A DC-DC CONVERTER SUITABLE FOR CONTROLLING A PHOTOVOLTAIC POWERED PUMPING SYSTEM

William A. Passmore

B.A.Sc. (Electrical Engineering) University of British Columbia

A THESIS SUBMITTED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF APPLIED SCIENCE

in

THE FACULTY OF GRADUATE STUDIES
DEPARTMENT OF ELECTRICAL ENGINEERING

We accept this thesis as conforming
to the required standard

THE UNIVERSITY OF BRITISH COLUMBIA
March 1989

Department of Electrical Engineering

The University of British Columbia
Vancouver, Canada

Date March 7, 1989

Abstract

A photovoltaic powered pumping system offers an attractive means of supplying fresh water in remote areas not serviced by a utility grid. In order to extract the maximum amount of energy from the solar panels, it is necessary to match the characteristics of the photovoltaic array to the DC motor which drives a pump. A one quadrant DC-DC converter is capable of adjusting the effective load impedance for maximum power transfer under most lighting conditions.

Three styles of DC-DC converters used to control the pumping system are described and compared. The voltage tracking style of converter fixes the array voltage at a level considered optimum. The power tracking converter measures, and attempts to maximize, the output power of the photovoltaic array. The microprocessor based power tracking, voltage tracking converter toggles between the two methods of control. Experimental results are included.

Table of Contents

List of Tables

List of Figures

Acknowledgement

The author would like to express his appreciation to his supervisor, Dr. W. G. Dunford, for guidance and assistance during this project, and to Dr. M. D. Wvong, for assuming the role of acting supervisor during the final stages of this project.

The author also wishes to express his gratitude to his father, Mr. C. E. Passmore, for his help and assistance, and to his wife, Carol, for her understanding and support. Gratitude is also expressed to colleagues Nick Jaeger, Ben Giudici, and C.E. Sudhakar for their help.

Appreciation is also expressed to the staff of Electrical Engineering, UBC, for their helpful assistance, particularly to Alan Prince and Lloyd Welder. Appreciation is also expressed to Guang Liu for making available data on the photovoltaic array.

Chapter 1

Introduction

A photovoltaic-powered pumping system offers an attractive means of supplying water from medium head wells in locations not serviced by a utility grid. The capital cost of a system is higher than a diesel-powered pumping station but the photovoltaic system utilizes a free and inexhaustible energy source, requires very little maintenance, and can be completely automated. Therefore the long term cost and reliability of the system compares favorably with that of a diesel powered station [1].

The regions that could benefit most from such a pumping system can generally least afford it. However, industrialized countries such as Canada are willing to fund such projects. They recognize the benefits that quantities of clean drinking water would allow in the areas serviced by these pumping stations. The Canadian International Development Agency, CIDA, is currently installing test pumping stations in Morroco and is considering many more sites in various locations throughout the world. Dry tropical countries offer particularly attractive sites because of their great need of water, consistently long hours of sunshine, and lack of alternate energy sources. However, there are areas within Canada, such as the prairies or the interior of B.C. which could benefit from such a pumping system. During dry spells, water is urgently needed to feed livestock. Where ground water is present and mains power is not, a photovoltaic-powered pumping system could supply the needed water.

1.1 System components

The photovoltaic-powered pumping system consists of an array of solar panels, a power converter, a motor, and a pump. The photovoltaic array produces a DC voltage and current, therefore, a DC motor and a DC to DC converter are the logical component choices. Some systems do employ an AC motor requiring a DC to AC converter, but for a small or medium sized system the inherent extra complexity and reduced efficiency rule it out.

A permanent-magnet DC motor is preferable to a separately-excited or compound-wound machine. The increased efficiency and reduced heating of the permanent magnet machine will justify the higher cost. A 1.1kW permanent-magnet BROT motor equipped with samarium cobalt magnets and interpole windings is used for many of the tests described in this report. This motor achieved measured efficiencies as high as 87% and has a long brush life to minimize maintenance.

The Mono progressive cavity pump is the pump of choice for this application. It is efficient over a large range of speeds and well depths [2] and is well known at many of the target sites. The pump itself is submersed at the bottom of the well and driven by a shaft running up the length of the well and connected to a motor at the top. This gives the dual advantage of having a dry, easily accessible electric motor and a submersed pump, capable of efficiently delivering water from depths of up to 150m.

1.1.1 Photovoltaic Array

The photovoltaic array is the most expensive system component. It is therefore wise to maximize its effectiveness by extracting the maximum amount of energy from its' panels.

The voltage-current curves displayed in Figure 1.1 were obtained from measurements performed on the photovoltaic array located on the roof of the Hector MacLeod building at UBC. This array is made up of two parallel strings of five series panels rated at 16V and 35w each. The curves show that the current delivered by the panels increases significantly with the insolation level as compared to the open circuit voltage which increases only a small amount. Consistent with theory [3], the open circuit voltage decreases as the temperature increases, creating intersecting curves.

A curve displaying how the power output of the array varies with the array voltage at a particular insolation level is shown in Figure 1.2. Its jagged appearance is due to the limited eight-bit resolution of the measurement equipment. Nevertheless the general shape of the curve is clear. The peak of the power curve occurs on the knee of the corresponding voltage-current curve. The locus of maximum power points for different insolation levels is shown in Figure 1.1 and is approximately a constant voltage, variable current curve. It is apparent that a reasonable approximation to a maximum power tracking converter would be a voltage tracking converter. Such a converter would fix the array voltage at a level considered to be optimum.

The voltage-current curve tends to flatten out as the panels age, shifting the optimum operating voltage. Add to this effect the shift in the open circuit voltage with temperature and a case could be made for using a converter which can adapt to changing conditions. More practically, an adaptive converter would be able to automatically locate the best operating point regardless of the array configuration. The need for careful on-site measurements and adjustments would then be eliminated.

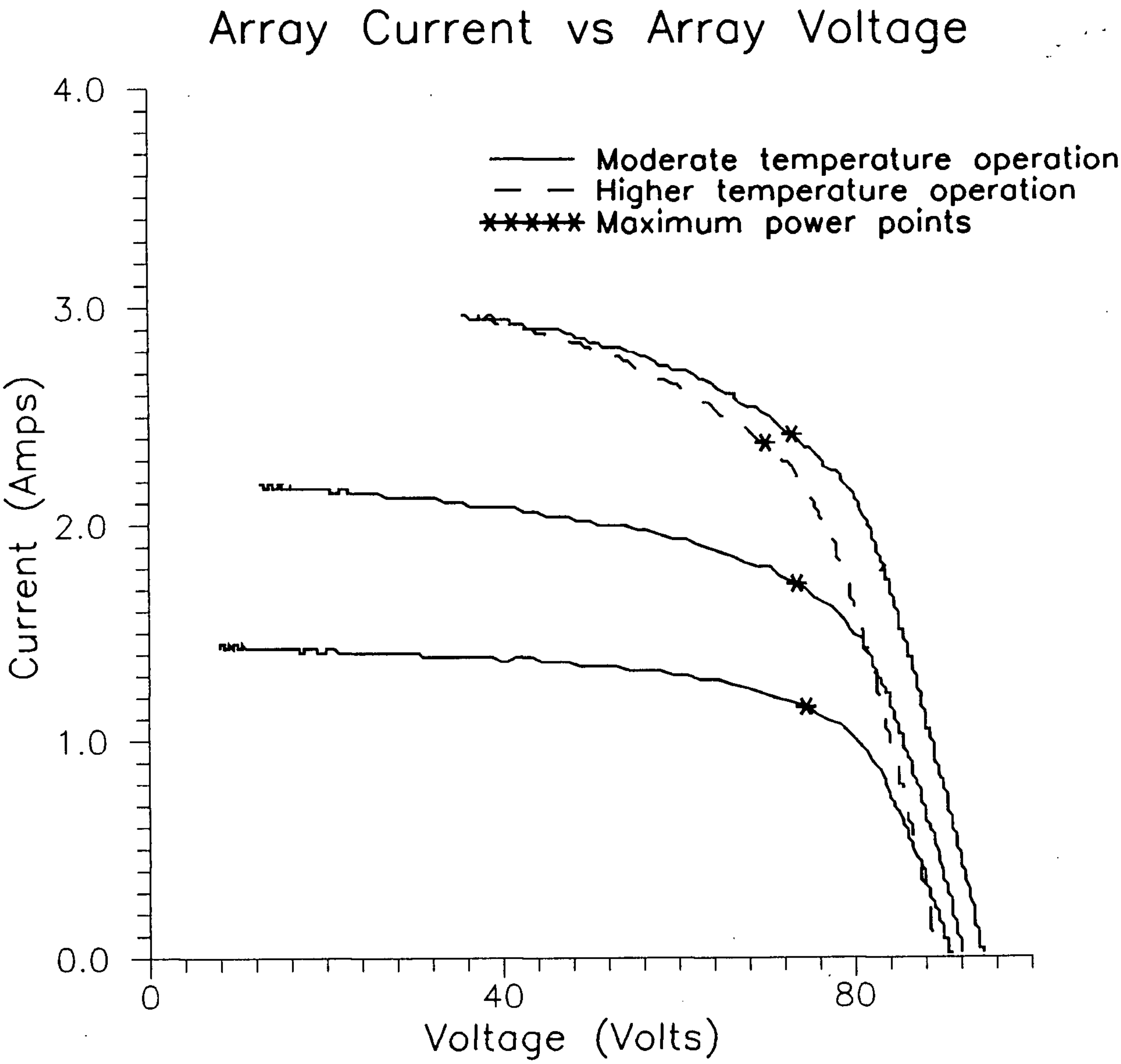

Figure 1.1: Voltage vs Current Curves of a Photovoltaic Array

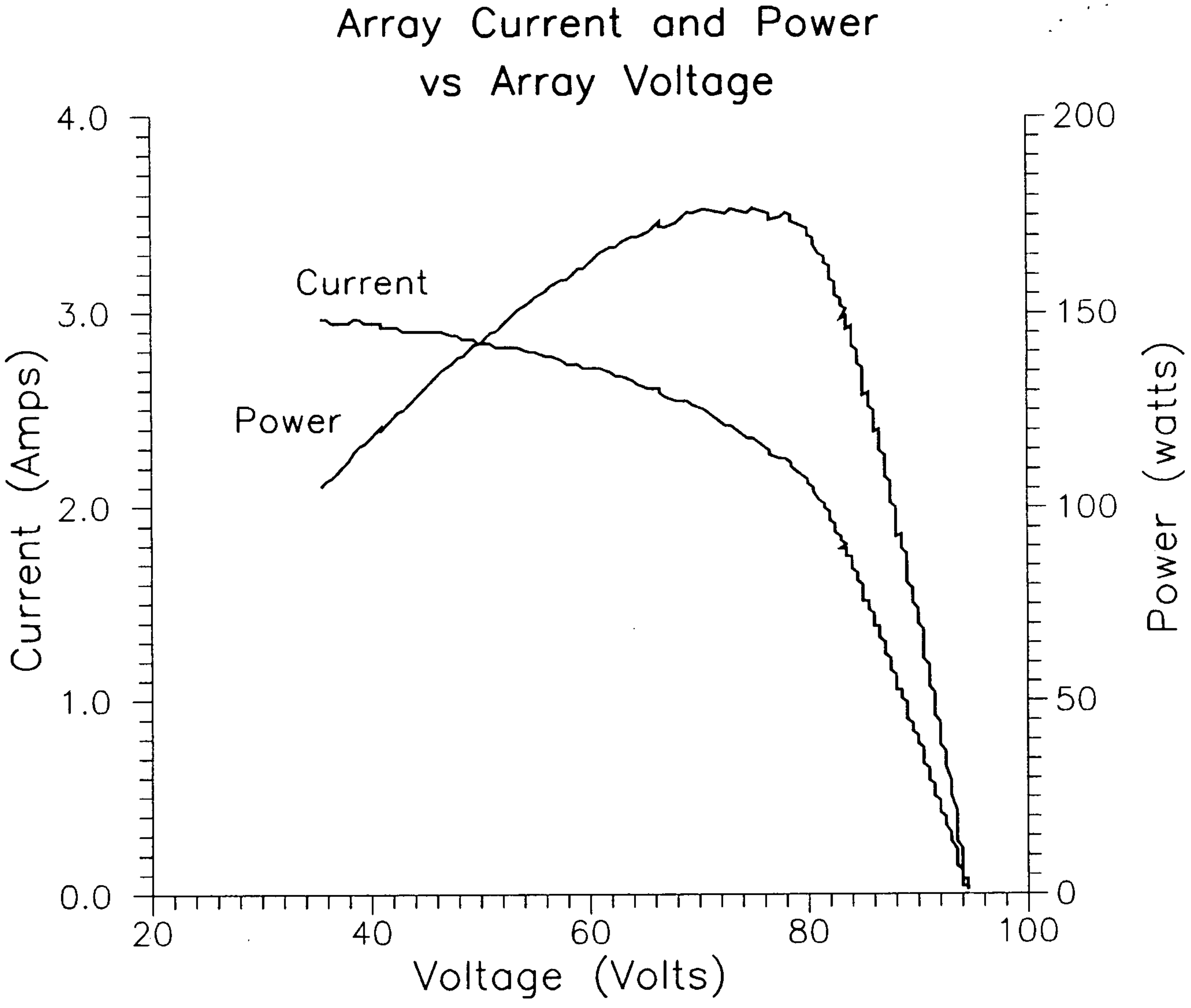

Figure 1.2: Current and Voltage vs Power at a Set Insolation Level

1.2 Thesis

This thesis covers the design and testing of three different styles of converters. The basic design of the power circuit is common to all three converters and is described in Chapter 2. The device specifications, thermal and RMS ripple current calculations, and heatsink sizing are presented in this chapter.

The voltage tracking style of converter is described in Chapter 3. It was built as a prototype for Optima Design Ltd. to be considered for use in Morroco. The control is simple, yet it is a rugged reliable device. The array voltage is set on site by means of a potentiometer which is accessible through the front of the converter.

Chapter 4 discusses an analogue maximum power tracking style of converter. It continuously searches for the maximum power point of the array and it does not require any field adjustments, however, the logic circuit is more complicated.

A hybrid maximum power tracking, voltage tracking style of converter is presented in Chapter 5. In the maximum power tracking mode it searches for the optimum operating point. Once this point has been found the array voltage is maintained at its optimum level in the voltage tracking mode. Periodically the power tracking mode is reentered to make minor adjustments to the operating voltage. The MC68HC11 makes it possible to use this more sophisticated algorithm without increasing the circuit complexity. It is able to perform the A/D conversions, execute the control algorithms, keep track of timing, and output a series of pulse width modulated output waveforms. Only a few external chips are required to assist the MC68HC11 in controlling the converter.

Chapter 2

Power Converter

2.1 Component Matching

It is necessary to match the motor, pump, and photovoltaic array characteristics. Matching the D.C. motor to the Mono progressive cavity pump is relatively straightforward. Their torque-speed characteristics must be matched with possibly the aid of a mechanical gearing system. Matching the photovoltaic array to the DC motor-pump combination is more challenging. A large starting current of at least twice the rated value of the machine may be necessary to overcome the static friction of the Mono pump. Once rotating, the motor will draw an almost constant current over most of its speed range for a fixed head.

The DC-DC converter illustrated in Figure 2.3, is well suited to matching the photovoltaic and motor characteristics. It is capable of increasing the impedance of the load as seen by the source so that it is possible to extract the maximum power from the source for most lighting conditions.

This power conditioning is achieved by adjusting the ratio of the on-time to off-time, i.e. the duty cycle, of the power mosfets. During their *on* state the full source voltage, V_S, appears across the load while the load current, I_L, is supplied by the source and the filter capacitor. During the *off* state the load voltage, v_L, drops to zero while the current, maintained by the load inductance, flows through the freewheeling diode. The average load voltage, V_L, is therefore:

$$V_L = \frac{t_{ON}}{T} \cdot V_S = d \cdot V_S \tag{2.1}$$

where t_{ON} is the *on* time, T is the period and d is the duty cycle. If the chopping frequency and load inductance are high enough the load current remains almost constant with a small ripple component. Figure 2.4 displays the voltage and current waveforms of the load, source, and filter capacitor. In the steady state the average capacitor current is zero, which implies:

$$I_S = I_L \cdot \frac{t_{ON}}{T} = I_L \cdot d \tag{2.2}$$

The resistance of the load as seen by the source is:

$$\begin{aligned} R_{IN} &= \frac{V_S}{I_S} = \frac{V_S}{I_L \cdot d} \\ &= \frac{V_L}{I_L \cdot d^2} = R_L \cdot \frac{1}{d^2} \end{aligned} \tag{2.3}$$

The chopper is in effect a transformer with turns ratio equal to the duty cycle for the purpose of transforming the voltage, current and effective resistance from one side to another. The chopper can therefore increase the load resistance by the factor of d^{-2} to capture all maximum power points lying above the base load line.

2.2 Component Ratings

The reliability and efficiency of the system will depend to a large degree upon the choice of suitable components. Reliability is essential, as in most practical applications the system will be located in remote areas. Efficiency is important, as the photovoltaic panels are expensive. An increase in converter efficiency will usually result in an even greater increase in overall efficiency, as the pump and motor are generally more efficient at higher speeds.

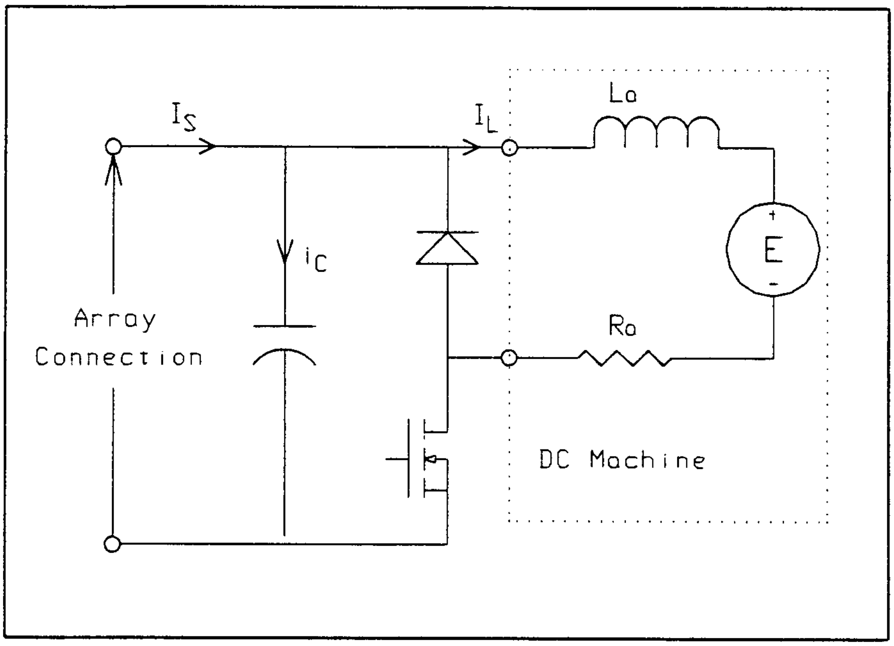

Figure 2.3: DC to DC Converter

2.2.1 Filter Capacitor

The DC-DC input capacitor is essential to fix the array voltage. It accepts current from the array during the *off* state of the mosfets and delivers current to the load while the mosfets are switched on. The value of capacitance will be determined by the allowable input voltage ripple.

Consider an example where a $0.5V$ ripple is acceptable with the array delivering $5A$ of current. The worst case will occur when the switch is either open or closed for almost all of the switching period. If the switching frequency is $20kHz$, then a capacitor of at least

$$C = \frac{q}{V} = \frac{50\mu s \times 5A}{0.5V} = 500\mu F \,, \tag{2.4}$$

would be required. The capacitor chosen must satisfy the RMS current requirements and withstand the peak open-circuit array voltage. From the current waveforms of Figure 2.4 and assuming a constant load current I_L the capacitor RMS current can be

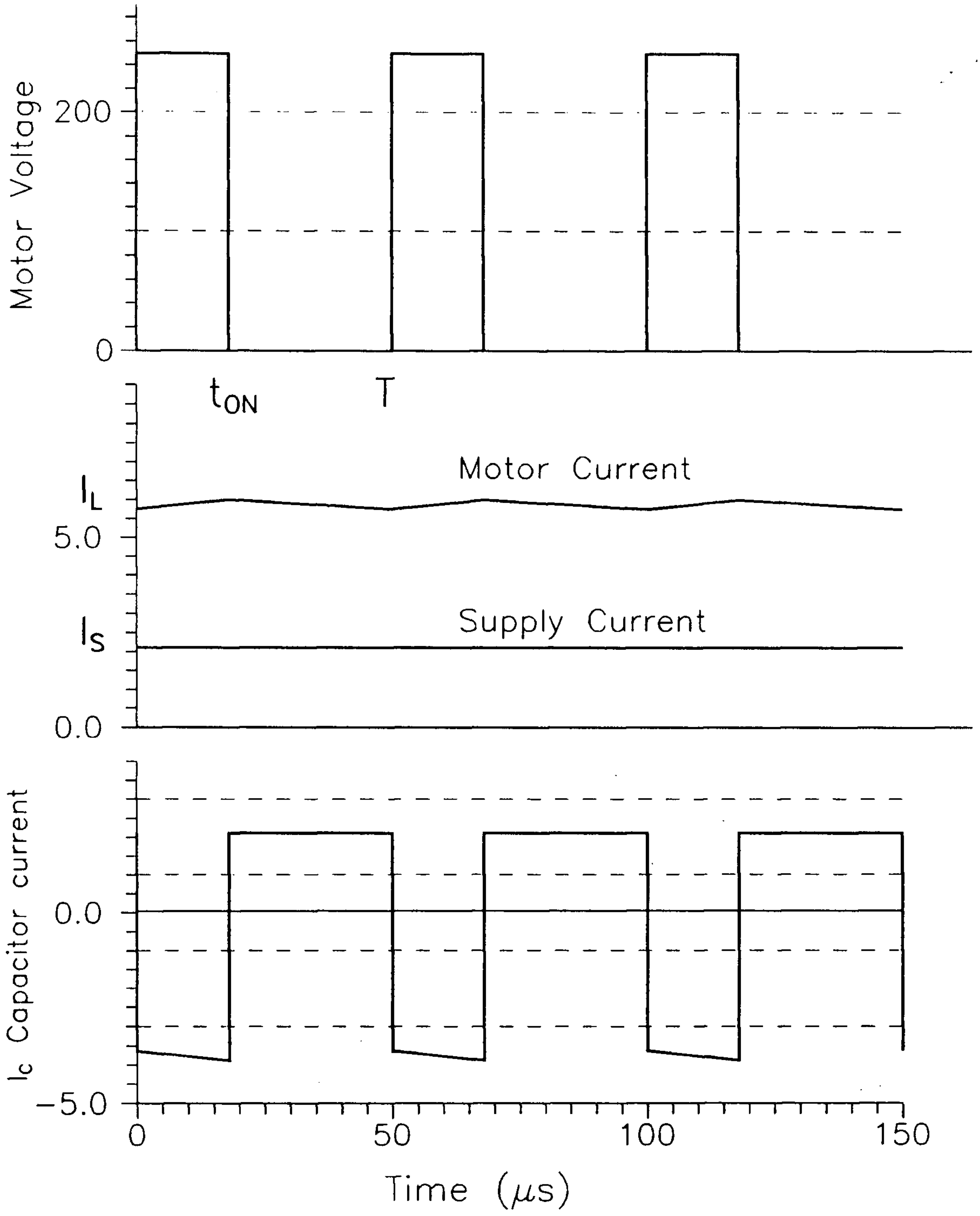

Figure 2.4: Converter Voltage and Current Waveforms

calculated:

$$\begin{aligned} I_{rms} &= \sqrt{\frac{1}{T}\left[\int_0^{t_{ON}} (I_S - I_L)^2 dt + \int_{t_{ON}}^{T} I_S^2 dt\right]} \\ &= \sqrt{\frac{1}{T}\left[I_L^2 \cdot t_{ON} + I_S^2 \cdot T - 2 I_S I_L \cdot t_{ON}\right]} \,. \end{aligned} \tag{2.5}$$

Substituting equation 2.2 for the source current yields:

$$I_{rms} = \sqrt{I_L^2 \frac{t_{ON}}{T}\left[1 - \frac{t_{ON}}{T}\right]} \,. \tag{2.6}$$

Which has a maximum when:

$$t_{ON} = \frac{T}{2} \tag{2.7}$$

Therefore:

$$I_{rms}(max) = \frac{I_L}{2} \tag{2.8}$$

The maximum continuous load current is usually a known quantity, so that the maximum continuous RMS capacitor current rating can be easily calculated.

2.2.2 Power Mosfets

The logical choice of a switching device for the converter is the power mosfet. They are easily driven by either CMOS or TTL logic chips, switch rapidly, and do not require a commutation circuit. They are robust and are readily available at the current and voltage levels typically encountered in this application. They also can easily be driven at a high enough frequency to ensure a continuous motor current.

Although the power mosfet is rugged, the designer must ensure that it is operated within its specified ratings. The ratings of concern are the maximum gate-to-source and drain-to-source voltage levels and the maximum junction operating temperature.

In this application it is not difficult to adhere to the maximum voltage ratings. A zener diode inserted between the gate and source prevents the gate voltage from rising beyond its limit. The drain-to-source voltage rating is maintained by choosing a device rated high enough to withstand the open-circuit voltage together with the voltage spike generated as the device is switched off. This spike is minimized by paying careful attention to circuit layout and by the use of a free-wheeling diode. For further protection a zener diode may be inserted between the drain and source, or third generation devices used that have a built-in zener diode.

The current ratings listed in the device specifications are misleading as they assume a junction operating temperature of 25^oC which is impractical. Realistically current ratings of the device are derived from the maximum operating junction temperature of the mosfet. It is safe to force current through the device as long as the junction temperature remains below 150^oC. If the device is operated above 150^oC premature failure can occur.

To arrive at an operating junction temperature for a specified current the *on* resistance must be known together with the junction to case, case to heatsink, and heatsink to ambient thermal resistances.

As an example consider the IRF730 mosfet chosen for the voltage tracking converter described in Chapter 3. Six parallel devices are used and must be capable of supplying a continuous load current of 12A and a peak current of 22A. The maximum ambient temperature is assumed to be 55^oC and a heatsink will be chosen to operate at a maximum of 30^oC above ambient, under rated conditions. Table 2.1 summarizes the operating conditions and device ratings.

Consider the mosfets turned fully on and delivering 12A of load current. Assume the heatsink temperature is 85°C and the junction temperature is 115°C. The calculated junction operating temperature is:

$$T_j \;=\; T_H + d * (\frac{I_L}{n})^2 * R_{ON(t)} * (T_{JC} + T_{CH})$$

Table 2.1: IRF730 Device Specifications:

Description	Symbol	Value
Ambient temperature	T_A	55^oC
Heatsink temperature	T_H	85^oC
Junction to case thermal resistance	T_{JC}	$1.5^oC/W$
Case to heatsink thermal resistance with an electrically insulating silica pad	T_{CH}	$1.7^oC/W$
On resistance at a junction temperature. of 25^oC	$R_{ON}(25^o)$	1.0Ω
On resistance at a junction temperature of 115^oC	$R_{ON}(115^o)$	1.9Ω

$$\begin{aligned} &= 85^oC + (2.0A)^2 * 1.9\Omega * (1.5^oC/W + 1.7^oC/W) \\ &= 109.3^oC \end{aligned} \tag{2.9}$$

where n is the number of parallel mosfets.

This temperature is below the assumed value of 115°C used to determine the *on* resistance of the mosfets. It is also well within the safe operating temperature of 150°C. The power that must be dissipated by the heatsink due to losses within the mosfets for the above operating conditions is:

$$P_{loss} = 6 * (2.0A)^2 * 1.9\Omega = 45.6W \tag{2.10}$$

Consider now the converter providing peak load current, 24A, at a duty cycle of 25%. The same assumptions are made as above. The junction operating temperature would then be:

$$\begin{aligned} T_j &= 85^oC + 0.25 * (4.0A)^2 * 1.9\Omega * (1.5^oC/W + 1.7^oC/W) \\ &= 109.3^oC \end{aligned} \tag{2.11}$$

with a loss of:

$$P_{loss} = 0.25 * 6 * (4.0A)^2 * 1.9\Omega = 45.6W \tag{2.12}$$

The mosfets are operating at a safe temperature. The array current would be 6A for these conditions and it is unlikely the duty cycle would exceed 25% as most array configurations would not be able to deliver such a large current. The load current is prevented from exceeding 24A by the current limit. Note that for these conditions there are also losses in the diode to consider.

2.2.3 Diode

A freewheeling diode is essential when switching inductive loads such as a DC motor. Even a resistive load will usually contain enough stray inductance to produce an excessive voltage spike as the mosfets are switched off if the freewheeling diode is removed from the circuit.

A fast or ultra fast recovery diode is necessary to cope with the fast switching speed of the mosfets. If a slow diode is used a large reverse recovery current spike will be generated as the converter is switched on, which may damage the mosfets. This current spike also creates noise which may interfere with the operation of the logic circuits. Even when a fast recovery diode is used, it is often wise to slow down the mosfets turn-on time in order to reduce noise levels.

The converter will need to supply maximum current at a low duty cycle when the DC machine is being turned on to overcome starting torque. The diode, therefore, must be able to handle the maximum current on a continuous basis. For example, for the converter described in Chapter 3, a current rating of 30A should be adequate. The diode should also be rated at 400V to be compatible with the mosfets. The MUR3040PT ultra fast recovery diode in a T0-218AC package is the device of choice. This device is made up of two parallel 15A, 400V diodes in the same package, with a recovery time of 50ns. The diodes are matched and thermally coupled, enabling parallel operation of the two diodes.

If the diode is operated under the same conditions used in equation (2.11) the power loss and operating junction temperature would be:

$$\begin{aligned} P_{loss} &= (1-d) * I_L * V_{ON} \\ &= 0.75 * 24A * 0.75V \ = \ 13.5W \qquad (2.13) \\ T_J &= T_H + \frac{Ploss}{n} * (T_{CH} + T_{JC}) \\ &= 85^oC + \frac{13.5W}{2} * (1.7^oC/W + 1.5^oC/W) \\ &= 106.6^oC \qquad (2.14) \end{aligned}$$

2.2.4 Power Supply

The logic circuit requires its own power supply. The most practical way to derive this supply is directly from the photovoltaic array. The array voltage will vary at different installations and under different operating conditions while the logic voltage must remain constant.

The current requirements of the logic supply are modest, approximately 100mA, so that a simple supply is adequate. The simplest supply would consist of a resistor charging an output filter capacitor whose voltage is set by a reference zener diode. At higher input voltages excess current would be drained through the zener diode. The power supply loss increases with the square of the input voltage over its operating range. This loss is high but may be acceptable for certain applications.

The supply can be made more efficient by replacing the dropping resistor with a mosfet. The effective resistance of the mosfet is automatically adjusted to maintain a constant output voltage as the input voltage varies. The output voltage level is maintained by the mosfet three volts below the value set by a reference zener diode attached to the gate. Only as much current as is necessary to maintain the source voltage is supplied resulting in a net loss which is directly proportional to the input voltage. This form of supply is used by the converter described in Chapter 3 and is shown in detail in Figure 3.5. The power loss in the logic supply when the array is operating at 250V is:

$$P_{loss} = 250V \times 100mA = 25W \tag{2.15}$$

A more sophisticated and efficient switched mode power supply could be used. However the extra circuit complexity would not justify the power savings achieved as only a small logic current is required. The added complexity would reduce the overall circuit reliability as well as increase costs.

Once a steady output voltage is established a single, dual or triple supply can be derived. For example the analogue maximum power tracking circuit of Chapter 4 requires a 15V, 5V and −7V supply. A linear voltage regulator is used to provide the 15V and 5V supplies from the 18V supply. The negative supply is derived with the aid of a switching regulator and a few external components.

2.2.5 Heatsink

Proper sizing of the heatsink is essential for reliable operation. The heatsink must be able to maintain the temperature of the active devices within their safe operating region over a large range of ambient temperatures. To calculate the heatsink size the power being dissipated must be known along with the temperature rise above ambient which can be tolerated. It is assumed the device is operated in the shade and with its fins positioned vertically.

From the power losses calculated in equations (2.10) and (2.15) it is determined the heatsink used in the sample converter may have to dissipate up to 70W under normal circumstances. From equations (2.11), (2.12) and (2.15) it can be seen that the losses under peak current conditions can reach as high as 84W. However, this is abnormal and peak current should only be delivered on a temporary basis. If peak current is supplied for a prolonged period, it could be expected the heatsink would warm up causing a thermal cutout to shut down the converter.

It was decided while making thermal calculations that a 30°C temperature rise between the heatsink and the environment could be tolerated. This implies a heatsink with a thermal coefficient of 0.43°C/W is required. There are many shapes and sizes of heatsinks available. A suitable heatsink for this application is heatsink #2001 from AHAM TOR INC., California.

Chapter 3

Voltage Tracking Converter

3.1 Specifications

A DC-DC converter is to be designed for use with photovoltaic arrays of up to two kilowatts. The converter attempts to maximize the power output of the array by optimizing the operating voltage.

The converter is built to be both efficient and reliable. The power mosfets are derated to ensure a long life and reduce overall losses. A high chopping frequency minimizes the ripple voltage, ripple current and harmonic motor losses. An over-temperature cut out is built into the converter to turn it off if the motor remains stalled for a prolonged period of time. Table 3.1 summarizes the device specifications.

3.2 Circuit Description

The voltage tracking converter approximates a maximum power tracking converter by fixing the array voltage at a point considered optimum. This design is based on a circuit developed and tested by Dr. W.G. Dunford and Dr. P. Ward. The logic adjusts the duty cycle of the power mosfets according to the value of the array voltage. If the voltage is too high the duty cycle is increased to bring the voltage down and vice versa. The circuit diagram is detailed in Figure 3.5.

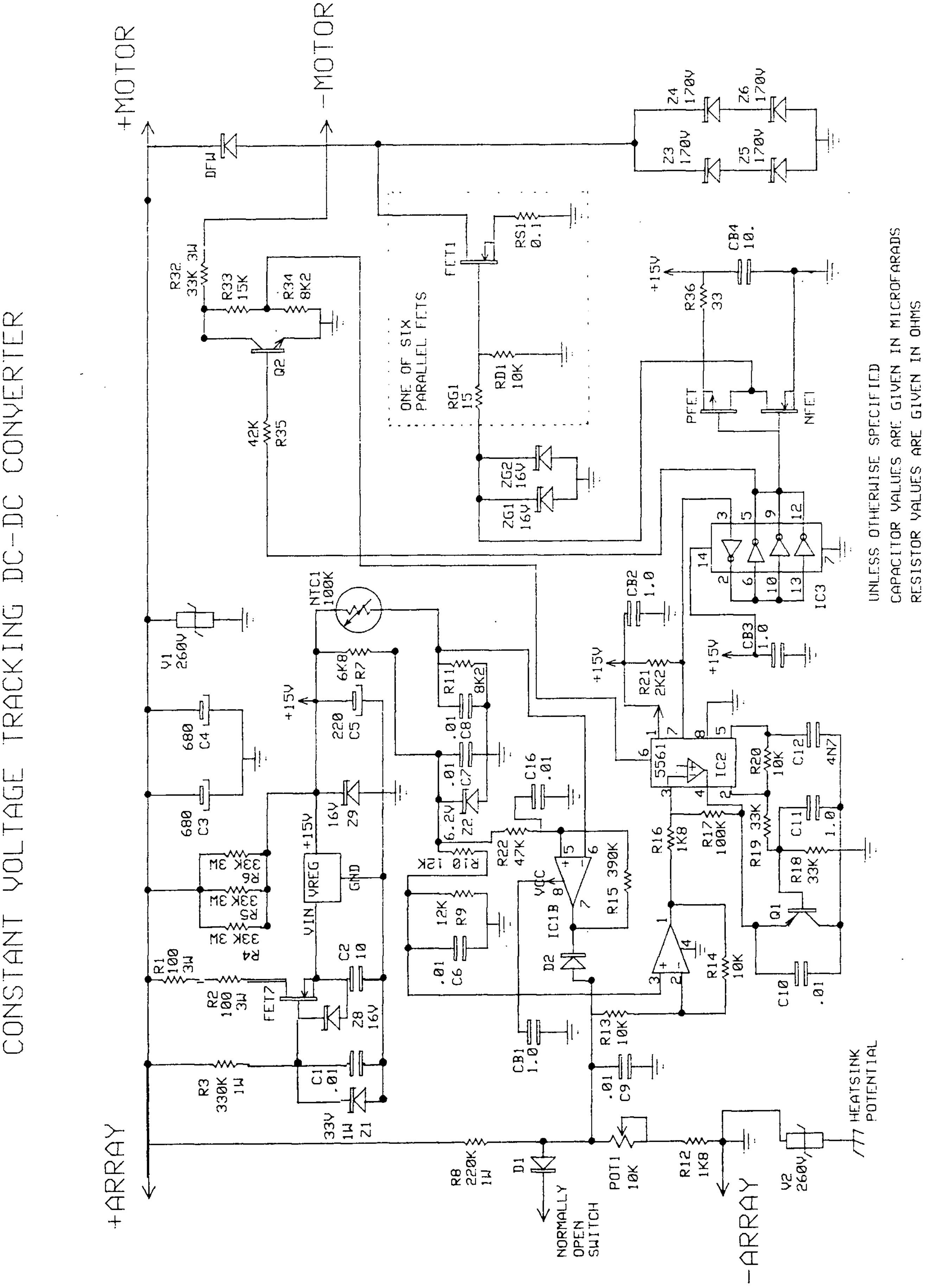

Figure 3.5: Circuit Diagram of the Voltage Tracking Converter

Table 3.2: Voltage Tracking Converter Specifications.

Electrical Specifications:

Power Rating	2.0kW
Efficiency @ 2kW	0.90
Input Operating Voltage	75V – 250V
Input Current	up to 12A
Output Voltage	30V – 240V
Continuous Output Current	14A
Peak Output Current	24A
Chopping Frequency	20kHz
Operating ambient temperature	–10°C to +55°C
Relative humidity	0 to 100%
Automatic thermal shutdown	
External shutdown through a normally open switch contact	

3.2.1 Basic Operation

The logic senses the array voltage via an adjustable resistive voltage divider. This signal is inverted via an LM358 inverting operational amplifier which pivots around a 3.1V reference with a gain of minus one. The resultant signal is amplified by the internal opamp of the NE5561 PWM with a gain of minus 100 to form the reference voltage which determines the duty cycle. One volt corresponds to a 0% duty cycle while 5V corresponds to a 98% duty cycle.

The chopping frequency is set by the RC oscillator of the NE5561 to approximately 20kHz. The chopped output of the NE5561 PWM is buffered and then used as the gate drive signal.

3.2.2 Gate Drive

A 4041 buffer is placed between the 5561 PWM chip and the gate drive. The complementary NFET-PFET pair which forms the gate drive requires a little more switching current than the PWM can provide. Also the extra buffer helps to isolate the PWM from the power supply.

A resistor is placed in series with the positive supply of the complementary NFET-PFET pair to limit the amount of gate current which can be supplied. This slows down the switching speed of the mosfets and limits the reverse recovery current through the devices. The series resistor should be between 29Ω and 47Ω to be effective. There is no problem switching the mosfets off as fast as possible, so it is not necessary to place resistance in the ground line of the complementary FET pair.

3.2.3 Overcurrent Protection

Cycle by cycle overcurrent protection is provided by feeding a voltage signal proportional to the current through the mosfets to pin 6 of the NE5561. If this voltage rises above 0.6V the NE5561 output is forced high and turns *off* the mosfets for the remainder of the cycle. This current signal is derived from the voltage across the mosfets. A resistive voltage divider feeds a portion of the mosfets *on-state* voltage to the NE5561. The high *off-state* voltage is ignored by switching on an NPN transistor, effectively shorting the segment of the voltage divider that provides the current signal.

3.2.4 Thermal Protection

The internal temperature of the converter is monitored with the aid of a thermistor. If the temperature rises beyond the limit as indicated by the 6.2V reference zener diode the converter will be shut down. Hysteresis is built around the operational amplifier, acting as a comparator, to give the converter time to cool down before the converter is restarted. Resistance values for the NTC1, R11, R15, and R22 are chosen such that the converter is shut down at 85°C and restarted at 65°C.

The sealed box which encloses the converter forms part of the heatsink so that in the steady state the internal temperature will approximately equal the temperature of the heatsinks. Also resistors and integrated circuits mounted on the printed circuit board itself produce heat that should raise the operating temperature slightly above the heatsink temperature.

All electronic components must be rated to operate in an ambient temperature of up to 85°C. The mosfets, diode and capacitor are rated to operate at this high temperature. However, all the integrated circuits used in the prototype were not. The LM358 operational amplifier should be replaced with an LM258 opamp and the NE5561 PWM should be replaced with the SE5561 PWM. These devices are rated for use over a wider temperature range and are only moderately more expensive.

3.2.5 Protection

Some extra components have been added to the circuit to protect the power devices:

- Two pairs of two zener diodes in series have been placed in parallel with the power mosfets to protect against overvoltages caused by any stray circuit inductance. Two pairs are used instead of one to maintain circuit symmetry. Two zeners are placed in series in each branch to form a high enough voltage rating.

- A 0.1Ω resistor is placed between source and ground of each power mosfet. This small resistance enhances the current sharing capabilities of the mosfets during switching thereby minimizing the effects of varying device current gains.

- A 16V zener diode is placed across the power supply. If the resistors provide more logic current than required, then the excess current is is bled off by the zener diode.

3.3 Testing and Results

To simulate a solar array source a variable DC source was used in series with a variable resistance. This would produce a linear voltage, current curve rather than the humped curve of Figure 1.1. This setup is however adequate to demonstrate the operation of the converter.

3.3.1 Waveforms

The converter was first tested with a load made up of a 11.5mH inductor in series with a variable resistance. The converter was run at various input voltages ranging between 50V and 250V and with input currents between 0A and 12A. Output voltages ranged between 0V to 200V and output currents between 0A to 24A.

The drain to source voltage waveform is displayed in Figure 3.6. A voltage spike is evident during turn off. This is due to stray circuit inductance which is impossible to completely eliminate. The magnitude of the spike is approximately 70V and can easily be tolerated as the mosfets are rated 150V higher than the maximum input voltage. A magnified view of this spike is shown in Figure 3.7. It is seen here as a damped sinusoid with a natural frequency of 40MHz. The mosfets are protected with zener diodes which should clamp the voltage appearing across them at 350V. Even at conditions of maximum input voltage and peak output current the voltage spike across the mosfets did not approach the 350V limit.

The voltage across the mosfets drops very rapidly, even though the rise of the gate

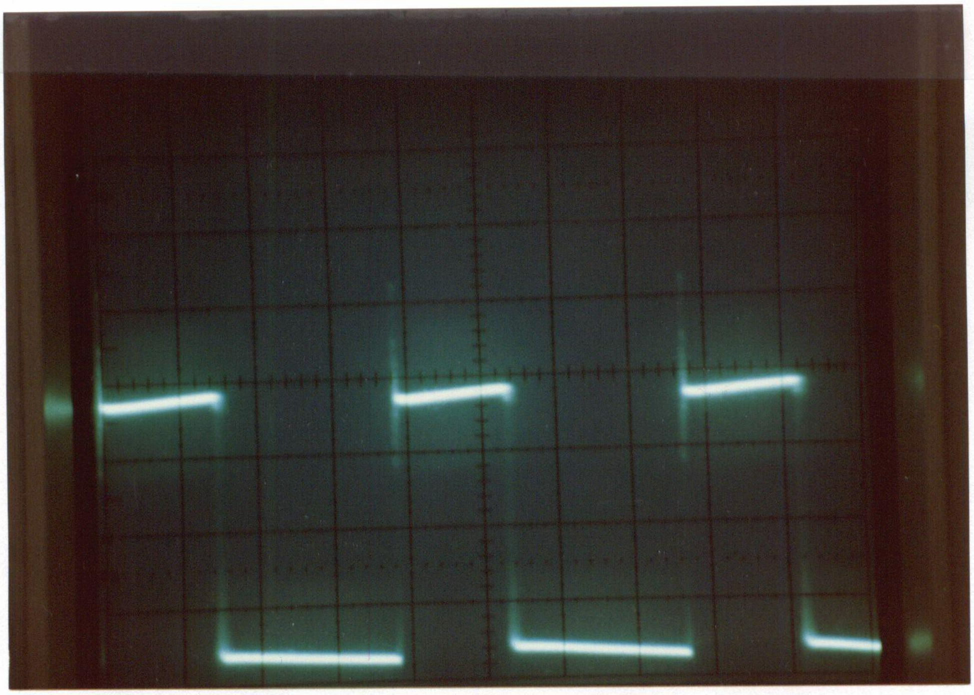

Figure 3.6: Drain-to-Source Voltage

V_{IN} :	200V	V_{OUT} :	123V	50V/div.
I_{IN} :	7.7A	I_{OUT} :	12.1A	10.0μs/div.

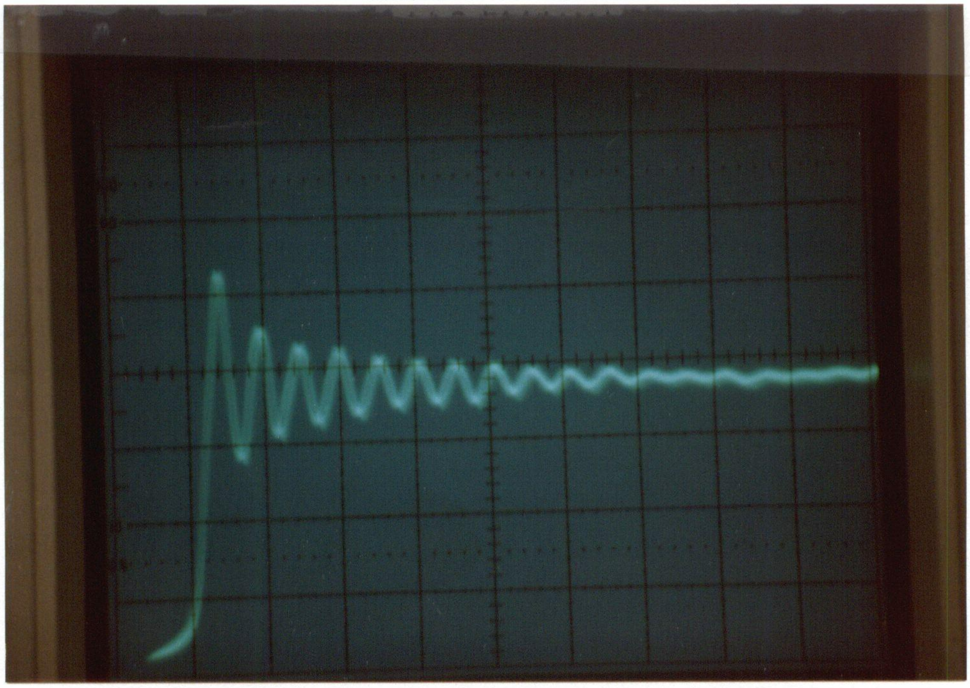

Figure 3.7: Drain-to-Source Voltage at Turn-off

V_{IN} :	202V	V_{OUT} :	125V	50V/div.
I_{IN} :	7.52A	I_{OUT} :	11.95A	0.05μs/div.

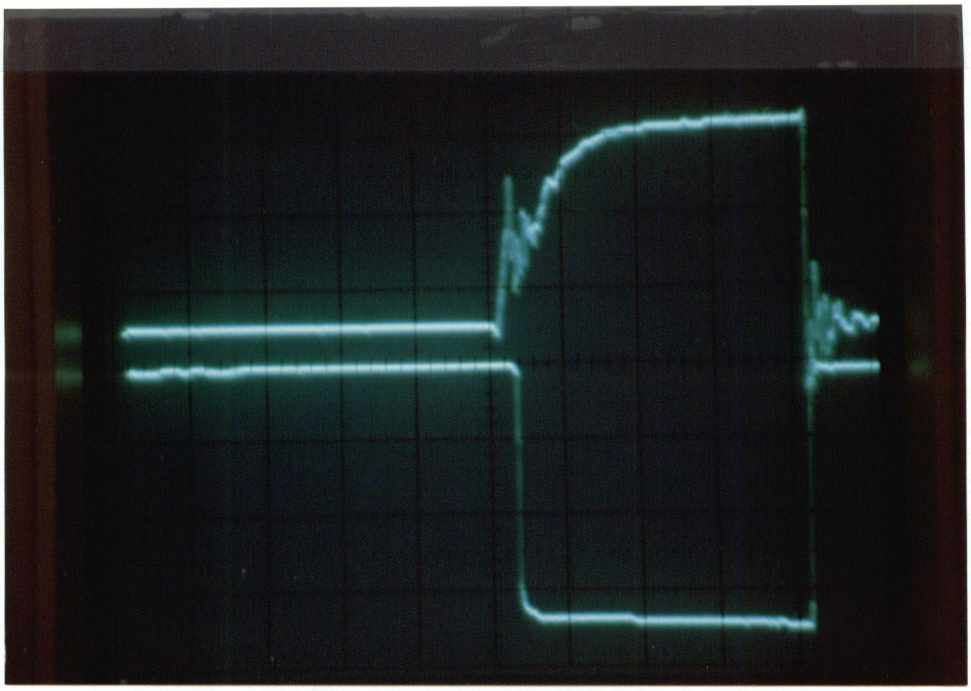

Figure 3.8: Gate-to-Source Voltage with the Drain-to-Source Voltage

Top trace.......	Gate to source voltage:	5V/div	2.0μs/div.
Bottom trace....	Drain to source voltage:	50V/div	2.0μs/div.

V_{IN} :	200V	V_{OUT} :	35V
I_{IN} :	1.7A	I_{OUT} :	9.3A

voltage is limited by a series resistor . The drain to source voltage together with the gate voltage is shown in Figure 3.8. The gate voltage takes approximately 0.5μs to rise to near its' peak value, however the drain to source voltage falls so rapidly that it appears instantaneous. There is a delay of about 0.15μs from the instant that the gate voltage starts to rise and the drain to source voltage falls. This is because the drain to source voltage is constrained to remain high until the mosfets are conducting the full load current. The rate at which current through the FET can rise is limited by the current gain of the device during switching. Limiting the gate current therefore limits the rate of rise of the current through the mosfet. This slows down the switching process and limits the peak recovery current through the device.

The signal used by the PWM to measure the current through the mosfets is shown in Figure 3.9. The displayed signal is turning off the mosfets on a cycle by cycle basis as the voltage exceeds the threshold level. The level at which the overcurrent protection engages is determined by the relative resistance values of R32, R33, and R34 in the voltage divider of Figure 3.5.

The PWM only samples the current signal when the mosfets are turned on. The signal voltage during the mosfets off-state is ignored even though it may be higher than the cutoff threshold value. Also during the switching intervals a considerable amount of noise is present in the current signal. However, the operation of the overcurrent detector is not adversely effected.

Figure 3.9 shows the current signal dipping down when the mosfets are first turned on. The signal grows as stray circuit capacitance is charged up through the resistive divider network. When the voltage builds up to the internal reference level of the NE5561, the mosfets are shut off creating the noise spikes visible on the waveform. The shorting transistor, Q2, is turned on when the mosfets are turned off preventing the current signal from going excessively high.

The over-temperature protection was tested by operating the device with a peak load current for a prolonged period of time. The converter shut itself off when the internal temperature reached approximately 85°C and automatically restarted when the temperature cooled to 65°C. The slow-start circuitry on the NE5561 and the overcurrent protection prevented excessive current from flowing through the converter as it was restarted.

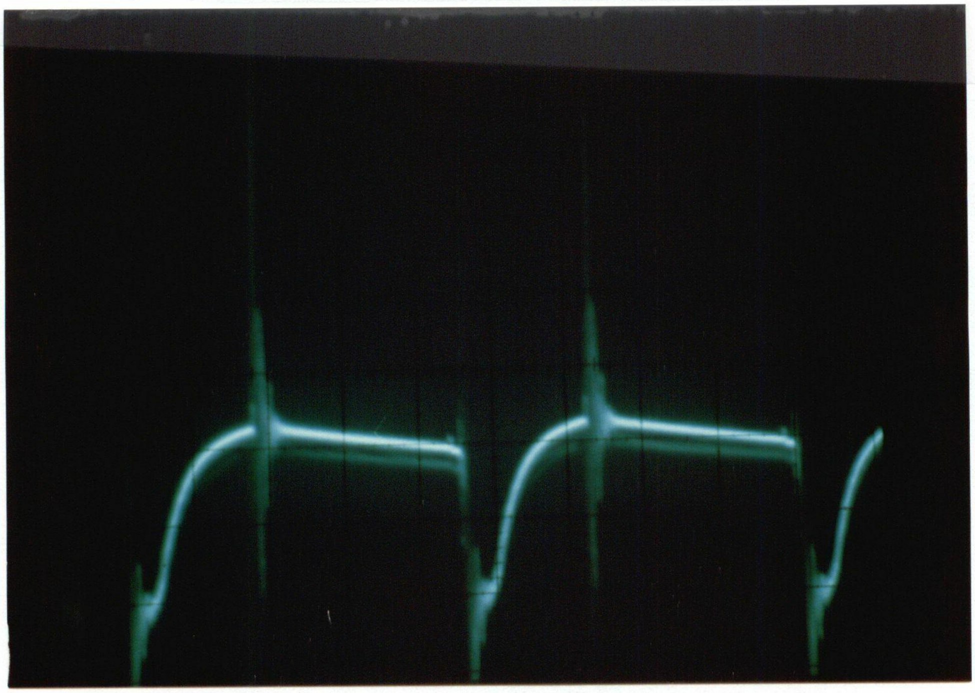

Figure 3.9: Current Signal

V_{IN} :	223V	V_{OUT} :	84V	20mV/div.
I_{IN} :	7.1A	I_{OUT} :	18.5A	10μs/div.

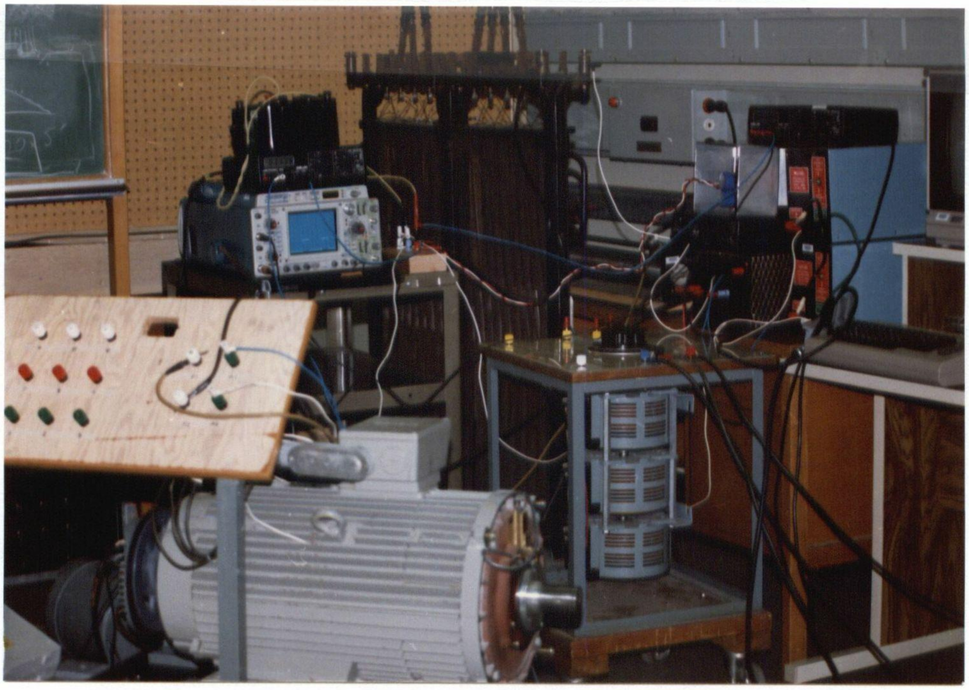

Figure 3.10: Experimental Set-up

3.3.2 Efficiency

The converter was also tested by driving a 2.5hp DC motor which in turn drove an induction machine operating as a generator. The generator supplied power to a resistive three-phase load. The generator load could be adjusted in order to vary the loading of the DC machine. Figure 3.10 shows the experimental set up.

No problems were experienced by the converter driving the DC machine. The overcurrent protection operated successfully during machine start up and the starting torque was easily overcome. Measurements made to determine the converter efficiency are displayed in Table 3.3. The peak efficiency of the converter is about 96% which exceeds the design specifications.

Table 3.3: Converter Efficiency

Duty Cycle	Input Voltage	Input Current	Input Power	Output Voltage	Output Current	Output Power	Efficiency η
Resistive and inductive load:							
0%	15mA	180V	2.7W	0A	0V	0W	0
14%	198.5V	1.4A	278W	30.0V	6.8A	204W	0.734
28%	199.8V	3.55A	709.3W	55V	12.2A	671W	0.946
40%	201.5V	7.3A	1471W	80V	18A	1440W	0.978
9.3%	200.2V	0.22A	44W	18.3V	1.9A	34.8W	0.789
25%	200.5V	1.3A	261W	49V	5.1A	250W	0.958
47%	201.5V	4.57A	921W	93.5V	9.5A	888W	0.964
66%	204V	8.95A	1826W	133V	13.25A	1762W	0.965
71%	205V	10.0A	2050W	141V	14.0A	1974W	0.963
76%	206.5V	4.44A	917W	155V	5.7A	883W	0.963
90%	208V	6.4A	1331W	187V	6.97A	1303W	0.979
95%	209V	7.4A	1547W	202V	7.53A	1521W	0.983

Speed r/min	Input Voltage	Input Current	Input PoWer	Output Voltage	Output Current	Output PoWer	Efficiency η
DC Motor load:							
1000	141V	4.45A	627W	98.5V	6.1A	601W	0.958
1000	141V	5.18A	730W	99V	7.1A	703W	0.962
1000	141.5V	7.59A	1074W	98V	10.5A	1029W	0.958
1500	190.8V	2.14A	408W	137V	2.74A	375W	0.920
1500	191.2V	3.60A	688W	138V	4.75A	655W	0.952
1500	192.1V	5.89A	1131W	142V	7.68A	1090W	0.964
1500	200V	9.1A	1820W	145V	11.86A	1720W	0.945
2000	215.6V	2.45A	528W	188V	2.18A	410W	0.776
2000	216V	3.21A	693W	187V	3.6A	673W	0.971
2000	242.8V	4.1A	995W	185V	5.0A	925W	0.929
2000	246V	6.85A	1685W	186V	8.42A	1566W	0.929
2000	248.6V	9.46A	2352W	187V	12.1A	2262W	0.962

Chapter 4

Maximum Power Tracking Converter

A true maximum power point controller that will automatically adjust to different input conditions is described in this chapter. This converter is designed to operate at power levels of up to one kilowatt and was tested using the facilities at B.C. Hydro-Research, Surrey, B.C.

4.1 The Logic Circuit

The control circuit illustrated in the block diagram of Figure 4.11 provides true maximum power point tracking. The controller adjusts the conversion ratio to maximize the array voltage and current product.

The array voltage is sampled directly using a resistive voltage divider. The array current, however, is derived from the voltage across the mosfet. During its *on* state, the mosfet appears as a resistive element. The average *on* state voltage is proportional to the current through the device which is in turn proportional to the steady state array current. Changes in the mosfet resistance with temperature are unimportant as only relative changes in the current signal are of interest and not its absolute value.

The average array current and voltage signals are multiplied using an analog multiplier to produce an output power signal. A low frequency (3 Hz.) clock latches the power level into the sample and hold unit. The controller then changes the conversion ratio of the DC to DC converter. At the end of the cycle the instantaneous power level is compared with the sampled power level.

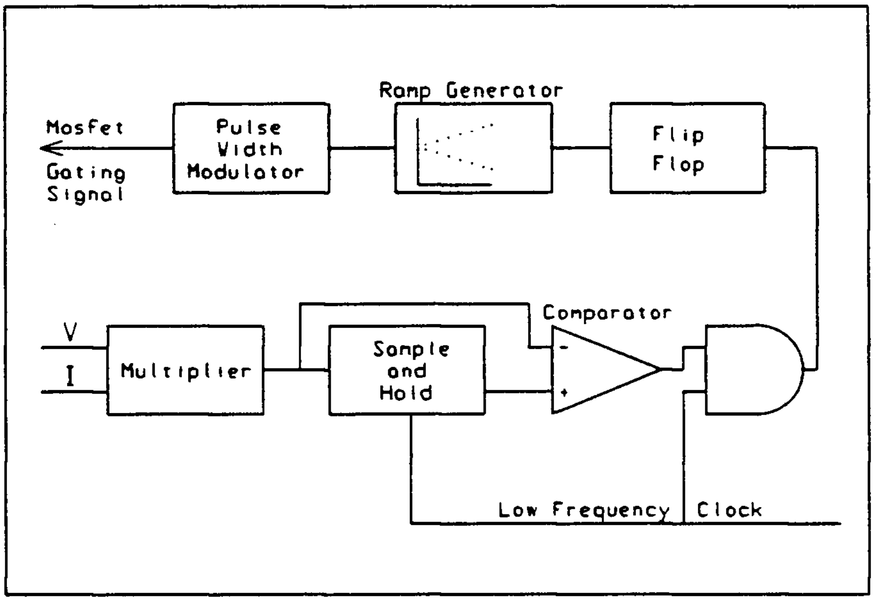

Figure 4.11: Block Diagram of Maximum Power Point Tracking Circuit

If the power level has increased, another change is made to the conversion ratio in the same direction. If the power level has decreased, the change in the conversion ratio is made in the opposite direction.

When the converter is first switched on, the conversion ratio is set to its minimum value and *climbs the hill* to the value corresponding to the maximum power point. In the steady state, the duty cycle toggles around this optimum value.

Two sets of low frequency timing pulses are required to synchronize the controller. Both pulses are very short, approximately $10\mu s$, with the second pulse immediately following the first.

The first pulse is logically *anded* with the power level comparator output and then used as the clock input to a JK flip-flop. When the power level is decreasing, the comparator output goes high, allowing the timing pulse to propagate through to the flip-flop, causing its outputs to toggle.

The complementary flip-flop outputs are each logically *anded* with the second timing pulse. One of the resulting signals has its polarity reversed. Once every cycle, there is either a positive or negative pulse generated, depending on the state of the flip flop outputs.

These positive and negative pulses are then integrated and scaled to form the input of a pulse width modulator. As the pulses are narrow, the duty cycle is changed quickly at the beginning of each cycle. For the rest of the cycle the duty cycle is held constant while the motor and control circuit transients decay. The instantaneous power signal then represents a steady state value and a true comparison can be made with the last sampled power level.

Some additional features include over-current protection which is provided on a cycle by cycle basis. If the measured current signal rises above a threshold level, a flip flop is set, turning off the power mosfet. The next gating signal resets the flip flop allowing the power mosfet to be turned on again.

The converter is also capable of being shut down if the motor overheats. The temperature signal provided by the DC motor can disable the gating signal as it switches from an open to a short circuit, or vice versa.

4.2 Results

The hydraulic test equipment consisted of a 200-litre storage tank in which various Mono progressive cavity pumps could be inserted. The pump is discharged into a pressure tank regulated by a back pressure sustaining valve. By adjusting the pressure, well depths ranging between 10 to 65 meters can be simulated. Water from the pressure tank is discharged back into the storage tank. The power output of the pump is calculated by multiplying the water flow rate with the back pressure.

The electrical system consisted of the panels, converter and motor. The photovoltaic array consisted of two parallel strings of 11 panels, producing a maximum power of 770 watts and nominal voltage of 165 V. The array was kindly supplied by British Columbia Hydro and Power Authority, Research and Development Division, which also provided laboratory space. Power from the array was routed through the DC - DC converter to a Brot 1.1 kW permanent magnet DC motor which directly coupled to the Mono pump.

The system was tested at various insolation levels and well depths. The greatest system losses occur during the conversion of sunlight to electricity, which proceeds at an efficiency of 0.083. A large loss is inevitable due to the physics of the conversion process.

The converter successfully altered the conversion ratio to track the maximum power point of the array for most light conditions. Figure 4.12 displays the power developed

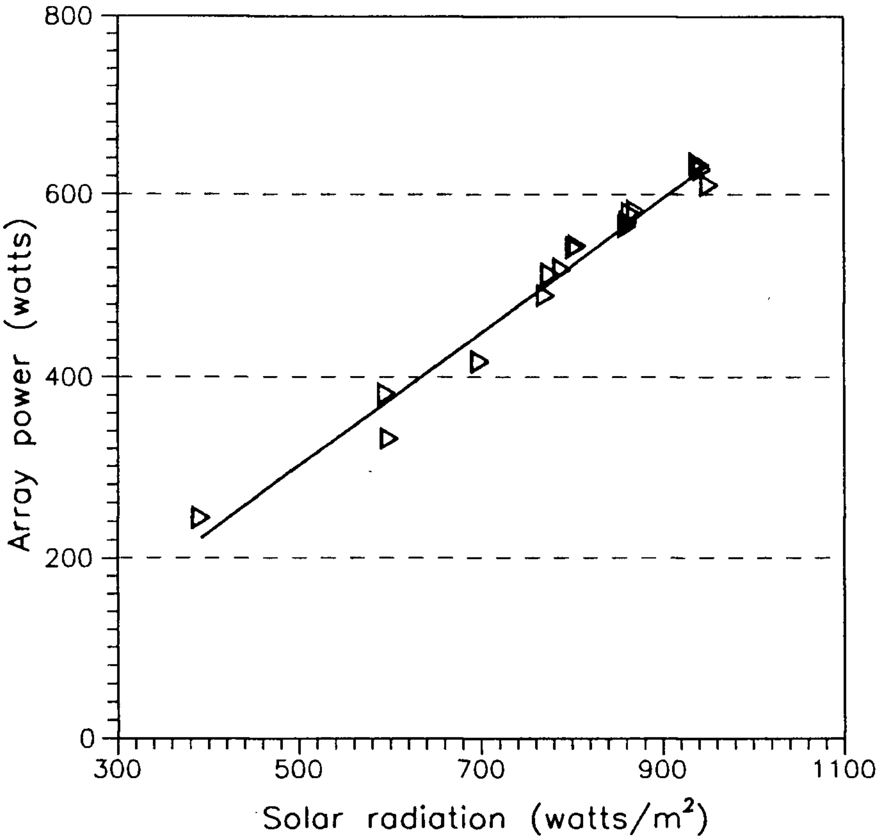

Figure 4.12: Array Power vs. Solar Radiation Level

by the array for various levels of solar radiation. The converter was also able to deliver enough current to develop the required starting torque at all simulated well depths. The efficiency of the converter, as calculated by the output converter power divided by input array power (as shown in Figure 4.13), ranged from 0.75 to 0.91 with an average of 0.84. Losses were incurred in the power mosfets, the freewheeling diode and the logic power supply. The voltage tracking converter of Chapter 3 turned out to be slightly more efficient because it used six parallel power mosfets of a similar rating to the two used here. Also the voltage tracking converter used a more efficient power supply.

The Brot permanent magnet DC motor operated with an average efficiency of 0.82. The efficiency ranged from 0.78 to 0.87 and tended to increase with motor speed as shown in Figure 4.14. Harmonic motor losses were small as the ripple current was kept

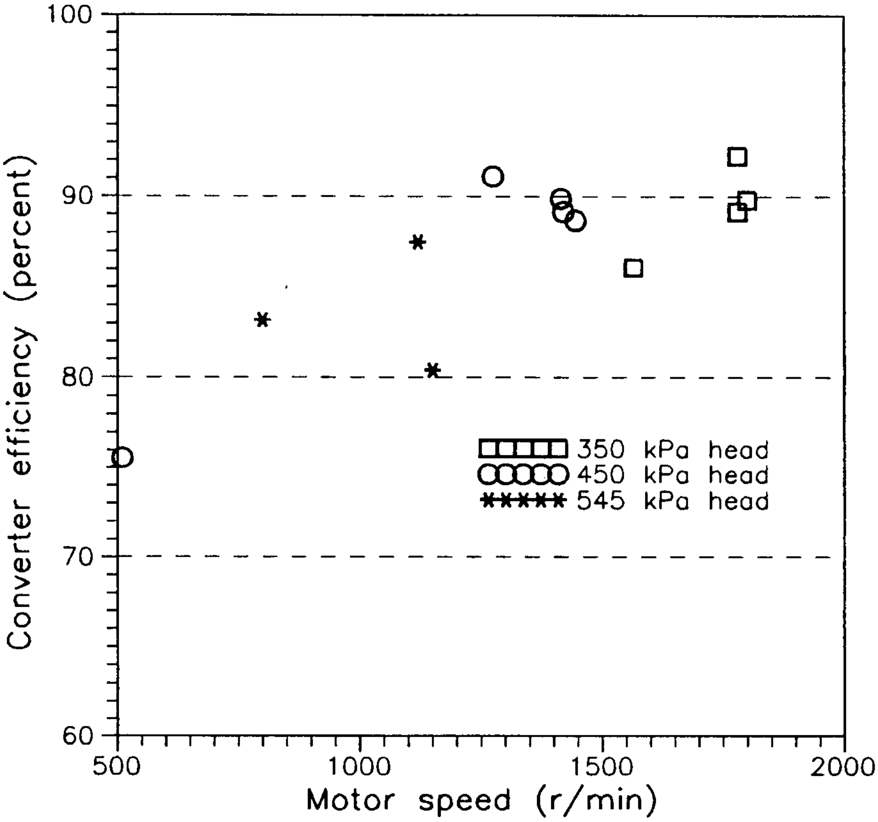

Figure 4.13: Converter Efficiency

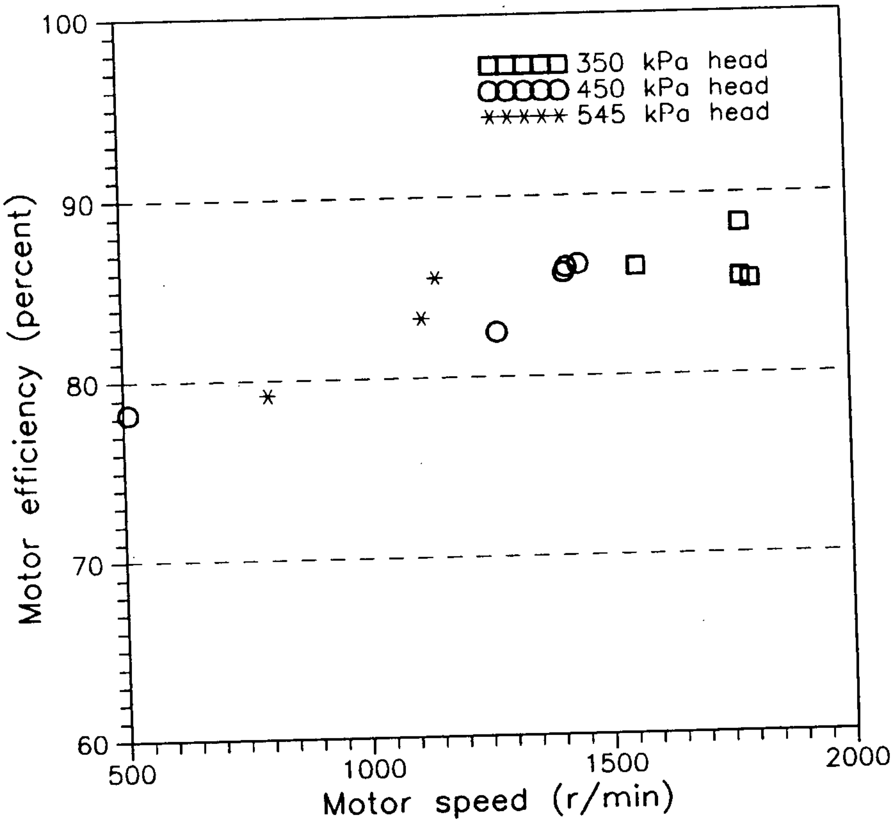

Figure 4.14: Motor Efficiency

below 0.1A at the 20kHz chopping frequency.

The efficiency of the Mono pump also increased with speed up to approximately 1600 r/min as shown in Figure 4.14. Both the increased mechanical vibrations in the driveshaft and the increased hydraulic resistance at larger flow rates reduce the pump efficiency at higher speeds.

4.3 Constant Voltage vs. Maximum Power Tracking

A simpler constant voltage tracking converter could perform many of the same functions as the maximum power tracking converter described in this chapter. It does not, however, provide the flexibility of the maximum power tracking system. The voltage tracking converter must be carefully adjusted at each individual installation while the maximum power tracking circuit automatically determines the optimum operating point. It also tracks the changes to this optimum operating point as the ambient temperature and insolation levels change with the time of day and the seasons. A small increase in the power extracted from the array (possibly 5%) is multiplied to form an even greater overall efficiency as the power converter, motor and pump all operate with a greater efficiency at higher speeds and power levels.

The maximum power tracking system may, however, have difficulty tracking fast changes in the insolation level. The converter assumes a constant or slowly varying insolation level. Changes in the observed power levels are assumed to be the result of a change to the conversion ratio, and not the result of a change in the insolation level. Normally, insolation levels change slowly and the converter is able to accurately track the maximum power point. However, occasionally weather conditions will cause rapid transitions in the light levels. It senses the changing power levels and attributes it to the last conversion ratio change. The conversion ratio then drifts away from the optimum value until the insolation level stabilizes.

Chapter 5

Microprocessor Based Hybrid Control

There are certain merits to using either a voltage tracking or a maximum power tracking control scheme. A voltage tracker has a simple control algorithm that is not affected by fast variations in insolation levels. A power tracker is self adjusting and can optimize the output of the array as the characteristics of the photovoltaic panels change with temperature and time.

A hybrid power tracking, voltage tracking converter retains the advantages of both control schemes. When the converter is first turned on, it enters the power tracking mode. The voltage corresponding to the maximum power point is located and retained. The converter then switches to the voltage tracking mode where the array voltage is held constant. Periodically the power tracking mode is reentered to make fine adjustments to the optimum operating voltage.

An analogue control circuit which could perform all of these functions would be quite complex. However, a digital implementation of this control scheme using a single chip microprocessor is feasible. Most of the logic functions can be implemented in software resulting in a low chip count. The system is flexible and additional functions can be incorporated to handle faults or special conditions as more sophisticated software is developed.

5.1 Logic Circuit

The majority of the control logic is handled by the Motorola MC68HC11 single chip microcomputer. It reads the photovoltaic array voltage and current signals, keeps track of timing, regulates the duty cycle and outputs a pulse-width-modulated waveform.

Some external logic is required to support the MC68HC11 and to drive the power mosfets. Figure 5.15 displays the circuit diagram of the external logic and gate drive chips.

5.1.1 MC68HC11 Single Chip Microprocessor

The MC68HC11 incorporates the following features which make it possible to perform these multiple tasks:

- An 8-bit central processing unit, CPU, containing two internal 16-bit index registers, *X and Y*, one 16-bit stack pointer, *S*, a 16-bit program counter, two 8-bit accumulators,*A and B*, and a condition code register, *CC*. The two 8-bit accumulators can be concatenated into one 16-bit register, *D*. The CPU can be operated with a full 64K bytes of external memory or in the single chip mode. In this application the single chip mode of operation is selected.

- There are 8K bytes of read only memory, ROM, available to store the resident program and also 512 bytes of EPROM and 256 bytes of RAM to store variables and constants.

- A 16-bit free-running timer is available for use by the program. The external lines associated with the timer are attached to port A. Some timer features used in this application include:

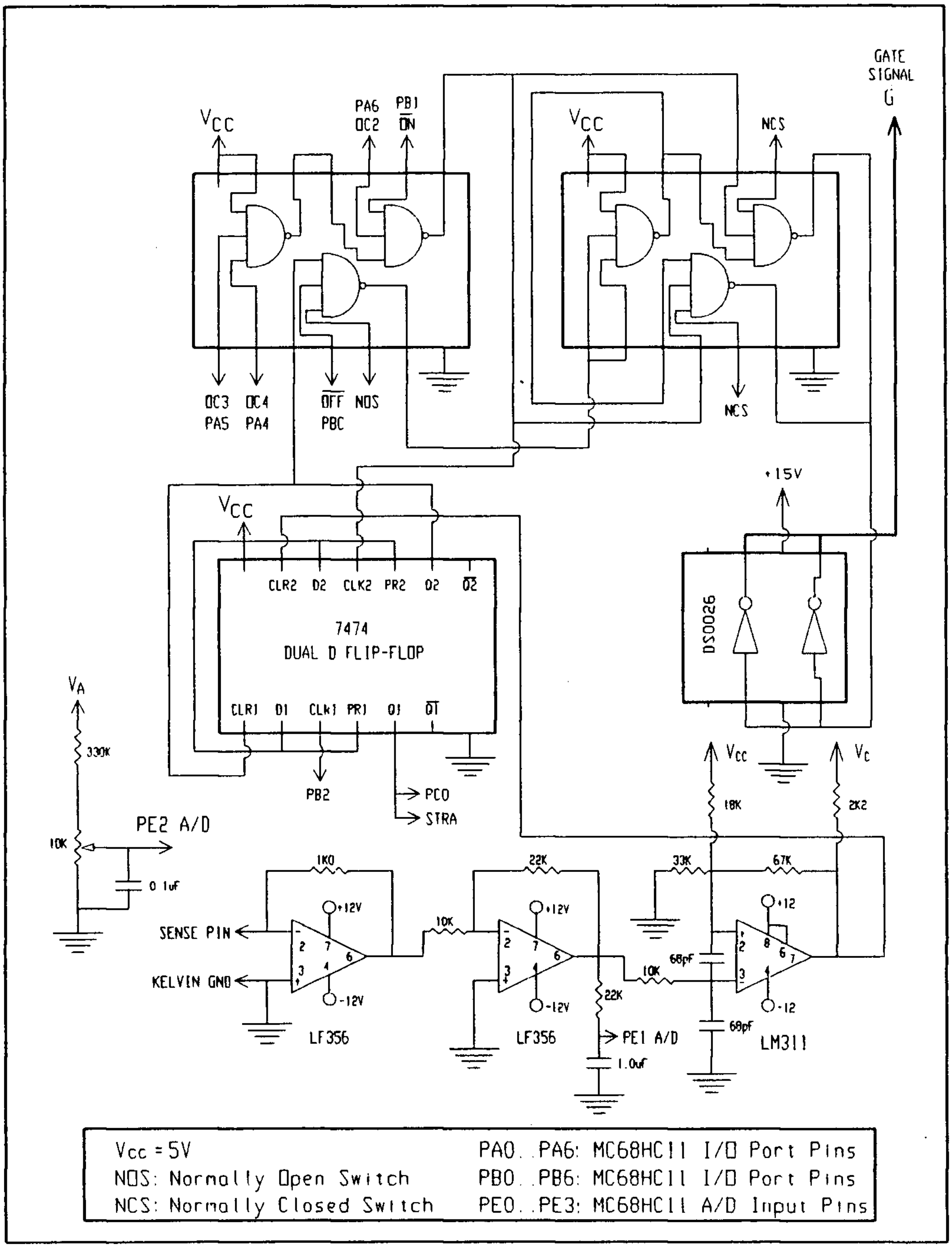

Figure 5.15: External Logic Circuit

  - Five output compare registers attached to external pins PA3 through PA7. The voltage levels on these pins can be forced high or low, when the free-running timer count equals the number stored in the corresponding compare register. Interrupt requests can be optionally generated upon a successful output compare.
  - A periodic real time interrupt can be generated at various rates. This interrupt can be used to update a real time clock.
  - Some other features include a pulse accumulator, which can count external events or be used as the external timer clock, and an input capture register which can hold the timer count when a transition is sensed on an external pin.

- There are two eight-bit general purpose ports, B and C, available. Port C can be used for input or output while port B is strictly an output port. Ports A, D, and E, which are set aside for the timer, A/D converter, and communications interface can also be used as general purpose I/O ports when the special functions attached to these ports are not being used.

- A Serial Peripheral Interface, SPI, and Serial Communications Interface, SCI, are built into the chip and connected to port D. These interfaces are used to communicate with peripheral devices or external systems and have various modes of operation.

- An eight-bit analogue-to-digital, A/D, converter is included with four, or in some packages eight, input channels and connected to port E. The input channels can be individually selected or all four channels can be read consecutively. The conversions can take place continuously or under program control.

- Two non-maskable and fifteen maskable interrupt sources are possible. The interrupts obey a fixed hardware priority structure to resolve simultaneous requests. However, the priority of a maskable interrupt source can be raised under program control. Maskable interrupts are disabled by setting the Enable bit of the CC register. Internally generated interrupt requests also have a local mask. Each interrupt source has a corresponding interrupt vector containing the address of the interrupt routine. The CPU responds to an active request by saving the register state on the stack, setting the enable bit, and jumping to the address indicated by the interrupt vector.

The power requirements of the MC68IIC11 are modest making it attractive for high efficiency, low power applications. It requires 20mA at 5V in the run mode, and even less in the special wait or stop modes.

5.1.2 Current Sensing

A current signal is derived with the aid of a current sensing power mosfet. This device has identical characteristics to the regular power mosfets, except the source of a few transistor cells are isolated and connected to a separate external pin. Under ideal conditions, the current diverted to the *current sense* pin would be the ratio of the number of isolated cells to the total number of parallel cells in the power device. A separate *Kelvin source* pin is provided to increase the accuracy of the current measurement. This pin is internally connected to the source of the power device and does not share the metalization, bonding wire, and pin resistance with the external source pin.

A virtual earth sensing circuit is used to amplify the current signal. This method is superior to a resistor sensing circuit in terms of speed, accuracy and noise immunity. However, the resultant signal is inverted and a second operational amplifier is required to produce a signal of the correct sign. This current signal is suitable to be used as a reference for the cycle by cycle overcurrent protection.

The current waveform is filtered to provide the A/D input pin, PE2, with a signal proportional to the average current through the power mosfets. In the steady state this signal is also proportional to the current provided by the supply.

5.1.3 Overcurrent Protection

Cycle by cycle overcurrent protection is necessary when using a chopper to drive a variable impedance load such as a DC motor. During start up the motor may draw large currents to overcome static friction and inertia. The protection must respond quickly to be effective.

The protection operates by comparing the instantaneous current signal to a reference voltage level. If the current signal exceeds the reference, the LM311 comparator output toggles low, which in turn asynchronously latches the output of a D flip-flop low. When low, this flip-flop output turns off the power mosfets. At the start of the next cycle, the flip-flop clock input is strobed, resetting the output, which enables the power mosfets to be turned on once more. If the current signal again exceeds the reference, the power mosfets are switched off and the process is repeated.

During an overcurrent fault the duty cycle of the chopper is governed by the overcurrent hardware and not the main logic program. The microprocessor therefore needs to be flagged when an overcurrent condition occurs. This is achieved with the aid of a second D flip-flop. The overcurrent signal is attached to the clear pin of this flip-flop to latch the output low. This output forces the STRA pin low, which in turn, is capable of generating an interrupt request. The output also forces bit 0 of port C low, which can be polled to determine if the overcurrent condition persists. The flip-flop clock line, bit 2 of port B, must be strobed to reset the flip-flop.

5.1.4 Voltage Sensing

A portion of the input voltage to the chopper is sampled by the A/D converter via a resistive voltage divider. A potentiometer adjusts the sampled voltage to just below 5 V under open circuit conditions to utilize the full scale of the A/D converter. This manually adjusted potentiometer could be replaced with a digitally controlled potentiometer, such as the Xicor X9MME, for complete automation of the adjustment process.

5.1.5 Pulse Width Modulation

The conventional means of creating a pulse-width-modulated waveform is with a PWM IC such as the NE5561. The frequency is fixed by an R,C oscillator while the pulse width is dependent on the input voltage level. When a digital controller is used, a D/A conversion is necessary to arrive at an analogue input voltage to the PWM. Yet the gating signal is basically a digital waveform.

The D/A conversion is necessary because the digital processor is limited. It cannot toggle an output port bit fast or accurately enough to create a pulse-width-modulated waveform at a useful frequency. It may be possible, however, to create the waveform with the aid of external timers. The logic circuit would then become more complex and costly. What is gained by eliminating the D/A converter and PWM chip is offset by the additional timers.

The MC68HC11 however has an on-board timer. A pulse-width-modulated waveform can be created with the aid of only limited additional hardware in the following manner:

The timer is free running and counts from 0 through to FFFF continuously. Attached to the timer are five output compare registers OC1 through OC5. Registers

OC2 through OC5 are attached to the output port bits PA6 through PA3 respectively. Register OC1 can be attached to PA7 and it can also affect the output state of PA6 through PA3.

When a number loaded into one of the compare registers equals the timer count, a specified action will occur. For example, when the timer count equals the number contained in the OC2 register, PA6 can be forced high, or low, or be made to toggle, depending on the control register settings. An interrupt request can also be generated upon a successful compare.

Compare register OC1 is special. It can affect all output port bits associated with the compare registers simultaneously. The change called for by OC1 will override a change called for by any other output compare register in the event of a conflict.

A pulse-width-modulated waveform can be created at the output of pin PA6 by using OC1 to set the pin high and by using OC2 to reset the pin low. The compare registers would be updated each cycle by responding to an interrupt request generated by an OC1 successful compare. The processor would add a number representing the period to each compare register within the interrupt routine. The number contained in register OC2 would be offset from the number in OC1 by the *on* time. In this scheme the *on* time must be long enough for the processor to respond to the interrupt request and update register OC2 ruling out the use of a small duty cycle.

It is possible to use a full range of duty cycles by switching interrupt sources. When the duty cycle is greater than 50% OC1 could generate interrupt requests as already explained. When the duty cycle is less than 50% interrupt requests could be generated by successful OC2 comparisons. Switching interrupt sources must be done with care to ensure a smooth transition.

The maximum frequency of the generated waveform is limited by the speed of the CPU. Before it can respond to an interrupt request the CPU must complete the present instruction, save the register state, and load the address of the interrupt routine into the program counter. Once in the interrupt routine, the CPU must update the next compare register before the timer count runs past it. These processes may take up to 37 machine cycles or 18.5μs and must be completed within one half of a period of the pulse width modulated waveform. Updating the next compare register, clearing the interrupt request and returning to the interrupted task will take another 35 machine cycles. The maximum frequency is therefore limited to about 25kHz.

At an operating frequency of 25kHz there would be very little free time for the CPU to service other procedures. Only one or two instructions could be executed between interrupt requests. Also the adjustment to the pulse width would be coarse. The minimum adjustment to the pulse width is a 0.5μs step, so at 25kHz there are only 80 steps between a 0% and 100% duty cycle.

It is possible to free up CPU time and effectively decrease the step size by creating three output waveforms instead of one. These waveforms are combined in a set of external NAND gates to produce a single waveform at twice the frequency. A set of these waveforms are displayed in Figure 5.16. The waveforms produced at pins PA5 and PA4 are NANDED together. The output is then NANDED together with the waveform produced at pin PA6 to produce the resultant waveform shown in Figure 5.16.

The waveform at pin PA5 remains at a 50% duty cycle while the waveforms at PA6 and PA4 are adjustable. By adjusting the waveforms at PA6 and PA4 separately the duty cycle resolution is effectively doubled. For example if the period is 50μs and the *on* time of the first pulse stream is 18.0μs while the on time of the second pulse stream is 18.5μs the effective duty cycle is 36.5%. The duty cycle is adjustable in 0.5% steps instead of 1.0% steps, which would be the limit with a single pulse stream.

With some additional NAND gates a full range of duty cycles from 0% to 100% is

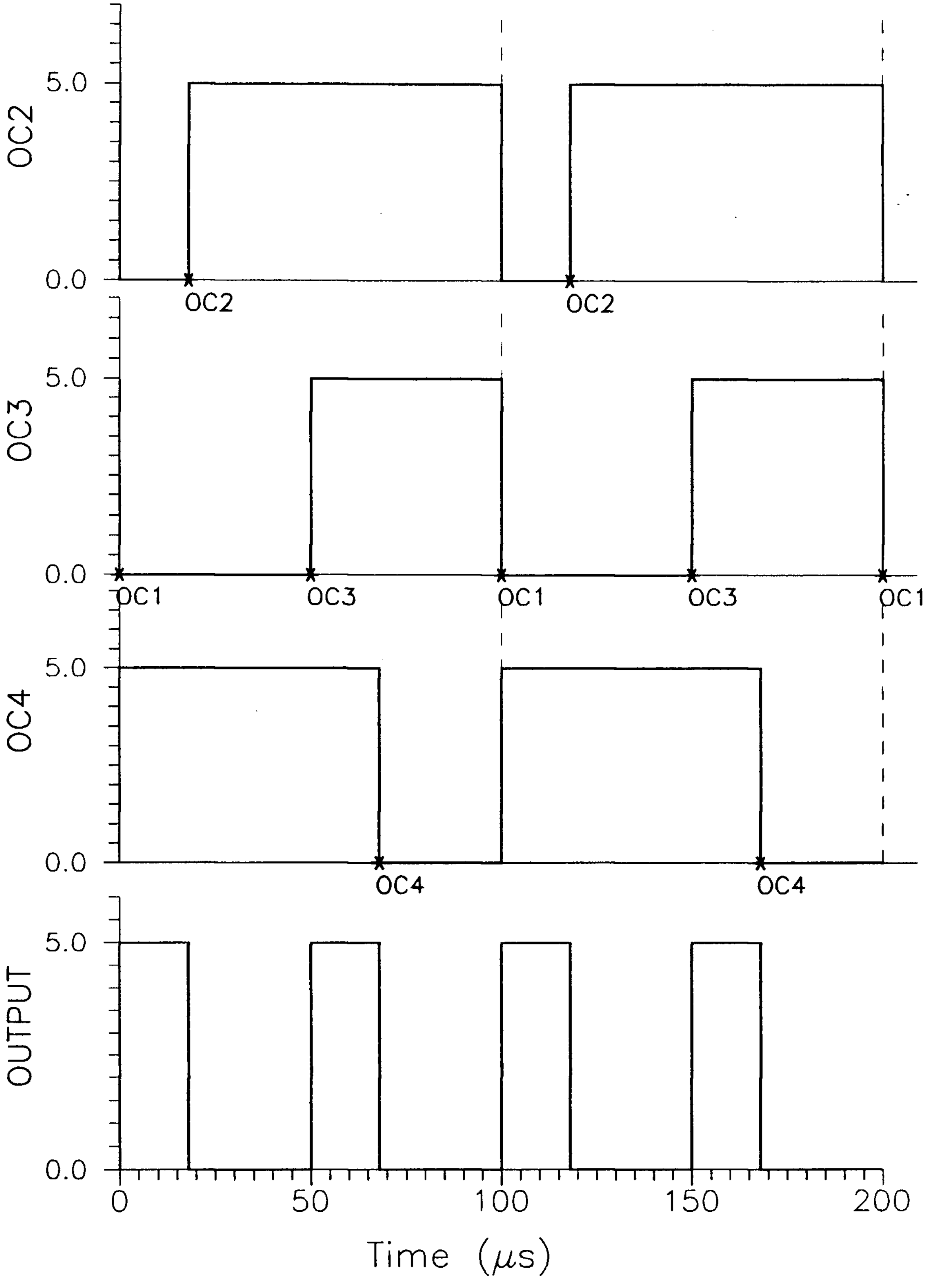

Figure 5.16: Pulse Width Modulated Waveforms

possible, unlike conventional PWM chips, which are capable only of duty cycles between 0% and 98%. Bit one of port A is wired to the input of one NAND gate and dedicated to turning the converter fully *off* when low, while bit two is wired to the input of a second NAND gate and dedicated to turning the converter *on*. Figure 5.15 shows the detailed circuit diagram. Addition NAND gate inputs are used to externally switch the converter off.

When turned fully *off* or *on* the pulse width modulation continues at a minimum or maximum duty cycle. This simplifies the restart process as timer synchronization within the program is never lost.

5.1.6 Gate Drive

The gates of the three parallel power mosfets are driven by a DS0026 dual inverting buffer. This device can deliver a large peak current, 1.5A, to rapidly switch the power mosfets. A series resistance of 33Ω is inserted between each power mosfet and the gate to limit the switching speed and prevent oscillations between parallel devices.

5.2 Program

The main function of the logic program is to alternate between the maximum power and voltage tracking modes of operation. Various sub-tasks must be coordinated for the main program to successfully operate.

5.2.1 Maximum Power Tracking

The program begins in the maximum power tracking mode and periodically reenters it. In this mode a search is made for the optimum operating point and the voltage corresponding to this point is recorded.

Initially the converter is turned fully *off*. The open circuit voltage is read together with the DC offset, if any, in the current measurement hardware. The duty cycle is then slowly increased in steps of approximately 5%. After each change to the duty cycle a delay of two seconds is introduced to allow system transients to decay. The voltage and current signals are sampled at each step and the operating power calculated. A record of the last three power and operating voltage levels are kept for future reference.

The present power level is compared with previous power levels. If the measured power is increasing the duty cycle is again changed in the same direction. If, however, the power level has twice decreased, the direction of the search is reversed. The direction of the search will also reverse if the converter reaches the limit of being turned either fully on or fully off. Each time the direction of the search is changed, the maximum recorded operating power and corresponding voltage level of the latest sweep are recorded in an array. Also, the amount the *on* time is changed between samples is reduced for a finer gradient search.

Once enough sweeps past the maximum power point have been made, presently 12, the power tracking mode is discontinued. A search is made for the highest recorded six power levels. The voltages corresponding to these power levels is then averaged and passed on to the voltage tracking routine.

After a period of time, half an hour, the power tracking mode is reentered. The principle of sweeping past the maximum power point is maintained. This time, however, the search for the maximum power point originates at the present duty cycle rather than starting from a fully off position.

5.2.2 Voltage Tracking

The controller enters the voltage tracking mode after the power tracking routine has determined the optimum operating voltage. The voltage is held constant by periodically, ten times a second, comparing the operating voltage to the reference. An adjustment to the duty cycle is made to compensate for the difference between the optimum and measured voltage, ΔV.

To calculate the magnitude of the change in the *on* time, Δt_{ON}, a simplified model will be used. The photovoltaic array will be modeled by a voltage source, V_S, with an internal resistance, R_S. The DC machine will be represented by a resistor and an inductor.

The converter, by altering the duty cycle, adjusts the effective resistance of the load, R_{EFF}, as seen by the source. This system with a simple resistive and inductive load is shown in Figure 5.17 and Figure 5.18. The voltage at the terminals of the source, V_A, is:

$$V_A = V_S - R_S I_S \tag{5.16}$$

where I_S is the source current. Substituting

$$I_S = I_L \cdot \frac{t_{ON}}{T} \tag{5.17}$$

into Equation 5.16, where I_L is the load current and T is the period yields:

$$V_A = V_S - R_S I_L \cdot \frac{t_{ON}}{T} \tag{5.18}$$

Assume the converter is operating near the maximum power point. Then it can be shown for the simple resistive and inductive load modeled here that $\frac{\Delta I_L}{\Delta t_{ON}} = 0$. Also if one considers the constant torque Mono pump as the load, then I_L is constant for most operating conditions. In any case, for the purpose of approximating the derivative of Equation 5.18 near the maximum power point, I_L shall be considered constant. Therefore:

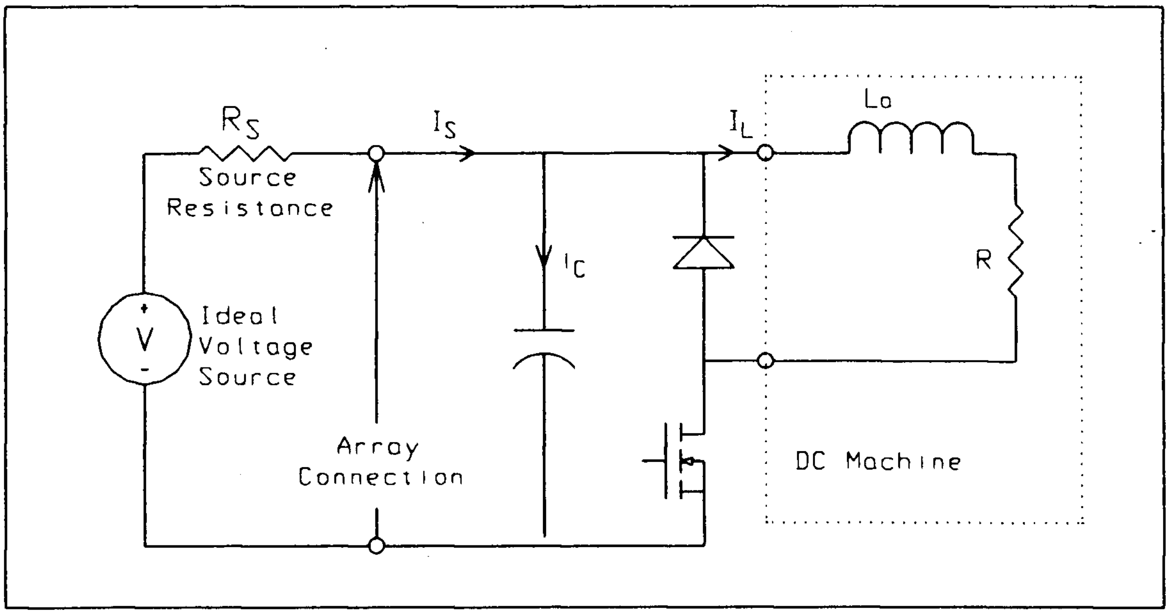

Figure 5.17: Simplified Power Circuit

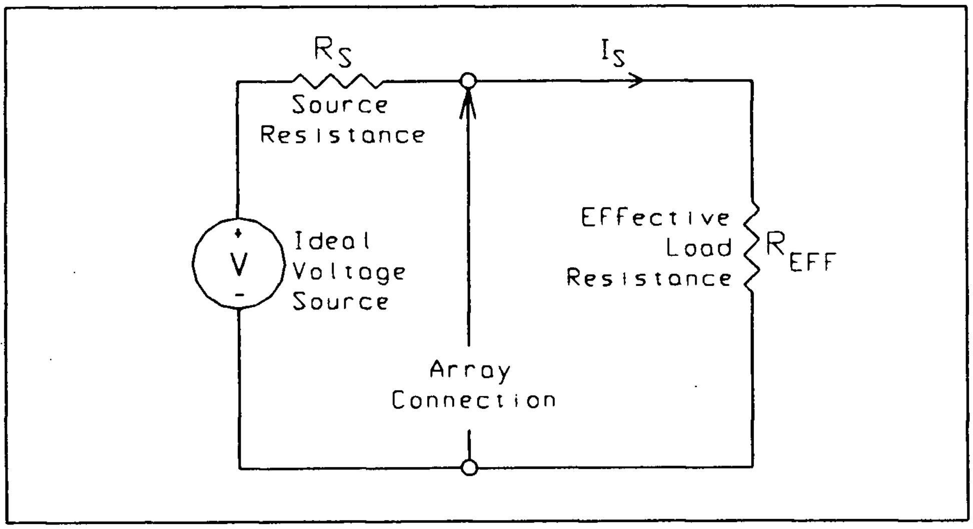

Figure 5.18: Equivalent Power Circuit

$$\begin{aligned}\frac{\Delta V_A}{\Delta t_{ON}} &= -\frac{R_S I_L}{T} \\ \Delta t_{ON} &= -\frac{\Delta V_A T}{R_S I_L}\end{aligned} \tag{5.19}$$

From the maximum power transfer theorem, it is known that $R_S = R_{EFF}$ at the maximum power point. Therefore:

$$R_S = R_{EFF} = \frac{V_A}{I_S}. \tag{5.20}$$

Substituting Equations 5.20 and 5.17 into Equation 5.19 yields:

$$\Delta t_{ON} = -\frac{\Delta V_A t_{ON}}{V_A}. \tag{5.21}$$

All the quantities on the right hand side of Equation 5.21 are either known, can be measured, or can be easily calculated. The magnitude of the change in the on time, Δt_{ON}, is therefore easily determined.

Some of the assumptions made in the above equations may be either crude or do not hold true away from the maximum power point. The actual system contains a non-linear source impedance, and the usual equivalent circuit of a DC machine includes a back electromotive force, EMF, which is proportional to the machine speed. However, it is standard practice to linearize a non-linear system around an operating point. Also it is valid to lump the back EMF of the DC machine into an equivalent resistance, if the current through the machine is constant. In any case, there are no strict constraints on the performance of the control system. All that is necessary is that the operating voltage be held constant and that the system remain stable. These criteria are easily met by this control scheme.

5.2.3 Interrupts

The hardware arbitrated priority structure of the MC68HC11 assists in coordinating the various tasks. At any time three interrupt requests are active. One to service the pulse width modulation routine, one to service the real time clock, and one to service the overcurrent routine.

The main program receives the lowest priority and can be interrupted at any time. Timing within the main program is coarse, and not critical, so that interrupt requests can be easily serviced without interfering with the logical flow of the program.

- **Pulse Width Modulation**

  The routine servicing the pulse width modulation interrupt receives the highest priority. This is achieved by appropriately setting the *HPRIO, highest priority I interrupt register.* It is critical that the output compare registers be updated before the timer count exceeds the updated compare register count. Otherwise gate voltage transitions will be lost and the power mosfets will be turned either off or on for a prolonged period of time.

  An interrupt routine can only be serviced after the CPU has completed the present instruction. To reduce the maximum response time to an interrupt request, a wait instruction, WAI, is placed before each instruction which requires a long time to complete. The wait instruction saves the register state and halts the program execution until an interrupt request is received. In this way the longer instructions, such as divide, FDIV, and multiply, MUL, are executed just after the interrupt routine has been serviced and it is very unlikely that another interrupt request would be generated while these instructions are being executed.

- **Overcurrent Protection**

The routine which services the overcurrent interrupt request receives the next highest priority. Response time to this routine is not critical as the hardware detector protects the power mosfets. This routine serves the purpose of clearing the fault and acknowledging its receipt.

First the routine calls for a delay and then checks to see if the fault persists. If the fault persists the duty cycle is reduced, another delay is called for, and the overcurrent input line is read. This procedure continues until the fault is at last cleared.

Before returning to the main program a flag bit is set to signal that an overcurrent fault has occurred. This flag bit is checked each time an increase in the duty cycle is called for. If the flag bit is set, an overcurrent counter is incremented before clearing the flag and increasing the duty cycle. If the overcurrent counter indicates a persistent fault, ie. more than six faults have occurred in a given period, then the converter is shut down for a period of half an hour.

- **Real Time Clock**

It is convenient to use a real time clock to synchronize events within the program. Delays ranging from a fraction of a second to several days can be handled with the aid of a real time clock.

The clock is derived from the free running timer. As the timer overflows an interrupt request is generated. The interrupt service routine is then able to update the *time of day* node by a fraction of a second. The *second*, *minute*, *hour* and *day* counters are appropriately updated as required. More details as to the updating process are given in the program listing.

To create a delay, the program must first access a timer node. The delay is added to the *time of day* and the result is placed in the timer node. The timer node is periodically compared with the *time of day* until they match. The delayed procedure is then executed.

Several delays can be handled simultaneously. Timer nodes may be sequentially tested until a match is found. The procedure associated with the matched node is then executed and the timer node released. This timer coordination is handled by the program. To handle a more complex coordination problem a special operating system could be written. However this is beyond the scope of this project.

5.2.4 Changing the Duty Cycle

Although it is possible to make large, sudden changes to the duty cycle, it is more desirable to make changes slowly and smoothly. In this way large transient currents, due to the filter capacitor discharging, are avoided. In any case the motor together with the pump have a large inertia and cannot change speed quickly.

The subroutine *CHANGE* ramps up or down the duty cycle in small steps. A minimal change, $0.5\mu s$, is made to one of the two sets of pulse streams in each step. The pulse width modulation process then continues for several cycles before the next small change is made. The pulse width is changed again until either the desired value is reached or the converter is turned fully *on* or *off*.

5.3 Results

The program is able to handle the multiple simultaneous tasks well. The pulse width modulation process continues without interruption, the maximum power point is located within 1% of the optimum value, and the voltage tracking routine is both accurate and stable.

5.3.1 Voltage Waveform

The drain-to-source voltage waveform is displayed in Figure 5.19. There is some ringing evident as the power mosfets switched off, but this is not important as the peak voltage is not very high. A magnified view of the drain-to-source voltage as the converter is switched off is displayed in Figure 5.20 while the drain-to-source voltage during turn-on is displayed in Figure 5.21. In both cases the switching is fast and the peak voltage is not significantly higher than the input terminal voltage.

5.3.2 Current and Voltage Measurements

The accuracy of the power tracking routine is only as accurate as the voltage and current measurements made by the microprocessor, as the power level is calculated from this voltage-current product. The load current and voltage are also calculated from the supply current and voltage .

The accuracy of the current, voltage and power measurements are verified by comparing the values obtained by the microprocessor with actual measurements. Each measurement is taken with a fixed duty cycle, supply voltage, and load impedance. To obtain several data points the duty cycle is varied, while the other variables are held constant. Measurements made by the microprocessor are compared with the measurements obtained from the voltage and current meters.

The signal read by the microprocessor is not calibrated, but in this application this is not important as only the relative current signal is of interest. All that is required is that an increase in the supply current be accompanied by a proportionate increase in the measured current. However, if it were required, it would not be difficult to calibrate the current and voltage signals.

The instantaneous current signal through the power mosfets is shown in Figure 5.22.

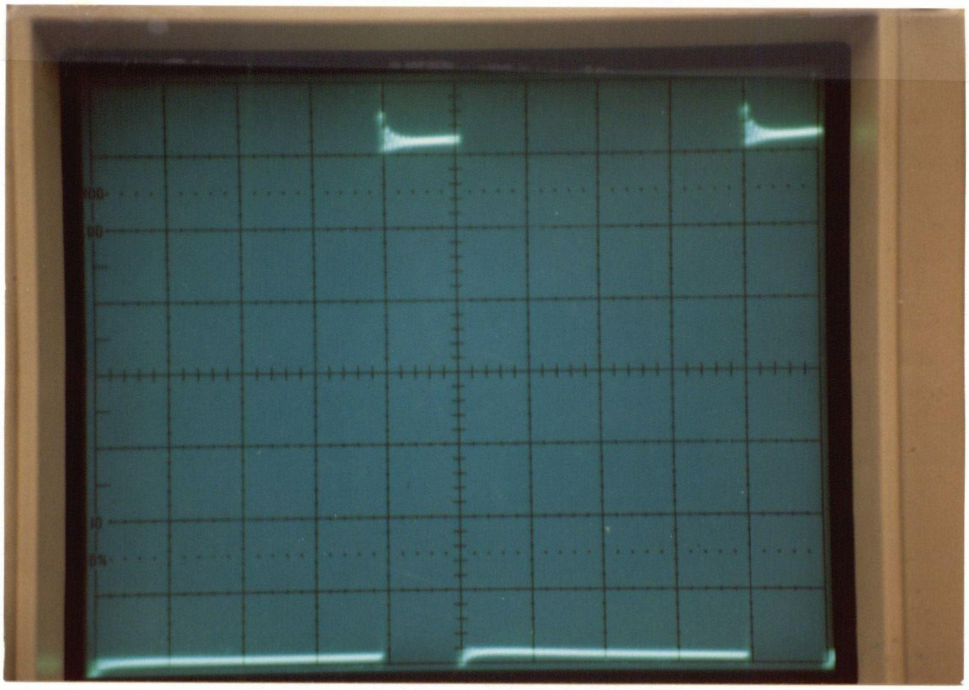

Figure 5.19: Drain-to-Source Voltage Waveform

V_{IN} :	145V	V_{OUT} :	111V	20V/div.
I_{IN} :	4.9A	I_{OUT} :	6.1A	10.0ms/div.

Apart from the ringing as the devices are first switched on the signal is an accurate representation of the current. The ringing, induced by the drain-to-source capacitance of the isolated current sensing cells, is removed by a low pass filter. The signal, shown in Figure 5.23, is then suitable to be used for the cycle-by-cycle overcurrent protection. Figure 5.24 displays the filtered current signal when the overcurrent protection is active.

The supply current measured by the A/D converter is compared with the meter-measured current in Figure 5.25. The linear relationship between the two methods of measuring current is evident, even though the temperature of the power mosfets is changing as the current through the device increases.

The load current is derived by the microprocessor by dividing the supply current by the duty cycle. The measured load current is compared with the derived load current

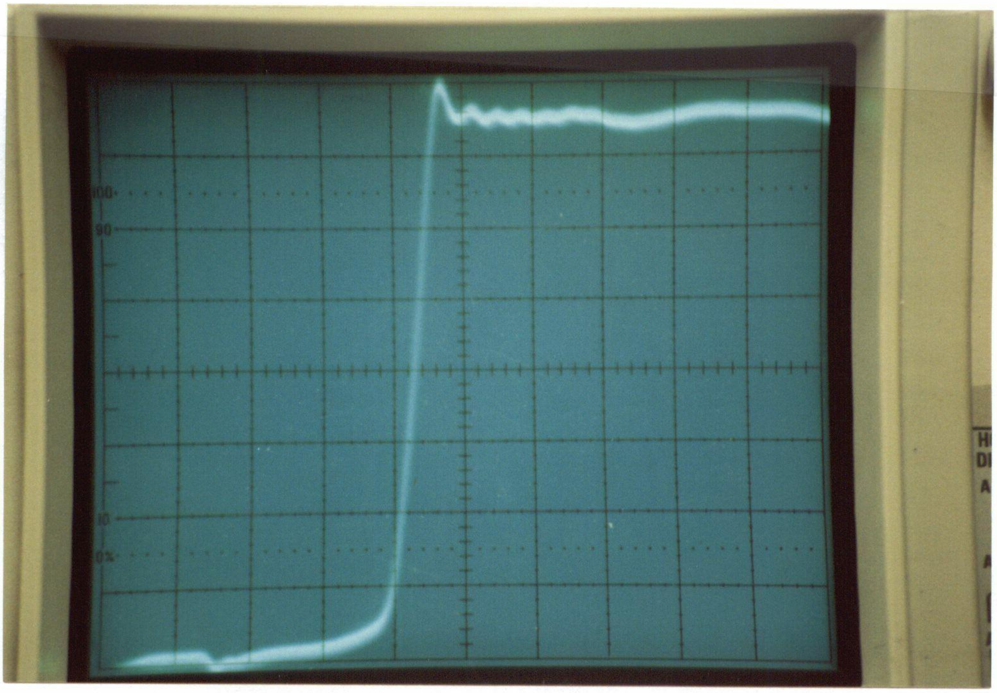

Figure 5.20: Drain-to-Source Voltage Waveform at Turn-off

V_{IN} :	145V	V_{OUT} :	111V	20V/div.
I_{IN} :	4.9A	I_{OUT} :	6.1A	0.02μs/div.

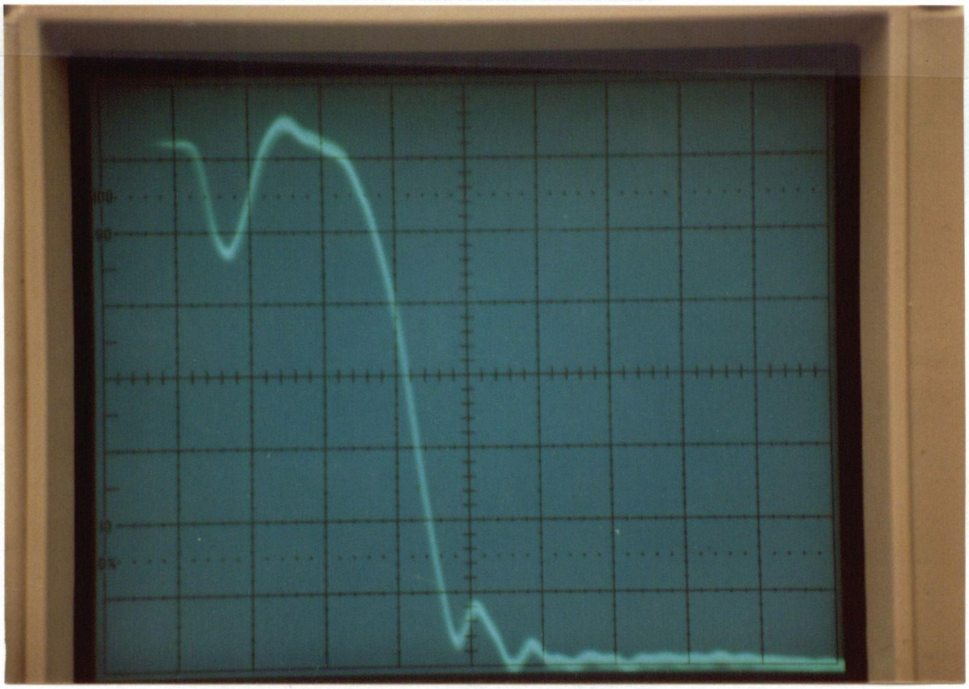

Figure 5.21: Drain-to-Source Voltage Waveform at Turn-on

V_{IN} :	145V	V_{OUT} :	111V	20V/div.
I_{IN} :	4.9A	I_{OUT} :	6.1A	0.02μs/div.

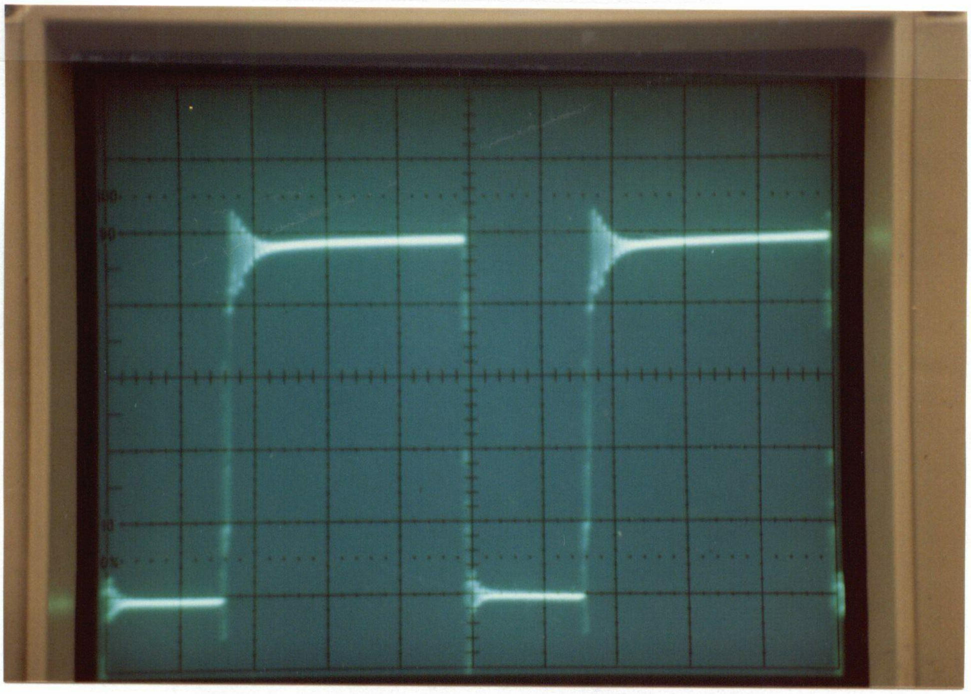

Figure 5.22: HEXSense Current Waveform

V_{IN} :	145V	V_{OUT} :	90.7V	1V/div.
I_{IN} :	6.37A	I_{OUT} :	9.4A	10.0ms/div.

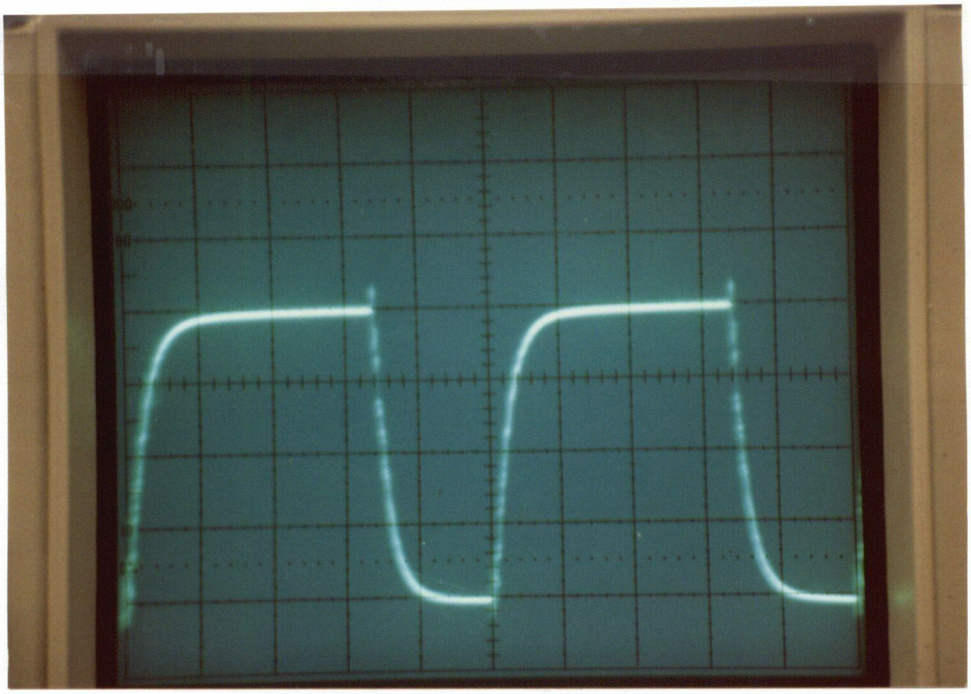

Figure 5.23: Filtered HEXSense Current Waveform

V_{IN} :	145V	V_{OUT} :	90.7V	1V/div.
I_{IN} :	6.37A	I_{OUT} :	9.4A	10.0ms/div.

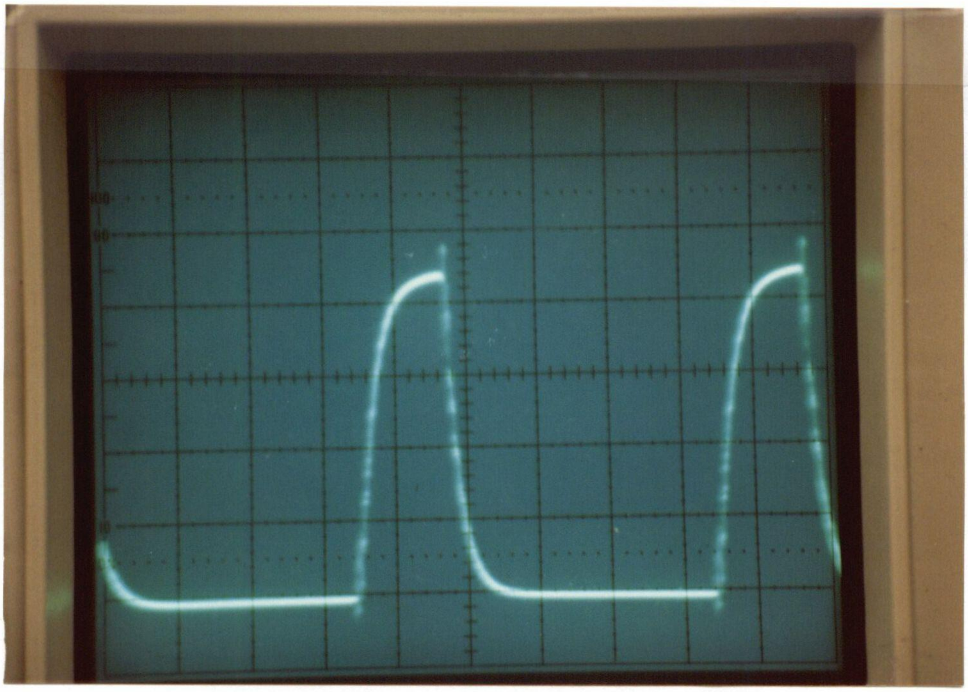

Figure 5.24: Current Waveform with Active Overcurrent Protection

V_{IN} :	232V	V_{OUT} :	52.9V	1V/div.
I_{IN} :	2.6A	I_{OUT} :	10.4A	10.0ms/div.

in Figure 5.25. The calculated load current is accurate within a few percent except when the duty cycle is very small. The point on the load current curve of Figure 5.25 with a significant error is calculated at a 2% duty cycle, where a quantization error in the supply current is magnified 50 times when the load is calculated.

The supply voltage measured by the microprocessor is compared with the supply voltage read from a voltmeter in Figure 5.26. The calculated and measured load voltages are also displayed. It is evident that both the calculated load voltage and the supply voltage measurements obtained by the microprocessor are accurate. The *bump* that is observed in the region between the load and supply voltage measurements is due to losses in the converter which are unaccounted for in the computation of the load voltage. Other small discrepancies are due to fluctuations in the supply voltage.

The power calculated by the microprocessor is compared with the measured input and output power in Figure 5.27. The measured power is the product of the measured voltage and current. There is a good correspondence between the two methods of measuring the power. There is a slight difference between the input power and the power delivered to the load. The difference between these two curves is the converter losses. There are also inaccuracies in the measurements due to the meters and variations in the supply voltage.

5.3.3 Maximum Power Point

The controller attempts to maximize the power delivered to the load. To test the accuracy of the power tracker the following test circuit was constructed: A stiff, adjustable DC source was derived from the rectified output of a three-phase variac. The source impedance of the photovoltaic array is simulated by an adjustable power resistor. The source resistance remains fixed for any given set of measurements. The load is made

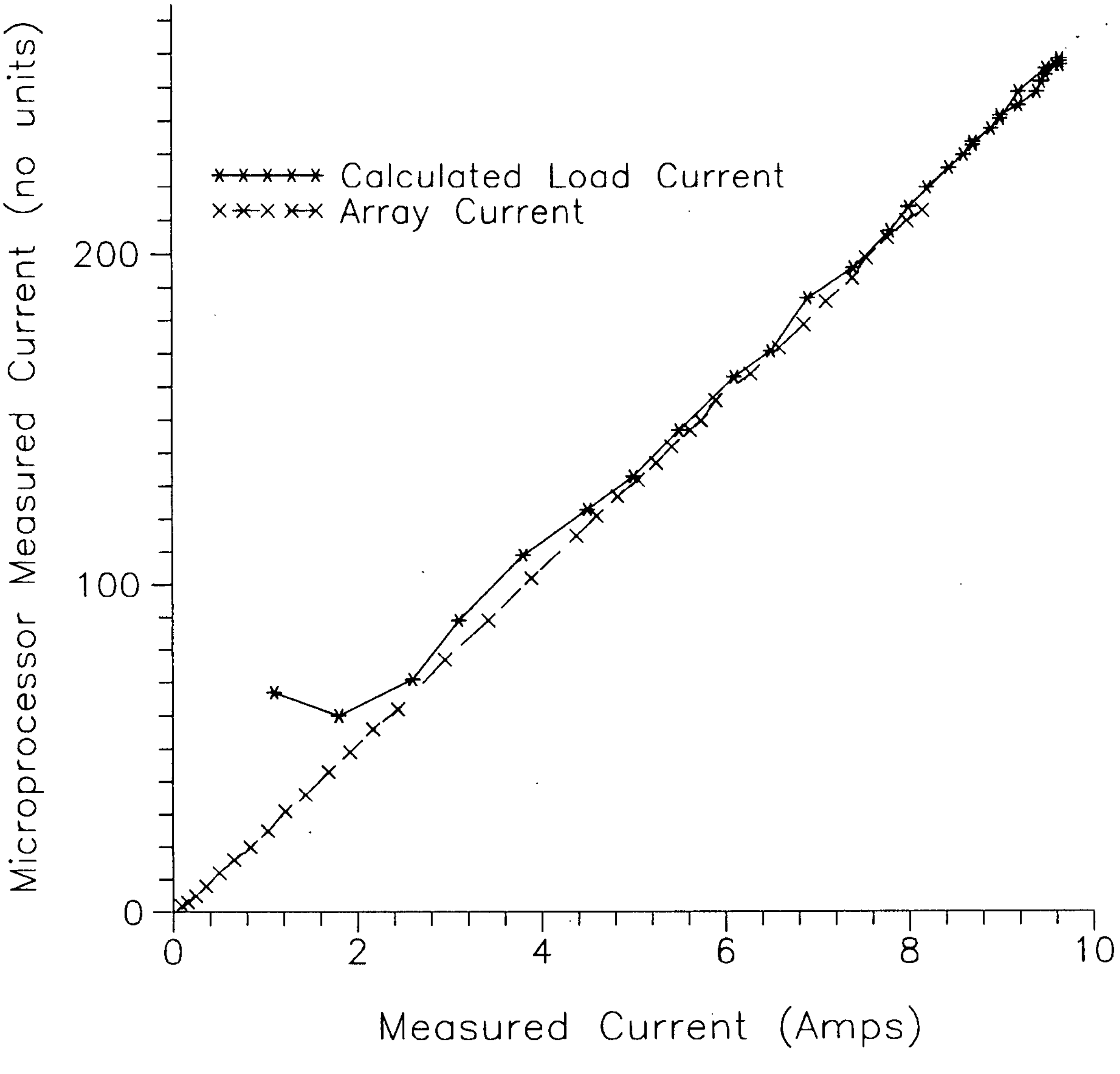

Figure 5.25: Microprocessor Current Measurements vs Metered Current Measurements

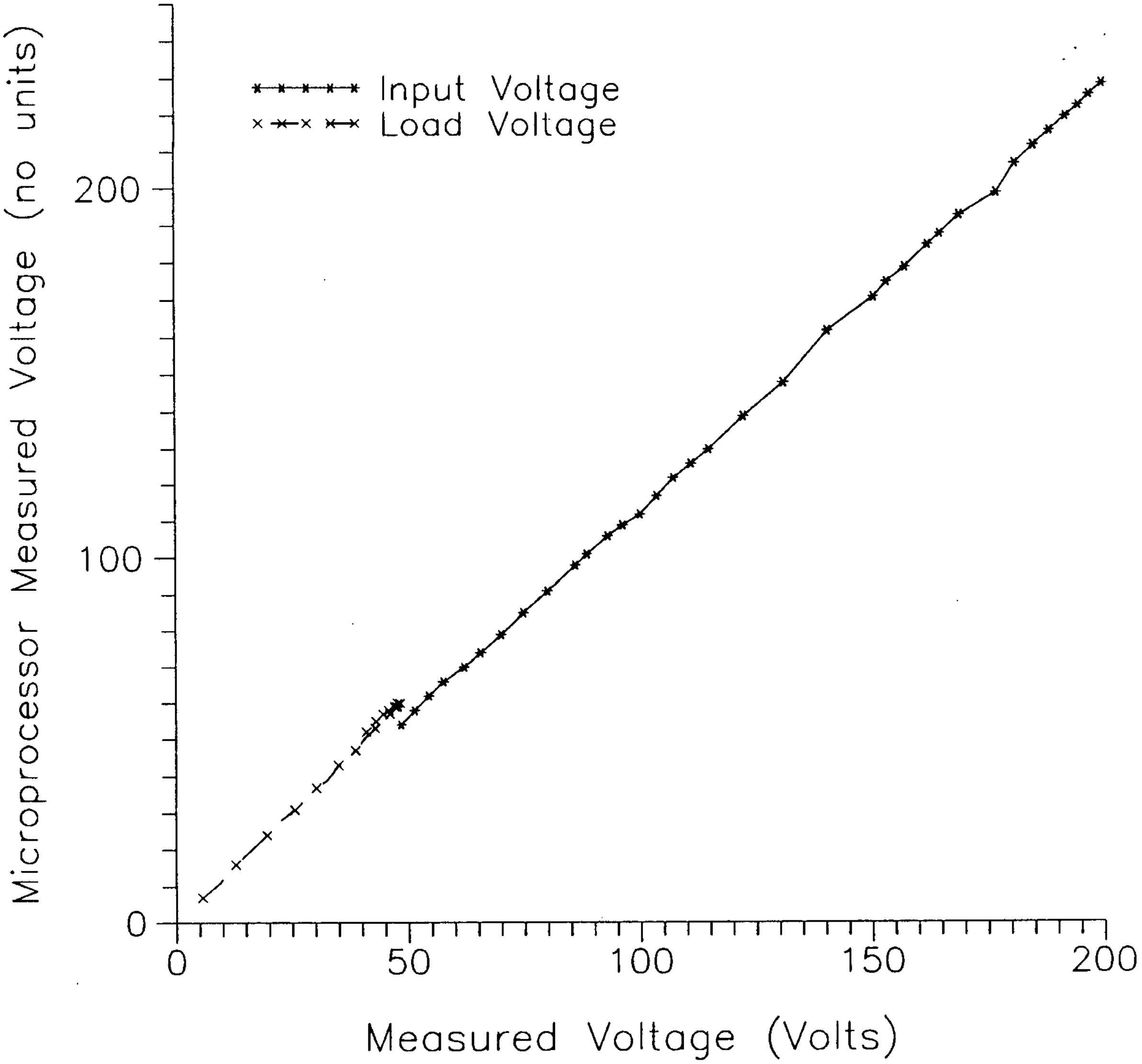

Figure 5.26: Microprocessor Voltage Measurements vs Metered Voltage Measurements

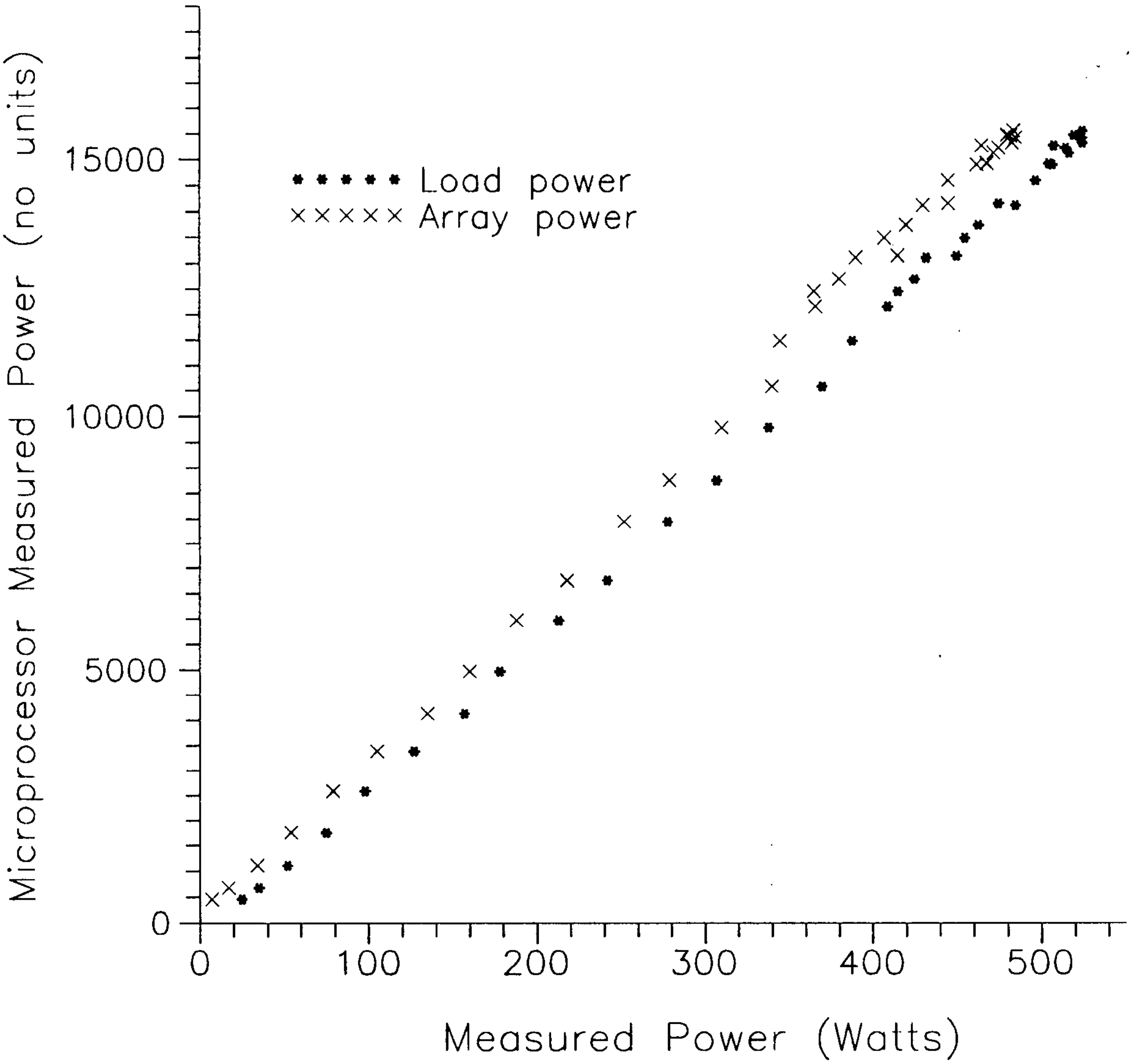

Figure 5.27: Calculated Power vs Metered Power

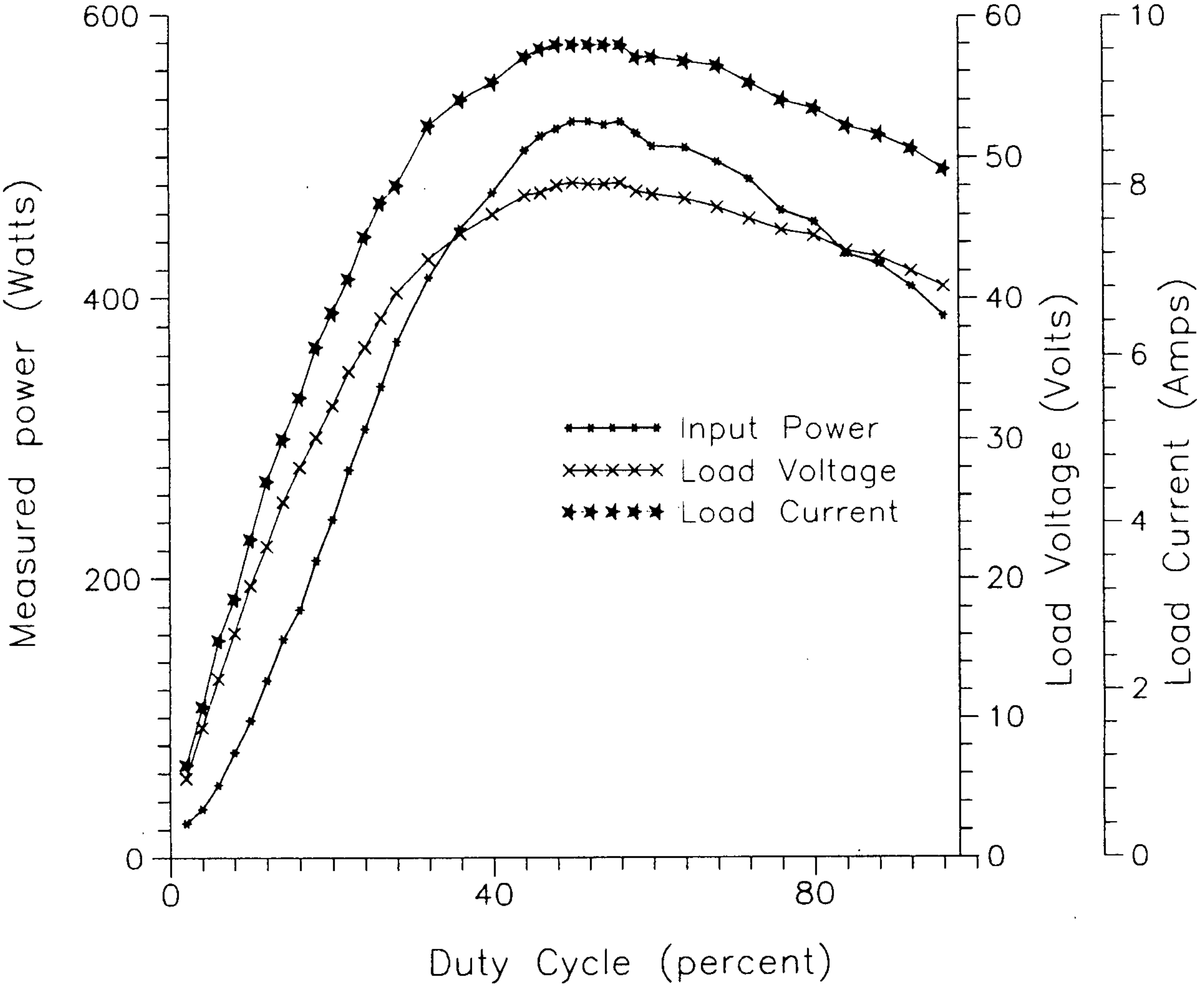

Figure 5.28: Power, Load Current, and Load Voltage vs Duty Cycle

up of a 11.5mH inductor in series with an adjustable power resistor which also remains fixed for a given set of measurements.

From the maximum power transfer theorem it is known that the load and source resistance should be equal at the maximum power point. Note that the effective load appears resistive to the source as both the voltage and current into the chopper are DC. The voltage at the terminals of the chopper will be half the source voltage when the effective load resistance equals the source resistance. The accuracy of the power tracking system is therefore tested by running the test circuit with various loads and open circuit voltages. These conditions are recorded and if the power tracker is accurate the voltage level at the terminals of the chopper will settle to one half the source voltage.

The test results are summarized in Table 5.3.3. The maximum power point is located within 1.0% in all cases and within 0.5% in most cases. The operating voltage which the converter settles upon is located within a few volts of the optimum value. Near the maximum power point the power delivered to the load changes only slightly as the operating voltage is varied. Figure 5.28 displays how the input power, load voltage and load current change as the duty cycle is varied from 2% to 98%. The load current and load voltage curves are quite flat near the peak, which occurs at the same point as the peak on the power curve.

As a final test the converter is set up to run a DC machine which in turn drives an induction generator. The same stiff DC source together with a series power resistor is used to simulate a photovoltaic array. The test results were encouraging. The starting torque is easily overcome, the maximum power point is located within 1.0% and the voltage tracking is stable and accurate. The converter meets all requirements.

Table 5.4: Power Tracking Results

Open Circuit Voltage	Operating Voltage	Input Power	Load Voltage	Output Power	Efficiency η	Maximum Input Power	Power Tracking Error
120V	55.7V	202W	42.3V	192W	95%	203W	0.49%
120V	57.8V	202W	29.8V	186W	92.1%	202W	0.0
150V	71.4V	307W	52.3V	294W	95.8%	310W	0.97%
151V	71.5V	309W	42.8V	290W	93.9%	311W	0.64%
175V	88.7V	257W	65.9V	247W	96.1%	258W	0.39%
175.8V	91V	406W	60.3V	386W	95.1%	410W	0.98%
175.8V	88.1V	410W	49.3V	384W	93.6%	410W	0.0
201V	101V	530W	62.1V	500W	94.3%	530W	0.0
201V	98.6V	335W	75.4V	323W	96.4%	335W	0.0
201V	96V	532W	69.1V	505W	94.9%	532W	0.0
225V	109.7V	659W	76.4V	640W	97.1%	660W	0.15%
225.6V	104V	414W	60.6V	386W	93.2%	415W	0.24%
225V	109.7V	659W	76.4V	640W	97.1%	660W	0.15%
250V	117V	511W	93.5V	492W	96.3%	512W	0.20%
251V	123V	518W	72.7V	492W	95.0%	520W	0.38%
250.6V	125V	512W	66.0V	475W	92.8%	512W	0.0
300V	144V	731W	112.4V	710W	97.1%	735W	0.54%
304V	144V	940W	127.7V	910W	96.8%	942W	0.21%
301V	146V	925W	90.9V	870W	94.0%	930W	0.54%

Chapter 6

Conclusions

The amount of water delivered by a photovoltaic powered pumping system can be maximized with the aid of a one quadrant DC to DC converter. The converter is capable of matching the combined motor-pump characteristics to the characteristics of the photovoltaic array for maximum power transfer under most lighting conditions.

Three schemes of controlling the DC to DC converter are presented. The simplest control scheme involves fixing the photovoltaic array voltage at some manually set reference level. Close to maximum power is delivered to the load under most insolation levels and the control scheme is stable, even under fluctuating lighting conditions. Only a simple analogue circuit is required.

A true power tracking control scheme can also be used to maximize the power output of the DC to DC converter. This controller finds and maintains the best operating point automatically at the expense of a more complex logic circuit. Maximum power output will be maintained, even if the photovoltaic array characteristics drift with temperature and age. Maximizing the output is important because the photovoltaic panels are expensive. A small increase in the power delivered to the load will result in an even greater increase in overall efficiency, as both the pump and DC motor operate more efficiently at higher speeds and power levels. One drawback is that the controller can temporarily drift away from the optimum operating point if the insolation levels are fluctuating.

To maintain the advantages of both the power tracking and the voltage tracking control schemes a microprocessor based hybrid scheme is developed. A single chip microcomputer provides most of the logic functions of the controller when used with only a few external chips. During the power tracking mode, the optimum operating point is automatically located and adjusted. Many data points are sampled and averaged to increase the accuracy of the search. The majority of the time is spent in the voltage tracking mode. In this mode the array voltage is held constant even as the insolation levels fluctuate.

These basic functions have been implemented and tested. But perhaps the most important advantage of the microprocessor based control scheme is its flexibility. More sophisticated features can be added with little or no increase in the complexity of the hardware. For instance, it should be possible to identify a dry well and quickly shut down the converter preventing a catastrophic failure. This feature, although easily implemented, was not incorporated due to inadequate test facilities.

Another possibility is that the microprocessor based system could act as a data logger, keeping a record of the power level, motor voltage and current, ambient temperature, etc., for future reference and study. This information, along with any fault alarms, could conceivably be transmitted to a central location to be recorded and speed the detection and repair of faulty equipment.

The extra cost involved in developing and manufacturing the microprocessor based system would be justified by the extra flexibility and protection the system would provide.

Appendix A

Assembly Language Program

```
       ORG    $00
***************************************************************************
*
*      Hybrid Power Tracking, Voltage Tracking Controller
*
***************************************************************************
*
*       This assembly language program is targeted for the Motorola
*       MC68HC11A8 HCMOS single-chip microcomputer.
*
***************************************************************************
*
*       This program controls the pulse width of the gating pulses applied
*       to the gates of a set of parallel power mosfets.
*
*       The controller has two modes of operation. In the power tracking
*       mode the duty cycle is slowly increased from its minimum value
*       in a search for the maximum power point. Once this point has been
*       found the voltage tracking mode is entered. In this mode the
*       operating is held steady at the value corresponding to the maximum
*       power point.
*
*****************************************************************************
*
*      Label some memory locations to hold useful variables
*
PERIOD RMB  2       Memory location to hold the period.
ONCNT1 RMB  2       Memory location to hold the ON count one.
OFCNT1 RMB  2       Memory location to hold the OFF count one.
ONCNT2 RMB  2       Memory location to hold the ON count two.
OFCNT2 RMB  2       Memory location to hold the OFF count two.
STEP   RMB  1
FLG    RMB  1       Define a user flag register.
MAXCUR RMB  1       Register to hold the short circuit current.
OPEN   RMB  1       Register to hold the open circuit voltage.
```

```
CURR   RMB  1       Register to hold the operating current.
MOTCUR RMB  2       Register to hold the load current.
VOLT   RMB  1       Register to hold the operating voltage.
MOTVLT RMB  1       Register to hold the average load voltage.
POWER  RMB  2       Register to hold the operating power.
PWRMAX RMB  2       Register to hold the maximum operating power.
OFFSET RMB  1       Register to hold the current offset.
CORECT RMB  1       Register to hold the current correction factor.
MAXVLT RMB  1       Maximum power voltage.
COUNT  RMB  1       Reserve a space for a counter.
TEMP   RMB  2       Define a space for a temporary variable.
TEMP2  RMB  1
TMPVT1 RMB  1
TMPVT2 RMB  1
*
*
*      Label  memory locations to hold the timer variables.
*
FRCSEC  RMB  2      Holds the fraction of a second count.
SECOND  RMB  1      The second counter.
MINUTE  RMB  1      The minute counter.
HOUR    RMB  1      The hour counter.
DAY     RMB  1      The day counter.
*
TMENOD  RMB  6      A 6 byte timer node.
TMNOD2  RMB  6      A 6 byte timer node.
*
*                   The last item is memory reserved for an array.
*                   Although there is only one byte reserved the array
ARRAY  RMB  1       can grow as far as the stack boundary.
*
MSKOC1 EQU  $80     Mask for the OC1 flag and mask registers.
MSKOC2 EQU  $40     Mask for the OC2 flag and mask registers.
*
*      Define the software flag register bits.
*
STRTUP EQU   1      Bit one signals pulse width modulation start-up.
ONGR   EQU   2      When set, signals the ON time is greater than
*                   the OFF time.
UP     EQU   4      When set, indicates an increasing pulse width.
DECRS  EQU   8      When set, indicates a decreasing power level.
FRST   EQU   $10    Bit to indicate the initiation of the maximum power
```

```
*                       tracking routine.
OVRCUR EQU    $20       Over-current flag bit.
*
*      Define some other useful constants.
*
RDCURR EQU    1
RDVLT  EQU    2
OFF    EQU   $01
ON     EQU   $02
RUN    EQU   $03
CLRCUR EQU   $04
MIN    EQU   $2
DLAY   EQU    1         One second delay.
DELAY  EQU   $0600      Fraction of a second delay.
*
*      Define the I/O port locations
*
PORTA  EQU   $1000        I/O port A
PORTB  EQU   $1004        I/O port B
PORTC  EQU   $1003        I/O port C
PRTCL  EQU   $1005        Alternate latched port C.
DDRC   EQU   $1007        Data direction for port C.
PIOC   EQU   $1002        Parallel I/O Control Register.
PORTD  EQU   $1008        I/O port D
*
*
*
```

A.1 Main Program

```
       ORG   $E000
*
*
*
BEGIN  SEI                 Inhibit interrupt requests.
       LDS   #$00FF
*
       LDY   #PORTB
       BCLR ,Y OFF         Turn the power MOSFETS off.
       BSET ,Y ON
       LDD   #$0064        Load a number representing the period.
```

```
      STD  PERIOD
      LSRD         Divide down the period to form the initial
      LSRD         step size to perturb the pulse width.
      LSRD
      STAB STEP
      LSRD
      ADDB STEP
      STAB STEP
*
*     Set up the overcurrent detector.
*
       JSR  OCRSET
       CLRB
       STD  PWRMAX
*
*      Initialize the A/D converter.
*
       BSR   ADSET
*
*      Initialize the timer.
*
       BSR   TIMER
       CLI          Enable the interrupt system.
*
*      Determine the open circuit voltage
*      and offset current.
*
       JSR   ADJOPN
*
*      Find the maximum power point.
*
LOOP   JSR   MAXPWR
*
*      Place the converter in the voltage tracking mode.
*
       CLR   COUNT    Clear the counter which records the number of
*                     overcurrent incidents.
LOP1   LDX   #TMENOD  Clear the timer node.
       JSR   CLRTME
       LDAB  #30      Add 30 minutes to the time.
       STAB  3,X
       JSR   ADDTME
```

```
LOP2    JSR   VLTRAK    Call the voltage tracking routine.
*
*       Test for overcurrent and branch if clear.
*
        BRCLR FLG OVRCUR LOP2B
*
*       This section is entered if an overcurrent fault has been detected.
*
        LDAA  COUNT
        INCA            Increment the overcurrent counter.
        BCLR  FLG OVRCUR  Clear the overcurrent flag.
        CMPA  #7
        BNE   LOP2A
        JMP   OVRDLY    Exit if the number of overcurrent occurrences = 7.
LOP2A   STAA  COUNT
*
LOP2B   PSHX            Save the timer node address.
        LDX   #TMNOD2
        JSR   CLRTME    Clear the new timer node.
        LDD   #DELAY    Add a fraction of a second to the time
        STD   ,X        of day and place the result in the timer node.
        JSR   ADDTME
LOP3    JSR   CMPTME    Compare the timer node to the current time.
        TSTA
        BNE   LOP3      Loop if time is not up.
        PULX            Restore the first timer node address.
        JSR   CMPTME    Compare to the current time.
        TSTA
        BNE   LOP2      Loop again if time is not up.
*
*       Adjust the maximum power tracking voltage.
*
        LDD   PERIOD
        LSRD
        LSRD             Redefine the step size.
        LSRD
        STAB  STEP
        BRA   LOOP      Find the maximum power point again.
```

A.2 A/D Converter Set-Up

```
*
***************************************************************
*
*                    SUBROUTINE ADSET
*
*     Subroutine ADSET sets up the A/D converter to read in data
*     from all four data lines under program control.
*
*     INPUT and OUTPUT: NONE
*
*     REGISTERS EFFECTED:  The CC and A register.
***************************************************************
*
*
*      Define some memory locations associated with the A/D
*      converter.
*
ADCTL  EQU  $1030     A/D Control/Status register.
OPTION EQU  $1039     Configuration options register.
ADR1   EQU  $1031     A/D Result register 1.
ADR2   EQU  $1032     A/D Result register 2.
ADR3   EQU  $1033     A/D Result register 3.
ADR4   EQU  $1034     A/D Result register 4.
ADDAT  EQU  $10       A data byte to configure the ADCTL to read all
*                     four channels under program control.
ADSET  LDAA OPTION
       ORAA #$80      Set the A/D power up bit in the options reg.
       STAA OPTION
       LDAA #ADDAT    Configure the A/D converter to read lines one
       STAA ADCTL     through four under program control.
       RTS
```

A.3 Timer Set-Up

```
*
*
***************************************************************
*
*                    SUBROUTINE TIMER
*
*     Subroutine TIMER sets up the free running timer.
*     from all four data lines under program control.
*
*     INPUT and OUTPUT: NONE
*
*     REGISTERS EFFECTED:  The CC and A register.
**************************************************************
*
*
*      Define some memory locations associated with the timer.
*
TIME   EQU  $100E     16 bit free running timer address.
REGOC1 EQU  $1016     Output compare register one.
REGOC2 EQU  $1018     Output compare register two.
REGOC3 EQU  $101A     Output compare register three.
REGOC4 EQU  $101C     Output compare register four.
TMSK1  EQU  $1022     Main timer interrupt mask register 1.
TCTL1  EQU  $1020     Timer control register 1 address.
OC1M   EQU  $100C     Output compare 1 mask register address.
OC1D   EQU  $100D     Output compare 1 data register address.
CFORC  EQU  $100B     Timer compare force register.
TMSK2  EQU  $1024     Timer interrupt mask register 2.
TFLG2  EQU  $1025     Timer interrupt flag register 2.
TFLG1  EQU  $1023     Timer interrupt flag register 1.
*                     Specify the timers' mode of operation.
*
TIMER  LDAA #$F8      Set the timer control register to force OC2 and
       STAA TCTL1     OC3 high, and OC4 low after a successful compare.
       LDAA #$FF      Specify  A3-A7 to be effected by a successful
       STAA OC1M      OC1 compare in the output compare mask register.
       LDAA #$10      Set the compare 1 data register to force OC2,OC3
       STAA OC1D      low and OC4 high after a successful OC1 compare.
       LDAA #$F8      Set the timer compare force register to force
       STAA CFORC     A3-A7 low.
*
*      Set the timer registers to zero.
*
       LDX  #FRCSEC
       JSR  CLRTME
       LDAA  TFLG2    Clear any pending overflow interrupts.
       STAA  TFLG2
       LDAA  #$80     Enable the overflow interrupt request by setting
       STAA  TMSK2    timer mask register bit.
       LDAA #0       Set the FLG register to indicate a start up condition.
       STAA FLG
       RTS
```

A.4 Overcurrent Initialization

```
*
*
****************************************************************
*
*                    SUBROUTINE OCRSET
*
*     Subroutine OCURST sets up the overcurrent detector.
*
*     Both flip flop clocks are strobed to place the detector
*     in a known state. Also internal registers are set
*     to receive an overcurrent interrupt request.
*
*     INPUT and OUTPUT: NONE
*
*     REGISTERS EFFECTED:  CC.
****************************************************************
*
*
*       Strobe the output of PORTA to ensure overcurrent
*       flip flop one latches to a high value.
*
OCRSET  PSHX
        PSHA
        LDAA PIOC
        LDAA PRTCL     Clear any pending interrupt flags.
        LDX  #PORTA
        BCLR ,X  $40
        BSET ,X  $40
        BCLR ,X $40
*
        LDX  #PORTB
        BSET ,X  4     Strobe the clock of overcurrent flip flop 2.
        BCLR ,X  4
        BSET ,X  4
        BCLR ,X  4
```

```
*
        LDAA #0
        STAA DDRC      Set up port C as an input port.
        LDX  #PIOC
        BCLR ,X $2     Activate an interrupt request upon a falling
*                      edge of STBA.
        BSET ,X $40    Set the interrupt enable mask.
        PULA
        PULX
        RTS
```

A.5 Read Input Data

```
*
*****************************************************************
*
*                   SUBROUTINE READ
*
*     Subroutine READ reads in the data from a single channel
*     of the A/D converter. The data is read in four times and
*     averaged.
*
*     INPUT:      Register A is input with the number of the
*                 A/D line to be read.
*                 none
*
*     OUTPUT:     Register A is returned with a value of:
*                     0: Successful read.
*                    -1: Illegal input data.
*
*                 Register B is returned with the average
*                 of four A/D conversions.
*
*     REGISTERS EFFECTED:  A,B and CC registers.
*****************************************************************
*
READ    PSHX
        CMPA  #3        Test for illegal input data.
        BHI   ERR1
        PSHA
LQ2     STAA ADCTL      Initiate an A/D conversion process.
```

```
        LDX   #ADR1
LQ3     LDAA ADCTL      Wait for valid data.
        BPL  LQ3
*
*       Test the data to ensure the four readings do not vary more
*       than one bit.
*
LQ3B    LDAB  ,X
        SUBB  1,X       Compare two readings.
        BHS   LQ3C      Form the absolute vale of the difference.
        NEGB
LQ3C    CMPB  #1
        BLS   LQ3D      If the readings vary more than one bit.
        TSX                 Fetch another set of readings.
        LDAA  ,X
        BRA   LQ2
*                       Else the two readings vary less than one bit.
LQ3D    INX
        CMPX  #ADR4     All for readings tested?
        BNE   LQ3B        If not, test the next pair.
*
*       Average the four readings.
*
        PULA
        CLRA
        STAA TEMP2
        LDAB ADR1   Add the four A/D conversion results and
        ADDB ADR2   average them.
        BCC  LQ4
        INCA
LQ4     ADDB ADR3
        BCC  LQ5
        INCA
LQ5     ADDB ADR4
        BCC  LQ6
        INCA
LQ6     LSRD        Divide the result by two.
        BCC  LQ7
        INC  TEMP2
LQ7     LSRD        Divide by two again.
        BCC  LQ8
        INC  TEMP2
```

```
LQ8     LDAA #1     Check if both discarded bits from the averaging
        SUBA TEMP2  process were set.
        BCC  LQ9    If so increment the result.
        INCB
LQ9     CLRA        Signal a successful A/D conversion.
        PULX
        RTS
ERR1    LDAA  #$FF      Signal illegal input data.
        CLRB
        PULX
        RTS
```

A.6 Adjust the Pulse Width

```
*
*******************************************************************
*
*                    SUBROUTINE ADJST
*
*       Subroutine ADJST adjusts the pulse width of the modulated
*       output voltage waveform. The waveform is updated two pulses
*       at a time and the width of each of these two pulses can be
*       separately adjusted.
*
*       ASSUMPTION:
*             The  ON times input for the two pulse trains is valid and
*             strictly less than the period.
*
*       INPUT:
*          1) The FLG register is set to indicate which
*             compare line activates an interrupt request.
*                 a) FLG bit 0 = zero implies the waveform
*                    has not yet been started.
*                 b) FLG bit 1 = 1 implies the ON time is
*                    greater than the OFF time and OC1 is currently
*                    generating interrupt requests upon a successful
*                    compare.
*                 c) FLG bit 1 = 0 implies the OFF time is
*                    greater than the ON time and OC2 is currently
*                    generating interrupt requests upon a successful
*                    compare.
```

```
*
*          2) The ON time of the first pulse is input via
*             the D register.
*          3) The ON time of the second pulse is input via
*             the X register.
*          4) The period of the waveform is contained in the
*             memory location PERIOD.
*
*        OUTPUT:
*                The pulse width suitably modified and the FLG register
*                suitably set.
*
*        REGISTERS EFFECTED: CC
*
*************************************************************************
*
*
*       Define some constants.
*
HPRIO  EQU   $103C    Highest priority I interrupt register address.
HPOC1  EQU   $8       Data byte used to set OC1 as the highest priority.
HPOC2  EQU   $9       Data byte used to set OC2 as the highest priority.
*
ADJST  PSHY
       PSHX           Save the ON time for pulse two.
       PSHB           Save the ON time for pulse one.
       PSHA
*                     Branch if the pulse train has not yet been started.
       BRCLR FLG 1 START
*                     Test bit 2 of FLG to see if the ON time is greater
*                     Branch if OFF > ON.
       BRCLR FLG 2 OFFGR
*
*      This program section is entered when the FLG register indicates
*      that the ON time was previously greater than the OFF time.
*
*
*      Determine whether the new on or off time is greater.
*      If the on time is larger, there is no need to change the interrupt
*      source. If the off time is now larger the interrupt source
*      must be changed to OC2.
*
```

```
        XGDX            Place the ON time for pulse two in D reg.
        TSX
        ADDD  ,X        Add the ON time for pulse one.
        SUBD  PERIOD    Branch if the sum of the ON times is less than
        BCS   OC2SEL    the period.
        LDD   PERIOD
        SUBD  ,X        Calculate the new OFF time for pulse one.
        XGDY            Save the OFF time in the Y register.
        LDD   PERIOD    Calculate the OFF time for pulse two.
        SUBD  2,X
        WAI
        SEI             Disable interrupts while updating the ON and
        STY   OFCNT1    OFF registers.
        STD   OFCNT2
        LDD   ,X
        STD   ONCNT1    Save the ON time for pulse one.
        LDD   2,X
        STD   ONCNT2    Save the ON time for pulse two.
        CLI             Enable interrupts
        PULA
        PULB            Restore the register state.
        PULX
        PULY
        RTS
*
*       This program section is entered if the current ON time is greater
*       than the OFF time and the new OFF time is greater than the new
*       ON time.
*
OC2SEL  LDD   PERIOD
        SUBD  ,X        Calculate the new OFF time for pulse one.
        XGDY            Save the OFF time in the Y register.
        LDD   PERIOD    Calculate the OFF time for pulse two.
        SUBD  2,X
        PULX            Load the ON time for pulse 1.
        WAI
        SEI             Disable interrupts while updating the ON and
        STY   OFCNT1    OFF registers.
        STD   OFCNT2
        STX   ONCNT1    Save the ON time for pulse one.
        PULX
        STX   ONCNT2    Save the ON time for pulse two.
```

```
       LDD    REGOC1   Prepare to change the OC2 compare register to account
       ADDD   ONCNT1   for the new pulse width.
       LDX    #TFLG1
LQ10   BRCLR ,X MSKOC2 LQ10  Mark time until OC2 changes.
       STD    REGOC2
       LDAA   #MSKOC2   Set the OC2 bit of the timer interrupt mask register
       STAA   TMSK1     to select an interrupt request upon a successful
*                       OC2 compare.
       LDAA   #HPOC2    Raise OC2 to the highest priority interrupt request.
       STAA   HPRIO
       LDAA   TFLG1     Clear any pending interrupt requests
       STAA   TFLG1
       CLI              Enable interrupts
*                       Set the FLG register to show the OFF time is
       BCLR   FLG ONGR  greater than the ON time.
*
       LDD    ONCNT1   Restore the register state.
       LDX    ONCNT2
       PULY
       RTS
*
START  BRA STRT
*
*      This program section is entered when the FLG register indicates
*      that the OFF time was previously greater than the ON time.
*
*
*      Determine whether the new on or off time is greater.
*      If the off time is larger, there is no need to change the interrupt
*      source. If the on time is now larger the interrupt source
*      must be changed to OC1.
*
OFFGR  XGDX            Place the ON time for pulse two in D reg.
       TSX
       ADDD   ,X       Add the ON time for pulse one.
       SUBD   PERIOD   Branch if the sum of the ON times is less than
       BCC    OC1SEL   the period.
       LDD    PERIOD
       SUBD   ,X       Calculate the new OFF time for pulse one.
       XGDY            Save the OFF time in the Y register.
       LDD    PERIOD   Calculate the OFF time for pulse two.
       SUBD   2,X
       WAI
       SEI              Disable interrupts while updating the ON and
       STY    OFCNT1    OFF registers.
       STD    OFCNT2
       LDD    ,X
       STD    ONCNT1    Save the ON time for pulse one.
       LDD    2,X
       STD    ONCNT2    Save the ON time for pulse two.
       CLI              Enable interrupts
       PULA
       PULB             Restore the register state.
       PULX
       PULY
       RTS
*
*      This program section is entered if the current OFF time is greater
*      than the ON time and the new ON time is greater than the new
*      OFF time.
*
OC1SEL TSX
       LDD    PERIOD
       SUBD   ,X        Calculate the new OFF time for pulse one.
       XGDY             Save the OFF time in the Y register.
       LDD    PERIOD    Calculate the OFF time for pulse two.
       SUBD   2,X
       WAI
       SEI              Disable interrupts while updating the ON and
       STY    OFCNT1    OFF registers.
       STD    OFCNT2
       LDD    .X
       STD    ONCNT1    Save the ON time for pulse one.
       LDD    2.X
       STD    ONCNT2    Save the ON time for pulse two.
       LDAA   #MSKOC1   Set the OC1 bit of the timer interrupt mask register
       STAA   TMSK1     to select an interrupt request upon a successful
*                       OC1 compare.
       LDAA   TFLG1     Clear any pending interrupt requests.
       STAA   TFLG1
       LDAA   #HPOC1    Raise OC1 to the highest priority interrupt request.
       STAA   HPRIO
       CLI              Enable interrupts
       BSET   FLG ONGR  Set the FLG register to show the ON time is
       PULA           greater than the off time.
       PULB           Restore the register state.
       PULX
       PULY
       RTS
*
*      This next program section is entered only during start up.
*
STRT  LDD  PERIOD
       TSX
       SUBD ,X         Calculate and store the ON and OFF times.
       STD  OFCNT1
       LDD  ,X
       STD  ONCNT1
       INX
       INX
       LDD  PERIOD
       SUBD ,X
       STD  OFCNT2
       LDD  ,X
       STD  ONCNT2
*
*      Determine whether the on or off time is greater.
*      If the on time is larger  select OC1 to generate interrupt
*      requests and initialize the compare registers in the appropriate
*      order.
*      Otherwise select OC2 successful comparisons to generate
*      interrupt requests and choose the appropriate order to
*      initialize the compare registers.
*
       ADDD  ONCNT1  Add the two ON times together.
       SUBD  PERIOD
       BCS   SELOC2  Branch if the OFF time is greater than the ON time.
       LDAA #MSKOC1   Set the OC1 bit of the timer interrupt mask register
       STAA TMSK1     to select an interrupt request upon a successful
*                     OC1 compare.
       BSET FLG ONGR  Set the FLG register to show the ON time is
*                     greater than the OFF time.
       LDD  ONCNT1
       STD  REGOC2    Store the on time in the output compare 2 register.
       ADDD OFCNT1
       STD  REGOC3    Store the result in the output compare 3 register.
       ADDD ONCNT2
       STD  REGOC4
       ADDD OFCNT2
       STD  REGOC1    Store the double period in the output compare 1 reg.
*
       LDAA #HPOC1    Raise OC1 to the highest priority interrupt.
       STAA HPRIO
       BRA  ENABLE
SELOC2 LDAA #MSKOC2   Set the OC2 bit of the timer interrupt mask register
       STAA TMSK1     to select an interrupt request upon a successful
*                     OC2 compare.
       BCLR  FLG ONGR Set the FLG register to show the OFF time is
*                     greater than the ON time.
       LDD  OFCNT1   Start up the timer sequence.
       STD  REGOC3    Store the number in the compare register 3.
       ADDD ONCNT2
       STD  REGOC4    Store the result in the compare register 4.
       ADDD OFCNT2
       STD  REGOC1    Store the number used by the compare reg. 1.
       ADDD ONCNT1
       STD  REGOC2    Store the number used by the compare reg. 2.
       LDAA #HPOC2    Raise OC2 to receive the highest priority.
       STAA HPRIO
*
ENABLE CLI            Enable the interrupt system.
       BSET FLG STRTUP  Set the FLG register to show startup has occurred.
       PULA
       PULB           Restore the register state.
       PULX
       PULY
       RTS
```

A.7 Real Time Timer

```
*
*************************************************************************
*
*                          REAL  TIME  TIMER
*
*************************************************************************
*
```

```
*    The real time timer will monitor the time in days, hours, minutes
*  seconds, and fractions of a second. Tasks can then be scheduled
*  according to the time of day or delayed for some specified time period
*  as required.
*    The timer will be driven by the "overflow" interrupt request which
*  is assigned a lower priority than the "output compare" interrupt
*  request. An overflow occurs once every 2 to the power of 16 clock
*  cycles, the clock rate being two megahertz. Each time the processor is
*  interrupted the 16 bit "fraction of a second" register is updated. This
*  register is then compared to a number representing one second. If the
*  fraction of a second register is large enough The number representing
*  one second is subtracted from it and the "second" register incremented.
*  If 60 seconds have elapsed the "minute" register is incremented and so
on.
*   In binary 2,000,000 is 0001 1110 1000 0100 1000 000. This is the number
*  which should be subtracted from the fraction of a second register once
*  it is large enough. It is however unnecessary to keep a record of the
*  last seven zeros. Shifted seven places the number becomes 0011 1101
*  0000 1001 and once every interrupt period 0000 0010 0000 0000 will be
*  added to this 16 bit fraction of a second register.
*
************************************************************************
*
*       Declare some constants
*
MAXSM   EQU  60      The maximum number of seconds or minutes.
MAXHR   EQU  24      The maximum number of hours.
CNTSEC  EQU  $3D09   The number representing one second.
INCFRC  EQU  $0200   The number added to the fraction of a second
*                    register each interrupt period.
*
```

A.7.1 Time of Day Interrupt Service

```
*
**********************************************************************
*
*                TIME  OF  DAY  INTERRUPT  SERVICE
*
**********************************************************************
*
TMINTR  LDAA  TFLG2     Clear the interrupt request.
```

```
        STAA  TFLG2
        CLI                 Enable the interrupt system.
        LDD   FRCSEC        Increment the count by one time period.
        ADDD  #INCFRC
        CPD   #CNTSEC       Subtract the count representing one second.
        BCC   INCSEC
        STD   FRCSEC        Save the fraction of a second count.
        RTI
INCSEC  SUBD  #CNTSEC       Add back the count representing one second.
        STD   FRCSEC        Save the fraction of a second count.
        BSR   INCTME
        RTI
```

A.7.2 Increment The Time of Day

```
*
**************************************************************************
*
*                      SUBROUTINE  INCTME
*
**************************************************************************
*
*    Subroutine INCTME increments the time of day counter
*    by one second.
*
*  The time of day is divided into days, hours, minutes and seconds.
*
*
INCTME  INC   SECOND   Increment the second counter.
        LDAA  SECOND
        SUBA  #MAXSM   Check if the counter has reached 60.
        BEQ   INCMIN
        RTS
INCMIN  STAA  SECOND   Reset the second counter to zero.
        INC   MINUTE   Increment the minute counter.
        LDAA  MINUTE
        SUBA  #MAXSM   Check if the counter has reached 60.
        BEQ   INCHR
        RTS
INCHR   STAA  MINUTE   Reset the minute counter to zero.
        INC   HOUR     Increment the hour counter.
        LDAA  HOUR
        SUBA  #MAXHR   Check if the count has reached 24.
        BEQ   INCDAY
        RTS
INCDAY  STAA  HOUR     Reset the hour counter to zero.
        INC   DAY
        RTS
```

A.7.3 Add Time

```
*
************************************************************************
*
*                         SUBROUTINE  ADDTME
************************************************************************
*
*
*   This subroutine adds a time increment to the current time
*   as specified in the time of day register.
*
*   INPUT:  A pointer to the node containing the time increment.
*
*   OUTPUT: The time increment added to the time of day in the
*           time node pointed to by the X register.
*
*   REGISTERS EFFECTED:  D,CC
***********************************************************************
*
ADDTME  CLRA            Disable the time of day interrupt.
        STAA  TMSK2
        LDD   ,X        Fetch the fraction of a second increment.
        ADDD  FRCSEC    Add the fraction of a second portion of
*                       the time of day.
        SUBD  #CNTSEC   Determine if there is an overflow and the
*                       second counter needs to be updated.
        BCC   LP1       If yes, branch.
        ADDD  #CNTSEC   Restore the fraction of a second count.
        BRA   LP2
LP1     INC   2,X       Increment the second counter.
LP2     STD   ,X        Save the fraction of a second portion of
*                       the result.
        LDAB  2,X       Load the seconds portion of the time increment.
        ADDB  SECOND    Add the seconds portion of the time of day.
        SUBB   #MAXSM     Determine if there is an overflow and the
*                         minute counter needs to be updated.
        BCC    LP3        If yes, branch.
        ADDB   #MAXSM     Restore the second counter.
        BRA    LP4
LP3     INC    3,X        Increment the minute counter.
LP4     STAB   2,X        Save the second portion of the result.
        LDAB    3,X        Load the minute portion of the time increment.
        ADDB   MINUTE     Add the minute portion of the time of day.
        SUBB   #MAXSM     Determine if there is an overflow in the
*                         minute counter.
        BCC    LP5        If yes, branch.
        ADDB   #MAXSM     Restore the minute counter.
        BRA    LP6
LP5     INC    4,X        Increment the hour counter.
LP6     STAB   3,X        Save the minute portion of the result.
        LDAB   4,X        Load the hour portion of the time increment.
        ADDB   HOUR       Add the hour portion of the time of day.
        SUBB   #MAXHR     Determine if there is an overflow in the
*                         hour counter.
        BCC    LP7        If yes, branch.
        ADDB   #MAXHR     Restore the hour counter.
        BRA    LP8
LP7     INC    5,X        Increment the day counter.
LP8     STAB   4,X        Save the hour portion of the result.
        LDAB   5,X        Load the day portion of the time increment.
        ADDB   DAY        Add the day portion of the time of day.
        STAB   5,X        Save the day portion of the result.
*                         An overflow in the day counter is not accounted
*                         for as this counter cycles back to zero when full.
        LDAB   #$80
        STAB   TMSK2      Enable the time of day interrupt.
        RTS
```

A.7.4 Compare Time

```
*
**********************************************************************
*
*                         SUBROUTINE  CMPTME
**********************************************************************
*
```

```
*
*   This subroutine compares the pending time contained in a timer node
*   to the time of day.
*
*   ASSUMPTION: This routine is called often enough so that
*               that the day and hour registers of the pending time
*               will equal the day and hour time of day registers
*               for a successful match.
*
*   INPUT:  A pointer to the node containing the pending time.
*
*   OUTPUT: Register A is returned with a value of:
*
*               0: The time of day is greater than or equal to
*                  the pending time.
*               1: The pending time is greater than the time of day.
*
*   REGISTERS EFFECTED:  D,CC
*
**************************************************************************
*
CMPTME  CLRA                Disable the time of day interrupt.
        STAA  TMSK2
        LDAB  DAY           Load the current day.
        CMPB  5,X           Compare the pending day.
        BNE   RTN1          Return if the day registers don't match.
        LDAB  HOUR          Load the current hour.
        CMPB  4,X           Return if the hour registers don't match.
        BNE   RTN1
        LDAB  MINUTE        Load the minute portion of the time.
        CMPB  3,X           Compare to the pending time.
        BLO   RTN1          Return if the current time less than the pending
time.
        BHI   RTNO
        LDAB  SECOND        Load the second portion of the time.
        CMPB  2,X           Compare to the pending time.
        BLO   RTN1          Return if the current time less than the pending
time.
        BHI   RTNO
        LDD   FRCSEC        Load the fraction of a second portion of the time.
        CPD   ,X            Compare to the pending time.
        BLO   RTN1          Return if the current time less than the pending
```

```
time.
*
RTNO    LDAA  #$80      Enable the time of day interrupts.
        STAA  TMSK2
        CLRA            The time of day is greater than or equal to the
        RTS             pending time so return with A set to zero.
*
RTN1    LDAA  #$80      Enable the time of day interrupts.
        STAA  TMSK2
        LDAA  #1        The time of day is less than the pending
        RTS             time so return with register A set to one.
```

A.8 Clear Timer Node

```
*
*************************************************************************
*
*                       SUBROUTINE  CLRTME
*************************************************************************
*
*
*   This subroutine sets to zero the timer node pointed to by the
*   address contained in the X register.
*
*
*   INPUT:  The address of the timer node in the X register.
*
*   OUTPUT: None.
*
*   REGISTERS EFFECTED:  D,CC
*
**************************************************************************
*
CLRTME  LDD   #0
        STD   ,X        Clear the fraction of a second register.
        STD   2,X       Clear the second and minute registers.
        STD   4,X       Clear the hour and day registers.
        RTS
```

A.9 Maximum Power Tracking

```
*
***************************************************************************
*
*                        SUBROUTINE  MAXPWR
***************************************************************************
*
*
*   This subroutine attempts to maximize the power delivered to the load
*   by adjusting the duty cycle of the pulse width modulated output.
*   The routine starts by calculating the power level at the present pulse
*   width setting. The pulse width is perturbed and the power level
*   recalculated. If the power level decreases twice the direction of the
*   pulse width perturbation is reversed. The peak power level and its'
*   associated voltage level is recorded for further reference.
*      A number of peak power levels and their associated voltage levels
*   are recorded. The maximum power tracking voltage is then calculated
*   by averaging the voltages corresponding to the top few of the peak
*   power levels.
*
*   ASSUMPTIONS:
*         1) The current signal is tied to AN1 and the array voltage
*            signal to line AN2.
*         2) The load current is approximately continuous throughout
*            the ON and OFF converter states, ie. the load is
*            inductive.
*
*   INPUT:   The initial step size by which the pulse width is to
*            be initially perturbed in STEP.
*
*   OUTPUT:
*         1) The duty cycle is adjusted for maximum power output.
*         2) The voltage corresponding to the maximum power level
*            is placed in MAXVLT.
*         3) The offset current is returned in OFFSET and the open
*            circuit voltage in OPEN.
*         4) The value of the maximum power level is returned
*            in MAXPWR and the maximum current reading in MAXCUR.
*
*   REGISTERS EFFECTED:  CC
*
```

```
************************************************************************
*
NMBR    EQU   6
ARYMAX  EQU   12*3+ARRAY
*
MAXPWR  PSHA             Save the register state.
        PSHB
        PSHX
        PSHY
        CLR   COUNT      Clear the counter which records the number of
*                        overcurrent incidents.
        LDD   #0
        STD   POWER      Clear the variables used.
        STD   TEMP
        STD   TMPVT1
        STAB  MAXVLT
        LDY   #PORTB
        LDX   #ARRAY     Load the base Array address.
        BSET  FLG FRST   Set the bit to indicate the first pass.
        BSET  FLG UP     Set the bit to increase the pulse width.
*                        Clear the bit signalling a decreasing power level.
        BCLR  FLG DECRS
*
*       Enter the loop which perturbs the pulse width and records peak
*       power levels along with their operating voltage levels.
*
MAXLP1  PSHX             Save the array address.
*                        Call for a delay.
        LDX   #TMENOD    Load the address of a timer node.
        JSR   CLRTME
        LDAB  #DLAY
        STAB  2,X
        JSR   ADDTME
MLP1    JSR   CMPTME
        TSTA
        BNE   MLP1
        LDAA  #RDVLT     Read in the array voltage.
        JSR   READ
        STAB  VOLT
        JSR   CURNT      Read in the corrected current value.
        STAB  CURR
        LDAA  VOLT       Calculate the power output.
        WAI
        MUL
        PULX              Retrieve the array address.
*
        BRCLR FLG FRST MLP2  If not the first power reading, branch.
        BCLR  FLG FRST  Clear the flag bit indicating a first pass.
        STD   POWER     Save the power reading.
        LDAB  VOLT
        STAB  TMPVT1    Save the operating voltage.
        BRA   MLP4
*
MLP2
        CPD   POWER     Check if the power level is increasing.
        BLS   MLP3      If NO, branch.
*
*       This program section is entered when the
*       power level is increasing.
*
*       If the converter is turned fully ON, branch.
        BRCLR .Y ON CHDIR1
*
        BCLR  FLG DECRS Clear the decreasing power level flag.
        JSR   PRUPD     Update the power levels.
        BRSET .Y OFF MLP4
*                       If the converter is turned fully off an error
        JMP   BEGIN     has been detected, so start again.
*
*       This program section is entered when the power level
*       is decreasing.
*
*       If the power has twice decreased, branch.
MLP3    BRSET FLG DECRS CHDIR2
        BRCLR .Y ON CHDIR3    If converter turned fully ON, branch.
        BRCLR .Y OFF CHDIR3   If the converter is turned fully OFF, branch.
        BSET  FLG DECRS       Set the decrease flag.
        JSR   PRUPD           Update power levels.
*
*       Change the pulse width
*
MLP4    LDAB  STEP
        LDAA  #1             Indicate an increase in the pulse width.
*                            Branch if the pulse width is to be
```

```
        BRSET FLG UP MLP4B    increased.
        CLRA                  Else indicate a negative change.
MLP4B   JSR   CHANGE          Change the duty cycle.
*
*       Test for overcurrent and branch if clear.
*
        BRCLR FLG OVRCUR MLP4D
*
*       This section is entered if an overcurrent fault has been detected.
*
        LDAA  COUNT
        INCA           Increment the overcurrent counter.
        BCLR  FLG OVRCUR  Clear the overcurrent flag.
        CMPA  #7
        BNE   MLP4C
        JMP   OVRDLY   Exit if the number of overcurrent occurrences = 7.
MLP4C   STAA  COUNT
*
MLP4D   JMP   MAXLP1          Enter the power tracking loop again.
*
*       CHDIR is local to the maximum power routine.
*
*       CHDIR reverses the direction of the maximum power search.
*       It also places the last peak power value and its' associated
*       operating voltage in the array pointed to by the X register.
*       The array pointer is updated and the power search is ended
*       after enough data points have been collected.
*
*       CHDIR1 is called when the power level is increasing and
*       the converter is turned fully ON.
*
CHDIR1  PSHA
        PSHB
        STD   ,X       Save the power reading in the array.
        STD   POWER
        LDAB  VOLT     Save the associated voltage reading.
        STAB  2,X
        BRA   CHDIR4
*
*       CHDIR2 is called when the power has decreased twice.
*
CHDIR2  PSHA
```

```
        PSHB
        LDD    TEMP
        STD    ,X         Save the power reading in the array.
        LDAB   TMPVT2
        STAB   2,X        Save the associated voltage reading.
        BRA    CHDIR4
*
*       CHDIR3 is called when the power level has decreased
*       once and the converter is turned fully ON.
*
CHDIR3  PSHA
        PSHB
        LDD    POWER      Save the last power reading in the
        STD    ,X         maxpower array.
        LDAB   TMPVT1
        STAB   2,X        Save the associated voltage reading.
*
*       This program section is common to all three subroutines
*       which change the direction of the maximum power search.
*
CHDIR4  PULB              Retrieve the last power reading.
        PULA
        STD    POWER      Save the last power reading.
        LDAB   VOLT
        STAB   TMPVT1     Save the last voltage reading.
        CLR    TEMP
        CLR    TEMP+1
        CLR    TMPVT2
        INX               Update the array pointer.
        INX
        INX
        CPX    #ARYMAX    IF enough data points have been collected.
        BHS    EXITL1     EXIT the maximum power searching loop.
*
*       Toggle the direction of the maximum power search.
*
        LDAB   FLG
        COMB
        ANDB   #UP
        BEQ    CHDIR5
        BSET   FLG UP     Set the flag to increase the pulse width.
        BRA    CHDIR6
```

```
CHDIR5  BCLR  FLG UP    Clear the flag so that the pulse width will
*                       be decreased.
CHDIR6  BCLR  FLG DECRS Clear the flag which signals one power decrease.
*
*       Reduce the step size.
*
        LDAB  STEP
        LSRB            Divide the step by two.
        CMPB  #2        Compare the step to a minimum value.
        BHS   CHDIR7
        LDAB  #2
CHDIR7  STAB  STEP      Save the new step value.
        BRA   MLP4
*
*
*       This program section is entered when enough data points
*       have been collected and it is time to exit the power
*       search loop.
*
*
EXITL1  CLRA
        CLRB
        PSHB            Set a counter to zero.
        STD   POWER     Clear memory locations.
        STD   TEMP
        STD   TMPVT1
MAXLP2  LDX   #ARRAY    load the array base address.
        PSHX
MAXLP3  LDD   ,X        Load the power level of array item(i).
        CPD   POWER
        BLO   MLP5      If power level is the highest yet found,
        STD   POWER     Save the value of the power level and a pointer
        INS             to its' place in the array.
        INS
        PSHX
MLP5    INX             Update the array pointer.
        INX
        INX
        CPX   #ARYMAX   If the end of the array has not been found,
        BLO   MAXLP3    loop again.
*
*       The maximum power value has been located.
```

```
*
        PULX              Load the pointer to the highest power level.
        LDD    TMPVT1
        BNE    MLP7       If not the first value, branch.
        LDD    POWER
        CPD    PWRMAX     If this is the highest power level yet obtained,
        BLS    MLP6       save it.
        STD    PWRMAX
MLP6    CLRA
        LDAB   2,X        Load the associated voltage level.
        STD    TMPVT1     Save it to add extra weight to the voltage level
*                         associated with the highest power level.
        PULB
        INCB              Increment the counter.
        PSHB
MLP7    CLRA
        LDAB   2,X
        ADDD   TMPVT1
        STD    TMPVT1
        LDD    #0
        STD    POWER      Set the power level of the value just processed
        STD    ,X         to zero.
        PULB
        INCB
        PSHB
        CMPB   #NMBR      Have the desired number of data points been processed?
        BLO    MAXLP2     No, collect another data point.
*
*       Divide the sum of the voltage levels by the number of samples.
*
        PULB
        CLRA              Fetch the number of data samples.
        LDX    TMPVT1     Fetch the accumulated voltage sum.
        XGDX
        WAI
        IDIV
        LSLD              Round off the result.
        SUBD   #NMBR
        BLO    MLP8
        INX               Increment the quotient.
MLP8    XGDX
        TSTA
        BEQ   MLP9
*
*       An error has been detected so restart the program.
*
        JMP   BEGIN
*
MLP9    STAB  MAXVLT    Store the voltage level to track which corresponds
        PULY            to the maximum power point.
        PULX
        PULB
        PULA
        RTS
```

A.9.1 Change Search Direction

```
*
*
***************************************************************************
*       This subroutine which can be considered local to the maximum
*       power routine updates the power levels.
*
*       INPUT:  1) Current power level in D register.
*               2) Operating voltage level in VOLT.
*       OUTPUT: 1) Current power level is placed in POWER.
*               2) Last power level is placed in TEMP
*               3) Current voltage level is placed in TMPVT1
*               4) Last voltage level is placed in TMPVT2
****************************************************************************
*
PRUPD   PSHA            Save the operating power level.
        PSHB
        LDD    POWER    Move the last power level to TEMP.
        STD    TEMP
        LDAB   TMPVT1   Move the last voltage reading to TMPVT2.
        STAB   TMPVT2
        LDAB   VOLT     Move the current voltage reading to TMPVT1.
        STAB   TMPVT1
        PULB
        PULA            Store the current power level in POWER.
        STD    POWER
        RTS
```

A.10 Current Adjustment

```
*
*************************************************************************
*
*                          SUBROUTINE  CURNT
*************************************************************************
*
*
*   This subroutine reads and adjusts the value of the current being
*   read by the A/D converter. An adjustment may be necessary if the current
*   signal has a DC offset.
*
*   ASSUMPTIONS: 1) The current has settled to a steady state value.
*                2) The offset current has been recorded.
*
*   INPUT:       None.
*
*   OUTPUT:      The adjusted current value in the B register.
*
*   REGISTERS EFFECTED:  B,CC.
*
*************************************************************************
*
CURNT   PSHA
        LDAA   #RDCURR       Read in the current from the A/D.
        JSR    READ
        SUBB   OFFSET        Subtract the DC offset.
        BHS    CUR1          Did the result overflow?
        CLRB                     Yes? Set the result to zero.
CUR1    PULA                 Else continue.
        RTS
```

A.11 Voltage Tracking Routine

```
*
*************************************************************************
*
*                          SUBROUTINE  VLTRAK
*************************************************************************
*
```

```
*
*   This subroutine maintains the array voltage at the level specified
*   in the memory location MAXVLT. The array voltage is adjusted by
*   varying the duty cycle of the power MOSFETS.
*
*   ASSUMPTIONS:
*         1) The timers are initialized and running.
*         2) The voltage corresponding to the maximum power point
*            has been determined.
*
*   INPUT:   The reference array voltage in MAXVLT.
*
*   OUTPUT:
*         1) The duty cycle is adjusted to track the reference
*            voltage.
*         2) The operating array current, load current, array
*            voltage and power output levels are updated.
*
*   REGISTERS EFFECTED:  CC
*
************************************************************************
*
VLTRAK  PSHA              Save the state of the A and B registers.
        PSHB
        JSR   CURNT       Read the current.
        STAB  CURR
        LDAA  #RDVLT      Read in the operating voltage.
        JSR   READ
        PSHB
*                         Take two sets of readings and average them.
        LDAA  #RDVLT
        JSR   READ
        PULA
        ABA               Add the first and second readings.
        RORA              Divide the result by two.
        TAB               Place the result in the B register.
        STAB  VOLT
*
*       Calculate the error voltage, delta V.
*
        SUBB  MAXVLT
        BEQ   VLTRTN
        BHI   INCRSE
*
*       If this program section is entered the array is operating
*       at less than its' optimum voltage. Therefore decrease
*       the MOSFETS' ON time.
*
        NEGB             Form the absolute value of the error voltage.
        BSR   DELTA      Calculate the change in the ON time required.
*                        Delta ON will be returned in the B register.
        LDAA  #1         Indicate a decrease in the pulse width.
        JSR   CHANGE     Ramp down the duty cycle.
        BRA   VLTRTN
*
*       If this program section is entered the array voltage is
*       higher than its' optimum value. Therefore increase
*       the MOSFETS' ON time.
*
INCRSE  BSR   DELTA      Calculate the change in the ON time required.
*                        Delta ON will be returned in the B register.
        CLRA             Indicate an increase in the pulse width.
        JSR   CHANGE     Ramp up the duty cycle.
VLTRTN  JSR   LOAD       Determine the load current and voltage.
        PULB
        PULA             Restore the A and B registers.
        RTS
```

A.11.1 Adjustment to the Duty Cycle

```
*
*************************************************************************
*
*                       SUBROUTINE  DELTA
*************************************************************************
*
*
*   This subroutine calculates the change to the ON time of the pulse
*   width modulated waveform according to the formula:
*
*          delta ON = delta V * ON / Varray
*
*    Where ON is the last ON time, delta V is the error voltage and
*    Varray is the operating array voltage.
```

```
*
*    delta ON = delta V * ON / Varray
*
*    ASSUMPTION: The timers are initialized and running.
*
*
*    INPUT:    1) The error voltage in the B register.
*              2) The operating voltage in VOLT.
*              3) The current on time in ON.
*    OUTPUT:
*              1) The change in the ON time in the B register.
*
*    REGISTERS EFFECTED:  B,CC
*
*******************************************************************************
*
DELTA   PSHA                Save the contents of the A and X registers.
        PSHX
        LDAA  ONCNT1+1       Multiply the ON time by the error voltage.
        WAI
        MUL
        XGDX                Store the result in the X register.
        LDAB  VOLT          Load the operating voltage.
        CLRA
        XGDX
        WAI                 Wait for an interrupt to complete as the next
*                           instruction takes 41 cycles.
        IDIV                (delta V * ON) / Varray.
*
*       Check the remainder to see if the result should
*       be incremented.
*
        LSLD                Multiply the remainder by two.
        TSTA                If the reminder spills into the A register
        BNE   DLP1          the result should be incremented.
*
        CMPB  VOLT          If 2 * remainder < VOLT
        BLO   DLP2             do nothing
*                           Else
DLP1    INX                    increment the result.
DLP2    XGDX
        TSTA
        BNE   ERROR     If A is not equal to zero an error has occurred.
        TSTB
        BNE   DLP3      Set the change to one if the result was zero.
        INCB
*
DLP3    CMPB  #1
        BEQ   DLP4
*
        LSLB            Multiply the result by two.
*
DLP4    PULX            Restore the X and A registers.
        PULA
        RTS
ERROR   JMP   BEGIN     Restart the whole process.
```

A.12 Read Open Circuit Voltage and Current

```
*
*************************************************************************
*
*                       SUBROUTINE  ADJOPN
*************************************************************************
*
*
*   This subroutine reads in the open circuit voltage and current
*   offset upon start-up.
*
*   ASSUMPTION: Timers are initialized.
*
*
*   INPUT:       None.
*
*   OUTPUT:  1)  The open circuit voltage in OPEN.
*            2)  The offset current in OFFET.
*            3)  The portion of the current signal created by noise in CORECT
*
*   REGISTERS EFFECTED:  CC.
*
**************************************************************************
*
ADJOPN  PSHA            Save registers used by the routine.
```

```
        PSHB
        PSHX
        PSHY
        LDY    #PORTB    Turn the converter off.
        BCLR   ,Y ON
        BCLR   FLG 1     Signal the start up of the pulse width
*                        modulation process.
        LDX    #TMENOD   Load the address of a timer node.
        JSR    CLRTME    Clear the timer node.
        LDAA   #DLAY     Call for a delay.
        STAA   2,X
        JSR    ADDTME
ADLP1   JSR    CMPTME    Wait
        TSTA
        BNE    ADLP1
        LDAA   #RDVLT    Read in the open circuit voltage.
        JSR    READ
        STAB   OPEN      Save the value of the open circuit voltage.
        LDAA   #RDCURR   Read in the offset current.
        JSR    READ
        STAB   OFFSET    Save the value of the offset current.
*
*       Start the pulse width modulation process
*
        LDD    #1
        LDX    #1
        JSR    ADJST
        PULY
        PULX             Restore the machine registers.
        PULB
        PULA
        RTS
```

A.13 Calculate Load Voltage and Current

```
*
*
***************************************************************************
*
*                            SUBROUTINE  LOAD
***************************************************************************
```

```
*
*
*   This subroutine calculates the average load voltage and current
*   knowing the average supply voltage and current and the duty cycle.
*
*   ASSUMPTIONS: 1) The count representing the period is less than 256.
*
*   INPUT:       1) The supply voltage in VOLT.
*                2) The supply current in CURR
*                3) A number representing the on time in ONCNT.
*                4) A number representing the period in PERIOD.
*
*   OUTPUT:      1) The load voltage in MOTVLT.
*                2) The load current in MOTCUR.
*
*   REGISTER EFFECTED:  CC.
*
*************************************************************************
*
LOAD    PSHA
        PSHB
        PSHX
        LDAA  PORTB
        COMA
        ANDA  #OFF      Check if the converter is turned fully OFF.
        BNE   LDOFF     Branch if the converter is turned off
        LDAA  PORTB     Check if the converter is turned fully ON.
        COMA
        ANDA  #ON
        BNE   LDON      Branch if the converter is turned fully ON.
*
*       Calculate the average load voltage.
*
        LDD   ONCNT1
        ADDD  ONCNT2
        LSRD            Average the two ON counts.
        LDAA  VOLT
        WAI
        MUL
        LDX   PERIOD
        WAI             Load voltage = supply voltage * ONCNT / PERIOD.
        IDIV
        LSLD            Round off the quotient by inspecting the
        CPD    PERIOD   remainder.
        BLO    LD1      Increment the quotient if 2 * remainder is
        INX             greater than the divisor.
LD1     XGDX
        TSTA            If A .NE. 0 an error has occurred.
        BNE    LDERR
        STAB   MOTVLT   Record the load voltage.
*
*       Calculate the average load current.
*
        LDD    PERIOD
        LDAA   CURR     Fetch the current reading.
        WAI
        MUL
        LDX    ONCNT1
        WAI             Load current = supply current * PERIOD / ONCNT.
        IDIV
        LSLD            Round off the quotient by inspecting the
        CPD    ONCNT1   remainder.
        BLO    LD2      Increment the quotient if 2 * remainder is
        INX             greater than the divisor.
LD2     XGDX
        STD    MOTCUR   Record the load current as a 16 bit data item.
        PULX
        PULB
        PULA
        RTS
*
*       This program section is executed if the converter is turned
*       fully ON.
*
LDON    LDAB   CURR
        CLRA
        STD    MOTCUR   Load current = supply current.
        LDAB   VOLT
        STAB   MOTVLT   Load voltage = supply voltage.
        PULX
        PULB
        PULA
        RTS
*
```

```
*       This program section is executed if the converter is turned
*       fully OFF.
*
LDOFF   CLR   MOTVLT    Load voltage = 0.
        CLR   MOTCUR
        CLR   MOTCUR+1  Load current = 0.
        PULX
        PULB
        PULA
        RTS
*
LDERR   JMP   BEGIN     An error has been detected so start again.
```

A.14 Change the Duty Cycle in Small Increments

```
*
*
************************************************************************
*
*                       SUBROUTINE  CHANGE
************************************************************************
*
*
*   Subroutine change ramps up or down the duty cycle from its present
*   value to its' desired value in one step increments.
*
*   ASSUMPTIONS: 1) The count representing the period is less than 256.
*
*   INPUT:       1) An integer which indicates the direction of the
*                   change in the A register.
*                   0: Increase the pulse width.
*                   1: Decrease the pulse width.
*                2) An unsigned integer representing the change in the ON
*                   count in the B register.
*                3) The current on time in ONCNT1 and ONCNT2.
*                4) The period in PERIOD.
*
*   OUTPUT:      1) The duty cycle is ramped up or down.
*
*   REGISTERS EFFECTED:  B,CC.
*
```

```
***************************************************************************
*
CHANGE  PSHY
        PSHX
        PSHA
        PSHB
        LDY    #PORTB
        LDX    ONCNT2
*                       Check if the duty cycle is to be increased
        CMPA   #1       or decreased.
        BHI    CHERR    Branch if the parameter is illegal.
        BEQ    NEGCHG   Branch if the duty cycle is to be decreased.
        TSTB
        BEQ    CHRTN    If change = 0 return.
*
*       If an overcurrent condition is encountered, return.
*
CHLP1   BRSET FLG OVRCUR CHRTN
*
*       Increase the duty cycle in one count increments.
*
        PSHB
        LDD    ONCNT1
        CPD    ONCNT2   Branch if ONCNT2 < ONCNT1
        BHI    CHLP2
        INCB            Increment the ON time.
        CPD    PERIOD   If ONCNT = PERIOD turn the converter
        BHS    CHON     fully ON.
        CPX    #0
        BNE    CHNXT    Make sure ONCNT2 does not equal zero.
*                       If ONCNT2 = 0, increment it.
        XGDX
        PULB            Decrement the step counter.
        DECB
        PSHB
        XGDX
        LDX    #ONCNT2  Increment ONCNT2
        INX
CHNXT   JSR    ADJST    Adjust the duty cycle.
        WAI
        BSET   ,Y RUN
        PULB
        DECB              Return if the adjustment of the duty cycle
        BLS    CHRTN      is complete.
        BRA    CHLP1
CHLP2   INX
        CPX    PERIOD
        BEQ    CHON
        JSR    ADJST      Adjust the duty cycle.
        WAI
        BSET   ,Y RUN
        PULB
        DECB              Return if the adjustment of the duty cycle
        BEQ    CHRTN      is complete.
        BRA    CHLP1
*
*       Decrease the duty cycle in one step increments.
*
NEGCHG  PSHB
        LDD    ONCNT1
        CPD    ONCNT2     Branch if ONCNT2 > ONCNT1
        BLO    CHLP3
        DECB              Decrement the ON time.
        BEQ    CHOFF
        JSR    ADJST      Adjust the duty cycle.
        WAI
        BSET   ,Y RUN
        PULB
        DECB              Return if the adjustment of the duty cycle
        BEQ    CHRTN      is complete.
        BRA    NEGCHG
CHLP3   DEX
        BEQ    CHOFF      If the ON count = 0 turn the converter fully OFF.
        JSR    ADJST      Adjust the duty cycle.
        PULB
        DECB              Return if the adjustment of the duty cycle
        BLS    CHRTN      is complete.
        BRA    NEGCHG
CHRTN   PULB
        PULA
        PULX
        PULY
        RTS
*
```

```
*       Start again if an error in the input parameter indicating the sign
*       of the change is detected.
*
CHERR   JMP   BEGIN
*
*       Turn the converter fully ON.
*
CHON    BCLR ,Y ON
        BSET ,Y OFF
        INS             Restore the register state.
        PULB
        PULA
        PULX
        PULY
        RTS
*
*       Turn the converter fully OFF.
*
CHOFF   BCLR ,Y OFF
        BSET ,Y ON
        INS             Restore the register state.
        PULB
        PULA
        PULX
        PULY
        RTS
```

A.15 Overcurrent

```
*
*
*************************************************************************
*
*                            OVRDLY
*************************************************************************
*
*
*    Control will be passed to this routine if the overcurrent
*    interrupt has been repeatedly activated.
*
*    This routine turns off the Mosfets, introduces a half hour
*    delay, then restarts the program.
*
***************************************************************************
*
OVRDLY  LDY   #PORTB
        BCLR ,Y OFF     Turn the power MOSFETS off.
        BSET ,Y ON
        LDD   ONCNT1    Reduce the pulse width modulator to its minimum
        ADDD  ONCNT2    value.
        LDAA  #1
        JSR   CHANGE
OVDLY1  LDX   #TMENOD   Clear the timer node.
        JSR   CLRTME
        LDAB  #30       Add 30 minutes to the time.
        STAB  3,X
        JSR   ADDTME
OVDLY2  JSR   CMPTME    Compare the timer node to the current time.
        TSTA
        BNE   OVDLY2    Loop if time is not up.
        JMP   START     Start again.
```

A.15.1 Overcurrent Interrupt

```
*
*************************************************************************
*
*      Control will be passed to this program section if the overcurrent
*      protection hardware is active.
*      A falling edge of STRA initiates this interrupt routine. Bit zero
*      of PORTC is also set low when the overcurrent protection is active.
*
*      When the current level is reduced, bit zero of PORTC is reset by
*      strobing bit 2 of PORTB which acts as the flip flop clock.
*************************************************************************
*
OVRCRR  LDX  #PIOC
        BCLR ,X $40   Disable additional overcurrent interrupts.
        CLI           Enable output compare interrupt requests.
*
*       Create a timer node on the stack.
*
        TSX
        XGDX          Subtract six bytes from the stack base address.
        SUBD  #6
        XGDX           Place the  new address on the stack.
        TXS
*                      The X register now contains an address with six free
*                      bytes available for a timer node.
OVR1    JSR  CLRTME    Clear the timer node.
        LDAB #DLAY     Add a  second to the time
        STAB 2,X       of day and place the result in the timer node.
        JSR  ADDTME
OVR2    JSR  CMPTME    Compare the timer node to the current time.
        TSTA
        BNE  OVR2      Loop if time is not up.
*
        LDY  #PORTB
        BSET ,Y  4     Strobe the clock of overcurrent flip flop 2
        BCLR ,Y  4    in an attempt to clear the interrupt request.
        BSET ,Y  4
        BCLR ,Y  4
*
*       Branch if the overcurrent condition has been cleared.
*
        LDY  #PORTC
        BRSET ,Y 1 CLEAR
*
*       The overcurrent condition persists so reduce the pulse width.
*
        LDAB #2        Reduce the pulse width.
        LDAA #1
        JSR  CHANGE
        BRA  OVR1      Attempt to clear the fault again.
*
*       The overcurrent condition has been cleared so return to the
*       interrupted routine.
*
CLEAR   LDAB #6        Restore the stack to its original condition.
        ABX
        TXS
*                      Set the overcurrent flag bit.
        BSET FLG OVRCUR
        LDAA PIOC
        LDAA PRTCL     Clear the interrupt flag.
        LDX  #PIOC
        WAI
        SEI            Inhibit interrupt requests.
        BSET ,X $40    Enable additional overcurrent interrupts.
        RTI
```

A.16 Output Compare One Interrupt

```
*
************************************************************************
*
*      Control will be passed to this program section in the event of
*      a successful compare of OC1 if the interrupt is enabled.
************************************************************************
*
INTR1  LDAA   TFLG1     Clear the interrupt request and enable the
       STAA   TFLG1     next interrupt.
       LDD    REGOC1    Load the contents of the output compare
       ADDD   ONCNT1    1 reg. which caused an interrupt.
       STD    REGOC2    Update the output compare 2 register.
       ADDD   OFCNT1
       STD    REGOC3    Update the output compare 3 register.
       ADDD   ONCNT2
       STD    REGOC4    Update the output compare 4 register.
       ADDD   OFCNT2
       STD    REGOC1    Update the output compare 1 register.
       RTI
```

A.17 Output Compare Two Interrupt

```
*
*      Control will be passed to this program section in the event of
*      a successful compare of OC2 if the interrupt is enabled.
*
INTR2  LDAA   TFLG1     Clear the interrupt and enable the
       STAA   TFLG1     next interrupt request.
       LDD    REGOC2    Load the contents of compare register 2
       ADDD   OFCNT1   which just caused an interrupt.
       STD    REGOC3    Update the output compare 3 register.
       ADDD   ONCNT2
```

```
        STD     REGOC4     Update the output compare 4 register.
        ADDD    OFCNT2
        STD     REGOC1     Update the output compare 1 register.
        ADDD    ONCNT1
        STD     REGOC2     Update the output compare 2 register.
        RTI

*       Place the interrupt vectors in memory.
*
        ORG     $FFDE
        FDB     TMINTR
        ORG     $FFE6
        FDB     INTR2      Insert the interrupt vectors in memory.
        FDB     INTR1
        ORG     $FFF2
        FDB     OVRCRR
        END
```

Bibliography

[1] J.A. Roger, *Water and Photovoltaics in Developing Countries*, Solar Cells (Switzerland), vol. 6, no. 3, Aug. 1982, pp.295-308.

[2] P.R.B. Ward, W.G. Dunford and D.L. Pulfrey, *Performance of Small Pumps with Solar Power*, Proc. Canadian Society of Civil Engineering, 7^{th} Canadian Hydrotechnical Conference, Saskatoon, pp.249-258, 1984.

[3] M.A. Green *Solar Cells: Operating Principals, Technology and System Applications*, Prentice-Hall series in solid state physical electronics, 1982.

[4] D. L. Pulfrey, P.R.B. Ward and W.G. Dunford, *A Photovoltaic-powered system for medium head pumping* Solar Energy, vol. 38, no. 4, 1987, pp255-256.

[5] M.B.Aylward and B. McNeils, *Small scaled solar pumping systems - present status and future prospects*, Int. Jour. of Ambient Energy, vol 5, no. 3, July 1984, pp147-158.