2

NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

A PROTOTYPE FAULT DIAGNOSIS SYSTEM FOR NASA SPACE STATION POWER MANAGEMENT AND CONTROL

by

Gina L. Hester

September 1988

Thesis Advisor: Robert B. McGhee

Unclassified
Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization Naval Postgraduate School; 6b Office Symbol *(If Applicable)* 39	7a Name of Monitoring Organization Naval Postgraduate School
6c Address *(city, state, and ZIP code)* Monterey, CA 93943-5000	7b Address *(city, state, and ZIP code)* Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization; 8b Office Symbol *(If Applicable)*	9 Procurement Instrument Identification Number
8c Address *(city, state, and ZIP code)*	10 Source of Funding Numbers: Program Element Number; Project No; Task No; Work Unit Accession No

11 Title *(Include Security Classification)* A Prototype Fault Diagnosis System for NASA Space Station Power Management and Control.

12 Personal Author(s) Gina L. Hester

13a Type of Report	13b Time Covered	14 Date of Report *(year, month,day)*	15 Page Count
Master's Thesis	From To	September 1988	145

16 Supplementary Notation The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms *(continue on reverse if necessary and identify by block number)*
Field	Group	Subgroup	Expert System, Computer Graphics, Communications, Space Station.

19 Abstract *(continue on reverse if necessary and identify by block number*

The Power Management and Distribution System (PMAD) Prototype utilizes a computer graphics interface with a computer expert system running transparent to the user and a computer communications interface that links the two together, all enabling the diagnosis of PMAD system faults. The prototype design is based on the concept that an astronaut on a space station will instruct an expert system through a graphics interface to run a system or component check on the PMAD system. The graphics interface determines which type of evaluations was requested and sends that information through the communications interface to the expert system. The expert system receives the information and, based on the type of evaluation requested, executes the appropriate rules in the knowledge base and sends the resulting status back to the graphics interface and the astronaut. The PMAD System Prototype serves as a proposed training tool for NASA to use in the training of new personnel who will be designing and developing the NASA Space Station expert systems.

20 Distribution/Availability of Abstract [✓ unclassified/unlimited] ☐ same as report ☐ DTIC users	21 Abstract Security Classification Unclassified	
22a Name of Responsible Individual R. B. McGhee	22b Telephone *(Include Area code)* (408) 646-2095	22c Office Symbol 52Mz

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted   security classification of this page
All other editions are obsolete   Unclassified

A Prototype Fault Diagnosis System for NASA Space Station Power Management and Control

Gina L. Hester
Lieutenant, United States Navy
B.S., United States Naval Academy, 1983

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY
(SPACE SYSTEMS OPERATIONS)

from the

NAVAL POSTGRADUATE SCHOOL
September 1988

Author: Gina L. Hester
Gina L. Hester

Approved by: [illegible]
Robert B. McGhee, Thesis Advisor

[illegible]
Lois M. Brunner, Second Reader

[illegible]
Michael J. Zyda, Third Reader

[illegible]
Rudolf Panholzer, Chairman,
Space Systems Academic Group

[illegible]
Gordon E. Schacher,
Dean of Science and Engineering

ABSTRACT

The Power Management and Distribution System (PMAD) Prototype utilizes a computer graphics interface with a computer expert system running transparent to the user and a computer communications interface that links the two together, all enabling the diagnosis of PMAD system faults. The prototype design is based on the concept that an astronaut on a space station will instruct an expert system through a graphics interface to run a system or component check on the PMAD system. The graphics interface determines which type of evaluations was requested and sends that information through the communications interface to the expert system. The expert system receives the information and, based on the type of evaluation requested, executes the appropriate rules in the knowledge base and sends the resulting status back to the graphics interface and the astronaut. The PMAD System Prototype serves as a proposed training tool for NASA to use in the training of new personnel who will be designing and developing the NASA Space Station expert systems.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I would like to thank NASA Ames Research Center in Sunnyvale, California for the support and cooperation of its personnel during the course of this research. Mr. Henry Lum, Chief of the Information Sciences Division, and Mrs. Carla Wong and Mr. William Erickson of the Systems Automony Demonstration Project office provided me with documents, computer resources and their time, all of which were invaluable to me in the development of this prototype. I am thankful to Mr. William McKellar (the NASA Ames staff assistant) and Mr. Robert Jay (the Naval Postgraduate School (NPS) deputy comptroller) for setting up my travel funding account at the Naval Postgraduate School for trips between NPS and NASA Ames.

I would also like to thank the entire NPS computer Science staff for their time and patience in helping me upload and debug alpha-tested software, learn how to utilize the resources in the AI and Graphics and Video labs and troubleshoot PC modem problems over the phone. I would especially like to thank Lois Brunner, Professor Robert McGhee and Professor Michael Zyda for their wonderful support and for sharing their knowledge, wisdom and friendship with me.

I would like to thank Hania La Born for contributing her invaluable time, extraordinary talent and everlasting patience in supporting me during this endeavor. I would like to thank my parents for their calm support over the last two years in comforting phone calls and pleasant visits. I am thankful for LT Beth Allinder. She buoyed me during difficult times and provided me with the type of honest and caring friendship like that of a sister. Finally, I would truly like to thank LT Kevin Scott for his strength and support in helping me cope with and get through unexpected surgery, his undying faith in me and my abilities and for his loving friendship that I will cherish for a lifetime.

Without all of the efforts of these individuals and organizations, none of this would have been possible. Thank you.

I. INTRODUCTION

A. DESCRIPTION

Space... the final frontier. Mankind has always had a desire to explore space. This desire has been accompanied by a dream for a permanent structure in space on which people can live while observing and collecting data on Earth's solar system. This dream has already been realized by the Soviet Union with its currently operational manned MIR Space Station. The United States is attempting to realize this same dream by striving to develop, design and employ a manned space station by the late 1990's.

The space station will need a smooth mechanism for the collection, processing and storage of data and dissemination of information. It will also need a method for monitoring the major systems on the space station, isolating predicted and actual faults, diagnosing the problem and providing a possible solution. Such a system would probably involve an amalgam of three technologies: computer expert systems, computer graphics, and computer networks.

Most major systems on the space station will utilize an expert system. This expert system will contain specific knowledge about the major system to include an indepth model, rules for the diagnosis of system or component failures and the capability to provide possible solutions to diagnosed problems. There will be a simple graphics interface that will be the main interface to the expert system for the astronauts. This will be necessary to enable the United States to save resources on astronaut training on the expert systems. This interface will require some form of networking to allow the graphics interface and the expert system to communicate. It will provide autonomy on the space station, insuring a safe environment that will allow the crew to perform to their maximum potential.

B. PURPOSE.

This thesis presents a high-level fault diagnosis prototype for the NASA Space Station Power Management and Distribution (PMAD) system. This prototype serves as a proposed training tool for NASA in the training of new personnel on space station expert systems.

C. THESIS OUTLINE.

The following outline of this thesis lays a clear path for the discussion of the PMAD system prototype. Chapter II provides a background on the NASA project and Chapter III provides an overview of the PMAD system prototype. Chapters IV through VI describe the software involved in the expert system, the graphics and communications interfaces, respectively. Chapter VII presents the integrated PMAD system prototype utilizing a consolidation of the concepts presented in Chapters IV through VI. It also provides an analysis of the PMAD system prototype and briefly discusses the PMAD System Manual. Chapter VIII closes the thesis with results and ideas for the expansion of the PMAD system prototype. It also discusses proposed NASA utilization of the prototype as well as its practical usage by the United States Navy.

II. NASA PROJECT BACKGROUND

A. AUTOMATION AND SPACE STATION

On July 18, 1984, the United States Congress passed Public Law 98-371 that stated that NASA would identify "specific Space Station systems which advance automation and robotics technologies, not in use in existing spacecraft, and that the development of such systems shall be estimated to cost no less than 10 per centum of the total Space Station costs." [Ref. 1:pp. 80-81] In response to this legislation, NASA established an Advanced Technology Advisory Committee (ATAC) with the express purpose of reporting on the potential uses and impacts that automation and robotics could have in the Space Station program [Ref. 1:p. 81].

The Office of Aeronautics and Space Technology granted approval for the Systems Autonomy Demonstration Program (SADP) in November 1985. The SADP was created to address NASA's need to develop, integrate and demonstrate various technologies for incorporation into the Space Station. The organization that was given the lead on the SADP was the Ames Research Center Information Sciences Office. Four milestone demonstrations are envisioned to occur from 1988 through 1996. Each demonstration includes more difficult tasks in order to integrate more complex autonomous capabilities into a technologically advancing society. [Ref. 1:p. 81]

B. THE SADP

The SADP was set up as a joint venture between the Ames Research Center (ARC) and the Johnson Space Center (JSC), research and operation centers, respectively. ARC functions involve the following:

- "Program-wide planning, scheduling, budgeting, and evaluation
- In-house conduct of a significant portion of the core technology effort
- Coordination of the core technology efforts conducted at sites other that ARC
- Overall management of the development of the automation technologies into software and hardware suitable for integration into the demonstration framework
- Provision of the technical support essential for facilitating the transfer of technologies to future Space Station activities." [Ref. 1:p. 81]

The role of JSC includes:

- "Supply of the application domain expertise necessary to guide the development of software and hardware tools in order to support the demonstration
- Adaptation of the core technology products to the specific demands of the demonstration environment
- Actual conduct of the milestone demonstrations
- Promotion of the transfer of the developed and demonstrated technologies into Space Station applications." [Ref. 1:pp. 81-82]

C. THE THERMAL CONTROL SYSTEM DEMONSTRATION

A major objective of the 1988 demonstration is the automated monitoring, operation, and control of a complete mission operations subsystem [Ref. 1:p. 82]. The Space Station Testbed Facility at JSC was identified as a demonstration site for the Space Station Thermal Management System. A main goal of the Testbed is to develop, test and evaluate these new technologies for the Space Station via ground-base simulations [Ref. 1:p. 82]. As part of these simulations, "a knowledge-based support system has been developed to provide fault recognition, diagnosis and isolation, monitoring and design and configuration aids for the thermal testbed." [Ref. 1:p. 82]

D. THE POWER CONTROL SYSTEM DEMONSTRATION

The 1990 demonstration will focus on the automated control of multiple subsystems. It will show the coordinated control of multiple subsystems. It will also demonstrate operator aids for unanticipated failures, planning and reasoning about nonstandard procedures. NASA has already begun its preparation for this demonstration by beginning the design of the Power Control System (PCS). The PCS Testbed Facility is located at the Lewis Research Center while autonomous technologies for PCS are being developed by ARC. This M.S. thesis presents a high-level prototype for the PCS, the Power Management and Distribution (PMAD) system prototype. An overview of the PMAD system prototype design and associated hardware and software will be discussed in Chapter III. [Ref. 1:p. 83]

III. THE PMAD SYSTEM PROTOTYPE OVERVIEW

The PMAD system prototype involves the use of a *LISt Processing* (*LISP*) machine (which runs expert system software), a color graphics workstation and associated software. Paragraphs III.A through III.C discuss the roles played by these machines and their software in comprising the PMAD System.

A. THE SYMBOLICS 3675 LISP MACHINE

The Symbolics 3675 LISP machine is manufactured by Symbolics, Inc. The PMAD system utilizes the Symbolics because it has the largest memory and disk capacity of the 3600 series Symbolics machines, enabling it to accommodate the software necessary to run the PMAD system.

The Symbolics runs the Genera 7.1 operating system and utilizes the LISP artificial intelligence language. The expert system software available on this machine is the Knowledge Engineering Environment (KEE) Expert System Shell. KEE is a window (screen partition) and mouse oriented program that provides the functionality needed for development of the PMAD system knowledge base. The mouse is a pointing device with three buttons that can perform certain operations and that can be moved around on a flat surface.

The NASA *Model Toolkit* (*MTK*) is additional support software for the PMAD system which sets up a template for the development of a knowledge base for any type of system that can be broken down into components. MTK enables the placement of general icons (picture representations) for these components into a window called the *Library*. The icons in this Library window are used to create specific instances of the components which, when pieced together with connections and connection links, comprise a picture or diagram of the system in another window called the Model. All of the Symbolics software enables the setting up of an environment where fault diagnosis can be performed. The performance of this fault diagnosis will be discussed in Chapter VII.

B. THE SILICON GRAPHICS, INC., IRIS-3120 GRAPHICS WORKSTATION

The IRIS is a high performance color graphics workstation with mouse interface manufactured by Silicon Graphics, Inc. The IRIS runs the ATT system 5.3 version of the UNIX operating system. The IRIS comes with the *Multiple EXposure* (*MEX*) window manager which provides a pop-up menu facility. The operating system and MEX both utilize the *C* structured programming language. The IRIS's main feature is a set of graphics and utility routines that provide high-level and low-level graphics support. All of this software assists in providing the PMAD system with a simple color graphics interface.

C. THE COMMUNICATIONS SOFTWARE INTERFACE

Both of the aforementioned machines utilize the Transmission Control Protocol/Internet Protocol (TCP/IP) standard to communicate with each other. The Symbolics uses LISP functions written on top of TCP/IP to enable ease of use by the user. The IRIS uses TCP/IP and an intercomputer communications package that can be customized to communicate with other IRIS, Symbolics or Texas Instruments Explorer machines.

D. SUMMARY

The associated software for the Symbolics and the IRIS have been discussed in a general fashion in this chapter. In Chapters IV and V the software for these machines is discussed in detail along with a brief discussion on how the different software layers interrelate to support the PMAD system.

IV. SYMBOLICS LISP MACHINE SOFTWARE

A. THE LAYERS

There are four major software layers that the PMAD system utilizes on the Symbolics. Each one plays a significant role in the support of the PMAD system. The following paragraphs give a brief description of each layer.

1. Genera Version 7.1

The operating system on the Symbolics is Genera 7.1. It provides a total operating environment for LISP processing. This includes manipulating the screen using the mouse. The mouse is a pointing device with three buttons that can perform certain operations and that can be moved around on a flat surface. Also, included in the Genera 7.1 environment are the control of the keyboard and the creation and selection of windows. [Ref. 2:pp. 6-9]

2. Common LISP

Common LISP is the result of an attempt to consolidate variations of LISP into a collection of capabilities that could be considered a language. LISP functions are quite similar to a glorified hand calculator; i.e., arguments and an operation to be performed are typed in and LISP does the operation and prints out an answer. It is this evaluative behavior that makes Common LISP a popular language in the Artificial Intelligence field. [Ref. 3:p. xii]

3. The KEE Expert System Shell 3.1

KEE is a development system for building expert systems. An expert system has a knowledge base that is composed of information blocks called *units*. These units contain *slots* which represent information about these units and how that information relates to other units. Slots have values that can contain descriptive information (facts) or procedural information (rules). [Ref. 4:pp. 4-8]

KEE enables the user to not only organize facts and rules in the expert system's knowledge base, but also allows their manipulation through the use of an *inference engine*. [Ref. 5:p. 16]. An inference engine prioritizes facts and rules, executes them and based on the rules' results, adds new facts to the knowledge base. Thus, KEE enables the easy use and expansion of an expert system.

4. The NASA Model Toolkit (MTK)

The Model Toolkit (MTK) is a package developed to be closely integrated with a number of KEE version 3.1 utilities (that provide basic support for a number of MTK functions) in order to provide expert system developers support for designing and implementing expert systems that utilize model-based reasoning. Such reasoning is necessary since many expert system problems in simulation, monitoring, and fault diagnosis concern physical systems. It is this model-based reasoning that MTK uses to organize the PMAD system knowledge base. [Ref. 6:p. 2]

a. The PMAD System Knowlege Base

MTK provides the basic organization for representing the physical and conceptual components that comprise a physical system (*structures*), the ways that these components interact with each other (*connections*) and the crucial measurements (*parameters*) that define how these components can change over time. [Ref. 6:p. 2]

(1) Structures. MTK represents objects to be modelled in a system by units called structures of which there are two types, component and functional (only component structures will be discussed) [Ref. 6:p. 5]. Component structures represent distinct physical objects. Some examples of such objects in the PMAD system are **joints**, **switches** and **batteries**.

(2) Connections. Within MTK there are units called connections which represent how one structure may effect another. Such influences can be the transfer of energy such as heat, electricity or force [Ref. 6:p. 6]. The PMAD system connections are called **electrical.connections**.

(3) Parameters. Significant measurable values in the PMAD system are represented in MTK by parameters. Two types of parameters are used, simple and complex. Simple parameters are used to handle a single qualitative value. Complex parameters are used to handle parameters that need to represent both quantitative and qualitative values. Also, parameters can be associated with both structures and connections. The PMAD system utilizes complex parameters. They are **charge.level**, **power.level** and **voltage.level**. These examples, as well as those in paragraphs IV.A.4.a(1) and IV.A.4.a(2) above, can be seen in the PMAD system knowledge base representation in Figure 1. [Ref. 6:p. 7]

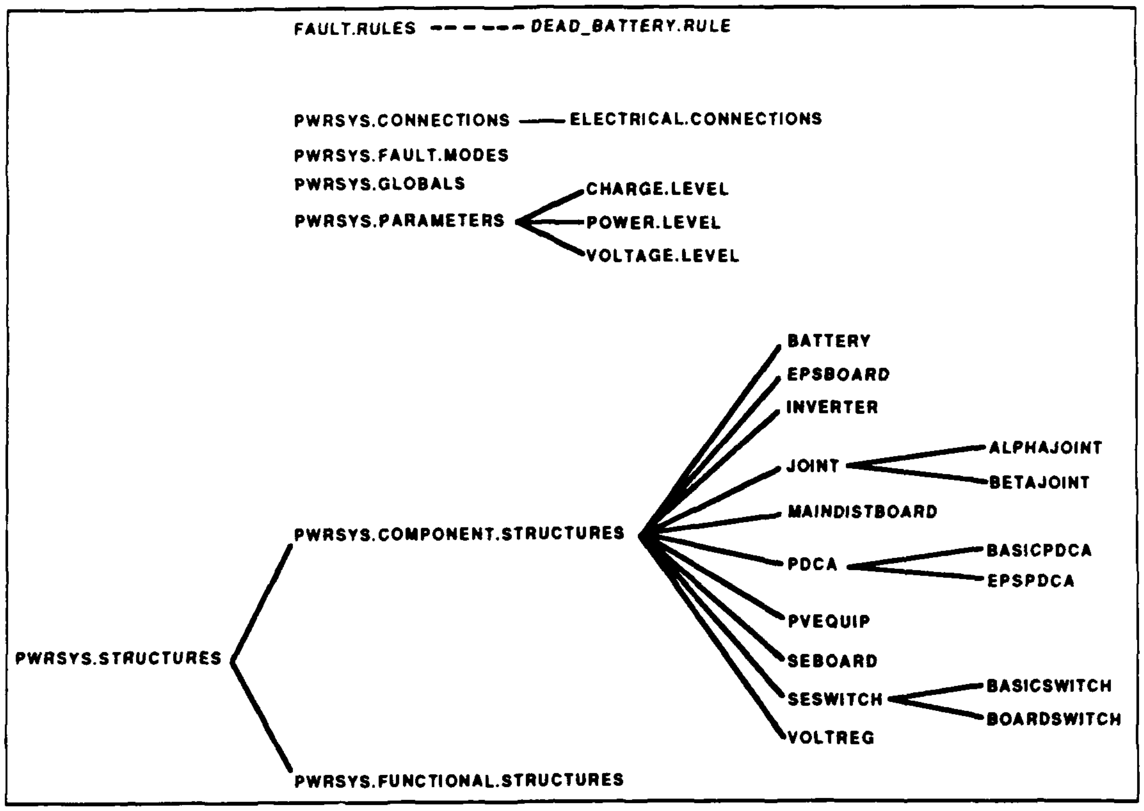


Figure 1. The PMAD System Knowledge Base

Within these parameter units are slots. The slots of the most importance are called **value**, **value.state** and **trend.state**. The **value** slot contains the numerical or range value of the parameter. The **value.state** slot contains the information on the parameter labelled either **negative**, **zero** or **positive**. The **trend.state** slot has information on the parameter of either **steady**, **increasing** or **decreasing**. Examples of these slots are discussed later in this chapter. [Ref. 6:pp. 7-8]

b. The PMAD System Library

The library has knowledge bases and an icon window associated with it. Collectively, the library builds domain-specific representations on top of MTK's generic structures [Ref. 6:p. 1]. The icons in the library window can be modified to exactly resemble physical component structures and their associated connections. The PMAD system library can be seen in Figure 2.

c. *The PMAD System Model*

The model is quite similar to the library, in that it uses definitions made in the library knowledge base, except it contains specific instances of components and defines how these components interact. The PMAD system model can be seen in Figure 3. [Ref. 6:p. 1]

B. THE INTEGRATION OF THE LAYERS

Now that all of the major software layers for the Symbolics that relate to the PMAD system have been explained, the integrated software environment will be discussed. The battery component will be the source of the examples.

The battery component structure (or class) in the PMAD system knowledge base in Figure 1 is represented in the PMAD system library by a battery class icon in Figure 2 that has three connections, **vr.n** (voltage regulator node), **n** (negative) and **p** (positive). The battery component is also represented in the PMAD system model in Figure 3 by another icon that is an instance of the battery class icon in the library in Figure 2. This model icon is shown to be linked or connected to the voltage regulator through the battery's **vr.n** connection. The battery's **n** and **p** connections are unlinked. This instance of the battery is known to the model knowledge base as **battery.1**.

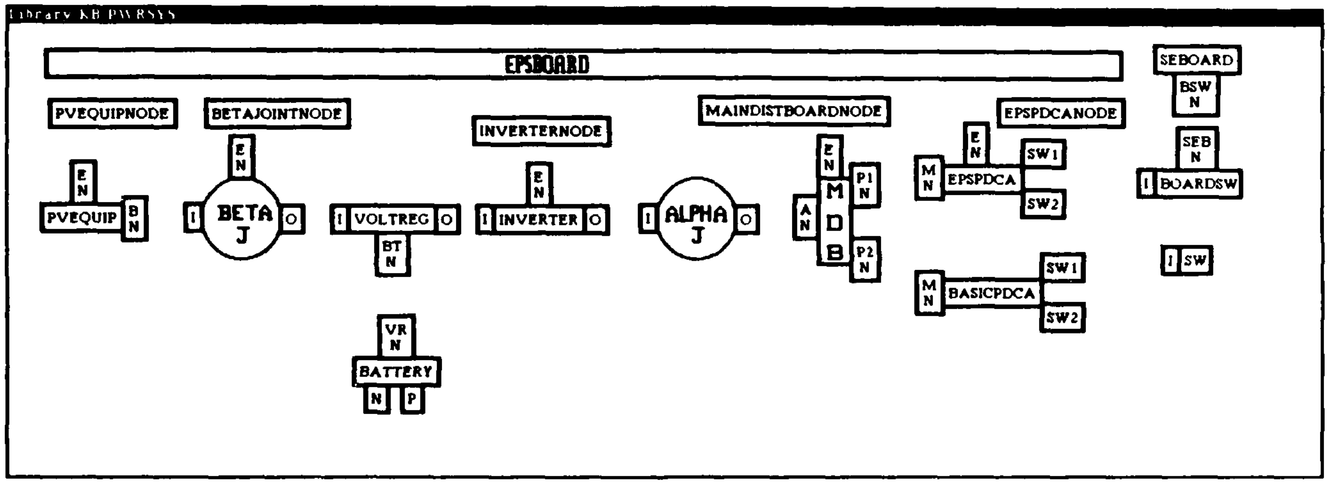


Figure 2. The PMAD System Library

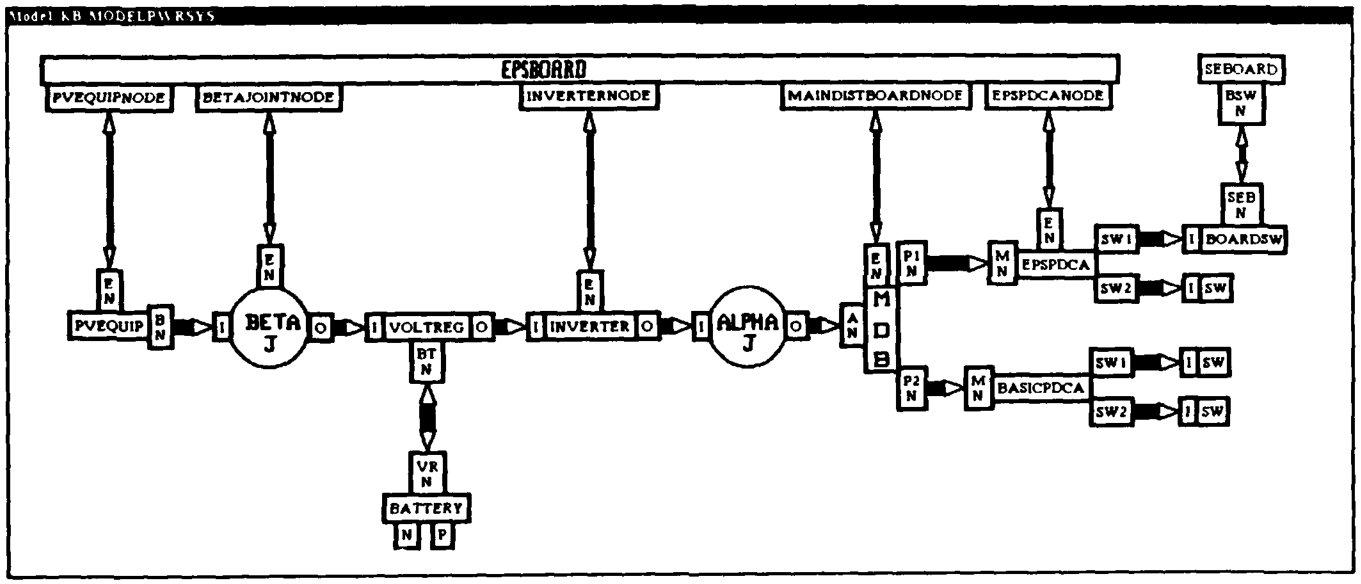
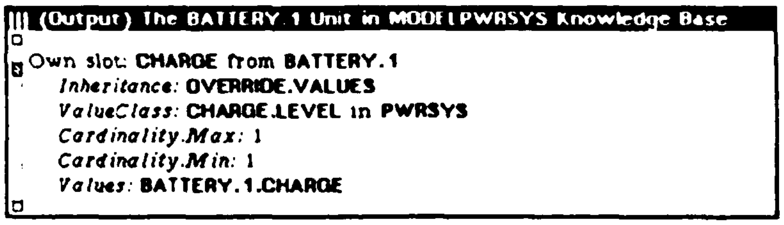


Figure 3. The PMAD System Model

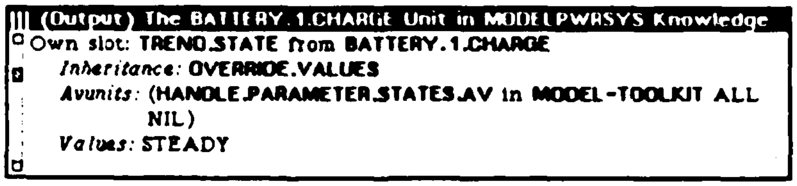
Battery.1 also has parameters that are associated with itself and its connections. There is a unit called **battery.1.charge** which contains information about the **charge** on **battery.1**. The **battery.1.charge** unit mainly keeps track of the **trend.state** of the **charge** on **battery.1**. The **battery.1.vr.n** connection unit has a parameter unit associated with it called **battery.1.vr.n.powerload**. This parameter unit's main function is to keep track of the **value.state** of the **powerload** on the **vr.n** connection on **battery.1**.

Figures 4a and 4b show partial output of the **battery.1** structure and the **battery.1.charge** parameter units and their important slots. Figures 4c and 4d show partial output of the **battery.1.vr.n** connection and the **battery.1.vr.n.powerload** parameter units and their important slots.



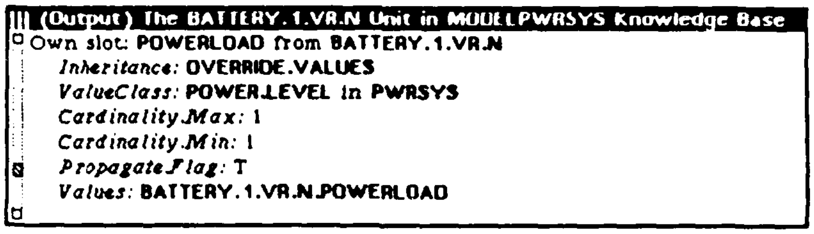
```
(Output) The BATTERY.1 Unit in MODELPWRSYS Knowledge Base
Own slot: CHARGE from BATTERY.1
   Inheritance: OVERRIDE.VALUES
   ValueClass: CHARGE.LEVEL in PWRSYS
   Cardinality.Max: 1
   Cardinality.Min: 1
   Values: BATTERY.1.CHARGE
```

Figure 4a. CHARGE Parameter Slot for BATTERY.1 Unit



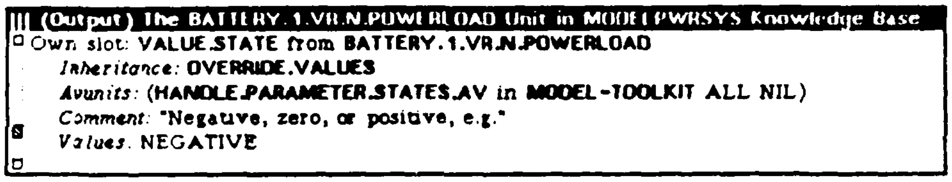
```
(Output) The BATTERY.1.CHARGE Unit in MODELPWRSYS Knowledge
Own slot: TREND.STATE from BATTERY.1.CHARGE
   Inheritance: OVERRIDE.VALUES
   Avunits: (HANDLE.PARAMETER.STATES.AV in MODEL-TOOLKIT ALL
          NIL)
   Values: STEADY
```

Figure 4b. TREND.STATE Slot for BATTERY.1.CHARGE Unit



```
(Output) The BATTERY.1.VR.N Unit in MODELPWRSYS Knowledge Base
Own slot: POWERLOAD from BATTERY.1.VR.N
    Inheritance: OVERRIDE.VALUES
    ValueClass: POWER.LEVEL in PWRSYS
    Cardinality.Max: 1
    Cardinality.Min: 1
    Propagate.Flag: T
    Values: BATTERY.1.VR.N.POWERLOAD
```

Figure 4c. POWERLOAD Parameter for BATTERY.1.VR.N Unit



```
(Output) The BATTERY.1.VR.N.POWERLOAD Unit in MODELPWRSYS Knowledge Base
Own slot: VALUE.STATE from BATTERY.1.VR.N.POWERLOAD
    Inheritance: OVERRIDE.VALUES
    Avunits: (HANDLE.PARAMETER.STATES.AV in MODEL-TOOLKIT ALL NIL)
    Comment: "Negative, zero, or positive, e.g."
    Values: NEGATIVE
```

Figure 4d. VALUE.STATE for BATTERY.1.VR.N.POWERLOAD Unit

V. THE IRIS GRAPHICS WORKSTATION

This chapter discusses the various software layers of the IRIS and how they interrelate.

A. THE LAYERS

There are four major software layers that the PMAD system utilizes on the IRIS. Each one provides a building block on which the PMAD system can be firmly supported. The following paragraphs give a brief descriptive of each layer.

1. UNIX ATT Version 5.3

The PMAD system relies on the UNIX operating system. UNIX allows the user to set up a custom environment that allows more ease of use of the applications on the IRIS. It also is the foundation for many programs since the operating system has many useful tools that can be utilized with the C programming language, the primary language of all UNIX operating system-based machines. [Ref. 7:p. ix]

2. The C Programming Language

C is a programming language that has economy of expression, modern flow of control and data structures and a diverse set of operators. C is not considered a high-level language and is not limited to any particular area of application. It is this generality that makes C more effective and convenient for many tasks than supposedly more powerful languages. [Ref. 7:p. ix]

3. The MEX Window Manager

MEX allows for the creation of several independent displays or windows on the screen of an IRIS workstation [Ref. 8:p. 78]. One of the most useful features of MEX is its pop-up menu facility which enables a clean interface with the three button mouse on the IRIS.

Utilizing MEX and the mouse, the user can click on white space and select options off the main menu depicted in Figure 5a or click on a component or connection (in blue) and select options off the component menu shown in Figure 5b. The main menu and its subordinate menus are displayed in Figure 6. Since the main menu contains all the options that are available at lower level menus, it will drive the following discussion.

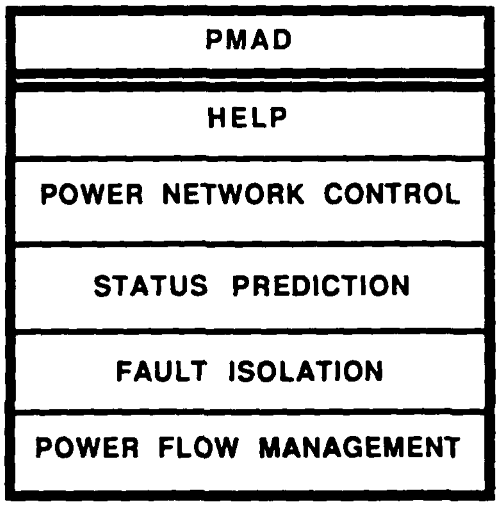


Figure 5a. The PMAD System Graphics Interface Main Menu

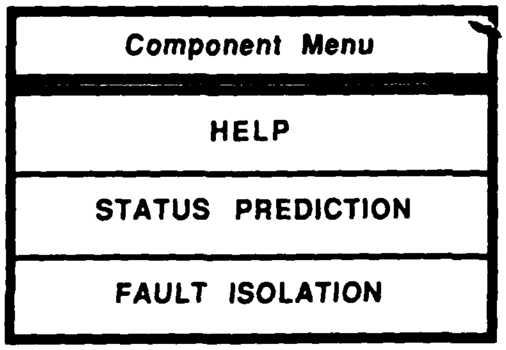


Figure 5b. The PMAD System Graphics Interface Component Menu

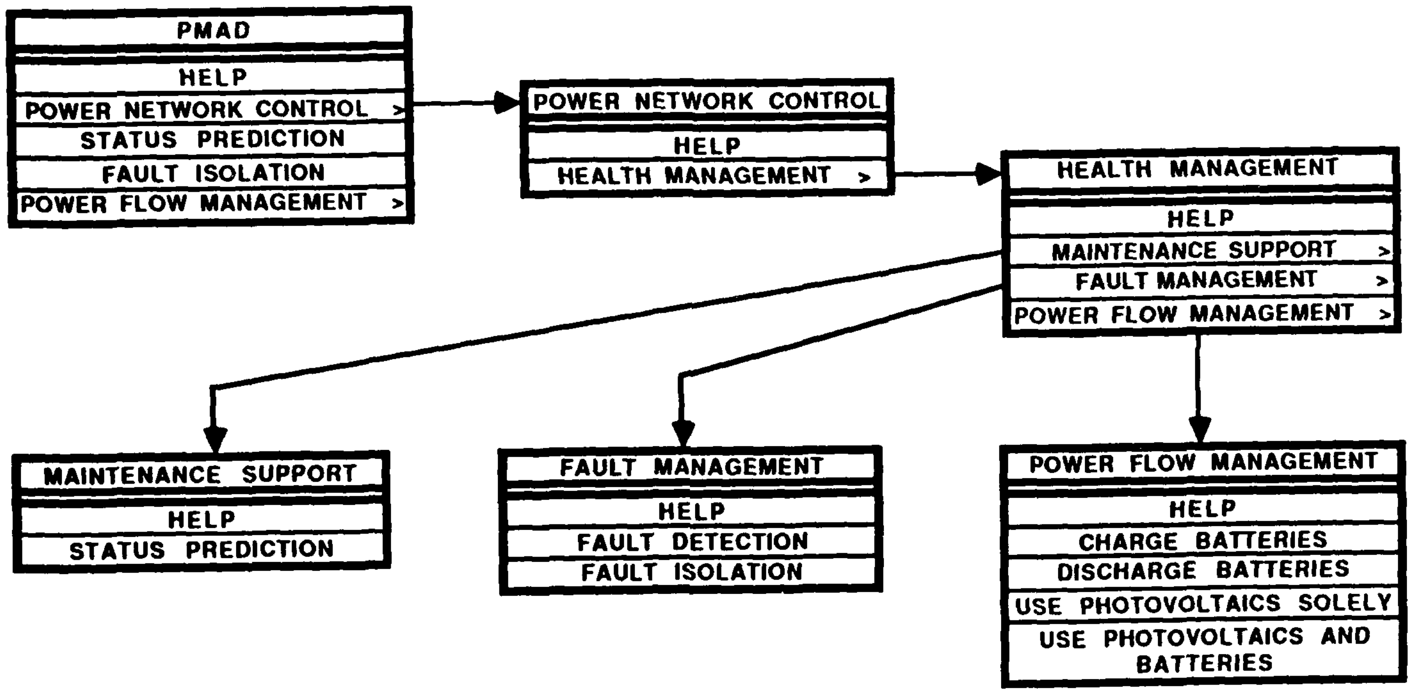


Figure 6. The PMAD System Main and Subordinate Menus

MEX allows the user to access the main menu help panel of the PMAD system, lower level menus and their help panels. It also provides access to the animation panel that depicts the flow of power through the system. The main menu enables the user to conduct system evaluations to determine whether or not predicted or actual faults exits. Similarly, the component menu enables the evaluation of predicted or actual single component failures.

4. The Silicon Graphics GL Package

The PMAD system utilizes the graphics and utility routines provided on the IRIS to support high- and low- level graphics. These routines support the use of C, FORTRAN and Pascal language routines. The IRIS graphics package supports, but is not limited to, the following routines: drawing, coordinate transformation, pattern and font, input and output, object creation and editing, curve and surface and shading. [Ref. 8:p. 2]

B. THE INTEGRATION OF THE LAYERS

All four of the discussed layers work together to provide the simplicity in the PMAD system graphics interface. The UNIX operating system, C programing language, MEX and the graphics package enable the depiction of the PMAD system in Figure 7. The user is able to have a color graphics interface and a pop-up menu facility in the same screen.

VI. THE COMMUNICATIONS INTERFACE

The PMAD system communications interface utilizes the Inter-computer Communication Package (which is comprised of LISP and C code) developed by students at the Naval Postgraduate School [Ref. 9]. This package utilizes the TCP/IP standard and the UNIX client/server socket stream capability. The high-level routines for this package have been tested on different machines to include the Symbolics and the IRIS. An IRIS can be either a server waiting for a client to call and establish a connection or the client. The Symbolics LISP machine must always be the client. This is because "the IRIS simulates the environment and the Symbolics simulates intelligence. The environment must exist before intelligence can be applied to it." This package enables the passing of integers, single floating point numbers, single characters and character strings from the IRIS to the Symbolics and vice versa. [Ref. 10:p. 1]

The following is a brief discussion of the IRIS and the Symbolics portions of this interface. The directory which contains all of the necessary code for the Inter-computer Communications Package can be viewed on the IRIS (IRIS2) at the Naval Postgraduate School Graphics and Video Laboratory. Additional details are available in [Ref. 9].

A. THE IRIS

In the utilization of TCP/IP and C, two ports are necessary to establish communication with each machine. These ports are connected to sockets in TCP/IP (which can be conceptually thought of as electrical sockets in a wall). Once the two channel link has been established, each channel is used in an asynchronous mode; i.e., enabling reading and writing of information as desired by both ends of the link. There are three different ways to establish the link with another machine. The simplest is using the function

machinepath to create a link between two machines. Since this is the only method used by the PMAD system it is the only one that will be discussed. [Ref. 10:p. 1]

In utilizing the **machinepath** function certain rules must be followed. Once the call to machinepath has been made, other dynamic allocation (such as **makeobject** calls to the graphics library) cannot be made and only one **machinepath** call can be made in a program. There are two independent processes, **receive** and **send**, that communicate with the PMAD system C application program using the **machinepath**. Each **receive** process sleeps after receiving a message for its socket until its buffer is emptied by the application program and each **send** process sleeps after sending a message to its socket until the application program requests that it send another message. This method reduces processing overhead. Once finished with communications, the links created by **machinepath** can be broken with the function **deletemachinepath**. This function deletes the links from memory, kills the **receive** and **send** processes and shuts down and closes the TCP/IP socket connections. [Ref. 10:pp. 1-2]

B. THE SYMBOLICS

Within the LISP artificial intelligence language exists the *Flavor System*. This system is a "mechanism for defining and creating active objects, that is objects which 'remember' their state and 'know' how to perform certain operations." A *flavor* is a class of such objects, while conversely, each object of this type is an instance of that flavor. Two primary characteristics of a flavor are the set of state variables that an instance of a flavor has (instance variables) and the set of operations that may be performed on all instances of that flavor. The operations that may be performed on these flavor instances are implemented by functions called *methods*. These methods provide behavior for instances of a flavor. [Ref. 11:pp. 97-99]

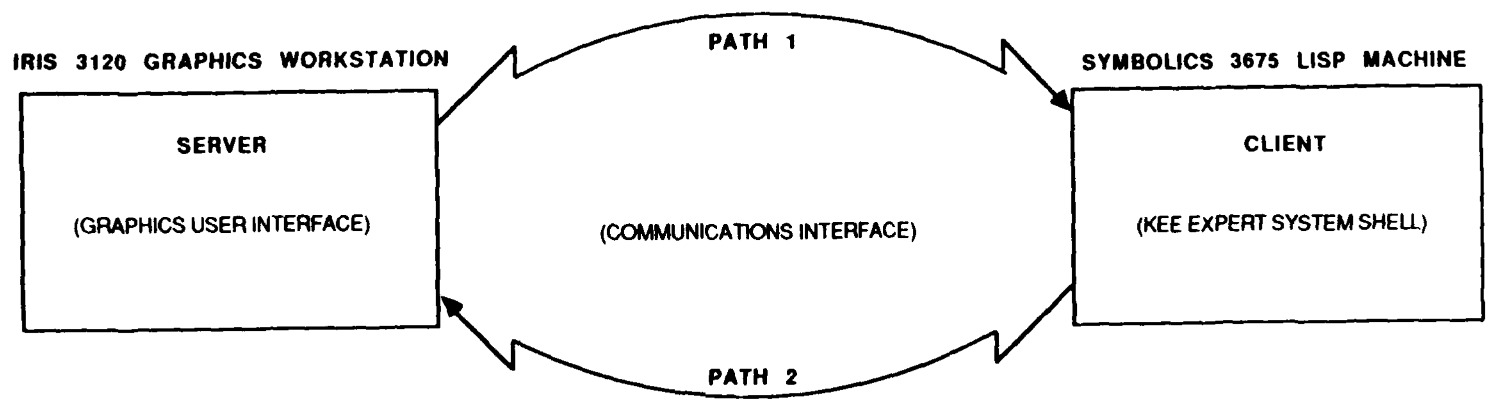
The important flavor in the Inter-computer Communication Package implemented on the Symbolics is called **conversation-with-iris**. There is a method called **put-iris** which converts an argument of any type to a string and sends it to the IRIS host. The method **get-iris** returns the proper type, depending on what was sent. [Ref. 9:p. 4]

VII. THE INTEGRATED PMAD SYSTEM PROTOTYPE

The expert system, graphics and communications interfaces come together smoothly to create the integrated PMAD system prototype. A conceptual picture of how these three portions of the PMAD system function together can be seen in Figure 8.

A. THE MAN-MACHINE INTERFACE

The man-machine interface is based on the concept that an astronaut on a space station will utilize a graphics interface with an expert system running in the background and a communications interface linking the two together. The astronaut will instruct the expert system through the graphics interface to run a system or component check on the PMAD system. The graphics interface determines which type of evaluation was requested and send that information through the communications interface to the expert system. The expert system receives the information and, based on the type of evaluation requested, executes the appropriate rules in the knowledge base and sends the resulting status back to the graphics interface and the astronaut. Based on the status of the component(s), the graphics interface either flashes the component(s) red if failed or yellow if predicted to fail for a system check [Ref. 12:p. 71]. For a single component check, the graphics interface flashes a component green if the component is functioning properly.


IRIS 3120 GRAPHICS WORKSTATION
SERVER
(GRAPHICS USER INTERFACE)
PATH 1
(COMMUNICATIONS INTERFACE)
PATH 2
SYMBOLICS 3675 LISP MACHINE
CLIENT
(KEE EXPERT SYSTEM SHELL)

PATH 1: SEND TYPE NUMBER AND COMPONENT NUMBER

PATH 2: SEND COMPONENT STATUS AND COMPONENT NUMBER

Figure 8. The Integrated PMAD System

B. FAULT DIAGNOSIS

The expert system for the PMAD system receives information from the graphics interface as stated in paragraph VII.A above. The expert system utilizes this information to perform fault diagnosis by accessing facts that have been inserted into the knowledge base (simulating sensors that gather information for the PMAD system) and by executing fault diagnosis rules written in the KEE expert system shell *Tell and Ask* language. Tell and Ask is a high level English-like language that enables the composition of if-then rules and the use of the forward and backward chaining (part of the KEE inference engine). A forward chaining algorithm searches through the knowledge base to find facts to satisfy the if-portion of a rule. If it can satisfy all the conditions of the if-portion, the then-portion is deduced to be true and is added as a fact to the knowledge base. A backward chaining algorithm starts with the then-portion of a rule, also called the goal, and searches back through the knowledge base to find the facts needed to prove the then-portion true. That means finding all the facts to satisfy the if-portion or finding the then-portion as a fact already in the knowledge base. The results are then sent back to the graphics interface via the communications interface. The battery structure will be utilized to illustrate this fault diagnosis concept.

The graphics interface, when queried by a user, will request a component check on **battery.1** (known to the graphics as #5). This component number and the type of check (status prediction for predicted failure (0) or fault detection for actual failure (1)) are sent via the communications interface to the expert system. In this case it will be a one (1) for an actual failure.

Figure 9a shows the facts for **battery.1** and its connections that were inserted into the PMAD system knowledge base prior to establishing communications with the IRIS. Figure 9b shows the external form of the Tell and Ask fault rule for **battery.1**. Figure 9c shows the Tell and Ask backward chaining rule for **battery.1**. This backward chaining rule, when executed, will use the asserted facts and the fault rule to determine if **battery.1** has failed. (Note that the backward chaining rule is written in a form that allows for a complete system check or a single component check, whichever is requested.) If the then-portion of the backward chaining rule is found to be true, then a slot in **battery.1** called **fault.mode** has its value changed to **failed**. This slot of **battery.1** is shown in Figure 10. This information is then sent back to the graphics interface in the form of a component number (5) and a message containing the appropriate color to flash the component and a status for the component ("R-The **fault.mode** of **battery.1** is **failed**," the 'R' in the message representing the color red).

```
(defun init-values1 ()
      (assert '(the value.state of battery.1.vr.n.powerload is negative()
      (assert '(the trend.state of battery.1.charge is steady))
)
```

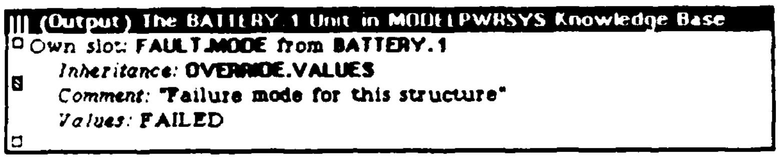
Figure 9a. BATTERY.1 Unit Facts

```
Own slot: EXTERNAL.FORM  from  DEAD_BATTERY.RULE
Inheritance: OVERRIDE.VALUES
Avunits: RULEPARSE  in  RULESYSTEM3,  RULE-COMPILER-AV  in  ACTIVEVALUES

Cardinality Max: 1
Comment: "The text of the rule in the form the user entered. The rule is parsed by the RULEPARSE active value.
             Parsed premises are placed in the PREMISE slot and conclusions are placed in the CONCLUSION
          slot."

Values: (IF (?PART IS IN CLASS BATTERY)
             (THE VALUES.STATE OF (THE POWERLOAD OF (THE VR.N OF ?PART)) IS NEGATIVE)
             (THE TREND.STATE OF (THE CHARGE OF ?PART) IS STEADY)
            THEN
            DEDUCE
             (THE FAULT.MODE OF ?PART IS FAILED))
```

Figure 9b. BATTERY.1 Unit Tell and Ask Rule

```
(defun start_diagnosis (user::*comp* rules world)
     (setq comp (aref *pwrsys_array* user::*comp*))
     (query `(a fault.mode of ,(if (null comp) '?comp comp is ?what) rules world)
)
```

Figure 9c. BATTERY.1 Unit Backward Chaining Rule



```
(Output) The BATTERY.1 Unit in MODELPWRSYS Knowledge Base
Own slot: FAULT.MODE from BATTERY.1
    Inheritance: OVERRIDE.VALUES
    Comment: "Failure mode for this structure"
    Values: FAILED
```

Figure 10. FAULT.MODE Slot for BATTERY.1 Unit

C. THE PMAD SYSTEM ANALYSIS

The PMAD system, being composed of an expert system, graphics interface and communicatins interface, makes it very complex. Both the Symbolics and the IRIS rely on mouse interfaces when dealing with the PMAD system. The use of so many different interfaces causes the PMAD system to be a system with a delicate balance. If any one of these interfaces fails, that delicate balance will be disrupted and the PMAD system prototype will not function.

The fault diagnosis rules for the component connections determine a failure or predicted failure by utilizing probability routines called **faultroutine** and **statusroutine** (both routines are listed in Appendix B) instead of actual Tell and Ask fault rules.

Having the IRIS with its MEX window manager pop-up menu facility, helps to make the PMAD system environment easier to navigate and more organized. The help menus at each level of the pop-up menus have a toggle feature that alternates between specific help for that menu and information on the use of the mouse. This on-line help enables a new user to more easily use the PMAD system. Therefore, although the PMAD system is complex and needs total coordination between its several parts, its setup will enable it users (astronaut trainees for the Space Station) to be more efficient at their main tasks by reducing collateral duties pertaining to systems monitoring.

D. THE PMAD SYSTEM MANUAL

This manual, attached as Appendix A, contains information on how to utilize the PMAD system. It also contains suggested ways to expand the existing system that will make it more useful and efficient.

VIII. CONCLUSIONS AND RECOMMENDATIONS

A. CONCLUSIONS

The PMAD system prototype proved to be viable. Although not totally completed and error-free, the PMAD system validated the concept of using graphics as a main interface while leaving the expert system transparent to the user. Although KEE was utilized in the design of this prototype, one other expert system shell, ART, was briefly considered. Although a useful tool and faster in processing than KEE, ART did not provide the necessary ease of use associated with KEE that is paramount in getting individuals quickly up to speed on an expert system shell. Not only was the PMAD system a successful prototype, it also enabled the expansion of knowledge in artificial intelligence, expert systems and their software support tools.

B. RECOMMENDATIONS

1. PMAD System Prototype Expansion

In order to stay on the cutting edge of technology, one must be willing to expand or improve. The following paragraphs contain ideas for possible expansion of the PMAD system.

a. On the Symbolics

Currently, there is only one fault rule written for the PMAD system and that is for the battery component. Fault rules need to be written for the other components and their connections. Also, as was stated in Chapter VII, the fault rules for the connections of the components in the PMAD system knowledge base can be written in the KEE Tell and Ask Language, instead of using probability functions. Finally, the PMAD system library and model icons can be modified to more closely resemble the components that they are representing.

b. On the IRIS

Given more detailed information on the PMAD system, the current high level diagram can be extended or PAN/ZOOM capabilities for individual components can be developed. Extended screens can be implemented utilizing an icon in the middle of each side of the screen or, as an option, off of the PMAD system main menu. PAN/ZOOM options could be added to the existing pop-up component menu. Finally, a training program could be a choice off the main or component menu that would allow users to practice diagnosing component and system faults based on provided scenarios. This program would be displayed in the lower left quadrant of the PMAD system graphics interface display.

2. NASA Utilization of PMAD System Prototype

The NASA SADP had the concept of developing an expert system with color graphics as a more clearly understandable side display. This prototype delivers a different perspective on the foreseen interface by having the color graphics as the controller with the expert system running transparently to the user. This prototype could enable the accelerated training of new personnel on artificial intelligence projects by giving them a prototype to learn on that is simple and in step with the current goals of NASA. It is a prototype idea that can be expanded, modified and completedly changed. This flexibility is the true beauty of the PMAD system prototype. It will provide the ability to stimulate ideas for its improvement and, as a result, sharpen the minds of those who will design the actual Power Control System.

3. Practical Application to the United States Navy

Today's Navy currently relies heavily on satellites for navigation. As a result, there are many naval personnel that stand long hours of watch at ground statins, monitoring the health and welfare of these satellites. If the idea of the PMAD system prototype was modified to accomodate satellites (such as monitoring, change of station and fault diagnosis) then many of the individuals already performing this task manually, as a full time job, could be utilized in a more efficient capacity. It will be costly to incorporate such a system into the existing framework but, it is foreseen that such systems as the PMAD system prototype will reduce the wear and tear on a most precious resource, people.

APPENDIX A

THE POWER MANAGEMENT AND DISTRIBUTION (PMAD) SYSTEM PROTOTYPE MANUAL

TABLE OF CONTENTS

LIST OF FIGURES

I. PMAD SYSTEM USER'S MANUAL

This manual is for those who will utilize the PMAD system. You should familiarize yourself with the IRIS 3120 graphics workstation, the Symbolics 3675 LISP machine and their associated software before trying to use the system. It may make the first few accesses to the PMAD system more pleasant.

A. GETTING STARTED

In order to get started on using the PMAD system, you must have an account on both the Symbolics and the IRIS and have access to the NASA directory on both machines. Once you have become oriented on both machines, the following files should be copied to your directory on the respective machine.

1. Necessary Symbolics Files

As was stated, access to the NASA directory on the Symbolics is needed. The necessary files will be set up for use. Also the **init.lisp** file that sets up your Symbolics environment should contain the following LISP command: **(load "syml:>sys>site>thermal.translations")** to enable you to access the PMAD system.

2. Necessary IRIS Files

Access to the NASA directory on the IRIS is needed. The necessary files will be available for use. Your **.login** file that sets up your IRIS environment should include the following line:

```
alias pmad source runprog
```

to enable you to access the PMAD system with fewer commands.

3. Logging On

a. The Symbolics

Make sure the system in booted up in Genera 7.1 with KEE 3.1. At the "Command:" prompt type "login" and a space. You will then be prompted for your user name. Type in "NASA" and a CR. The Symbolics should run the init.lisp file and load the **syml::sys>site>thermal.translations** file. Hit SELECT-K to bring up KEE. At the LISP Listener window type the LISP command (load file kt:build-system :mtk) to bring up MTK. Then click the left mouse button on the KEE icon, choose the load KB option and load the following knowledge bases: **syml:>NASA>pwrsys** and **syml:>NASA>modelpwrsys**. Once loaded, click the middle mouse button on the MODELPWRSYS knowledge base in the KB window. Position the resulting window, MODELPWRSYS UNITS, in a convenient location. Click the left mouse button on MODELPWRSYS.GLOBALS and choose the send message option. On this window choose the initialize parameters option. Now, at the LISP Listener window type the command **(SET PACKAGE COMMON-LISP-USER)**. Then type the command **(load "main-pwrsys.lisp")**. After this file is loaded you are ready to proceed to section I.A.3.b.

b. The IRIS

You will need both a side terminal and the IRIS graphics workstation. Since the communications package sends status messages to the screen, the side terminal must be used as the logon machine. So, at the "IRIS Login" prompt, input your account name and a carriage return (CR). At the password prompt, input your password and a CR. You will then be prompted for a terminal type. Type **vt100** and a CR. Change to the NASA directory by typing **cd /user/work/NASA** and a CR.

c. The Integrated PMAD System

To run the PMAD system program, type "PMAD" at the IRIS system prompt. The window manager, MEX, will be invoked on the IRIS workstation, making its current screen CONSOLE and then the graphics interface of the PMAD system will be displayed on the IRIS. Two messages should come up on the side terminal saying "Awaiting connection with SYM1". At this time, go over to the Symbolics (SYM1) and type the LISP command **(sympwrsys)** at the LISP Listener. This command starts the Symbolics portion of the PMAD system running. Now the system is ready to use.

4. Use of the Mouse

The mouse has three buttons with functions as described in Figure A1.

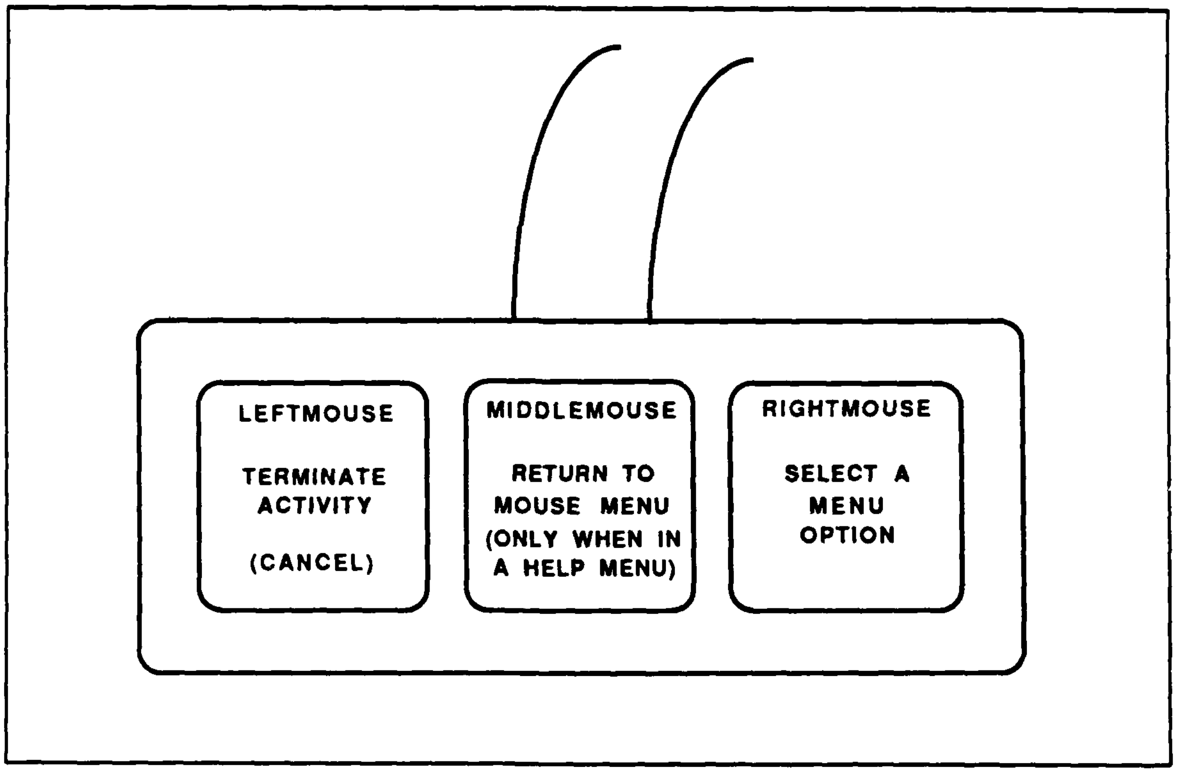


Figure A1. PMAD System Mouse Help Menu

5. Select an Option

To select an option, press the right mouse button to bring up the desired menu panel. Options with an arrow in the right hand corner have submenus that can be displayed by scrolling off either end of that particular menu. Once you have reached an option you desire, release the mouse button. Additionally, once done with a selected option and the display, press LEFT MOUSE (CANCEL) to go back to the main menu level.

6. The Main Menu

The main menu panel of the PMAD system can always be reached by: (1) pressing the RIGHT MOUSE or, if several levels down, (2) rolling the mouse off of all options letting go of right mouse and then pressing it again as in option 1.

7. Terminate Activity (CANCEL)

This option will terminate fault simulations, power flows and help menus. It returns you to the main menu level which can be reached as explained in Section I.A.6.

8. Help Menus

To choose a particular help menu, press the RIGHT MOUSE, move cursor to the HELP option and release the mouse button. The help menu has a toggle feature which lets you toggle between the mouse diagram above and specific help for that menu. To leave a help menu, press the cancel key, MIDDLE MOUSE.

9. Exiting the PMAD System

To exit the program, press the RIGHT MOUSE, place cursor on EXIT option and release. The connections between the IRIS and the Symbolics will automatically be closed.

B. PMAD SYSTEM OVERVIEW

This section provides an overview of the PMAD system. This includes all menus and their options, along with a brief description of each.

1. PMAD System Main Menu

The PMAD system Main Menu represents a system based on a photovoltaic power supply with battery back-up. It would normally include options for HELP, POWER CONDITIONING, POWER DISTRIBUTION and POWER NETWORK CONTROL. For simplicity, only the HELP and POWER NETWORK CONTROL options were implemented. The STATUS PREDICTION and FAULT ISOLATION and POWER FLOW MANAGEMENT options exists are lower level menus and were added to the main menu for ease of access. POWER NETWORK CONTROL and POWER FLOW MANAGEMENT have submenus. Flow of power through the system can be seen by clicking menu option POWER FLOW MANAGEMENT and scrolling off of its left or right side. STATUS PREDICTION and FAULT ISOLATION enable the running of a complete system check to determine whether a predicted or an actual failure has occurred somewhere in the PMAD system. A diagram of the PMAD System Main Menu can be seen in Figure A2.

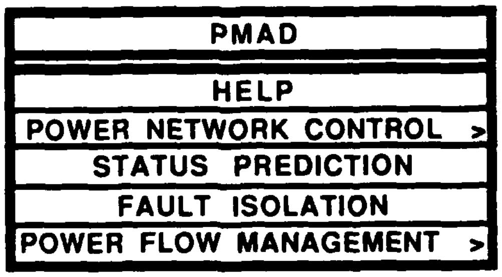


Figure A2. PMAD System Main Menu

2. Power Network Control Menu

The Power Network Control menu is designed to break down system analysis into smaller subtasks which can be managed independently and more efficiently. It would normally include options for HELP, DISTRIBUTION MANAGEMENT, LOAD MANAGEMENT, HEALTH MANAGEMENT and COMMAND/ DATA I/F. Again, only the HELP and HEALTH MANAGEMENT options were implemented. HEALTH MANAGEMENT has submenus. This menu can be seen in Figure A3.

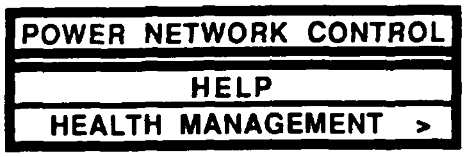


Figure A3. Power Network Control Menu

3. Health Management Menu

The Health Management menu's function is to monitor the health of the PMAD system and to predict its future status to enable parts' replacement before failure. The options are HELP, MAINTENANCE SUPPORT, FAULT MANAGEMENT and POWER FLOW MANAGEMENT. POWER FLOW MANAGEMENT is not one of the original options but was added to provide the user animation that shows the different ways power may flow through the PMAD system. All options except HELP have submenus. This menu can be seen in Figure A4.

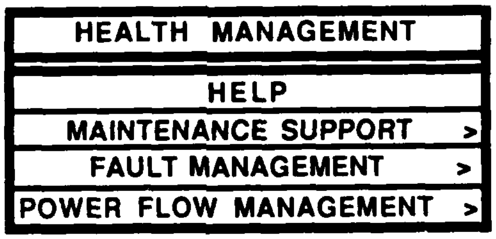


Figure A4. Health Management Menu

4. Maintenance Support Menu

The Maintenance Support menu provides information needed to carry out service procedures, both unscheduled and routine, and step-by-step instructions for these procedures to include contingency information to handle foreseeable problems. The options would normally be HELP, STATUS PREDICTION, PREVIOUS MAINTENANCE SCHEDULING, NETWORK SOLUTION, MONITORING and HISTORY RECORDS GENERATION. The only options implemented were HELP and STATUS PREDICTION. This menu can be seen in Figure A5.

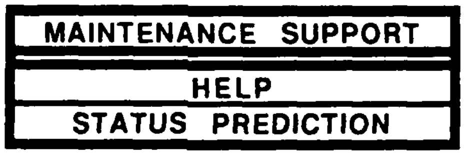


Figure A5. Maintenance Support Menu

5. Fault Management Menu

The Fault Management menu provides options that enable the detection of an abnormal state in the PMAD system, the isolation of faults that cause this state and suggested actions that could bring the system back to an operational state. The options that would normally be present are HELP, FAULT DETECTION, FAULT ISOLATION, FAULT COMPENSATION and FAULT LOGGING. The only implemented options are HELP, FAULT DETECTION and FAULT ISOLATION, where FAULT DETECTION and FAULT ISOLATION currently call the same detection routine. This menu can be seen in Figure A6.

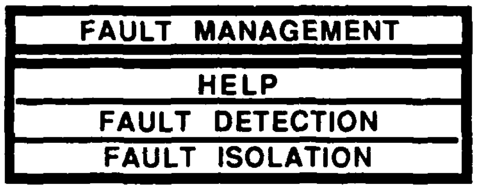


Figure A6. Fault Managment Menu

6. Power Flow Management

The Power Flow Management menu's purpose is to show the flow of power through the system. The options are HELP, CHARGE BATTERIES, DISCHARGE BATTERIES, USE PHOTOVOLTAIC SOLELY, USE PHOTOVOLTAIC AND BATTERIES. This menu can be seen in Figure A7.

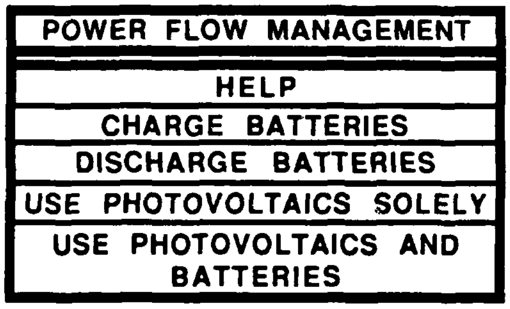


Figure A7. Power Flow Management Menu

7. The Component Menu

The Component Menu provides options HELP, FAULT ISOLATION and STATUS PREDICTION. These options when chosen from the Component menu enable the checking of only a single component vice a complete system's check. This menu can be seen in Figure A8.

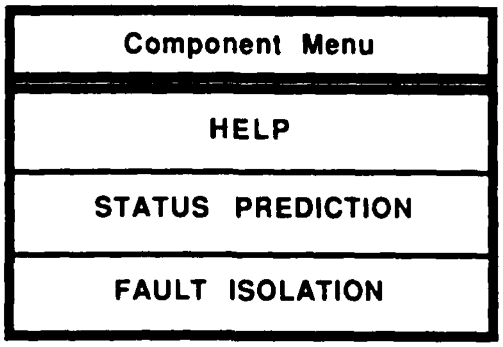


Figure A8. PMAD System Component Menu

C. ON-LINE HELP

This manual, in its entirety, will be available in a file called **README**. It can be viewed by typing the command **pmadman**.

II. PMAD SYSTEM PROGRAMMER'S MANUAL

This manual is for those who would like to modify the PMAD system. This manual discusses applications software on the Symbolics and the IRIS and possible ideas for PMAD system expansion. Before attempting modifications, it is recommended that the following courses be taken before utilizing this system:

CS3313 - Introduction to Artificial Intelligence,
CS4202 - Introduction to Computer Graphics, and
CS4313 - Computers for Artificial Intelligence.

A. SYMBOLICS APPLICATION SOFTWARE ORGANIZATION

The application software developed on the Symbolics is consolidated into four LISP files: **init-pwrsys.lisp**, **keefiles.lisp**, **pwrsys-net.lisp** and **main-pwrsys.lisp**. The subroutines that these files consist of will be discussed in this section. Familiarization with packages and how they interrelate on the Symbolics will be quite helpful, especially the packages COMMON-LISP-USER and KEE, since the LISP command **(zl:pkg-goto [package name])** is used throughout the LISP files in order to transfer control between these two packages.

1. Init-pwrsys.lisp

This file contains functions that are used to set up the initial PMAD system environment under the package COMMON-LISP-USER, before package KEE functions are utilized. This file consists of the following variable declarations (all variables with asterisks (*) around them are global, known to all the files).

typenum - This variable holds the type of system check requested (zero (0) for fault detection and one (1) for status prediction).

message_out - This variable holds the message that is sent from the Symbolics to the IRIS containing the status of the component that was checked.

user::*comp* - This variable holds the component number sent from the IRIS to the USER package on the Symbolics.

randnum - This variable holds the random number generated to determine if a connections of component has failed.

user::*fault_list* - This variable holds the list of failed components based on backward chaining through the facts and rules of the PMAD system knowledge base.

part - This variable holds the component from the pwrsys_array that corresponds to the component number sent from the IRIS.

data_in - This variable holds information sent from the IRIS to the Symbolics.

data_out - This variable holds information sent from the Symbolics to the IRIS.

***finished_processing** - This variable holds the flag sent from the Symbolics to the IRIS to signal the end of processing for that particular system or component check.

The following functions are used:

randroutine - This routine generates the random number to be used by the **faultroutine** and **predictedroutine.**

faultroutine - This routine determines, based on whether the random number from **randroutine** was between 0.9 and 1.0, if the component connection in question (***comp***) has failed.

predictedroutine - This routine determines, based on whether the random number from randroutine was between 0.8 and 1.0, if the component connection in question (*comp*) is predicted to fail.

process_data - This routine determines whether the part in question is a component or a component connection (part number 0 or 15 through 30). If the part is a connection, it runs the appropriate probability function, depending on the type of check desired.

2. Keefiles.lisp

This file contains those functions that need to run on the Symbolics under package KEE. They are as follows:

start_diagnosis - This routine sets a variable equal to a component or connection from the PMAD system component and connection array.

pwrsys_array - That variable is used in the backward chaining rule that determines fault occurrences.

init-values1 - This routine inserts facts concerning the **battery.1** component into the PMAD system knowledge base.

reset-values1 - This routine deletes facts concerning the **battery.1** component from the PMAD system knowledge base.

pwrsys_array - An array of the components and connections of the PMAD system knowledge base.

predicted_array - This is an array of messages for the predicted failure of PMAD system components and connections, using the names that correspond to those on the graphics interface side of the PMAD system.

fault_array - This is an array of messages for the failure of PMAD system components and connections, using the names that correspond to those on the graphics interface side of the PMAD system.

okay_array - This array is only accessed when a single component check results in a correctly functioning component.

check_predicted_array - This routine determines which predicted failure message to send back to the IRIS.

check_fault_array - This routine determines which failed message to send back to the IRIS.

output_routine - This routine takes a line in the fault list that is generated from the results of backward chaining through the PMAD system knowledge base and, based on whether it was an actual or predicted failure, runs the **check_predicted_array** or **check_fault_array** routine and sends the appropriate message to the IRIS.

process_list - This routine cycles through the fault list, line by line, in order to provide the IRIS with information on PMAD system components and connections.

3. Pwrsys-net.lisp

This file contains the functions, flavors and methods that run under package COMMON-LISP-USER and that enable the Symbolics to communicate with the IRIS. This code was developed by Sehung Kwak and Captain Andy Nelson. Please see Major Ted Barrow's M.S. Thesis dated June 1988 for detailed descriptions of the functions in this file. The following are two functions that were set up to provide ease in utilization of this file.

receive_data - Method for the Symbolics to receive information (integers, single floating-point numbers, single characters and characters strings) from the IRIS.

send_data - Method for the Symbolics to send information to the IRIS.

4. Main-pwrsys.lisp

This file controls the Symbolics portion of the PMAD system. It loads the previously mentioned files, connects the Symbolics to the IRIS by executing the

start_talking function which resides in **pwrsys-net.lisp** and contains the functions that have primary control of the PMAD system on the Symbolics, **sympwrsys**.

Sympwrsys initializes the variables that signal the end of a system or component check (***finished_processing***) or the end of communications between the two machines (***done***). It changes the current packages to KEE and inserts the facts concerning **battery.1** into the PMAD system knowledge base. It then returns the current package to COMMON-LISP-USER.

The large **do** loop will loop until the ***done*** variable is set to true. Diagnostic print messages are still in the code due to the debugging that was in process at the time of the printing of this manual. Data is sent from the IRIS to the Symbolics in the form of the type of check desired (predicted or actual failure or end of communications flag) and a component number (corresponding to a component, connection or complete system check).

When the end of communication flag (999) is received, the **stop-talking** function is executed, the ***done*** variable is set to true and communication is terminated. Otherwise, the type of check and component number are saved in variables ***typenum*** and **user::*comp***, respectively. **Sympwrsys** then calls the **process-data** function. The variable ***part*** is set to the input component number. If ***part*** corresponds to the **kee::*pwrsys_array** entry 'nil' (for system check) then run the block LISP code (**progn**) to process the fault list (if any) returned from the **process_data** function by executing the **kee::process_list** function. Otherwise, run the **progn** that processes the single message in the fault list using the **kee::process_list** function.

Both **progn** sections send the component number and a status message back to the IRIS. Before connection is broken, the facts concerning **battery.1** are retracted from the PMAD system knowledge base using the reset-values1 command and the current package is changed back to COMMON-LISP-USER.

B. IRIS APPLICATION SOFTWARE ORGANIZATION

This section will provide more of an overview of IRIS application software since in-line documentation is provided. Also, when possible, files of similar subject will be grouped together.

Makefile - A file that enables an organized method for compiling files.

README - An on-line version of the entire PMAD system manual.

Header files - **Shared.h** contains information for the files that utilize communications software. **Pwrsys.h** contains numerical definitions for the PMAD system parts, a structure definition for these parts and other useful declarations.

Help files - **Help.c** contains the main control module for all the help menus called **help_menu**. It toggles between specific help and the mouse help. It also contains the **processhelp** routine that enables the creation and display of specific help for the current menu. **faulthelp.c, healthhelp.c, mainhelp.c, maintspthelp.c, mousemenhelp.c, nethelp.c** and **pwrflowhelp.c** contain the help menu text for their respective menus.

Executable files - **Nasapwrsys** brings up the PMAD system with communications capability when executed. **Nonetpwrsys** brings up the PMAD system without the communications. **Runprog** is a command file that invokes the MEX window manager and that executes the nasapwrsys file.

Newpwrsys.c - This is the primary control module for the entire graphics interface. It is mouse-oriented and utilizes the IRIS window manager, MEX.

Component files - The **componenthit.c** file contains routines that determine whether the current mouse location corresponds to a component hit. The **compmenu.c** file contains the routine that displays the menu that provides the choice of items that are predicted to fail or that have actually failed. This menu is only seen when the **nonetpwrsys** executable file is run.

Diagram.c - This file contains the routine that creates the different objects in the PMAD system diagram, **make_diagram**. It also contains the routines that change all objects in the PMAD system diagram back to their original color after processing, **cleanup_flow** and **cleanup_diagram**, and that call all the objects in the PMAD system for screen display, **call_diagram**.

Compstat.c - This file contains routines **faultisolation** and **statusprediction** that check for actual and predicted faults in the system, respectively. The **componentstatus** routine checks the status of individual components in the diagram. Routine **process_message** determines whether to flash a component red, yellow or green and displays the status message, based on messages received from the Symbolics.

Pwrflow.c - This file contains routines that show the flow of power from the photovoltaic equipment to the battery (**charge_battery**), the flow of power from the batteries through the rest of the system (**discharge_battery**), the flow of power from the photovoltaic equipment throughout the rest of the system (**use_pv_only**) and the flow of power from the photovoltaic equipment and the batteries throughout the rest of the system (**use_pv_and_batteries**).

Object files - All of the ".c" files discussed previously have a ".o" file associated with them. These ".o" files are the compiled versions of their ".c" source code.

C. PMAD SYSTEM EXPANSION

Now that the applications software for each machine has been discussed, ideas for PMAD system expansion will be presented. This presentation not only includes these ideas, but also methods for their implementation.

1. PMAD Expert System Expansion

Currently there is only one fault rule written for the PMAD system knowledge base and that is for the **battery.1** component. Fault rules need to be written for the other components and their connections. Before modifying the PMAD system, you should understand the KEE *Tell and Ask* language and how to write the **external.form** of a fault rule. Also, familiarity with the ZMACS editor on the Symbolics would be helpful. The following paragraphs will explain how to add a fault rule to the PMAD system knowledge base.

If your KEE environment does not have an output window displayed, choose the desktop option in the top left-hand corner. Choose the create output window option and follow the instructions provided by KEE.

Using the mouse, place the cursor on the PWRSYS knowledge base inside the KB window and click the left mouse button. A PWRSYS menu will pop-up. Choose the display option by placing the cursor there using the mouse and clicking the left mouse button. The output window will display the contents of the PMAD system knowledge base.

Place the cursor on the FAULT.RULES class at the top of the PMAD system knowledge base and click the left mouse button. Choose the "create new unit" option off of the pop-up menu and follow the instructions that are provided. When asked to input the **external.form** of the rule, realize that if an input error is made it can be corrected by backspacing or by saving what is already typed (with proper LISP syntax) and editing it using the ZMACS editor.

2. PMAD System IRIS Graphics Extended Screen and Pan/Zoom Capabilities

Given more detailed information on the PMAD system, a programmer can either extend the current high-level diagram or develop Pan/Zoom capabilities for the individual components. Extended Screens of the current high-level diagram can be done utilizing icons in the middle of the four corners of the screen (which can be created as objects) or as an option off of the PMAD system main pop-up menu. Depending on where the cursor is positioned, a corresponding extended screen will appear.

Pan/Zoom options can be added to the existing component pop-up menu. When chosen, that option could note the component in question and, based on that information, bring up a detailed schematic of that component. In either case, extended screen or pan/zoom, the following files should be considered for modification:

Diagram.c will need objects added to represent the new portion of the screen or the detailed schematic of the component.

Pwrsys.h will need "#define" statements that represent these additions.

Newpwrsys.c will need to call the routine that will perform these new tasks.

Compmenu.c and **componenthit.c** will need to reflect the additions of new components.

3. PMAD System Training Program

A training program could be written for the PMAD system. It would be a choice off of the main or component pop-up menu of the graphics interface. That would allow practicing of diagnosing faults that are chosen from a data base of system fault scenarios. This system can be displayed in the lower left quadrant of the IRIS screen. It could be an independent program that is pulled into the PMAD system through the pop-up menus. It would need to be incorporated into the graphics using the method already used by the PMAD system help routines, while still displaying the portion of the system desired for viewing.

4. PMAD System Help Files

The help files for the PMAD system on the IRIS can be expanded to contain detailed information. The following suggested method for expansion will use **mainhelp.c** for the example.

In the **static str80**, array add the desired number of quoted lines of information, not to exceed 80 characters per line. Change the NUMHELP1 definition at the top of the file (from three (3)) to reflect that desired number.

III. USEFUL REFERENCE DOCUMENTS AND MANUALS

The following list is provided to assist those who will be working with the PMAD system prototype:

Barrow, T., *Distributed Computer Communications in Support of Real- Time Visual Simulations*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

Bromley, H., and Lamson, R., *LISP Lore: A Guide to Programming the LISP Machine*, Kluwer Academic Publishers, 1987.

Erickson, W., and Nienart, J., *MTK Reference Manual and User Guide (draft)*, 10 April 1988.

Erickson, W., and others, "NASA Systems Autonomy Demonstration Program: A Step Toward Space Station Automation," *SPIE Space Station Automation II*, v. 729, 1986.

IntelliCorp KEE Software Development System User's Manual, IntelliCorp, 1986.

IRIS User's Guide: Volume 1 Graphics Programming, Silicon Graphics, Inc., 1986.

Kernighan, B., and Ritchie, D., *The C Programming Language*, Prentice-Hall, Inc., 1978.

Martin Marietta Denver Aerospace Task I Study Report MCR-86-583, *Space Station Automation of Common Module Power Management and Distribution*, by Miller, W., and others, July 1986.

User's Guide to Symbolics Computers, Symbolics, Inc., July 1986.

Wilensky, R., *Common LISPcraft*, W. W. Norton & Company, Inc., 1986.

IV. IMPORTANT POINTS OF CONTACT

In case the references listed in the previous section do not answer specific questions, the following points of contact are provided:

Dr. Henry Lum
MS244-7
NASA Ames Research Center
Moffett Field, CA 94035
AUTOVON 359-6544
COMMERCIAL 1-415-694-6544

Ms. Carla Wong
MS244-18
NASA Ames Research Center
Moffett Field, CA 94035
COMMERCIAL 1-415-694-4294

Mr. William Erickson
MS244-18
NASA Ames Research Center
Moffett Field, CA 94035
AUTOVON 359-3369
COMMERCIAL 1-415-694-3369

Dr. Robert McGhee
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5100
AUTOVON 878-2449
COMMERCIAL 1-408-646-2449

Dr. Michael Zyda
Department of Computer Science
Naval Postgraduate School
Monterey, CA 93943-5000
AUTOVON 878-2449
COMMERCIAL 1-408-646-2449

Lois Brunner
Joint Command, Control and Communications Academic Group
Naval Postgraduate School
Monterey, CA 93943-5000
AUTOVON 878-2618
COMMERCIAL 1-408-646-2618

LT Gina L. Hester
Fleet Surveillance Support Command
Chesapeake, VA 23322-5010
COMMERCIAL 1-804-421-8203

APPENDIX B

PMAD SYSTEM SOURCE LISTING

MAIN-PWRSYS.LISP

```
(load "init-pwrsys.lisp")

(load "pwrsys-net.lisp")

(load "keefiles.lisp")

(user::start-talking)

(defun sympwrsys ()

    (setq *done* 'nil)
    (setq *finished_processing* 555)

    (zl:pkg-goto 'kee)

    (kee::init-values1)

    (zl:pkg-goto 'common-lisp-user)

    (do ()
      (*done*)
      (progn (princ "Awaiting data from IRIS2.")
           (setq *data_in* (user::receive_data))

           (if (= *data_in* 999)
             (progn (user::stop-talking)
                  (setq *done* 't)
             )
             (progn (setq *typenum* *data_in*)
                  (print *data_in*)
                  (setq *data_in* (user::receive_data))
                  (print *data_in*)
                  (setq user::*comp* *data_in*)
                  (process_data *typenum* *data_in* user::*comp*)
```

```
                (print kee::*fault_list*)

                (setq *part* (aref kee::*pwrsys_array* *data_in*))

                (if (eq *part* 'nil)
                  (progn (if (car kee::*fault_list*)
                           (process_list kee::*fault_list*)


                           (progn (zl:pkg-goto 'common-lisp-user)

                             (user::send_data "NO SYSTEM PROBLEMS")
                             (user::send_data *finished_processing*)
                           )
                         )
                  )
                )
                (if (car kee::*fault_list*)
                  (kee::process_list kee::*fault_list*)
                  (progn (setq *data_out* *data_in*)
                         (print *data_out*)
                         (setq *message_out* (aref kee::*okay_array* *data_out*))
                         (princ *message_out*)

                         (zl:pkg-goto 'common-lisp-user)

                         (user::send_data *message_out*)
                         (user::send_data *data_out*)
                         (user::send_data "END OF COMPONENT CHECK")
                         (user::send_data *finished_processing*)
                  )
                )
              )
            )
          )
        )

        (kee::reset-values1)

        (zl:pkg-goto 'common-lisp-user)

)
```

INIT-PWRSYS.LISP

```
(load "sym1:>sys>site>thermal.translations")


(defvar *done* nil)
(defvar *typenum* nil)
(defvar *message_out* nil)
(defvar user::*comp* nil)
(defvar *randnum* nil)
(defvar user::*fault_list* nil)
(defvar *part* nil)

(defvar *data_in* nil)
(defvar *data_out* nil)

(defvar *finished_processing* nil)

(defun randroutine ()
      (setq user::num (random 1001))
      (setq *randnum* (/ user::num 1000))
)


(defun faultroutine (user::*comp* rand)
      (if (> rand 0.8)
         (progn (zl:pkg-goto 'kee)
              (assert `(a fault.mode of ,(aref *pwrsys_array* user::*comp*) is failed))
         )
      )
)


(defun predictedroutine (user::*comp* rand)
      (if (> rand 0.9)
         (progn (zl:pkg-goto 'kee)
              (assert `(a fault.mode of ,(aref *pwrsys_array* user::*comp*)
                     is predicted_failure))
         )
      )
)
```

```
(defun process_data (type partnum user::*comp*)
      (cond ((or (= partnum 0) (and (>= partnum 15)
                                   (<= partnum 50)))
            (randroutine)
            (if (= type 0)
                  (faultroutine user::*comp* *randnum*)
                  (predictedroutine user::*comp* *randnum*)
            )
         )
      )


      (setq user::*fault_list*
            (kee::start_diagnosis user::*comp* 'kee::fault.rules 'kee::basicworld))
)
```

PWRSYS-NET.LISP

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: USER -*-

; handy macro to have in the send message farthur down


(defmacro loopfor (var init test exp1 &optional exp2 exp3 exp4 exp5)
 `(prog ()
      (setq ,var ,init)
      tag
        ,exp1
        ,exp2
        ,exp3
        ,exp4
        ,exp5
        (setq ,var (1+ ,var))
        (if (= ,var ,test) (return t) (go tag))))


(defun convert-number-to-string (n)
  (princ-to-string n))

(defun convert-string-to-integer (str &optional (radix 10))
  (do ((j 0 (+ j 1))
     (n 0 (+ (* n radix) (digit-char-p (char str j) radix))))
    ((= j (length str)) n)))

(defun find-period-index (str)
  (catch 'exit
   (dotimes (x (length str) nil)
    (if (equal (char str x) (char "." 0))
       (throw 'exit x)))))

(defun get-leftside-of-real (str &optional (radix 10))
  (do  ((j 0 (1+ j))
      (n 0 (+ (* n radix) (digit-char-p (char str j) radix))))
     ((or (null (digit-char-p (char str j) radix)) (= j (length str)))  n)))

(defun get-rightside-of-real (str &optional (radix 10))
  (do ((index (1+ (find-period-index str)) (1+ index))
     (factor 0.10 (* factor 0.10))
     (n 0.0 (+ n (* factor (digit-char-p (char str index) radix)))))
    ((= index (length str)) n )))
```

```
(defun convert-string-to-real (str &optional (radix 10))
  (+ (float (get-leftside-of-real str radix)) (get-rightside-of-real str radix)))


(defvar *iris-port1* 1027)                ; this is the send port
(defvar *iris-port2* 1026)                ; this is the receive port
(defvar *local-talk-port* 1500)           ; this is the local send port
(defvar *local-listen-port* 1501)         ; this is the local receive port


(defflavor conversation-with-iris ((talking-port-number     *iris-port1*)
                                   (listening-port-number   *iris-port2*)
                                   (local-talk-port-number  *local-talk-port*)
                                   (local-listen-port-number *local-listen-port*)
                                   (talking-stream)
                                   (listening-stream)
                                   (destination-host-object)
                                   )
                                   ()
                                   :initable-instance-variables)


(defmethod (:init-destination-host conversation-with-iris)
        (name-of-host)
  (setf destination-host-object (net:parse-host name-of-host)))



(defmethod (:start-iris conversation-with-iris) ()
  (setf talking-stream
      (tcp:open-tcp-stream destination-host-object
                           talking-port-number
                           local-talk-port-number))
  (setf listening-stream
      (tcp:open-tcp-stream destination-host-object
                           listening-port-number
                           local-listen-port-number))
  "A conversation with the iris machine has been established")



(defmethod (:reuse-iris conversation-with-iris )
        ()
  )

; (setq *tcp-handler1* (send ip::*tcp-handler* :get-port)
;     *tcp-handler2* (send ip::*tcp-handler* :get-port)
;     talking-port *tcp-handler1*
;     listening-port *tcp-handler2*))
```

```
(defun read-string (stream num-chars)
 (let ((out-string ""))
  (dotimes (i num-chars)
   (setf out-string (string-append out-string (read-char stream))))
  out-string))


(defmethod (:get-iris conversation-with-iris) ()
 (let* ((typebuffer  " ")
        (lengthbuffer "    ")
        (buffer      " ")
        (buffer-length 1))
   (progn
    (setf typebuffer
        (read-string listening-stream 1))
    (setf lengthbuffer
        (read-string listening-stream 4))
    (setf buffer-length
        (convert-string-to-integer lengthbuffer))
    (setf buffer
        (read-string listening-stream buffer-length))


     (cond ((equal typebuffer "I") (convert-string-to-integer buffer))
          ((equal typebuffer "R") (convert-string-to-real   buffer))
          ((equal typebuffer "C") buffer)
          (t nil)))))


(defvar *step-var* 0)



(defun my-write-string(string stream)
 (let* ((num-chars (length string)))
  (dotimes (i num-chars)
   (write-char (aref string i) stream))))


(defmethod (:put-iris conversation-with-iris)
        (object)

 (let* ((buffer (cond
               ((equal (type-of object) 'bignum) (convert-number-to-string object))
               ((equal (type-of object) 'fixnum) (convert-number-to-string object))
               ((equal (type-of object) 'single-float) (convert-number-to-string object))
               ((equal (type-of object) 'string) object)
```

```
             (t "error")))

        (buffer-length  (length buffer))

        (typebuffer    (cond ((equal (type-of object) 'bignum) "I")
                             ((equal (type-of object) 'fixnum) "I")
                             ((equal (type-of object) 'single-float) "R")
                             ((equal (type-of object) 'string) "C")
                             (t "C")))

        (lengthbuffer   (convert-number-to-string buffer-length)))


     (progn
      (my-write-string typebuffer talking-stream)
      (send talking-stream :force-output)


       (if (= (length lengthbuffer) 4)
          (write-string lengthbuffer talking-stream)
          (progn
            (loopfor *step-var* (length lengthbuffer) 4
                   (write-string "0" talking-stream))

            (my-write-string lengthbuffer talking-stream)
            ))

       (send talking-stream :force-output)


       (my-write-string buffer talking-stream)
       (send talking-stream :force-output)

       )))




(defmethod (:stop-iris conversation-with-iris)
       ()
  (progn (send talking-stream :close)
       (send listening-stream :close)))


(setq *comm_handler* (make-instance 'conversation-with-iris))
(send *comm_handler* :init-destination-host 'iris2)

(defun start-talking ()
 (send *comm_handler* :start-iris))
```

```
(defun stop-talking ()
 (send *comm_handler* :stop-iris)
 (send *comm_handler* :reuse-iris))

(defun send_data (x)
 (send *comm_handler* :put-iris x))

(defun receive_data ()
 (send *comm_handler* :get-iris))

;(setq myvar (receive_data))

;(send_data "1" )    ; 1   1.0 "1 1 1"
;(send_data 1 )    ; 1   1.0 "1 1 1"
;(send_data 1.0 )    ; 1   1.0 "1 1 1"
```

KEEFILES.LISP

```
;;; -*- Mode: LISP; Syntax: Common-lisp; Package: KEE -*-


(defun start_diagnosis (user::*comp* rules world)

      (setq comp (aref *pwrsys_array* user::*comp*))

      (query `(a fault.mode of ,(if (null comp) '?comp
               comp) is ?what) rules world)
)

(defun init-values1 ()
     (assert '(the value.state of battery.1.vr.n.powerload is negative))
     (assert '(the trend.state of battery.1.charge is steady))
)


(defun reset-values1 ()
     (initialize.keeworlds)
     (retract '(the value.state of battery.1.vr.n.powerload is negative))
     (retract '(the trend.state of battery.1.charge is steady))
)



(defvar *pwrsys_array* (make-array `(52) :initial-contents
                         `(epsboard.1
                          pvequip.1
                          betajoint.1
                          voltreg.1
                          battery.1
                          inverter.1
                          alphajoint.1
                          maindistboard.1
                          epspdca.1
                          basicpdca.1
                          seboard.1
                          boardswitch.1
                          basicswitch.1
                          basicswitch.2
                          basicswitch.3
                          epsboard.1.pvequipnode
```

```
                              epsboard.1.betajointnode
                              epsboard.1.inverternode
                              epsboard.1.maindistboardnode
                              epsboard.1.epspdcanode
                              pvequip.1.e.n
                              pvequip.1.b.n
                              betajoint.1.i
                              betajoint.1.e.n
                              betajoint.1.o
                              voltreg.1.i
                              voltreg.1.bt.n
                              voltreg.1.o
                              battery.1.vr.n
                              inverter.1.i
                              inverter.1.e.n
                              inverter.1.o
                              alphajoint.1.i
                              alphajoint.1.o
                              maindistboard.1.a.n
                              maindistboard.1.e.n
                              maindistboard.1.p1.n
                              maindistboard.1.p2.n
                              epspdca.1.m.n
                              epspdca.1.e.n
                              epspdca.1.sw1
                              epspdca.1.sw2
                              basicpdca.1.m.n
                              basicpdca.1.sw1
                              basicpdca.1.sw2
                              seboard.1.bsw.n
                              boardswitch.1.i
                              boardswitch.1.seb.n
                              basicswitch.1.i
                              basicswitch.2.i
                              basicswitch.3.i
                              nil))
)


(defvar *predicted_array* (make-array `(51) :initial-contents
 `("Y-THE FAULT.MODE OF EPSCNTL IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF PVEQUIP IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF BETAJ IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF VOLTREG IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF BATTERIES IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF DC_AC_INVERT IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF ALPHAJ IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF MAINDIST IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF PDCA1 IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF PDCA2 IS PREDICTED-FAILURE"
   "Y-THE FAULT.MODE OF SESWCNTL IS PREDICTED-FAILURE"
```

```
  "Y-THE FAULT.MODE OF SW1_1 IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW1_2 IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW2_1 IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW2_2 IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF EPSPVNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF EPSBJTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF EPSINVERTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF EPSMDBNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF EPSPDCA1NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PVEQUIPEPSNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PVEQUIPBJTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF BJTPVEQUIPNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF BJTEPSNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF BJTVOLTREGNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF VOLTREGBJTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF VOLTREGBATTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF VOLTREGINVERTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF BATTVOLTREGNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF INVERTVOLTREGNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF INVERTEPSNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF INVERTAJTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF AJTINVERTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF AJTMDBNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF MDBAJTNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF MDBEPSNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF MDBPDCA1NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF MDBPDCA2NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA1MDBNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA1EPSNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA1SW11NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA1SW12NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA2MDBNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA2SW21NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF PDCA2SW22NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SESWCNTLSW11NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW11PDCA1NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW11SESWCNTLNODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW12PDCA1NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW21PDCA2NODE IS PREDICTED-FAILURE"
  "Y-THE FAULT.MODE OF SW22PDCA2NODE IS PREDICTED-FAILURE"))
)

(defvar *fault_array* (make-array '(51) :initial-contents
 '("R-THE FAULT.MODE OF EPSCNTL IS FAILED"
  "R-THE FAULT.MODE OF PVEQUIP IS FAILED"
  "R-THE FAULT.MODE OF BETAJ IS FAILED"
  "R-THE FAULT.MODE OF VOLTREG IS FAILED"
  "R-THE FAULT.MODE OF BATTERIES IS FAILED"
  "R-THE FAULT.MODE OF DC_AC_INVERT IS FAILED"
  "R-THE FAULT.MODE OF ALPHAJ IS FAILED"
  "R-THE FAULT.MODE OF MAINDIST IS FAILED"
  "R-THE FAULT.MODE OF PDCA1 IS FAILED"
  "R-THE FAULT.MODE OF PDCA2 IS FAILED"
  "R-THE FAULT.MODE OF SESWCNTL IS FAILED"
  "R-THE FAULT.MODE OF SW1_1 IS FAILED"
  "R-THE FAULT.MODE OF SW1_2 IS FAILED"
  "R-THE FAULT.MODE OF SW2_1 IS FAILED"
  "R-THE FAULT.MODE OF SW2_2 IS FAILED"
  "R-THE FAULT.MODE OF EPSPVNODE IS FAILED"
  "R-THE FAULT.MODE OF EPSBJTNODE IS FAILED"
  "R-THE FAULT.MODE OF EPSINVERTNODE IS FAILED"
  "R-THE FAULT.MODE OF EPSMDBNODE IS FAILED"
  "R-THE FAULT.MODE OF EPSPDCA1NODE IS FAILED"
  "R-THE FAULT.MODE OF PVEQUIPEPSNODE IS FAILED"
  "R-THE FAULT.MODE OF PVEQUIPBJTNODE IS FAILED"
  "R-THE FAULT.MODE OF BJTPVEQUIPNODE IS FAILED"
  "R-THE FAULT.MODE OF BJTEPSNODE IS FAILED"
  "R-THE FAULT.MODE OF BJTVOLTREGNODE IS FAILED"
  "R-THE FAULT.MODE OF VOLTREGBJTNODE IS FAILED"
  "R-THE FAULT.MODE OF VOLTREGBATTNODE IS FAILED"
  "R-THE FAULT.MODE OF VOLTREGINVERTNODE IS FAILED"
  "R-THE FAULT.MODE OF BATTVOLTREGNODE IS FAILED"
  "R-THE FAULT.MODE OF INVERTVOLTREGNODE IS FAILED"
  "R-THE FAULT.MODE OF INVERTEPSNODE IS FAILED"
  "R-THE FAULT.MODE OF INVERTAJTNODE IS FAILED"
  "R-THE FAULT.MODE OF AJTINVERTNODE IS FAILED"
  "R-THE FAULT.MODE OF AJTMDBNODE IS FAILED"
  "R-THE FAULT.MODE OF MDBAJTNODE IS FAILED"
  "R-THE FAULT.MODE OF MDBEPSNODE IS FAILED"
  "R-THE FAULT.MODE OF MDBPDCA1NODE IS FAILED"
  "R-THE FAULT.MODE OF MDBPDCA2NODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA1MDBNODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA1EPSNODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA1SW11NODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA1SW12NODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA2MDBNODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA2SW21NODE IS FAILED"
  "R-THE FAULT.MODE OF PDCA2SW22NODE IS FAILED"
  "R-THE FAULT.MODE OF SESWCNTLSW11NODE IS FAILED"
  "R-THE FAULT.MODE OF SW11PDCA1NODE IS FAILED"
  "R-THE FAULT.MODE OF SW11SESWCNTLNODE IS FAILED"
  "R-THE FAULT.MODE OF SW12PDCA1NODE IS FAILED"
  "R-THE FAULT.MODE OF SW21PDCA2NODE IS FAILED"
  "R-THE FAULT.MODE OF SW22PDCA2NODE IS FAILED"))
)


(defvar *okay_array* (make-array `(51) :initial-contents
 `("G-THE FAULT.MODE OF EPSCNTL IS OKAY"
  "G-THE FAULT.MODE OF PVEQUIP IS OKAY"
  "G-THE FAULT.MODE OF BETAJ IS OKAY"
```

```
    "G-THE FAULT.MODE OF VOLTREG IS OKAY"
    "G-THE FAULT.MODE OF BATTERIES IS OKAY"
    "G-THE FAULT.MODE OF DC_AC_INVERT IS OKAY"
    "G-THE FAULT.MODE OF ALPHAJ IS OKAY"
    "G-THE FAULT.MODE OF MAINDIST IS OKAY"
    "G-THE FAULT.MODE OF PDCA1 IS OKAY"
    "G-THE FAULT.MODE OF PDCA2 IS OKAY"
    "G-THE FAULT.MODE OF SESWCNTL IS OKAY"
    "G-THE FAULT.MODE OF SW1_1 IS OKAY"
    "G-THE FAULT.MODE OF SW1_2 IS OKAY"
    "G-THE FAULT.MODE OF SW2_1 IS OKAY"
    "G-THE FAULT.MODE OF SW2_2 IS OKAY"
    "G-THE FAULT.MODE OF EPSPVNODE IS OKAY"
    "G-THE FAULT.MODE OF EPSBJTNODE IS OKAY"
    "G-THE FAULT.MODE OF EPSINVERTNODE IS OKAY"
    "G-THE FAULT.MODE OF EPSMDBNODE IS OKAY"
    "G-THE FAULT.MODE OF EPSPDCA1NODE IS OKAY"
    "G-THE FAULT.MODE OF PVEQUIPEPSNODE IS OKAY"
    "G-THE FAULT.MODE OF PVEQUIPBJTNODE IS OKAY"
    "G-THE FAULT.MODE OF BJTPVEQUIPNODE IS OKAY"
    "G-THE FAULT.MODE OF BJTEPSNODE IS OKAY"
    "G-THE FAULT.MODE OF BJTVOLTREGNODE IS OKAY"
    "G-THE FAULT.MODE OF VOLTREGBJTNODE IS OKAY"
    "G-THE FAULT.MODE OF VOLTREGBATTNODE IS OKAY"
    "G-THE FAULT.MODE OF VOLTREGINVERTNODE IS OKAY"
    "G-THE FAULT.MODE OF BATTVOLTREGNODE IS OKAY"
    "G-THE FAULT.MODE OF INVERTVOLTREGNODE IS OKAY"
    "G-THE FAULT.MODE OF INVERTEPSNODE IS OKAY"
    "G-THE FAULT.MODE OF INVERTAJTNODE IS OKAY"
    "G-THE FAULT.MODE OF AJTINVERTNODE IS OKAY"
    "G-THE FAULT.MODE OF AJTMDBNODE IS OKAY"
    "G-THE FAULT.MODE OF MDBAJTNODE IS OKAY"
    "G-THE FAULT.MODE OF MDBEPSNODE IS OKAY"
    "G-THE FAULT.MODE OF MDBPDCA1NODE IS OKAY"
    "G-THE FAULT.MODE OF MDBPDCA2NODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA1MDBNODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA1EPSNODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA1SW11NODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA1SW12NODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA2MDBNODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA2SW21NODE IS OKAY"
    "G-THE FAULT.MODE OF PDCA2SW22NODE IS OKAY"
    "G-THE FAULT.MODE OF SESWCNTLSW11NODE IS OKAY"
    "G-THE FAULT.MODE OF SW11PDCA1NODE IS OKAY"
    "G-THE FAULT.MODE OF SW11SESWCNTLNODE IS OKAY"
    "G-THE FAULT.MODE OF SW12PDCA1NODE IS OKAY"
    "G-THE FAULT.MODE OF SW21PDCA2NODE IS OKAY"
    "G-THE FAULT.MODE OF SW22PDCA2NODE IS OKAY"))
)
```

```
(defun check_predicted_array (comp)
      (do ((index 0 (1+ index)))
         (> index 50)
         (if (eq comp (aref *pwrsys_array* index))
            (progn (setq user::*message_out* (aref *predicted_array* index))
                  (princ *message_out*)
                  (setq *data_out* index)
                  (print *data_.out*)
            )
         )
      )
)


(defun check_fault_array (comp)
      (do ((index 0 (1+ index)))
         (> index 50)
         (if (eq comp (aref *pwrsys_array* index))
            (progn (setq *message_out* (aref *fault_array* index))
                  (princ *message_out*)
                  (setq *data_out* index)
                  (print *data_out*)
            )
         )
      )
)


(defun output_routine (line)
      (cond ((eq (sixth line) 'failed)
            (check_fault_array (fourth line))
            )
           ((eq (sixth line) 'predicted_failure)
            (check_predicted_array (fourth line))
            )
      )

      (zl:pkg-goto 'common-lisp-user)

      (user::send_data *message_out*)
      (user::send_data *data_out*)
)

(defun process_list (list)
      (cond ((null (cdr list))
            (output_routine (car list))

            (zl:pkg-goto 'common-lisp-user)

            (user::send_data "END OF SYSTEM CHECK")
```

```
          (user::send_data *finished_processing*)
          )
          (t output_routine (car list))
        )
        (process_list (cdr list))
    )
```

MAKEFILE

```
CFLAGS = -Zg -Zf -lm -ldbm -g

COMM = /work/hester/commdir/

OBJS1 = newpwrsys.o\
    compmenu.o\
    componenthit.o\
    compstat.o\
    diagram.o\
    faulthelp.o\
    healthhelp.o\
    help.o\
    maintspthelp.o\
    mainhelp.o\
    mousemenhelp.o\
    nethelp.o\
    pwrflow.o\
    pwrflowhelp.o

OBJS2 = $(COMM)io_single.o\
    $(COMM)mpath.o\
    $(COMM)semaphore.o\
    $(COMM)shareseg.o\
    $(COMM)support.o

pwrsys: $(OBJS1) $(OBJS2)
    cc -o pwrsys $(OBJS1) $(OBJS2) $(CFLAGS) -lbsd

$(OBJS1): pwrsys.h

$(COMM)mpath.o:  $(COMM)shared.h
    cc -c -o $(COMM)mpath.o $(COMM)mpath.c $(CFLAGS)

$(COMM)support.o: $(COMM)shared.h
    cc -c -o $(COMM)support.o $(COMM)support.c $(CFLAGS)

$(COMM)semaphore.o:
    cc -c -o $(COMM)semaphore.o $(COMM)semaphore.c $(CFLAGS)

$(COMM)io_single.o: $(COMM)shared.h
    cc -c -o $(COMM)io_single.o $(COMM)io_single.c $(CFLAGS)

$(COMM)shareseg.o:
    cc -c -o $(COMM)shareseg.o $(COMM)shareseg.c $(CFLAGS)
```

PWRSYS.H

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "device.h"
#include "math.h"
#include "/work/hester/commdir/shared.h"
#define INITXVALUE 0
#define INITYVALUE 0
#define menuymin 43.8
#define menuymax 131.5
#define menu1xmin 43.0
#define menu2xmin 150.8
#define menu3xmin 322.5
#define menu4xmin 494.2
#define menu5xmin 665.9
#define menu6xmin 837.6

#define EPSCNTL     0
#define PVEQUIP     1
#define BETAJ       2
#define VOLTREG     3
#define BATTERIES   4
#define DC_AC_INVERT 5
#define ALPHAJ      6
#define MAINDIST    7
#define PDCA1       8
#define PDCA2       9
#define SESWCNTL    10
#define SW1_1       11
#define SW1_2       12
#define SW2_1       13
#define SW2_2       14
#define EPSPVNODE   15
#define EPSBJTNODE  16
#define EPSINVERTNODE 17
#define EPSMDBNODE    18
#define EPSPDCA1NODE  19
#define PVEQUIPEPSNODE 20
#define PVEQUIPBJTNODE 21
#define BJTPVEQUIPNODE 22
#define BJTEPSNODE    23
#define BJTVOLTREGNODE 24
#define VOLTREGBJTNODE 25
```

```
#define VOLTREGBATTNODE 26
#define VOLTREGINVERTNODE 27
#define BATTVOLTREGNODE  28
#define INVERTVOLTREGNODE 29
#define INVERTEPSNODE    30
#define INVERTAJTNODE    31
#define AJTINVERTNODE    32
#define AJTMDBNODE       33
#define MDBAJTNODE       34
#define MDBEPSNODE       35
#define MDBPDCA1NODE     36
#define MDBPDCA2NODE     37
#define PDCA1MDBNODE     38
#define PDCA1EPSNODE     39
#define PDCA1SW11NODE    40
#define PDCA1SW12NODE    41
#define PDCA2MDBNODE     42
#define PDCA2SW21NODE    43
#define PDCA2SW22NODE    44
#define SESWCNTLSW11NODE 45
#define SW11PDCA1NODE    46
#define SW11SESWCNTLNODE 47
#define SW12PDCA1NODE    48
#define SW21PDCA2NODE    49
#define SW22PDCA2NODE    50
#define EPSCN1       51
#define EPSCN2       52
#define EPSCN3       53
#define EPSCN4       54
#define EPSCN5       55
#define CN0        56
#define CN1a       57
#define CN1b       58
#define CN2        59
#define CN3        60
#define CN4        61
#define CN5        62
#define CN6        63
#define CN7a       64
#define CN7b       65
#define CN8a       66
#define CN8b       67
#define CN8c       68
#define CN8d       69
#define SESWCN       70
#define XMAXDIAGRAM  7.0
#define YMAXDIAGRAM  9.5
#define FLASHBLACKR  61
#define FLASHBLACKY  78
#define EXIT       99
#define PWRWINX1  10
```

```
#define PWRWINX2  1000
#define PWRWINY1  10
#define PWRWINY2  730
#define CHECKSYSTEM 51
#define FINISHED_PROCESSING 555

typedef char str80[80];

struct sys_component
{
  Object objname;
  Tag    tagname;
  int    voltlevel;
  int    freqlevel;
  char   volttype[4];
  str80  compstatus;
};

struct sys_component power[100];

Machine remotemachine;

char commbuffer[LARGESTREAD];

float commfloat;

int commint;
```

SHARED.H

```
/**********************************************************************
***********************************************************************
* TITLE  : Inter-Computer Communication Package
*
* MODULE : shared.h
*
* VERSION: 4.0
*
* DATE   : 15 December 1987
*
* AUTHOR : Theodore H. Barrow
***********************************************************************
* HISTORY:
*
*   VERSION: 1.0
*
*   DATE   : 6 February 1987
*
*   AUTHOR : Michael J. Zyda
*
*   DESC.  : Contains all defines and special constants for shared
*            memory socket system.
*
*   VERSION: 2.0
*
*   DATE   : 27 May 1987
*
*   AUTHOR : Theodore H. Barrow
*
*   DESC.  : Added a typedef of structure for use by various routines.
*            Added message types for high level read/write protocol.
*
*   VERSION: 3.0
*
*   DATE   : 21 October 1987
*
*   AUTHOR : Theodore H. Barrow
*
*   DESC.  : Changed dependencies of buffer calculation constants so that
*            only one need change.  Added additional message types.
*
*   VERSION: 4.0
*
*   DATE   : 15 December 1987
```

```
*
*    AUTHOR : Theodore H. Barrow
*
*    DESC. : Added field to buffer set so that each link would have its
*           own area to handle partial receipt of messages.
************************************************************************
************************************************************************
/*
  the following 3 defines are the changeable parameters

        LARGESTREAD MUST be divisible by 4
*/

#define SENDLOCATION "/work/hester/commdir/send"  /* the name of the program
                                    to run for the sender */

#define RECEIVELOCATION "/work/hester/commdir/receive" /* the name of program
                                    to run for the receiver */

#define LARGESTREAD 252     /* the largest read (i.e. buffer size) */


/* The following defines are constants or are derived from LARGESTREAD */

#define SENDEROFFSET (LARGESTREAD + 4)  /* the sender data starts here */

#define WSENDEROFFSET (SENDEROFFSET / 4) /* long word offset for sender data */

#define RECEIVEROFFSET 0    /* the receiver data starts at byte 0 */

#define WRECEIVEROFFSET 0    /* the receiver data starts at long word 0 */

#define PROTOCOLHOLDOFFSET (SENDEROFFSET * 2)  /* holding area starts after
                                    sender area */
#define MAXSHAREDSIZE (PROTOCOLHOLDOFFSET + 12) /* the number of bytes in
the
                                    shared segment */

#define CHARACTER_TYPE      'B' /* code for characters */
#define INTEGER_TYPE        'I' /* code for integers */
#define FLOAT_TYPE          'R' /* code for floats */
#define CHARACTER_ARRAY_TYPE 'C' /* code for character arrays */
#define INTEGER_ARRAY_TYPE   'J' /* code for integer arrays */
#define FLOAT_ARRAY_TYPE     'S' /* code for float arrays */

#define CHARACTER_SIZE 1        /* character size in bytes */
#define INTEGER_SIZE   sizeof(1)  /* integer size in bytes */
#define FLOAT_SIZE     sizeof(1.0) /* float size in bytes */

/* the following is the structure type definition needed for each machine
  you want to communicate to...
```

```
*/

  typedef struct {
            char *segment;  /* ptr to shared memory segment */

            int shmid;      /* system generated shared mem. id */

            int sendsem;    /* semaphore used to wakeup the sender
                                process.
                              */

            int receivesem; /* semaphore used to wakeup the
                                receiver process...
                              */
          } Machine ;
```

HELP.C

```
#include "pwrsys.h"

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

                    HELP_MENU

        The main control module for help menus

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

help_menu(menunum)
  int menunum;
{

  short data;
  Boolean CANCEL;


  CANCEL = FALSE;



  mousemenu();

  while (!CANCEL)
  {
    if (qtest())
    {
      switch(qread(&data))
      {
      /* Cancel to depart loop */

        case REDRAW:     reshapeviewport();
                  break;


        case LEFTMOUSE:  CANCEL = TRUE;
                  break;
```

```
            case RIGHTMOUSE:  mousemenu();
                      break;

            case MIDDLEMOUSE: processhelp(menunum);
                      break;

            default:       break;
          }
        }
    } /* end while */
    system("clear\n");
} /* end help_menu */


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

                    PROCESSHELP

        Allows the creation and display of specific help for current
        menu

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

processhelp(menunum)
    int menunum;
{
    color(BLUE);
    clear();
    color(WHITE);

    switch(menunum)
    {
      case 1:  mainhelp();
               break;

      case 2:  nethelp();
               break;

      case 3:  healthhelp();
               break;

      case 4:  maintspthelp();
               break;

      case 5:  faulthelp();
               break;
        case 6:  pwrflowhelp();
                 break;

    }
} /* end processhelp */
```

MAINHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP1 3

mainhelp()
{

  static str80 helpmain[NUMHELP1] =
  {
     "Hi, I'm main help.",
     " ",
    "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP1; j = j + 1)
  {
     incr = 6.0 - (0.2 * (float)j);
     cmov2(2.0, incr);
     charstr(helpmain[j]);
  }

  swapbuffers();

} /* end mainhelp */
```

NETHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP2 3

nethelp()
{

  static str80 helpnet[NUMHELP2] =
  {
     "Hi, I'm net help.",
     " ",
    "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP2; j = j + 1)
  {
     incr = 6.0 - (0.2 * (float)j);
     cmov2(2.0, incr);
     charstr(helpnet[j]);
  }

  swapbuffers();

} /* end nethelp */
```

HEALTHHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP3 3

healthhelp()
{

  static str80 helphealth[NUMHELP3] =
  {
    "Hi, I'm health help.",
    " ",
    "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP3; j = j + 1)
  {
    incr = 6.0 - (0.2 * (float)j);
    cmov2(2.0, incr);
    charstr(helphealth[j]);
  }

  swapbuffers();

} /* end healthhelp */
```

MAINTSPTHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP4 3

maintspthelp()
{

  static str80 helpmaintspt[NUMHELP4] =
  {
     "Hi, I'm maintspt help.",
     " ",
     "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP4; j = j + 1)
  {
     incr = 6.0 - (0.2 * (float)j);
     cmov2(2.0, incr);
     charstr(helpmaintspt[j]);
  }

  swapbuffers();

} /* end maintspthelp */
```

FAULTHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP5 3

faulthelp()
{

  static str80 helpfault[NUMHELP5] =
  {
     "Hi, I'm fault help.",
     " ",
    "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP5; j = j + 1)
  {
     incr = 6.0 - (0.2 * (float)j);
     cmov2(2.0, incr);
     charstr(helpfault[j]);
  }

  swapbuffers();

} /* end faulthelp */
```

PWRFLOWHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMHELP6 3

pwrflowhelp()
{

  static str80 helppwrflow[NUMHELP6] =
  {
     "Hi, I'm pwrflow help.",
     " ",
    "Use RIGHT MOUSE to return to mouse help menu."
  };

  int j;
  float incr;

  for(j = 0; j < NUMHELP6; j = j + 1)
  {
     incr = 6.0 - (0.2 * (float)j);
     cmov2(2.0, incr);
     charstr(helppwrflow[j]);
  }

  swapbuffers();

} /* end pwrflowhelp */
```

MOUSEMENHELP.C

```
#include <strings.h>
#include "stdio.h"
#include "gl.h"
#include "pwrsys.h"
#define NUMARRAY2 23

mousemenu()
{
  static str80 mousemen[NUMARRAY2] =
  {
    "                                        /   /",
    "                                       /   /",
    "                                      /   /",
    "                                      |   |",
    "                                      |   |",
    "                                      |   |",
    "______________________________________|___|______________________________",
    "|                                                                        |",
    "|  ------------------        --------------------      ------------------ |",
    "|  |                |        |                  |      |                | |",
    "|  |  LEFTMOUSE     |        |   MIDDLEMOUSE    |      |  RIGHTMOUSE    | |",
    "|  |                |        |                  |      |                | |",
    "|  |  TERMINATE     |        |     GO BACK      |      |    SELECT      | |",
    "|  |  ACTIVITY      |        |    TO MOUSE      |      |     MENU       | |",
    "|  |                |        |   MENU WHEN      |      |    OPTION      | |",
    "|  |  (CANCEL)      |        |    IN HELP       |      |                | |",
    "|  |                |        |      MENU        |      |                | |",
    "|  ------------------        --------------------      ------------------ |",
    "|________________________________________________________________________|",
    "    ",
    "    ",
    "Use MIDDLE MOUSE to go to help for this menu panel.",
    "Use LEFT MOUSE to cancel back to previous menu."
  };

  int j;
  float incr;

  color(BLUE);
  clear();
  color(WHITE);
```

```
    for(j = 0; j < NUMARRAY2; j = j + 1)
    {
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.0, incr);
      charstr(mousemen[j]);
    }

    swapbuffers();

} /* end mousemenu */
```

NEWPWRSYS.C

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

#include "pwrsys.h"

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

                    MAIN()

This is the primary control module for the entire menu interface.  It
is mouse-oriented and utilizes the IRIS window manager, MEX.

    LEFT MOUSE   (MOUSE3)  -> Pop up a Menu (CANCEL)
    RIGHT MOUSE  (MOUSE1)  -> Select a Menu

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

main()
{

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

  /* menu item names */


  short data;
  static int pupval = 1;
  int level1, level2, level3, level4a, level4b, level4c, leveld;
  int compnum;

  /* initialize the IRIS system */

  initialize();

  /* orthographic projection 2D for world coordinate system */

   ortho2(0.0, 9.5, 0.0, 7.0);

  /* set mouse limits */
```

```
    setvaluator(MOUSEX, INITXVALUE, 0, XMAXSCREEN);
    setvaluator(MOUSEY, INITYVALUE, 0, YMAXSCREEN);

/* define two blinking colors -
     one to flash between black and yellow for predicted faults
     one to flash between black and red for actual faults */

    mapcolor(FLASHBLACKY, 0, 0, 0);
    mapcolor(FLASHBLACKR, 0, 0, 0);

    blink(32, FLASHBLACKR, 255, 0, 0);
    blink(32, FLASHBLACKY, 255, 255, 0);


/* build the system diagram */

    make_diagram();

    /* build pop-up menus */

    level4c = defpup("POWER FLOW %t | Help %x13 | Charge Battery %x14 |\
            Discharge Battery %x15 | Use Voltaics %x16 |\
            Use Voltaics & Batteries %x17");

    level4b = defpup("FAULT MANAGEMENT %t | Help %x11 |\
            Fault Detection %x12");

    level4a = defpup("MAINTENANCE SUPPORT %t | Help %x9 |\
            Status Prediction %x10");

    level3  = defpup("HEALTH MANAGEMENT %t | Help %x5 |\
            Maintenance Support %x6 %m | Fault Management %x7 %m |\
            Power Flow %x8 %m", level4a, level4b, level4c);

    level2  = defpup("POWER NETWORK CONTROL %t | Help %x3 |\
            Health Management %x4 %m", level3);

    level1  = defpup("PMAD %t | Help %x1 | Power Network Control %x2 %m |\
            Status Prediction %x10 | Fault Detection %x12 |\
            Power Flow %x8 %m | EXIT %x99", level2, level4c);

    leveld  = defpup("COMPONENT STATUS %t | Help %x5 |\
            Fault Detection %x12 | Status Prediction %x10");

    color(WHITE);
    clear();
    call_diagram();
    swapbuffers();
```

```
  machinepath(7, "nps-sym1", 1026, 1027, "server",
    &remotemachine);


  while(pupval != EXIT)
  {
    if (qtest())
    {
      switch(qread(&data))
      {
        case REDRAW:     reshapeviewport();
                         break;

        case RIGHTMOUSE: if (data == 1)
                         {
                          compnum = componenthit();

                          if(compnum >= EPSCNTL
                          && compnum <= SW22PDCA2NODE)
                             pupval = dopup(leveld);
                          else
                             pupval = dopup(level1);

                          processmenuhit(pupval, compnum);
                         }
                         break;

        default:         break;
      }
    }
    color(WHITE);
    clear();
    call_diagram();
    swapbuffers();

  } /* end while */

  commint = 999;
  write_integer(&remotemachine, &commint);
  deletemachinepath(&remotemachine);

  blink(0, FLASHBLACKR, 255, 0, 0);
  blink(0, FLASHBLACKY, 255, 255, 0);
  ginit();
  textinit();
  gexit();

  system("gclear\n");

} /* end main */
```

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

                PROCESSMENUHIT(PUPVAL)

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

processmenuhit(pupval, num)
   int pupval;
{
   int menunum;
   char help_file[80]; /* help file for each menu level */

   switch(pupval)
   {
      case -1:
      case  2:
      case  4:
      case  6:
      case  7:
      case  8:
      case 99:  break;

      case  1:  menunum = 1;
                help_menu(menunum);
                break;

      case  3:  menunum = 2;
                help_menu(menunum);
                break;

      case  5:  menunum = 3;
                help_menu(menunum);
                break;

      case  9:  menunum = 4;
                help_menu(menunum);
                break;

      case 10:  statusprediction(num);
                break;

      case 11:  menunum = 5;
                help_menu(menunum);
                break;

      case 12:  faultisolation(num);
                break;

      case 13:  menunum = 6;
```

```
          help_menu(menunum);
          break;

    case 14: charge_batt();
          break;

    case 15: discharge_batt();
          break;

    case 16: use_pv_only();
          break;

    case 17: use_pv_batt();
          break;

    default: break;
  }
} /* end processmenuhit */



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

                INITIALIZE()

    This subroutine executes several routines that set up the
    graphics environment for this program.

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

initialize()
{
  keepaspect(XMAXSCREEN + 1, YMAXSCREEN + 1);
  prefposition(PWRWINX1, PWRWINX2, PWRWINY1, PWRWINY2);
  winopen("power");
  wintitle("SPACE STATION POWER MANAGEMENT & DISTRIBUTION SYSTEM
(PMAD)");
  winattach();
  doublebuffer();
  gconfig();
  qdevice(REDRAW);
  qdevice(RIGHTMOUSE);
  qdevice(MIDDLEMOUSE);
  qdevice(LEFTMOUSE);
} /* end initialize */
```

COMPONENTHIT.C

```
#include "pwrsys.h"

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

                    COMPONENTHIT

  The following functions determines if the current mouse location
  corresponds to a component item hit.

  The assumption is made that the mouse moves in a world space
  that is the same as the screen space.

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

int componenthit()
{
  int i;  /* temp loop counter */

  int pos; /* location of hit menu item */

  float x,y; /* location of valuator */

  int inside_rect();  /* function that returns true if x,y is inside
              the defined box */

  int inside_circ();  /* function that returns true if x,y is inside
              the defined circle */

  static float comppos[49][5] =
  {  0.4, 5.4, 8.0, 5.8, EPSCNTL,
     0.4, 4.4, 1.4, 4.8, PVEQUIP,
     2.5, 4.4, 2.9, 4.8, VOLTREG,
     2.0, 3.5, 3.4, 3.8, BATTERIES,
     3.2, 4.4, 4.2, 4.8, DC_AC_INVERT,
     6.4, 2.6, 6.8, 5.0, MAINDIST,
     7.4, 4.2, 8.0, 4.8, PDCA1,
     7.4, 3.0, 8.0, 3.6, PDCA2,
     8.4, 5.4, 9.2, 5.8, SESWCNTL,
     8.6, 4.6, 9.0, 5.0, SW1_1,
     8.6, 4.0, 9.0, 4.4, SW1_2,
     8.6, 3.4, 9.0, 3.8, SW2_1,
     8.6, 2.8, 9.0, 3.2, SW2_2,
        0.8, 5.3, 1.0, 5.4, EPSPVNODE,
        1.9, 5.3, 2.1, 5.4, EPSBJTNODE,
        3.6, 5.3, 3.8, 5.4, EPSINVERTNODE,
        6.5, 5.3, 6.7, 5.4, EPSMDBNODE,
        7.6, 5.3, 7.8, 5.4, EPSPDCA1NODE,
        0.8, 4.8, 1.0, 4.9, PVEQUIPEPSNODE,
        1.4, 4.5, 1.5, 4.7, PVEQUIPBJTNODE,
        1.7, 4.5, 1.8, 4.7, BJTPVEQUIPNODE,
        1.9, 4.8, 2.1, 4.9, BJTEPSNODE,
        2.2, 4.5, 2.3, 4.7, BJTVOLTREGNODE,
        2.4, 4.5, 2.5, 4.7, VOLTREGBJTNODE,
        2.6, 4.3, 2.8, 4.4, VOLTREGBATTNODE,
        2.9, 4.5, 3.0, 4.7, VOLTREGINVERTNODE,
        2.6, 3.8, 2.8, 3.9, BATTVOLTREGNODE,
        3.1, 4.5, 3.2, 4.7, INVERTVOLTREGNODE,
        3.6, 4.8, 3.8, 4.9, INVERTEPSNODE,
        4.2, 4.5, 4.3, 4.7, INVERTAJTNODE,
        5.3, 3.8, 5.4, 4.0, AJTINVERTNODE,
        5.8, 3.8, 5.9, 4.0, AJTMDBNODE,
        6.3, 3.8, 6.4, 4.0, MDBAJTNODE,
        6.5, 5.0, 6.7, 5.1, MDBEPSNODE,
        6.8, 4.4, 6.9, 4.6, MDBPDCA1NODE,
        6.8, 3.2, 6.9, 3.4, MDBPDCA2NODE,
        7.3, 4.4, 7.4, 4.6, PDCA1MDBNODE,
        7.6, 4.8, 7.8, 4.9, PDCA1EPSNODE,
        8.0, 4.6, 8.1, 4.8, PDCA1SW11NODE,
        8.0, 4.2, 8.1, 4.4, PDCA1SW12NODE,
        7.3, 3.2, 7.4, 3.4, PDCA2MDBNODE,
        8.0, 3.4, 8.1, 3.6, PDCA2SW21NODE,
        8.0, 3.0, 8.1, 3.2, PDCA2SW22NODE,
        8.7, 5.3, 8.9, 5.4, SESWCNTLSW11NODE,
        8.5, 4.6, 8.6, 4.8, SW11PDCA1NODE,
        8.7, 5.0, 8.9, 5.1, SW11SESWCNTLNODE,
        8.5, 4.2, 8.6, 4.4, SW12PDCA1NODE,
        8.5, 3.4, 8.6, 3.6, SW21PDCA2NODE,
        8.5, 3.0, 8.6, 3.2, SW22PDCA2NODE

};
  float Sx, Sy;

  Sx = getvaluator(MOUSEX);
  Sy = getvaluator(MOUSEY);


  /* Compute world coordinates of mouse...*/


  x =  (((9.5 - 0.0)/(PWRWINX2 - PWRWINX1)) * (Sx - PWRWINX1)) + 0.0;
  y =  (((7.0 - 0.0)/(PWRWINY2 - PWRWINY1)) * (Sy - PWRWINY1)) + 0.0;
```

```
/* we assume that the mouse coordinates returned match world coords*/

/* say that we have not yet found the hit */

pos = -1;

for(i = 0; i <= 48; i = i+1)
{
  if(inside_rect(x,y,comppos[i][0],comppos[i][1],
    comppos[i][2],comppos[i][3]))
  {
    /* this is the guy we have selected...
      set a flag indicating that this selection
      is the current component item */

    pos = comppos[i][4];
    break;

  }
}

    if(inside_circ(x,y,2.0,4.6,0.2))
      pos = BETAJ;
   else
     if(inside_circ(x,y,5.6,3.9,0.2))
      pos = ALPHAJ;

 return(pos);

} /* end componenthit */


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

                    INSIDE_RECT

  this function determines if (x,y) is inside the box defined by
  the coordinates (xmin,ymin)-(xmax,ymax)

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

int inside_rect(x,y,xmin,ymin,xmax,ymax)
   float x,y;  /* location of the cursor */
   float xmin,ymin,xmax,ymax;  /* bounding box to check if cursor
                        is inside */
{
  if(((x > xmin) && (x < xmax)) && ((y > ymin) && (y < ymax)))
    return(TRUE);
  else
```

```
    return(FALSE);
}


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

                    INSIDECIRC

  this function determines if (x,y) is inside the circle defined
  by the center coordinates (xcent, ycent) and the radius

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

int inside_circ(x,y,xcent,ycent,radius)
   float x,y;  /* location of the cursor */
   float xcent,ycent,radius;  /* bounding box to check if cursor
                          is inside */
{
   float D;

   D = sqrt(((x - xcent) * (x - xcent)) + ((y - ycent) * (y - ycent)));
   if(D < radius)
     return(TRUE);
   else
     return(FALSE);
}
```

COMPMENU.C

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

              COMPMENU

     Displays the menu that provides the choice of items that are
     predicted to fail or that have actually failed

************************************************************************
************************************************************************/
#include "pwrsys.h"
#define NUMARRAY1 20

compmenu(type)
   int type;
{

   static str80 compmenu[NUMARRAY1] =
   {
     "  ___________________COMPONENTS MENU__________________",
     "                                 ",
     "    EPS CONTROL BOARD  - 0 SWITCH 2-1     - 18",
     "    EPS CONNECTION 1   - 1 SWITCH 2-2     - 19",
     "     EPS CONNECTION 2  - 2 CONNECTION 0   - 20",
     "     EPS CONNECTION 3  - 3 CONNECTION 1a  - 21",
     "     EPS CONNECTION 4  - 4 CONNECTION 1b  - 22",
     "     EPS CONNECTION 5  - 5 CONNECTION 2   - 23",
     "     PHOTOVOLTAIC EQUIP - 6 CONNECTION 3   - 24",
     "     BETA JOINT        - 7 CONNECTION 4   - 25",
     "     VOLTAGE REGULATOR - 8 CONNECTION 5   - 26",
     "     BATTERIES         - 9 CONNECTION 6   - 27",
     "     DC-AC-INVERTER    - 10 CONNECTION 7a  - 28",
     "     ALPHA JOINT       - 11 CONNECTION 7b  - 29",
     "     MAIN DISTRIB BOARD - 12 CONNECTION 8a  - 20",
     "     PWRDISTCNTL ASSEM1 - 13 CONNECTION 8b  - 31",
     "     PWRDISTCNTL ASSEM2 - 14 CONNECTION 8c  - 32",
     "    SYS ELEM SW CNTL   - 15 CONNECTION 8d  - 33",
     "     SWITCH 1-1        - 16 SYS ELEM CONN  - 34",
     "     SWITCH 1-2        - 17"
   };

   int j;
   float incr;
   char chr;
```

```
    for(j = 0; j < NUMARRAY1; j = j + 1)
    {
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.0, incr);
      charstr(compmenu[j]);
    }


    if (type == 0)
    {
      j = NUMARRAY1 + 2;
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.5, incr);
      charstr("Enter the numbers corresponding to the items");
      j = j + 1;
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.5, incr);
      charstr("that have failed, separated by commas.");
    }

    else
    {
      j = NUMARRAY1 + 2;
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.5, incr);
      charstr("Enter the numbers corresponding to the items that");
      j = j + 1;
      incr = 6.0 - (0.2 * (float)j);
      cmov2(2.5, incr);
      charstr("are predicted to fail, separated by commas.");
    }

  } /* end compmenu */
```

DIAGRAM.C

```
#include "pwrsys.h"

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

                    MAKE_DIAGRAM

      Creates the thirty five different objects in the system
      diagram.

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

make_diagram()
{
  int i;

  for(i = EPSCNTL; i <= SESWCN; i = i + 1)
  {
   power[i].objname = genobj();
    makeobj(power[i].objname);
    power[i].tagname = gentag();
    maketag(power[i].tagname);

    switch(i)
    {
      case EPSCNTL:     color(BLACK);
                    rect(0.4, 5.4, 8.0, 5.8);
                    cmov2(2.3, 5.5);
                   charstr("ELECTRICAL POWER SYSTEM ");
                   charstr("MANAGEMENT AND CONTROL");
                   break;

/* Photovoltaic equipment - the solar power source */

      case PVEQUIP:     color(BLACK);
                    rect(0.4, 4.4, 1.4, 4.8);
                    cmov2(0.8, 4.65);
                   charstr("PV");
                    cmov2(0.5, 4.5);
                   charstr("EQUIPMENT");
                   break;
```

```
/* Betajoint - secondary rotation joint for solar panels */

        case BETAJ:      color(BLACK);
                         circ(2.0, 4.6, 0.2);
                         cmov2(1.9, 4.55);
                         charstr("BJT");
                         break;

/* Voltage Regulator between PVEQUIP and BATTERIES */

        case VOLTREG:    color(BLACK);
                         rect(2.5, 4.4, 2.9, 4.8);
                         cmov2(2.55, 4.65);
                         charstr("VOLT");
                         cmov2(2.55, 4.5);
                         charstr("REG");
                         break;

        case BATTERIES:  color(BLACK);
                         rect(2.0, 3.5, 3.4, 3.8);
                         cmov2(2.3, 3.6);
                         charstr("BATTERIES");
                         break;

/* Converter of DC power to AC power */

        case DC_AC_INVERT: color(BLACK);
                         rect(3.2, 4.4, 4.2, 4.8);
                         cmov2(3.45, 4.65);
                         charstr("DC-AC");
                         cmov2(3.3, 4.5);
                         charstr("INVERTERS");
                         break;

/* Alphajoint - primary rotation joint for solar panels */

        case ALPHAJ:     color(BLACK);
                         circ(5.6, 3.9, 0.2);
                         cmov2(5.5, 3.85);
                         charstr("AJT");
                         break;

/* Main Distribution Board - distributes power throughout the rest
                  of the system                 */

        case MAINDIST:   color(BLACK);
                         rect(6.4, 2.6, 6.8, 5.0);
                         cmov2(6.55,4.9);
                         charstr("M");
                         cmov2(6.55,4.75);
                         charstr("A");
```

```
                 cmov2(6.55,4.6);
                 charstr("I");
                 cmov2(6.55,4.45);
                 charstr("N");
                 cmov2(6.55,4.25);
                 charstr("D");
                 cmov2(6.55,4.10);
                 charstr("I");
                 cmov2(6.55,3.95);
                 charstr("S");
                 cmov2(6.55,3.80);
                 charstr("T");
                 cmov2(6.55,3.65);
                 charstr("R");
                 cmov2(6.55,3.50);
                 charstr("I");
                 cmov2(6.55,3.35);
                 charstr("B");
                 cmov2(6.55,3.20);
                 charstr("U");
                 cmov2(6.55,3.05);
                 charstr("T");
                 cmov2(6.55,2.90);
                 charstr("I");
                 cmov2(6.55,2.75);
                 charstr("O");
                 cmov2(6.55,2.6);
                 charstr("N");
                 break;

/* Power Distribution and Control Assemblies 1 and 2 */

     case PDCA1:     color(BLACK);
                 rect(7.4, 4.2, 8.0, 4.8);
                 cmov2(7.45, 4.5);
                 charstr("PDCA 1");
                 break;

     case PDCA2:     color(BLACK);
                 rect(7.4, 3.0, 8.0, 3.6);
                 cmov2(7.45, 3.3);
                 charstr("PDCA 2");
                 break;

/* System Element Switch Control */

     case SESWCNTL:    color(BLACK);
                 rect(8.4, 5.4, 9.2, 5.8);
                 cmov2(8.45, 5.65);
                 charstr("SYS ELEM");
                 cmov2(8.5, 5.5);
```

```
                  charstr("SW CNTL");
                  break;

/* The four switches coming off the PDCAs */

        case SW1_1:     color(BLACK);
                   rect(8.6, 4.6, 9.0, 5.0);
                  cmov2(8.65, 4.7);
                  charstr("SW11");
                  break;

        case SW1_2:     color(BLACK);
                   rect(8.6, 4.0, 9.0, 4.4);
                  cmov2(8.65, 4.2);
                  charstr("SW12");
                  break;

        case SW2_1:     color(BLACK);
                   rect(8.6, 3.4, 9.0, 3.8);
                  cmov2(8.65, 3.5);
                  charstr("SW21");
                  break;

        case SW2_2:     color(BLACK);
                   rect(8.6, 2.8, 9.0, 3.2);
                  cmov2(8.65, 3.0);
                  charstr("SW22");
                  break;


        case EPSPVNODE:          color(BLUE);
                        rectf(0.8,5.3,1.0,5.4);
                      break;

        case EPSBJTNODE:          color(BLUE);
                        rectf(1.9,5.3,2.1,5.4);
                      break;

        case EPSINVERTNODE:        color(BLUE);
                        rectf(3.6,5.3,3.8,5.4);
                      break;

        case EPSMDBNODE:          color(BLUE);
                        rectf(6.5,5.3,6.7,5.4);
                      break;

        case EPSPDCA1NODE:        color(BLUE);
                        rectf(7.6,5.3,7.8,5.4);
                      break;
```

```
case PVEQUIPEPSNODE:      color(BLUE);
                     rectf(0.8,4.8,1.0,4.9);
                     break;

case PVEQUIPBJTNODE:      color(BLUE);
                     rectf(1.4,4.5,1.5,4.7);
                     break;

case BJTPVEQUIPNODE:      color(BLUE);
                     rectf(1.7,4.5,1.8,4.7);
                     break;

case BJTEPSNODE:          color(BLUE);
                     rectf(1.9,4.8,2.1,4.9);
                     break;

case BJTVOLTREGNODE:      color(BLUE);
                     rectf(2.2,4.5,2.3,4.7);
                     break;

case VOLTREGBJTNODE:      color(BLUE);
                     rectf(2.4,4.5,2.5,4.7);
                     break;

case VOLTREGBATTNODE:     color(BLUE);
                     rectf(2.6,4.3,2.8,4.4);
                     break;

case VOLTREGINVERTNODE:   color(BLUE);
                     rectf(2.9,4.5,3.0,4.7);
                     break;

case BATTVOLTREGNODE:     color(BLUE);
                     rectf(2.6,3.8,2.8,3.9);
                     break;

case INVERTVOLTREGNODE:   color(BLUE);
                     rectf(3.1,4.5,3.2,4.7);
                     break;

case INVERTEPSNODE:       color(BLUE);
                     rectf(3.6,4.8,3.8,4.9);
                     break;

case INVERTAJTNODE:       color(BLUE);
                     rectf(4.2,4.5,4.3,4.7);
                     break;

case AJTINVERTNODE:       color(BLUE);
                     rectf(5.3,3.8,5.4,4.0);
                     break;
```

```
case AJTMDBNODE:          color(BLUE);
                          rectf(5.8,3.8,5.9,4.0);
                          break;

case MDBAJTNODE:          color(BLUE);
                          rectf(6.3,3.8,6.4,4.0);
                          break;

case MDBEPSNODE:          color(BLUE);
                          rectf(6.5,5.0,6.7,5.1);
                          break;

case MDBPDCA1NODE:        color(BLUE);
                          rectf(6.8,4.4,6.9,4.6);
                          break;

case MDBPDCA2NODE:        color(BLUE);
                          rectf(6.8,3.2,6.9,3.4);
                          break;

case PDCA1MDBNODE:        color(BLUE);
                          rectf(7.3,4.4,7.4,4.6);
                          break;

case PDCA1EPSNODE:        color(BLUE);
                          rectf(7.6,4.8,7.8,4.9);
                          break;

case PDCA1SW11NODE:       color(BLUE);
                          rectf(8.0,4.6,8.1,4.8);
                          break;

case PDCA1SW12NODE:       color(BLUE);
                          rectf(8.0,4.2,8.1,4.4);
                          break;

case PDCA2MDBNODE:        color(BLUE);
                          rectf(7.3,3.2,7.4,3.4);
                          break;

case PDCA2SW21NODE:       color(BLUE);
                          rectf(8.0,3.4,8.1,3.6);
                          break;

case PDCA2SW22NODE:       color(BLUE);
                          rectf(8.0,3.0,8.1,3.2);
                          break;

case SESWCNTLSW11NODE:    color(BLUE);
                          rectf(8.7,5.3,8.9,5.4);
```

```
                        break;

        case SW11PDCA1NODE:     color(BLUE);
                         rectf(8.5,4.6,8.6,4.8);
                        break;

        case SW11SESWCNTLNODE:   color(BLUE);
                         rectf(8.7,5.0,8.9,5.1);
                        break;

        case SW12PDCA1NODE:     color(BLUE);
                         rectf(8.5,4.2,8.6,4.4);
                        break;

        case SW21PDCA2NODE:     color(BLUE);
                         rectf(8.5,3.4,8.6,3.6);
                        break;

        case SW22PDCA2NODE:     color(BLUE);
                         rectf(8.5,3.0,8.6,3.2);
                        break;

/* The five connections from EPSCNTL to other components */

        case EPSCN1:     color(BLACK);
                    rectf(0.85, 4.9, 0.95, 5.3);
                   break;

        case EPSCN2:     color(BLACK);
                    rectf(1.95, 4.9, 2.05, 5.3);
                   break;

        case EPSCN3:     color(BLACK);
                    rectf(3.65, 4.9, 3.75, 5.3);
                   break;

        case EPSCN4:     color(BLACK);
                    rectf(6.55, 5.1, 6.65, 5.3);
                   break;

        case EPSCN5:     color(BLACK);
                    rectf(7.65, 4.9, 7.75, 5.3);
                   break;

/* CN0 through CN8 - Connections between components that will show
            power flow through the system          */

        case CN0:        color(BLACK);
                    rectf(1.5, 4.55, 1.7, 4.65);
                   break;
```

```
case CN1a:      color(BLACK);
            rectf(2.3, 4.55, 2.4, 4.65);
            break;

case CN1b:      color(BLACK);
            rectf(2.65, 3.9, 2.75, 4.3);
            break;

case CN2:       color(BLACK);
            rectf(3.0, 4.55, 3.1, 4.65);
            break;

case CN3:       color(BLACK);
            rectf(4.3, 4.55, 4.8, 4.65);
            break;

case CN4:       color(BLACK);
            rectf(4.7, 3.85, 4.8, 4.55);
            break;

case CN5:       color(BLACK);
            rectf(4.8, 3.85, 5.3, 3.95);
            break;

case CN6:       color(BLACK);
            rectf(5.9, 3.85, 6.3, 3.95);
            break;

case CN7a:      color(BLACK);
            rectf(6.9, 4.45, 7.3, 4.55);
            break;

case CN7b:      color(BLACK);
            rectf(6.9, 3.25, 7.3, 3.35);
            break;

case CN8a:      color(BLACK);
            rectf(8.1, 4.65, 8.5, 4.75);
            break;

case CN8b:      color(BLACK);
            rectf(8.1, 4.25, 8.5, 4.35);
            break;

case CN8c:      color(BLACK);
            rectf(8.1, 3.45, 8.5, 3.55);
            break;

case CN8d:      color(BLACK);
            rectf(8.1, 3.05, 8.5, 3.15);
            break;
```

```
/* System Element Switch Connections */

        case SESWCN:       color(BLACK);
                              rectf(8.75, 5.1, 8.85, 5.3);
                              break;

        default:          break;

      } /* end switch */

    closeobj();
   } /* end for */

} /* end makediagram */

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

          CLEANUP_FLOW and CLEANUP_DIAGRAM

      Procedure that changes all objects in system diagram to
      original color.

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

cleanup_flow()
{
   int i;
   for (i = CN0; i <= CN8d; i = i + 1)
   {
      editobj(power[i].objname);
         objreplace(power[i].tagname);
         color(BLACK);
      closeobj();
   }
} /* end cleanup_flow */

cleanup_diagram()
{
   int i;

   for (i = EPSCNTL; i <= SW2_2; i = i + 1)
   {
      editobj(power[i].objname);
         objreplace(power[i].tagname);
         color(BLACK);
      closeobj();
   }
```

```
  for (i = EPSPVNODE; i <= SW22PDCA2NODE; i = i + 1)
  {
    editobj(power[i].objname);
      objreplace(power[i].tagname);
      color(BLUE);
    closeobj();
  }
} /* end cleanup_diagram */

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

                CALL_DIAGRAM

    Procedure that calls all the objects in the system diagram.

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

call_diagram()
{
  int i;

  for (i = 0; i <= 70; i = i + 1)
    callobj(power[i].objname);

} /* end call_diagram */
```

PWRFLOW.C

```
#include "pwrsys.h"

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

                    CHARGE_BATTERY

    Show the flow of power from the photovoltaic equipment to the
    battery.

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

charge_batt()
{
   int flashpoint, i, num, min, max;

   Boolean CANCEL;
   short data;

   min = CN0;
   max = CN1b;
    i =  0;
   num = min;
   flashpoint = 0;

   CANCEL = FALSE;

   while(!CANCEL)
   {

      i = i + 1;

/* Pause briefly, then move "electricity" flash */

      sleep(1);
      {
         i = 0;
    switch(flashpoint)
    {
      case 0:  editobj(power[num].objname);
                 objreplace(power[num].tagname);
                 color(RED);
               closeobj();

               editobj(power[max].objname);
                 objreplace(power[max].tagname);
                 color(BLACK);
               closeobj();
               break;

      case 1:
      case 2:  editobj(power[num].objname);
                 objreplace(power[num].tagname);
                 color(RED);
               closeobj();

               editobj(power[num - 1].objname);
                 objreplace(power[num - 1].tagname);
                 color(BLACK);
               closeobj();
               break;

     default:  break;

    } /* end switch */

    call_diagram();
    swapbuffers();

    flashpoint = (flashpoint + 1) % 3;
    if (num == max)
      num = min;
    else
      num = num + 1;

  } /* end if */


  if (qtest())
  {
    switch(qread(&data))
    {
      case REDRAW:    reshapeviewport();
                break;
```

```
            case LEFTMOUSE:  CANCEL = TRUE;
                             break;

            default:         break;

          }
        }

    } /* end while */

    color(WHITE);
    clear();
    cleanup_flow();
    swapbuffers();

} /* end charge_batt */

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

                    DISCHARGE_BATTERY

    Show the flow of power from the batteries through the rest of
    the system.

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


discharge_batt()
{
    int flashpoint, i, j, num, min, max;

    Boolean CANCEL;
    short data;

    min = CN1b;
    max = CN8d;
     i =  0;
     j =  0;
    num = min;
    flashpoint = 0;

    CANCEL = FALSE;
```

```
while(!CANCEL)
{

  i = i + 1;

  sleep(1);
  {
    i = 0;

    switch(flashpoint)
    {
      case 0:  editobj(power[num].objname);
                 objreplace(power[num].tagname);
                 color(RED);
               closeobj();

               for (j = num + 8; j <= num + 11; j = j + 1)
               {
                 editobj(power[j].objname);
                   objreplace(power[j].tagname);
                   color(BLACK);
                 closeobj();
               }
               break;

      case 1:
      case 2:
      case 3:
      case 4:
      case 5:  editobj(power[num].objname);
                 objreplace(power[num].tagname);
                 color(RED);
               closeobj();

               editobj(power[num - 1].objname);
                 objreplace(power[num - 1].tagname);
                 color(BLACK);
               closeobj();
               break;

      case 6:  for (j = num; j <= num + 1; j = j + 1)
               {
                 editobj(power[j].objname);
                   objreplace(power[j].tagname);
                   color(RED);
                 closeobj();
               }
```

```
                editobj(power[num - 1].objname);
                  objreplace(power[num - 1].tagname);
                  color(BLACK);
                closeobj();
                num = num + 1;
                break;

          case 7:  for (j = num; j <= num + 3; j = j + 1)
                {
                  editobj(power[j].objname);
                    objreplace(power[j].tagname);
                    color(RED);
                  closeobj();
                }
                for (j = num - 2; j <= num - 1; j = j + 1)
                {
                  editobj(power[j].objname);
                    objreplace(power[j].tagname);
                    color(BLACK);
                  closeobj();
                }
                num = num + 3;
                break;

         default:  break;

       } /* end switch */

       call_diagram();
       swapbuffers();

       flashpoint = (flashpoint + 1) % 8;
       if (num == max)
         num = min;
       else
         num = num + 1;

    } /* end if */

    if (qtest())
    {
       switch(qread(&data))
       {
         case REDRAW:    reshapeviewport();
                    break;
```

```
            case LEFTMOUSE:  CANCEL = TRUE;
                     break;

            default:        break;
          }
       }

    } /* end while */

    color(WHITE);
    clear();
    cleanup_flow();
    swapbuffers();

} /* end discharge_batt */
```

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

                USE_PV_ONLY

    Show the flow of power from the photovoltaic equipment
    throughout the rest of the system.

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


use_pv_only()
{
   int flashpoint, i, j, num, min, max;

   Boolean CANCEL;
   short data;

   min = CN0;
   max = CN8d;
    i =  0;
    j =  0;
   num = min;
   flashpoint = 0;

   CANCEL = FALSE;


   while(!CANCEL)
   {

      i = i + 1;

      sleep(1);
      {
         i = 0;

         switch(flashpoint)
         {
            case 0:  editobj(power[num].objname);
                     objreplace(power[num].tagname);
                     color(RED);
                  closeobj();
```

```
          for (j = num + 10; j <= num + 13; j = j + 1)
          {
            editobj(power[j].objname);
               objreplace(power[j].tagname);
               color(BLACK);
            closeobj();
          }
          break;

case 1:
case 3:
case 4:
case 5:
case 6:   editobj(power[num].objname);
            objreplace(power[num].tagname);
            color(RED);
          closeobj();

          editobj(power[num - 1].objname);
            objreplace(power[num - 1].tagname);
            color(BLACK);
          closeobj();
          if (flashpoint == 1)
            num = num +  1;
          break;

case 2:   editobj(power[num].objname);
            objreplace(power[num].tagname);
            color(RED);
          closeobj();
          callobj(power[num].objname);

          editobj(power[num - 2].objname);
            objreplace(power[num - 2].tagname);
            color(BLACK);
          closeobj();
          callobj(power[num - 2].objname);
          break;

case 7:   for (j = num; j <= num + 1; j = j + 1)
          {
            editobj(power[j].objname);
               objreplace(power[j].tagname);
               color(RED);
            closeobj();
          callobj(power[j].objname);
          }
```

```
editobj(power[num - 1].objname);
   objreplace(power[num - 1].tagname);
   color(BLACK);
closeobj();
callobj(power[num - 1].objname);
num = num + 1;
break;
```

```
        case 8:  for (j = num; j <= num + 3; j = j + 1)
                 {
                   editobj(power[j].objname);
                      objreplace(power[j].tagname);
                      color(RED);
                   closeobj();
                 callobj(power[j].objname);
                 }
                 for (j = num - 2; j <= num - 1; j = j + 1)
                 {
                   editobj(power[j].objname);
                      objreplace(power[j].tagname);
                      color(BLACK);
                   closeobj();
                 callobj(power[j].objname);
                 }
                 num = num + 3;
                 break;

        default:  break;

      } /* end switch */

      call_diagram();
      swapbuffers();

      flashpoint = (flashpoint + 1) % 9;
      if (num == max)
         num = min;
      else
         num = num + 1;

   } /* end if */


   if (qtest())
   {
      switch(qread(&data))
      {
        case REDRAW:     reshapeviewport();
                      break;

        case LEFTMOUSE:  CANCEL = TRUE;
                      break;

        default:        break;
      }
   }

} /* end while */
```

```
  color(WHITE);
  clear();
  cleanup_flow();
  swapbuffers();

} /* end use_pv_only */

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

                    USE_PV_AND_BATTERIES

    Show the flow of power from the photovoltaic equipment and the
    batteries throughout the rest of the system.

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


use_pv_batt()
{
   int flashpoint, i, j, num, min, max;

   Boolean CANCEL;
   short data;

   min = CN0;
   max = CN8d;
     i =  0;
     j =  0;
   num = min;
   flashpoint = 0;

   CANCEL = FALSE;


   while(!CANCEL)
   {

      i = i + 1;

      sleep(1);
      {
         i = 0;

         switch(flashpoint)
         {
            case 0:  editobj(power[num].objname);
                     objreplace(power[num].tagname);
                     color(RED);
                  closeobj();
```

```
          callobj(power[num].objname);

          for (j = num + 10; j <= num + 13; j = j + 1)
          {
            editobj(power[j].objname);
              objreplace(power[j].tagname);
              color(BLACK);
            closeobj();
            callobj(power[j].objname);
          }
          break;

     case 3:
     case 4:
     case 5:
     case 6:  editobj(power[num].objname);
                objreplace(power[num].tagname);
                color(RED);
              closeobj();
              callobj(power[num].objname);

              editobj(power[num - 1].objname);
                objreplace(power[num - 1].tagname);
                color(BLACK);
              closeobj();
              callobj(power[num - 1].objname);
              break;

     case 2:  editobj(power[num].objname);
                objreplace(power[num].tagname);
                color(RED);
              closeobj();
              callobj(power[num].objname);

              for (j = num - 2; j <= num - 1; j = j + 1)
              {
                editobj(power[j].objname);
                  objreplace(power[j].tagname);
                  color(BLACK);
                closeobj();
              callobj(power[num - 2].objname);
              }
              break;
```

```
        case 1:
        case 7:  for (j = num; j <= num + 1; j = j + 1)
                 {
                   editobj(power[j].objname);
                      objreplace(power[j].tagname);
                      color(RED);
                   closeobj();
                   callobj(power[j].objname);
                 }

                 editobj(power[num - 1].objname);
                    objreplace(power[num - 1].tagname);
                    color(BLACK);
                 closeobj();
                 callobj(power[num - 1].objname);
                 num = num + 1;
                 break;

        case 8:  for (j = num; j <= num + 3; j = j + 1)
                 {
                   editobj(power[j].objname);
                      objreplace(power[j].tagname);
                      color(RED);
                   closeobj();
                   callobj(power[j].objname);
                 }
                 for (j = num - 2; j <= num - 1; j = j + 1)
                 {
                   editobj(power[j].objname);
                      objreplace(power[j].tagname);
                      color(BLACK);
                   closeobj();
                   callobj(power[j].objname);
                 }
                 num = num + 3;
                 break;

      default:  break;

    } /* end switch */

    call_diagram();
    swapbuffers();
    flashpoint = (flashpoint + 1) % 9;
    if (num == max)
       num = min;
    else
       num = num + 1;

} /* end if */
```

```
        if (qtest())
        {
          switch(qread(&data))
          {
            case REDRAW:     reshapeviewport();
                             break;

            case LEFTMOUSE:  CANCEL = TRUE;
                             break;

            default:         break;
          }
        }

    } /* end while */

    color(WHITE);
    clear();
    cleanup_flow();
    swapbuffers();

} /* end use_pv_batt */
```

COMPSTAT.C

```
#include "pwrsys.h"

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

                    FAULT ISOLATION

                Shows actual faults in systems

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

faultisolation(num)
   int num;
{
   int type;

   type = 0;

   componentstatus(type, num);
}

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

                    STATUS PREDICTION

                Shows predicted faults in systems

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

statusprediction(num)
   int num;
{
   int type;

   type = 1;

   componentstatus(type, num);
}
```

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

                    COMPSTAT

    Checks the status of individual components in the diagram.

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

componentstatus(type, num)
   int type, num;
{
   int i, index;
   Boolean CANCEL;
   short data;

   commint = type;

   write_integer(&remotemachine, &commint);

   if (sender_is_free(&remotemachine))
   {
      if((num >= 0) && (num <= 50))
      {
         commint = num;
      }
      else
      {
         commint = CHECKSYSTEM;
      }

      write_integer(&remotemachine, &commint);
      process_message();
   }

   CANCEL = FALSE;

   while (!CANCEL)
   {

      if (qtest())
      {
         switch(qread(&data))
         {
            case REDRAW:     reshapeviewport();
                         break;
```

```
                case LEFTMOUSE:  CANCEL = TRUE;
                         break;

                default:        break;
             }
          }

       } /* end while */


       system("clear\n");

       color(WHITE);
       clear();
       cleanup_diagram();
       swapbuffers();
    } /* end componentstatus */

    process_message()
    {
       Boolean done;
       char compcolor, mess_pointer[80], commessage[80];

      while (!done)
      {

       while (!receiver_has_data(&remotemachine));

       read_characters(&remotemachine, commbuffer,
          number_received(&remotemachine));

       while (!receiver_has_data(&remotemachine));

       read_integer(&remotemachine, commint);

       compcolor = commbuffer[0];

       if (compcolor == 'G' || compcolor == 'Y' || compcolor == 'R')
       {
         editobj(power[commint].objname);
            objreplace(power[commint].tagname);
            switch(compcolor)
            {
              case 'G':  color(GREEN);
                     break;

              case 'Y':  color(FLASHBLACKY);
                     break;
            case 'R':  color(FLASHBLACKR);
                       break;

            default:   break;
          }
        closeobj();

        strcpy(mess_pointer, strchr(commbuffer, '-'));
        strcpy(commessage, mess_pointer);
      }

      else if (commint = FINISHED_PROCESSING)
      {
        done = FALSE;
        strcpy(commessage, commbuffer);
      }

      color(WHITE);
      clear();
      swapbuffers();
      clear();
      swapbuffers();
      call_diagram();
      color(BLACK);
      cmov2(2.5, 1.0);
      charstr(commessage);
      sleep(1);
    }
  } /* end process_message */
```

LIST OF REFERENCES

1. Erickson, W., and others, "NASA Systems Autonomy Demonstration Program: A Step Toward Space Station Automation," *SPIE Space Station Automation II*, v. 729, 1986.

2. *User's Guide to Symbolics Computers*, Symbolics, Inc., July 1986.

3. Wilensky, R., *Common LISPcraft*, W. W. Norton & Company, Inc., 1986.

4. *IntelliCorp KEE Software Development System User's Manual*, IntelliCorp, 1986.

5. Winston, P. H., "Artificial Intelligence: A Perspective," *AI in the 1980s and Beyond*, The MIT Press, 1987.

6. Erickson, W., and Nienart, J., *MTK Reference Manual and User Guide (draft)*, 10 April 1988.

7. Kernighan, B., and Ritchie, D., *The C Programming Language*, Prentice-Hall, Inc., 1978.

8. *IRIS User's Guide: Volume 1 Graphics Programming*, Silicon Graphics, Inc., 1986.

9. Barrow, T., *Distributed Computer Communications in Support of Real- Time Visual Simulations*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

10. Barrow, T., "Inter-computer Communication Package," unpublished paper, Naval Postgraduate School, Monterey, California, May 1988.

11. Bromley, H., and Lamson, R., *LISP Lore: A Guide to Programming the LISP Machine*, Kluwer Academic Publishers, 1987.

12. Martin Marietta Denver Aerospace Task I Study Report MCR-86-583, *Space Station Automation of Common Module Power Management and Distribution*, by Miller, W., and others, July 1986.

BIBLIOGRAPHY

Barrow, T., *Distributed Computer Communications in Support of Real- Time Visual Simulations*, Master's Thesis, Naval Postgraduate School, Monterey, California, June 1988.

Barrow, T., "Inter-computer Communication Package," unpublished paper, Naval Postgraduate School, Monterey, California, May 1988.

Bromley, H., and Lamson, R., *LISP Lore: A Guide to Programming the LISP Machine*, Kluwer Academic Publishers, 1987.

Erickson, W., and Nienart, J., *MTK Reference Manual and User Guide (draft)*, 10 April 1988.

Erickson, W., and others, "NASA Systems Autonomy Demonstration Program: A Step Toward Space Station Automation," *SPIE Space Station Automation II*, v. 729, 1986.

IntelliCorp KEE Software Development System User's Manual, IntelliCorp, 1986.

IRIS User's Guide: Volume 1 Graphics Programming, Silicon Graphics, Inc., 1986.

Kernighan, B., and Ritchie, D., *The C Programming Language*, Prentice-Hall, Inc., 1978.

Martin Marietta Denver Aerospace Task I Study Report MCR-86-583, *Space Station Automation of Common Module Power Management and Distribution*, by Miller, W., and others, July 1986.

User's Guide to Symbolics Computers, Symbolics, Inc., July 1986.

Wilensky, R., *Common LISPcraft*, W. W. Norton & Company, Inc., 1986.

Winston, P. H., "Artificial Intelligence: A Perspective," *AI in the 1980s and Beyond*, The MIT Press, 1987.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Commander Naval Space Command Attn: Code N3 Dahlgren, VA 22448	1
4.	Commander United States Space Command Attn: Technical Library Peterson AFB, CO 80914	1
5.	Navy Space System Division Chief of Naval Operations (OP-943) Washington, DC 20305-2000	1
6.	Space Systems Academic Group Attn: Prof. Panholzer (Code 72) Naval Postgraduate School Monterey, CA 93943	1
7.	Mr. Henry Lum MS244-7 NASA Ames Research Center Moffett Field, CA 94035	2
8.	Ms. Carla Wong MS244-18 NASA Ames Research Center Moffett Field, CA 94035	2
9.	Mr. William Erickson MS244-18 NASA Ames Research Center Moffett Field, CA 94035	2

10.	Department of Computer Science ATTN: Dr. Robert McGhee (Code 52MZ) Naval Postgraduate School Monterey, CA 93943-5100	5
11.	Superintendent, Code 74 ATTN: Lois Brunner Naval Postgraduate School Monterey, CA 93943-5000	3
12.	Department of Computer Science ATTN: Dr. Michael Zyda (Code 52ZK) Naval Postgraduate School Monterey, CA 93943-5100	2
13.	Superintendent, Code 39 ATTN: LTC Linda Cromback Naval Postgraduate School Monterey, CA 93943-5000	1
14.	LT Kevin Scott Naval Postgraduate School Hermann Hall SMC #2847 Monterey, CA 93943	1
15.	LCDR Gracie Thompson TAMPA MEPS 144 1st Avenue South Room #315 St. Petersburg, FL 33701	1
16.	LT Gina L. Hester 1408 Chesterbrook Drive Virginia Beach, VA 23464	2
17.	LT Beth Allinder 300 Glenwood Circle Apt. #298 Monterey, CA 93940	1