistor placed in the main current path of the boost converter. Since this resistor must be kept very small to hold down the power it consumes, this voltage will be small and needs amplification to a five volt full scale value. Complicating this somewhat is the fact that the boost converter switches current through this resistor and so the voltage will be pulsating, yet what is really desired is the *average* current. In addition, since this is a small voltage, any noise induced voltages will affect this signal the most, lowering the signal-to-noise ratio a significant amount. As such, great care must be taken to eliminate differential noise through any amplification used, as well as in low-pass filtering the signal to produce a DC average of the current flowing through the

resistor. Designing this portion of the tracker provided an answer to one of the problems discovered in the old tracker: the inconsistency between the computer power readings and an analog power meter was mostly due to a problem in the design of the old current sensing circuit. The new sensing circuit is shown in Figure 5.2.

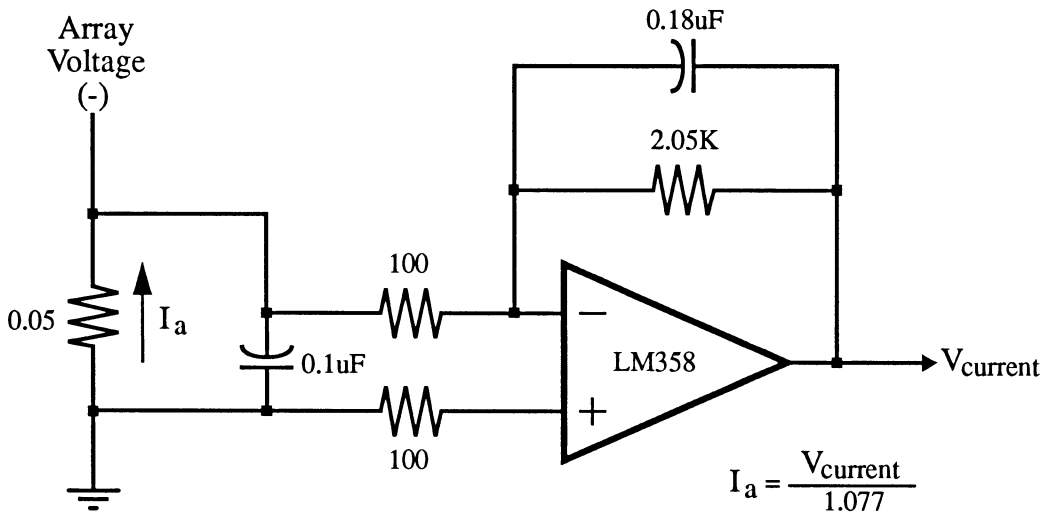

Figure 5.2: Current shunt amplification circuit

Since the 87C196KC microcontroller is a CMOS part, it cannot source or sink enough current to effectively drive the gate of the Motorola MTH30N20 power transistor used to switch the boost converter. In addition, the MTH30N20 saturates around seven volts, so the zero to five volt output of the 87C196KC PWM signal cannot drive such a MOSFET to saturation. Further, two such MOSFETs must be used in the boost converter to keep the heat dissipation of the transistors at a moderate level. As a result, a circuit to repower the PWM drive signal coming from the microcontroller and step the voltage up above seven volts is necessary. In order to maximize the efficiency of the converter, the gates of the MOSFETs should be driven as hard as possible to produce very quick turn-on and turn-off times. The circuitry to drive the MTH30N20 gates from the microcontroller PWM output is shown in Figure 5.3. A

smaller MOSFET which saturates at three volts is used to drive the power MOSFETs. Note that this produces an inversion on the PWM signal. This circuitry was simulated along with the boost converter using SPICE to verify the microcontroller output could operate the converter.

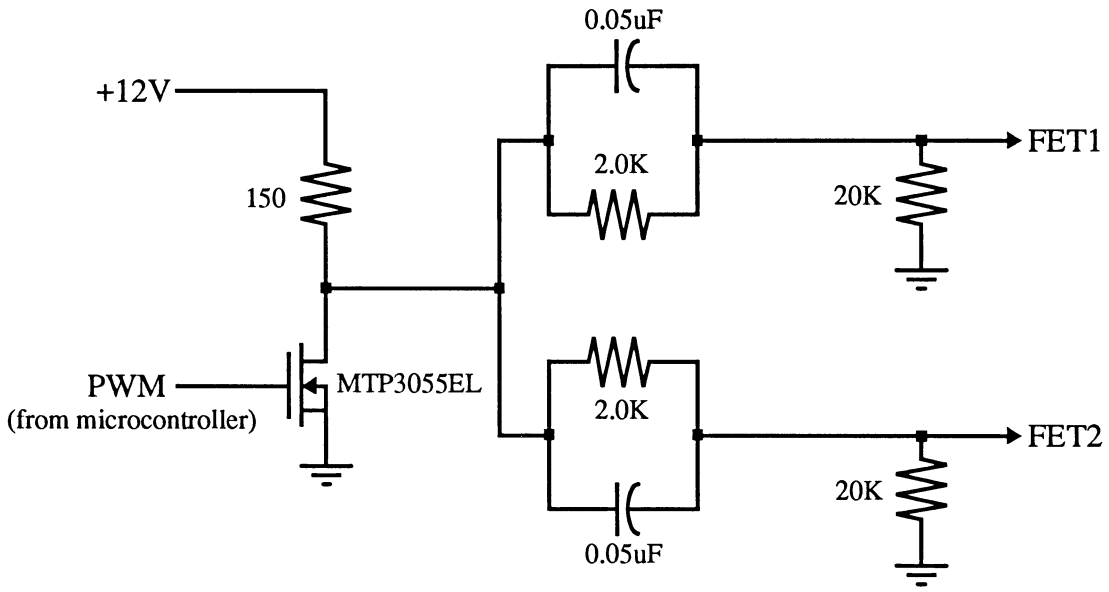

Figure 5.3: PWM repowering circuitry

This takes care of all the analog input and output signals needed for the power tracker, leaving only the interface to the telemetry computer. A minimum RS-232 serial connection requires three wires: transmit data, receive data, and ground. Connecting the ground of any power tracker to the telemetry computer's ground will create an undesired ground loop, especially since the ground of the tracker will vary with the amount of power the panel is generating (remember the resistance of the wire from each tracker to battery ground). Further, connecting all of the control system units to the telemetry computer in this manner will tie the ground of all these units together. The chance of ground for all units being at exactly the same potential in such a high-current system is nonexistent. The serial connection would cause the entire system to destroy itself.

What is needed, therefore, is a means to provide serial communication between each unit

and the telemetry computer without directly connecting the units. Provided the speed of communication is relatively slow (4800 baud or less), optical isolation can be used to perform this task. For this to work, the following conditions must be met: the telemetry computer should have a sufficient pullup on its receive line, the driver of the telemetry computer's transmit line must be strong enough to pull all of the optoisolator inputs on the line quickly enough, and the telemetry computer ground must be made available for each unit to pull the receive line low when needed. The circuitry necessary on the tracker side, along with most of what is needed for the telemetry computer's side, is shown in Figure 5.4.

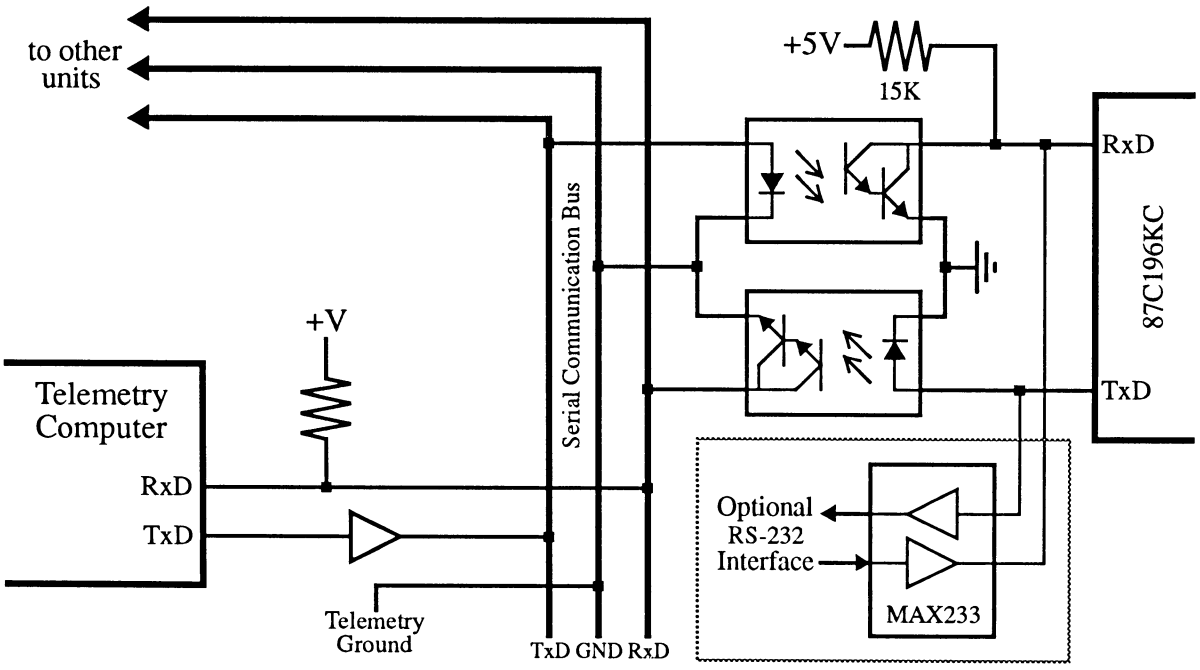

Figure 5.4: Optically isolated serial connection

Also shown in the serial connection is a Maxim chip to provide a conventional RS-232 interface. This component requires very little external circuitry, generating its own voltages to operate its RS-232 drivers. Including this chip in the final design allows the tracker to be used in other applications where it can be interfaced into a personal computer to transmit power data for recording.

Power Generation

One of the most difficult circuit design problems can be that of generating the different voltages necessary for a unit to work. With unlimited power or when efficiency is not a prime consideration, this will generally not be a difficult task. But in the case of a solar car, every watt wasted in conversion efficiency or dissipated in a resistor is a watt that could have been put to use charging the battery or powering the motor. This leads to a strong desire to use ultra-efficient DC-DC converters (buck or boost, depending on the application) whenever possible. However, such converters have limits, and it is unlikely one could find a commercial converter designed to generate five volts at fifty milliamps from a one hundred volt source. Also, since conversion efficiency is related to output current load, such light current demand puts a converter at the low end of its efficiency range.

A solution to this is to use one power supply for many units, thus increasing the load on the supply. However, this immediately leads to a ground loop problem similar to that encountered for the serial connection. Ground loops in a power supply can be tolerated by certain units, however. It might be possible to stack five power trackers next to each other and place one twelve volt supply on the end, powering all of the units. While this possibility should be provided for, in other situations a self-powering unit may be desirable to allow the tracker to run in the field without being constantly monitored.

Two voltages are needed for the tracker to operate: a five volt supply for the microcontroller, and a supply higher than seven volts to drive the gates of the power MOSFETs. This second supply was chosen to be twelve volts. Since the current demand on the five volt supply is rather small (less than 100 milliamps), a series voltage regulator can be used to produce this voltage from the twelve volt supply without significant power loss. Another small DC-DC converter might have been used to provide the twelve volt supply, but with the addition of a control system for this converter it is about as efficient to simply derive this voltage from

the array with a twelve volt zener diode. This consumes about a watt of power from the array but is a simple way to make the unit self-powering. In situations where power consumption is critical, a separate DC-DC converter can be used and the zener diode removed from the circuit board. The power supply generation circuitry is shown in Figure 5.5.

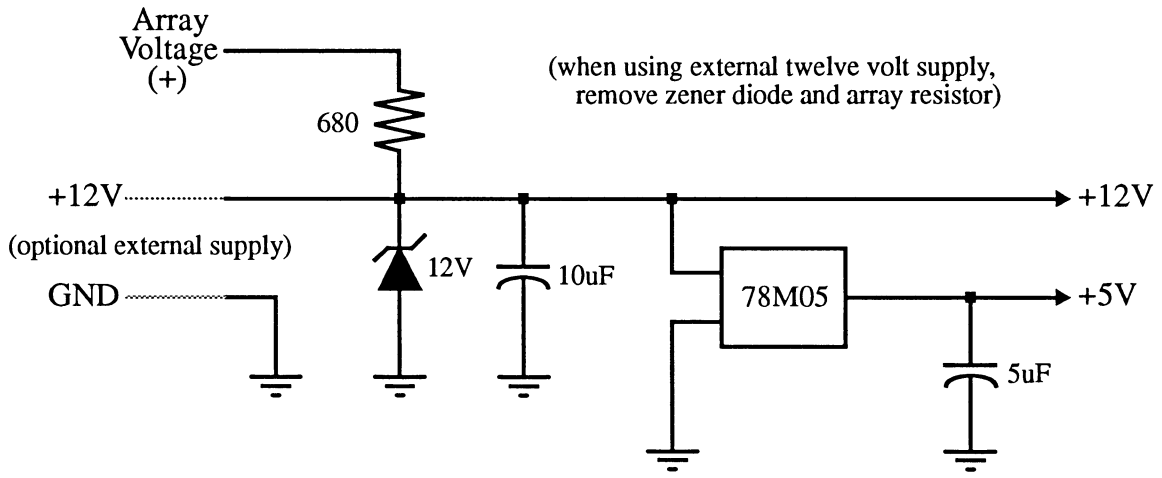

Figure 5.5: Power supply for power tracker

With the noise environment and switching characteristics of the componentry used, care must be taken to place decoupling capacitors close to critical devices and on the supply lines, as shown in Figure 5.5. While not the most efficient power supply that could be designed and built, the simplicity of the components used should help the reliability of the overall unit. A more complex solution would require quite a few more components to implement.

Miscellaneous Circuitry

Additional circuitry is necessary which provides input signals to start the microcontroller and keep it running. Foremost of these is a clock source for the chip. This clock is provided using a crystal; the value can be changed within the limits of the 87C196KC to change the operating speed of the microcontroller. Another input necessary is a reset signal to the chip during power-up. The simplest circuit for this is a capacitor connected to ground, since the

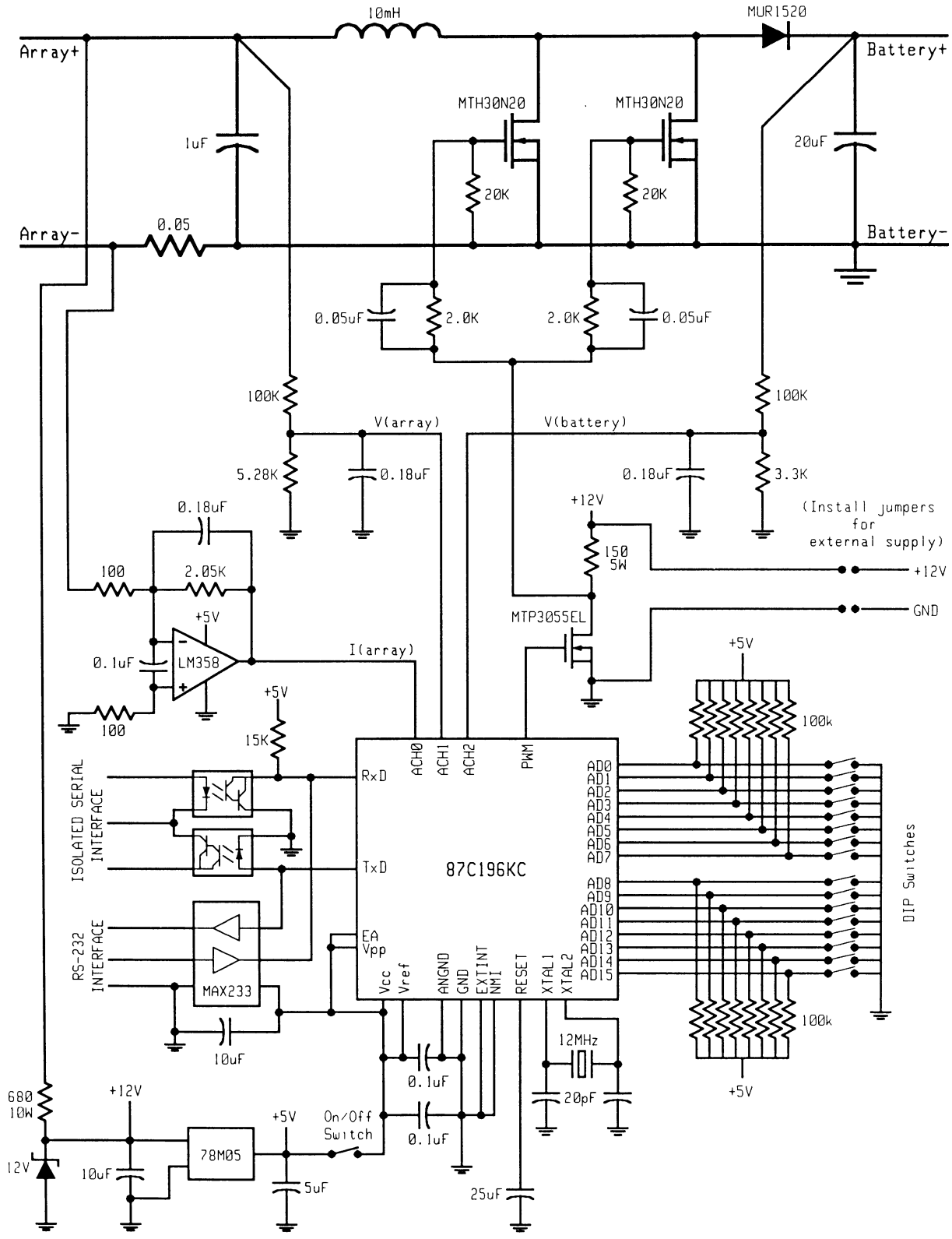

Figure 5.6: Schematic of new power tracker

87C196KC has an internal pullup resistor on its reset line. Decoupling of power connections is also critical for this microcontroller for it to function properly. A schematic of the entire power tracking system, with all of this miscellaneous circuitry, is given in Figure 5.6.

Circuit Modifications

As can be seen from the interface circuitry presented, changing the basic control system to fit other array and battery configurations is not that difficult (provided the battery voltage is higher than the maximum array voltage). In theory, the same control software could be used; all that is required is to change the resistor values in the voltage dividers and the gain of the current sense amplifier. The inductor value of the boost converter may need to be changed and/or power MOSFETs and capacitors with different maximum voltages used, but the basic control scheme remains the same. Also, since the response of the system can be changed more by software than hardware, one tracker can be made to function in many different ways simply by modifying the control algorithm. With the addition of some DIP switches to change software options, these modifications can be accomplished without writing new software.

Control Software Theory

At this point a designer must now forget about the hardware and think only about how the system should respond and how to program that response. Under most circumstances the tracker should only lightly adjust the duty cycle of the PWM output a few times per second. In this way deviation from maximum power will be minor and brief in occurrence. There will be times, however, when a very fast response is necessary to bring the boost converter/array combination to a new maximum power point, and this transition should be as brief as possible.

The simplest approach is to sample the array voltage and current, compute the power, compare this to an old power, and adjust the duty cycle of the PWM output up or down by one (out of a maximum of 255). A simple example of this code is given in Listing 5.7.

```
direction = 1;
pwm_duty = 128;
while (true)
  {
    voltage = <sampled array voltage>;
    current = <sampled array current>;
    power = voltage * current;
    if (power < power_old)
      direction = direction * -1;
    pwm_duty = pwm_duty + direction;
    power_old = power;
    <delay>
  }
```

Listing 5.7: Pseudo-code of simple control algorithm

Performing such an iteration several times a second will track to maximum power, with the only penalty being that overall response time to get to a new power point (induced by a shadow, etc.) will be several seconds, if not longer. This algorithm is inherently stable, however, since it adjusts so slowly the gain of the system is small and instability is very difficult to induce. This type of code would work exceptionally well for a stationary array (or an array on a car at rest), but cannot even begin to effectively track the 0.16 second changes that occur in a ground-moving array as shown in Chapter 3.

At the other end of the spectrum is a control algorithm with a very high gain. This can be implemented with an exponentially adjusting algorithm: as the power deviates from maximum, the size that the PWM output is changed increases by a power of two until maximum power is crossed. The algorithm then changes the direction of adjustment and shrinks the size of PWM adjustment until maximum power is regained. A simple example of this algorithm (much more complex examples abound) is given in Listing 5.8. In principle this algorithm should work well; in reality it yields disastrous results. A control system implemented with this unbounded algorithm is almost impossible to keep stable.

```
adjust = 1;
mode = 1;
pwm_duty = 128;
while (true)
  {
    voltage = <sampled array voltage>;
    current = <sampled array current>;
    power[2] = power[1];
    power[1] = power[0];
    power[0] = voltage * current;
    if ((power[0] < power[1]) && (power[1] > power[2]))
      mode = 0;
    if (mode == 1)
      adjust = adjust * 2;
    else
      adjust = adjust / 2;
    if (abs(adjust) == 1)
      mode = 1;
    if (power[0] < power[1])
      adjust = adjust * -1;
    pwm_duty = pwm_duty + adjust;
    <delay>
  }
```

Listing 5.8: Pseudo-code of control algorithm with high gain

Other control algorithm variants are possible, but a quick look at the requirements for the system yields the insight that an effective system should be very stable under slowly changing input conditions, yet have a very high gain when a sharp change in sunlight is detected. This can be accomplished by utilizing a variant of the simple control algorithm whenever possible, but if power decreases or increases rapidly, a modification of the exponential algorithm is used to find the new maximum power point. Control is then returned to the slow algorithm. When the tracker is operating under the simple algorithm it is termed in "slow hunt" mode, and when the exponential algorithm is in control the tracker is in "fast hunt" mode. Since the fast hunt mode must somehow be kept stable it is never allowed to change the duty cycle of the PWM output by more than a set amount. This effectively bounds the gain of that mode. Provided the fast hunt mode finds a new maximum power point and the threshold which switches between the two modes is properly set, this control philosophy should provide quick response to sharp

shadows while "waiting out" the scatter produced by trees and the like.

The way that slow and fast hunt modes can achieve their results is by the number of times per second each is allowed to adjust the duty cycle of the PWM output. The initial guess used was that slow hunt adjusts five times per second, and fast hunt adjusts twenty-five times per second. To keep the boost converter in continuous mode the duty cycle should never fall below 30% or exceed 80%. With this constraint, fast hunt mode can sweep the entire remaining range in less than a second. Fast hunt mode should therefore be able to at least move through the maximum power point quickly; hopefully, it should find maximum power without a full sweep and return control to the slow hunt algorithm.

In addition to the basic functions of operating the boost converter, the control software must also check that the battery voltage does not drop below the array voltage, in which case the boost converter cannot work (and is blocked from draining the battery by a power diode). Also, the battery voltage must be monitored to insure it does not begin to rise sharply. This indicates the possibility a battery lead has been disconnected and the controller must immediately shut off the boost converter, preventing the capacitor placed across the converter output from becoming overcharged.

A listing of the control software used on the 87C196KC microcontroller is found in Appendix D. This software also includes routines to report power information the telemetry computer by means of the microcontroller's serial port.

CHAPTER 6. TESTING AND ANALYSIS

Prototype Construction

Based upon the design presented in the previous chapter, a prototype of the new power tracker was built. Since high-speed digital (and analog) signals are sometimes difficult to manage in a wire-wrapped prototype, additional filtering and component decoupling was added to insure noise in the circuit would be kept as low as possible. Once the final circuit is laid out on a ground plane by means of a printed-circuit board, this additional filtering should not be necessary. Extra component decoupling, on the other hand, may still be necessary, depending on system performance. Microcontroller chips tend to be sensitive to decoupling problems in order to be relatively immune to noise.

Test Setup

To test the new power tracker, a test setup was created which placed new and old power trackers next to each other, allowing comparisons to be made. All of the initial bench testing was conducted using a stationary array of six panels mounted on the roof of Coover Hall. Once the tracker had been sufficiently debugged, an actual PrISUm panel was used to bench test against a more dense, powerful, and smaller array. A pictorial of the bench test setup is shown in Figure 6.1. Since an old tracker was being used, some means of generating the several different voltages it required was necessary. In order to keep the battery from being connected to earth ground, the power supply card from the old PrISUm electronics system was wired into the circuit. This card uses several commercially-manufactured DC-DC converters to generate positive twelve, positive fifteen, negative fifteen, and positive five volts from the battery voltage.

The new power tracker circuitry was constructed and tested in stages, with the analog interface circuitry built and tested first. During this testing was the discovery of a problem in

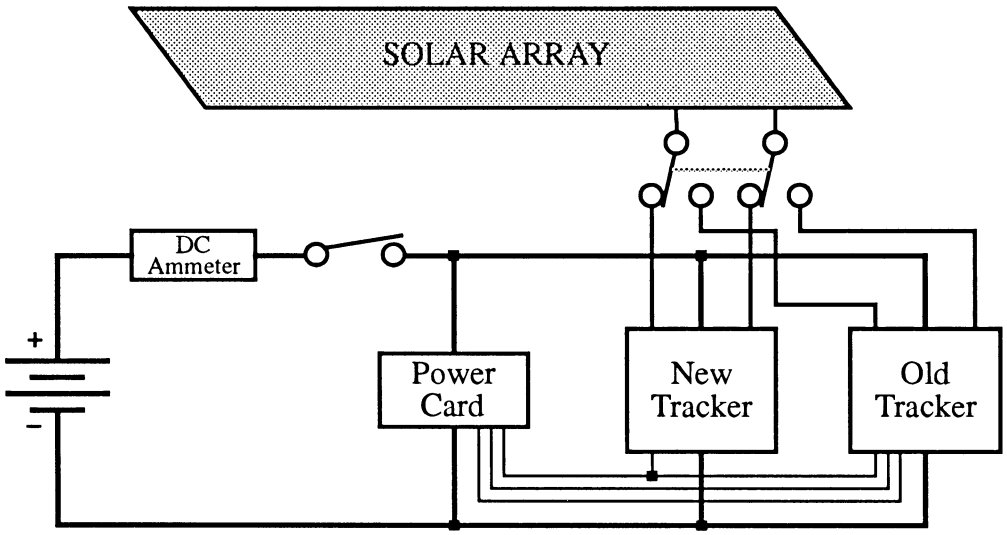

Figure 6.1: Bench test setup for power trackers

the old current sense circuitry. This led to a minor redesign of the new current sense circuitry, as presented in Figure 5.2. With all of the analog filtering built and tested using an existing boost converter module from an old tracker, the PWM repowering circuitry was built, optimized a bit by changing resistor and capacitor values to match the drive requirements of the MOSFET gates in the boost converter, and tested. At this point a simple pulse-width modulated signal generator was used to drive the repowering circuit to "run" the new tracker circuitry. With the old tracker set in a manual adjust mode and the operating frequencies of the PWM drive signals on both trackers equalized, direct comparisons of non-computer controlled trackers could then be made.

In order to determine the efficiency of the DC-DC boost converter module, the power consumption of the supply card and appropriate control module first was found by determining the current flowing out of the battery with all of the desired circuitry active and no solar array attached. Doing this provided a base power consumption of roughly eight watts for the power supply card and less than a watt for the interface board of the old tracker. With this information,

the array was re-attached. The amount of power flowing from the array was then measured by multiplying the voltages on the capacitor across the array and the shunt resistor in series along the current path. Multiplying the voltage across the battery and the current flowing into the battery (measured by means of a DC ammeter) provided the power flowing into the battery. By addition of the power consumed by the supply card and interface board to the battery power, the efficiency of the boost converter was found to be:

$$\text{efficiency} = \frac{110\text{watts} + 8.5\text{watts}}{123.2\text{watts}} \cong 96\%$$

This test was conducted repeatedly yielding a mean of 96.1 percent and a standard deviation of 0.326. Considering the components (and wire) used, this is an extremely good converter efficiency.

This efficiency test was next conducted on the new tracker using the PWM signal generator built to provide a manually adjustable tracker. Measurements were conducted as before; in addition, however, the array voltage and current were measured using the new analog interface circuitry to verify their correct function. The efficiency of the boost converter using the new tracker was found to be:

$$\text{efficiency} = \frac{112\text{watts} + 9.2\text{watts}}{127\text{watts}} \cong 95\%$$

This efficiency is slightly lower than the old tracker. Note, however, the power consumption of the control circuitry increased slightly (mostly due to the temporary PWM signal generator) and the new tracker required only a single twelve volt supply to operate. In this test the more efficient Vicor DC-DC converter on the power supply card was being used to supply this voltage instead of the zener diode circuit shown in Figure 5.5. The other two DC-DC converters on this card were not needed and as such provided only power consumption. These converters were disconnected and the power consumption of the supply card measured again, yielding a consumption of barely three watts. The efficiency test was performed again.

$$\text{efficiency} = \frac{117.5\,\text{watts} + 4.7\,\text{watts}}{127\,\text{watts}} \cong 96\%$$

Repeated trials yielded a mean of 96.2 percent with a standard deviation of 0.231.

Microcontrolled "Slow" Power Tracking

With the testing and verification of the new interface modules now complete, the last step in prototype construction was the completion of the microcontroller board. Getting the microcontroller working using a wire wrapped board was not an easy task and required several weeks of effort before it was fully functional. The noise generated by the inductor in the boost converter was seen everywhere in the prototype, especially in the connections between boards. This problem is the primary motivation for placing all of the components on a ground plane for the final production version. A rough picture of the physical test setup for the new tracker is shown in Figure 6.2.

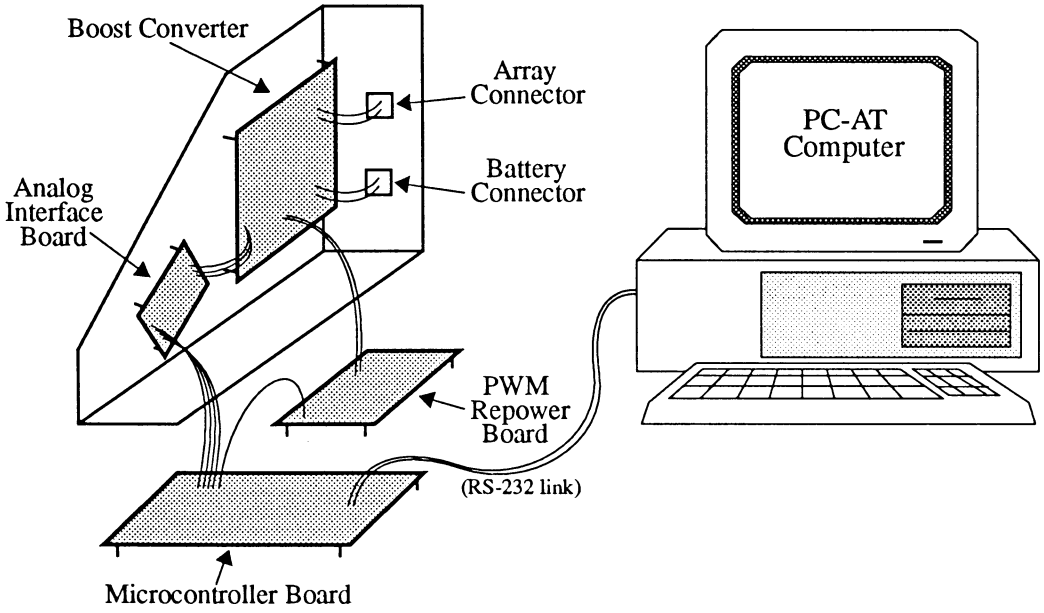

Figure 6.2: Physical construction of prototype tracker

Initial attempts at control software concentrated on the basic components required for operation of any control algorithm: a stable timebase to conduct all adjustment operations from, communication with a host computer by means of the microcontroller's serial port, and correct function of the A/D converter. Several days were spent writing software to test each of these items, starting with serial communication. Programming the 87C196KC proved relatively easy once a PC board had been made to run the microcontroller in "Auto Programming" mode. In this mode, the 87C196KC programs itself from a standard 16Kbyte (or larger) EPROM. A compiler bought from Intel was used to transform high-level programs written in C to 87C196 machine instructions. The output from the C compiler was programmed into an Intel 27C256 32Kbyte EPROM and this was used to program the microcontroller.

Once the various necessary software modules were written and tested, control of the tracker was attempted using a variant of the simple control algorithm given in Listing 5.7. The control software used adjusted the duty cycle of the pulse-width modulated output of the 87C196KC five times per second, sampling array voltage, array current, and battery voltage before making each adjustment. Using only normal sunny and party cloudy days as input to the system (no fast moving shadows simulated), this algorithm was evaluated for stability and the accuracy of the data being used for control adjustments was observed.

The simple control algorithm using only one previous power value proved to be unable to maintain a maximum power point. While the algorithm would find maximum power initially, albeit slowly, it would lose maximum power soon thereafter. Analysis of the sampled array voltage and current (by transmitting these values from the microcontroller to a PC-AT and running an analysis program on the telemetry data) showed some noise still present in the system, affecting the control algorithm by giving it false power trends on a regular basis. These false trends gradually caused the algorithm to deviate from true maximum power in one direction and slowly continue changing the duty cycle in that direction. Only after power reached zero would the algorithm reverse itself (due to some error checking statements) and

move back up to the "true" maximum power point for awhile.

One point must be made, however. Even while trying to debug this algorithm, the first thing apparent was that the power data being delivered by the microcontroller to the PC was far more accurate than was ever accomplished using the Winsystem control computer in the old PrISUm arrangement. Data in the microcontroller were deliberately kept in integer form to speed up the control algorithm as much as possible. Only on the other end of the telemetry connection was conversion to volts, amperes, and watts necessary for human comprehension. It was easy to directly measure the power coming from the array (as in the efficiency tests) and compare this to the converted values displayed on the PC. Looking at the power average of ten sample "sets" from the microcontroller and comparing this to measured power yielded values which deviated from each other by less than five percent -- certainly better than the fifteen (or more) percent error seen in the previous PrISUm electronics.

The simple algorithm was next modified to look at not only the previous power, but the power prior to that as well. Changing the direction of adjustment in the PWM duty cycle was only performed if the current power was less than both previous power values. This modified algorithm is shown in Listing 6.3.

```
direction = 1;
pwm_duty = 128;
while (true)
  {
    voltage = <sampled array voltage>;
    current = <sampled array current>;
    power[0] = voltage * current;
    if ((power[0] < power[1]) && (power[0] < power[2]))
      direction = direction * -1;
    pwm_duty = pwm_duty + direction;
    power[2] = power[1];
    power[1] = power[0];
    <delay>
  }
```

Listing 6.3: Pseudo-code of modified simple control algorithm

This additional power value stabilized the performance of the algorithm for a ground-based

solar array on both clear and cloudy days. Additional power values were added to the algorithm to determine if this would increase stability: the only noticeable difference in performance was an increase in the adjustment oscillation around the maximum power point -- under normal operation this control algorithm continually varies the duty cycle above and below the maximum power point (i.e. for a window of two starting at maximum power the duty cycle adjusts up two, then down five, then up five, and repeats this until solar intensity changes). Depending on the array arrangement, increasing the oscillation will noticeably deviate the power being delivered by the array and is undesirable for power tracking. Since no observable improvement was obtained by the addition of more past power values, two previous power values were considered enough to stabilize this algorithm for optimum performance.

With the slow algorithm now stable, fast-moving shadows were simulated on the roof array to observe the response speed of this algorithm. With such a large array it was impossible for a human to shadow the entire array in less than two-tenths of a second, so a large piece of cardboard was used to cover only part of the array (two panels out of six) as quickly as possible. Repeated shadowing and uncovering of these panels was performed while a stopwatch was used to record average response times from input change to maximum power re-attainment as read from the power meter in front of the battery. Average response was found in this way to be roughly four seconds. This compared favorably to the seven second (or more) response of the old PrISUm tracking system, and was largely due to duty cycle adjustments being performed five times a second instead of two.

Simulation of scattered shadows was a bit more difficult, and was approximated by simply having a person run in front of the array (without falling over the edge of the roof!). This did not exactly approximate the one-half second shadow shown in Chapter 3, but was about the best and easiest simulation possible. Under such simulated scatter, the simple control algorithm withstood the input change and returned to maximum power well within two seconds of shadow completion. Since the duty cycle was only being adjusted by one five times a second,

the algorithm could not fall far off the maximum power point duty cycle in a second or so.

To summarize, the modified simple control algorithm exhibited the following characteristics:

- more accurate power data and overall stability
- four second average sharp shadow to maximum power response time
- at most two second deviation from maximum power for scatter

All of these characteristics are considerably better than the previous PrISUm tracking system.

"Fast" Power Tracking

With the slow control algorithm stabilized and characterized, emphasis was now shifted to observing the high gain algorithm presented in Listing 5.8. An extreme example of this algorithm would use an adjustment maximum of half the possible duty cycle. In the case of the 87C196KC this amounts to eight bits, half of which is 128. Setting this as the maximum adjustment, however, allows the algorithm to go into a wildly unstable state where it varies the PWM duty cycle beyond the usable range for the boost converter, possibly putting the converter into discontinuous mode (see Appendix A). With the converter in discontinuous mode, the DC average of current falls and restoring continuous mode would be difficult unless the gain could be reduced to a reasonable level.

With this in mind, a smaller maximum adjustment value of thirty-two was used and a high gain algorithm tried which adjusted the duty cycle five times a second. Direct implementation of this simple high gain algorithm proved somewhat stable in unobstructed sunlight. The oscillation around the maximum power point was noticeably larger than the slow algorithm. Occasional cirrus clouds would, however, cause a sizable power deviation until the algorithm corrected itself. Analyzing the telemetry data from the microcontroller showed that this deviation was caused by the switching noise from the inductor and the slowly varying changes in solar intensity constantly causing the algorithm to change the gain (adjustment

value). Simulating fast-moving shadows on the array only worsened this instability by placing the algorithm in a mode where it maintained maximum gain and varied the duty cycle up and down several steps in each direction without reducing gain. This obviously would not work.

Similarly to the simple low gain algorithm, additional previous power values were added to the high gain algorithm in an attempt to stabilize it. This modified high gain algorithm is given in Listing 6.4.

```
adjust = 1;
mode = 1;
pwm_duty = 128;
while (true)
  {
    voltage = <sampled array voltage>;
    current = <sampled array current>;
    power[2] = power[1];
    power[1] = power[0];
    power[0] = voltage * current;
    if ((power[0] < power[1]) && (power[1] > power[2]))
      mode = 0;
    if ((mode == 1) && (abs(adjust) < MAX))
      adjust = adjust * 2;
    else if (abs(adjust) > 1)
      adjust = adjust / 2;
    if (abs(adjust) == 1)
      mode = 1;
    if ((power[0] < power[1]) && (power[0] < power[2]))
      adjust = adjust * -1;
    pwm_duty = pwm_duty + adjust;
    <delay>
  }
```

Listing 6.4: Pseudo-code of control algorithm with high gain

Adding additional power values in this way helped to keep the algorithm stable under clear sunlight, but the drastic instability during simulation of scattered shadows observed previously remained. Analysis of the telemetry data showed that the continually changing input conditions during scattered shadows caused the algorithm to oscillate wildly between increasing and decreasing gain modes, largely due to the checks performed to find the "maximum power" peak and reverse the direction of adjustment. This analysis matched very well the observed behavior

of the algorithm under simulated scattered shadows.

Some attempts were made to counteract this instability but no success was achieved, due largely to the high response gain of the algorithm. Reducing the gain or implementing a noise threshold only served to slow the response of the algorithm to simulated sharp shadows. Since in the final "combined" control software this algorithm should not have to deal with scattered shadows, the algorithm gain was limited to eight and the adjustment rate increased to twenty-five times per second. Stability under clear sunlight was again observed to be fair, with the increased oscillation observed earlier still present. Simulation of solid shadows yielded an observed response time of less than one second from shadow onset to maximum power attainment. This speed of response meets the performance goals for a tracker entering and leaving solid shadows as established in Chapter 3. When the adjustment rate was increased it proved impossible for the test software to deliver power data via the serial link fast enough to keep up with the sampling rate. Due to this problem only every fourth sample set was transmitted to the PC for analysis.

To summarize, the modified high gain control algorithm exhibited the following characteristics:

- one second average sharp shadow to maximum power response time
- limited stability to extended amounts of scattered shadows

"Combined" Power Tracking

While the simple control algorithm exhibited good stability and immunity to scattered shadows, it did not have a fast enough response to sharp shadows. The high gain control algorithm exhibited just the opposite: fast response to sharp shadows but limited immunity to scattered shadows. By combining both of these algorithms into one and controlling when each is allowed to run the system, the performance goals of immunity to scatter and one second response to sharp shadows should be attainable.

The only time the combined algorithm should run in high gain (fast) mode is when it is known a solid shadow has just been entered or left. Since simple (slow) mode is the default for its stability, power is generally known only five times per second. As was shown in Chapter 3, a solid shadow will overtake a panel moving at 25 mi/hr in 0.16 seconds. This is roughly equivalent to one sampling period. Since, however, the shadow could begin to overtake the array just prior to a set of samples and fast mode should not be called when the input is still changing (remember the instability to scattered shadows), using two sampling periods (0.4 seconds) to compare should be sufficient to wait out the shadow onset. Thus, if the current power is greater (or less than) the next to last power by some amount (chosen to be larger than a scattered shadow would cause), the algorithm should switch from slow mode to fast mode.

Going the reverse way is somewhat simpler since fast mode is only entered when power has fallen or risen drastically and fast mode must only find the new maximum power point quickly. Thus, once a new maximum power point is found, control should be returned to slow

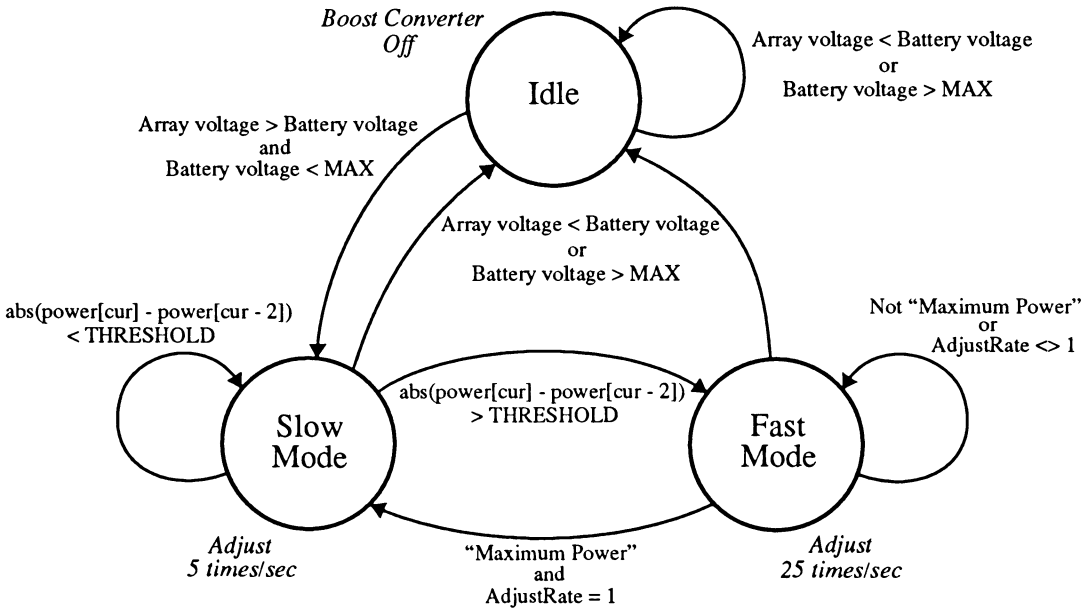

Figure 6.5: State diagram of combined power tracking algorithm

mode for stability reasons. Determining when maximum power is found is relatively easy but an additional constraint must be added: the duty cycle adjustment rate must be reduced to one before returning to the slow algorithm to insure it operates correctly. A state diagram of this control algorithm is shown in Figure 6.5.

Provided all the software thresholds are set adequately for a moving panel application, this combined algorithm should yield fast response to solid shadows, stability, and immunity to scattered shadows. Also, since these are software thresholds being used, rather than hardware thresholds, their values can be easily changed to adjust the operation of the control algorithm. This allows one algorithm to be tailored for many different applications (moving or stationary). There will inevitably be situations where this algorithm may not yield the fastest response possible (a scattered shadow immediately following a very short solid shadow will leave the algorithm in fast mode during a scattered shadow), but for a large majority of situations this algorithm will provide quick and appropriate responses to the various shadow types its array encounters. Algorithm performance can be modified either by changing the software thresholds, changing the adjustment rate for the slow and fast algorithms, or both. Further, several pre-programmed thresholds and rates can be selected by means of the DIP switches connected to the microcontroller's data lines. This provides an extremely flexible maximum point power tracker for almost any application.

The software listed in Appendix D is the implementation of the "combined" power tracking algorithm. Performance and optimization of this algorithm are still underway at present; however, some initial tests were conducted using the fixed array and a PrISUm array on the roof of Coover Hall. The algorithm remained in slow mode for any normal cloud shadows encountered over several weeks of testing. Simulating scattered shadows also kept the algorithm in slow mode with a "mode switch" threshold of five watts. Depending on the magnitudes of scattered shadows encountered when the tracker is drive tested on PrISUm, this threshold may have to be raised to keep slow mode in effect during scatter.

Response to simulated fast shadows was observed at less than a second; due to the difficulty of determining when exactly the shadow was being applied on the roof it was impossible to obtain a more accurate value. Some work was also done on the transition from fast mode back to slow mode. The first few attempts of defining "at maximum power" while in fast mode were not working exactly as desired, so the code was re-written to better define this threshold between modes. Here again, until the tracker is actually drive tested this threshold will not be completely established.

In summary, the combined power tracking algorithm exhibited the following characteristics in tests using roof mounted arrays and simulated fast moving shadows:

- quick response to fast moving solid shadows
- good immunity to scattered shadows
- overall system stability and reliable power data

These all match the performance goals established for an optimized fast response power tracker for a moving solar array.

Control Systems Analysis

While this algorithm was observed to be stable under all of the input situations it was subjected to, it is desirable to characterize this system very precisely and prove the stability of the system. The overall system response is a combination of the responses of the array, boost converter, and control system. The response of the battery is taken to be a constant since it is a fixed voltage source (in reality it will vary with the pulse charging output of the boost converter enough to be significant, but as will be shown the other three responses are complex enough to merit this approximation). Further, additional error checking performed to shut off and turn on the boost converter has no noticeable impact on the ordinary performance of the control algorithm and does not need to be included in the analysis. A block diagram of the overall system to investigate is shown in Figure 6.6.

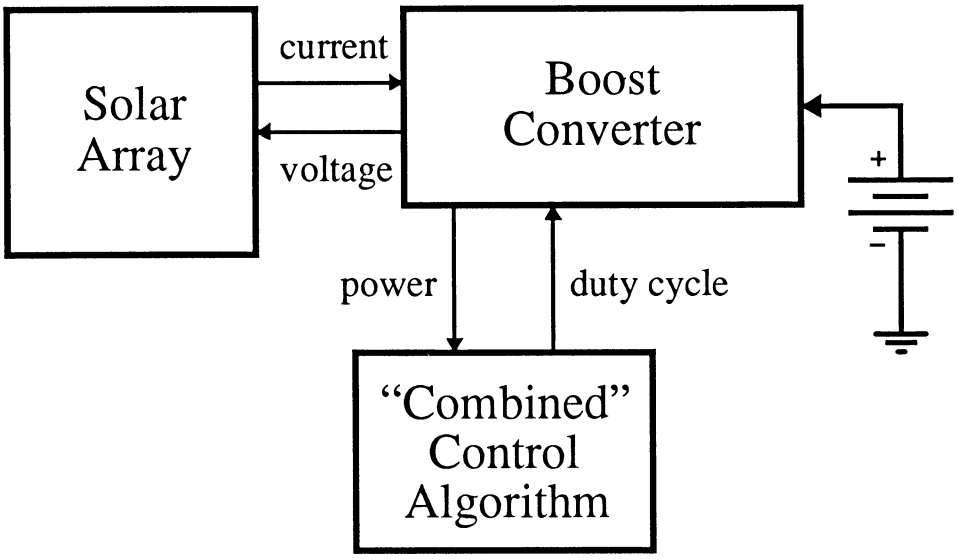

Figure 6.6: Block diagram of tracker control system

Although this might seem to be a fairly simple system to analyze, attempting to find transfer functions for each of the parts quickly demonstrates the complexity of the overall system. Solar cell responses to individual input changes can be characterized (see Appendix B), but attempting to include the effects of insolation, temperature, component resistances, and voltage changes into one formula is almost impossible. Assuming an ideal boost converter (no component losses) the power change for a change in duty cycle can be found as follows:

$$I_L = I_P - I_0 e^{\frac{qV_L}{kT}} + I_0 \tag{6.1}$$

where I_P varies with sunlight and I_0 is the diode reverse saturation current [3]. Since I_0 is small, the last term is usually neglected. Power at any given point in time can be found by

$$P_A[n] = I_L V_A[n] = I_P V_A[n] - I_0 V_A[n] e^{\frac{qV_A[n]}{kT}} \tag{6.2}$$

and power change from the previous value to the current value can be expressed as

$$\Delta P_A[n] = -I_p \Delta V_A - I_0 V_A[n-1] e^{\frac{qV_A[n-1]}{kT}} + I_0 (V_A[n-1] + \Delta V_A) e^{\frac{q(V_A[n-1]+\Delta V_A)}{kT}} \tag{6.3}$$

For an ideal boost converter and a fixed battery voltage, the change in voltage corresponding to a change in the duty cycle across a single solar cell can be written as

$$\Delta V_A = \frac{V_b \Delta d}{255n} \tag{6.4}$$

where V_b is the battery voltage, n is the number of cells in a series string in the array, and the duty cycle is expressed as a number between 0 and 255 (as is the case with the 87C196KC microcontroller). The change of power for a duty cycle change is given in Equation 6.5.

$$\Delta P_A[n] = -I_P \left(\frac{V_b \Delta d}{255n}\right) - I_0 V_A[n-1] e^{\frac{qV_A[n-1]}{kT}} + I_0 \left(V_A[n-1] + \frac{V_b \Delta d}{255n}\right) e^{\frac{q\left(V_A[n-1] + \frac{V_b \Delta d}{255n}\right)}{kT}} \tag{6.5}$$

A block diagram of this nonlinear "transfer function" for an ideal boost converter and a simplified solar cell (from an array) is pictured in Figure 6.7.

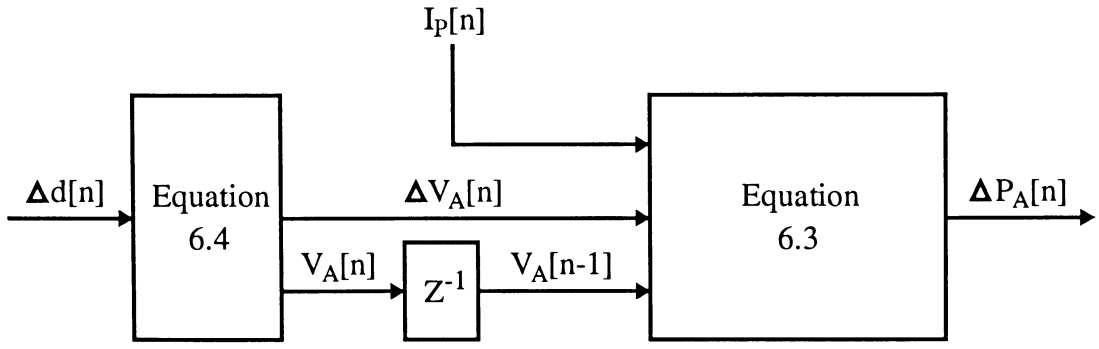

Figure 6.7: Solar cell and ideal boost converter

The boost converter is no less complex to characterize. The steady-state analysis shown in Appendix A gives the output to input voltage with respect to the duty cycle of the PWM signal used to drive the converter switch. Since inductor and other component losses must be taken into account for a large current system, this response becomes more complicated. Adding response to small-scale changes in duty cycle further complicates the equation beyond use. Even in this low-frequency situation, the capacitor placed across the array must also be taken into account. With all of these factors included, the boost converter input to output voltage is given in Equation 6.6:

$$V_a = V_b(1-D) + r_L\left(I_P - I_0 e^{\frac{qV_a}{kT_c}} + I_0\right) - r_L C\frac{dV_a}{dt} \qquad (6.6)$$

where r_L is the inductor resistance, C is the capacitor placed across the array, and I_P is the time-varying current flowing from the array. Note that unlike previous formulas this is in the continuous-time domain. Solution of this in the discrete-time domain is beyond the scope of this text, but the net result of accounting for losses in the boost converter and the capacitor across the input is to make the change in array voltage ΔV_A from the boost converter now dependent upon I_P as well. Previously only the array was dependent upon this value.

As if this is not complex enough, the impulse response of the "combined" control algorithm must somehow be derived. Since this algorithm was written with little thought toward emulating a linear control system, there is no direct way of describing this control algorithm and using this to find the Z transform of its impulse response. Further, the system cannot be described using purely linear terms in data flow diagram form.

To attempt to find the impulse response of the combined algorithm, the impulse responses of both the slow and fast modes must first be found. Using various discrete-time processing techniques to analyze both algorithms, the data flow diagrams shown in Figure 6.8 and 6.9 were derived [4].

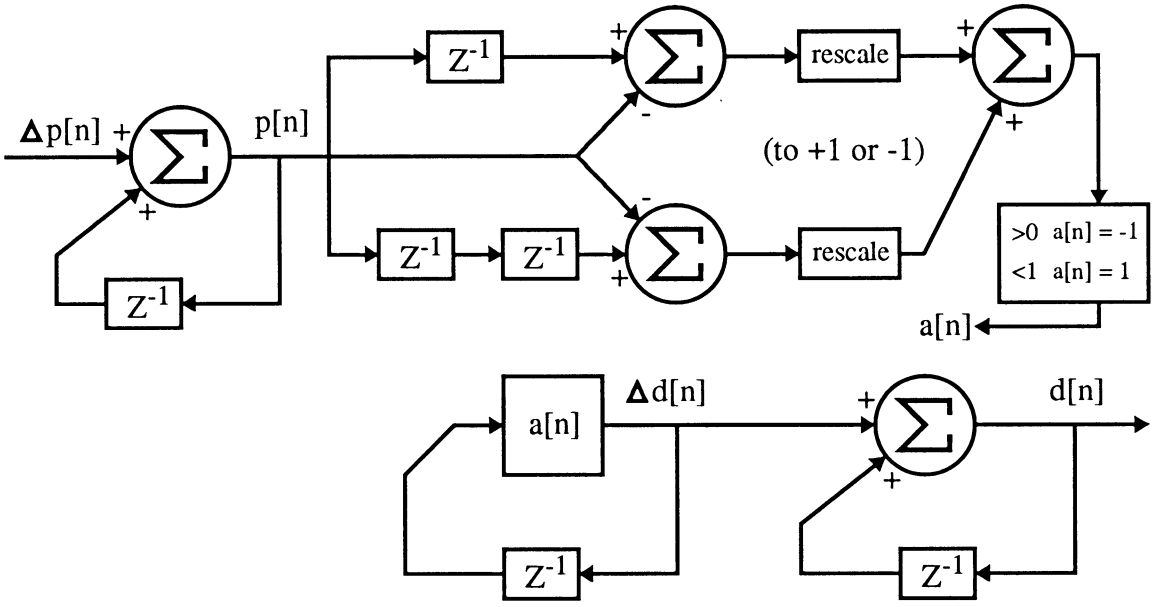

Figure 6.8: Data flow diagram of slow mode algorithm

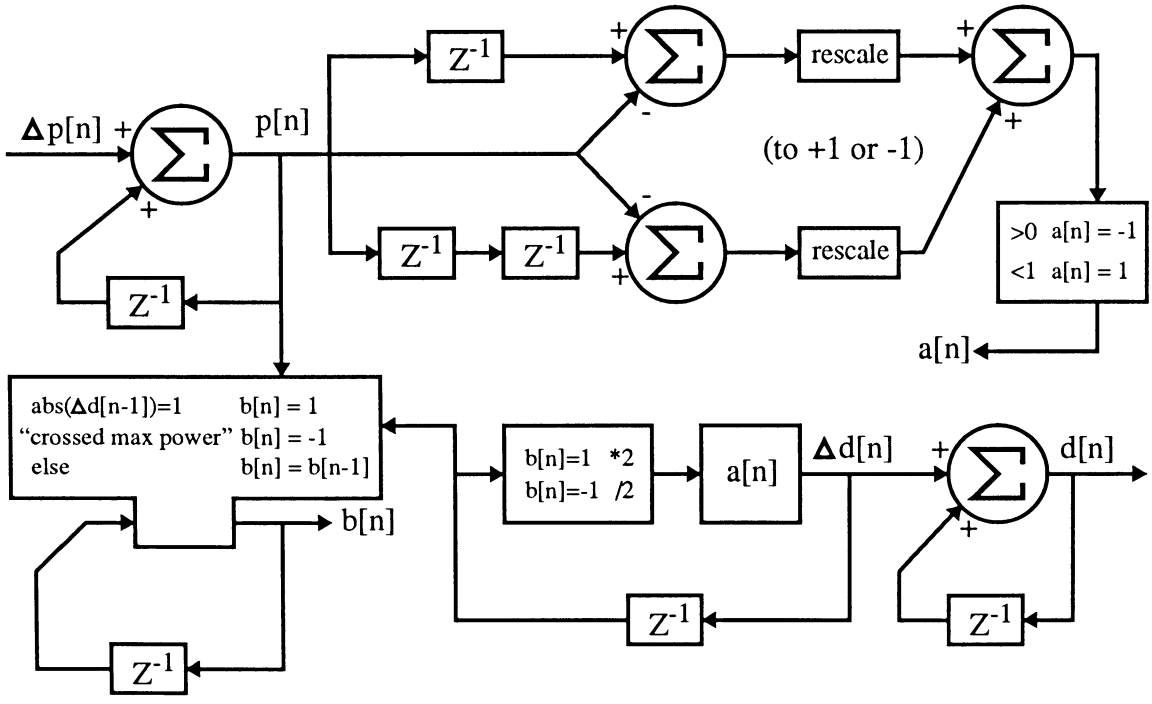

Figure 6.9: Data flow diagram of fast mode algorithm

Immediately noticeable in these data flow diagrams is the presence of several time-varying scalar multiplication values controlled by other portions of the diagram. This leads to drastic problems in finding a system response since no direct methods are available to derive a Z transform from a data flow diagram with such time-varying values.

Because neither algorithm has a readily computable impulse response it is impossible to characterize the overall system using conventional control systems analysis. Some information can still be derived, however, from the overall control feedback loop, as shown in Figure 6.10. This loop consists of the boost converter response (with an assumed constant battery voltage), the application of this to the array and an observed power in return, and the (impossible to find) impulse response of the control algorithm which returns a duty cycle to the boost converter.

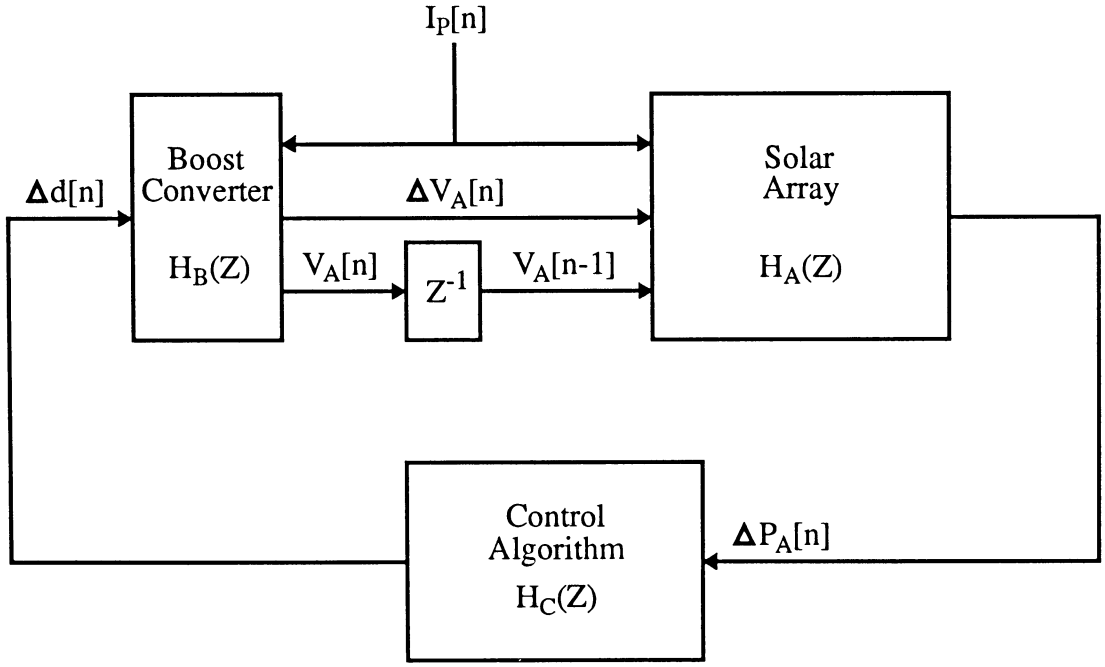

Figure 6.10: Overall system feedback loop

The two predominant factors seen in the feedback loop are the array response to changes in the array voltage and the duty cycle response from the control algorithm to changes in array

power. Since both of these have some form of time-varying multiplier values in the feedback path, neither is a linear system. If the assumption is made that I_P (and thus V_{mp}) changes in step fashion and is then constant for a period of time, the system should (hopefully) stabilize at some equilibrium (with some small deviation for the oscillation in the control algorithm). To find a usable response, another assumption must be made that the response of the boost converter is sufficiently quick to be instantaneous when viewed from the microcontroller's point of view (since this is the limiting factor in the discrete-time analysis). By noting that in slow mode small adjustments are made at low frequency the step response of the remainder of the feedback loop can be roughly computed.

Using duty cycle adjustments of only one out of 255 possible, the voltage change on an individual cell in the array will be

$$\Delta V_a = \frac{V_{bat}}{255n} \tag{6.7}$$

with a corresponding change in cell current dependent upon sunlight and the position of the previous power point on the effective I-V curve of the array. These values will be array-dependent, so some values must be substituted to evaluate the effects of stepped duty-cycle changes to the boost converter. Typical values for the PrISUm arrays are $I_0 = 6.91 \times 10^{-9}$ A, $I_P = 1.5$ A, and $n = 60$ cells. Considering a change from the maximum power point (V_A[n-1] = 0.5 volts typically) with a battery voltage of 100 volts and a temperature of 300 Kelvin, the change in power will be

$$\Delta P_A = (-1.5\text{A})\left(\frac{100\text{V}}{255\,(60)}\right) - 6.91\times10^{-9}\,(0.5\text{V})\,e^{\frac{q\,(0.5\text{V})}{k\,(300\text{K})}} +$$

$$6.91\times10^{-9}\,(0.5\text{V}+\frac{100\text{V}}{255\,(60)})\,e^{\frac{q\,(0.5\text{V}+\frac{100\text{V}}{255\,(60)})}{k\,(300\text{K})}}$$

which, with substitution of the appropriate constants, reduces to $\Delta P_A = 0.25$ watts.

Repeating this calculation with the maximum duty cycle adjustment for fast mode (eight) yields a power change from maximum power of $\Delta P_A = 6.2$ watts. Fast mode is obviously capable of more rapidly changing the amount of power coming from the array in a short amount of time. Out of a maximum array power of well over 150 watts (in full sunlight) both of these power changes are small enough to not cause any great concern about system instability. If the upper bound on the fast mode adjustment was increased from eight, however, that mode would be able to change the power between samples by a larger amount, leading to the possibility of drastically overshooting the maximum power point. This could cause difficulties in stabilizing this mode, as was experienced with early tests of the fast mode algorithm (presented earlier in this chapter).

The boundaries between slow and fast modes must be evaluated as well. This can be done intuitively simply by realizing that switching from slow to fast mode is accomplished by the current power differing from the next to last power by more than a specified amount. This is an absolute case with no flexibility.

The reverse case is a bit more complex. Two items must be attained by the fast mode before it will transfer control back to the slow mode: the adjustment rate must shrink back to one and the tracker must be close to ("at") maximum power. Here again, intuitively this can be shown with the realization that the algorithm will begin to shrink the adjustment rate when maximum power has been crossed and will continue to do so down to the minimum of one. Once this is accomplished control is returned to the slow mode (refer to the software listing in Appendix D and note that when the adjustment rate is reduced to one in the fast mode handler, the mode is switched to slow). Since the fast mode should be stable overall, at some point it will reduce the adjustment rate to one and the mode will switch back to slow. This is an absolute case with no flexibility, so this transition is also stable.

Thus, although a "traditional" control systems analysis could not be performed on this system, the feedback loop derived indicates that small changes in duty cycle should not cause

major overall stability problems in this system. The only additional concerns in such a discrete-time system are the effects of quantization error and noise on the control system [4]. As shown earlier, noise immunity was designed into the control software where needed, and quantization error was not found to affect system performance during any tests conducted.

Production Version Additions

The last step in implementing this fast response power tracker is the design of a printed circuit board to place all components on one ground plane. All of the noise problems encountered in a solar powered vehicle (or any electric vehicle) can be minimized by placing all signals as close to a ground plane as possible. This is almost equivalent to running each signal through a coaxial cable with the outer shield connected to ground. Using a multiple-layer printed circuit board will reduce the space between each signal and ground. Shrinking this dimension reduces the susceptibility of each signal to induced voltages and helps significantly in the reduction of RFI problems throughout the board [5].

In addition to the ground plane, placing the noisiest component in the boost converter (the inductor) as far away from the most noise-sensitive component in the controller (the microcontroller) as possible will help reduce the noise problems encountered by the microcontroller. To shield the microcontroller from all other noise sources (the motor and radios), it can be enclosed in a metal box connected to the ground plane of the printed circuit board. Provided space is left for the traces on the top side of the board to "escape", a very tight shield is placed around the microcontroller and should greatly reduce all noise experienced by that chip.

Connectors for the external twelve volt supply must be included, as well as a DIP socket for the Maxim chip used for the optional RS-232 interface. Also desirable for a production version of this power tracker are: a power switch to turn on and off the tracker, and a reset switch to the microcontroller (this should never need to be used, since a watchdog timer on the

microcontroller will reset the chip within a second if the software ever enters an infinite loop).

The final element to be added deals more with a stationary application than a car-based one. In order for this tracker to function correctly unattended for days (or longer) at a time, some means must be provided for the microcontroller to shut off the twelve volt supply to the PWM repowering circuitry. This is necessitated by the fact that in order to "turn off" the boost converter, the PWM signal must be held high to pull the gates of the power MOSFETs low (remember the inversion performed by the PWM repowering circuit). This causes continuous current to flow through the 150 ohm resistor connected from the drain of the intermediate MOSFET to twelve volts (see Figure 5.3). This power consumption is obviously undesirable and some means to selectively switch this voltage on and off should be provided.

CHAPTER 7. CONCLUSIONS

While solar powered vehicles are years away from commercial availability, systems capable of effectively power tracking arrays mounted on such vehicles are currently feasible. Such power tracking systems must be able to respond to varying types of shadows and the differing speeds with which they overtake a solar array in order to keep the power output of the array at maximum.

The work described here involved the analysis and redesign of a switch-mode power tracker used to maximize the solar power output of an array mounted on a moving vehicle. This new power tracker is built around a DC to DC boost converter and is controlled by an Intel 87C196KC microcontroller. Control software for this microcontroller was developed to respond to solid and scattered shadows while maintaining stability at all times. The operational flexibility and accuracy of telemetry data for this system were also investigated.

Given the rising costs of energy production and the ecological problems associated with other forms of energy production, the need for solar powered or solar assisted systems is rising. With the varying power tracking demands of each application, a flexible and inexpensive solar power tracker will be a necessity in the near future. The power tracker presented in this work is one implementation of such a flexible system capable of maximizing the solar power output of an array for almost any application.

BIBLIOGRAPHY

[1] Stone, H. *Microcomputer Interfacing*. Reading: Addison-Wesley, 1982.

[2] Cuk, S. *Basics of Swithed Mode Power Conversion: Topologies, Magnetics and Control*. Washington, D.C.: Office of Naval Research.

[3] Hu, C. and White, R. M. *Solar Cells: From Basic to Advanced Systems*. New York: McGraw Hill, Inc., 1983.

[4] Oppenheim, A. and Schafer, R. *Discrete-Time Signal Processing*. Englewood Cliffs: Prentice-Hall, 1989.

[5] Williamson, T. "Designing Microcontroller Systems for Electrically Noisy Environments." *Embedded Applications*. Mount Prospect: Intel, 1990.

[6] Sum, K. K. *Switch Mode Power Conversion Basic Theory and Design*. New York: Marcel Dekker, Inc., 1984.

[7] Mitchell, D.M. *DC - DC Switching Regulator Analysis*. New York: McGraw Hill, Inc., 1987.

[8] Maycock, P. and Stirewalt, E. *A Guide to the Photovoltaic Revolution*. Emmaus: Rodale Press, 1985.

[9] Pulfrey, D. *Photovoltaic Power Generation*. New York: Van Nostrand Reinhold, 1978.

[10] Rauschenbach, H. *Solar Cell Array Design Handbook*. New York: Van Nostrand Reinhold, 1980.

ACKNOWLEDGEMENTS

I would like to thank my major professor, Dr. Allan G. Potter, without whose help and guidance the original PrISUm system or this work would never have been completed. It was a welcome relief to always be able to fall back upon Dr. Potter's expertise whenever something inexplicable manifested itself.

I also would like to thank the other two "die hard" members of the original electrical systems group, Dave Swift and Kerry VanderKamp. Without Dave's insight and Kerry's leadership the solar car project would never have started in the first place. Special thanks must also be extended to Mr. Roland G. Struss, Assistant Director for Operations of the Ames Laboratory, for his help in obtaining everything from the car I needed when I really needed to use it, and for funding construction of several printed circuit boards for design prototyping.

In addition, I must also thank the various faculty members and staff of the Electrical and Computer Engineering Department at Iowa State University who helped me with their support and patience in allowing me to work away from my assistantship for several months.

APPENDIX A. DC TO DC BOOST POWER CONVERSION

Analysis of Operation

Although several types of DC to DC voltage converters are currently in use, the most efficient of these converter types is a switched-mode converter. This type of converter employs an energy storage device and a means to selectively switch this device between the converter's input and output. A practical realization of a switched-mode boost converter is shown in Figure A.1. Under normal operation the MOSFET is switched between off and saturated, causing two distinct circuit topologies. The equivalent circuits with the boost converter in ON and OFF states are shown in Figure A.2 and A.3.

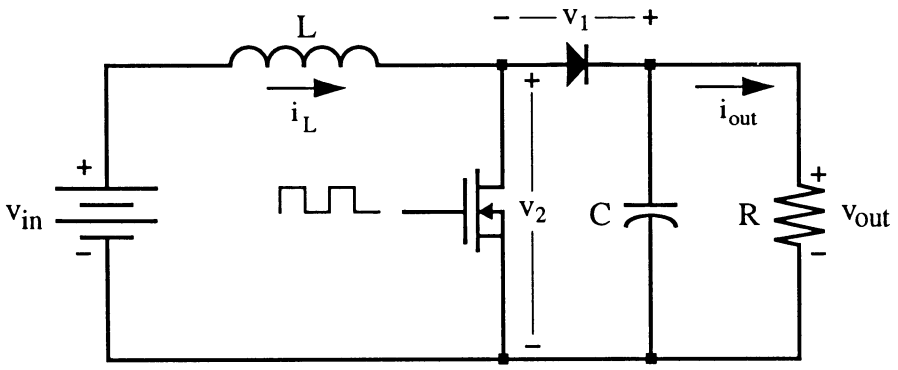

Figure A.1: Practical boost converter

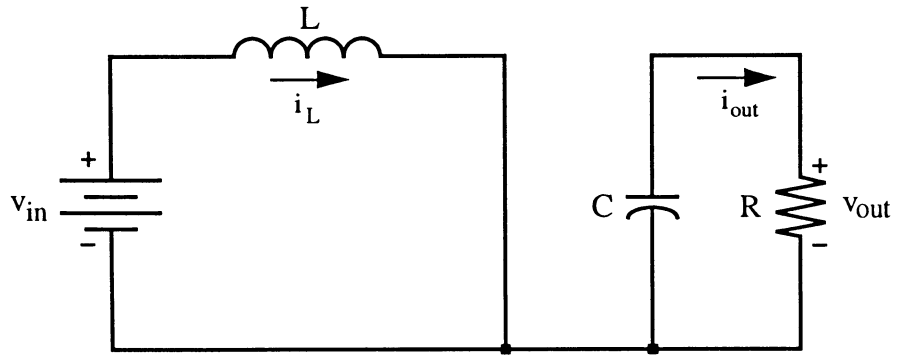

Figure A.2: Boost converter in ON state

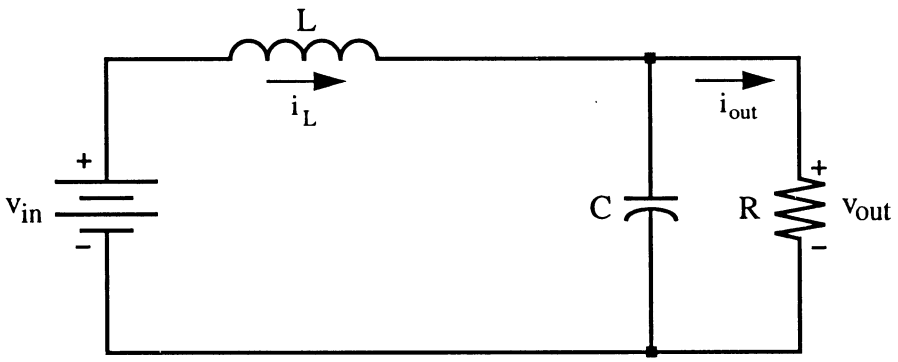

Figure A.3: Boost converter in OFF state

When the boost converter is in the ON state, the inductor current i_L rises from its initial value of i_a to a maximum value of i_b, which varies with the amount of time in the ON state and the values of the components used. During this time the capacitor is discharged to provide power to the load. Switching to the OFF state causes the energy stored in the inductor to flow to the output, so i_L falls from i_b during the OFF state. Two possibilities exist during normal operation of this boost converter: the inductor current could fall to zero sometime during the OFF state, or it could remain non-zero upon return to the ON state. These possibilities are referred to as *discontinuous* and *continuous* modes, respectively [2].

When a boost converter is operating in discontinuous mode, its input current is pulsating and is zero a portion of the time (see Figure A.4). For a battery source this is not necessarily a problem; a solar array, on the other hand, is constantly generating power under sunlight and must somehow dissipate this power whenever the boost converter is not accepting current. This power is dissipated in the form of heat in the cells, raising each cell's temperature, lowering its efficiency, and increasing the probability of it cracking. This is very undesirable and as such a boost converter used to power track a solar array should run in continuous mode at all times.

In continuous mode the inductor current never falls below zero and under stable conditions will vary between i_a and i_b continuously. Thus the ripple in the input current has a

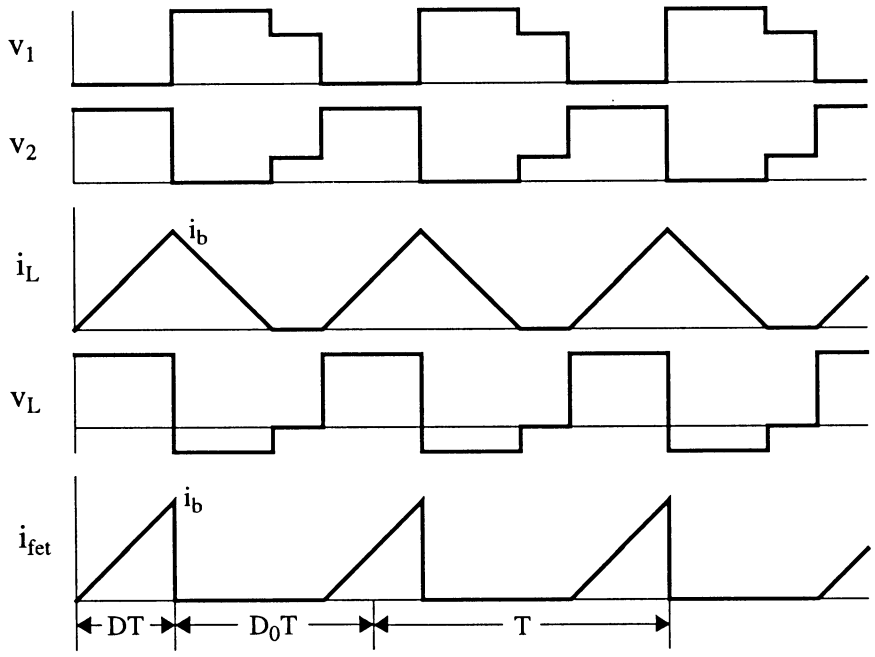

Figure A.4: Boost converter in discontinuous mode

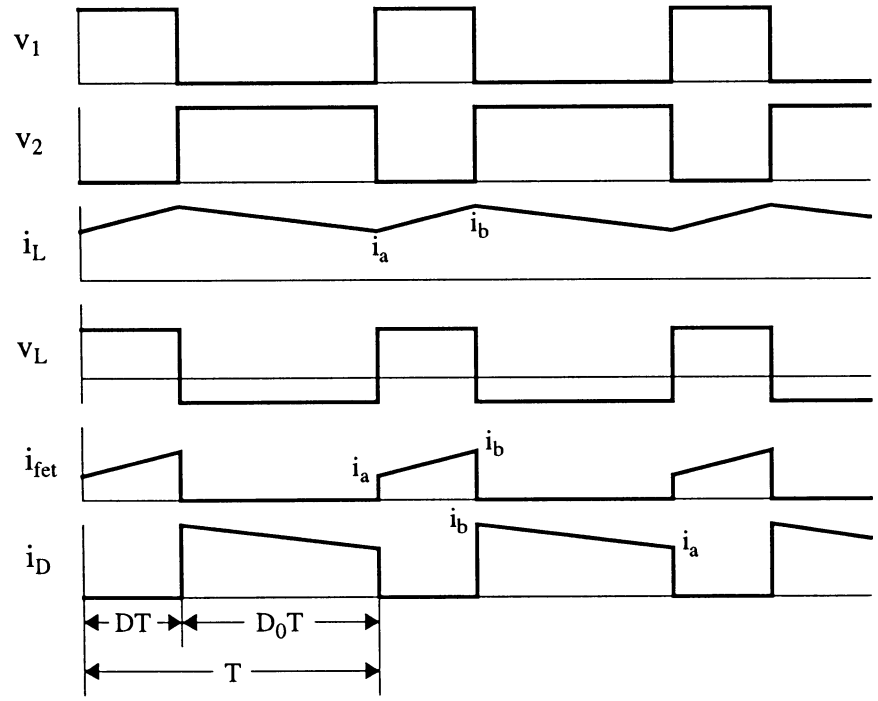

Figure A.5: Boost converter in continuous mode

magnitude of i_b minus i_a. If the source is capable of providing a fairly constant current (as in the case of a solar array in good sunlight) this ripple magnitude can be decreased by increasing the value of L to effectively match the input impedance of the boost converter to the source. Ripple in the voltage delivered to the load from the boost converter can be reduced by increasing the value of C. There is a trade-off between the losses in both L and C and the efficiency increase gained by increasing the sizes of these components [6].

Continuous Mode Analysis

A steady-state (constant duty cycle) analysis of a boost converter in continuous mode operating as shown in Figure A.5 begins as follows [7]:

$$V_{in}DT = (V_{out} - V_{in})(1 - D)T \tag{A.1}$$

which reduces to

$$\frac{V_{out}}{V_{in}} = \frac{1}{1 - D} \tag{A.2}$$

for voltage and

$$\frac{I_{out}}{I_{in}} = 1 - D \tag{A.3}$$

for current. The input current ripple (i_b - i_a) can be written as

$$\Delta I = \frac{V_{in}D_L T}{2L_b} \tag{A.4}$$

where D_L is the duty cycle corresponding to maximum input voltage and L_b is the minimum inductance required to maintain continuous conduction for a given minimum load as given by

$$L_b = \frac{R_{max}TD_L(1 - D_L)^2}{2} \tag{A.5}$$

The output voltage ripple can be calculated by looking at the discharge of C through R during the ON time of the converter. Assuming an almost constant average voltage across the

capacitor *V(t)* during this discharge, the output ripple can be written as:

$$-\frac{V(t)}{RC} \cong \frac{\Delta V}{\Delta t} \cong \frac{\Delta V_{out}}{DT} \tag{A.6}$$

which reduces to

$$\frac{\Delta V_{out}}{V_{out}} = \frac{D_H T}{R_{min} C} \tag{A.7}$$

assuming no losses in any of the components. D_H is the duty cycle corresponding to minimum input voltage. Since the inductor used can be quite large and will have some resistance r_L this must be taken into account. Substituting the loss due to the inductor into Equation A.2 yields

$$\frac{V_{out}}{V_{in}} = \frac{R(1-D)}{r_L + R(1-D)^2} \tag{A.8}$$

Differentiating (A.8) with respect to D and equating it to zero simplifies to

$$D_m = 1 - \sqrt{\frac{r_L}{R}} \tag{A.9}$$

Substitution of (A.9) into (A.8) gives the maximum boost converter gain [6].

$$\left.\frac{V_{out}}{V_{in}}\right|_{max} = \frac{1}{2}\sqrt{\frac{R}{r_L}} \tag{A.10}$$

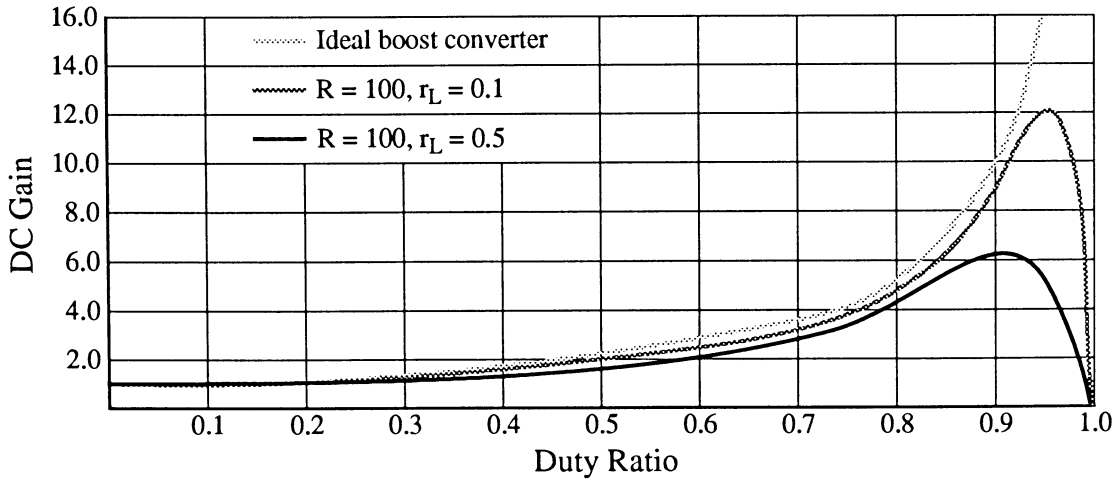

Figure A.6: Plot of boost converter gain

Component Value Calculation

For the solar panels on PrISUm fair minimum and maximum voltages are 40 volts and 80 volts, respectively. The resistive load is actually either a lead-acid or silver-zinc battery, roughly modeled by a series resistance and a large capacitance. Since the cost to weight ratio of silver-zinc battery cells is much better than that of lead acid cells, silver-zinc cells are used as the main battery whenever possible. The series resistance of these cells is higher than that of lead acid cells, and placing many silver-zinc cells in series to create a one hundred volt battery increases the effective series resistance. A fair estimate of the maximum total series resistance for such a battery pack is 100 ohms.

To find the correct inductor value for a boost converter, first compute D_L:

$$\frac{V_{out}}{V_{in_{max}}} = \frac{D_L}{1-D_L} = \frac{100\text{volts}}{80\text{volts}} \Rightarrow D_L = 0.56$$

Most pulse-width modulated control chips available commercially can run around 25 kHz, and the PWM outputs of all microcontrollers available on the market run at or below this frequency. Choosing the highest available frequency will help to keep the converter in continuous mode at all times since L will discharge less at higher frequencies. Thus

$$T = \frac{1}{25\text{kHz}} = 40\mu\text{sec}$$

and R_{max} = 100 ohms (for the silver-zinc battery pack) yields an L_b of

$$L_b = \frac{R_{max}T(1-D_L)^2}{2} = \frac{(100\Omega)\ (40\mu\text{sec})\ (1-0.56)^2}{2} = 0.387\text{mH}$$

Looking at the maximum input current ripple for this inductance:

$$\Delta I_{max} = \frac{V_{in}D_L T}{L_b} = \frac{(80\text{volts})\ 0.56\ (40\mu\text{sec})}{0.387\text{mH}} = 4.63\text{A}$$

this is obviously far too large a current ripple for an array producing something less than 200

watts at roughly 62 volts (i.e. 3.23 amps). Since 0.387 millihenries is a *minimum* value required to maintain continuous conduction, increasing L to 10 millihenries will yield a maximum input current ripple of:

$$\Delta I = \frac{V_{in} D_L T}{L} = \frac{(80\text{volts})\,0.56\,(40\mu\text{sec})}{10\text{mH}} = 0.179\text{A}$$

This is a more acceptable current ripple for a solar array.

The load is actually a fixed voltage, so the output ripple observed will be mostly loss in the series resistance of the battery cells and the wires connecting the converter to the battery. Choosing an output voltage ripple of 0.1 volts and a minimum battery resistance of 25 ohms, C can be calculated to be:

$$\frac{V_{out}}{V_{in_{min}}} = \frac{D_H}{1 - D_H} = \frac{100\text{volts}}{40\text{volts}} \Rightarrow D_H = 0.714$$

$$\frac{\Delta V_{out}}{V_{out}} = \frac{D_H T}{R_{min} C} = \frac{0.714\,(40\mu\text{sec})}{(25\Omega)\,C} = \frac{0.5}{100} \Rightarrow C = 228\mu\text{F}$$

The output voltage is really not free to float as with a solely resistive load, and since finding 228 microfarad capacitors with breakdown voltages over 100 volts is not an easy (or cheap) task, a smaller capacitor can be used. For this project 200 volt 20 microfarad capacitors are relatively inexpensive and will perform the task sufficiently.

Input voltage ripple should also be kept to a minimum, as this varies the voltage across each cell and will move the cell from maximum power. Adding a 1 microfarad capacitor across the input to the boost converter (before the inductor) should be sufficient to smooth out the input voltage and additionally help with the input current ripple already reduced significantly by a larger inductor.

APPENDIX B. SILICON SOLAR CELLS AND SOLAR ARRAYS

Photovoltaic Cells

Silicon photovoltaic cells are by far the most common cells manufactured and available today, but the basic principles of operation for silicon photovoltaic cells apply to other types of cells as well. Silicon cells are not the most efficient cells available, but due to the rapid growth of the silicon semiconductor industry much manufacturing research has taken place which has lowered the cost of production for both silicon ICs and solar cells. This low cost and wide availability make silicon solar cells applicable to many power generation applications.

A basic silicon solar cell is simply a very large p-n diode doped to provide an effective current flow when biased in the forward direction. Unlike a conventional diode, however, external emf is not necessary to induce current flow in a circuit. Photons striking the upper surface of the cell (the n side of the diode) with sufficient energy can free electrons in the silicon atoms and cause current to flow if the n and p regions of the diode are connected together [8]. An illustrated cross-section of a silicon photovoltaic cell is shown in Figure B.1.

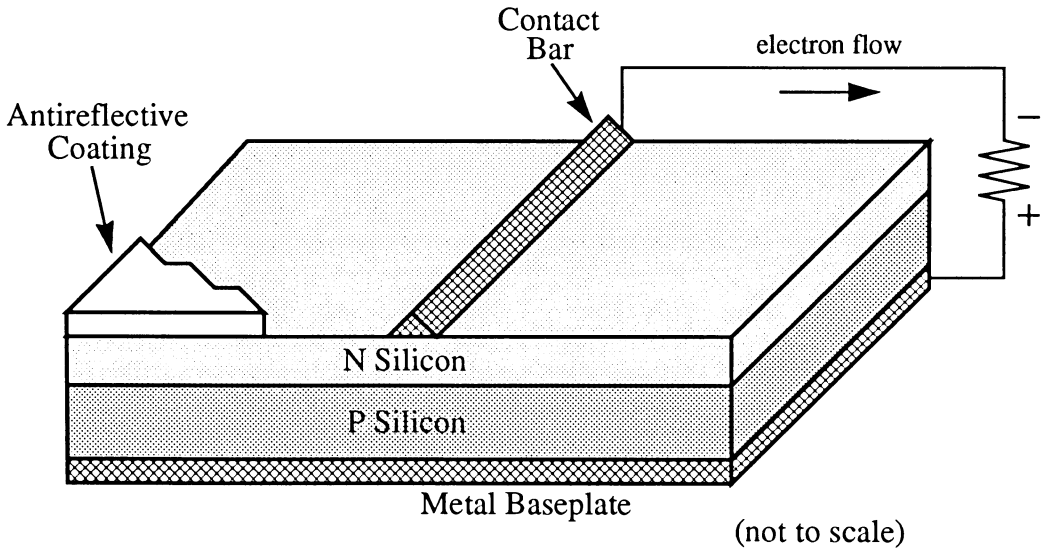

Figure B.1: Silicon solar cell

A precise model of a silicon cell is rather complicated, but a simple lumped-parameter model is sufficient to approximate the performance of such a cell. The two predominant resistances present in the cell are the shunt resistance (R_{sh}) present in any diode, and a parasitic series resistance (R_s) which is a combination of the bulk resistance in the substrate, contact resistance between silicon and metal, and the resistance of the metal leads used to conduct current. The lumped-parameter equivalent circuit for a solar cell is shown in Figure B.2 [9].

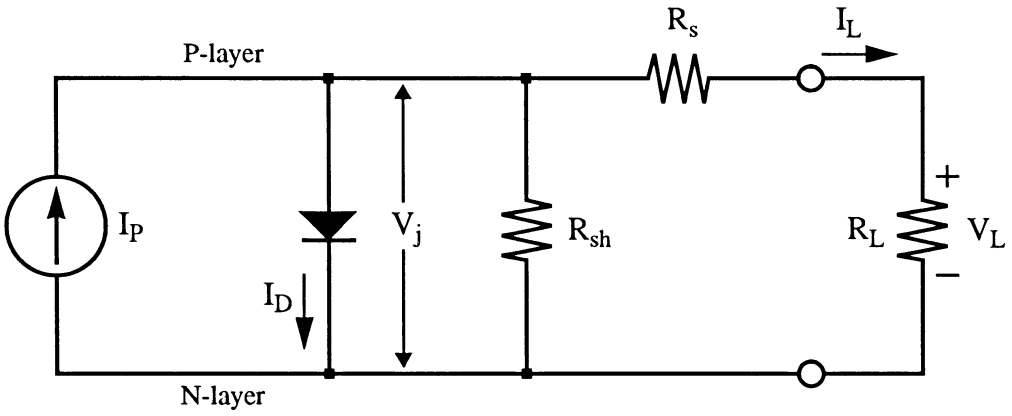

Figure B.2: Solar cell equivalent circuit

Assuming I_P and I_D are independent, applying the principle of superposition to this circuit yields description of a solar cell (although certainly not complete) as given by

$$I_L\left(1+\frac{R_s}{R_{sh}}\right) = I_P - I_D(V_j) - \frac{V_L}{R_{sh}} \qquad \text{(B.1)}$$

where V_j is the familiar voltage drop across the junction of a diode. I_P will be time-varying with incident solar intensity on the cell, and I_D will vary with cell temperature. Various characteristics differ between "identical" solar cells straight off a manufacturing line, but it is apparent that R_s should be small and R_{sh} be large to keep efficiency as high as possible. Typical values for silicon cells are $R_s < 0.5$ ohms and $R_{sh} > 500$ ohms. From this model a typical solar cell current versus voltage characteristic can be derived and is shown in Figure B.3.

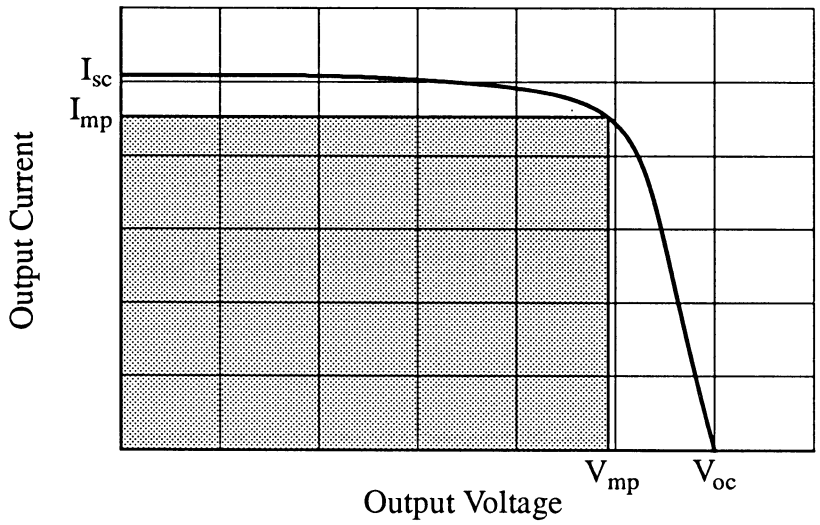

Figure B.3: Solar cell I-V characteristic

Several important values are denoted on this curve. First is the open-circuit voltage (V_{oc}) present across the terminals of an unloaded cell under sunlight. Next is the short-circuit current (I_{sc}) measured by shorting the terminals of a solar cell. The current and voltage at maximum power (represented by the area of the largest rectangle that can be fit under the I-V curve) are denoted as I_{mp} and V_{mp}. If possible, it is clearly desirable to operate the cell at this point at all times regardless of the load demand. If a cell is operating at maximum power its efficiency can be described by

$$\eta = \frac{I_{mp}V_{mp}}{P_i a} \tag{B.2}$$

where a is the effective area of the cell (some area is lost by the upper surface contacts) and P_i is the incident solar power density the cell is currently receiving. It is more common to express this efficiency in terms of the **fill factor**, defined as

$$FF = \frac{V_{mp}I_{mp}}{V_{oc}I_{sc}} \tag{B.3}$$

so that cell efficiency can now be expressed as

$$\eta = \frac{FF\,(I_{sc})\,V_{oc}}{P_i a} \qquad \text{(B.4)}$$

Typical fill factors for production silicon solar cells are between 0.75 and 0.80.

Differing values of R_s and R_{sh} affect the I-V curve of a solar cell in rather dramatic ways; it is very important for a manufacturer to optimize these resistances as much as possible by doping the silicon with appropriate levels of boron and phosphorous (or whatever doping agents are being used). The effects of R_s and R_{sh} are shown in Figure B.4 [9].

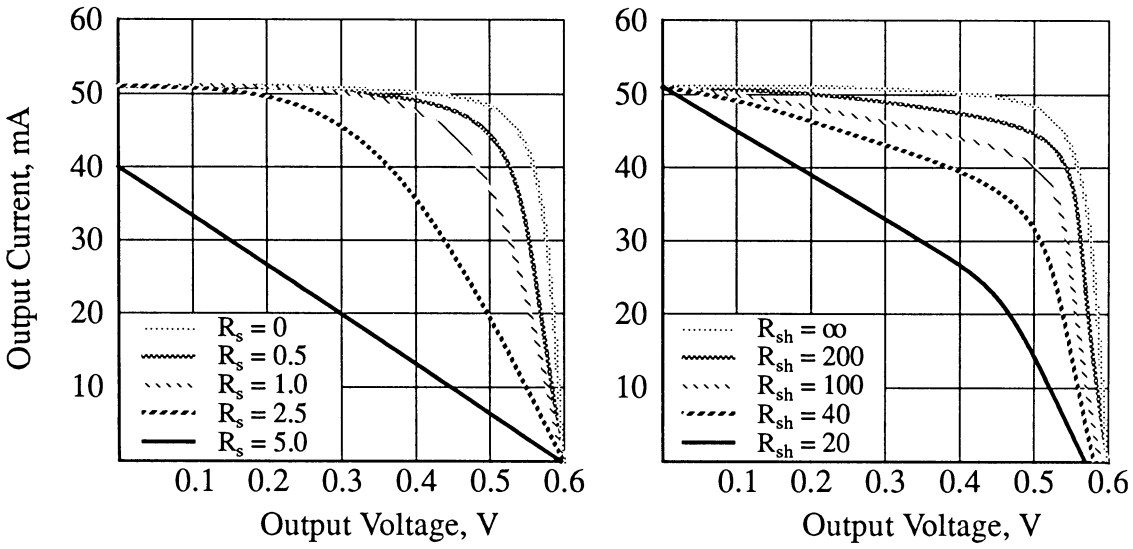

Figure B.4: Effects of series and shunt resistance on I-V curves

The two most important factors in the operational characteristics of a solar cell are those of solar intensity and cell temperature. Solar intensity, or insolation, will vary with the angle of incidence on the cell (adding an anti-reflection coating helps this somewhat), the season (distance from earth to sun), atmospheric losses, cloud cover, and shadows cast upon the array. Whatever the cause of the change, a reduction in insolation will translate the I-V curve along the cell's series resistance line, lowering I_{sc} more than V_{oc}. This effect is illustrated in Figure B.5. Note that the maximum power point voltage shifts very little with insolation changes.

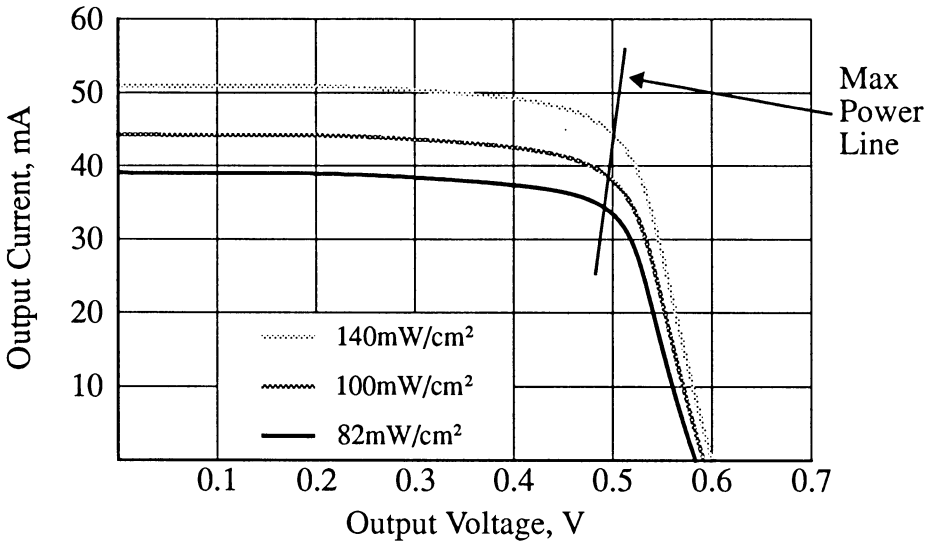

Figure B.5: I-V Curves for differing insolation

The effect of temperature on a solar cell's performance is a little more complex and is due predominantly to the change in cell conduction efficiency with temperature. Increasing a cell's temperature increases the short-circuit current slightly and significantly decreases the open-circuit voltage. The "knee" portion of the I-V curve also tends to become more rounded with increasing temperature. The net effect of increasing cell temperature is to lower maximum

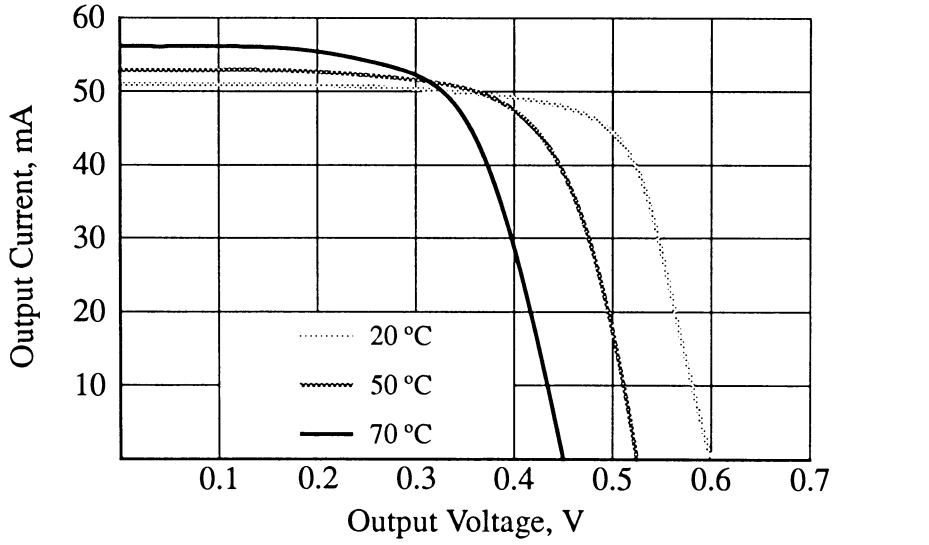

Figure B.6: Solar cell I-V variations with temperature

power. This is the dominant reason cells should never be allowed to dissipate energy as heat by forcing a non-ideal load upon the cell. In this way maximum point power tracking helps to keep the cells as cool as possible, further increasing the power produced by an array. The effects of temperature on cell performance are shown in Figure B.6 [10].

Arrays of Photovoltaics

Some rather important interactions occur when photovoltaic cells are cascaded together in series and parallel strings to increase the voltage and current produced by the array. It is virtually impossible to manufacture two completely identical solar cells using existing techniques, and as such a string of solar cells will contain some cells more efficient than others. Rather than increasing the power output of the array, however, the more efficient cells will simply cause the weaker cells to dissipate the additional power generated. The net result is that a series string of solar cells will be only as efficient as the least efficient cell. In addition, this cell (and any other weaker cells) will be dissipating additional power, increasing their temperature and the possibility of cell fracture. It is very important, therefore, to attempt to "match" cells prior to the construction of arrays.

Cells which crack pose additional power consumption problems. Cracked cells will most likely still conduct and produce some power, but will cause a large amount of power to be consumed in that series string. Most array implementations isolate separate series strings from each other by means of a blocking diode on each string. This concept is illustrated in Figure B.7. These diodes prevent a string from consuming power produced by any other string. This concept could be extended down to each cell, but this is rather costly and impractical under most circumstances. These diodes also provide additional help when the array is only partially shadowed. Without the diodes, strings in the shadow would consume power generated by strings in sunlight, heating these cells and reducing their efficiency when the shadow is removed. Thus, the minor power loss in these diodes is more than acceptable for the

performance increase they create in a large array of photovoltaic cells [10].

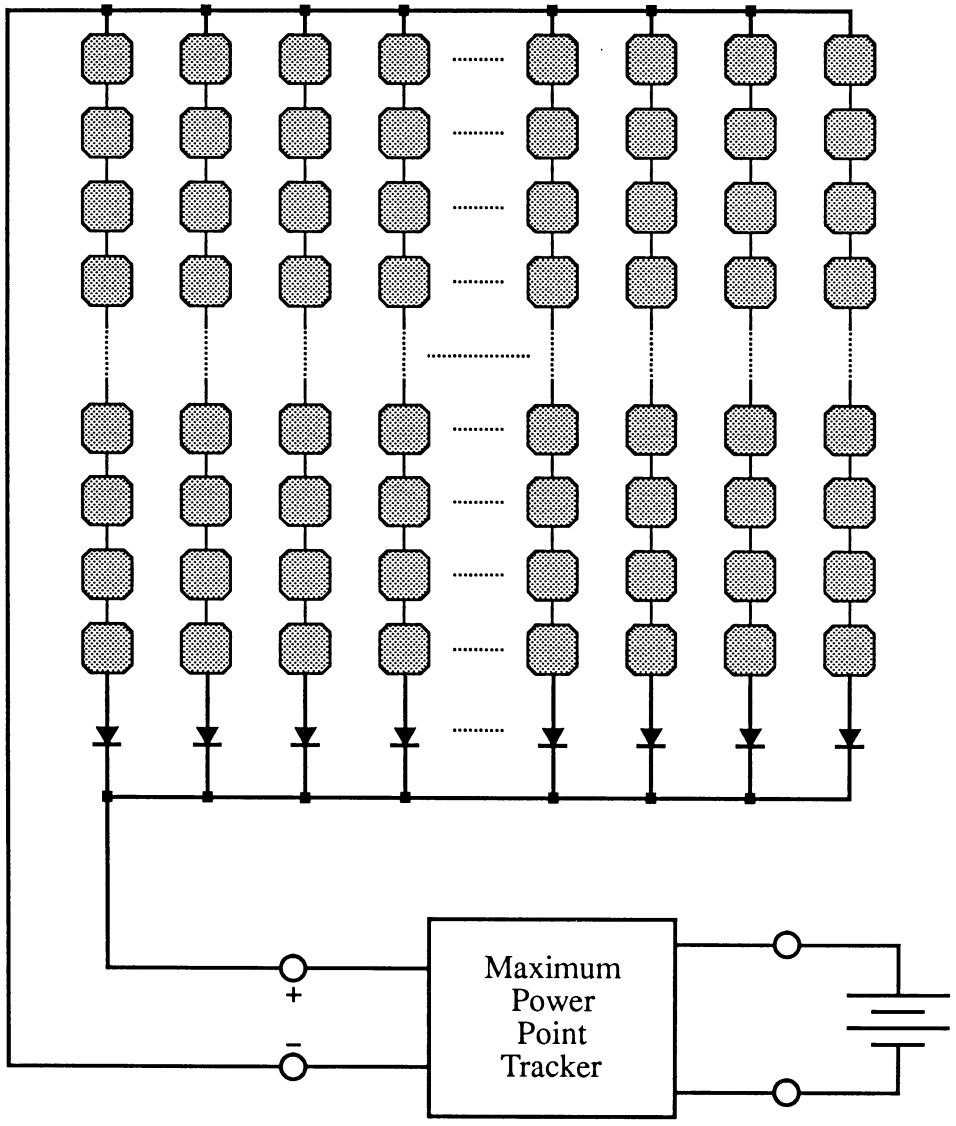

Figure B.7: Photovoltaic array with blocking diodes

APPENDIX C. DESCRIPTION OF THE 87C196KC MICROCONTROLLER

The following pages are reprinted courtesy of Intel Corporation from their *Embedded Controllers* data book.

intel®

ADVANCE INFORMATION

8XC196KC
16-BIT HIGH PERFORMANCE CHMOS MICROCONTROLLER

— 87C196KC—16 Kbytes of On-Chip EPROM
80C196KC—ROMless
83C196KC—16 Kbytes of On-Chip ROM

- 16 MHz Operation
- 232 Byte Register File
- 256 Bytes of Additional RAM
- Register-to-Register Architecture
- 28 Interrupt Sources/16 Vectors
- Peripheral Transaction Server
- 1.75 μs 16 x 16 Multiply (16 MHz)
- 3.0 μs 32/16 Divide (16 MHz)
- Powerdown and Idle Modes
- Five 8-Bit I/O Ports
- 16-Bit Watchdog Timer
- Dynamically Configurable 8-Bit or 16-Bit Buswidth
- Full Duplex Serial Port
- High Speed I/O Subsystem
- 16-Bit Timer
- 16-Bit Up/Down Counter with Capture
- 3 Pulse-Width-Modulated Outputs
- Four 16-Bit Software Timers
- 8- or 10-Bit A/D Converter with Sample/Hold
- $\overline{\text{HOLD}}$/$\overline{\text{HLDA}}$ Bus Protocol
- OTP One-Time Programmable Version

The 80C196KC 16-bit microcontroller is a high performance member of the MCS®-96 microcontroller family. The 80C196KC is an enhanced 80C196KB device with 488 bytes RAM, 16 MHz operation and an optional 16 Kbytes of ROM/EPROM. Intel's CHMOS IV process provides a high performance processor along with low power consumption.

The 87C196KC is an 80C196KC with 16 Kbytes on-chip EPROM. In this document, the 80C196KC will refer to all products unless otherwise stated.

Four high-speed capture inputs are provided to record times when events occur. Six high-speed outputs are available for pulse or waveform generation. The high-speed output can also generate four software timers or start an A/D conversion. Events can be based on the timer or up/down counter.

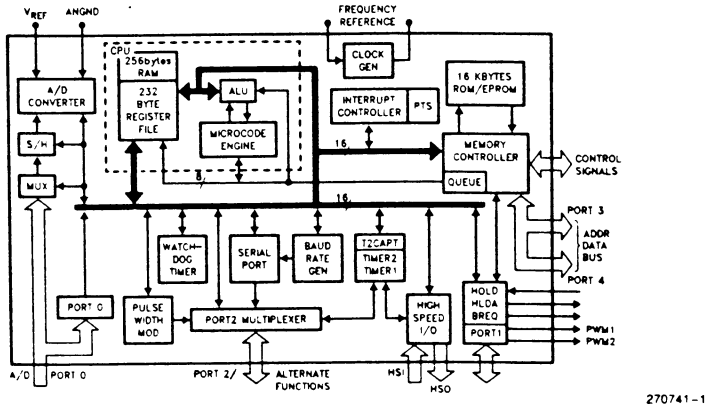

Figure 1. 80C196KC Block Diagram

MCS®-96 is a registered trademark of Intel Corporation

October 1989
Order Number: 270741-002

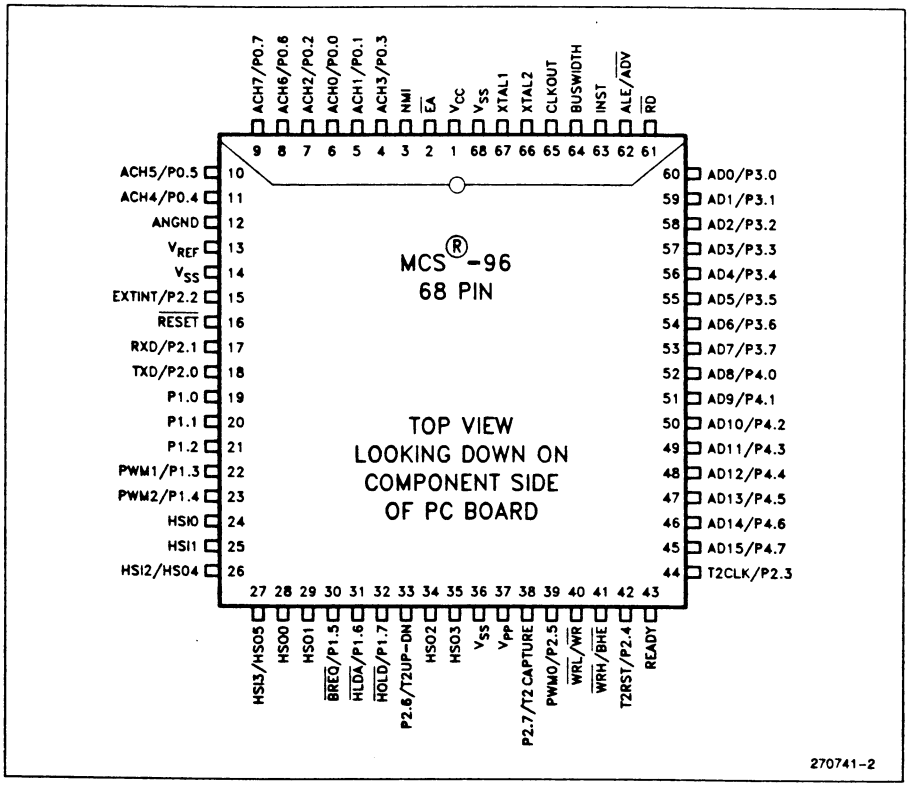

Figure 3. 68-Pin Cerquad and PLCC Package

Table 1. Prefix Identification

	CERQUAD	PLCC
80C196KC		N80C196KC
83C196KC		N83C196KC
87C196KC	CJ87C196KC	N87C196KC *

*OTP Version

(Reprinted courtesy of Intel Corporation.)

The MSB of this word also must be set. For example, with a 10 MHz Clock, the word at 4014H should be loaded with 8005H for the correct pulse width.

The 87C196KC then reads a word from external memory, and the Modified Quick-Pulse Programming Algorithm (described later) programs the corresponding EPROM location. Since the erased state of a byte is 0FFH, the Auto Programming Mode skips locations containing 0FFH. PVER will go low anytime there is a programming error. When all 16K have been programmed, $\overline{PACT}$ goes high.

In the Auto Programming Mode, CCR is loaded with the PCCB. The PCCB must correspond to the memory system of the programming setup, which is not necessarily the same as the memory system of the actual application.

V_{PP} and EA must be kept noise-free and must never go above 12.75V at any time.

PCCB Programming Mode

The PCCB (Programming Chip Configuration Byte) can be treated just like any other EPROM location. When in the Slave Programming Mode, the PCCB can be programmed at location 0218H. But the PCCB Programming Mode is a simple way to program the PCCB when no other locations need be programmed. Figure 17-5 shows a block diagram for using the PCCB Programming Mode.

With PMODE = 0DH and 0FFH on Port 4, the PCCB will be programmed with the data on Port 3 when a 0 is placed on PALE. After programming is complete, PVER is driven high if the PCCB was programmed correctly, and low if the programming failed. PALE can be pulsed to repeat programming.

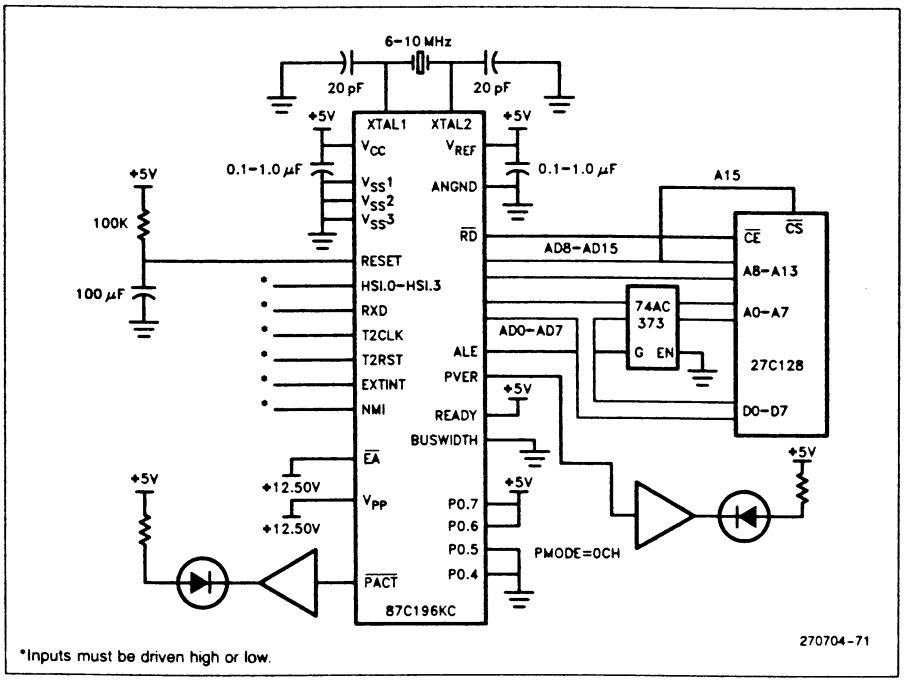

Figure 17-4. Auto Programming Mode

APPENDIX D. MICROCONTROLLER SOFTWARE LISTING

The following is a listing of the preliminary control software employed on the Intel 87C196KC microcontroller to perform maximum power point tracking of a moving solar array.

```
/*************************************************************/
/*                                                           */
/* Main initialization routines for 87C196KC microcontroller */
/*                                                           */
/*     Copyright (C) 1991     ISU Solar Car Project          */
/*                                                           */
/*              Written by:    Paul J. Dorweiler             */
/*                                                           */
/*************************************************************/

#pragma model(KB)

#include <80C196.h>
#include <stdlib.h>
#include "defs.h"

#define THRESHOLD (unsigned int)680       /* theoretically 5 watts */
#define BATTERY_MAX  0xFF0    /* max battery voltage before shutdown */

#pragma interrupt (software_timer = 5)
#pragma interrupt (ad_convert_done = 1)


void serial_init(void)        /* routine to initialize the serial port */
{
   baud_rate = 0x9b;
   baud_rate = 0x80;     /* 4800 baud @ 12 MHz */

   sp_con = 0x09;        /* serial port: mode 1, no parity, receiver on */
   sbuf = 0x0d;          /* send a CR to set the TI bit */
}


unsigned char putc(data)
char data;
{
   if ((sp_stat & 0x20) != 0x00)
    {
      sbuf = data;       /* send char */
      return(1);
    }
   else
      return(0);         /* transmitter busy */
}


char getc(void)
{
   if ((sp_stat & 0x40) != 0x00)/* data in receive buffer */
      return(sbuf);
   else
      return(0);
}
```

```
void putstr(string)
char *string;
{
   char *ptr;

   ptr = string;
   while (*ptr != '\0')
    {
      if (putc(*ptr))
      ptr++;
    }
}


/*********************************************************************/

short mode;       /* 0 = idle, 1 = slow, 2 = fast up, 3 = fast down */

unsigned char ad_channel;
short timer_loop;

short pwm_value;
short pwm_dir;

unsigned char packet[10];
unsigned char pack_flag, pack_loop;

unsigned char varray, vbattery, iarray;
unsigned int power[4];


void software_timer(void)
{
   unsigned char temp;

   hso_command = 0x18;
   hso_time = timer1 + 30075; /* interrupt 40 ms from now */

   if (mode == 1)             /* slow mode */
    {
      timer_loop++;
      if (timer_loop >= 5)
       {
         ad_command = 8;      /* start a/d channel 0 */
         timer_loop = 0;
       }
    }
   else                       /* fast mode */
      ad_command = 8;         /* start a/d channel 0 */

   temp = ioport1;            /* toggle bit 0 for outside world */
   if ((temp & 0x01) == 0x01)
      temp &= 0xfe;
   else
      temp |= 0x01;
   ioport1 = temp;
}
```

```
void creat_packet(void)        /* routine to make a packet for xmission */
{
   if (pack_flag != 1)         /* if last packet not complete, don't make */
    {                          /* a new one to send yet....   */
      packet[0] = 0x0a;
      packet[1] = 0x0d;

      packet[2] = 0xc0 + ((varray >> 4) & 0x0f);
      packet[3] = 0x40 + (varray & 0x0f);

      packet[4] = 0xd0 + ((iarray >> 4) & 0x0f);
      packet[5] = 0x50 + (iarray & 0x0f);

      packet[6] = 0xe0 + ((vbattery >> 4) & 0x0f);
      packet[7] = 0x60 + (vbattery & 0x0f);

      packet[8] = 0xf0 + ((pwm_value >> 4) & 0x0f);
      packet[9] = 0x70 + (pwm_value & 0x0f);

      pack_loop = 0;           /* just in case */
      pack_flag = 1;           /* new packet to xmit */
    }
}



void ad_convert_done(void)
{
   unsigned char value;

   value = ad_result_hi;
   if (((ad_result_lo & 0x80) != 0x00) && (value != 0xff))
      value++;                 /* round up if bit 1 set */

   if (ad_channel == 0)
      varray = value;
   else if (ad_channel == 1)
      iarray = value;
   else if (ad_channel == 2)
      vbattery = value;

   if (ad_channel < 2)
    {
      ad_channel++;
      ad_command = (8 + ad_channel);      /* start next conversion */
    }
   else
    {                                     /* adjust pwm */
      power[2] = power[1];
      power[1] = power[0];
      power[0] = (unsigned int)(varray * iarray);

      if (mode == 1)               /* slow mode */
       {
         if ((power[0] < power[1]) && (power[0] < power[2]))
            pwm_dir *= -1;         /* change direction */
         if (((power[0] < power[2]) && ((power[2] - power[0]) > THRESHOLD))
            || ((power[0]> power[2]) && ((power[0] - power[2]) > THRESHOLD)))
             {
               mode = 2;           /* go to fast mode, increasing */
               pwm_dir *= 2;       /* boost gain right away */
```

```
            timer_loop = 0;       /* reset for return to slow mode */
          }
        }
      else if (mode == 2)           /* fast mode, increasing pwm_dir */
          {
            if (abs(pwm_dir) < 8)       /* kick up adjust rate & cap it
               pwm_dir *= 2;
            if ((power[0] < power[1]) && (power[0] < power[2]))
                pwm_dir *= -1;           /* change direction */
            if ((power[0] < power[1]) && (power[1] > power[2]))
             {                           /* crossed max power */
               mode = 3;
               if (abs(pwm_dir) > 1)
                  pwm_dir /= 2;         /* slow down right away */
             }
          }
      else if (mode == 3)/* fast mode, decreasing pwm_dir */
          {
            if (abs(pwm_dir) > 1)       /* shrink adjust rate */
               pwm_dir /= 2;
            if ((power[0] < power[1]) && (power[0] < power[2]))
               pwm_dir *= -1;           /* change direction */
            if (abs(pwm_dir) == 1)
             {
               mode = 1;                /* go back to slow mode */
               timer_loop = 0;
             }
          }

      pwm_value += pwm_dir;
      if (pwm_value > 254)    /* too big */
       {
         pwm_dir *= -1;
         pwm_value = 254;
       }
      if (pwm_value < 1)      /* too small */
       {
         pwm_dir *= -1;
         pwm_value = 1;
       }

/*    if ((varray < vbattery) || (vbattery > BATTERY_MAX))
       {
         mode = 0;* error - go to idle mode *
         pwm_value = 255;* shut off tracker *
         pwm_dir = 1;* reset for re-entry *
       }                                                        */

      if (mode == 1)
         creat_packet();            /* spit out current data */

      pwm_control = (unsigned char)pwm_value;   /* set new pwm value */
      ad_channel = 0;
   }
}



/*************************************************************************
```

```
main(void)
{
   short loop;
   unsigned char retval;

   /* initialize everything in the microcontroller to correct states *

   int_mask = 0x22;     /* timer and A/D interrupts */
   imask1 = 0x00;
   int_pending = 0;

   serial_init();

   ioport3 = (unsigned char *)0x1ffe;
   ioport4 = (unsigned char *)0x1fff;

   timer_loop = 0;
   ad_channel = 0;
   pwm_dir = -1;              /* start with increasing duty cycle */
   pwm_value = 128;
   for (loop=0; loop<4; loop++)
      power[loop] = 0;
   mode = 1;                  /* start in slow mode */

   pack_flag = 0;             /* no packet in buffer */
   pack_loop = 0;

   hso_command = 0x18;
   hso_time = timer1 + 30075; /* 40 ms timer interrupt */
   pwm_control = 128;         /* start at 50% duty cycle */

   enable();                  /* start up interrupts */

   while (1)
   {
      if (mode == 0)
       {
         disable();
         /* sample array voltage */
         /* sample battery voltage */
         if (varray < vbattery)
          {
            pwm_control = pwm_value = 128;
            mode = 1;
            timer_loop = 0;
            int_mask = 0x22;
            hso_command = 0x18;
            hso_time = timer1 + 30075;    /* 40 ms interrupt */
            enable();
          }
       }
      else                                /* not in idle mode */
       {
         if (pack_flag)                   /* packet to xmit */
          {
            retval = putc(packet[pack_loop]);
            if (retval != 0)
             pack_loop++;
            if (pack_loop > 9)
             {
               pack_loop = 0;
```

```
                pack_flag = 0;
              }
            }
           for (loop=0; loop<20; loop++)
              ;
        }
    }

}
```