To the Graduate Council:

I am submitting herewith a thesis written by Laura Marlino Clark entitled "Power Conversion Circuitry and System Aspects of Utilizing a Superconducting Coil in a Low Earth Orbit Satellite". I have examined the final copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Electrical Engineering.

J M Bailey

J.M. Bailey, Major Professor

We have read this thesis and recommend its acceptance:

J M Hoagg

J S Lawler

Accepted for the Council:

C W Minkel

Associate Vice Chancellor and Dean of The Graduate School

POWER CONVERSION CIRCUITRY AND SYSTEM ASPECTS OF UTILIZING A SUPERCONDUCTING COIL IN A LOW EARTH ORBIT SATELLITE

A Thesis

Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Laura Marlino Clark

August 1992

ACKNOWLEDGMENTS

The author is indebted to Dr. J.M. Bailey for his support and patience in the completion of this work.

Additionally, the author gratefully acknowledges the technical suggestions and contributions given by others. Among these are Robert Young, Jr., for his technical assistance, John J. Jordon, for his programming expertise, Dr. J.S. Lawler, for technical contributions, and Robert Marlino for his knowledge and assistance in CAD programming.

A large token of indebtedness is owed to Dr. M.B. Scudiere for both technical and programming assistance.

The author is deeply appreciative of the support and encouragement given by Mr. and Mrs. P.J. Marlino and Mr. and Mrs. F.H. Clark.

Finally, this work could not have been completed without the unending reassurances and sustaining confidence supplied by Bruce R. Clark.

The author wishes to dedicate this work to Christopher and Bradley Clark, at times the greatest hinderance, and the largest motivation to its completion.

ABSTRACT

The problem of doubling the mission life of Low Earth Orbit Satellites from five to ten years lies primarily with the longevity of the batteries used. These batteries are currently utilized on satellites to supply power to circuitry during the time the satellite's solar array is obscured from the sun. By replacing the batteries with a superconducting coil, to be utilized as a rechargeable energy source, greater energy efficiency and longer mission life can be obtained.

A historical overview of this technology is presented and some of the problems associated with its implementation are discussed.

The issues of charging the coil from a solar array are addressed and analytically examined. A possible design for a chopper circuit used to operate a resistive load from both the coil and array is then developed.

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER I

GENERAL CONSIDERATIONS

1.0 Introduction

With the advent of high temperature superconductivity, the prospect for application of this technology to energy storage is becoming more attractive. While a high temperature superconductor that can readily serve as an engineering material for coil winding is not currently available, it is possible that one will soon be developed. This technology could replace the electrochemical batteries commonly used for spacecraft power.

Beneficial features of such a system would include greater efficiency, significantly extended mission life, higher reliability, and superior responsiveness to load transients with concomitant increased system stability.

The term Superconducting Magnetic Energy Storage (SMES) has traditionally been reserved for such systems used for terrestrial utility applications. With the application of this technology to space the acronym CHESS has been adopted to distinguish between the two utilizations. CHESS stands for Compact High temperature superconducting Energy Storage System.

This thesis is primarily concerned with the power conversion circuitry necessary to provide the interface between a superconducting coil or a solar array and a low earth orbit (LEO) spacecraft.

A major concern that will be addressed is that of charging a superconducting coil from a solar array. Both the array, if operating within a prescribed voltage range, and the coil can be modeled as current sources.

Secondarily, concerns and problems from a systems design viewpoint will be presented and dealt with in the course of this thesis.

1.1 Problem Statement

Historically, spacecraft power systems have been low voltage, low power DC systems. In low earth orbit applications, the power system has consisted of solar photovoltaic arrays with conventional batteries to supply power during the satellites' eclipse period. These batteries have defined the boundaries of the mission's life, as they are inherently limited by the number of their charge/discharge cycles and are temperature sensitive. The rate of energy removal from these batteries is also restricted. Additionally, the weight of these batteries is relatively high, and the space they occupy on board the spacecraft is considerable.

A number of unique requirements exist for a spaceborne power system. Perhaps foremost among these is the requirement that a satellite's components be lightweight and of minimal volume. A spaceborne power system must also be able to withstand the high forces associated with launch and withstand space radiation, as well as expel heat loss to the surrounding environment. Moreover, it must not induce interference with sensitive satellite circuitry and must be capable of operating in a zero gravity environment. The system must additionally be capable of prolonged autonomous operation.

As power requirements and the need for longer missions have increased, more attention has been devoted to alternatives to conventional space power technology. A superconducting magnetic energy storage system is one option that is currently being considered and will become even more appealing as high temperature superconductors (HTSC) become available.

Table 1.1-1[1] details various options for supplying spacecraft power during the eclipse phase of an orbit and some of their operating characteristics.

Although the choice of method to be utilized must be assessed on an case by case basis, in general certain attributes of a CHESS system make it amenable to a wide variety of applications. Its high reliability and efficiency could very well decrease the need for other components usually required in conventional spacecraft power systems and could also lead to a decrease in the size of their solar cell arrays. These secondary benefits could in turn reduce the drag and orbital decay of the satellite.[2]

A CHESS system is also inherently simple in design. Such simplicity in practice translates into high reliability. The

Table 1.1-1 Energy Storage Technology Alternatives for Photovoltaic Space Power Systems Low Earth Orbit Applications: Less than 1MW Power Systems

Energy Storage Technology	WtHrs/kg 1988	WtHrs/kg 2000+	Round Trip Efficiency	Cycling Flexibility	Rate of Discharge	Estimated Life
Batteries						
NiCad	10-20	NA	75%	Low	Low	5 years
NiH_2	50	80	80%	Medium	Medium	5 years
NaS	NA	100	80%	Medium	Medium	10 years
Lithium	NA	80	80%	?	?	10 years
Fuel Cells						
H_2-O_2	30	180	60%	High	Medium	10 years
Capacitors						
High Voltage Systems	10	20	90%	High	Very High	Long
Flywheels						
Composite Materials (Man Rated)	5	20-30	80%	High	High	Long
Superconducting Magnetic Storage						
LHe SMES	30	85	95%	Very High	Very High	Indefinate
LN_2 SMES	NA	200	95%	Very High	Very High	Indefinate
HTSC SMES	NA	>300	95%	Very High	?	Indefinate

system's ability to accommodate rapid charging and discharging cycles serves moreover to augment the photovoltaic cell's power to the load during peak demands.

With the escalating costs associated with the construction and launching of satellites, as well as the increasing need for more sophisticated missions, the greater operational flexibility of a CHESS system and its potential for greatly extending the life of the satellite are significant factors to be weighed in the assessment of future schemes for satellite power requirements.

CHAPTER II

SMES AND CHESS DEVELOPMENT

2.0 Background

Studies of SMES in the United States first began in the early 1970s. In 1970, H.A. Peterson[3] proposed the use of thyristor Graetz bridges for both the charging and discharging of a superconducting coil. From this innovation, the use of a superconducting coil as a means to store energy was developed. It was first thought that a SMES system would be utilized primarily for electrical utility load leveling functions. In 1971, a study was initiated at the University of Wisconsin to evaluate SMES potential and to perform initial conceptual design work. Shortly thereafter, in 1972, Los Alamos National Laboratories became involved in a SMES study sponsored by the U.S. Atomic Energy Commission. The study evaluated the costs of a SMES system and detailed a program for its development. By this time, the electric utility companies were showing a growing interest in the SMES concept.

By 1976, the Bonneville Power Administration was prepared to construct a small scale SMES in an effort to stabilize voltage fluctuations in the southern California power grid. A 30 MJ SMES was built and put into operation. Its performance was incident free during 1983-1984. An alternative solution to the stabilization problem was discovered, and the SMES was later taken off line.

Through the 1980s, various studies were performed to evaluate the SMES technology for terrestrial applications.

In 1986, the Strategic Defense Initiative Office along with the Electric Power Research Institute began a program to design and build a SMES engineering test model of approximately 10 to 20 MWHr. Two SMES teams were formed and are currently involved in this effort. Bechtel has teamed with GE, General Dynamics, and General Atomics Technologies, while Ebasco has joined in a venture with CBI, Westinghouse and the University of Wisconsin. The program, which is divided into two phases, will examine the economic feasibility and technical viability of SMES for implementation in the mid 1990s.

Phase one of the program, which began in 1987, includes conceptual design, development of fabrication methodology, cost estimation, site selection, and environmental assessments. One of the two teams will be chosen at this phase's completion to continue work. Phase two will then consist of the construction and testing of the SMES unit.

There is a simultaneous effort underway in Japan to study and evaluate the potential usages of SMES.[4] The New Energy Development Organization in Japan concluded a feasibility design study using conventional superconductors in 1985. The Engineering Advancement Association of Japan has performed a similar study.

In May, 1986, a SMES utility group was formed. The Research Association of Superconducting Magnetic Energy

Storage was organized to study and develop commercial SMES plants for electric utility uses.

Currently, there is a study underway at Ft. Monmouth, New Jersey. Its objective is to determine the feasibility of using SMES as a replacement for batteries in terrestrial military applications. This study is expected to last several more years.

At present, no terrestrial SMES plant larger than 30 MJ has yet been built. It is estimated that a full scale SMES will be operational by the year 2000.

To date, the role of superconducting magnetic systems in space has been small.

In 1967, the USSR launched the COSMOS 140 satellite which contained a superconducting magnet. The flight was successful and lasted a total of 32 hours. A second system was then put into space in 1968 by the USSR aboard the COSMOS 213. It was incorporated in a device to analyze cosmic radiation. Approximately seventeen hours into the mission, the device quenched due to a fault in the design of the dewar. Some lessons gleamed from these two flights were that the traumas associated with launch did not appear to adversely affect the coil but that small gravational forces did influence the operation. In the 1970s, a superconducting magnetic system was built by AVCO and was flown on board an Apache rocket. It was constructed using a NiTi solenoid magnet. It withstood 50 g acceleration without quenching. These coils were all initially charged prior to launch.[5] Other superconducting magnets used as magnetic spectrometers in cosmic ray experiments have also flown in space.[6]

Although no orbital system has yet flown under power supplied from a CHESS, valuable information has been gained from these spaceborne superconducting magnet systems. All three systems described survived the rigors of launch. This indicates that high acceleration and vibrations do not pose severe problems for CHESS implementation in spaceflight. Helium was used as a coolant for the superconductor in all flights with complete success, and in all cases the actual magnets operated without failure.

During the spring and summer months of 1987, the Aero Propulsion Laboratory of the Air Force Wright Aeronautical Laboratories at Wright Patterson AFB, Ohio, in cooperation with the Air Force Space Technology Center at Kirtland AFB, New Mexico began to develop a Strategic Plan for Space Power Technology Development.[7] The planning process involved the participation of other government organizations together with major aerospace companies and universities. Among the conclusions reached by the group was that the foremost space power technology option was hardened solar photovoltaic power systems technology coupled with advanced energy storage. This technology was to emphasize:

1. survivability
2. continuous power to 100 KW

3. low weight

4. low earth orbit to geosynchronous earth orbit applications

The NASA Lewis Research Center in cooperation with Argonne National Laboratory also began work in 1987 to evaluate the potential space and aeronautical uses of HTSC's. This study included an evaluation of whether the HTSC might lead to viable magnetic energy storage systems.

Efforts and studies to date have been concentrated on SDI and deep space applications of these technologies. However, with the probable development of the high temperature superconductors, applications to satellites in low earth orbits are now becoming more practical.

2.1 System Description

A simplified diagram of the proposed CHESS system is shown in Figure 2.1. It is composed of the superconducting magnet and its supporting structure, necessary cryogenic equipment to provide thermal isolation, power conversion circuitry (PCC), and the load.

In this application, the superconducting magnet assumes the role that has traditionally belonged to NiCad batteries in the spacecraft.

2.1.1 Coil and Coil Support Structure

The superconducting coil must be lightweight and capable of surviving the high forces encountered during a launch. It

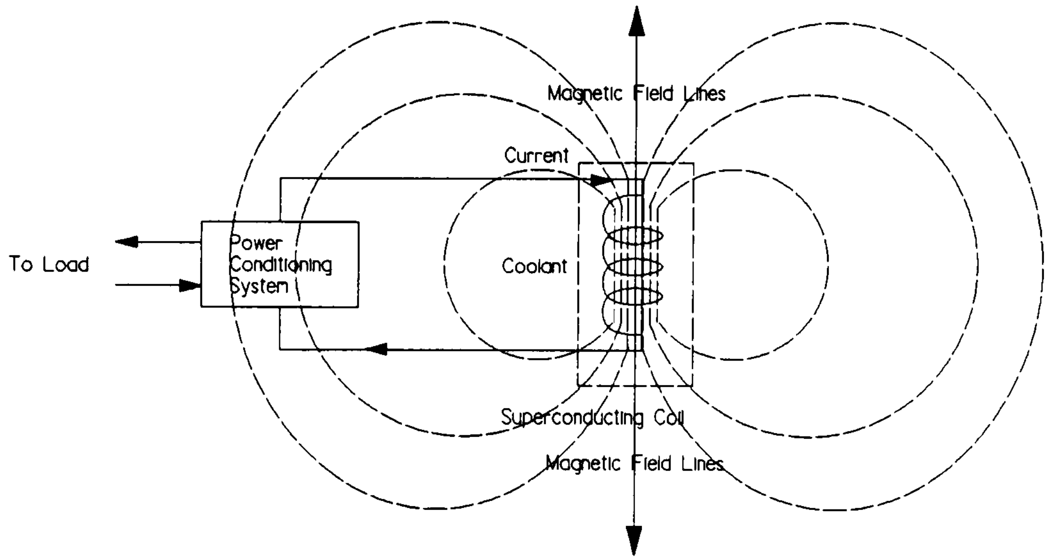


Figure 2.1 Conceptual Diagram of CHESS Unit

must be housed within a low weight cryogenic support structure and be capable of operating in a zero gravity environment. Additionally, it must be capable of carrying extremely high currents with low electrical losses and of operating reliably with very little heat loss.

The superconducting coil is normally housed within a dewar or cryostat which consists of both a vacuum and a coolant vessel. Helium has traditionally been utilized as a coolant; however, with the increase of the critical temperatures associated with superconductors, coolant requirements can likely be altered. The coolant vessel encloses the coil and contains the necessary coolant for the coil. For space applications, the vacuum vessel could, in all likelihood, be eliminated. Dimensions of the coil will depend upon the coil design and the energy storage capacity required. Traditionally, the conductor's mass has accounted for approximately 30% of the weight of the magnet.

The dewar ideally should be constructed in a shape that closely conforms to the shape of the magnet. The inner surface of the dewar should be impervious to any coolant diffusion and of a substance which would prohibit losses to the cryogenic system.

The coil support system must be designed to protect the coil during the high forces associated with the satellites launch. It must position and support the conductor, and provide cooling channels and electrical isolation for the conductor. Various structures have been utilized in the past to accomplish these goals. Weight and volume are primary factors in the design of such support mechanisms. For very large systems the supporting structure can account for as much as 80% of the mass of the structure and coil.[8] In general, this mass must be cooled to the same temperature as the magnet.

Heat generated by the high current through the coil is transferred to the coolant through the power leads. With the current technology this causes some helium boil-off. An important consideration in this regard is convection heat transfer in a zero gravity environment. As there will be no force available to remove heated vapors, an innovative cooling scheme must be devised.

The operation of the coil will consist of three phases, as shown in Figure 2.2.

During Phase 1, the superconducting coil will be charged by the photovoltaic cells to a maximum energy level equal to

$$E=(1/2)L(I)^2 \qquad (2.2\text{-}1)$$

After the coil has been charged, it is forced into its quiescent state and is isolated from the load. The stored current then circulates through the coil and a persistency switch (Phase 2). The path through which the current travels is composed of superconducting material inside the dewar which contains the coil. As the dewar is maintained below the

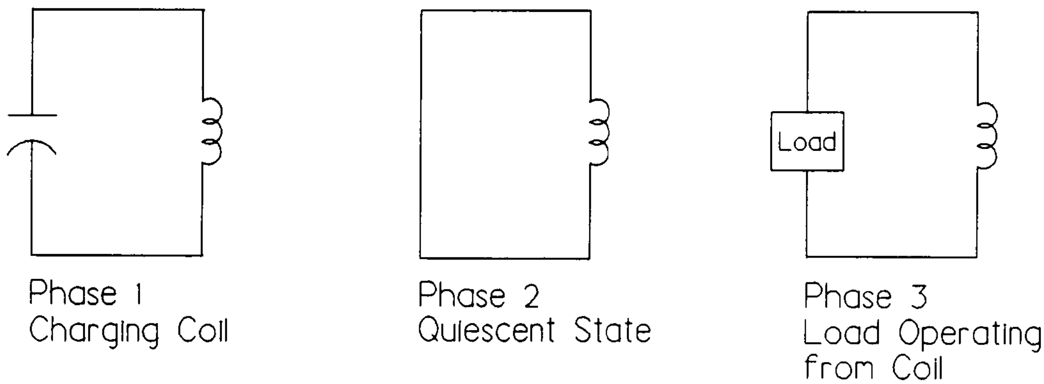


Figure 2.2 Superconducting Coil Operational Modes

superconductor's critical temperature, the superconductor should have close to zero resistance resulting in negligible losses.

Phase 3 begins as the satellite enters its eclipse period and can no longer operate from its solar cells. It is during this time that the superconducting coil provides constant power to the load via the PCC.

To charge and discharge the coil, it is necessary to open the persistency switch. As of this writing, no superconducting switch has yet been developed. Currently switching is being accomplished through the use of sections of superconductor transformed from normal to a superconducting state by altering the its temperature. Such a switching scheme has been tried and tested to 200 A at zero field.[9]

The persistency switch should ideally have low inductance to minimize switching time and the energy stored within the switch. It must also have high resistance in the normal state so that it appears as an open to the coil during charging time.

Research is currently underway at various institutions to develop a superconducting switch. An innovative mechanism being pursued at the University of Illinois and W.J. Schafer Associates, Inc. involves a semiconductor switched thyratron with a fiber optic GaAs activated cathode[10]. It would be capable of high current (10 KA) operation at a maximum switching frequency of 100 Khz at low temperatures (77 K).

Other promising research is being conducted at Oak Ridge National Laboratories[11] towards the development of field controlled, thin film, high speed superconducting switches. A major concern being addressed with this work is with switching losses in the devices. At ORNL schemes utilizing soft switching or zero voltage switching topologies are being utilized to produce inverter circuits which are being designed to operate in the neighborhood of 400 Khz.

Once the persistency switch is opened the current in the superconducting coil passes through a heat gradient from the helium coolant to the ambient outside temperature along a set of low loss power leads (see Figure 2.3).

A new HTSC electrical lead, which has set new records for its current carrying capability, has recently been developed by Westinghouse in association with Argonne National Laboratory.[12] The lead has been tested to 2000 A and has also been able to reduce coolant boil-off by up to 40%. This could translate into reduced refrigeration equipment and associated costs on board the spacecraft.

A primary system design consideration is the coil containment structure. The dewar and coil supporting mechanism must be able to contain the large magnetic forces generated by the coil. During the time the coil is charging, large magnetically induced mechanical forces, known as Lorentz forces, have to be reacted by the coil structure. These

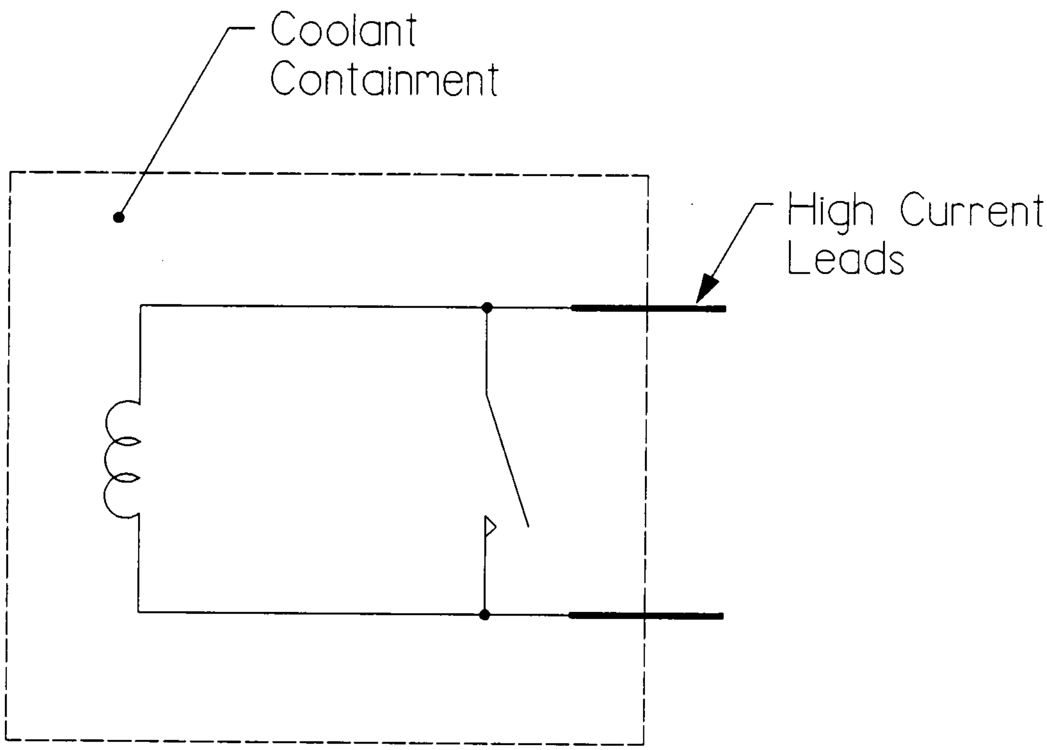


Figure 2.3 Superconducting Coil and High Current Leads

forces expand the coil radially and compress the coil axially. There are also thermal cool down stresses which must be contained.

The minimum structural mass required to store a given amount of magnetic energy is directly proportional to the energy and is represented by the Virial Theorem[13] as

$$M_T - M_C = E\ \rho/\sigma_A \tag{2.2-2}$$

where M_T= mass in tension

M_C=mass in compression

E= stored energy

ρ= density of the structural material

σ_A= allowable stress

If only tensile loads are present, the structural efficiency is maximized and the equation becomes

$$M_T = E\ \rho/\sigma_A \tag{2.2-3}$$

As the need for a lightweight containment structure is critical, new high-strength, non-metallic composite material technology will play an important role in any CHESS design for space vehicles.

The most efficient coil configuration in terms of energy stored per unit weight for space purposes are solenoids. These magnets require less conductive and structural mass than others. Additionally, a "rippled" solenoid presents minimal structural requirements. A toroidal configuration requires roughly twice as much conductor as does a solenoidal or dipolar scheme. The structure necessary for a toroid is also greater. The toroidal magnet, however, has an advantage in that it contains the coil's magnetic field.[14] It is also possible to construct a more compact magnet using this geometry. This could prove to be an important factor in both thermal considerations and in determining critical satellite EMI requirements.

Table 2.4-1[15] illustrates some general effects of varying magnetic field strengths on nearby electronic devices.

The field strength emitted by a power system of the magnitude under discussion needs to be taken into consideration both in the design and physical placement of the coil and the load.

Some work has been done by Mawardi[16] on magnetic configurations using toroidal and poloidal coils in which the fields and currents are force free. A scheme by which the Lorentz forces for the system cancel would reduce structural requirements with accompanying cost and weight decreases.

For a superconductor to remain superconducting it must operate within prescribed limits of current density, applied magnetic field strengths, and temperature. If it exceeds critical levels in any of these limits, the current density decreases sharply. An approximate design rule is that the operating temperature of a superconductor should be no higher

Table 2.4-1 Field Effects on Electronic Devices

FIELD	PROBLEM
0.3 G	Unshielded photo tubes are affected
100-150 G	Ferrite cores start losing their performance
300-600 G	Relays and solenoids start sticking. 5" photomultiplier tubes become hard to shield
3K-5K G	All photo tubes impossible to shield. Some circuits may be affected by EMF's induced by wire motion or vibration.

than 70% of its critical temperature to assure usable fields and current densities.[17]

Serious efforts began in 1961 to develop superconducting magnets. In 1961, Nb_3Sn was discovered to have a critical field of 8.8 T at 4.2 K with a current density of 10^5 A/cm^2. By 1987, materials with critical temperatures of up to 93 K had been discovered. To date transition temperatures of 110 K and 125 K have been reported.[18]

Advances in superconducting technology have yielded both Nb_3Sn and NbTi composites capable of operation at 1.8 K which can carry currents up to 10^9 A/mm^2 in a 10 T field.[19] The use of Nb_3Sn material has been hampered by its brittle characteristics and the difficulty involved in its fabrication. To date the most effective and economical superconductor currently available is a Niobium and Titanium copper composite.

2.1.2 Cryogenics

The refrigeration/liquification equipment necessary to maintain the superconducting coil below its critical temperature must be low in both weight and volume. It must be capable of maintaining constant temperature control despite coolant boil off due to steady state thermal and lead losses.

Helium currently presents unique problems to be overcome in refrigerating superconducting coils. It is a diamagnetic substance and is therefore repelled by magnetic fields. This effect will drive any liquid helium away from the coil and thus necessitates a more sophisticated cryogenic system of perhaps excessive weight and volume.

As high temperature superconductors are put to practical use, coolant requirements will change. Helium will in all probability be replaced by nitrogen as the coolant in CHESS technology.

As critical temperatures rise, refrigeration requirements should decrease correspondingly. As the weight of the cryogenic system depends on the refrigeration system's cycle and capacity, economies in system weight will be realized by reducing the scale of the refrigeration assembly. The cost of the refrigeration unit for a liquid helium cooled superconductor scales according to $P^{0.7}$ where P is the compressor power according to

$$P = \gamma \frac{(T_a - T_o)}{T_o} \qquad (2.3\text{-}1)$$

where γ is a proportionality constant which depends on the surface area of the cold region as well as the insulation quality used, T_o= the operating temperature, and T_a= the ambient temperature.[20]

The cryogenic equipment will account for a large part of the total system weight of a CHESS. The substitution of nitrogen for helium as a coolant would result in a significant benefit to the system's refrigeration weight, volume, and cost requirements.

2.1.3 Solar Cells and Load

The photovoltaic cells to be employed in concert with a CHESS system already exist and have operated successfully on numerous satellite missions. These solar cells will power the load as well as charge the superconducting coil through the power conversion circuitry.

Figure 2.4 illustrates the voltage/current (VI) characteristics of a typical solar cell.

Below the knee of the curve, the array, for all practical purposes, can be simulated as a constant current source. A single cell typically has zero current or has an open circuit voltage of .57 V. By placing a sufficient number of cells in series, various voltage levels can be achieved. By further connecting cells in parallel, the current necessary for a particular application can be realized.

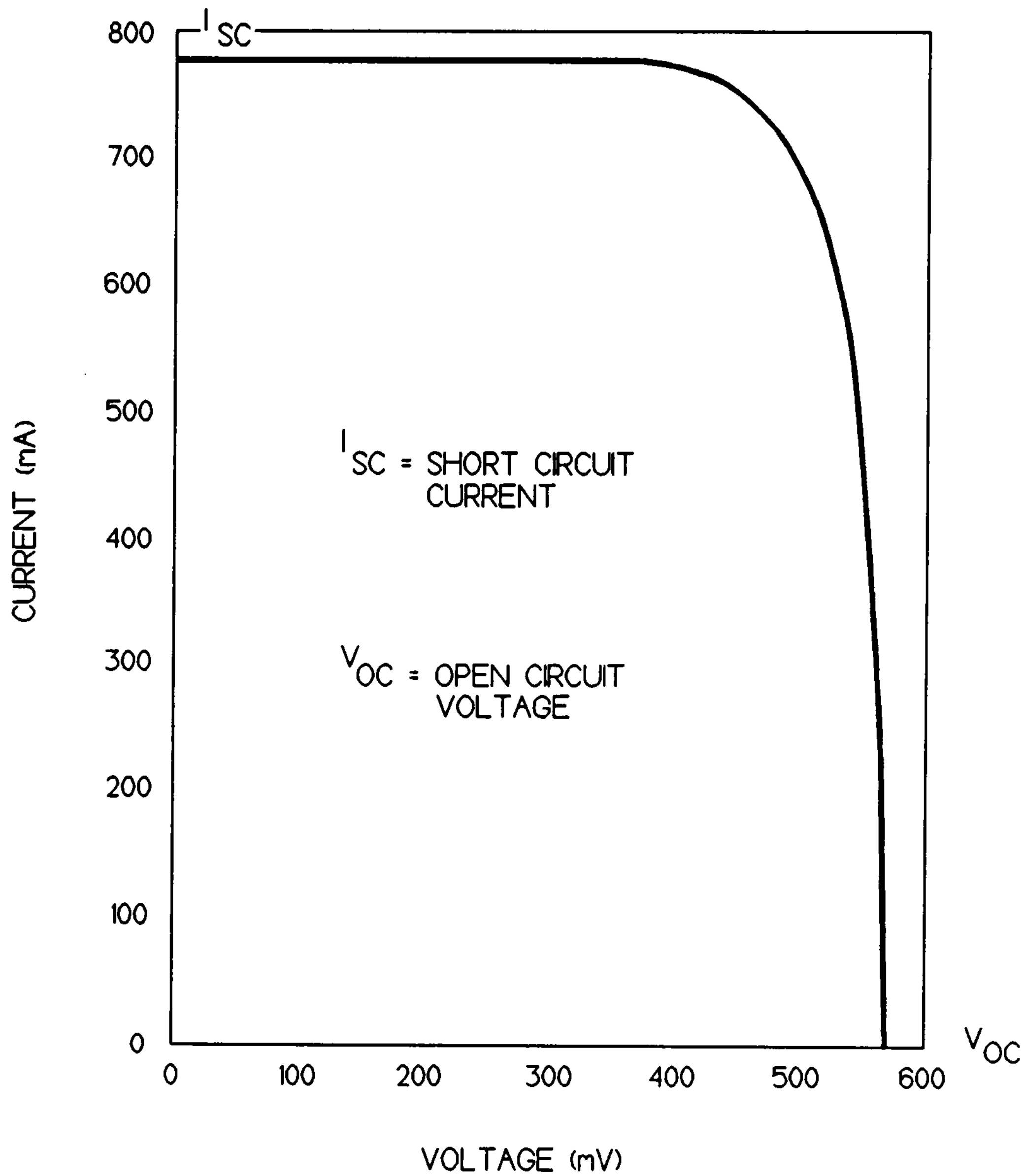


Figure 2.4 Solar Cell VI Characteristics

CHAPTER III

POWER CONDITIONING CIRCUITRY

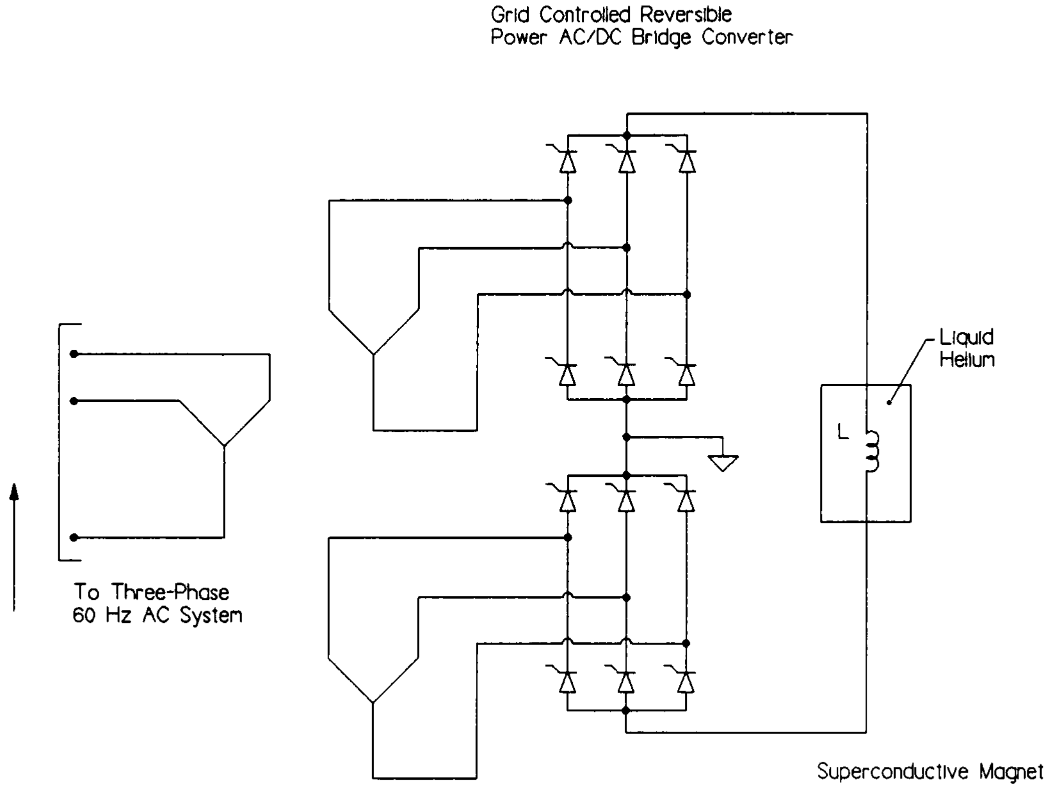
3.0 Goals

Historically, for utility usages, power conversion from the coil to the system's load has been accomplished through three-phase Graetz bridges constructed from GTO thyristors. NASA has proposed taking the current directly from the coil, converting it to AC through the Graetz bridges, stepping it up or down, then converting it back to DC through a second series of Graetz bridges for possible space station power generation[21] (See Figure 3.1).

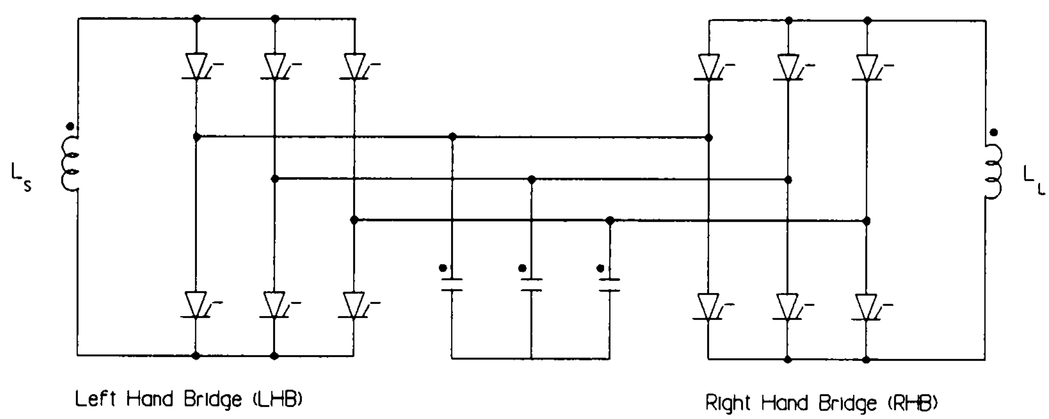
In the application under discussion, the conversion to AC will be eliminated, and the electronics will consist of a switch mode power supply with feedback control provided through pulse width modulation techniques. Parts of the circuit operation will be under microprocessor control for autonomous operation aboard the satellite. A simplified diagram of the power conditioning circuitry is shown in Figure 3.2.

As previously mentioned, the PCC must perform the following functions:

1. Provide constant power to the load from the satellite's solar array.
2. Charge the superconducting coil.



Graetz Bridge for AC/DC Energy Conversion



DC/AC/DC Energy Conversion using Graetz Bridges

Figure 3.1 Graetz Bridge Power Conversion Schemes

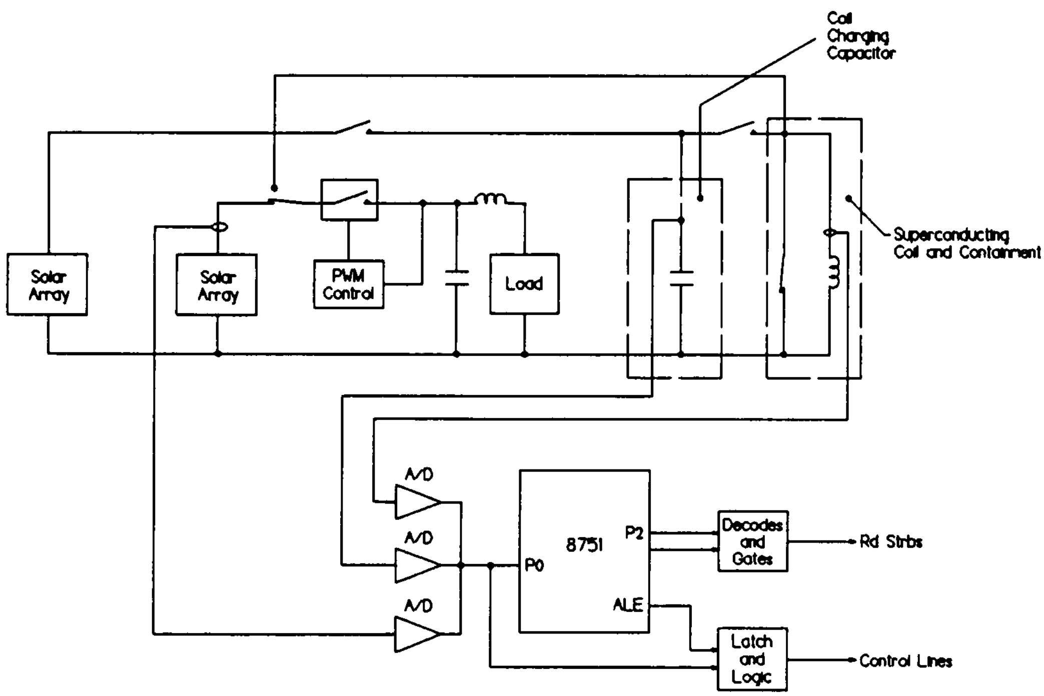


Figure 3.2 Simplified PCC Circuitry

3. Convert the energy from the coil to a form usable by the load.

4. Regulate the flow of energy to the load from the source.

These goals will be accomplished by operating the superconducting coil in the three modes shown in Figure 2.3.

3.1 General Specifications

If the orbit time is approximated at ninety minutes with a nominal eclipse time of thirty minutes, the following general specifications for the PCC can be derived:

1. The length of time the satellite will function from the solar array is nominally sixty minutes.
2. Using an error margin of 10% the time available in which to charge the superconductor is fifty four minutes (approximately 50 minutes or 3000 seconds).
3. Given the same 10% error margin the maximum time the superconducting coil must supply power to the load is thirty three minutes or 1980 seconds.

It has been specified that the load will consume 1000 W of power, and the voltage bus required is 28 Vdc. Given these specifications a purely resistive load will draw 36 A of current. For the purposes of this paper, it is assumed that an excess of 10 A will be available from the solar array.

The switches utilized in the design of the PCC are modeled as superconducting. Any numerical values necessary for design purposes involving the switches are taken from generic MOSFET data sheets. All other elements are considered ideal.

Given this information and assuming a maximum voltage ripple of ± 1 V and current ripple of ± 1.25 A (approximately 7%), the power conversion circuitry can be developed.

3.2 Design

During the portion of the satellite's orbit when the load is operating from the solar cells, the circuit shown in Figure 3.3 will be regulating the voltage across the load at 28 Vdc.

The capacitor C_1, and inductor L_1, will charge to maximum allowable values via the solar array. A constant voltage to the load will then be maintained by controlling the state of Sa through pulse width modulation (PWM).

In DC to DC converters, the use of PWM is accomplished by feeding back the actual voltage or current and comparing it to a reference representing the desired value at the feedback point (see Figure 3.4). The difference between these two voltages is amplified and constitutes an error signal. One common method of pulse width modulation compares this error signal to a constant frequency repetitive waveform. This waveform has a constant peak and establishes the switching frequency of the circuit.

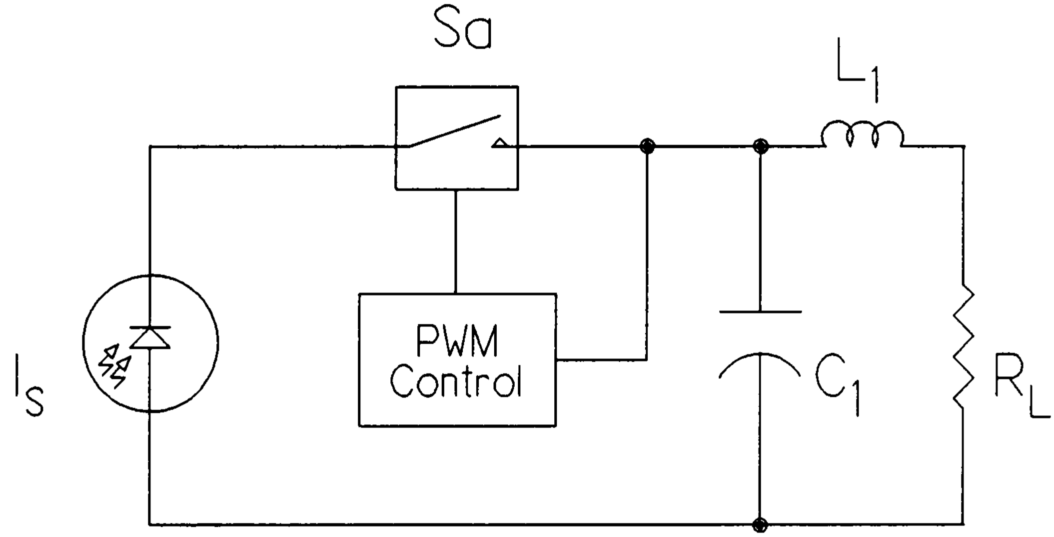


Figure 3.3 Operation from Solar Array

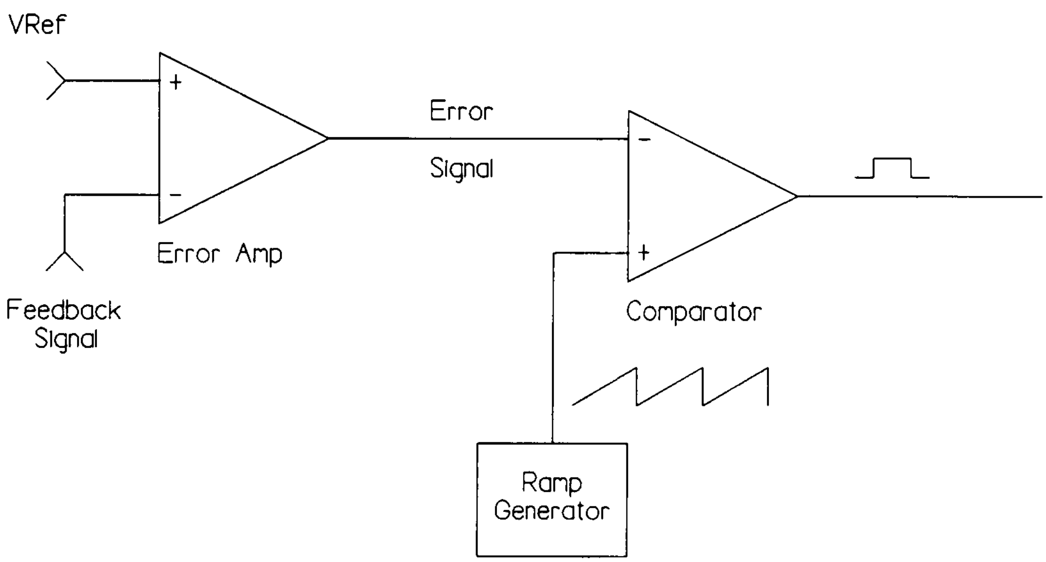
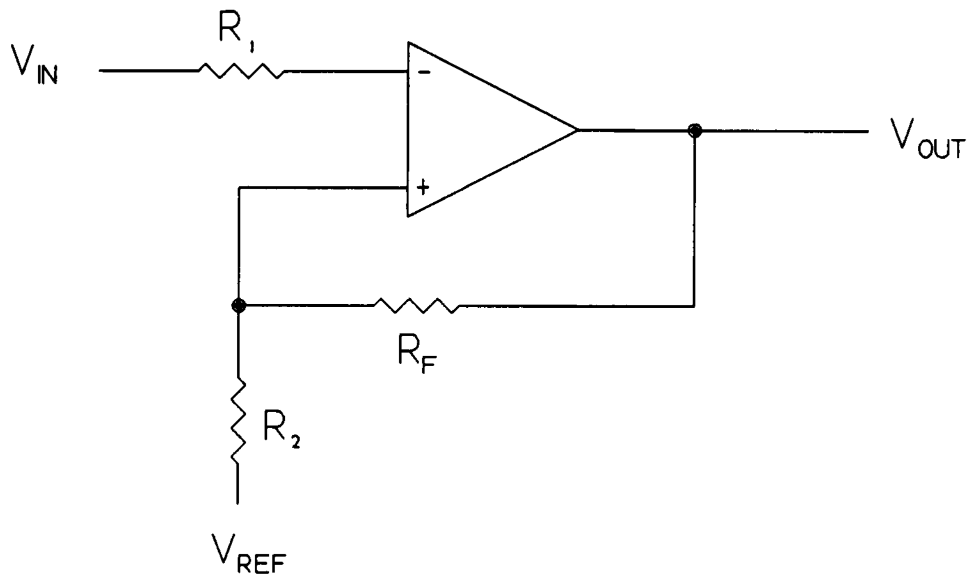


Figure 3.4 Simplified PWM Operation

A switch control pulse is generated from the comparator whose pulse width will vary in direct proportion to the error signal. This control signal is used as a gating pulse to the switch or pass transistor. The error signal is continuously compared to the repetitious waveform. As the magnitude of the error voltage varies, the length of time the pass transistor is biased on will increase or decrease. By modulating the on time of the switch, different charge periods are made available to the capacitor and inductor. The end result will be a controlled average DC output voltage regardless of input voltage or load fluctuations.

In the power conversion circuitry utilized in this design, a simpler type of PWM is employed where the constant frequency waveform shown in Figure 3.4 is eliminated. This modification is due to the large difference in the time involved in charging the energy storage components when the solar cells are active as opposed to when the superconducting coil is supporting the load. A more general hysteresis type controller, shown in Figure 3.5, will be used in place of the more common PWM arrangement. This will permit different switching frequencies for the time when the solar array is active and for the period when the superconducting coil is providing power to the load. The values for the resistors necessary in the circuit are dependent on future drive requirements for the switches. V_o^+ and V_o^- are the output limiting voltages of the operational amplifier.



$$V_U = V_{REF} + \left| V_O^{+} \left(\frac{R_2}{R_2 + R_F} \right) \right|$$

$$V_L = V_{REF} - \left| V_O^{-} \left(\frac{R_2}{R_2 - R_F} \right) \right|$$

Figure 3.5 Hysteresis Comparator

Pulse width modulation is not necessary to maintain a constant voltage across a purely resistive load during the time the satellite is powered from the constant current solar array. It is included as part of the solar array operational circuit topology to illustrate a more generic design in which the PCC would be capable of accommodating a varying or unknown load.

The PWM feedback point for this particular application is taken across C_1 as opposed to monitoring the voltage at the load. This is done to minimize the load ripple. As the voltage across a capacitor cannot change instantaneously when Sa opens the voltage across C_1 will remain constant. Following the opening of Sa the load current will grow due to the reversal of polarity across L_1 which adds to the voltage at C_1. By feeding back the voltage at C_1 a tighter voltage ripple is maintained at the load.

Approximate component values for C_1 and L_1 can be derived from Figure 3.3 and the ripple specifications. Linear voltage and current charge and discharge profiles are assumed and approximate values for the voltages and currents across L_1 and C_1 are utilized.

$$V_{L_1} = L_1 \frac{\Delta I_0}{\Delta t} \qquad (3.2\text{-}1)$$

$$V_{C_1} - V_0 = L_1 \frac{\Delta I_0}{\Delta t} \qquad (3.2\text{-}2)$$

With an output current ripple of 2.5 A equation (3.2-2) becomes

$$V_{C_1}-V_0 = L_1\frac{2.5}{\Delta t} \tag{3.2-3}$$

$$I_{C_1} = C_1\frac{\Delta V_{C_1}}{\Delta t} \tag{3.2-4}$$

The output voltage ripple has been specified as 2 V resulting in

$$I_S-I_L = C_1\frac{2}{\Delta t} \tag{3.2-5}$$

Solving (3.2-5) for Δt and substituting into (3.2-3) with the source current, I_S= 46 A:

$$V_{C_1}-V_0 = \frac{L_1}{C_1}1.25(46-I_L) \tag{3.2-6}$$

If it is assumed the voltage across C_1 will reach approximately 30 V when V_0= 29 V and C_1 is chosen to be 5 mF L_1 is calculated to be 410 μH≈ 400 μH.

3.2.1 Circuit Operation from Solar Array

During the time in the orbital path when the circuit in Figure 3.3 is active, there are two operational topologies corresponding to the periods when the pass switch, Sa, is open and closed. These modes are shown in Figure 3.6.

Mode 1 occurs when the switch Sa is closed. During this period C_1 and L_1 will charge to their maximum permitted values. Upon power up initial conditions are $v_L(0)= v_L'(0)= 0$. Referring to Figure 3.6, assuming no voltage drop across L_1 and summing currents into the nodes designated as v_C and v_L:

$$I_S = C_1\frac{dv_C}{dt} + \frac{1}{L_1}\int(v_C-v_L)\,dt \tag{3.2.1-1}$$

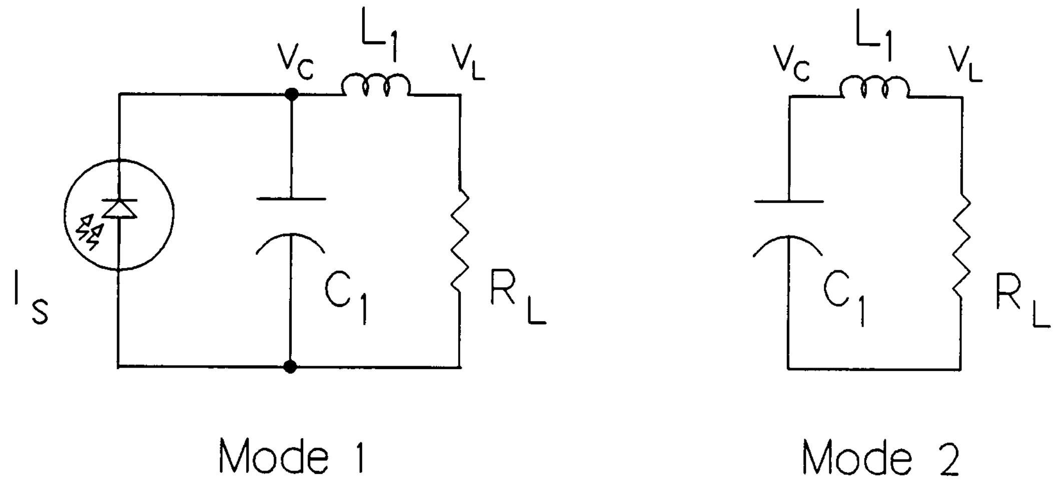


Figure 3.6 Solar Array Operational Topologies

$$\frac{1}{L_1}\int (v_C-v_L)\,dt = \frac{v_L}{R_L} \qquad (3.2.1\text{-}2)$$

Taking the derivative of equation (3.2.1-2), solving it for $v_C(t)$, and plugging it into equation (3.2.1-1) results in the second order differential equation for the voltage at v_L:

$$\frac{d^2v_L}{dt^2} + \frac{R_L}{L_1}\frac{dv_L}{dt} + \frac{1}{C_1L_1}v_L = I_S\frac{R_L}{C_1L_1} \qquad (3.2.1\text{-}3)$$

Solving equation (3.2.1-3) using the methods of constant and undetermined coefficients gives the equation for $v_L(t)$:

$$v_L(t) = A\ e^{-292.9t} + B\ e^{-1707.11t} + 36.8 \qquad (3.2.1\text{-}4)$$

The constants A and B are obtained from the initial conditions resulting in the complete equation for $v_L(t)$:

$$v_L(t) = -44.42\ e^{-292.9t} + 7.62\ e^{-1707.11t} + 36.8 \qquad (3.2.1\text{-}5)$$

The time that the voltage at the load reaches 29 V is found using the Mathcad software package to be t= 5.939 msec.

Upon $v_L(t)$ reaching 29 V the switch Sa will open and the circuit shown in Figure 3.6 Mode 2 will become active. It is during this time that C_1 and L_1 will discharge to provide energy to the load while maintaining the output voltage within its prescribed range.

The solution to the differential equation found for the Mode 1 voltage at the load is the complete response. The solution of any linear differential equation may be expressed as the summation of both the complementary solution (natural response) and the particular solution (forced response). In the Mode 2 circuit, when I_S is removed as a source, the solution to the differential equation for the voltage at the load is the same as that found for the Mode 1 condition minus the forced response:

$$v_L(t) = A\ e^{-292.9t} + B\ e^{-1707.11t} \qquad (3.2.1\text{-}6)$$

The initial condition for $v_L(t)$ is now 29 V. Taking the first derivative of equation (3.2.1-6) and obtaining a value for $v_L'(t)$ at t= 5.939 msec gives the new initial condition for v_L' at the instant the switch is opened. The voltage at v_C is also determined at this point by taking the derivative of equation (3.2.1-2) and solving for v_C. This results in a value for v_C= 30.14 V. Using these new initial conditions the defining equation for the load voltage during the Mode 2 operation is

$$v_L(t) = 36.62\ e^{-292.9t} - 7.62\ e^{-1707.11t} \qquad (3.2.1\text{-}7)$$

The time at which the voltage across the load will decay to 27 V is found through utilization of Mathcad software to be t_1= .805 msec. A value for v_C is found at this instant using the same means as before yielding v_C= 24.41 V. At the moment the voltage at the load reaches 27 V the switch Sa will again close and the voltage at the load will rise back to 29 V. The defining equation is of the same form as equation (3.2.1-5).

The new initial conditions are now $v_L(t) = 27$ and the new value for $v_L'(t)$ is found as before resulting in the equation

$$v_L(t) = -8.167\ e^{-292.9t} - 1.633\ e^{-1707.11t} + 36.8 \qquad (3.2.1\text{-}8)$$

Once more the time for the load voltage to reach 29 V is found from the Mathcad software package to be $t_2 = .485$ msec. Equations (3.2.1-7) and (3.2.1-8) are valid for the steady state operation of the PCC while the load is being supported from the solar array. The switching frequency during this period is

$$f_s = \frac{1}{T_s} = \frac{1}{t_1 + t_2} = 775.4\ \text{Hz} \qquad (3.2.1\text{-}9)$$

The two most significant losses within the switch are the switch's conduction loss and its switching loss. Conduction losses are the result of resistive drops across the component itself and of current flowing through the device junctions. Superconducting switches have been proposed which would be structurally similar to MOSFET devices. MOSFETs are majority carrier devices and have rapid switching characteristics. By designing superconducting switches with topologies similar to MOSFETs, switching losses would be minimized. Furthermore, conduction losses will be ideally zero as the device is composed of superconducting material.

Switching losses are directly proportional to switching frequency, and switching time, as well as the voltage across the switch and current through it. The loss can be approximated from the general expression[22]

$$P_s = V_d I_o (t_{on} + t_{off}) \frac{f_s}{2} \qquad (3.2.1\text{-}10)$$

where

V_d = voltage across the switch

I_o = current through the switch

f_s = switching frequency

t_{on} = switch turn on time

t_{off} = switch turn off time

In this particular application, the switching losses have been minimized to a large extent by maintaining the operating point of the circuit at the knee of the solar cell VI curve. Figure 3.7 graphs power derived from a single solar cell on the same chart as its VI characteristics. By operating near the knee, the largest power efficiency can be obtained from the solar array.

While Sa is closed, the operating point will be at the curve's knee. As Sa begins to transition from a superconducting to a normal state, the current from I_S drops rapidly to zero until the point on Figure 3.7 marked V_{OC} is reached. By operating at the curve's knee the voltage difference, ΔV, across Sa, is therefore minimal.

In order to find a value for ΔV at the operating point desired, it is necessary to extrapolate the data from Figure 3.7. An estimate of 450 mV is made from the graph at the

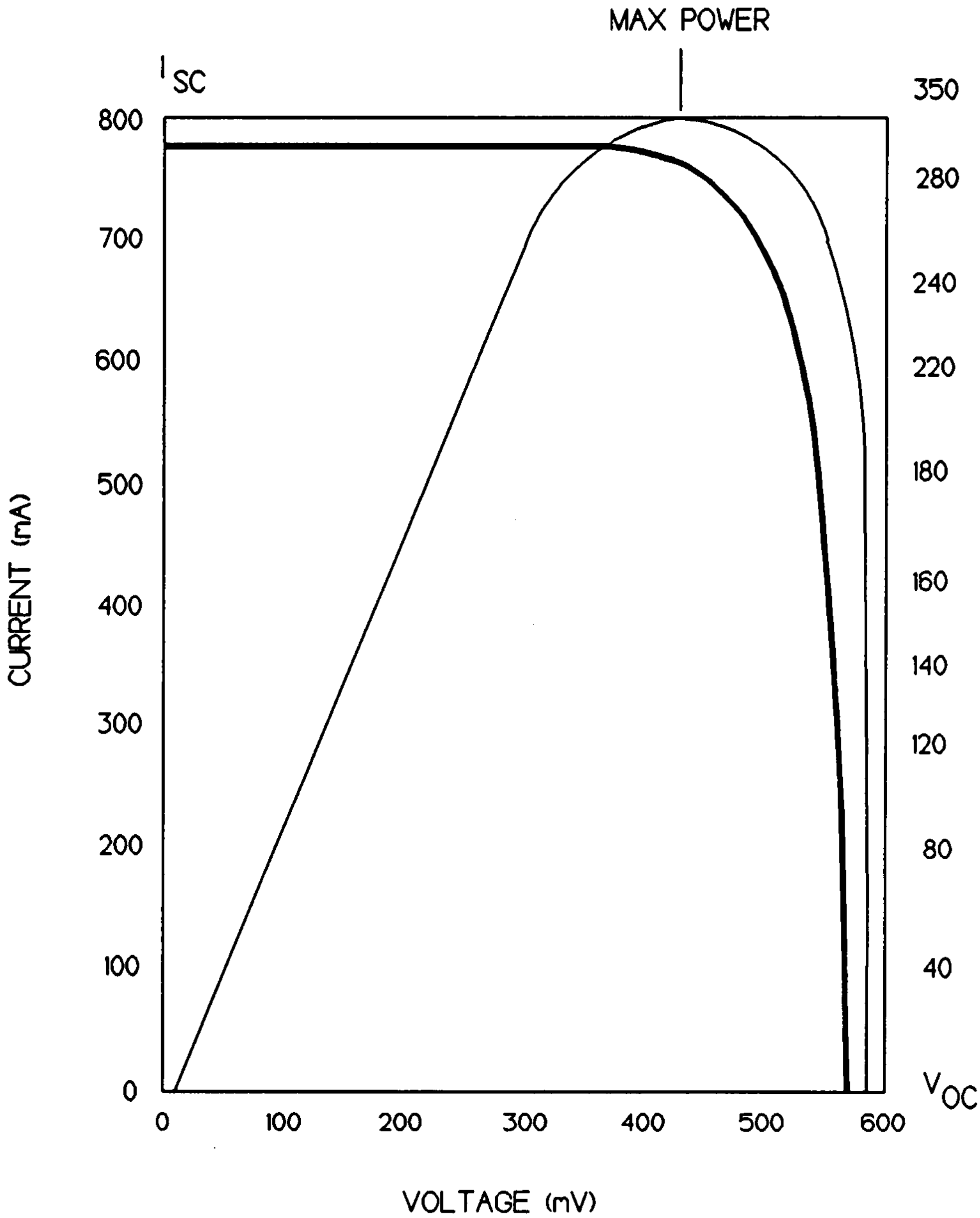


Figure 3.7 Solar Cell Power/Voltage/Current Characteristics

point where the current is still equal to I_{SC} on the edge of the knee. V_{OC} is assigned a value of 570 mv. The maximum voltage at $v_c(t)$ is 30.14 V. Dividing 30.14 by 450 mv reveals that 67 cells in series will result in a solar panel with the correct voltage level. The value for ΔV is then

$$\Delta V = (570*10^{-3})(67) - (450*10^{-3})(67)$$

$$\Delta V \approx 8 \text{ V} \qquad (3.2.1\text{-}11)$$

ΔV represents the minimum difference in voltage between V_{OC} and v_c. While Sa is open, the voltage at the load will decay to a minimum of 27 V and $v_c(t)$= 24.41 V. At this moment, the voltage across Sa will be

$$\Delta V_{TOT} = V_d = \Delta V + (30.14-24.41) = 13.7 \text{ V}. \qquad (3.2.1\text{-}12)$$

The power dissipated in the switch, Sa, due to the switching action is now calculated from equation (3.2.1-10) using typical MOSFET values for t_{on} and t_{off}.

$$13.7(46)(150*10^{-9}+150*10^{-9}) \frac{775.4}{2} = 73.3 \text{ mW} \qquad (3.2.1\text{-}13)$$

A program written to simulate the operation of the PCC in all modes is included in the Appendix. Figures 3.8 and 3.9 show the start up and steady state regulation of the load while power is being supplied from the solar array.

3.2.2 Superconducting Coil Charging Scheme

During the period the solar array is supporting the load, the microcontroller must monitor levels and issue commands to

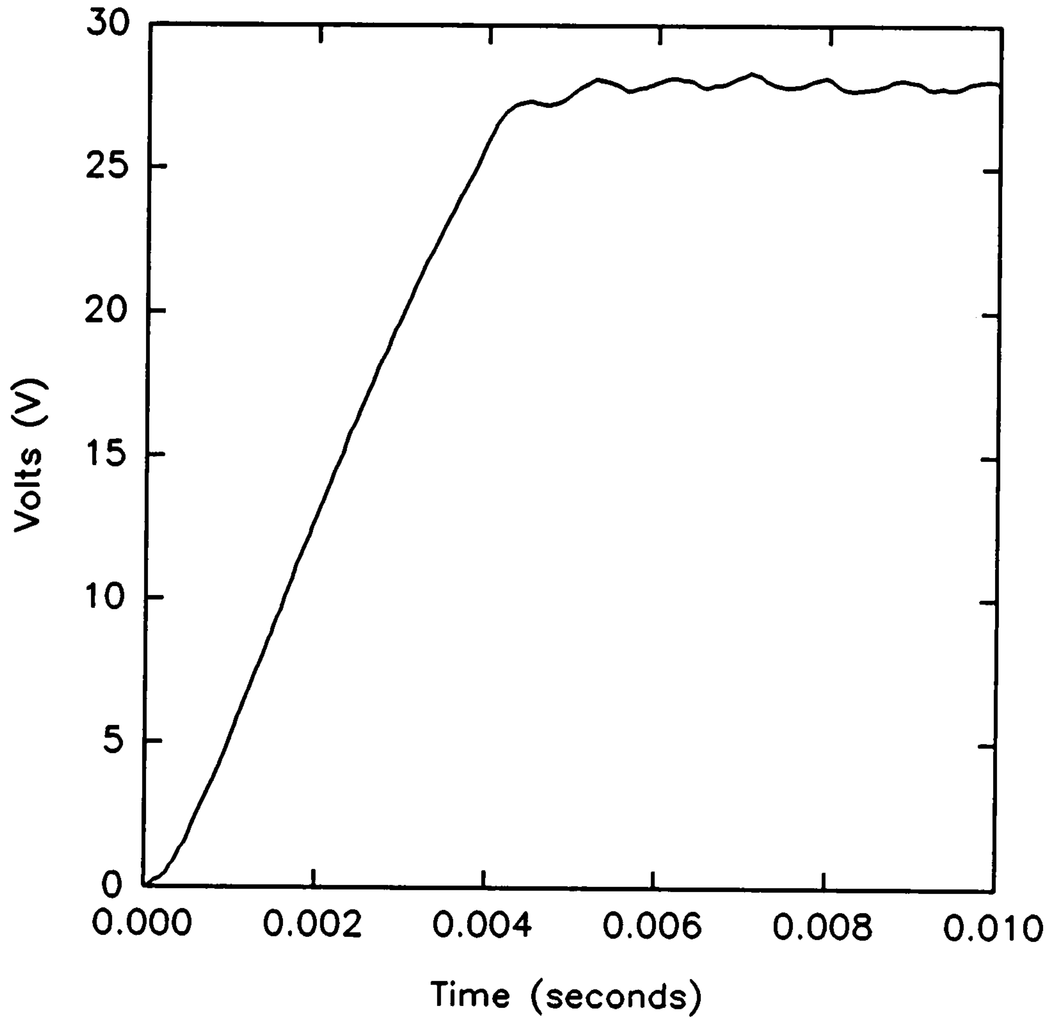


Figure 3.8 Start Up Operation from Solar Array

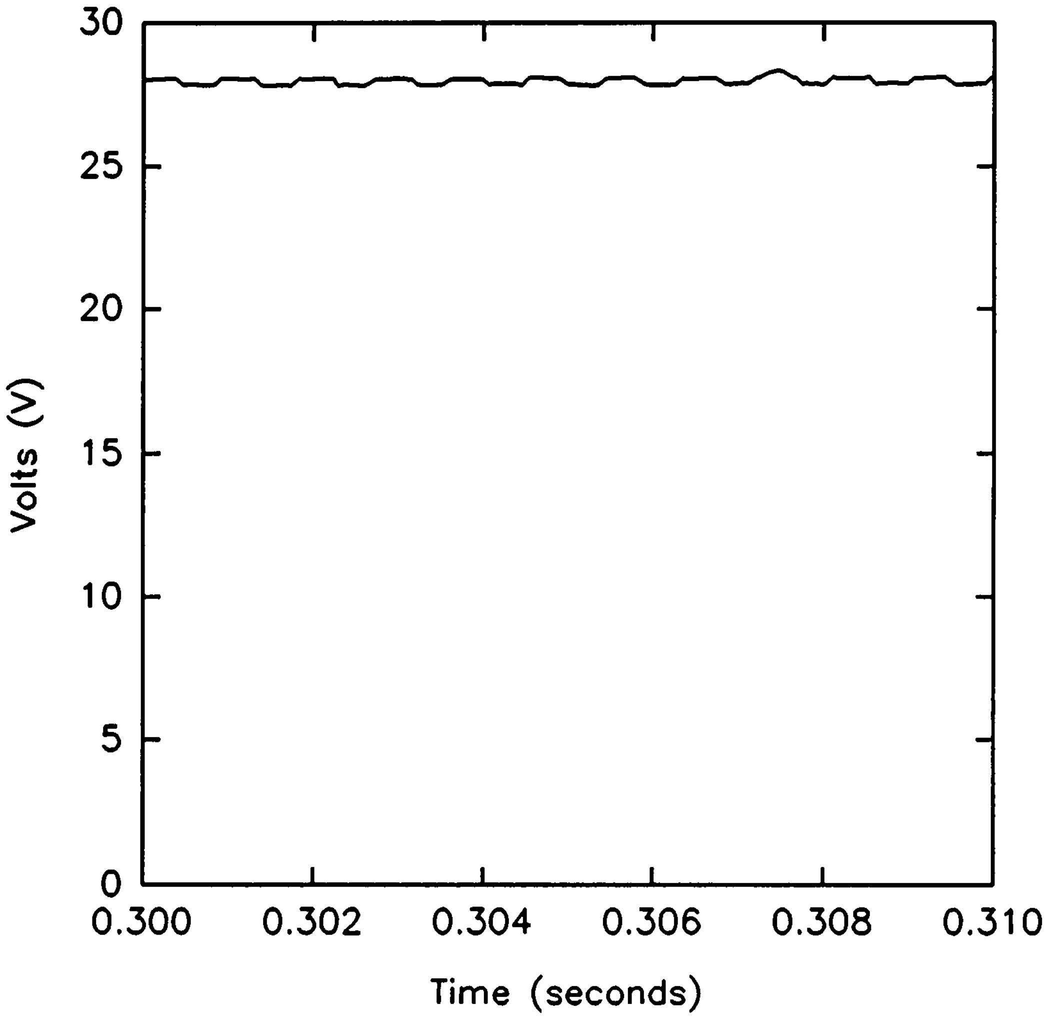


Figure 3.9 Steady State Solar Array Operation

charge the superconducting coil. It is during this phase of the circuits operation that the controller oversees the voltage across C_c. Based on this value, the switches Sb, Sc, and Ps will be set open or closed.

One microcontroller which might be used for this application is the Intel 8751. It is an NMOS chip commonly used in control applications. It is an 8 bit processor with 4 K bytes of on board EPROM as well as 128 bytes of resident data memory. It is capable of accessing up to 64 K bytes of external program and/or data memory.

The chip has 32 I/O lines grouped into four 8 bit ports. It has a serial port as well as two internal 16 bit timer/event counters. Additionally, it has a five source interrupt structure capable of being organized at two priority levels. The 8751 can operate at a maximum frequency of 12 MHZ and can be put into a standby or power savings mode of operation.

Data written to external devices by the 8751 is multiplexed on the P0 bus with the low order address byte. A signal, designated ALE, is provided by the microprocessor to latch this data when it is present on the bus. An 8 bit latch will be used to latch addresses used for decoding purposes.

All current and voltage monitoring operations will be implemented with analog to digital converters supplying data to the microcontroller. It is necessary to know the current through the coil L_s, to detect when the charge on it has reached the desired level. The voltage at C_c must be monitored to control the operation of Sb and Sc to regulate the charging of L_s. Lastly, the current from the solar array must be overseen by the microprocessor in order to determine the time at which to convert to operation from the superconductor.

A circuit diagram showing the address decoding and enabling signals provided by the 8751 for monitoring the points under discussion is shown in Figure 3.10. Switch control logic is also included to insure that switches are not inadvertently activated during the wrong operational mode. The discrete logic can easily be implemented into a small PAL. The MUX shown in the figure is included for future expansion and is necessary only if additional latches are needed. The control signals shown are defined below.

SM=0;	Solar Array Operation
SM=1;	Superconducting Coil Operation
Scon=0;	Sb open, Sc closed
Scon=1;	Sb closed, Sc open
C/NC=0;	L_s charging
C/NC=1	L_s in quiescent state

During the period in which the load is operating from the solar array, the circuit in Figure 3.11, shown without the microprocessor interface, will be utilized to charge the superconductor.

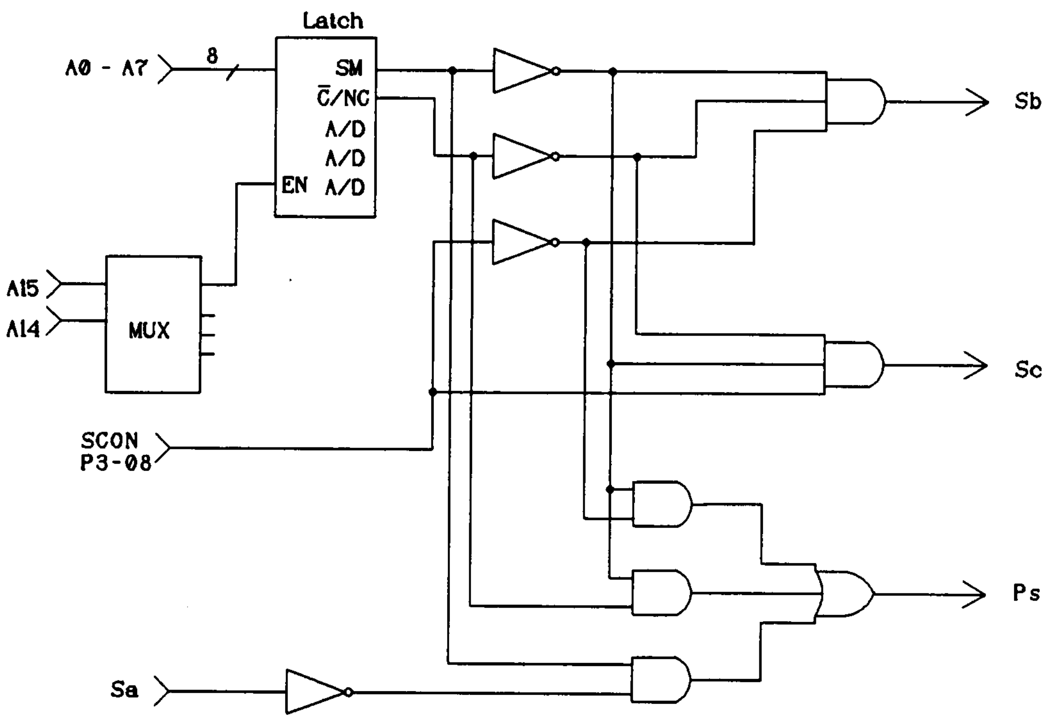


Figure 3.10 Address Decoding and Enabling Signals

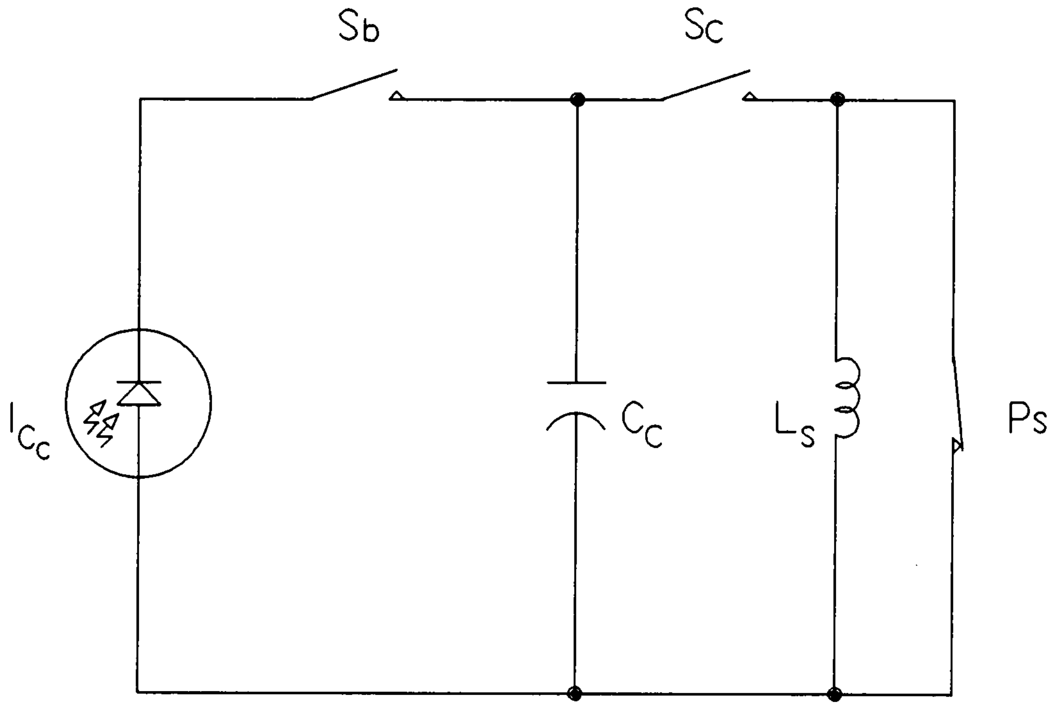


Figure 3.11 Superconductor Charging Circuit

The capacitor C_c will charge during the time switch Sb and Ps are closed and Sc is opened. When C_c has reached its peak voltage, Sb and Ps will open, and Sc will close. The energy in C_c will then be transferred to the superconducting coil, L_s.

The voltage at the capacitor, C_c, will be monitored by the microprocessor. As the voltage decays to a predetermined value Ps and Sb will close and Sc will open. The current through L_s will then circulate through the persistency switch, Ps, while the capacitor, C_c, charges back to its maximum allowable voltage. Once the voltage across the capacitor has reached its targeted value, the switches, Ps and Sb, will open, while Sc closes and the coil, L_s, charges to a higher value. This incremental charging process will be repeated until the current through L_s reaches its final valve as monitored by the controller.

Once this has occurred, Ps will close, Sb and Sc will open, and these conditions will persist until the controller senses the solar cells are not supplying the necessary amount of current to the load. At this time, the final mode of operation begins in which the load operates from the superconductor, L_s.

The coil must be capable of supporting the load for a maximum of 1980 seconds. Referring to Figure 3.6, the amount of energy necessary during this time can be calculated as

$$(1000\ \text{W})(1980\ \text{sec}) = 1.98*10^6\ \text{Wsec} \qquad (3.2.2\text{-}1)$$

The capacitor C_1 will store a maximum amount of energy from the coil equal to

$$E_{C_1} = \frac{1}{2}C_1V^2 = \frac{1}{2}(5*10^{-3})(30.14)^2 = 2.27 \text{ J} \tag{3.2.2-2}$$

The energy stored by L_1 is found in the same manner:

$$E_{L_1} = \frac{1}{2}L_1I^2 = \frac{1}{2}(400*10^{-6})(36.25)^2 = .263 \text{ J} \tag{3.2.2-3}$$

If C_1 and L_1 are considered ideal components the power stored by both these elements during the 1980 second eclipse period will be delivered back to the load with no losses. Equating only the power needed by the load to an amount to be supplied from the superconducting coil results in

$$\frac{1}{2}L_SI_{L_S}{}^2 = 1.98*10^{-6} \text{ Wsec} \tag{3.2.2-4}$$

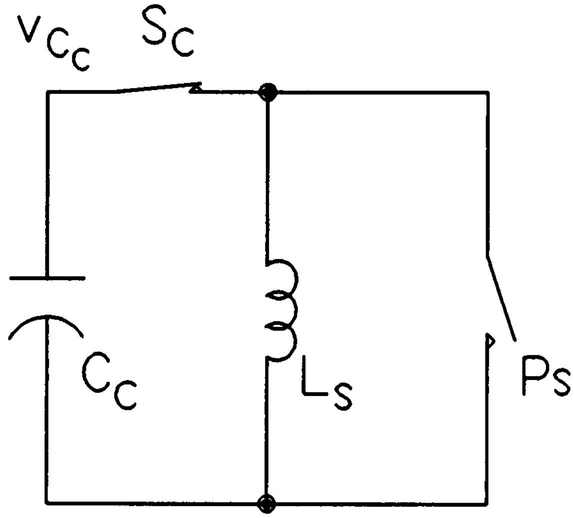
$$L_SI_{L_S}{}^2 = 3.96*10^{6} \text{ Wsec} \tag{3.2.2-5}$$

By equation (3.2.2-5) a 10 H coil will need to be charged to 629.3 A≈ 630 A in order to supply power to the load during the satellite's eclipse time.

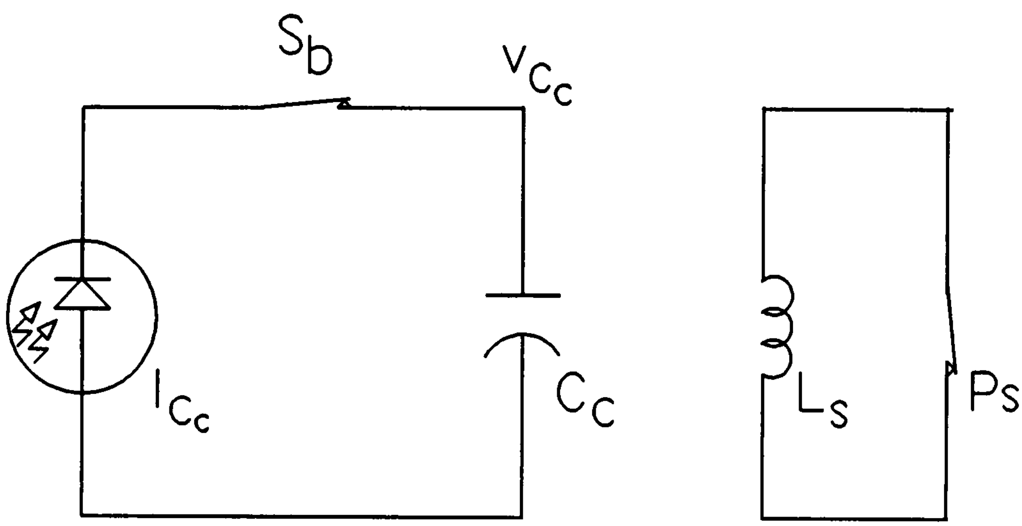
While the coil, L_S, is charging from the capacitor, C_C, the circuit will be as depicted in Figure 3.12, Mode 1. Mode 2 illustrates the conditions present during the time C_C is charging back to its maximum voltage.

If the charge time of the capacitor C_C is approximated as

$$\Delta t_{C_C} = \frac{C_C \Delta V_{C_C}}{I_{C_C}} \tag{3.3.3-6}$$



Mode 1



Mode 2

Figure 3.12 Coil Charging Topologies

and the discharge time of the capacitor C_c into the coil L_s as

$$\Delta t_{L_S} = \frac{C_C \Delta V_{C_C}}{i_{L_S}(t)} \qquad (3.2.2\text{-}7)$$

then the duty cycle of the capacitor charge time is given by

$$\frac{\Delta t_{C_C}}{\Delta t_{C_C} + \Delta t_{L_S}} = \frac{i_{L_S}(t)}{I_{C_C} + i_{L_S}(t)} \qquad (3.2.2\text{-}8)$$

If $\frac{\Delta V_{C_C}}{V_{C_C}}$ is small then the available power can be estimated as $\overline{V_{C_C}} I_{C_C}$. The power utilized in charging the coil, L_s, is then

$$\overline{V_{C_C}} I_{C_C} \frac{i_{L_S}(t)}{I_{C_C} + i_{L_S}(t)} \qquad (3.2.2\text{-}9)$$

By integrating equation (3.2.2-9) and equating it to the energy in L_s a value for the magnitude of the current source, I_{CC}, necessary to charge L_s in the required time period can be obtained:

$$\int_0^T \overline{V_{C_C}} I_{C_C} \frac{i_{L_S}}{I_{C_C} + i_{L_S}} \, dt = \frac{1}{2} L_S i_{L_S}^2 \qquad (3.2.2\text{-}10)$$

using Leibnitz's Rule for differentiation of integrals results in

$$\frac{i_{L_S}(T)}{I_{C_C} + i_{L_S}(T)} = \frac{L_S}{2 \overline{V_{C_C}} I_{C_S}} 2 i_{L_S}(T) \frac{di_{L_S}}{dT} \qquad (3.2.2\text{-}11)$$

which simplifies to

$$\frac{di_{L_S}}{dT} = \frac{\overline{V_{C_C}}}{L_S} \frac{I_{C_S}}{I_{C_S} + i_{L_S}(T)} \qquad (3.2.2\text{-}12)$$

Rearranging and integrating both sides of equation (3.2.2-12) results in

$$\int_0^{I_{L_S}} (I_{C_C} + i_{L_S}(T))\, dI_{L_S} = \frac{\overline{V_{C_C}}}{L_S}\int_0^t I_{C_C} dT \qquad (3.2.2.-13)$$

yielding

$$\frac{1}{2} I_{L_S}^2 + I_{C_C} I_{L_S} = I_{C_C} t \frac{\overline{V_{C_C}}}{L_S} \qquad (3.2.2-14)$$

Rearranging equation (3.2.2-14) and solving for V_{CC}:

$$\overline{V_{C_C}} = \frac{L_S}{t}\left[\frac{I_{L_S}^2}{2I_{C_C}} + I_{L_S}\right] \qquad (3.2.2-15)$$

Equation (3.2.2-15) is graphed in Figure 3.13 with I_{CC} shown as a function of V_{CC} and t= 50 min, I_{LS}= 630 A, and L_S= 10 H. From the graph it can be seen that in order to avoid either an excessively large charging voltage or source current it is necessary to operate near the knee of the curve. An average value for V_{CC} of 9.5 V is selected corresponding to a source current, I_{CC}, of 89 A.

A graph of equation (3.2.2-15) with coil current, I_{LS}, graphed as a function of time and V_{CC}= 9.5 V, I_{CC}= 89 A, and L_S= 10 H is shown in Figure 3.14. It confirms the results obtained from equation (3.2.2-5) that L_S can be charged to 630 A in approximately 3000 seconds.

In order to obtain maximum efficiency in the charging of the superconducting coil the solar array should be configured with VI and power characteristics as shown in Figure 3.15. This profile differs from that needed to support the load (see

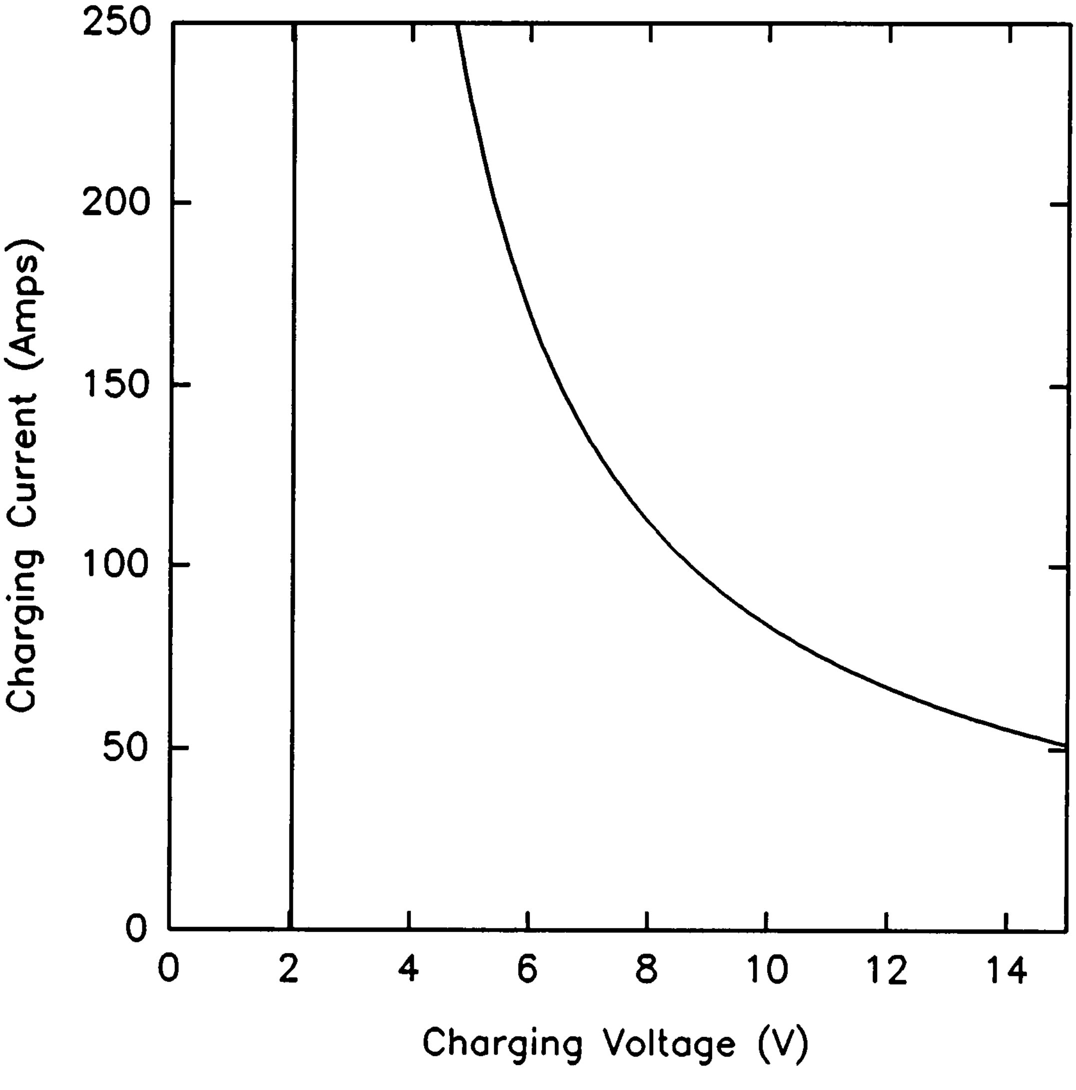


Figure 3.13 L_S Charging Current vs Voltage

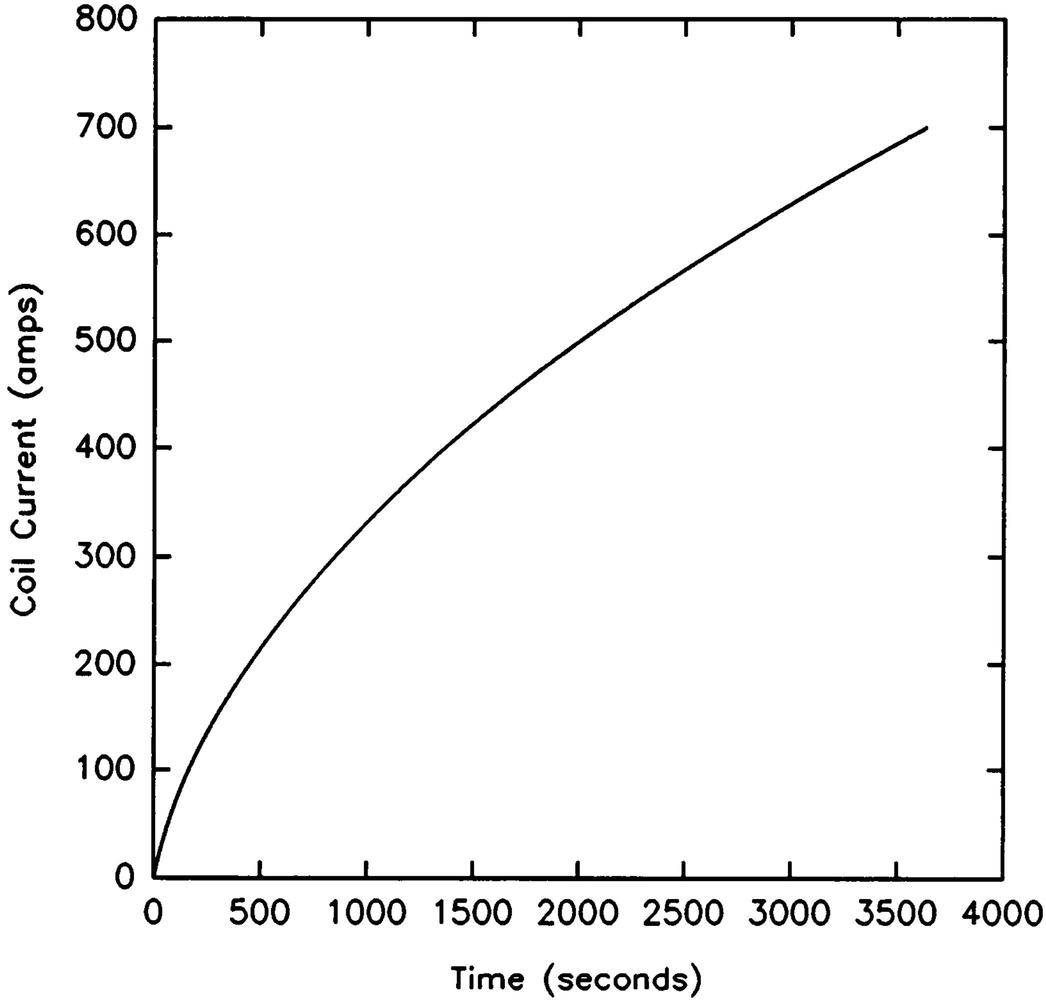


Figure 3.14 Charging Current vs Time

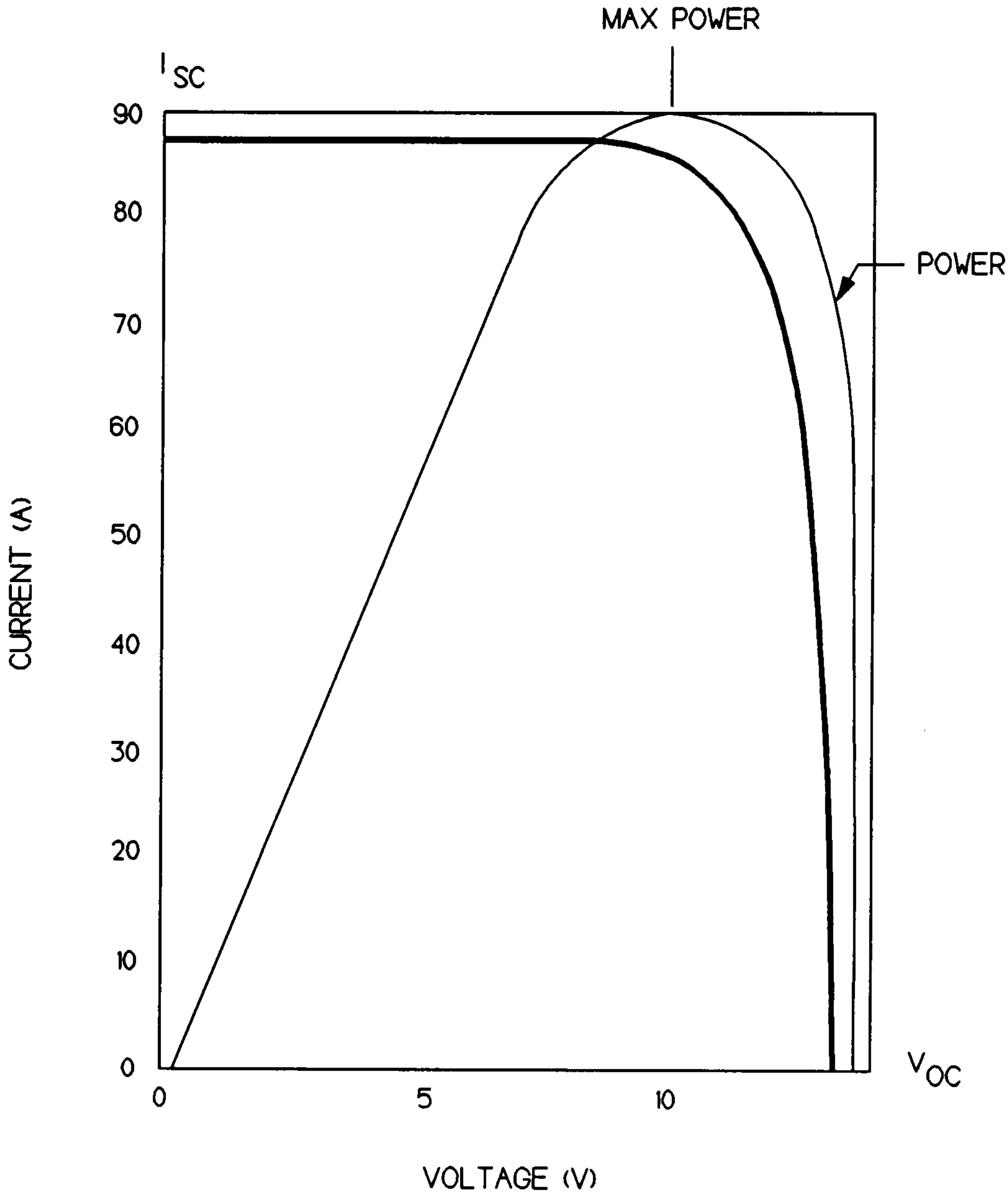


Figure 3.15 Charging Array Power/Voltage/Current Characteristics

Figure 3.7). The construction of a separate solar array with which to charge L_S provides a more efficient mechanism than utilizing a single large array. It also serves to isolate the charging circuitry from the load so that there is no effect on the regulation during the time that L_S is charging.

Solving equation (3.2.2-14) for I_{L_S} and taking the positive root gives

$$I_{L_S}= \frac{\sqrt{I_{C_C}L_S(I_{C_C}L_S+ 2V_{C_C}t)} - I_{C_C}L_S}{L_S} \qquad (3.2.2\text{-}16)$$

The right side of equation (3.2.2-8) represents the fraction of the source current utilized in charging L_S, or the efficiency of the charging profile. Substituting equation (3.2.2-16) into (3.2.2-8), integrating over the entire charging time, T, and dividing by the total time available in which to charge L_S results in an equation for the total efficiency for the charging cycle:

$$\eta= \frac{1}{T}\int_0^T \frac{\sqrt{I_{C_C}L_S(I_{C_C}L_S+ 2V_{C_C}t)} - I_{C_C}L_S}{\sqrt{I_{C_C}L_S(I_{C_C}L_S+ 2V_{C_C}t)}}dt \qquad (3.2.2\text{-}17)$$

Solving equation (3.2.2-14) for the source current, I_{CC}, and setting t= T, results in:

$$I_{C_C}= \frac{{I_{L_S}}^2}{2(\frac{V_{C_C}}{L_S}T- I_{L_S})} \qquad (3.2.2\text{-}18)$$

substituting this into equation (3.2.2-17) and solving the expression using the Derive software package gives

$$\eta = 1-\left[\frac{I_{Ls}L_s}{TV_{C_C}}\right] \tag{3.2.2-19}$$

Equation (3.2.2-19) represents the total efficiency of the charging cycle and is graphed in Figure 3.16 as a function of the charging voltage, V_{CC} with I_{LS}= 630 A, L_S= 10 H, and T= 3000 seconds. From Figure 3.16 it can be seen that an average V_{CC} of 9.5 V will result in an efficiency of 78%.

When the circuit is operating as shown in Mode 1, the charging current through L_S can be found by summing voltages around the loop:

$$\frac{1}{C_C}\int i\, dt - v(t_0) + L_S\frac{di}{dt} = 0 \tag{3.2.2-20}$$

It follows that:

$$L_S\,\frac{d^2i}{dt^2} + \frac{1}{C_C}i = 0 \tag{3.2.2-21}$$

$$\frac{d^2i}{dt} + \frac{1}{L_SC_C}i = 0 \tag{3.2.2-22}$$

Solving this differential equation by using the method of constant coefficients gives the charging current as

$$i(t) = A\cos\left(\frac{t}{\sqrt{L_SC_C}}\right) + B\sin\left(\frac{t}{\sqrt{L_SC_C}}\right) \tag{3.2.2-23}$$

with $i(0)= I_0(n)= A$ where $I_0(n)$ is the initial current at the beginning of each incremental charge on L_S and n is the iteration number. Substituting for A:

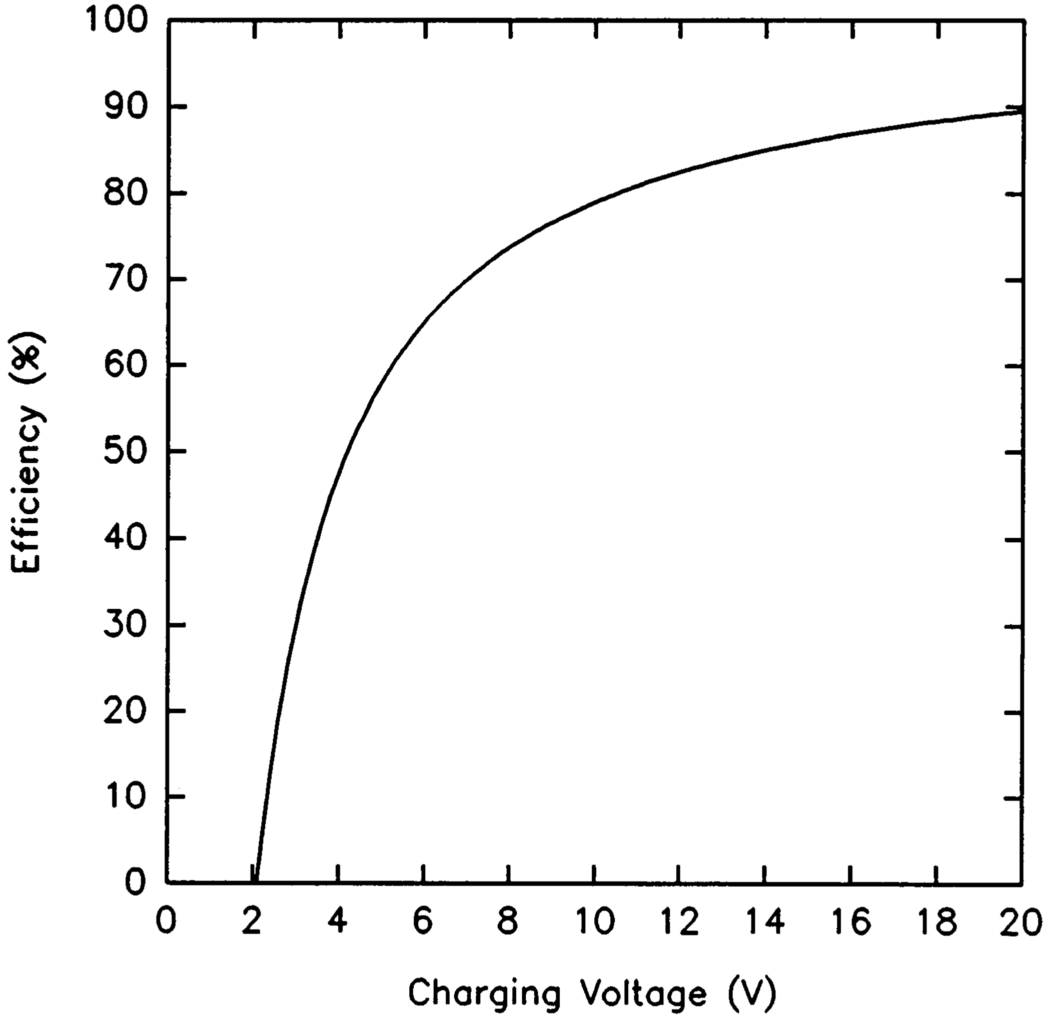


Figure 3.16 Charging Cycle Efficiency vs Charging Voltage

$$i(t) = I_0(n)\ \cos(\frac{t}{\sqrt{L_S C_C}}) + B\ \sin(\frac{t}{\sqrt{L_S C_C}}) \qquad (3.2.2\text{-}24)$$

taking the first derivative:

$$\frac{di(t)}{dt} = -\frac{I_0(n)}{\sqrt{L_S C_C}}\ \sin(\frac{t}{\sqrt{L_S C_C}}) + \frac{B}{\sqrt{L_S C_C}}\ \cos(\frac{t}{\sqrt{L_S C_C}}) \qquad (3.2.2\text{-}25)$$

at time t= 0:

$$\frac{di(0)}{dt} = \frac{B}{\sqrt{L_S C_C}} \qquad (3.2.2\text{-}26)$$

with $v_o(0)$ equal to the initial voltage across C_C and L_S with Sc closed:

$$v_o(0) = L_S \frac{di(0)}{dt} \qquad (3.2.2\text{-}27)$$

By rearranging (3.2.2-27) and substituting into (3.3.2-26):

$$B = V_0 \sqrt{\frac{C_C}{L_S}} \qquad (3.2.2\text{-}28)$$

Finally it is established that:

$$i(t) = I_0(n)\ \cos(\frac{t}{\sqrt{L_S C_C}}) + V_0 \sqrt{\frac{C_C}{L_S}}\ \sin(\frac{t}{\sqrt{L_S C_C}}) \qquad (3.2.2\text{-}29)$$

The current reaches a peak at $i'(t) = 0$. Setting the derivative equal to 0 and solving for $t_{(max)}$ gives

$$t_{(max)} = \sqrt{L_S C_C}\ \tan^{-1} \frac{\sqrt{V_0^2 \frac{C_C}{L_S}}}{I_0(n)} \qquad (3.2.2\text{-}30)$$

The current through L_s will increase to a higher value each time Sc closes and it is charged. From equation (3.2.2-30) it follows that the time required to reach each subsequent current peak on the coil, L_s, decreases from cycle to cycle. This charging cycle necessitates the use of the micro-controller for monitoring and control functions.

Allowing the charging current into L_s to reach a maximum value, which corresponds to a voltage across C_c of zero proves to be an inefficient charging method. It can be seen from the charging efficiency curve, Figure 3.16, that if a minimum efficiency of between 75%-80% is targeted then V_{CC} cannot decrease below ≈ 8 V. It is also noted from the graph that the voltage must increase a relatively substantial amount to obtain a much greater efficiency than 80%. By limiting ΔV_{CC} to 1 V, between 9 and 10 volts, the targeted efficiency is achieved.

From equations (3.2.2-6) and (3.2.2-7) the time necessary to accomplish one charging cycle is given by

$$\Delta t_{C_C} + \Delta t_{L_S} = C_C \Delta V_{C_C} \left[\frac{1}{I_{C_C}} + \frac{1}{i_{L_S}} \right] \qquad (3.2.2\text{-}31)$$

with I_{CC}= 89 and i_{LS} given by equation (3.2.2-16), the charging cycle frequency can be obtained as a function of time. This is shown in Figure 3.17 with a value for C_c selected to be 1 mF. At 50 minutes, the end point of the superconductors charging period, the charging cycle frequency is ≈ 78 KHz, corresponding to a time of 12.8 μsec. From equation (3.2.2-6)

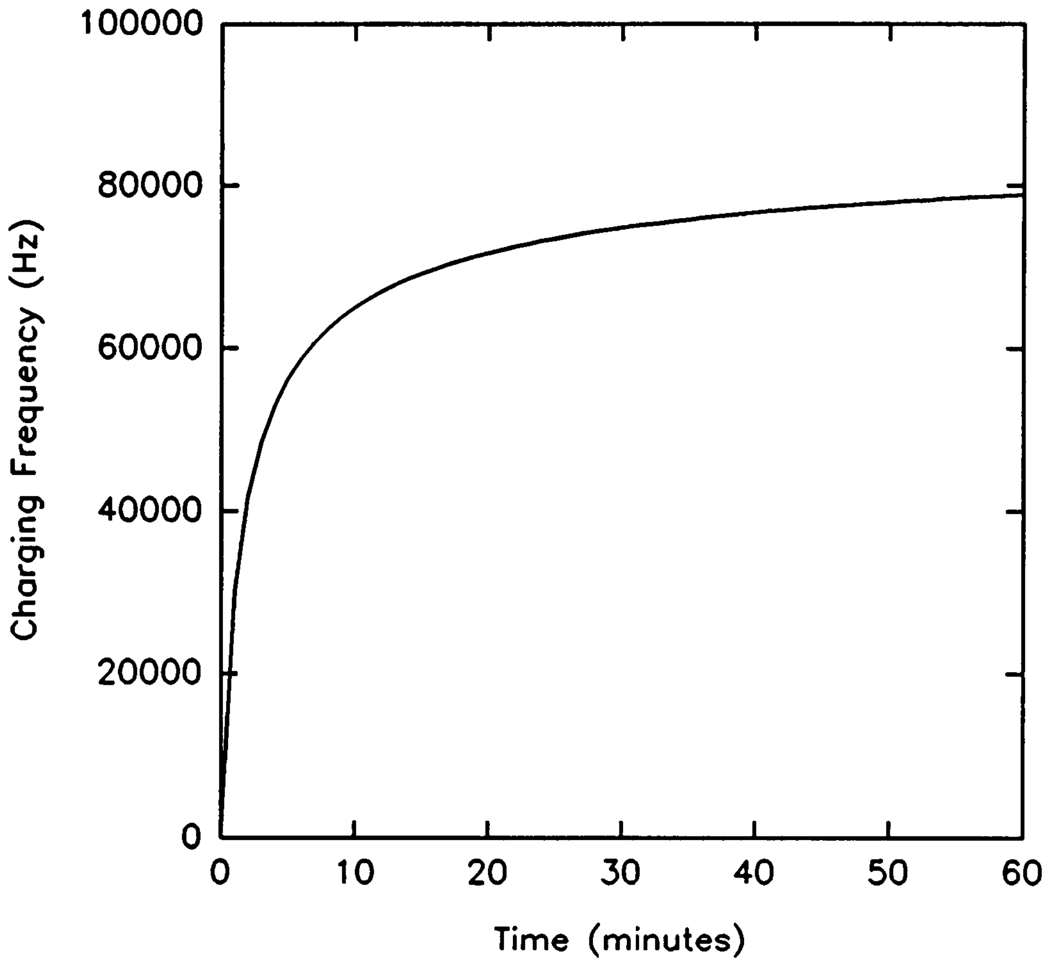


Figure 3.17 Charge Cycle Frequency vs Time

the time required for the voltage across C_c to go from 9 to 10 V is

$$t = \frac{1(.001)}{89} = 11.2 \text{ usec} \qquad (3.2.2\text{-}35)$$

At the completion of the charging of L_s the time taken in transferring the energy from C_c to L_s will be a minimum and is approximately (12.8-11.2) μsec= 1.6 μsec. The short switching time at the end of the charging cycle translates into a maximum switching frequency and higher power losses in the switches. The maximum switching power loss will be in Ps when 10 V and 630 A are imposed on it. By equation (3.2.1-10) the loss is calculated:

$$P_S = \frac{10}{2}(630)(300*10^{-9})(78K) = 74 \text{ W} \qquad (3.2.2\text{-}36)$$

3.2.3 Circuit Operation from Superconducting Coil

In the final operational mode, when the load is operating from the superconducting coil, L_S, the circuit shown in Figure 3.18 will be active.

In this configuration, the energy available from L_S will be used to charge the capacitor, C_1, and the inductor, L_1. Pulse width modulation operates to supply a constant voltage to the load.

The operational modes are similar to those shown in Figure 3.6 when the circuit is operating from the solar cells. The difference to the load is that the source, I_S, has been replaced with the superconducting coil, L_S.

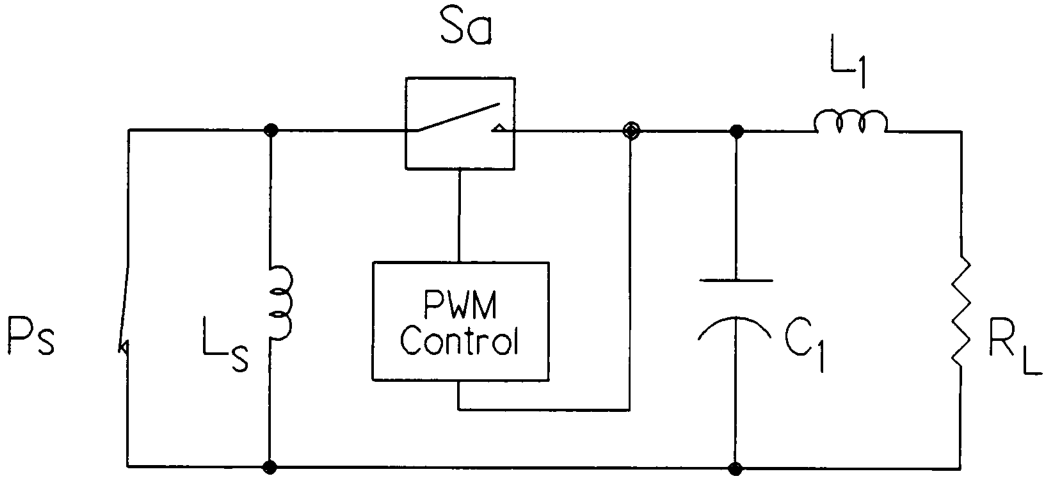


Figure 3.18 Circuit Operation from Superconducting Coil

The two operational topologies are shown in Figure 3.19. Mode 1 occurs when Sa is closed and the voltage at the load is ramping upward. The defining differential equation for the load voltage will be of the third order necessitating the use of substitution techniques or a computer. Summing currents into the two nodes, yields the two equations:

$$-\frac{1}{L_S}\int v_c\,dt = C_1\frac{dv_c}{dt} + \frac{1}{L_1}\int (v_c - v_L)\,dt \tag{3.2.3-1}$$

$$\frac{1}{L_1}\int (v_c - v_L)\,dt = \frac{v_L}{R_L} \tag{3.2.3-2}$$

Taking the first derivative of (3.2.3-2), solving for v_C, and substituting the result into (3.2.3-1) results in the defining differential equation for the load voltage,

$$\frac{d^3v_L}{dt^3} + \frac{R_L}{L_1}\frac{d^2v_L}{dt^2} + \frac{1}{C_1L_1}\left(1+\frac{L_1}{L_S}\right)\frac{dv_L}{dt} + \frac{R_L}{C_1L_1L_S}v_L = 0 \tag{3.2.3-3}$$

Utilizing the program in the Appendix, Figures 3.20 through 3.22 graphs the start up, approximate mid point, and end of the period in which L_S is powering the load. Figure 3.22 illustrates the point at which L_S no longer contains the energy necessary to regulate the load.

While operating from the superconducting coil, the switch Sa must be capable of switching at a substantially faster rate than was necessary during the time the load was supported from the solar cells. This is due to the large current with which the coil must be initially charged. Again assuming no drop

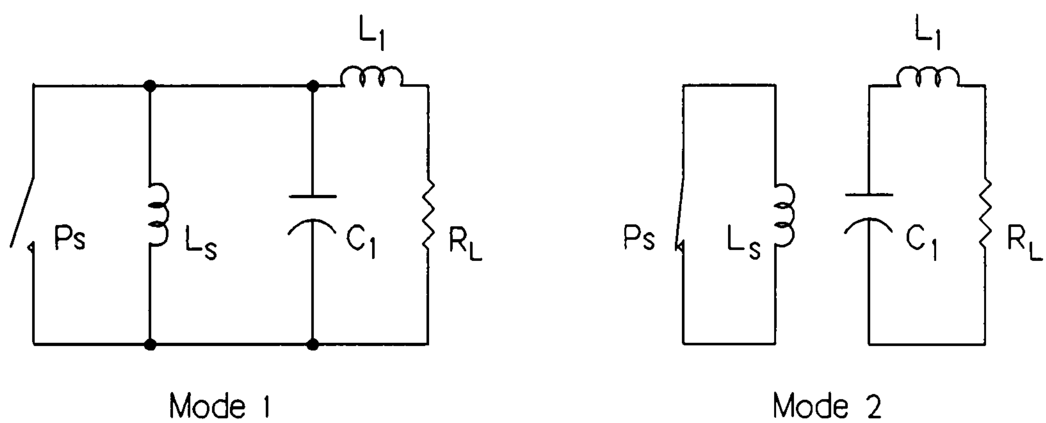


Figure 3.19 Superconducting Coil Operational Topologies

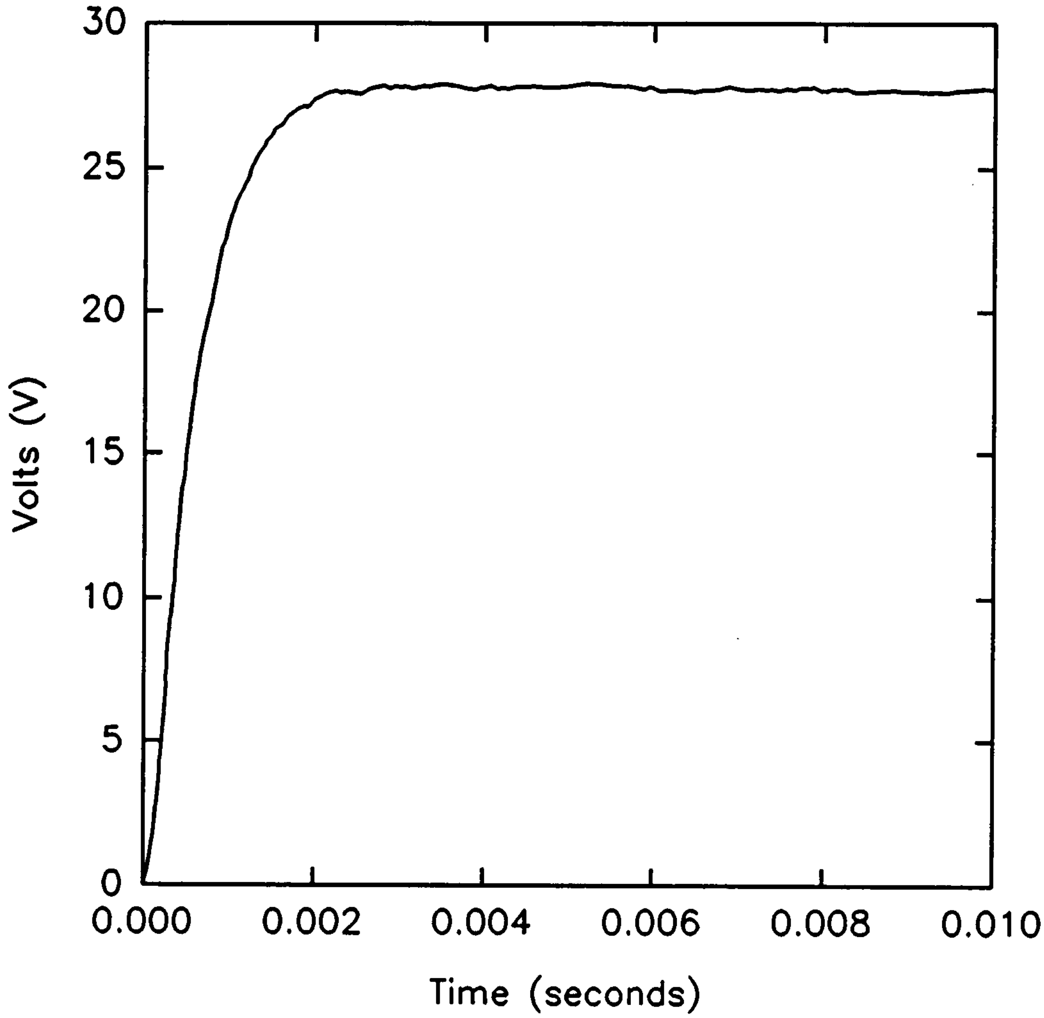


Figure 3.20 Start Up Operation from Coil

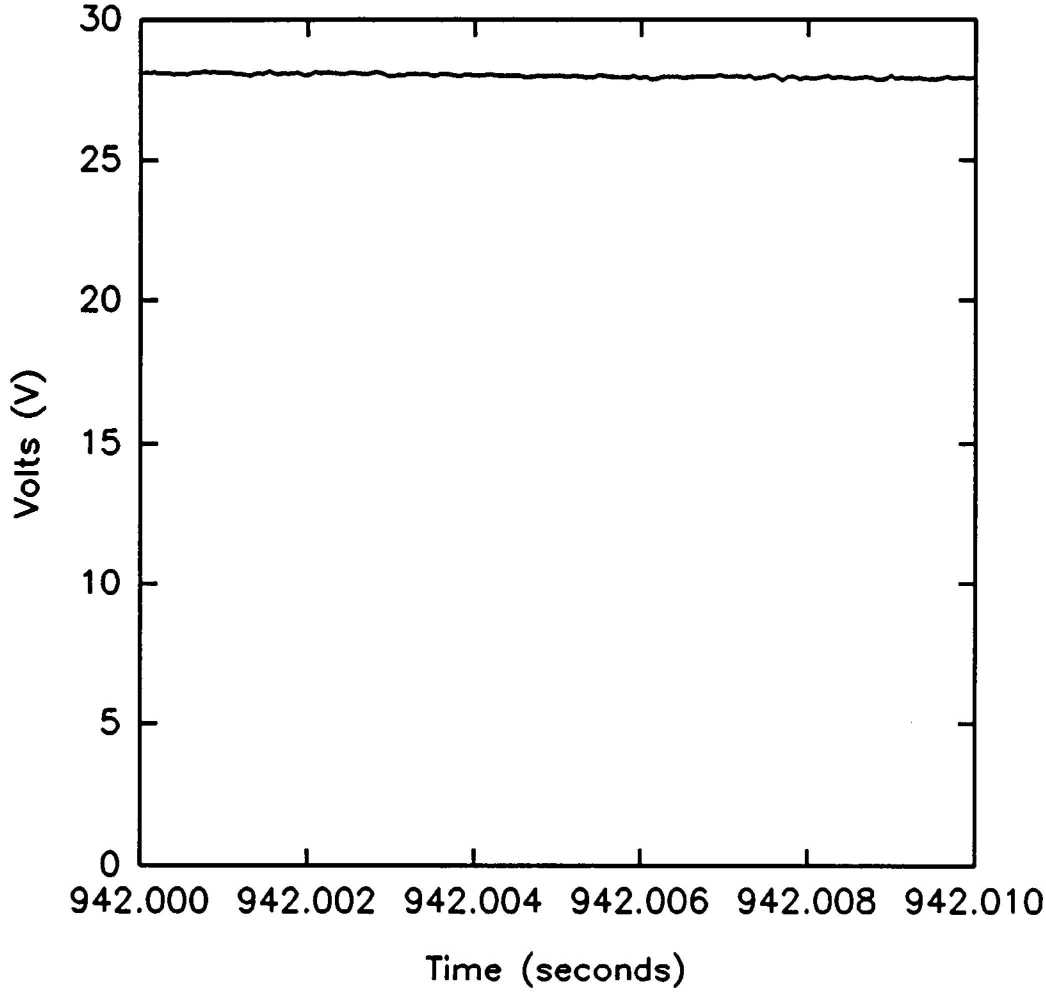


Figure 3.21 Steady State Coil Operation

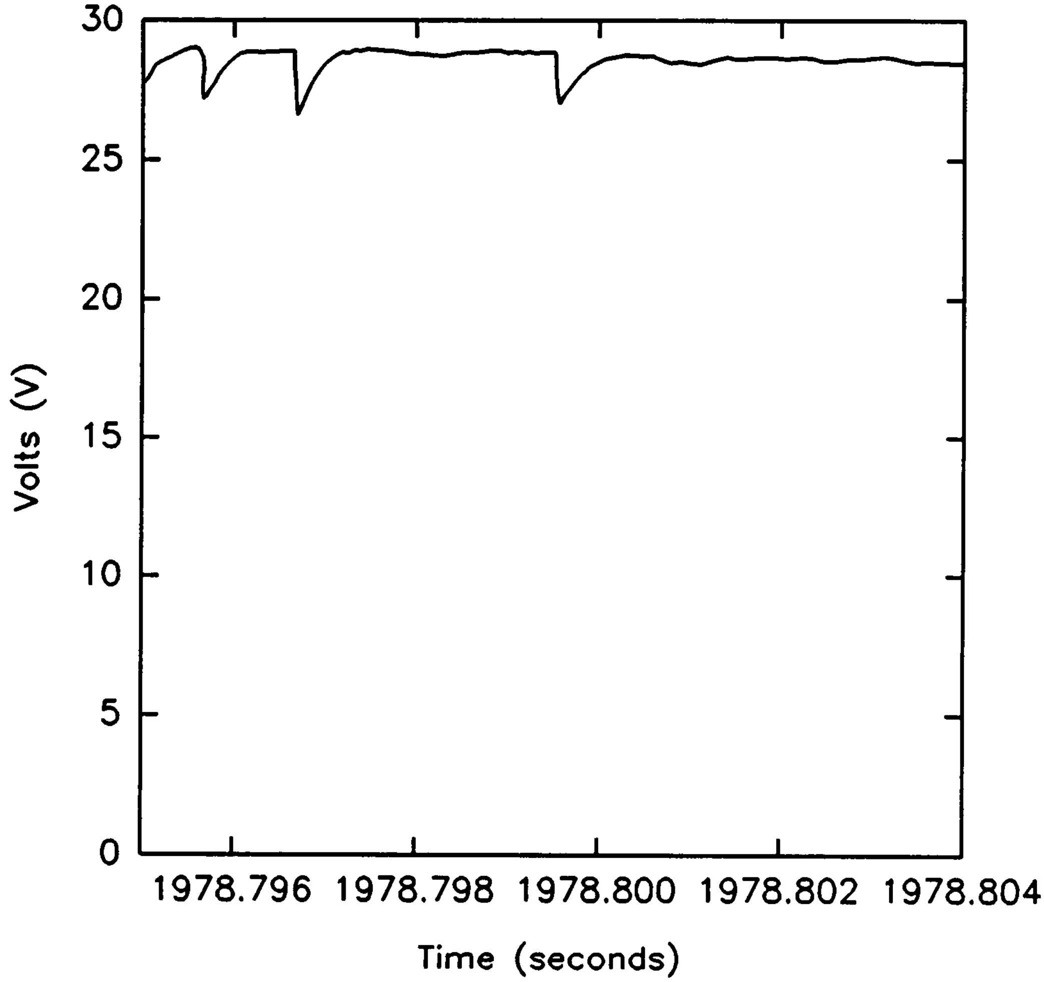


Figure 3.22 Operation from Coil at Edge of Regulation

across L_1 the time for the voltage at the load to go from 27 to 29 V can be approximated as

$$i_{L_S} = C_1 \frac{\Delta V_{C_1}}{\Delta t} + I_{R_L} \qquad (3.2.3\text{-}4)$$

$$\Delta t = \frac{5m(2)}{630-35} = 16.8\ \mu sec \qquad (3.2.3\text{-}5)$$

The time for the voltage at the load to go from 29 to 27 V is the same as found from equation (3.2.1-7); .805 msec. The maximum switching frequency is therefore

$$\frac{1}{.805\ msec + 16.8\ \mu sec} = 1.22\ KHz \qquad (3.2.3\text{-}6)$$

which results in a maximum switching loss in Sa of

$$P_S = \frac{30}{2}(630)(300*10^{-9})(1.22K) = 3.46\ W \qquad (3.2.3\text{-}7)$$

As the current in L_S decays from its initial value the duty cycle on Sa will increase and the switching frequency and power dissipation in Sa will correspondingly decrease.

When Sa closes L_S is in parallel with C_1 and a maximum voltage of approximately 30 V is placed across L_S. This violates the previously imposed constraint of maintaining an upper limit of only 10 V across L_S. Whether or not this higher voltage will cause the superconducting coil to go "normal" is unknown. If this proves to be the case one possible solution is to insert a boost circuit between L_S and C_1 as shown in Figure 3.23.

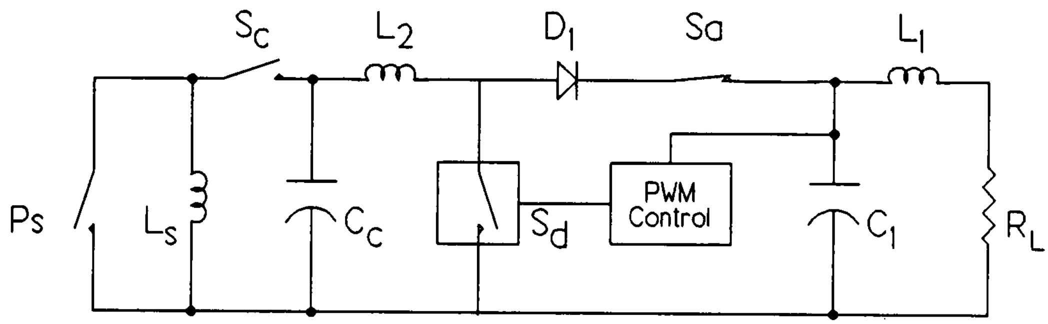


Figure 3.23 Operation from Boost Circuit

When the switches Sc and Sd are closed and Ps is open L_2 and C_c will charge to a maximum of 10 V from L_s. The diode D_1 will be reversed biased allowing the load to be isolated from L_2 and C_c during this time. When C_c has reached 10 V, Sc and Sd will open, Ps will close and D_1 will become forward biased. C_1 and L_1 will then receive energy from the boost circuit.

3.3 Complete Circuit

Figure 3.24 shows the schematic for the power conversion circuitry design without the boost circuitry discussed in the previous section.

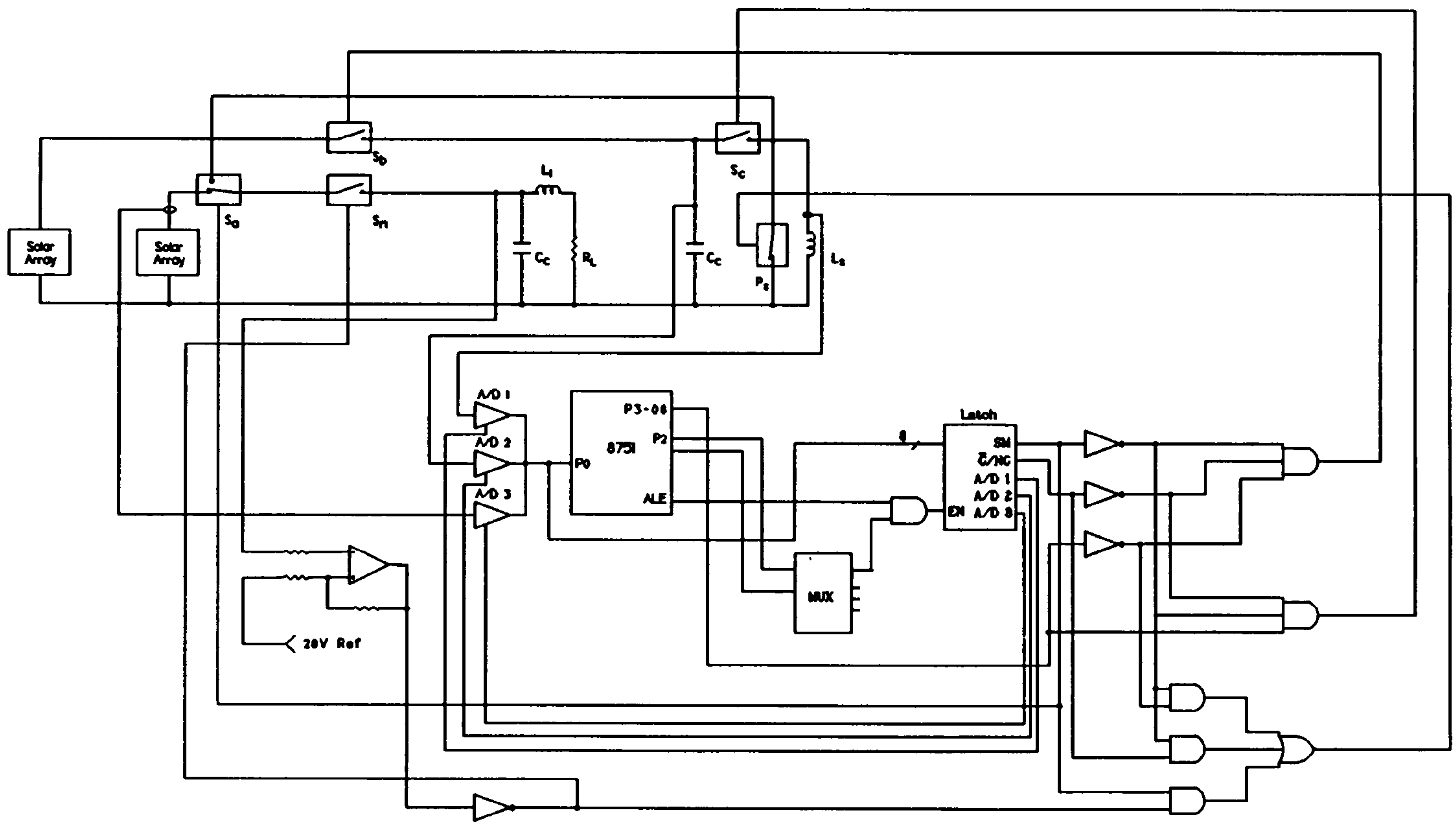


Figure 3.24 PCC Circuitry

CHAPTER IV

CONCLUSION

4.0 General Comments

Appreciable technical challenges exist with the actual realization of the power conversion circuitry proposed in this thesis.

The major obstacle to be overcome before the circuitry described can be implemented is that of developing a high temperature superconductor. The design of the PCC relies upon the eventual availability of a superconducting coil that can withstand the unique rigors of a space environment. This area of research is currently being explored at numerous institutions worldwide.

The development of the superconducting switches is also a technical hurdle to be negotiated for the successful operation of the proposed power system. During the time the solar cells are supplying power to the load, it is necessary that Ps not dissipate any energy circulating through it. This requirement mandates a superconducting switch by definition containing zero losses. Voltage levels and drive circuitry required to operate the switches are as yet to be determined.

The switch involved with the pulse width modulation must additionally be capable of switching at a high frequency with fast t_{on} and t_{off} values while withstanding high stresses due to the large switch current and switch voltages occurring. They must also be able to operate in unison at these high speeds. Whether or not the switching speeds necessary will be realizable is unknown at this writing. Some work currently being done in this field has already been cited in this paper with a reported switching frequency of 100 kHz.

Another problem that needs to be resolved is that of the instrument by which the current through Ps will be sensed. Although Hall Effect current monitors are available that can perform the task with the current levels proposed, it is questionable if any can do so while immersed in the cold bath containing L_S and Ps.

The energy storage components utilized for this design must be capable of handling high currents as well as the large current and voltage spikes created from the switching actions. While paralleling or combining components in series may eliminate stresses to some degree, the trade off in increased weight and volume needs to be considered.

Large di/dt and dv/dt values may spawn EMI problems which need to be dealt with as well. Passive and active shielding schemes may be required to guard against interference with guidance and instrumentation systems on board.

Any snubber design to relieve stresses on the switches must be designed to work with the varying switching speeds that will be occurring in the different operational modes. Energy losses in any snubber design will also need to be taken into account in the energy conservation equations utilized in the PCC design. The high frequencies with which it is necessary to operate the switches will also induce large switching losses during the L_s charging time and the period when the coil is powering the load.

The charging scheme proposed in the course of this paper assumes the charging of C_c to a maximum value of 10 V. It is possible that the charging of the coil will have to be accomplished utilizing a lower voltage. Excess flux migration has been shown to cause superconductors to revert to a normal state. If this proves to be the case, some form of boost circuit should be added as discussed. It may very well be possible that higher voltages could be tolerated by high temperature superconductors so that the coil can be directly connected to the circuit as shown in Figure 3.18. This may prove to be a problem that will have to be resolved empirically.

The component value of L_s= 10 H and the maximum charging current of 630 A were established by equating the amount of energy needed for the load to that stored in the coil. No other electrical losses were taken into account. The maximum charging voltage of 10 V with a deviation of 1 V was developed with a small amount of leeway, given that a charge time of 3000 seconds was used rather than 3600 seconds. This was also based on a charging efficiency of 78 %. Additionally, any other charge/discharge criteria were established solely on power requirements necessary to operate a 1 KW load. The power necessary for refrigeration purposes has yet to be determined and may require a substantial increase in this value.

The interaction of the coil's magnetic field with that of the Earth's will cause an undesirable torque to be exerted upon the satellite. To overcome this will require increased fuel expenditure. Two coils charged with opposite polarities both housed within the same container with enough separation to cancel the generated torque but not the fields might solve this problem.[23]

The physical size of the solar arrays necessary to perform the functions necessary for the PCC design is also a large consideration. For a LEO satellite lifetime of 10 years, it is estimated that the array will need to be oversized by approximately 20%. The degradation of the array is primarily due to damage from cosmic rays resulting in a reduction in the electron hole pairs in the silicon cells.

4.1 Suggestions

A more complex method of controlling the switching of Sa is necessary if the load will be varying. Due to L_1 the load current cannot change immediately so the load voltage will fluctuate if R_L varies. A more efficient feedback control method to regulate the load can be realized by eliminating the 28 V reference to the comparator in the PWM scheme. The controller can be utilized to monitor the voltages at multiple nodes in the circuit and based on these values an algorithm could be developed to output a voltage via a D/A to be used as a reference. This would result in better regulation with fluctuating loads.

Another possible alteration to the existing design would be to maintain a quiescent current level in L_S so that the load would never be in danger of falling out of regulation. This would involve initially charging L_S to a higher current level. If the coil was charged to 640 A then the current in L_S would never drop below approximately 100 A.

LIST OF REFERENCES

REFERENCES

1. Palmer, D.N., "Downsized Superconducting Magnetic Energy Storage Systems", Proceedings of the 24th Intersociety Energy Conversion Engineering Conference, 1989, pg. 453-458

2. Faymon, K.A., and Rudnick, S.J., "High Temperature Superconducting Magnetic Energy Storage for Future NASA Missions", Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference, 1988, pg. 511-514

3. Boom, R.W., Eyssa, Y.M., McIntosh, G.E., "Superconductive Energy Storage for Space Applications", Final Report to the National Aeronautics and Space Administration, Lewis Research Center for Grant NAG 3-170, March 30, 1981-July 29, 1982

4. Masuda M., Shintomi T., "Engineering Testing Plant for Superconducting Magnetic Energy Storage", Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference, 1988, pg. 541-546

5. Fickett, F.R., "Space Applications of Superconductivity: High Field Magnets", Cryogenics, Vol. 19, December 1979, pg. 691-701

6. Pope, W.L., Smith, L.H., Smoot, G.F., Taylor C.E., "Superconducting Magnet and Cryostat for Space Applications", Advanced Cryogenics Engineering, Vol. 20, 1975, pg. 47-60

7. Borger, W.V., Massie, L.D., "Space Power Technology for the 21st Century (SPT-21)", Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference, 1988, pg.511-513

8. Fickett, F.R., "Space Applications of Superconductivity: High Field Magnets", Cryogenics, Vol 19, December 1979, pg. 691-701

9. Pope, W.L., Smith, L.H., Smoot, G.F., Taylor C.E., "Superconducting Magnet and Cryostat for Space Applications", Advanced Cryogenics Engineering, Vol. 20, 1975, pg. 47-60

10. Kushner, M.J., and Petr, R.A., "Cryogenic Semiconductor Switched Tyratron Inverter Switches for Superconducting Inductive Energy Storage Systems", Proceedings of the 24th Intersociety Energy Conversion Engineering Conference, 1989, pg. 541-544

11. Bailey, J.M., Cates, M.R., Divan, D.M., McCleer, P., Young, R.W., "Adjustable Speed Drive Design using High Frequency High Temperature Superconducting Thin Film Devices", Oak Ridge National Lab Report

12. Balachandran, U., Dederer, J.T., Eckels, P.W., Hull, J.R., Lanagan M.T., Poeppel, R.B, Singh, J.P., Singh, S.K., R.B., Wu, J.L., Youngdahl, C.A. "Design and Testing of a High Temperature Superconducting Current Lead", Paper Delivered at 1990 Applied Superconductivity Conference, Snowmass Village, Colorado, September 24-28,1990

13. Boom, R.W., and Eyssa Y.M., "Considerations of a Large Force Balanced Magnetic Energy Storage System" IEEE Transactions on Magnetics, Vol. Mag-17, No. 1, January 1981, pg. 460-462

14. Boom, R.W., Eyssa, Y.M., McIntosh, G.E., "Superconductive Energy Storage for Space Applications", Final Report to the National Aeronautics and Space Administration, Lewis Research Center for Grant NAG 3-170, March 30, 1981-July 29, 1982

15. Golden, R.L., "Superconducting Magnets for Space Flight", Research Goals for Cosmic-Ray Astrophysics in the 1980s, European Space Agency ESRO SP-109, Available as N76-12934 (1975)

16. Mawardi, O.K., "Design of a Force-Free Inductive Storage Coil", Los Alamos National Lab Report LA-5935-MS, June 1975

17. Laquer, H.L., "High Temperature Superconductivity: What, Why, Where and How", Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference, 1988, pg. 493-499

18. Laquer, H.L., "High Temperature Superconductivity: What, Why, Where and How", Proceedings of the 23rd Intersociety Energy Conversion Engineering Conference, 1988, pg. 493-499

19. Boom, R.W., Eyssa, Y.M., McIntosh, G.E., "Superconductive Energy Storage for Space Applications", Final Report to the National Aeronautics and Space Administration, Lewis Research Center for Grant NAG 3-170, March 30, 1981-July 29, 1982

20. Laquer, H.L., "High Temperature Superconductivity: What, Why, Where and How" Proceedings of the 23rd Intersociety

Energy Conversion Engineering Conference, 1988, pg. 493-499

21. Boom, R.W., Eyssa, Y.M., McIntosh, G.E., "Superconductive Energy Storage for Space Applications", Final Report to the National Aeronautics and Space Administration, Lewis Research Center for Grant NAG 3-170, March 30, 1981-July 29, 1982

22. Mohan, N., Undeland, T.M., Robbins, W.P., <u>Power Electonics: Converters, Applications and Design</u>, John Wiley and Sons, 1989, pg. 14

23. Golden, R.L., "Superconducting Magnets for Space Flight", Research Goals for Cosmic-Ray Astrophysics in the 1980s, European Space Agency ESRO SP-109, Available as N76-12934 (1975)

APPENDIX

```
/*
  Solution to circuit simulation 6-11-92
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include <time.h>
#include <string.h>
#include <graph.h>
#include <conio.h>
#include <malloc.h>
#include <errno.h>

struct videoconfig vc;

double y[20], yp[20];    // main variable & derivitives
double yt[20], y2[20], y3[20], y4[20]; // work space
double py[20];        // plot temps

            // color assignments
#define black       0
#define blue        1
#define green       2
#define cyan        3
#define red         4
#define violet      5
#define brown       6
#define white       7
#define grey        8
#define Blue        9
#define Green       10
#define Cyan        11
#define Red         12
#define Magenta  13
#define Yellow      14
#define White       15

int plotcolor[16] = { black, blue, green, cyan, red,  violet, white, brown, grey, Blue, Green, Cyan,
Red, Magenta, White, Yellow };

void init_plot();
void plot_line(int color, int from_x, int from_y, int to_x, int to_y);
char term_plot(char flag);

void step_diffeq(int n, double nxt_y[], double yp[], double y[], double *t, double *inc, double incmax,
        double yt[], double y2[], double y3[], double y4[]);
int func(double yp[], double y[], double *t, double *inc);


/* ****************************************************************************** */
/* **************************  main()  *************************************** */
/* ****************************************************************************** */

int flag;

main(int argc, char *argv[])
{
int         i, n, j, k, maxx, maxy, rows, cols;
double      t, tt, *ny, inc, sa,sb,sc,sd,se,sp,st, incmax;
double      amp_x, off_x, amp_y, off_y;
int         to_x, to_y, pxo, pyo;
char        ch=0,txt[80];


   // initial conditions
   inc = t = 0.;
   n = func(0,y,&t,&inc);    // determine # of equations
```

```
    if (n<=0)
        {
        printf("'func()' does not define the number of equations correctly!\n");
        exit(0);
        }
    incmax = inc;               // save for max change allowed

    if (argc > 1)               //  *** print that sucker !
        {
        printf("Solving %d First Order Differential equations by Runge-Kutta Method.\n", n);

        printf(
">      t          V0          V1          V2          V3         inc\n");
        printf(
"      IL          V0'         V1'         V2'         V3      #itr    abcdept#\n");
        printf(
" ========== =========== =========== =========== =========== =========\n");
        printf(
" ========== =========== =========== =========== =========== ====     ========\n");
        for (i=0; y[5]<=631. ; i++)
            {
            if (kbhit())
                for (i=0; i<2; i++)
                    {
                    ch=getch();
                    if (ch == 'q')          exit(0);
                    }
            tt = t;         // save for print out
            sa = y[11];     sb = y[12];     sc = y[13];     sd = y[14];
            se = y[15];     sp = y[16];     st = y[10];

            step_diffeq(n, y, yp, y, &t, &inc, incmax, yt,y2,y3,y4);

            if ( (i%100) == 0     || y[17]>0. )        // print every 100-th pt or change in switch
                {                                   // or any switch changes
                printf("\n>%10.4f %11.4g %11.4g %11.4g %11.4g %9.3g",
                                tt, y[0],  y[1],  y[2],  y[3],   inc);
                printf("\n %10.6f %11.4g %11.4g %11.4g %11.4g %4d    %1d%1d%1d%1d%1d%1d%1d%1d ",
                                y[4],  yp[0], yp[1], yp[2],  yp[3],    i,
                    (int)y[11], (int)y[12], (int)y[13], (int)y[14],
                    (int)y[15], (int)y[16], (int)y[10], (int)y[17]);
                if ( y[17]>0. )
                    {
                    i = 0;
                    y[17] = 0.;
                    }
                }
            }
        }
    else
        {                       //  *** plot that sucker !
        amp_x =  .001;
        off_x =  .0;
        amp_y =  50.;
        off_y =  10.;

top_plot:
        init_plot();

        maxx = vc.numxpixels;
        maxy = vc.numypixels;
        cols = vc.numtextcols;
        rows = vc.numtextrows;
        pxo =      off_x/amp_x *maxx;
        pyo = (1.0-off_y/amp_y)*maxy;
        _setcolor( Blue );
        for (i=1; i<maxx; i+=3)
            _setpixel( i, pyo );
```

```
_setcolor( White );
_settextposition( 2, 10 );
sprintf( txt, "time = %15.6f %15.6f %15.6f", -off_x, amp_x, amp_x-off_x );
_outtext( txt );
_settextposition( 3, 10 );
sprintf( txt, "vert = %15.6f %15.6f %15.6f", -off_y, amp_y, amp_y-off_y );
_outtext( txt );
_settextposition( 4, 10 );
sprintf( txt, "I2   = %15.6f", tt=y[4] );
_outtext( txt );

   while (1)
       {
       if (kbhit())
           {
           for (i=0; i<2; i++)
               {
               ch=getch();
               if (ch == 'q')          // quit plot plot
                   goto done_plot;
               if (ch == '+')          // restart at 2* time scale
                   {
                   amp_x *= 2.0;
                   off_x = -t;
                   term_plot(0);
                   goto top_plot;
                   }
               if (ch == '-')          // restart at 1/2 time scale
                   {
                   amp_x /= 2.0;
                   off_x = -t;
                   term_plot(0);
                   goto top_plot;
                   }
               if (ch == 'r')          // restart at current posn
                   {
                   off_x = -t;
                   term_plot(0);
                   goto top_plot;
                   }
               if (ch == 'c')          // continue plotting without stoping
                   break;
               }
           }
       step_diffeq(n, y, yp, y, &t, &inc, incmax, yt,y2,y3,y4);

       for (j=0; j<=n ; j++) // *** added extra variable to plot with '=' ************
           {
           to_x =        (   t+off_x)/amp_x  * maxx;
           if (to_x >= maxx)
               {
               off_x = -t;
               if (ch == 'c')      // continue plotting without stoping
                   {
                   term_plot(0);
                   goto top_plot;
                   }
               else
                   {
                   printf("\007");
                  _settextposition( 4, 33 );
                   sprintf( txt, "%15.6f %15.6f", (y[4]-tt)/amp_x, y[4] );
                   _outtext( txt );
                   ch = term_plot(1);
                   }
               if (ch == 'q')      goto done_plot;
               if (ch == '+')      amp_x *= 2.0;
               if (ch == '-')      amp_x /= 2.0;
               goto top_plot;
```

```
                }
                to_y = (1.0 - (y[j]+off_y)/amp_y) * maxy;
                while(to_y > maxy) to_y -= maxy;
                while(to_y < 0    ) to_y += maxy;
                _setcolor( plotcolor[j+10] );
                _setpixel( to_x, to_y );
                }
            }

done_plot:
        term_plot(0);
        }
    while(kbhit()) getch();        // clear buff
}



/* ************************************************************************************** */
/* ************************** step_diffeq()  ******************************** */
/* ************************************************************************************** */



void step_diffeq(int n, double nxt_y[], double yp[], double y[], double *t, double *inc, double incmax,
        double yt[], double y2[], double y3[], double y4[])
/*
    solves a set of first order linear differential equations   by the Runge-Kutta Method.
    The number of equations is determined by calling 'func(0)'

    n       = number of equations to solve
    nxt_y   = Y values for next step
    yp      = first derivitives as computed by func()
    y       = Y values for current step
    *t      = independent variable (updated in calc to next val)
    *inc    = increment to use - may be altered in func(0,0,0,inc)
    maxinc= maximum change (>0) allowed for largest delta-Y or delta-t
             ( *inc is adjusted by *2,/2 to bracket [1.5 * maxinc, 0.7 * maxinc] )
    yt      = work space for temporary Y values       Y' = f(X       ,Y          )
    y2      = work space for temporary Y values       Y' = f(X+inc/2,Y+y1*inc/2)
    y3      = work space for temporary Y values       Y' = f(X+inc/2,Y+y2*inc/2)
    y4      = work space for temporary Y values       Y' = f(X+inc  ,Y+y3*inc  )
*/
{
int i,k,lst;
double t2, inc2;

    lst = k = 0;                          // adjustment counter

top:

    func(0,y,t,inc);                      // call for *inc / parameter adjustment
    inc2 = *inc * .5;                         // later use

    func(yp,  y,   t, inc);               // Y' = f(X       ,Y          )

    t2  = *t + inc2;                      // at half time step for y2 & y3
    for (i=0; i<n; i++)       yt[i] = y[i] + yp[i] * inc2;
    func(y2, yt, &t2, inc);               // Y' = f(X+inc/2,Y+y1*inc/2)

    for (i=0; i<n; i++)       yt[i] = y[i] + y2[i] * inc2;
    func(y3, yt, &t2, inc);               // Y' = f(X+inc/2,Y+y2*inc/2)

    t2  = *t + *inc;                      // at full time step for y4
    for (i=0; i<n; i++)       yt[i] = y[i] + y3[i] * *inc;
    func(y4, yt, &t2, inc);               // Y' = f(X+inc  ,Y+y3*inc  )

    inc2 = 0;
```

```
    for (i=0; i<n; i++)                     // find largest increment
        {
        t2 = yt[i] = *inc * (yp[i] + 2.*(y2[i] + y3[i]) + y4[i])/6.0;
        t2 = *inc * *inc + t2 *t2;          // dist to next pt
        inc2 = max(inc2, t2);
        }
    inc2 = incmax/sqrt(inc2);
    *inc *= inc2;

    if (inc2 > 1.5 || inc2 < 0.7)
        {
//      if (inc2 > 1.0)       printf("*");
//          else                    printf("/");
        if (++k < 4)     goto top;
        }

    *t += *inc;                  // update independent variable
    for (i=0; i<n; i++)          // define new Y values from derivitives
        nxt_y[i] = y[i] + yt[i];

    return;
}


/* ****************************************************************************** */
/* **************************  plot()  ***************************************** */
/* ****************************************************************************** */


#ifndef max
#define max( a, b ) ( ( a ) > ( b ) ? ( a ) : ( b ) )
#endif // max
#ifndef min
#define min( a, b ) ( ( a ) < ( b ) ? ( a ) : ( b ) )
#endif // min



//#undef  _clearscreen
//#define _clearscreen(_GCLEARSCREEN) printf("\x1B[2J")



void init_plot()
{
int     mode = _VRES16COLOR;
int maxx, maxy, cols, rows;


     while( !_setvideomode( mode ) )       /* Find a valid graphics mode  */
         mode--;
     if ( mode == _TEXTMONO )
         exit( 1 );                        /* No graphics available         */
     _getvideoconfig( &vc );
     _clearscreen( _GCLEARSCREEN );

    maxx = vc.numxpixels;
    maxy = vc.numypixels;
    cols = vc.numtextcols;
    rows = vc.numtextrows;
    _setviewport ( 0, 0, maxx-1, maxy-1 );
    _settextwindow ( 1, 1, rows, cols );
    _setwindow( 1, 0.0, 0.0, 1.0, 1.0 );
    _setcolor( Blue );
  _rectangle_w( _GBORDER, 0.0, 0.0, 1.0, 1.0 );

    return;
}
```

```
char term_plot(char ch)
{
int     mode = _VRES16COLOR;


    if (ch)   ch = getch();              // wait for char to return
     _setvideomode( _DEFAULTMODE );
     while (kbhit()) ch = getch(); // clear buffer
     return(ch);
}



void plot_line(int color, int from_x, int from_y, int to_x, int to_y)
{
    _setcolor( color );
    _moveto_w( from_x, from_y );
    _lineto_w(   to_x,   to_y );

}

/* ----------------------------------------------------------------- */




/* ******************************************************************************* */
/* *************************    func()    ***************************************** */
/* ******************************************************************************* */


int func(double yp[], double y[], double *t, double *inc)
/*
    n        = number of equations      (return value)
    yp[n]    = array of Y' values at a single step (to be defined here)
               [if null ptr then these are not calculated but update state & *inc changes]
    y[n]     = array of Y values at a single step -- dependent variable(s)
               [if null ptr then just return # of equations]
    t        = time increment -- independent variable
    *inc     = increment in use for step size NOTE - POINTER!
               [if *inc == 0 && y != null ptr then initialize y[] & *inc]
    func(0,y,t,&inc) will be called at the beginning of each step calculation
               This is the only valid time to change state conditions & *inc!
            If (*inc == 0 && y != 0)  then set initial conditions and define *inc.
*/
/*

    +-------S2-----------------+-------+--S3--R3---+---+ V3
    |                          |       |           |   |
    |            V0 +--S5--R5--+ V2    |           |   |   I2
    |               |                  |           S4  |
    |     +--S1-----+--L1--+V1         |           |   |
    |     |         |      |           |           |   L2
    I0    I1        C1     R1          C2          R4  |
    |     |         |      |           |           |   |
    +-----+---------+------+-----------+-----------+---+

Solar arrays I0 charges L2 & I1 runs load R1 during sunlight
During no sun S5 operates R1 from L2 energy.

Starting equations (balancing currents):
    I1*S1 + S5*(V2-V0)/R5 = C1*V0'+V1/R1
    1/L1*int(V0-V1)dt = V1/R1
    I0*S2 = S5*(V2-V0)/R5 + C2*V2' + S3*(V2-V3)/R3
    S3*(V2-V3)/R3 = S4*V3/R4 + I2
    I2 = 1/L2*int(V3)dt

    */
```

```
{
int n, i;

/*  switched inductor circuit */
static double I0 = 89.0;          // inductor charging current
static double I1 = 46.0;          // operating current
static double I2 = 630.;          // start current in L
static double L1 = 400.e-6;       // 300 uH
static double C1 = 5.e-3;         // 5000 uF
static double C2 = 1.e-3;         // 1000 uF
static double R1 = 0.8;           // .8 ohm
static double R3 = 1.e-6;         // tiny ohm
static double R4 = 1.e-6;         // tiny ohm
static double R5 = 1.e-6;         // tiny ohm
static double L2 = 10.0;          // 10 H (Big Mother!)

static double V0, V0p, V1, V1p, V2, V2p, V3, V3p, I2t, V3t;
static int      j=0, S1, S2, S3, S4, S5, Sp, St;

   n   =   4;          // define number of equations here MOST IMPORTANT !!!
   if (y == 0)         // tell calling pgm the # of equations
       return(n);

   if (*inc == 0)                  // define initial conditions & inc to use
       {                           // called at program start
       y[10] = St = 0;
       y[11] = S1 = 1;             // save starting switch settings
       y[12] = S2 = 1;
       y[13] = S3 = 0;
       y[14] = S4 = 1;
       y[14] = S5 = 0;
       y[16] = Sp = 0;

       y[0] = V0 = 0.;             // cold startup
       y[1] = V1 = 0.;
       y[2] = V2 = 0.;
       y[3] = V3 = (S3*V2/R3-I2)/(S3/R3+S4/R4);
       y[4] = I2;
       y[5] = 0.;
       y[6] = 0.;

       *inc = 0.1;                 // change of .1 V output max allowed
       *t = 0.0;                   // start at t=0

       printf("*** Initializing:\n Inc  = %g\n Time  = %g\n", *inc, *t);
       for (i=0; i<10; i++) printf(" y[%2d] = %g\n", i, y[i]);
       return(n);
       }

   // define variables to readable names
   V0  = y[0];      V1  = y[1];      V2  = y[2];      V3 = y[3];

   flag = 0;
   if (yp == 0)                        // perform non-linear op's here
       {                               // test switches for change of state
       I2t = S3*(V2-V3)/R3-S4*V3/R4;   // New I2 for print
       if (!St && V1 > 28.0)        St = 1;  // start up turn on (once on stays on)

       if (!S1 && V0 < 27.0)
           {           S1 = 1;         flag++;          }   // turn on
       if ( S1 && V0 > 29.0)
           {           S1 = 0;         flag++;          }   // turn off

       if (!S2 && V2 <  9.0)
           {       S2 = S4 = 1;     S3 = 0;      flag++;      }   // turn on
       if ( S2 && V2 > 10.0)
           {       S2 = S4 = 0;     S3 = 1;      flag++;      }   // turn off
```

```
        // maintain constant current across the inductor
        if (flag)
            {
            V3t = y[3];                         // old V3 for print
            y[3] = V3 = (S3*V2/R3-I2)/(S3/R3+S4/R4);
            I2  = S3*(V2-V3)/R3-S4*V3/R4;       // New I2 for print
            if (fabs(I2t-I2) > 1.e-6) printf("\n***\007\n");

            y[10] = St;  y[17] = flag;     // record for printout in main()
            y[11] = S1;  y[12] = S2;  y[13] = S3;
            y[14] = S4;  y[15] = S5;  y[16] = Sp;
            }
        return(n);            // skip defining derivitives if no place to put them
        }

    // calc derivatives

    yp[0] = V0p = (I1*S1 + S5*(V2-V0)/R5 - V1*R1)/C1;
    yp[1] = V1p = (V0-V1)*R1/L1;
    yp[2] = V2p = (I0*S2-S5*(V2-V0)/R5-S3*(V2-V3)/R3)/C2;
    yp[3] = V3p = (S3*V2p/R3-V3/L2)/(S3/R3+S4/R4);

  y[4] = I2  = S3*(V2-V3)/R3-S4*V3/R4;

    return(n);

}
```

```
/*
  Solution to circuit simulation 6-11-92
*/

#include <stdio.h>
#include <stdlib.h>
#include <math.h>

#include <time.h>
#include <string.h>
#include <graph.h>
#include <conio.h>
#include <malloc.h>
#include <errno.h>

struct videoconfig vc;

double y[20], yp[20];    // main variable & derivitives
double yt[20], y2[20], y3[20], y4[20]; // work space
double py[20];       // plot temps

               // color assignments
#define black       0
#define blue        1
#define green       2
#define cyan        3
#define red         4
#define violet      5
#define brown       6
#define white       7
#define grey        8
#define Blue        9
#define Green       10
#define Cyan        11
#define Red         12
#define Magenta  13
#define Yellow      14
#define White       15

int plotcolor[16] = { black, blue, green, cyan, red,  violet, white, brown, grey, Blue, Green, Cyan,
Red, Magenta, White, Yellow };

void init_plot();
void plot_line(int color, int from_x, int from_y, int to_x, int to_y);
char term_plot(char flag);

void step_diffeq(int n, double nxt_y[], double yp[], double y[], double *t, double *inc, double incmax,
        double yt[], double y2[], double y3[], double y4[]);
int func(double yp[], double y[], double *t, double *inc);


/* ****************************************************************************** */
/* ***************************  main()  ****************************************** */
/* ****************************************************************************** */

int flag;

main(int argc, char *argv[])
{
int         i, n, j, k, maxx, maxy, rows, cols;
double      t, tt, *ny, inc, sa,sb,sc,sd,se,sp,st, incmax;
double      amp_x, off_x, amp_y, off_y;
int         to_x, to_y, pxo, pyo;
char        ch=0,txt[80];


   // initial conditions
   inc = t = 0.;
   n = func(0,y,&t,&inc);    // determine # of equations
```

```
    if (n<=0)
        {
        printf("'func()' does not define the number of equations correctly!\n");
        exit(0);
        }
    incmax = inc;                   // save for max change allowed

    if (argc > 1)                   //  *** print that sucker !
        {
        printf("Solving %d First Order Differential equations by Runge-Kutta Method.\n", n);

        printf(
">      t          V0          V1          V2          inc\n");
        printf(
"      IL         V0'         V1'         V2'     #itr    apt#\n");
        printf(
" ========== =========== =========== =========== =========\n");
        printf(
" ========== =========== =========== =========== ====     ====\n");
        for (i=0; y[5]<=631. ; i++)
            {
            if (kbhit())
                for (i=0; i<2; i++)
                    {
                    ch=getch();
                    if (ch == 'q')          exit(0);
                    }
            tt = t;         // save for print out
            sa = y[11];       sp = y[12];       st = y[10];

            step_diffeq(n, y, yp, y, &t, &inc, incmax, yt,y2,y3,y4);

            if ( (i%100) == 0     || y[17]>0. )        // print every 100-th pt or change in switch
                {                                 // or any switch changes
                printf("\n>%10.4f %11.4g %11.4g %11.4g %9.3g",
                                tt, y[0],  y[1],  y[2],   inc);
                printf("\n %10.6f %11.4g %11.4g %11.4g %4d     %1d%1d%1d%1d ",
                                y[4],  yp[0], yp[1], yp[2],    i,
                    (int)y[11], (int)y[12], (int)y[13], (int)y[17]);
                if ( y[17]>0. )
                    {
                    i = 0;
                    y[17] = 0.;
                    }
                }
            }
        }
    else
        {                                   //  *** plot that sucker !
        amp_x =  .1;
        off_x =  .0;
        amp_y =  50.;
        off_y =  10.;

top_plot:
        init_plot();

        maxx = vc.numxpixels;
        maxy = vc.numypixels;
        cols = vc.numtextcols;
        rows = vc.numtextrows;
        pxo =       off_x/amp_x *maxx;
        pyo = (1.0-off_y/amp_y)*maxy;
        _setcolor( Blue );
        for (i=1; i<maxx; i+=3)
            _setpixel( i, pyo );

    _setcolor( White );
    _settextposition( 2, 10 );
```

```
sprintf( txt, "time = %15.6f %15.6f %15.6f", -off_x, amp_x, amp_x-off_x );
_outtext( txt );
_settextposition( 3, 10 );
sprintf( txt, "vert = %15.6f %15.6f %15.6f", -off_y, amp_y, amp_y-off_y );
_outtext( txt );
_settextposition( 4, 10 );
sprintf( txt, "I2   = %15.6f", tt=y[4] );
_outtext( txt );

    while (1)
        {
        if (kbhit())
            {
            for (i=0; i<2; i++)
                {
                ch=getch();
                if (ch == 'q')          // quit plot plot
                    goto done_plot;
                if (ch == '+')          // restart at 2* time scale
                    {
                    amp_x *= 2.0;
                    off_x = -t;
                    term_plot(0);
                    goto top_plot;
                    }
                if (ch == '-')          // restart at 1/2 time scale
                    {
                    amp_x /= 2.0;
                    off_x = -t;
                    term_plot(0);
                    goto top_plot;
                    }
                if (ch == 'r')          // restart at current posn
                    {
                    off_x = -t;
                    term_plot(0);
                    goto top_plot;
                    }
                if (ch == 'c')          // continue plotting without stoping
                    break;
                }
            }
        step_diffeq(n, y, yp, y, &t, &inc, incmax, yt,y2,y3,y4);

        for (j=0; j<=n ; j++) // *** added extra variable to plot with '=' ************
            {
            to_x =        (    t+off_x)/amp_x  * maxx;
            if (to_x >= maxx)
                {
                off_x = -t;
                if (ch == 'c')        // continue plotting without stoping
                    {
                    term_plot(0);
                    goto top_plot;
                    }
                else
                    {
                    printf("\007");
                   _settextposition( 4, 33 );
                    sprintf( txt, "%15.6f %15.6f", (y[4]-tt)/amp_x, y[4] );
                    _outtext( txt );
                    ch = term_plot(1);
                    }
                if (ch == 'q')        goto done_plot;
                if (ch == '+')        amp_x *= 2.0;
                if (ch == '-')        amp_x /= 2.0;
                goto top_plot;
                }
            to_y = (1.0 - (y[j]+off_y)/amp_y) * maxy;
```

```
                while(to_y > maxy) to_y -= maxy;
                while(to_y < 0    ) to_y += maxy;
                _setcolor( plotcolor[j+10] );
                _setpixel( to_x, to_y );
                }
            }

done_plot:
        term_plot(0);
        }
    while(kbhit()) getch();        // clear buff
}



/* ************************************************************************************* */
/* **************************   step_diffeq()   ******************************** */
/* ************************************************************************************* */



void step_diffeq(int n, double nxt_y[], double yp[], double y[], double *t, double *inc, double incmax,
        double yt[], double y2[], double y3[], double y4[])
/*
    solves a set of first order linear differential equations    by the Runge-Kutta Method.
    The number of equations is determined by calling 'func(0)'

    n       = number of equations to solve
    nxt_y   = Y values for next step
    yp      = first derivitives as computed by func()
    y       = Y values for current step
    *t      = independent variable (updated in calc to next val)
    *inc    = increment to use - may be altered in func(0,0,0,inc)
    maxinc= maximum change (>0) allowed for largest delta-Y or delta-t
            ( *inc is adjusted by *2,/2 to bracket [1.5 * maxinc, 0.7 * maxinc] )
    yt      = work space for temporary Y values        Y' = f(X       ,Y           )
    y2      = work space for temporary Y values        Y' = f(X+inc/2,Y+y1*inc/2)
    y3      = work space for temporary Y values        Y' = f(X+inc/2,Y+y2*inc/2)
    y4      = work space for temporary Y values        Y' = f(X+inc  ,Y+y3*inc  )
*/
{
int i,k,lst;
double t2, inc2;

    lst = k = 0                         // adjustment counter

top:

    func(0,y,t,inc);                    // call for *inc / parameter adjustment
    inc2 = *inc * .5;                   // later use

    func(yp,  y,   t, inc)              // Y' = f(X       ,Y           )

    t2  = *t + inc2;                    // at half time step for y2 & y3
    for (i=0; i<n; i++)       yt[i] = y[i] + yp[i] * inc2;
    func(y2, yt, &t2, inc);             // Y' = f(X+inc/2,Y+y1*inc/2)

    for (i=0; i<n; i++)       yt[i] = y[i] + y2[i] * inc2;
    func(y3, yt, &t2, inc);             // Y' = f(X+inc/2,Y+y2*inc/2)

    t2  = *t + *inc;                    // at full time step for y4
    for (i=0; i<n; i++)       yt[i] = y[i] + y3[i] * *inc;
    func(y4, yt, &t2, inc);             // Y' = f(X+inc  ,Y+y3*inc  )

    inc2 = 0;
    for (i=0; i<n; i++)                 // find largest increment
        {
```

```
        t2 = yt[i] = *inc * (yp[i] + 2.*(y2[i] + y3[i]) + y4[i])/6.0;
        t2 = *inc * *inc + t2 *t2;     // dist to next pt
        inc2 = max(inc2, t2);
        }
    inc2 = incmax/sqrt(inc2);
    *inc *= inc2;

    if (inc2 > 1.5 || inc2 < 0.7)
        {
//      if (inc2 > 1.0)      printf("*");
//          else                    printf("/");
        if (++k < 4)     goto top;
        }

    *t += *inc;                     // update independent variable
    for (i=0; i<n; i++)             // define new Y values from derivatives
        nxt_y[i] = y[i] + yt[i];

    return;
}


/* ******************************************************************************** */
/* ***************************  plot()  ****************************************** */
/* ******************************************************************************** */


#ifndef max
#define max( a, b ) ( ( a ) > ( b ) ? ( a ) : ( b ) )
#endif // max
#ifndef min
#define min( a, b ) ( ( a ) < ( b ) ? ( a ) : ( b ) )
#endif // min



//#undef  _clearscreen
//#define _clearscreen(_GCLEARSCREEN) printf("\x1B[2J")



void init_plot()
{
int     mode = _VRES16COLOR;
int maxx, maxy, cols, rows;


    while( !_setvideomode( mode ) )  /* Find a valid graphics mode  */
        mode--;
    if ( mode == _TEXTMONO )
        exit( 1 );                  /* No graphics available       */
    _getvideoconfig( &vc );
    _clearscreen( _GCLEARSCREEN );

   maxx = vc.numxpixels;
   maxy = vc.numypixels;
   cols = vc.numtextcols;
   rows = vc.numtextrows;
   _setviewport ( 0, 0, maxx-1, maxy-1 );
   _settextwindow ( 1, 1, rows, cols );
   _setwindow( 1, 0.0, 0.0, 1.0, 1.0 );
   _setcolor( Blue );
  _rectangle_w( _GBORDER, 0.0, 0.0, 1.0, 1.0 );

   return;
}


char term_plot(char ch)
```

```
{
int     mode = _VRES16COLOR;


    if (ch)   ch = getch();                 // wait for char to return
     _setvideomode( _DEFAULTMODE );
     while (kbhit()) ch = getch();         // clear buffer
     return(ch);
}



void plot_line(int color, int from_x, int from_y, int to_x, int to_y)
{
    _setcolor( color );
    _moveto_w( from_x, from_y );
    _lineto_w(    to_x,    to_y );

}

/* ------------------------------------------------------------------ */



/* ************************************************************************ */
/* **************************   func()  ********************************** */
/* ************************************************************************ */



int func(double yp[], double y[], double *t, double *inc)
/*
    n         = number of equations      (return value)
    yp[n]     = array of Y' values at a single step (to be defined here)
                [if null ptr then these are not calculated but update state & *inc changes]
    y[n]      = array of Y values at a single step -- dependent variable(s)
                [if null ptr then just return # of equations]
    t         = time increment -- independent variable
    *inc      = increment in use for step size NOTE - POINTER!
                [if *inc == 0 && y != null ptr then initialize y[] & *inc]
    func(0,y,t,&inc) will be called at the beginning of each step calculation
                This is the only valid time to change state conditions & *inc!
            If (*inc == 0 && y != 0)  then set initial conditions and define *inc.
*/
/*
       Operation of load RL from charged inductor L2:

                  V2            V0        V1
          +-------+---R3---Sa--+-----L1----+
          |       |            |           |
          |       |            |           |
          |       R4           |           |
          L2      |            C1          RL
          |       |            |           |
          |       Sp           |           |
          |       |            |           |
          +-------+------------+-----------+

Starting equations (ballancing currents):
    1/Ls*int(v2)dt = Ps * V2/R4 + Sa * (V2-V0)/R3
    V1/RL = 1/L1 * int(V0-V1)dt
    Sa*(V2-V0)/R3 = C1 * V0' + V1/RL

*/
{
int n, i;

/*  Laura Clark's switched inductor circuit */
```

```
static double I0 = 89.0;          // inductor charging current
static double I1 = 46.0;          // operating current
//static double I2 = 630.;        // start current in L
static double I2 = 36.6;          // start current in L
static double L1 = 400.e-6;       // 300  uH
static double C1 = 5.e-3;         // 5000 uF
static double C2 = 1.e-3;         // 1000 uF
static double RL = 0.8;           // .8       ohm
static double R3 = 1.e-6;         // tiny ohm
static double R4 = 1.e-6;         // tiny ohm
static double R5 = 1.e-6;         // tiny ohm
static double L2 = 10.0;          // 10       H (Big Mother!)

static double V0, V0p, V1, V1p, V2, V2p, V3, V3p, I2t, V2t;
static int      j=0, Sa, Sp, St;

   n   =   3;       // define number of equations here MOST IMPORTANT !!!
   if (y == 0)      // tell calling pgm the # of equations
       return(n);

   if (*inc == 0)                        // define initial conditions & inc to use
       {                                 // called at program start
       y[10] = St = 0;
       y[11] = Sa = 1;                   // save starting switch settings
       y[12] = Sp = 0;

       y[0] = V0 = 0.;                   // cold startup
       y[1] = V1 = 0.;
       y[2] = V2 = (I2+Sa*V0/R3)/(Sp/R4+Sa/R3);
       y[4] = I2;

       *inc = 0.1;                       // change of .1 V output max allowed
       *t = 0.0;                         // start at t=0

       printf("*** Initializing:\n Inc  = %g\n Time  = %g\n", *inc, *t);
       for (i=0; i<10; i++) printf(" y[%2d] = %g\n", i, y[i]);
       return(n);
       }

   // define variables to readable names
   V0  = y[0];       V1  = y[1];       V2  = y[2];

   flag = 0;
   if (yp == 0)                          // perform non-linear op's here
       {                                 // test switches for change of state
       I2t = Sp*V2/R4 + Sa*(V2-V0)/R3;   // New I2 for print
       if (!St && V1 > 28.0)          St = 1;  // start up turn on (once on stays on)

       if (!Sa && V0 < 27.0)
           {           Sa = 1;     Sp = 0;          flag++;          }   // turn on
       if ( Sa && V0 > 29.0)
           {           Sa = 0;     Sp = 1;          flag++;          }   // turn off


       // maintain constant current across the inductor
       if (flag)
           {
           V2t = y[2];                        // old V3 for print
           y[2] = V2 = (I2+Sa*V0/R3)/(Sp/R4+Sa/R3);
           I2 = Sp*V2/R4 + Sa*(V2-V0)/R3;     // New I2 for print
           if (fabs(I2t-I2) > 1.e-3) printf("\n***\007\n");

           y[10] = St;  y[17] = flag;    // record for printout in main()
           y[11] = Sa; y[12] = Sp;
           }
       return(n);        // skip defining derivitives if no place to put them
       }
```

```
    // calc derivatives

    yp[0] = V0p = ( Sa*(V2-V0)/R3 - V1/RL ) / C1;
    yp[1] = V1p = RL / L1 * (V0-V1);
    yp[2] = V2p = (-V2/L2+Sa*V0p/R3)/(Sa/R3+Sp/R4);

  y[4] = I2  = Sp*V2/R4 + Sa*(V2-V0)/R3;

    return(n);
}
```

VITA

Laura Marlino Clark was born in Oak Ridge, Tennessee on January 10, 1954. She attended elementary and secondary schools there and completed three years of work in Biology at the University of Tennessee. In 1975 she enlisted in the Air Force and worked as an electronics communications technician for four years. During this period she completed coursework required for two associate degrees. Following her discharge in 1979 she enrolled at the University of New Mexico and received her BSEE in December 1982. Upon graduation she began employment with Teledyne Camera Systems in Arcadia, California where she worked on a high resolution film to video tape transfer system. In August 1983 she accepted employment with Sperry Aerospace and Marine Systems in Albuquerque, NM where she worked on CRT displays for military aircraft. She remained there until beginning graduate work at the University of Tennessee in August 1989.

She is currently employed at the Oak Ridge National Laboratory in the Applied Technology Division, has two children and resides in Oak Ridge, Tennessee.