NASA Contractor Report 4260

Space Station Automation of Common Module Power Management and Distribution

W. Miller, E. Jones, B. Ashworth,
J. Riedesel, C. Myers, K. Freeman,
D. Steele, R. Palmer, R. Walsh,
J. Gohring, D. Pottruff, J. Tietz,
and D. Britt

CONTRACT NAS8-36433
NOVEMBER 1989

NASA Contractor Report 4260

Space Station Automation of Common Module Power Management and Distribution

W. Miller, E. Jones, B. Ashworth,
J. Riedesel, C. Myers, K. Freeman,
D. Steele, R. Palmer, R. Walsh,
J. Gohring, D. Pottruff, J. Tietz,
and D. Britt
Martin Marietta Aerospace
Denver Astronautics Group
Denver, Colorado

Prepared for
George C. Marshall Space Flight Center
under Contract NAS8-36433

National Aeronautics and
Space Administration
Office of Management
Scientific and Technical
Information Division

1989

This report was prepared by Martin Marietta Denver Astronautics Group for the National Aeronautics and Space Administration, George C. Marshall Space Flight Center (NASA/MSFC), in response to Contract, NAS8-36433, and is submitted as the Interim Final Report, as specified in the contract data requirements list. In particular, the work was performed for the Electrical Power Branch at NASA/MSFC.

Page

1.0 INTRODUCTION 1-1

1.1 Executive Overview 1-3

2.0 BREADBOARD 2-1

3.0 TASK I 3-1

4.0 TASK II 4-1

5.0 TASK III 5-1

5.1 Introduction 5-1

5.2 Overall Breadboard Configuration 5-2

5.2.1 Development Chronology 5-2

5.2.2 Final Configuration 5-4

5.2.3 Functional Interfaces & Dataflows (Scheduler, FRAMES, CAC, LLPs) 5-8

5.2.4 Grounding Scheme 5-53

5.3 Breadboard Theory of Operation 5-54

5.4 Resource Scheduling 5-62

5.4.1 Theory of Operation 5-62

5.4.2 Implementation 5-66

5.4.3 MAESTRO User Interface 5-78

5.5 FELES 5-85

5.5.1 Theory 5-85

5.5.2 Implementation 5-85

5.5.3 User Interface 5-86

5.6 LPLMS 5-89

5.6.1 Theory 5-89

5.6.2 Implementation 5-90

5.6.3 User Interface 5-90

5.7 Fault Management and Recovery 5-94

5.7.1 Lowest Level Processors 5-95

5.7.2 Fault Recovery and Management Expert System 5-96

5.7.2.1 Theory of Operation 5-97

5.7.2.2 Implementation 5-99

PRECEDING PAGE BLANK NOT FILMED

Page

5.7.2.3 User Interface 5-109
5.8 Power Distribution Management 5-112
5.8.1 Power Distribution Control Unit 5-112
5.8.2 Load Center 5-112
5.8.3 Switchgear 5-113
5.8.4 Automation 5-116
5.8.4.1 Hardware 5-116
5.8.4.2 Interfaces 5-117
5.8.4.3 Software 5-117
5.8.5 Redundancy Management 5-120
5.9 Operation Considerations 5-120
5.10 Breadboard Timing Considerations 5-121
5.11 Manual Override 5-123
5.11.1 Operation 5-123
5.11.2 Implementation 5-123
5.11.3 User Interface 5-124
5.12 Test Plan 5-127
6.0 TASK IV 6-1
7.0 SUMMARY 7-1
7.1 What's Next 7-3
8.0 REFERENCES 8-1
9.0 GLOSSARY 9-1
10.0 APPENDIX I: Task I Data I-1
11.0 APPENDIX II: Task II Data II-1
12.0 APPENDIX III: Task IV Data III-1
13.0 APPENDIX IV: Function Definitions IV-1
14.0 APPENDIX V: Crew Interfaces V-1
15.0 APPENDIX VI: Symbolics Environment VI-1
16.0 APPENDIX VII: ICD VII-1
17.0 APPENDIX VIII: Breadboard Usage VIII-1
18.0 APPENDIX IX: VCLRs IX-1

Page

19.0 APPENDIX X: SSM/PMAD Expository and Activity Plan X-1
20.0 APPENDIX XI: Fault and Symptom Matrix XI-1

Page

1.1.1-1 Automation Architecture Breakout 1-4
1.1.1-2 The SSM/PMAD Hardware Architecture Overview 1-4
1.1.1-3 User Access of System Functionality 1-5
1.1.2-1 Software to Hardware Allocation 1-7
1.1.3-1 SSM/PMAD System Initialization 1-7
1.1.3-2 SSM/PMAD Autonomous Mode Operation 1-8
1.1.3-3 SSM/PMAD Manual Mode Operation 1-9
2.1-1 SSM/PMAD Breadboard Diagram 2-2
2.1-2 LLP/Switchgear Hardware Diagram 2-5
2.3-1 SSM/PMAD Breadboard User Interfaces 2-9
5.2.1-1 SSM/PMAD Development Schedule 5-4
5.2.2.1-1 SSM/PMAD Automation Hardware Configuration 5-6
5.2.2.2-1 SSM/PMAD Automation Software Configuration 5-8
5.2.3.1-1 SSM/PMAD System Overview 5-10
5.2.3.2-1 Transaction Format 5-13
5.2.3.4.1-1 Initialization Dataflow Part 1 5-47
5.2.3.4.1-2 Initialization Dataflow Part 2 5-48
5.2.3.4.1-3 Initialization Dataflow Part 3 5-49
5.2.3.4.2-1 Normal Operation Dataflow 5-50
5.2.3.4.3-1 Source Power Contingency Dataflow 5-51
5.2.3.4.3-2 Power System Contingency Dataflow 5-52
5.3.3-1 The SSM/PMAD Operational Flow 5-60
5.4.2.2-1 SSM/PMAD Breadboard Diagram 5-67

Page

5.4.2.2-2 Symbolics Interface Processing Architecture 5-69

5.4.2.4.1-1 Contingency 1 5-73

5.4.2.4.1-2 Contingency 2 5-74

5.4.2.4.1-3 Contingency 3 5-75

5.4.3.1-1 Equipment Editor Screen 5-79

5.4.3.1-2 Equipment Editor Help Screen 5-80

5.4.3.2-1 Activity Editor Screen 5-81

5.4.3.2-2 Activity Editor Help Screen 5-82

5.4.3.3-1 Scheduler Screen 5-83

5.4.3.3-2 Scheduler Help Screen 5-84

5.5.3-1 Front End Load Enable Scheduler Screen 5-87

5.5.3-2 Front End Load Enable Scheduler Help Screen 5-88

5.6.3-1 Load Priority List Management Screen 5-91

5.6.3-2 Load Priority List Management Help Screen 5-92

5.7.2.2.5-1 FRAMES Software Configuration 5-107

5.7.2.3-1 FRAMES User Interface 5-109

5.12.4.1-1 SSM/PMAD Communications Architecture 5-128

5.12.4.1-2 Breadboard Block Diagram 5-129

5.12.5.10.1-1 Test Plan Switch Configuration 5-149

15.1-1 Symbolics Interface Processing Architecture VI-1

15.4-1 Example Symbolics Screen VI-10

15.4.1-1 Console Screen VI-12

15.4.1-2 Console Help Screen VI-13

Page

15.4.2-1 FELES Screen VI-15

15.4.2-2 FELES Help Screen VI-16

15.4.3-1 LPLMS Screen VI-18

15.4.3-2 LPLMS Help Screen VI-19

15.4.4-1 Scheduler Screen VI-20

15.4.4-2 Scheduler Help Screen VI-21

15.4.5-1 Activity Editor Screen VI-22

15.4.5-2 Activity Editor Help Screen VI-23

15.4.6-1 Equipment Editor Screen VI-24

15.4.6-2 Equipment Editor Help Screen VI-25

15.4.7-1 Resource Manager Screen VI-27

15.4.7-2 Resource Manager Screen VI-28

15.4.7-3 Resource Manager Help Screen VI-29

15.4.8-1 Communications Screen VI-31

15.4.8-2 Communications Help Screen VI-32

Page

5.3.2-1 Breadboard Functional Content 5-56

A/D	Analog to Digital Conversion
AC	Alternating Current
ACM/PMAD	Automation of Common Module Power Management and Distribution
AI	Artificial Intelligence
BLES	Baseline Load Enable Schedule
CAC	Communications Algorithmic Controller
CAS	Communication and Algorithmic Software
CLOS	Common Lisp Object System
DC	Direct Current
ECLSS	Environmental Control Life Support System
EMI	Electro-Magnetic Interference
EPLD	Erasable Programmable Logic Device
FELES	Front End Load Enable Scheduler
FRAMES	Fault Recovery and Management Expert System
GC	Generic Controller
H/W	Hardware
ICD	Interface Control Document
JSC	Johnson Space Center
KHz	KiloHertz
KVA	KiloVolt Amps
L-R	Inductive-Resistive
LC	Load Center
LES	Load Enable Scheduler
LISP	List Processing
LLF	Lowest Level Function
LLP	Lowest Level Processor
LPL	Load Priority List
LPLMS	Load Priority List Management System
MMAG	Martin Marietta Astronautics Group
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
NDA	Node Distribution Assembly

OMS	Operational Management System
PCL	Portable Common Loops
PCU	Power Control Unit
PDCU	Power Distribution Control Unit
PLES	Preliminary Load Enable Schedule
PPDA	Primary Power Distribution Assembly
RBI	Remote Bus Isolator
RCCB	Remote Control Circuit Breaker
RMS	Root Mean Square
RPC	Remote Power Controller
SADP	Systems Autonomy Demonstration Program
SDA	Subsystem Distributor Assembly
SI	Symbolics Interface
SIC	Switchgear Interface Controller
SPST	Single Pole Single Throw
SSM	Space Station Module
SSM/PMAD	Space Station Module Power Management and Distribution
SSS	Supervisor Subsystem Simulator
S/W	Software
TCP/IP	Transport Control Protocol / Internet Protocol
UI	User Interface
VCLR	Visual Control Logic Representation

1.0 INTRODUCTION

This document reports on the Contract, NAS8-36433, and is in response to the work which was performed in developing and delivering the automation software of the Space Station Module Power Management And Distribution (SSM/PMAD) system. The work was done by Martin Marietta Corporation, Denver Astronautics Group for the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, in support of the Electrical Power Branch's development of an advanced automation SSM/PMAD system test bed. The NASA Contracting Officer's Technical Representation for the Contract is Mr. David J. Weeks. Martin Marietta is reporting on all Tasks included within NAS8-36433. These consist of Task I, Task II, Task III, and Task IV.

Task I - "Common Module Power Management and Distribution (CM/PMAD) System Automation Plan Definition". This task forged an overall plan for attacking the problem of automating the power system breadboard. Various power hardware configurations were also analyzed as to performance and applicability to the problem. This task was completed in July of 1986 and the Task I Report is included in this document.

Task II - "Definition of Hardware and Software Elements of Automation". This task defined the various knowledge based and deterministic algorithms to be used within the SSM/PMAD. An overall implementation plan utilizing iterative refinement was established for the SSM/PMAD software. This task was completed in August of 1987 and the Task II Report is included in this document.

Task III - "Implementation/Verification of CM/PMAD Automation Approach in MSFC Breadboard". This task consists of obtaining the SSM/PMAD hardware, building the SSM/PMAD software, establishing and documenting operational plans and procedures, and integrating system components. Delivery and support of development components to NASA/MSFC were also performed within this task. Delivery of the initial Task III system was completed in December of 1988 and support is presently continuing. Task III is reported on in significant detail within Section 5.0 of this document.

Task IV - "Definition and Development of the MSFC CM/PMAD Breadboard Host Computer Environment". This task examined the need for a central host capability within the CM/PMAD. Requirements for such a machine were established and recommendations were made. This task was completed in October of 1985 and the Task IV Report is included in Appendix III of this document.

The overall document is a history and description of the work performed in defining, designing, and developing the SSM/PMAD breadboard, complete with software. The hardware architecture is described along with its associated deterministic software architecture, the AI systems architecture, and the methodology and process of the overall system integration.

System development was performed on a Symbolics 3640 utilizing ZetaLISP, a Xerox 1186 utilizing InterLISP D and the Common Lisp Object System (CLOS), a Motorola VME/10 utilizing PASCAL and Assembly Language, and Motorola 107 board level processors utilizing PASCAL and Assembly Language. The system delivery environment was the same as the that used in development with the exception of the Symbolics system which was a 3620 D.

1.1 Executive Overview

The purpose of this project is to automate a breadboard level Power Management and Distribution (PMAD) system which possesses many functional characteristics of a specified Space Station power system. The automation system was built upon a 20 kHz ac[1] source with redundancy of the power buses. There are two power distribution control units which furnish power to six load centers which in turn enable load circuits based upon a system generated schedule. This report documents the progress in building this specified autonomous system.

The resulting system possesses the capability to perform diagnosis whenever a distribution fault is encountered. The system autonomously reconfigures its operation during run-time to reschedule activities around the fault, rather than performing a system halt. The system functionality is previewed in the subsections of this overview.

Automation of Space Station Common Module Power Management and Distribution (SSM/PMAD) was accomplished by segmenting the complete task into the following four independent tasks.

Task I - Develop a detailed approach for PMAD automation.

Task II - Define the software and hardware elements of automation.

Task III - Develop the automation system for the PMAD breadboard.

Task IV - Select an appropriate host processing environment.

Early planning activity (prior to 1985), by Mr. David J. Weeks of NASA/MSFC, provided the capability to perform Task IV initially. This was done in order to establish an appropriate platform upon which to build the system.

1.1.1 System Architecture

The result of the initial defining work was to separate items as needed into hardware and software elements. Figure 1.1.1-1 shows the highest level breakout for the two areas.

SSM/PMAD

HARDWARE	***SOFTWARE***
- Switchgear	***- Complex Functions (AI)***
- Switch Control	***- Conventional***
- Analog to Digital	***- Control***
- Automation	***- Lowest Level***

FIGURE 1.1.1-1 Automation Architecture Breakout

The SSM/PMAD breadboard hardware consists of two distinct elements: the power control hardware through which current flows to power the target loads, and the automation hardware which is made up of computers and process oriented circuit cards. This is shown in Figure 1.1.1-2.

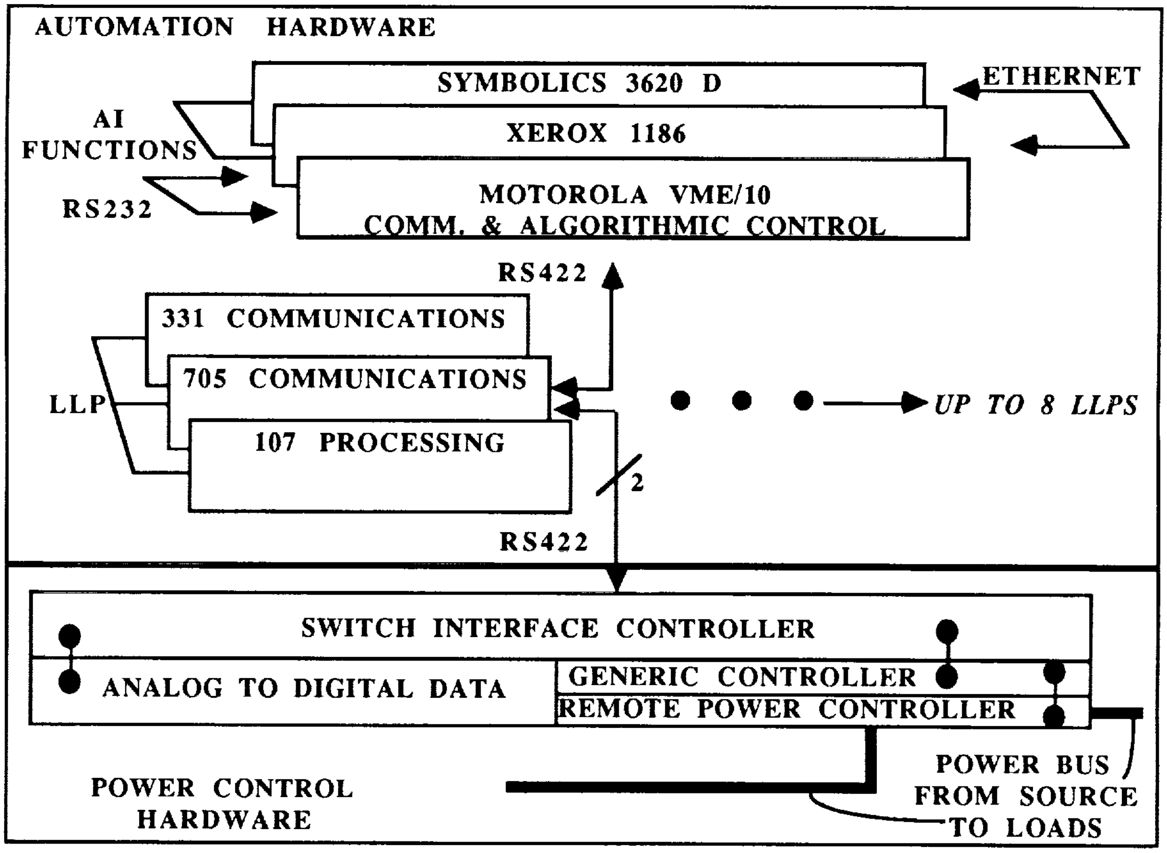


FIGURE 1.1.1-2 The SSM/PMAD Hardware Architecture Overview

The power control hardware consists of analog and digital level hardware units and is considered as part of the power system topology hardware. The automation hardware is part of the automation system and provides the interface between the user, the autonomous functions (FRAMES, FELES, LPLMS, and MAESTRO), the Communications and Algorithmic Controller (CAC), the Lower Level Functions (LLFs) that exist at the Lower Level Processors (LLPs), and the actual hardware control. This is depicted in Figure 1.1.1-3.

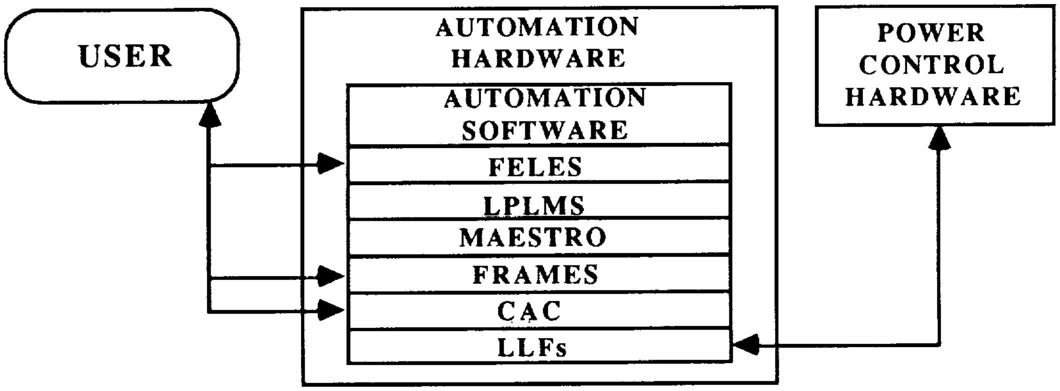


FIGURE 1.1.1-3 User Access of System Functionality

As can be seen in Figure 1.1.1-3, the SSM/PMAD is a multi-agent distributed system. The distribution of software functions will be described in the next section.

1.1.2 Content and Distribution of Software Functionality

The Front End Load Enable Scheduler (FELES) provides the user access to the scheduling environment, MAESTRO, and handles returning information from FRAMES. Whenever run-time rescheduling activities are required, FELES initiates MAESTRO and LPLMS activity with the appropriate update information.

The Load Priority List Management System (LPLMS) handles initializing and changing priorities of loads. Based upon heuristics, initial priorities for powered loads will change with occurrence of various system events such as changing availability of system power, passage of time, emergencies, and others. These priorities must be managed and allocated in proper ways to assure dependable system performance, preventing shedding of critical or higher priority loads, and LPLMS accomplishes this needed function.

MAESTRO is a load scheduling function. It contains basic model knowledge of the overall power system and the required heuristics to ensure correct allocation of resources. The result of MAESTRO's work is the production of a Load Enable Schedule (LES) to be carried out by the Lowest Level Processors (LLPs).

The Fault Recovery and Management Expert System (FRAMES) is the backbone of the run-time environment for the LES. FRAMES diagnoses faults and commands the overall system whenever faults or anomalies occur. FRAMES maintains the system status and provides the autonomous run-time user-interface. FRAMES understands the function and roles of all the operating agents within the SSM/PMAD. In total, FRAMES functions as a watch-dog over the entire power distribution environment and assumes management for the environment whenever faults or anomalies are detected.

The CAC is the central communication facility for tying the higher level automation hardware and the LLPs together. The various functions which exist on the CAC are bundled to form the Communications and Algorithmic Software (CAS). The primary responsibilities of the CAS are to sort and deliver the LES into its appropriate subcomponent representations for execution by the LLPs, and to stage and deliver data between the Xerox and the LLPs. It also contains the manual mode operations interface.

The Lower Level Functions (LLFs) perform algorithmic management of the LES. They also contain a lower level segment of the FRAMES diagnosis activity which provides rapid limit checking and initial levels of fault condition pattern matching.

The allocation of these software entities to the appropriate hardware is shown in Figure 1.1.2-1.

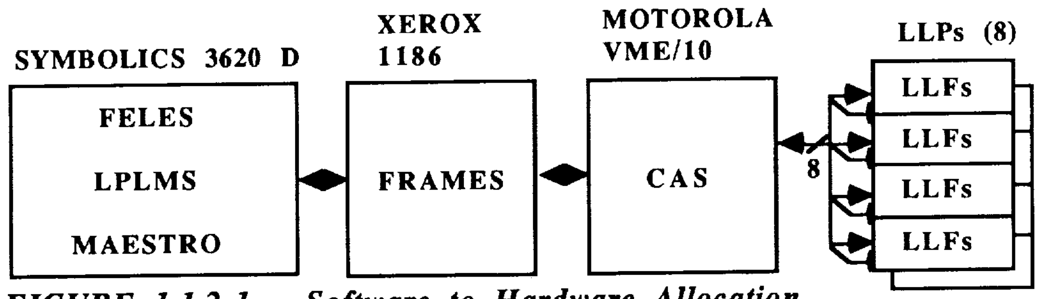
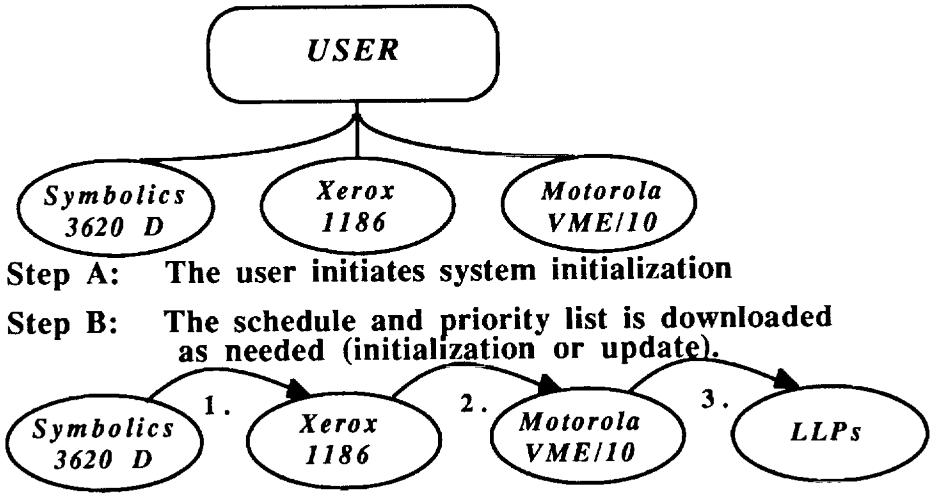


FIGURE 1.1.2-1 Software to Hardware Allocation

1.1.3 SSM/PMAD Functionality

SSM/PMAD system capability, resulting from the architecture, allocation, and design of the previously described entities, achieves the goals which were originally set out. The system activities therefore, rest on three basic operations.

First, the system must be initialized by the breadboard user. This operation is depicted in Figure 1.1.3-1.



Step A: The user initiates system initialization

Step B: The schedule and priority list is downloaded as needed (initialization or update).

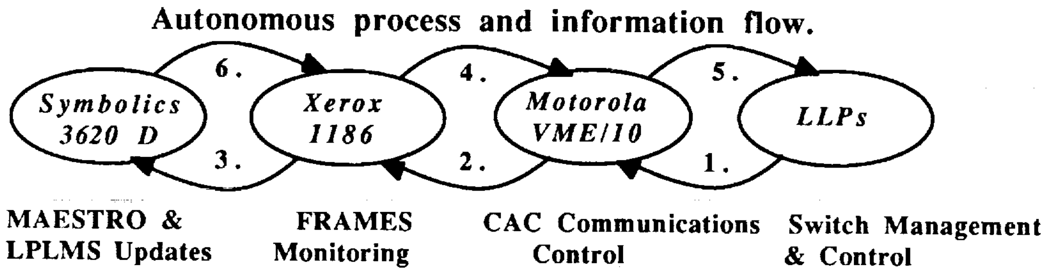
1. **The initial schedule events and priority lists are sent to FRAMES from MAESTRO & LPLMS**
2. **The schedules and priorities are transmitted complete to the CAC.**
3. **The CAC distributes schedules appropriately to the LLPs.**

FIGURE 1.1.3-1 SSM/PMAD System Initialization

Initialization provides the needed system synchronization to achieve start-up conditions. For instance, some notion of an initial set of loads with specified priorities must exist for the system to start. However, the schedule is produced automatically, giving the breadboard operator relief from extensive planning activities.

After system initialization, the SSM/PMAD system can attain two levels or modes of operation. These are autonomous mode and manual mode.

The autonomous mode operation engages all hardware and software entities. If a fault occurs in this mode, a series of autonomous activities take place, performing diagnosis, rescheduling, and reconfiguration operations without operator intervention. The actions of the autonomous mode operation are shown in Figure 1.1.3-2.

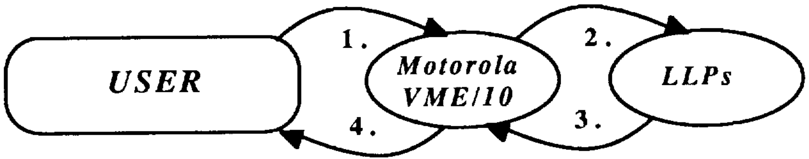


1. The LLPs send up available switch state information.
2. The CAC buffers information to FRAMES.
3. FRAMES sends fault and utilization information to FELES.
4. FRAMES requests further information or switch commanding in the event of a known or suspected fault (performing diagnosis).
5. The CAC distributes commands, schedules, priority lists, and upper level requests to the appropriate LLPs.
6. New schedules and priority lists are made available to FRAMES.

Note: Actions 4 & 5 take place only when the Xerox needs information. Action 6 occurs only if a new contingency schedule or update priority list is available.

FIGURE 1.1.3-2 SSM/PMAD Autonomous Mode Operation

Manual mode operation allows the user to seize total control of the breadboard. When this happens, all higher level AI functions cease operation within the system. This activity is shown in Figure 1.1.3-3.



1. **User requests switch commanding or data.**
2. **The CAC distributes the commands and requests to the LLPs.**
3. **The LLPs send up available switch state information.**
4. **The user requested information is presented.**

FIGURE 1.1.3-3 SSM/PMAD Manual Mode Operation

1.1.4 Recap

In achieving system success, the SSM/PMAD was developed by defining achievable tasks (I through IV) and by performing the crucial initial planning activities which provided allocation of goals.

Martin Marietta has now delivered an initial working SSM/PMAD to NASA/MSFC and is continuing work to provide follow-on capabilities in a natural growth path for the overall system. Changes have been identified which make this growth possible. Please refer to "What's Next" in the Summary Section of this document for a listing of suggested changes.

Operation of the current system provides for both manual and autonomous level activities. Under autonomous conditions, the operator may simply observe the system following initialization. The user-interfaces are always available to query. Under manual conditions, the operator possesses sole responsibility for system activities as the AI systems have been removed from operational execution.

[1] On December 14, 1988 a NASA Change Request specifying 120 V dc source power was put into effect.

2.0 BREADBOARD

The breadboard used in the SSM/PMAD system consists of two parts.

1) The power hardware and switchgear;
2) The automation and control.

The power hardware and switchgear part was supplied under another contract, NAS8-36583, and will be reported on in a separate final report, report number MCR-89-524. This report will focus almost exclusively upon the automation and control portion of the breadboard, which is known as the Automation of the Space Station Common Module Power Management and Distribution system, or simply the SSM/PMAD. Descriptive detail of the hardware will be provided here for the sake of clarity. All components discussed in this chapter are described from an introductory viewpoint. More detailed descriptions are provided in later sections of this document.

Breadboard design for the SSM/PMAD automation and control contained both hardware and software. The architecture for the system focuses on hardware modularization within a functional decomposition view. That is to say, hardware processors within the SSM/PMAD do not share tasks. Each stands independently. The software and algorithms however, are not so clearly segregated. For instance, a schedule request from the user interface is routed through at least four hardware processing environments before it is completed. As well, diagnosis activities within the system utilize the services of more than one computational and control engine before completion. Therefore, the system (breadboard) will be described from three different views.

2.1 The Hardware View

In Figure 2.1-1 the hardware for the SSM/PMAD is shown along with the component interconnections. Functionally, the hardware components from the top down are as follows:

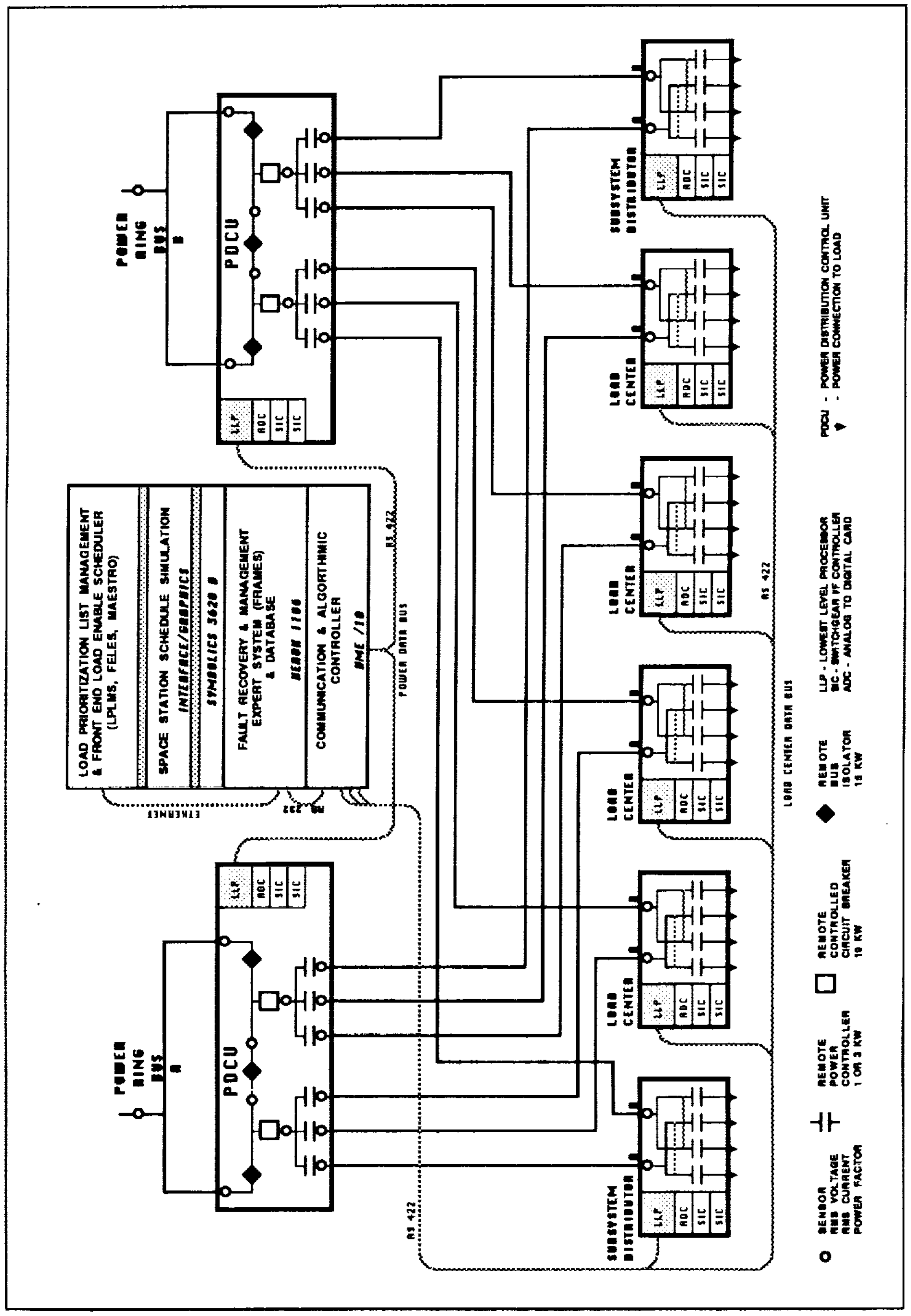

POWER RING BUS B
PDCU
LLP
ADC
SIC
SIC
LOAD PRIORITIZATION LIST MANAGEMENT & FRONT END LOAD ENABLE SCHEDULER (LPLMS, FELES, MAESTRO)
SPACE STATION SCHEDULE SIMULATION
INTERFACE/GRAPHICS
SYMBOLICS 3620 #
FAULT RECOVERY & MANAGEMENT EXPERT SYSTEM (FRAMES) & DATABASE
XEROX 1186
COMMUNICATION & ALGORTHMIC CONTROLLER
VME /10
ETHERNET
RS 232
RS 422
POWER DATA BUS
POWER RING BUS A
SUBSYSTEM DISTRIBUTOR
LOAD CENTER
LOAD CENTER
LOAD CENTER
LOAD CENTER
SUBSYSTEM DISTRIBUTOR
RS 422
LOAD CENTER DATA BUS
SENSOR RMS VOLTAGE RMS CURRENT POWER FACTOR
REMOTE POWER CONTROLLER 1 OR 3 KW
REMOTE CONTROLLED CIRCUIT BREAKER 10 KW
REMOTE BUS ISOLATOR 15 KW
LLP - LOWEST LEVEL PROCESSOR
SIC - SWITCHGEAR I/F CONTROLLER
ADC - ANALOG TO DIGITAL CARD
PDCU - POWER DISTRIBUTION CONTROL UNIT
- POWER CONNECTION TO LOAD

FIGURE 2.1-1 SSM/PMAD Breadboard Diagram.

ORIGINAL PAGE IS OF POOR QUALITY

1) A Symbolics 3620D computer utilizing Symbolics' ZetaLISP computer language;
2) A Xerox 1186 computer utilizing InterLISP D and Common Lisp Object System (CLOS) languages;
3) A Motorola VME/10 computer utilizing Pascal and Assembler languages;
4) Motorola 107 (68010) board level processors at the LLPs utilizing Pascal and Assembler languages.

Communication occurs as a rank ordered hierarchy with each level communicating up only one level and down only one level. This means communications from the Symbolics 3620 D only takes place to the Xerox 1186. In turn, the Xerox 1186 communicates to the Symbolics 3620 D going up and down only to the Motorola VME/10. The Motorola VME/10 communicates up to the Xerox 1186 and down to the Lower Level Processors (LLPs). The LLPs communicate up to the Motorola VME/10 and down to the Switch Interface Controllers (SICs). The SICs belong to the hardware contract for the SSM/PMAD, so the final report for that contract should be referenced for further detail. The rank ordering for the communications hierarchy may appear to be violated if the user interfaces at the Symbolics 3620 D, Xerox 1186, and Motorola VME/10 are considered as a level of communications activity. However, the system was defined as an automated system. And, using that definition, only one user interface is needed at run-time, that being the one at the Symbolics 3620 D.

Commands and data are passed back and forth among the various hardware components as needed. Whenever a schedule is passed down (the passage transcends the four levels of the Symbolics 3620 D, the Xerox 1186, the Motorola VME/10, and finally, the LLP) it is made available to the appropriate LLPs for execution. In turn, whenever relevant data are available from the LLPs, the necessary hardware entity (the Motorola VME/10) is selected for receipt of the data, so that it in turn can make the data available to the Xerox 1186, and if needed, appropriate data are then sent on to the Symbolics 3620 D. This up and down transfer of data provides the components at each level with the information necessary for their operation.

In addition, the LLPs command and receive information from the switchgear. The LLP may request data from the SIC card regarding the status of switches and sensors. Furthermore, the LLP may command the SIC to open or close any switch. Upon receiving a command from the LLP, the SIC determines where it must send or acquire data to fulfill the request. If the SIC must command a switch on or off, the appropriate Generic Controller (GC) card is so informed and the command is executed. If the SIC requires data from a switch, the appropriate GC card's enable line is asserted so it may send data. If the SIC requires data from the sensors, it accesses the Analog to Digital card directly and acquires the digitized sensor words. When the SIC finishes processing the LLP command, it issues a response. This hardware architecture is summarized in Figure 2.1-2.

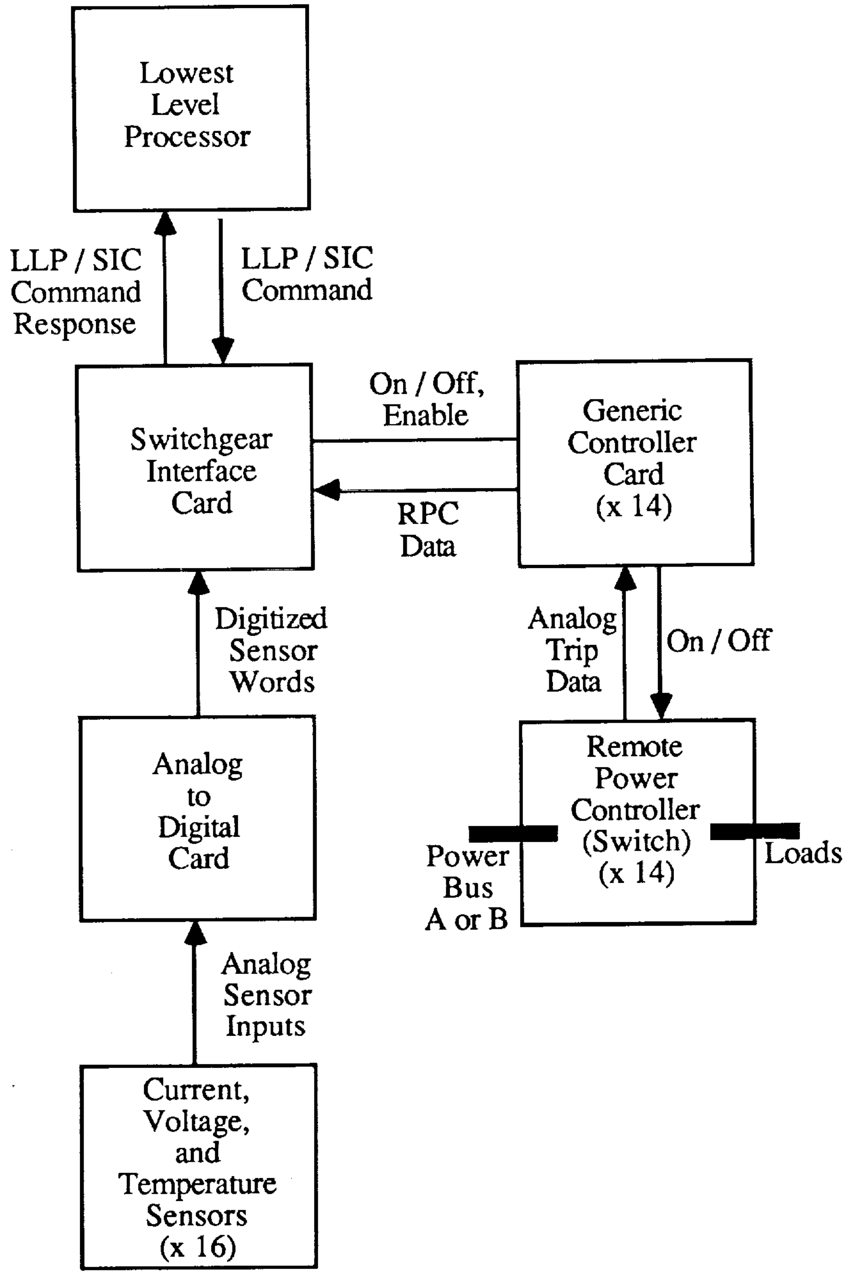


Figure 2.1-2 LLP/Switchgear Hardware Diagram

2.2 The Software View

The software view is functionally composed of:

1) A scheduling mechanism, MAESTRO;
2) A front end to the scheduling mechanism, FELES;
3) A load priority management function, LPLMS;
4) A fault management and recovery system, FRAMES;
5) Centralized communications and data management algorithms, CAC;
6) Lower level processing, control, and management functions, LLPs.

The functionality of the software spreads out over various elements of the hardware as needed. For example, the FRAMES may request information about voltage or current being used by a particular load at a particular load center, see Figure 2.1-1. Functionally, this request would utilize algorithms located on each computer (e.g., Xerox 1186, CAC, and LLP) necessary to complete the request.

The FELES (FELES will be considered to contain MAESTRO as a foreign operating element necessary to complete the scheduling task for the system user) receives information about loads and initial priorities as supplied from a system user. This task performs the necessary front end operations to the scheduler for the user and constructs the necessary schedules for use by the lower level functions. The FELES (with MAESTRO) exists solely on the Symbolics 3620 D computer.

The LPLMS constructs, maintains, and manages the load priority list. The LPLMS exists solely on the Symbolics 3620 D computer.

The FRAMES performs fault monitoring, fault recovery, and fault management and is an automated process leading to an overall autonomous power management system. The functions of the FRAMES are situated mainly within the Xerox 1186 computer, but some of its elements do exist in each LLP.

The CAC performs communications control and data handling and packaging. Its functions exist primarily at the Motorola VME/10 with some functional elements occurring in the LLPs.

The LLPs contain elements of FRAMES and Motorola VME/10 algorithmic functionality, as well as performing lower level data management and schedule control. LLP defined algorithms exist totally within each LLP.

Each software entity utilizes a specific database(s). The databases exist at the same hardware location with the software entity. In cases where data are to be shared, the software passes that data through a list or other appropriate data structure.

2.3 The User's View

Figure 2.3-1 reflects locations where users interface to the SSM/PMAD. Various activities occur at each of the locations and are representative of the software entities which exist there. During periods of typical system execution, the user's view of the system is:

1) The user-interface at the Symbolics 3620 D provides a facility for data input and scheduling activities, as well as an interrogation mechanism into the Symbolics 3620 D. The user-interface mechanisms on the Symbolics 3620 D consist of an alpha-numeric keyboard, a mouse, and screen displays of icons, multi-level menus, text, and graphics.

2) The user-interface at the Xerox 1186 provides the facility to interrogate FRAMES and to request specific hardware component information. The user-interface mechanisms on the Xerox 1186 are similar to those of the Symbolics 3620 D and consist of an alpha-numeric keyboard, a mouse, and screen displays of icons, multi-level menus, text, and graphics.

3) The user-interface at the Motorola VME/10 provides manual intervention into the system. The user-interface mechanisms provided at the Motorola VME/10 are an alpha-numeric keyboard and screen displays of single-level text menus and text.

All user interfaces are required to start the system and to load all appropriate software at system initialization time. Multi-level menus are item selection structures (usually choices are selected via use of a mouse) which can be linked. For example, selection of an icon or menu item by a user causes another menu to appear on the screen from which a selection must be made to complete the user action. This differs from a single-level menu action where the system displays a menu to obtain data via user interrogation on a menu by menu basis. Single-level menus generally require a single alpha-numeric character input which is read from either the keyboard or screen.

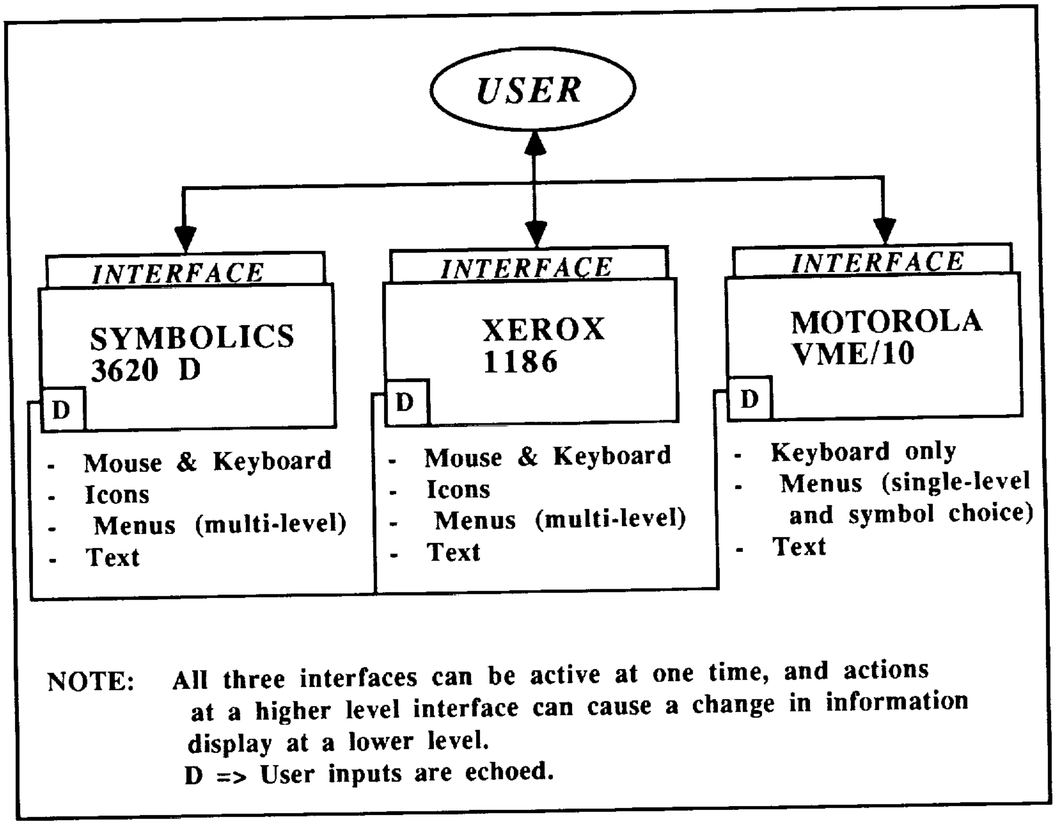

USER
INTERFACE
SYMBOLICS
3620 D
D
- Mouse & Keyboard
- Icons
- Menus (multi-level)
- Text
INTERFACE
XEROX
1186
D
- Mouse & Keyboard
- Icons
- Menus (multi-level)
- Text
INTERFACE
MOTOROLA
VME/10
D
- Keyboard only
- Menus (single-level and symbol choice)
- Text
NOTE: All three interfaces can be active at one time, and actions at a higher level interface can cause a change in information display at a lower level.
D => User inputs are echoed.

FIGURE 2.3-1 SSM/PMAD Breadboard User Interfaces

3.0 TASK I

Task I for the SSM/PMAD was completed in July of 1986. The fundamental activity of Task I was to review the overall process needed to implement an autonomous power system, given the Government provided candidate network topologies, and to define the primary functions needed to provide autonomy of the power system. As well, function partitioning was performed leading to the candidate architecture which was chosen for implementation.

The SSM/PMAD implementation came about as a result of the groundwork performed in Task I. The function partitioning of Task I led to the separation of activities into those of a knowledge based variety and those of a deterministic variety. This provided the basis which was later used in defining and allocating the individual functions FRAMES, FELES, LPLMS, the CAC, and the LLPs.

Review of the Government provided candidate network topologies established the appropriate type of hardware system architecture for control and automation. This, coupled with the appropriate functional decomposition, provided automation hardware selection criteria given that the study for the host computer had been completed in Task IV.

Detailed results of Task I are provided in the Task I Study Report, included as Appendix I within this document.

4.0 TASK II

Task II for the SSM/PMAD was completed in August of 1987. The fundamental activity of Task II was to define software and hardware for the automation system of the SSM/PMAD, given the results of Task I.

The approach to SSM/PMAD automation was defined in Task II. Knowledge base studies were performed and functional decomposition for the deterministic software defined in Task I was initialized. Also, the LISP computer language was chosen for use in developing the SSM/PMAD automation software within the knowledge based activities, while PASCAL was chosen for the host computer.

The hierarchically arranged distributed SSM/PMAD data handling and control system was realized by allocating knowledge based activities to the top layers and deterministic processing to the lower levels.

Some nomenclature has changed since the Task II report. In particular, the primary power distribution assembly (PPDA) is now called the power distribution control unit (PDCU).

The Task II Study Report suggested the use of Causal Reasoning as a primary AI technique used to develop the AI components of the SSM/PMAD. In-line coding of rules was chosen at implementation time during Task III. The reasons were subtle and hidden at the time of the Task II study. In summary, the reasons that causal reasoning has not yet been implemented are 1) a causal model for a developing hardware system usually does not exist at a reasonably high hierarchical component level; 2) performance for a theoretical causal reasoning system is not as yet measurable or predictable; and 3) expert rules at an initial level work well enough to provide system diagnosis within FRAMES. Causal reasoning should, however, be further investigated and used whenever data about the basic system behavior and performance have been accumulated.

Detailed results of Task II are provided in the Task II Study Report, included as Appendix II within this document.

5.0 TASK III

The Task III initial automation system for the SSM/PMAD was delivered to NASA/MSFC by Martin Marietta in December of 1988. The following sections and subsections describe the hardware and software that make up that system.

5.1 Task III Introduction

The purpose of Task III is to define, design, develop, integrate, test, document, and deliver the automation components necessary for initial automation of the SSM/PMAD system. Also, Martin Marietta is performing on-site support of the SSM/PMAD to NASA/MSFC, in particular, to personnel of the Electrical Division's Power Branch in the Information and Electronics Systems Laboratory at NASA/MSFC. NASA/MSFC personnel participated heavily in providing requirements and system level definition, especially in relation to the SSM/PMAD system level activities and definition related to the Space Station Freedom.

Task III reporting includes the areas of:

- Power automation breadboard configuration
- Theory of the breadboard operation
- Resource scheduling
- The Front End Load Enable Scheduler (FELES)
- The Load Priority List Management System (LPLMS)
- The Fault Recovery and Management Expert System (FRAMES)
- Power distribution management
- Breadboard timing considerations
- Manual override capabilities
- A test plan
- A breadboard usage plan (included in Appendix VIII within this document).

It is important to note a caveat concerning the use of the word "contingency" within the context of this document. In the development of the SSM/PMAD the ability to handle fault conditions as they occurred was paramount. If contingencies were handled by a preplanned activity, then all possible contingencies would be explicitly understood prior to system run-time, and the executed activities would simply be handled through a massive table look-up. In order to achieve a viable success within the SSM/PMAD, contingencies are instead handled by knowledge processing activities which occur at system run-time. Therefore the handling of fortuitous events in the form of symptoms occurs whenever a fault occurs, and the state of the system is maintained as it changes rather than as a predetermined plan.

5.2 Overall Breadboard Configuration

The overall breadboard configuration is described in the following sections. A development chronology will be described, showing a schedule of activities which led to the present system; the present configuration will be shown, the functional interfaces and dataflows for the software will be discussed, and the grounding scheme used for the hardware will also be described.

5.2.1 Development Chronology

A synopsis of the development of the SSM/PMAD breadboard with respect to time requires an understanding of when the defined tasks were performed. These were:

1) TASK IV - October 1985 - Host computer selection
2) TASK I - July 1986 - CM/PMAD function definition
3) TASK II - August 1987 - Automation H/W & S/W definition
4) TASK III - December 1988 - Initial automation system.

The overall approach to task development and completion was that of a cascading effort with each preceding task supplying impetus to a following task. This provided a natural flow for the overall planning and development activity and proved to be manageable.

An overall schedule of activities is shown in Figure 5.2.1-1. The schedule provides breakdowns by task and also gives individual element relations to the Contract's Work Breakdown Structure. Also shown is Martin Marietta's continuing work and development for the SSM/PMAD in the 1989 calendar year.

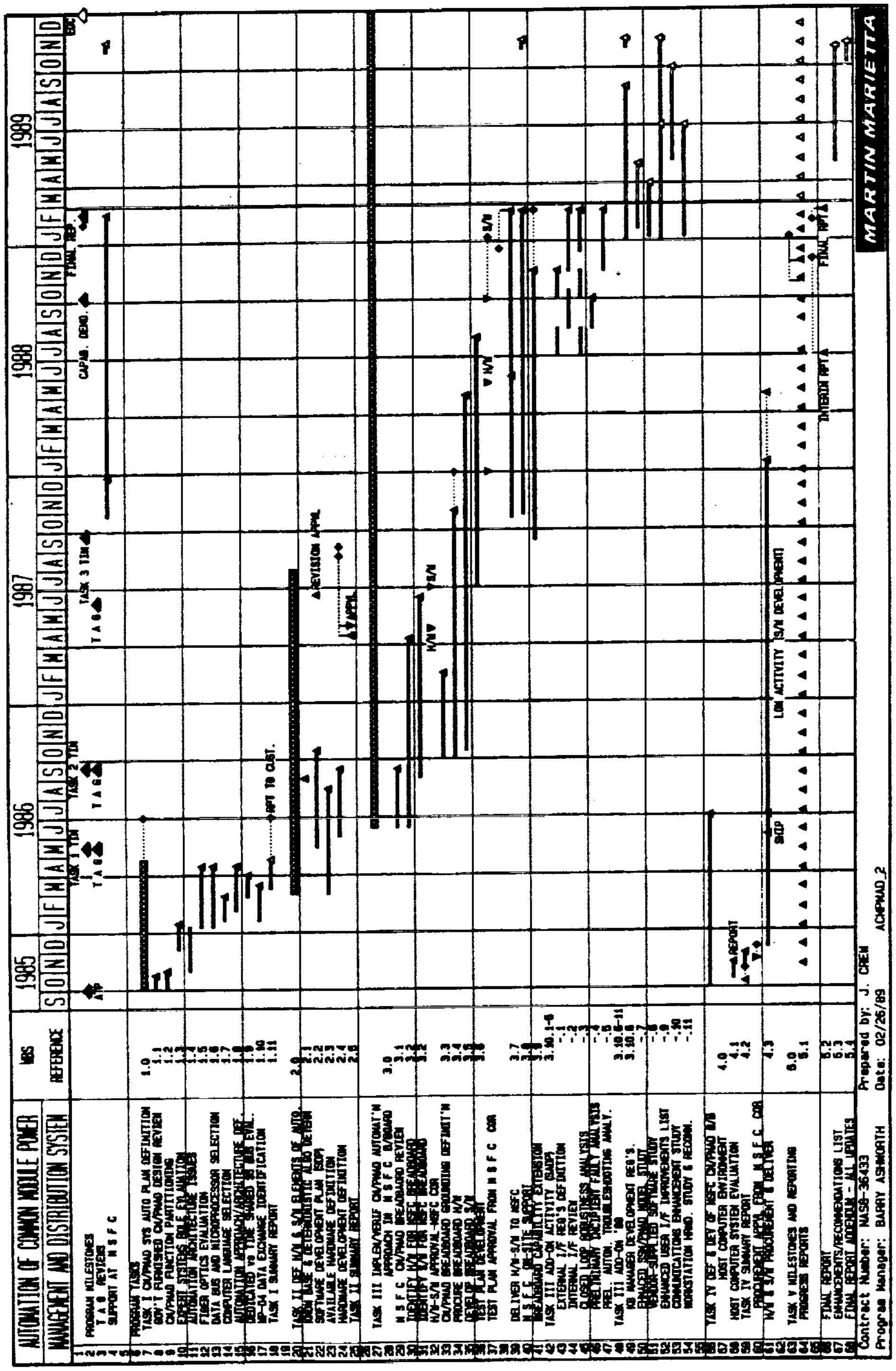


Figure 5.2.1-1 SSM/PMAD Development Schedule

5.2.2 Present Configuration

The SSM/PMAD automation system consists of two distinct segments. First, the computer hardware forms the computational engine segment. This is necessary for the second but distinct segment, the automation software. These two segments combine to form the facility for the system autonomy.

5.2.2.1 The Hardware Configuration

In describing the system hardware, it is important to note two points of interest. First, there are many individual computational engines making up the SSM/PMAD; and second, more than one type of communications interface is used. Various computer hardware components can be seen in Figure 5.2.2.1-1. Overall, the automation system hardware architecture is comprised of three levels. The top level is shared by the Symbolics 3620 D and the Xerox 1186. The middle level is occupied by the Motorola VME/10. And processing at the lower level is performed on Motorola 107 board level processors. There is an Ethernet connection between the Symbolics 3620 D and the Xerox 1186. The Xerox 1186 also is connected to the Motorola VME/10 via an RS 232 link. The Motorola VME/10 communicates to the lowest level processors (LLPs), and they in turn communicate to the switch interface controllers (SICs), both via an RS 422 interconnection. The LLPs provide control for either a power distribution control unit (PDCU) or a load center (LC). All requests for data originating at an LLP are routed to a SIC for completion.

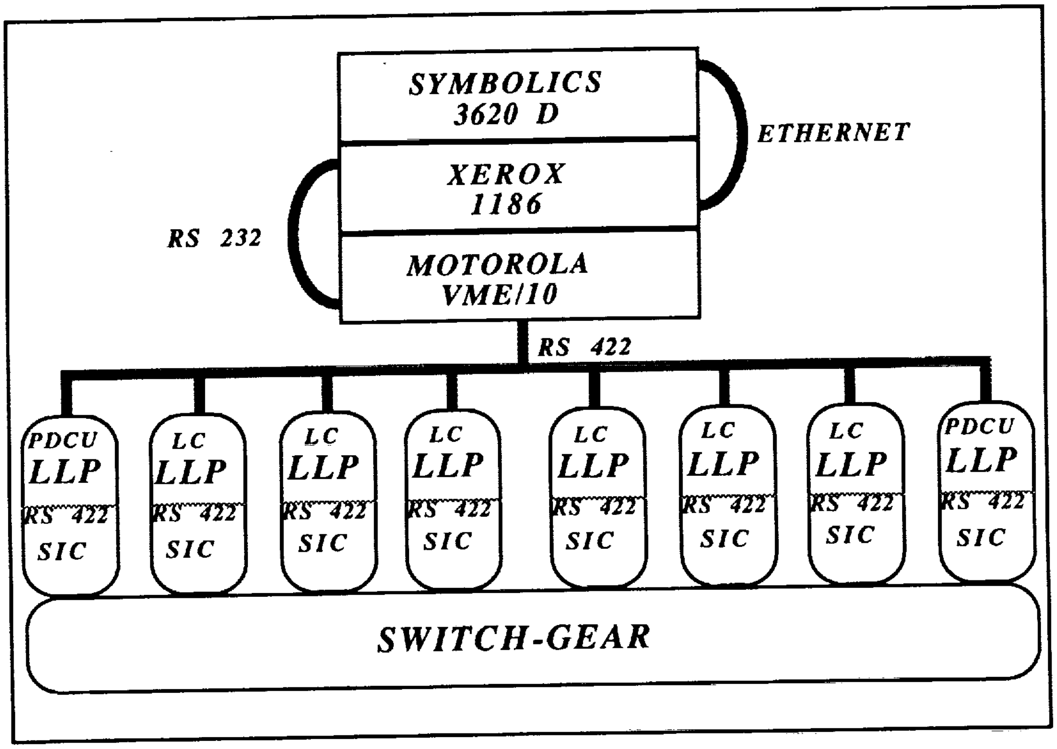


Figure 5.2.2.1-1 SSM/PMAD Automation Hardware Configuration

5.2.2.2 The Software Configuration

The software architecture, as can be seen in Figure 5.2.2.2-1, exists in a similar manner and represents the software part of the breadboard configuration. A scheduling example is used to describe the flow which is utilized within this configuration. Scheduling is accomplished using the FELES on the Symbolics 3620 D. The resultant schedule is passed along to the FRAMES at the Xerox 1186, which copies needed information from the schedule into its database. The schedule is passed intact to the CAC at the Motorola VME/10. The CAC then processes the schedule for inclusion by the lower level functions on the LLPs. The appropriate schedule segments are then sent to the respective LLPs and the system is ready to begin schedule execution. Data are then passed among the various software components on an as-needed basis. This provides strong partitioning to isolate needed knowledge base functions from deterministic functions.

As can be seen, some of the fault recovery and management functionality was isolated within a deterministic partition. This partition was attached to the lower level functions implemented in the LLPs and is known as the FRAMES lower level. This is an innovative approach within Artificial Intelligence (AI) development. The configuration strives for solution of the high level problem of power system automation rather than the more mundane approach of trying to define a simple problem domain which can be solved by a single expert system.

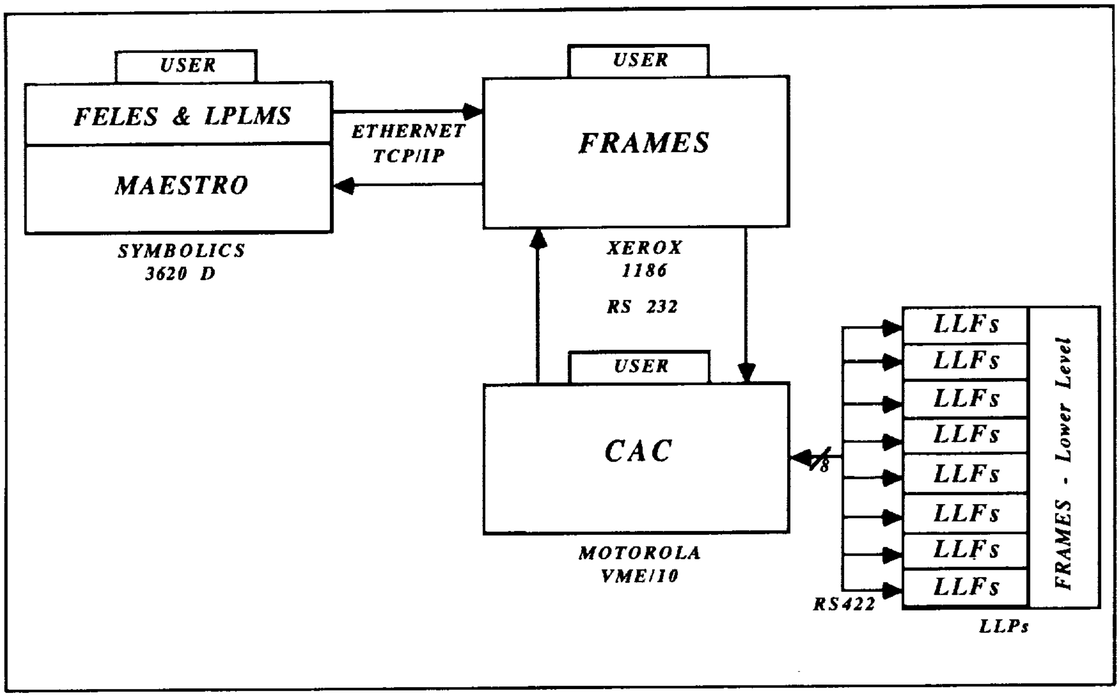


Figure 5.2.2.2-1 SSM/PMAD Automation Software Configuration

5.2.3 Functional Interfaces and Dataflow

This section of the report describes the functional interfaces between the logical modules of the automation software of SSM/PMAD. The first part gives a schematic overview of the logical modules and their overall responsibilities. The next two sections describe the specific data that flows between the logical modules, the transactions, and the LLP/SIC Interface Control Document, respectively. The last section describes the system-wide dataflows of SSM/PMAD.

5.2.3.1 Overview

Figure 5.2.3.1-1 gives a broad overview of the relationships between the logical modules of the automation software. The software residing on the Symbolics 3620 D, specifically, FELES, LPLMS, and MAESTRO are somewhat woven together via the Controller. FELES, LPLMS, and MAESTRO all use data from the same database of activities and equipment in order to perform their functions. For this reason it is not possible to completely breakdown the interfaces between these three modules. Rather, it is the Controller's responsibility to see that the processes on the Symbolics 3620 D operate cooperatively with one another.

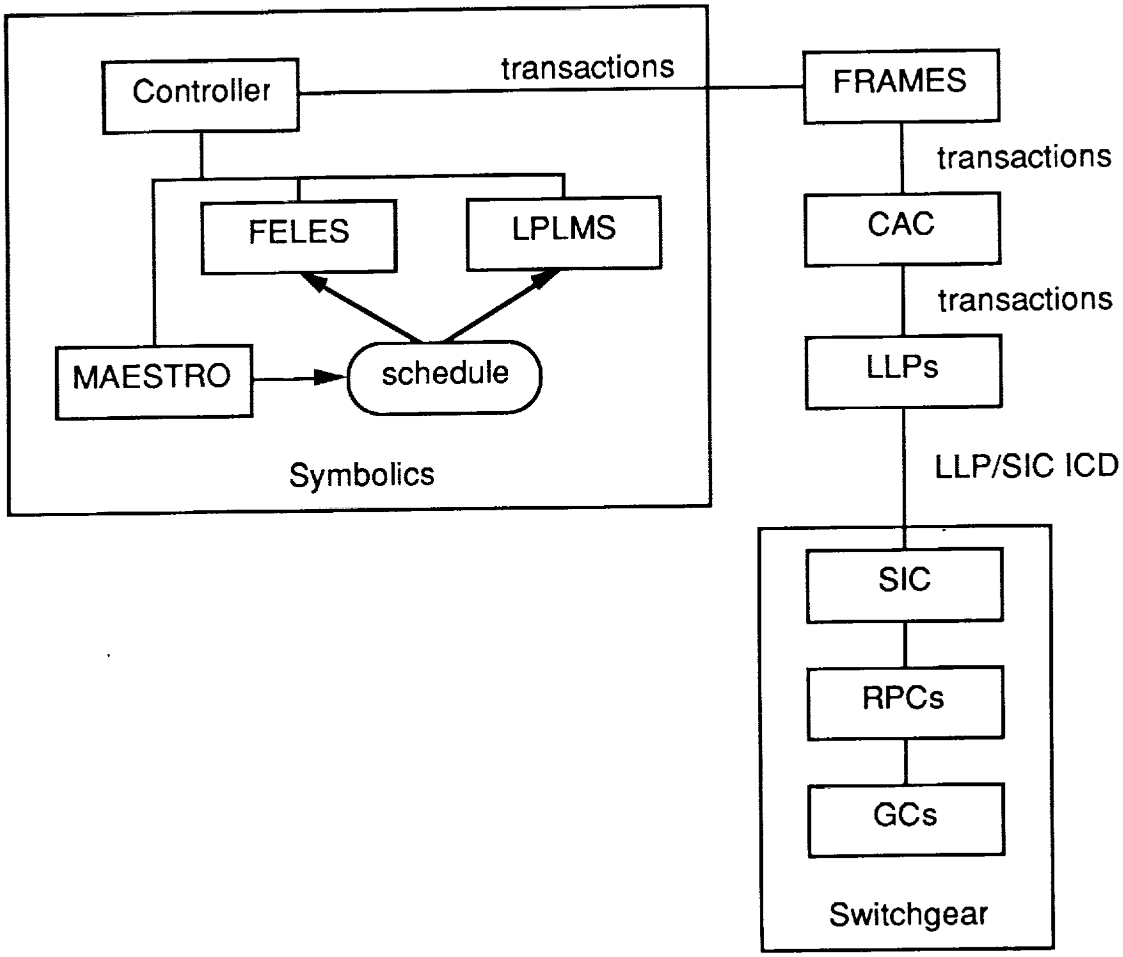


Figure 5.2.3.1-1 SSM/PMAD System Overview

5.2.3.1.1 The Controller

The Controller is responsible for maintaining the correct system state with respect to the processes residing on the Symbolics 3620 D. The Controller has a specific state transition network (described in the appendix) that it traverses. It controls execution of FELES, LPLMS, and MAESTRO, passing along any transactions directed toward them. In some sense, it also manages interaction of the user at the user interface, enabling and disabling some functionality. For example, during normal operation of the system the user is not allowed to create a new schedule. The Controller also manages contingencies so that MAESTRO, FELES, and LPLMS can respond correctly to contingency situations.

5.2.3.1.2 MAESTRO

MAESTRO, the scheduler, is responsible for taking a set of activities and a model of the power system network, and producing an efficient schedule for the activities. MAESTRO makes use of the loads and activities databases residing on the Symbolics 3620 D, as well as a model of the power system network to generate a schedule. MAESTRO is a complex scheduling system and has many capabilities which are discussed in detail in the section on resource scheduling (5.4).

5.2.3.1.3 Front End Load Enable Scheduler

The responsibility of FELES is to process and send the schedule generated by MAESTRO to FRAMES and the CAC. FELES partitions the schedule into 30 minute blocks and translates the scheduled information into RPC commands for FRAMES and the CAC. In addition to a simple translation, FELES makes use of the activities database to determine load parameters including power consumption, whether the load may be switched to redundant, whether the load is testable and may be interrupted or not, and the maximum and minimum current the load can draw. FELES is discussed in more detail in section 5.5.

5.2.3.1.4 Load Priority List Management System

The LPLMS is responsible for maintaining and transmitting a load priority list to FRAMES and the CAC. The load priority list is used to keep track of the priorities of loads (RPCs in the case of FRAMES and the CAC). This list is used by the LLPs in the event contingency situations require load shedding. The loads with the least priority are always shed first. This prevents the shedding of critical loads within the overall system. Priorities for loads are computed dynamically and take into account criticality of loads. LPLMS is discussed in more detail in section 5.6.

5.2.3.1.5 Fault Recovery and Management Expert System

The responsibility of FRAMES is the management and diagnosis of faults. The initial management of faults, keeping the power system safe, etc., is done by the lower level FRAMES functions within the LLP software, and by circuit breaker hardware units. The diagnosis function of FRAMES is performed on the Xerox 1186 and is implemented in an knowledge base system fashion. FRAMES maintains a model of the power system network, indicating what RPCs should be turned on and what RPCs are actually on. When FRAMES receives contingency information from the LLPs, it proceeds to isolate the fault and make a diagnosis as to what has occurred. This information is then communicated back to the Symbolics 3620 D for recovery of the schedule.

5.2.3.1.6 Communications and Algorithmic Controller

The CAC is responsible for processing communications between the LLPs and FRAMES. Schedules sent down from the Symbolics 3620 D are partitioned here and sent to the respective LC and PDCU LLPs. Data from LLPs are collected and passed on up to FRAMES.

5.2.3.1.7 Lowest Level Processors

The LLPs are responsible for executing scheduled RPC commands and responding to exception conditions intelligently. The LLPs maintain individual command lists and load priority lists. They support data compression and reporting software, command execution, load shed execution, and condition execution software. They also execute automatic switch control and verify configuration and limit allocations.

5.2.3.1.8 The Switchgear

The switchgear includes the hardware for both RPCs and data collection and commands from and to RPCs. For purposes of describing the functional interfaces of the automation software the various hardware is viewed as one component: switchgear. The interface between the switchgear and the LLPs is described in an interface control document and in the last part of this section on the functional interfaces.

5.2.3.2 Transactions

A transaction consists of the parts as shown in Figure 5.2.3.2-1. The message block is the actual data of each transaction. All the transactions are defined in the following subsections.

Message Start	x	Start of message indicator Control-A (ascii 1)
Destination	x	Address of unit where message is being sent
Source	x	Address of unit sending the message
Message Type	p	Type of message
Message Block	?	Contains message data bytes The format of this varies with each message type
Message End	x	End of message indicator CR (ascii 13)

Figure 5.2.3.2-1 Transaction Format

5.2.3.2.1 Time Transaction

Type	01
Source	FELES
Destination	CAC, FRAMES
Description	TIME provides the timing parameters for time synchronization of the breadboard software components. FELES will distribute this to the CAC and FRAMES. Now represents the current time. Start of Mission provides an actual calendar/clock time from which to base all scheduling offsets. It corresponds to mission time 00:00:00 (dd:hh:mm). All time based schedule data will be represented as minutes offset from the Start of Mission.

FIELD	LENGTH	FORMAT	DESCRIPTION
Now Month	2	numeric	Calendar/clock month
Now Day	2	numeric	Calendar/clock day
Now Year	2	numeric	Calendar/clock year
Now Hour	2	numeric	Calendar/clock hour
Now Minute	2	numeric	Calendar/clock minute
Now Second	2	numeric	Calendar/clock second
SOM Month	2	numeric	Start of Mission month
SOM Day	2	numeric	Start of Mission day
SOM Year	2	numeric	Start of Mission year
SOM Hour	2	numeric	Start of Mission hour
SOM Minute	2	numeric	Start of Mission minute
SOM Second	2	numeric	Start of mission second

5.2.3.2.2 Event List Transaction

Type	02
Source	FELES
Destination	CAC, FRAMES
Description	FELES will distribute the events for the load enable schedule to the CAC and FRAMES for operation of the breadboard.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	6	numeric	Effective time of the event list
Number of Events	2	packed79	Number of events
EVENT	29	GROUP	AN EVENT DESCRIPTOR
Time of Event	6	numeric	Time event is to be initiated
Component	3	alphanumeric	Id of the component
Event	1	alphanumeric	F-off, N-on, C-change
Max Power	5	numeric	watts (0-99999)
Permission to Test	1	alphanumeric	Y-yes, N-no
Redundancy	1	alphanumeric	Y-yes, N-no
Switch to Redundant	1	alphanumeric	Y-yes, N-no
Max Current	3	numeric	dAmps (0-999)
Min Current	3	numeric	dAmps (0-999)
Min Power	5	numeric	watts (0-99999)

5.2.3.2.3 Load Priority List Transaction

Type 03
Source FELES
Destination CAC, FRAMES
Description Whenever a new load priority list is created, FELES will distribute the list to FRAMES and the CAC.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	6	numeric	Effective time of the priority list
Number of Components	2	packed79	Number of components
Component	3	alphanumeric	Id of the component

5.2.3.2.4 Component Switch to Redundant Transaction

Type 06
Source FRAMES
Destination FELES
Description A transaction indicating those RPCs whose loads switched to redundant power supply.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Redundants	2	packed79	Number of Redundant specs
SWITCH TO REDUNDANT	9	GROUP	COMPONENT SWITCH TO REDUNDANTS
Component	3	numeric	Id of the component
Time of switch	6	numeric	Time switch is to be initiated

5.2.3.2.5 Loads Shed Transaction

Type 07
Source FRAMES
Destination FELES
Description FRAMES will notify the FELES anytime loads are shed, so that appropriate scheduling decisions may be made.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Sheds	2	packed79	Number of load shed specs
LOAD SHED	9	GROUP	LOAD SHED DESCRIPTOR
Component	3	numeric	Id of the component
Time of switch	6	numeric	Time the shed was initiated

5.2.3.2.6 Contingency Event List Transaction

Type	08
Source	FELES
Destination	CAC, FRAMES
Description	When FELES has completed its reaction to a contingency situation, CAC and FRAMES will be notified of the new load enable schedule as well as what state the power system components should be in in order to successfully execute the new set of events. All state entries will be listed before all event entries, both state and event entries have the same format, the key difference being all "time of state"s are filled with zeros.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	6	numeric	Effecitve time of the contingency event list
Number of States/Events	2	packed79	Number of states and events
STATE	29	GROUP	A CONTINGENCY STATE DESCRIPTOR
Time of State	6	numeric	Always 000000
Component	3	alphanumeric	Id of the component
Event	1	alphanumeric	F-off, N-on, C-change
Max Power	5	numeric	watts (0-99999)
Permission to Test	1	alphanumeric	Y-yes, N-no
Redundancy	1	alphanumeric	Y-yes, N-no
Switch to Redundant	1	alphanumeric	Y-yes, N-no
Max Current	3	numeric	dAmps (0-999)
Min Current	3	numeric	dAmps (0-999)
Min Power	5	numeric	watts (0-99999)
EVENT	29	GROUP	AN EVENT DESCRIPTOR
Time of Event	6	numeric	Time event is to be initiated
Component	3	alphanumeric	Id of the component
Event	1	alphanumeric	F-off, N-on, C-change
Max Power	5	numeric	watts (0-99999)
Permission to Test	1	alphanumeric	Y-yes, N-no
Redundancy	1	alphanumeric	Y-yes, N-no
Switch to Redundant	1	alphanumeric	Y-yes, N-no
Max Current	3	numeric	dAmps (0-999)
Min Current	3	numeric	dAmps (0-999)
Min Power	5	numeric	watts (0-99999)

5.2.3.2.7 Component Out of Service Transaction

Type	09
Source	FRAMES
Destination	FELES
Description	Whenever a component is known to be out of service FRAMES will notify FELES so that appropriate scheduling decisions may be made.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Services	2	packed79	Number of out of service entries
OUT OF SERVICE	15	GROUP	COMPONENT OUT OF SERVICE DESCRIPTOR
Component	3	alphanumeric	Id of the component
Begin Time	6	numeric	Beginning time the component is out of service
End Time	6	numeric	Ending time the component is out of service

5.2.3.2.8 Actual Power Utilization Transaction

Type	10
Source	FRAMES
Destination	FELES
Description	In order to report/graph actual power utilization of components vs. available and/or scheduled power FRAMES must notify FELES of those measurements.

FIELD	LENGTH	FORMAT	DESCRIPTION
BusA start time	6	numeric	Beginning time for BusA utilization
BusA end time	6	numeric	Ending time for BusA utilization
Number of Utilizations	2	packed79	Number of component utilization entries
UTILIZATION	8	GROUP	COMPONENT UTILIZATION
Component	3	alphanumeric	Id of the component
Power Utilization	5	numeric	watts (0-99999)
BusB start time	6	numeric	Beginning time for BusB utilization
BusB end time	6	numeric	Ending time for BusB utilization
Number of Utilizations	2	packed79	Number of component utilization entries
UTILIZATION	8	GROUP	COMPONENT UTILIZATION
Component	3	alphanumeric	Id of the component
Power Utilization	5	numeric	watts (0-99999)

5.2.3.2.9 Ready? Transaction

Type	11
Source	FELES
Destination	CAC, FRAMES
Description	READY? is send by FELES to tell FRAMES and the CAC to initialize and be prepared for the initial EVENTS and PRIORITIES, when the initialization has occured CAC and FRAMES will notify FELES with the delarative I'm READY! message.

FIELD	LENGTH	FORMAT	DESCRIPTION
Ready?	1	alphanumeric	? - are your ready?, initialize

5.2.3.2.10 Ready! Transaction

Type	12
Source	CAC, FRAMES
Destination	FELES
Description	The declarativ message READY! is sent by FRAMES and the CAC to the FELES as a notification that they are ready to receive the inital EVENTS and PRIORITIES.

FIELD	LENGTH	FORMAT	DESCRIPTION
Ready	1	alphanumeric	Y - yes

5.2.3.2.11 Initialized Transaction

Type	13
Source	CAC, FRAMES
Destination	FELES
Description	After receiving the initial EVENTS and PRIORITIES, FRAMES and the CAC send this message to the FELES, this notifies FELES that the other breadboard computer system components are ready for operation.

FIELD	LENGTH	FORMAT	DESCRIPTION
Initialized	1	alphanumeric	Y - yes, I've received initial EVENTS and PRIORITIES

5.2.3.2.12 Source Power Change Transaction

Type	14
Source	FELES
Destination	CAC, FRAMES
Description	The SOURCE POWER CHANGE is a simulated space station message, i.e. someone/thing of authority has notified the module that there will be change in the availability of power to the module.

FIELD	LENGTH	FORMAT	DESCRIPTION
Start time	6	numeric	Starting time of the specified available power
End time	6	numeric	Ending time of the specified available power
Power	5	numeric	watts (0-99999, 25000 max)

5.2.3.2.13 Contingency Start Transaction

Type	15
Source	FRAMES
Destination	FELES
Description	An anomolous condition has been recognized in the power system, FRAMES is working the situation and tells FELES so. Any transactions received by FELES between the CONTINGENCY-START and CONTINGENCY-END messages are considered pertinent information to the contingency situation.

FIELD	LENGTH	FORMAT	DESCRIPTION
Contingency time	6	numeric	Start time of the contingency situation

5.2.3.2.14 Contingency End Transaction

Type	16
Source	FRAMES
Destination	FELES
Description	The contingency situation has been handled by FRAMES and all pertinent information has been sent to FELES, FELES should now handle implications to the schedule.

FIELD	LENGTH	FORMAT	DESCRIPTION
Contingency time	6	numeric	End time of the contingency situation

5.2.3.2.15 LLP Availability Transaction

Type	40
Source	CAC
Destination	FRAMES
Description	Inform Frames of the availability of LLPs.

FIELD	LENGTH	FORMAT	DESCRIPTION
Unavailable LLPs	1	byte	Id of unavailable LLP (A-H)

5.2.3.2.16 Fault Event List Transaction

Type	17
Source	FRAMES
Destination	CAC
Description	FRAMES must be able to command tripped switches at the CAC level to complete its analysis of a contingency situation. Since this direct command capability is executed immediately, time of event is not necessary, the order of the events is important though.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Opens	2	numeric	Number of opens
Number of Flips	2	numeric	Number of flips
Number of Closes	2	numeric	Number of closes
Number of States/Events	2	packed79	Number of states and events
EVENT	29	GROUP	AN EVENT DESCRIPTOR
Time of Event	6	numeric	Always 00000 (not used)
Component	3	alphanumeric	Id of the component
Event	1	alphanumeric	F-off, N-on, C-change
Max Power	5	numeric	watts (0-99999)
Permission to Test	1	alphanumeric	Y-yes, N-no
Redundancy	1	alphanumeric	Y-yes, N-no
Switch to Redundant	1	alphanumeric	Y-yes, N-no
Max Current	3	numeric	dAmps (0-999)
Min Current	3	numeric	dAmps (0-999)
Min Power	5	numeric	watts (0-99999)

5.2.3.2.17 Switch Positions Transaction

Type	42
Source	LLP
Destination	FRAMES
Description	Inform FRAMES of actual state of switches after a switching operation.

FIELD	LENGTH	FORMAT	DESCRIPTION
Swdat	29	alphanumeric	F-off, N-on

5.2.3.2.18 Switch Performance Transaction

Type	43
Source	LLP
Destination	FRAMES
Description	Inform FRAMES of time based switch performance information.

FIELD	LENGTH	FORMAT	DESCRIPTION
Start Time	4	integer	Start of performance interval in seconds
End Time	4	integer	End of performance interval in seconds
Number Switches	4	integer	Number of switches (always 28)
CURRENT DATA	20	GROUP	A SWITCH PERFORMANCE DESCRIPTOR
Current Avg.	4	integer	Average current in dAmps
Current max.	4	integer	Maximum current in dAmps
Current min.	4	integer	Minimum current in dAmps
Max. Time	4	integer	Time of maximum current
Min. time	4	integer	Time of minimum current

5.2.3.2.19 Source Reduction Status Transaction

Type	44
Source	LLP
Destination	FRAMES
Description	Send FRAMES source reduction status data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Where	1	byte	Not used
Null Byte	1	byte	Not used
Switch Number	4	integer	Not used
Anomalous	1	byte	Flag denoting anomalous condition
Null Byte	1	byte	Not used
Number of Switches	4	integer	Number of switches (always 28)
STATUS	12	GROUP	SWITCH STATUS DESCRIPTOR
Word 0	4	integer	Bit defined
Bit 0		bit	Surge current trip
Bit 1		bit	Over current trip
Bit 2		bit	Under voltage trip
Bit 3		bit	Ground fault trip
Bit 4		bit	Over temparature flag
Bit 5		bit	Fast trip trip
Bit 6		bit	Already tripped flag
Bit 7		bit	Already on flag
Bit 8		bit	Already off flag
Bit 9		bit	Scheduled off, drawing current
Bit 10		bit	Schedule on, not drawing current
Bit 11		bit	SIC not present
Bit 12		bit	Generic Card not present
Bit 13		bit	Not enough power available
Bit 14		bit	Could not schedule flag
Bit 15		bit	Tripped flag (anomalous flag)
Word 1	4	integer	Bit defined
Bit 0		bit	Mechanical on, should be off
Bit 1		bit	Mechanical off, should be on
Bit 2		bit	Solid state on, should be off
Bit 3		bit	Solid state off, should be on
Bit 4		bit	RPC command lines on, should be off
Bit 5		bit	RPC command lines off, should be on
Bit 6		bit	RPC command lines in illegal state
Bit 7		bit	Out of current limits
Bit 8		bit	Out of power limits
Bit 9		bit	Switch has been shed
Bit 10		bit	Over temparature warning
Bit 11		bit	No change in RPC command lines
Bit 12		bit	Not used
Bit 13		bit	Unable to command
Bit 14		bit	Switched to redundant
Bit 15		bit	Current over range warning
Current	4	integer	Current through switch
Switch Timestamps	8	integer	Two switch status time stamps (busA and B)
Temp Timestamp	4	integer	Temparature status time stamp

5.2.3.2.20 Sensor Performance Transaction

Type	45
Source	LLP
Destination	FRAMES
Description	Send FRAMES sensor performance data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Start Time	4	integer	Start of performance interval (seconds)
End Time	4	integer	End of performance interval (seconds)
Number Sensors	4	integer	Number of sensors (2-LC, 13-PDCU)
SENSOR	68	GROUP	SENSOR PERFORMANCE DESCRIPTOR
Vrmsavg	4	integer	Average RMS voltage (Volts)
Vrmsmax	4	integer	Maximum RMS voltage (Volts)
Vrmsmin	4	integer	Minimum RMS voltage (Volts)
Irmsavg	4	integer	Average RMS current (dAmps)
Irmsmax	4	integer	Maximum RMS current (dAmps)
Irmsmin	4	integer	Minimum RMS current (dAmps)
Preavg	4	integer	Average real power (Watts)
Premax	4	integer	Maximum real power (Watts)
Premin	4	integer	Minimum real power (Watts)
Freqavg	4	integer	Average frequency (Hertz)
Freqmax	4	integer	Maximum frequency (Hertz)
Freqmin	4	integer	Minimum frequency (Hertz)
Pfavg	4	integer	Average power factor
Pfmax	4	integer	Maximum power factor absolute
Pfmina	4	integer	Minimum leading power factor
Pfminb	4	integer	Minimum lagging power factor
Enrgy	4	integer	Energy consumed (Watts)

5.2.3.2.21 Switch and Sensor Status Transaction

Type	46
Source	LLP
Destination	FRAMES
Description	Send FRAMES switch and sensor status data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Anomalous	1	byte	Flag denoting anomalous condition
Null Byte	1	byte	Not used
Number Switches	4	integer	Number of switches (always 28)
STATUS	12	GROUP	SWITCH STATUS DESCRIPTOR
Word 0	4	integer	Bit defined
Bit 0		bit	Surge current trip
Bit 1		bit	Over current trip
Bit 2		bit	Under voltage trip
Bit 3		bit	Ground fault trip
Bit 4		bit	Over temparature flag
Bit 5		bit	Fast trip trip
Bit 6		bit	Already tripped flag
Bit 7		bit	Already on flag
Bit 8		bit	Already off flag
Bit 9		bit	Scheduled off, drawing current
Bit 10		bit	Schedule on, not drawing current
Bit 11		bit	SIC not present
Bit 12		bit	Generic Card not present
Bit 13		bit	Not enough power available
Bit 14		bit	Could not schedule flag
Bit 15		bit	Tripped flag (anomalous flag)
Word 1	4	integer	Bit defined
Bit 0		bit	Mechanical on, should be off
Bit 1		bit	Mechanical off, should be on
Bit 2		bit	Solid state on, should be off
Bit 3		bit	Solid state off, should be on
Bit 4		bit	RPC command lines on, should be off
Bit 5		bit	RPC command lines off, should be on
Bit 6		bit	RPC command lines in illegal state
Bit 7		bit	Out of current limits
Bit 8		bit	Out of power limits
Bit 9		bit	Switch has been shed
Bit 10		bit	Over temparature warning
Bit 11		bit	No change in RPC command lines
Bit 12		bit	Not used
Bit 13		bit	Unable to command
Bit 14		bit	Switched to redundant
Bit 15		bit	Current over range warning
Current	4	integer	Current through switch
Switch Timestamps	8	integer	Two switch status time stamps (busA and B)
Temp Timestamp	4	integer	Temparature status time stamp
Number Sensors	4	integer	Number of Sensors (2-LC, 13-PDCU)
VALUES	36	GROUP	SENSOR DATA DESCRIPTOR
Vrms	4	integer	RMS voltage (Volts)
Irms	4	integer	RMS current (dAmps)
Vdc	4	integer	Voltage DC component (Volts)
Idc	4	integer	Current DC component (dAmps)
Pavg	4	integer	Real power (Watts)
Freq	4	integer	Frequency (Hertz)
Temp	4	integer	Temperature
Pfs	4	integer	Signed power factor (* 100000)
State	4	integer	Bit defined sensor state
Bit 0		bit	No error bit
Bit 1		bit	Power out of limits
Bit 2		bit	Current out of limits
Bit 3		bit	Voltage out of limits
Bit 4		bit	Temperature out of limits

5.2.3.2.22 Fault Event Status Transaction

Type	47
Source	LLP
Destination	FRAMES
Description	Send FRAMES fault event status data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Where	1	byte	1, 2, 3, or 4 for location of fault checking error
Null Byte	1	byte	Not used
Switch Number	4	integer	Switch where problem occured
Anomalous	1	byte	Flag denoting anomalous condition
Null Byte	1	byte	Not used
Number of Switches	4	integer	Number of switches (always 28)
STATUS	12	GROUP	SWITCH STATUS DESCRIPTOR
Word 0	4	integer	Bit defined
Bit 0		bit	Surge current trip
Bit 1		bit	Over current trip
Bit 2		bit	Under voltage trip
Bit 3		bit	Ground fault trip
Bit 4		bit	Over temparature flag
Bit 5		bit	Fast trip trip
Bit 6		bit	Already tripped flag
Bit 7		bit	Already on flag
Bit 8		bit	Already off flag
Bit 9		bit	Scheduled off, drawing current
Bit 10		bit	Schedule on, not drawing current
Bit 11		bit	SIC not present
Bit 12		bit	Generic Card not present
Bit 13		bit	Not enough power available
Bit 14		bit	Could not schedule flag
Bit 15		bit	Tripped flag (anomalous flag)
Word 1	4	integer	Bit defined
Bit 0		bit	Mechanical on, should be off
Bit 1		bit	Mechanical off, should be on
Bit 2		bit	Solid state on, should be off
Bit 3		bit	Solid state off, should be on
Bit 4		bit	RPC command lines on, should be off
Bit 5		bit	RPC command lines off, should be on
Bit 6		bit	RPC command lines in illegal state
Bit 7		bit	Out of current limits
Bit 8		bit	Out of power limits
Bit 9		bit	Switch has been shed
Bit 10		bit	Over temparature warning
Bit 11		bit	No change in RPC command lines
Bit 12		bit	Not used
Bit 13		bit	Unable to command
Bit 14		bit	Switched to redundant
Bit 15		bit	Current over range warning
Current	4	integer	Current through switch
Switch Timestamps	8	integer	Two switch status time stamps (busA and B)
Temp Timestamp	4	integer	Temparature status time stamp

5.2.3.2.23 Switch and Sensor Data Transaction

Type	48
Source	LLP
Destination	FRAMES
Description	Inform FRAMES of actual switch and sensor data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Dtime	4	integer	Time of update
Number Switches	4	integer	Number of switches (always 28)
Current	112	integer	Array [0..127] of switch currents (dAmps)
Number Sensors	4	integer	Number of sensors (2-LC, 13-PDCU)
SENSOR	12	GROUP	SENSOR DATA DESCRIPTOR
Idat	4	integer	Sensor current (dAmps)
Vdat	4	integer	Sensor voltage (Volts)
Pdat	4	integer	Sensor power (Watts)

5.2.3.3 LLP/SIC Transactions

The LLP to SIC transactions are defined in this part of the report. The first subsection describes the definitions of data used by the transactions, while the second subsection lists the various commands and responses between the LLP and SIC. The command definitions given here are directly adapted from the LLP/SIC Interface Control Document.

5.2.3.3.1 Definitions

This section defines the format of the various words used by the commands in the following section.

5.2.3.3.1.1 Status Format Definition

Byte 1

$30 --> status OK

$31 --> status NOT OK

Byte 2

cc --> copy of command received with MSB bit always set to 1

Byte 3

$80 --> status OK

$FF --> unknown command

$81 --> first byte not a command byte

$82 --> did not receive first data byte

$83 --> first data byte msb not high

$84 --> did not receive second data byte

$85 --> second data byte msb not high

$86 --> switch already on

$87 --> switch already tripped when tried to turn it on

$88 --> switch already off

$89 --> switch already tripped when tried to turn it off

$8A --> GC Data Valid error when getting switch data

$8B --> continuous buffer overflow (reset continuous buffer)

$8C --> once buffer overflow (redo once buffer)

$A1 --> SIC character buffer overrun

$A2 --> character overwritten (OE)

$A4 --> parity error from UART (PE)

$A6 --> OE and PE

$A8 --> framing error (FE)

$AA --> FE and OE

$AC --> FE and PE

$AE --> FE and OE and PE

$F7 --> SIC internal memory parity error

Byte 4

$0D --> end of status

5.2.3.3.1.2 Command Word Format Definition

Byte 1
cc --> command

Byte 2
dd1 --> first byte of data word

Byte 3
dd2 --> second byte of data word

Byte 4
$0D --> end of command

5.2.3.3.1.3 Switchword Format Definition

	bit 14 = 0 (switch not tripped)	bit 14 = 1 (tripped)
bit 0	current (1)	tripped surge current H
bit 1	current (1)	tripped fast trip H
bit 2	current (1)	spare
bit 3	current (1)	spare
bit 4	current (1)	tripped over current (I2t) H
bit 5	current (1)	tripped under voltage H
bit 6	current MSB (1)	tripped ground fault H
bit 7	always 1	always 1
bit 8	current overrange H (1)	tripped overtemp latched H
bit 9	S2 solid state switch on H	S2 solid state switch on H
bit 10	S1 mechanical switch on H	S1 mechanical switch on H
bit 11	over temperature H	over temperature H
bit 12	off control input H (2)	off control input H (2)
bit 13	on control input H (2)	on control input H (2)
bit 15	always 1	always 1

(1) RMS current

(2)

bit 13	bit 12	RPC command
0	0	on (error in hardware)
0	1	on
1	0	off
1	1	no change

5.2.3.3.1.4 GC Data Valid Word Format Definition

bit 0	GC Data Valid switch 7 H
bit 1	GC Data Valid switch 8 H
bit 2	GC Data Valid switch 9 H
bit 3	GC Data Valid switch 10 H
bit 4	GC Data Valid switch 11 H
bit 5	GC Data Valid switch 12 H
bit 6	GC Data Valid switch 13 H
bit 7	always 1
bit 8	GC Data Valid switch 0 H
bit 9	GC Data Valid switch 1 H
bit 10	GC Data Valid switch 2 H
bit 11	GC Data Valid switch 3 H
bit 12	GC Data Valid switch 4 H
bit 13	GC Data Valid switch 5 H
bit 14	GC Data Valid switch 6 H
bit 15	always 1

NOTE: L - data valid
H - data not valid

5.2.3.3.1.5 Sensorword Format Definition

bit 0	sensor data bit 0
bit 1	sensor data bit 1
bit 2	sensor data bit 2
bit 3	sensor data bit 3
bit 4	don't care
bit 5	don't care
bit 6	don't care
bit 7	always 1
bit 8	sensor data bit 4
bit 9	sensor data bit 5
bit 10	sensor data bit 6
bit 11	sensor data bit 7
bit 12	don't care
bit 13	don't care
bit 14	don't care
bit 15	always 1

A Sensorword-Set consists of 9 sensorwords:

V rms
I rms
V offset
I offset
V instantaneous
I instantaneous
P instantaneous
P real
frequency

5.2.3.3.2 Commands

The commands and responses between the LLP and SIC are defined here. This section only defines the syntax of the commands. The semantics of what the commands mean are derived from the command names and descriptions.

5.2.3.3.2.1 Unconditionally Command Switch Off

```
COMMAND:     Command switch off immediately even if already off or tripped.
FORMAT:          cc -->  $20
                dd1 -->  $80 + j
                         j:
                            bit 0 -->  switch 0
                            bit 1 -->  switch 1
                            bit 2 -->  switch 2
                            bit 3 -->  switch 3
                            bit 4 -->  switch 4
                            bit 5 -->  switch 5
                            bit 6 -->  switch 6
                dd2 -->  $80 + k
                         k:
                            bit 0 -->  switch 7
                            bit 1 -->  switch 8
                            bit 2 -->  switch 9
                            bit 3 -->  switch 10
                            bit 4 -->  switch 11
                            bit 5 -->  switch 12
                            bit 6 -->  switch 13
RESPONSE:    Status
```

5.2.3.3.2.2 Unconditionally Command Switch On

```
COMMAND:     Command switch on immediately even if already on or tripped.
FORMAT:         cc --> $21
               dd1 --> $80 + j
                       j:
                           bit 0 -->  switch 0
                           bit 1 -->  switch 1
                           bit 2 -->  switch 2
                           bit 3 -->  switch 3
                           bit 4 -->  switch 4
                           bit 5 -->  switch 5
                           bit 6 -->  switch 6
               dd2 --> $80 + k
                       k:
                           bit 0 -->  switch 7
                           bit 1 -->  switch 8
                           bit 2 -->  switch 9
                           bit 3 -->  switch 10
                           bit 4 -->  switch 11
                           bit 5 -->  switch 12
                           bit 6 -->  switch 13
RESPONSE:         Status
```

5.2.3.3.2.3 Reset Switch

```
COMMAND:        Reset switch.
FORMAT:               cc --> $22
                     dd1 --> $80 + j
                             j:
                                bit 0 --> switch 0
                                bit 1 --> switch 1
                                bit 2 --> switch 2
                                bit 3 --> switch 3
                                bit 4 --> switch 4
                                bit 5 --> switch 5
                                bit 6 --> switch 6
                     dd2 --> $80 + k
                             k:
                                bit 0 --> switch 7
                                bit 1 --> switch 8
                                bit 2 --> switch 9
                                bit 3 --> switch 10
                                bit 4 --> switch 11
                                bit 5 --> switch 12
                                bit 6 --> switch 13
RESPONSE:       Status
```

5.2.3.3.2.4 Select GC

COMMAND:	Select GC (all GC select codes will be set to zero).
FORMAT:	cc --> $23
	dd1 --> $86
	dd2 --> $85
RESPONSE:	Status

5.2.3.3.2.5 Execute SIC Firmware Reset

COMMAND: Execute SIC firmware reset (does not reset actual set configuration).

FORMAT:
cc --> $24
dd1 --> $80
dd2 --> $80

RESPONSE: Four bytes plus Status:

Byte 1:
bit 0 --> 0 if GC7 connected, 1 if not
bit 1 --> 0 if GC8 connected, 1 if not
bit 2 --> 0 if GC9 connected, 1 if not
bit 3 --> 0 if GC10 connected, 1 if not
bit 4 --> 0 if GC11 connected, 1 if not
bit 5 --> 0 if GC12 connected, 1 if not
bit 6 --> 0 if GC13 connected, 1 if not
bit 7 --> always 1

Byte 2:
bit 0 --> 0 if GC0 connected, 1 if not
bit 1 --> 0 if GC1 connected, 1 if not
bit 2 --> 0 if GC2 connected, 1 if not
bit 3 --> 0 if GC3 connected, 1 if not
bit 4 --> 0 if GC4 connected, 1 if not
bit 5 --> 0 if GC5 connected, 1 if not
bit 6 --> 0 if GC6 connected, 1 if not
bit 7 --> always 1

Byte 3:
bit 0 --> current SIC switch 0 setting
bit 1 --> current SIC switch 1 setting
bit 2 --> current SIC switch 2 setting
bit 3 --> current SIC switch 3 setting
bit 4 --> 0 if A/D connected, 1 if not
bit 5 --> don't care
bit 6 --> don't care
bit 7 --> always 1

Byte 4:
bit 0 --> don't care
bit 1 --> don't care
bit 2 --> don't care
bit 3 --> don't care
bit 4 --> don't care
bit 5 --> don't care
bit 6 --> don't care
bit 7 --> always 1

Status

5.2.3.3.2.6 Reset Continuous Buffer

COMMAND: Reset continuous buffer
FORMAT:
cc --> $25
dd1 --> $80
dd2 --> $80
RESPONSE: Status

5.2.3.3.2.7 Fill Continuous Buffer

COMMAND: Fill continuous buffer. First use reset continuous buffer, then use this command to download code that is to be continuously executed. Code will start executing as soon as the download is started. Up to 80 of these commands may be concatenated before the buffer space is overrun.
FORMAT:
cc --> $26
ee1 --> $80 + q
q:
higher 4 bits of downloaded 8 bit code
ee2 --> $80 + r
r:
lower 4 bits of downloaded 8 bit code
ee3 --> $26 until last command, then $0D
RESPONSE: Status

5.2.3.3.2.8 Fill Once Buffer

COMMAND: Fill once buffer. This command is used to download code that is to be executed only once. Code execution is started by the trigger once buffer command. Up to 80 of these commands may be concatenated before the buffer space is overrun.

FORMAT:
cc --> $27
ee1 --> $80 + q
q:
higher 4 bits of downloaded 8 bit code
ee2 --> $80 + r
r:
lower 4 bits of downloaded 8 bit code
ee3 --> $26 until last command, then $0D

RESPONSE: Status

5.2.3.3.2.9 Get Buffered Data

```
COMMAND:     Get buffered data.
FORMAT:          cc --> $29
                dd1 --> $80 + v
                        v:
                            bit 0 --> buffer 0  (sensors 0-3)
                            bit 1 --> buffer 1  (sensors 4-7)
                            bit 2 --> buffer 2  (sensors 8-11)
                            bit 3 --> buffer 3  (sensors 12-15)
                            bit 4 --> don't care
                            bit 5 --> don't care
                            bit 6 --> don't care
                dd2 --> $80
RESPONSE:    $20
             $ssssss - three bytes of status
             $8F - dip switch setting for SIC card (if not $8F, SIC not installed).
             $nnnn - position in loop counter
             $kk - times through the loop counter
             $mm - breakpoint
             $22 - start of data
             For each bit 0-3 of v (above) set the following:
                        14 Switchwords
                        1 GC Data Valid word
                        Temperature Sensorword of [bit[1]]TM
                        Temperature Sensorword of [bit[1]]TC
                        Temperature Sensorword of [bit[2]]TM
                        Temperature Sensorword of [bit[2]]TC
                        Temperature Sensorword of [bit[3]]TM
                        Temperature Sensorword of [bit[3]]TC
                        Temperature Sensorword of [bit[4]]TM
                        Temperature Sensorword of [bit[4]]TC
                        Frequency Sensorword of [bit[1]]
                        Sensorword-Set of [bit[1]]
                        Frequency sensorword of [bit[2]]
                        Sensorword-Set of [bit[2]]
                        Frequency Sensorword of [bit[3]]
                        Sensorword-Set of [bit[3]]
                        Frequency Sensorword of [bit[4]]
                        Sensorword-Set of [bit[4]]
             $22 - end of data
```

5.2.3.3.2.10 Trigger Once Buffer

COMMAND: Trigger once buffer.

FORMAT:
```
      cc --> $2A
     dd1 --> $80
     dd2 --> $80
```

RESPONSE: Status

5.2.3.3.2.11 Get Power Factor Sign

COMMAND: Get power factor sign. To calculate the power factor use pf1=[Pafg1/[Vrms1*Irms1]]. Use the same calculation to determine pf2 using Pavg2, Vrms2, and Irms2; if pf2 < pf1 denotes capacitive loading; if pf2 >= pf1 denotes inductive loading; i.e. voltage leading current.

FORMAT:
```
      cc --> $2B
     dd1 --> $80 + 0-$F depending on sensor pair used
     dd2 --> $80
```

RESPONSE: 6 sensor words:
```
     V rms 1
     I rms 1
     P real 1
     V rms 2
     I rms 2
     P real 2
```
Status

5.2.3.3.2.12 Get Data For Switch

COMMAND: Get data for one specified switch a specified number of times.

FORMAT:

cc --> $2C

dd1 --> $80 + j

j: 1-$7F depending on the number of times data is to be taken -- input buffer must be taken into account.

dd2 --> $80 + k

k: 0-$D depending on the switch specified.

RESPONSE: j Switchwords

Status

5.2.3.3.2.13 Get Data For Sensor

COMMAND: Get data for one specified sensor a specified number of times.

FORMAT:

cc --> $2D

dd1 --> $80 + j

j: 1-$EF depending on the number of times data is to be taken.

dd2 --> $80 + k

k: 0-$F depending on the sensor specified.

RESPONSE: j Sensorword-Sets for sensor k

Status

5.2.3.3.2.14 Conditionally Command Switch On

COMMAND: Command switch on checking switch on or tripped status first; if any of the above conditions exist, the switch command for that particular switch or switches is not executed.

FORMAT:

```
      cc --> $2E
     dd1 --> $80 + j
             j:
                 bit 0 --> switch 0
                 bit 1 --> switch 1
                 bit 2 --> switch 2
                 bit 3 --> switch 3
                 bit 4 --> switch 4
                 bit 5 --> switch 5
                 bit 6 --> switch 6
     dd2 --> $80 + k
             k:
                 bit 0 --> switch 7
                 bit 1 --> switch 8
                 bit 2 --> switch 9
                 bit 3 --> switch 10
                 bit 4 --> switch 11
                 bit 5 --> switch 12
                 bit 6 --> switch 13
```

RESPONSE: Status

5.2.3.3.2.15 Conditionally Command Switch Off

COMMAND: Command switch off checking switch off or tripped status first; if any of the above conditions exist, the switch command for that particular switch or switches is not executed.

FORMAT:

```
 cc --> $2F
dd1 --> $80 + j
        j:
            bit 0 -->  switch 0
            bit 1 -->  switch 1
            bit 2 -->  switch 2
            bit 3 -->  switch 3
            bit 4 -->  switch 4
            bit 5 -->  switch 5
            bit 6 -->  switch 6
dd2 --> $80 + k
        k:
            bit 0 -->  switch 7
            bit 1 -->  switch 8
            bit 2 -->  switch 9
            bit 3 -->  switch 10
            bit 4 -->  switch 11
            bit 5 -->  switch 12
            bit 6 -->  switch 13
```

RESPONSE: Status

5.2.3.3.2.16 Get Data For All Switches

COMMAND: Get data for all fourteen switches a specified number of times.

FORMAT:

```
        cc -->  $30
       dd1 -->  $80 + j
                j: 1-$7F depending on the number of times data is to be taken --
                   input buffer size must be taken into account.
       dd2 -->  $80
```

RESPONSE:

```
j times:
                14 Switchwords
                GC Data Valid word
        Status
```

5.2.3.3.2.17 Get Data For All Sensors

COMMAND: Get data for all sixteen sensors one time.

FORMAT:

```
        cc -->  $31
       dd1 -->  $80
       dd2 -->  $80
```

RESPONSE:

16 Sensorword-Sets
Status

5.2.3.3.2.18 Get Data For All Temperature Sensors

COMMAND:	Get all 16 temperature sensor readings one time.
FORMAT:	cc --> $32
	dd1 --> $80
	dd2 --> $80
RESPONSE:	16 * 2 Sensor words for the temperature sensors
	Status

5.2.3.3.2.19 Get All 16 Power Factors and Signs

COMMAND:	Get all 16 power factors and signs.
FORMAT:	cc --> $33
	dd1 --> $80
	dd2 --> $80
RESPONSE:	16 times (six Sensor words): V rms 1 I rms 1 P real 1 V rms 2 I rms 2 P real 2
	Status

5.2.3.4 System-wide Dataflows

This part describes the overall dataflow of SSM/PMAD under operating conditions. Six figures are presented. The first three represent initialization of SSM/PMAD. The fourth represents normal operation. The fifth represents data flow in a source reduction contingency situation while the sixth represents data flow in a contingency arising from a problem in the hardware.

The figures are depicted with boxes representing the locations of computing elements. The items between the boxes represent transactions that are sent between the computing elements. Where important, numbers on the left hand side of the transactions indicate ordering information among the transactions. The numbers in parantheses on the right of the transactions are the id numbers of the transactions. For example, in the first figure below, first a ready? message is sent from the Symbolics to FRAMES and the CAC. Second, the CAC both responds to the Symbolics with a ready! and an LLP availability to FRAMES. FRAMES also responds with a ready! transaction to the Symbolics. The other figures are interpreted analogously.

5.2.3.4.1 Initialization Dataflow

Figure 5.2.3.4.1-1 shows the Symbolics 3620 D asking if FRAMES and the CAC are ready. FRAMES and the CAC both respond appropriately. In addition, at this time, the CAC notifies FRAMES which LLPs are available.

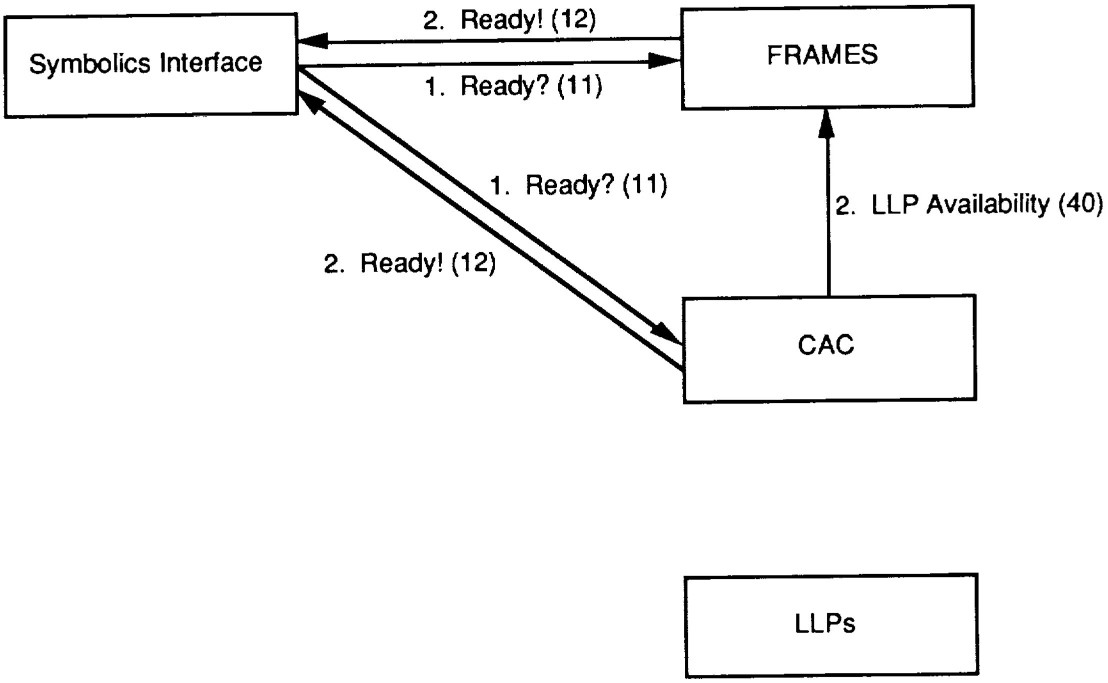


Figure 5.2.3.4.1-1 Initialization Dataflow Part 1

Once FRAMES and the CAC have indicated that they are ready the Symbolics 3620 D sends down the initial event list and load priority list as reflected in Figure 5.2.3.4.1-2. Upon receiving and processing these lists, FRAMES and the CAC respond with the initialized message.

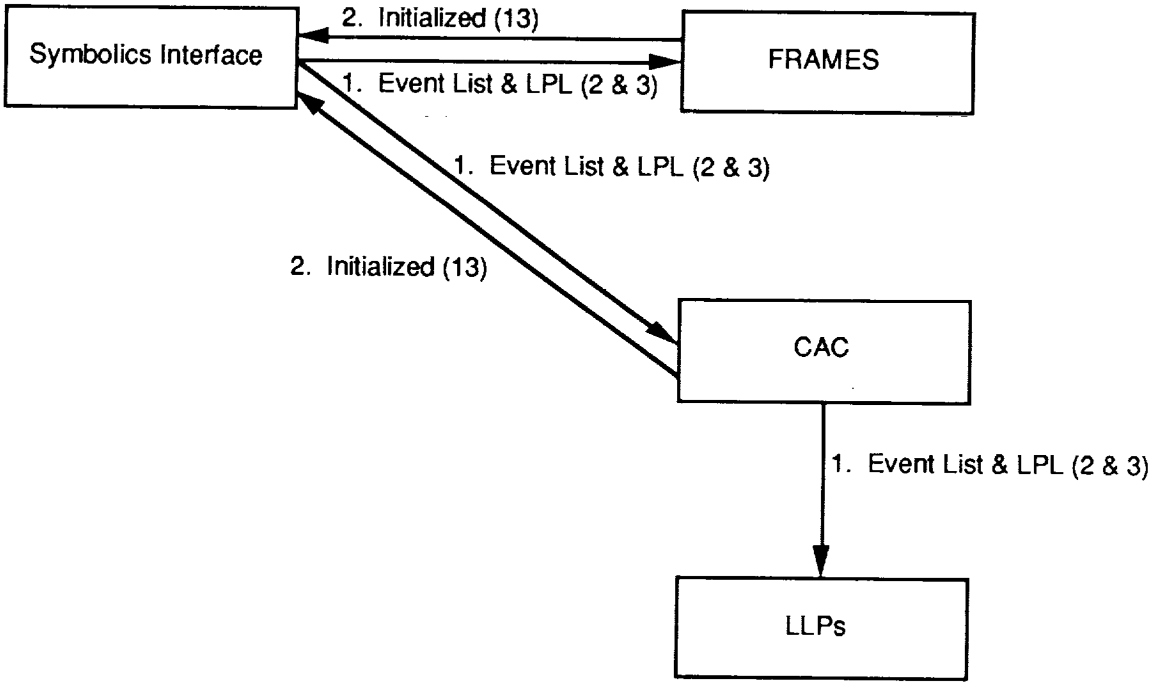

Symbolics Interface
2. Initialized (13)
FRAMES
1. Event List & LPL (2 & 3)
1. Event List & LPL (2 & 3)
2. Initialized (13)
CAC
1. Event List & LPL (2 & 3)
LLPs

Figure 5.2.3.4.1-2 Initialization Dataflow Part 2

When the Symbolics 3620 D has received the initialization messages from FRAMES and the CAC, it sends down the sync time message. At this point normal operation is entered. (Figure 5.2.3.4.1-3)

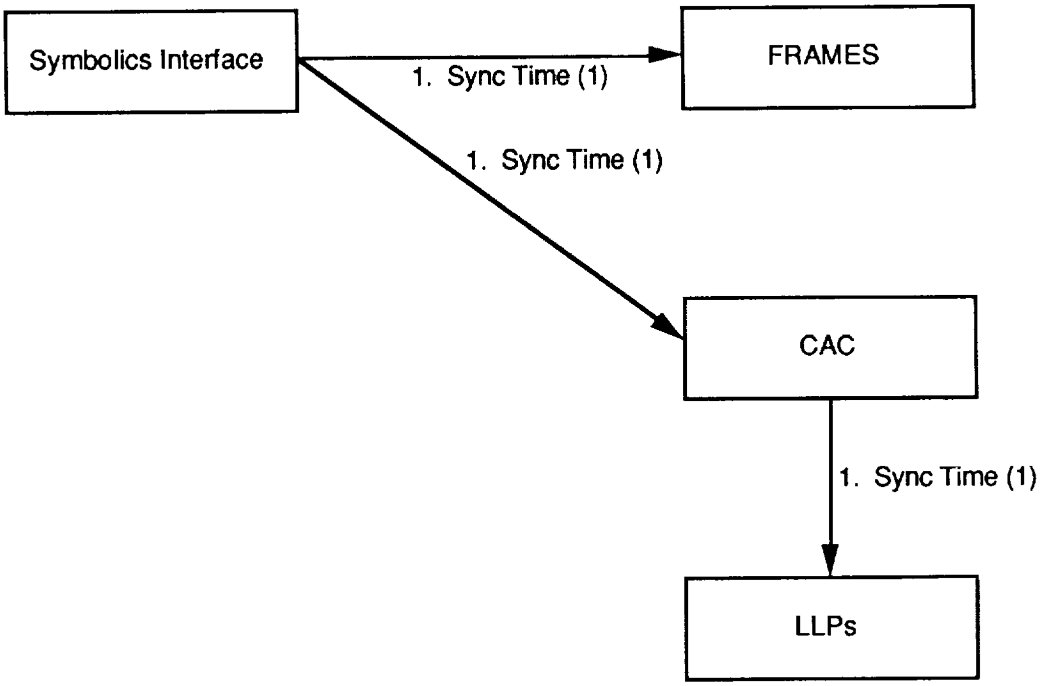


Figure 5.2.3.4.1-3 Initialization Dataflow Part 3

5.2.3.4.2 Normal Operation Dataflow

Figure 5.2.3.4.2-1 shows normal operation of SSM/PMAD. In normal operation a variety of switch and sensor data transactions are sent to FRAMES. FRAMES collects and processes this data and periodically reports utilization data to the Symbolics 3620 D.

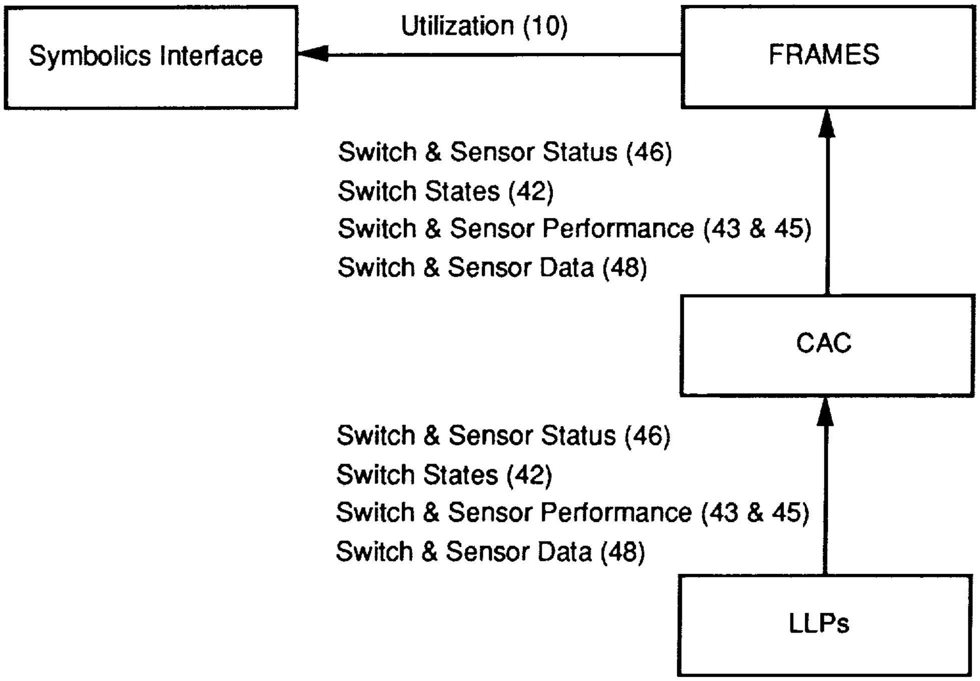

Symbolics Interface
Utilization (10)
FRAMES
Switch & Sensor Status (46)
Switch States (42)
Switch & Sensor Performance (43 & 45)
Switch & Sensor Data (48)
CAC
Switch & Sensor Status (46)
Switch States (42)
Switch & Sensor Performance (43 & 45)
Switch & Sensor Data (48)
LLPs

Figure 5.2.3.4.2-1 Normal Operation Dataflow

5.2.3.4.3 Contingency Situation Dataflows

Figure 5.2.3.4.3-1 represents the data flow in a source power reduction contingency. First the source power change is propagated down to the LLPs. This indicates a contingency situation which FRAMES recognizes and tells the Symbolics 3620 D. When the LLPs implement the source power change they report back the results via the source reduction status transaction. Any load sheds or switch to redundants are reported back to the Symbolics 3620 D followed by a contingency end transaction. This triggers the Symbolics 3620 D to process the contingency and send a contingency events list and load priority list to FRAMES and the CAC.

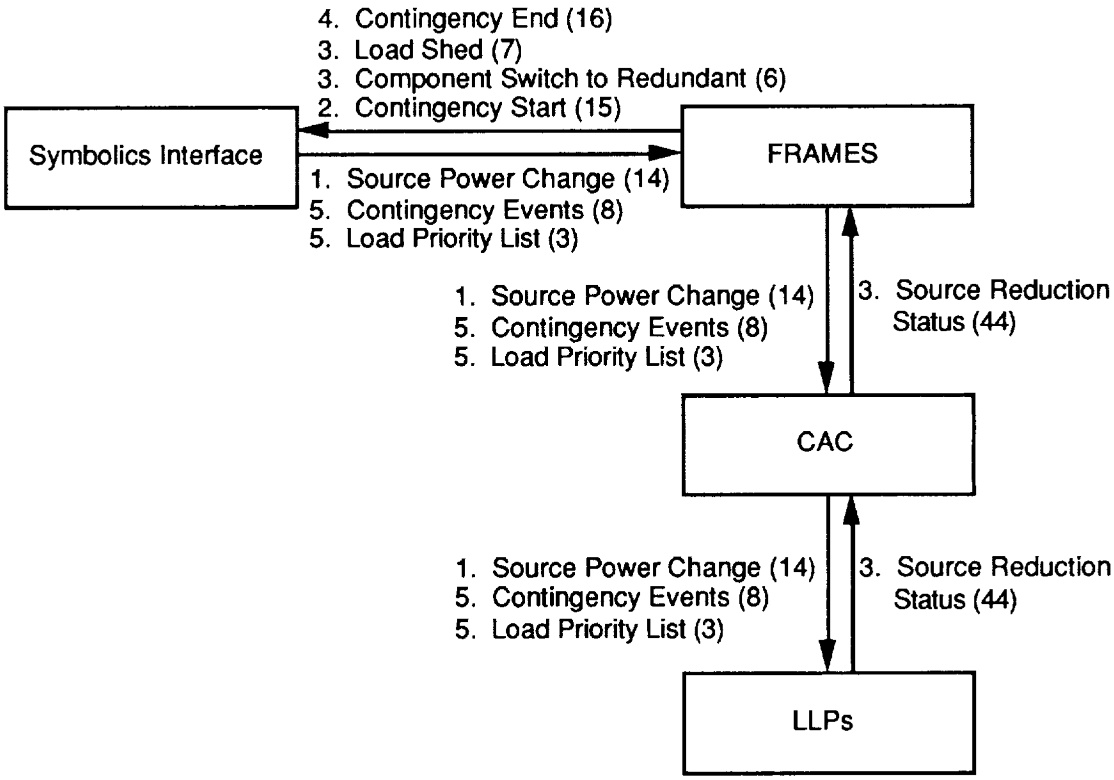

4. Contingency End (16)
3. Load Shed (7)
3. Component Switch to Redundant (6)
2. Contingency Start (15)
Symbolics Interface
FRAMES
1. Source Power Change (14)
5. Contingency Events (8)
5. Load Priority List (3)
1. Source Power Change (14)
5. Contingency Events (8)
5. Load Priority List (3)
3. Source Reduction Status (44)
CAC
1. Source Power Change (14)
5. Contingency Events (8)
5. Load Priority List (3)
3. Source Reduction Status (44)
LLPs

Figure 5.2.3.4.3-1 Source Power Contingency Dataflow

In the contingency situation where a fault occurs in the hardware, Figure 5.2.3.4.3-2 represents the resulting data flow. First, symptoms are detected via the switch and sensor status transaction from the LLPs. This indicates to FRAMES that there is a contingency and FRAMES sends a contingency start to the Symbolics 3620 D and necessary fault event lists to the LLPs to isolate the fault. The LLPs respond with fault event status transactions. When the fault is diagnosed by FRAMES, it sends load shed, component switch to redundant, and out of service messages to the Symbolics 3620 D followed by the contingency end message. The Symbolics 3620 D then processes the contingency and sends a contingency events and load priority list to FRAMES and the CAC.

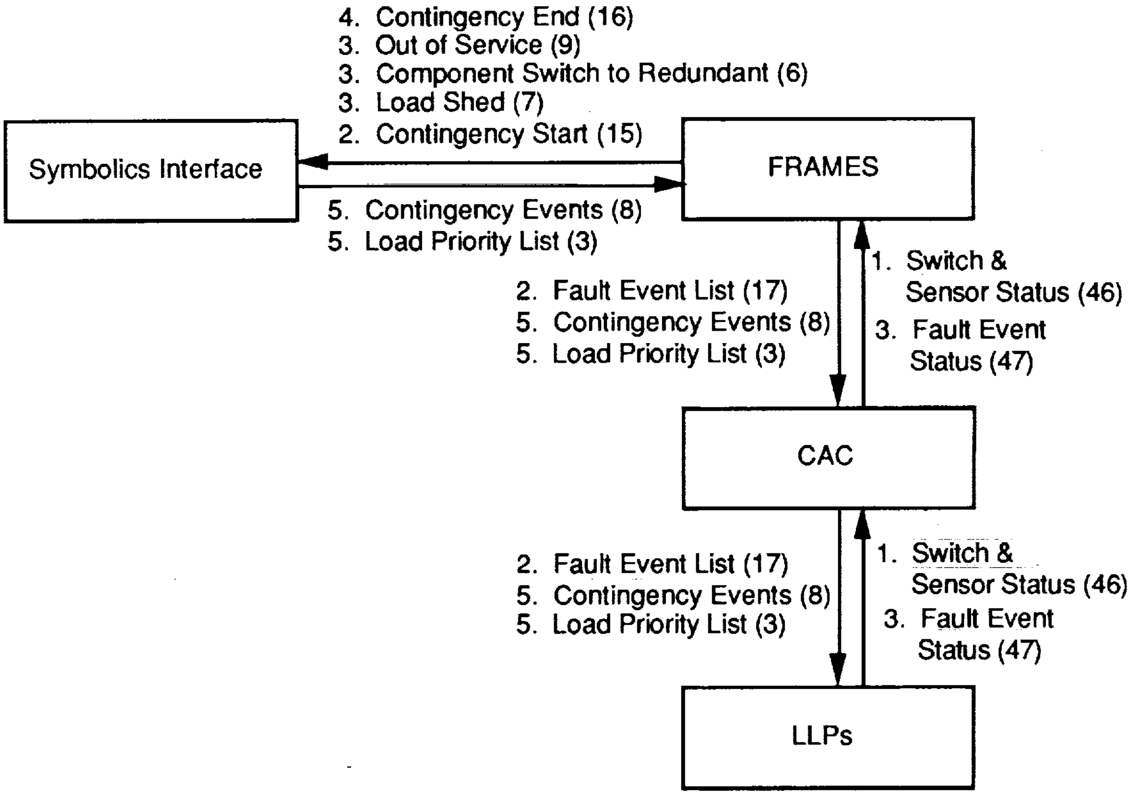

4. Contingency End (16)
3. Out of Service (9)
3. Component Switch to Redundant (6)
3. Load Shed (7)
2. Contingency Start (15)
Symbolics Interface
FRAMES
5. Contingency Events (8)
5. Load Priority List (3)
2. Fault Event List (17)
5. Contingency Events (8)
5. Load Priority List (3)
1. Switch & Sensor Status (46)
3. Fault Event Status (47)
CAC
2. Fault Event List (17)
5. Contingency Events (8)
5. Load Priority List (3)
1. Switch & Sensor Status (46)
3. Fault Event Status (47)
LLPs

Figure 5.2.3.4.3-2 Power System Contingency Dataflow

5.2.4 Breadboard Grounding Scheme

The SSM/PMAD breadboard was grounded with the objective of isolating high frequency noise generated from the 20kHz power source from the automation hardware. The output of the 20kHz source[1], the switchgear, and the loads were floated relative to ground to prevent high frequency noise from propagating through ground to the automation hardware. The automation hardware and switchgear control cards (SICs, GCs and A/Ds) all draw power directly from single-point grounded facility power (120 V ac, 60 Hz). All data communication cables have external shields which are grounded at the end away from the Motorola VME/10 (the VME/10 is particularly susceptible to noise in the system). Additionally, common mode filters and RC snubbers were added in strategic locations throughout the power system to minimize switching transient noise.

Note: In December of 1988 a Change Request to implement the Space Station Freedom power system as a 120 Volt dc system was signed into effect.

5.3 Breadboard Theory of Operation

The SSM/PMAD automation breadboard system incorporates both hardware and software activities. When considering the breadboard for its automation purpose, understanding software functional philosophy and how allocation within the hardware is accomplished becomes necessary for understanding its theory of operation.

The hardware activities center on computational and control engines and, especially on those types of activities which are concerned with Artificial Intelligence (AI) and the hardware needed to execute concurrent activities (these are activities which require a single goal from more than one cooperating knowledge based or expert system agent, which may also use the services of deterministic functions within their processing activities). Theory of operation for the hardware components of the SSM/PMAD is explained in Appendixes VII and VIII and in the vendor-supplied hardware manuals.

The software activities focus on supplying autonomous management and control to the system. The software appears in both deterministic and AI forms.

Note: For those who are interested in instruction for actual physical operation of the breadboard, reference is made to the SSM/PMAD Test Plan in section 5.12 and the Breadboard Usage Plan in Appendix VIII.

5.3.1 The SSM/PMAD Goal

In order to operate spacecraft, power systems must exist to supply the energy needed for the various components and subsystems to carry out their work. Up to now, these power systems were either managed by ground systems personnel performing planning and scheduling for the activities to be carried out by the spacecraft, or were managed by flight crew personnel carrying out the same activities on-board the space vehicle. In either case, a priori knowledge of the initial plan did not guarantee the production of a sound manageable power usage schedule, and the efforts of many people were necessary to complete the required iterations to produce a manageable power usage plan for a given mission profile.

In addition to this, power usage contingencies arise within practically all missions. Planning under the conditions of a contingency often does not allow for the key personnel or the time needed to complete the task in a safe manner, regarding the appropriate priorities and how they may change with respect to time and conditions. It is generally agreed that an expert who handles the management of a contingency replanning activity does so by knowing what the important system factors are, and by tracing through those factors until arriving at a safe and acceptable plan.

The primary goal of the SSM/PMAD is to autonomously provide, manage, and update as needed an appropriate, autonomously supplied power usage schedule (reflecting the needs of loads and their respective priorities), whether under nominal conditions or a contingency. This means that the loads are provided power in the best way that the automation system can provide. The line of reasoning within each knowledge processing environment of the SSM/PMAD instills this goal, and the deterministic processing supports it. Hence, the system has one direction and one philosophy; to simply implement and support the goal.

5.3.2 Theory of Functional Division

Software within the SSM/PMAD is partitioned into separate divisions based upon functional needs and differences (see the Task I Study Report in Appendix I). All deterministic functionality was assigned to algorithmic (closed-form) software functions. These functions exist primarily on the Communications and Algorithmic Controller (CAC) at the Motorola VME/10 or the Lowest Level Processors (LLPs). Software functions which would not necessarily have assured outcomes based upon given inputs (not closed-form) were assigned to knowledge based activities.

Some of the knowledge based activities were deemed expert systems due to their superior knowledge of the breadboard. For instance, the Load Priority List Management System (LPLMS) is a knowledge based activity along with the Front End Load Enable Scheduler (FELES). This is because they exercise general level knowledge in arriving at their conclusions. The Fault Recovery And Management Expert System (FRAMES) and MAESTRO however, are expert systems because they possess specific knowledge and expertise about the power automation breadboard, and they exercise this knowledge in arriving at their conclusions.

One of the general results of the functional partitioning was that software activities were generally allocated wholly within a single computational and control environment. There were two instances where this was not feasible. First, in controlling the communications upward from the LLPs to the Xerox 1186 there was need for a mediating activity. This activity took the form of a buffering extension in the Communications and Algorithmic Software (CAS). Second, in the lowest level diagnostic activities of FRAMES there was a need to interface to the Switch Interface Controllers (SICs) directly to provide range testing on data and to perform immediate testing for soft faults. This could not be accomplished at the Xerox 1186 and was therefore allocated as deterministic functions to the Lowest Level Functions (LLFs).

Table 5.3.2-1 displays the functional content for the allocations which were made for the above described reasons.

TABLE 5.3.2-1 Breadboard Functional Content

NAME	LOCATION	FUNCTION DESCRIPTION
FELES	Symbolics 3620 D	User front end and scheduling interface
MAESTRO	Symbolics 3620 D	Load scheduling
LPLMS	Symbolics 3620 D	Initialing and managing load priorities
FRAMES	Xerox 1186 & LLP	Fault diagnosis and contingency management
CAS	VME/10 (CAC)	Communications & execution list management
LLFs	LLPs	Lower level load management and reporting

The following subsections describe the theory of how the software entities function within the breadboard environment and what their individual goals are. The sum of the software component functional philosophies is representative of the breadboard theory of operation.

5.3.2.1 The FELES Functional Philosophy

The FELES serves as the user's front end to the MAESTRO scheduler in particular and to the Symbolics 3620 D in general. The FELES is viewed as possessing a generalized knowledge content necessary to complete its assigned task. It is therefore a knowledge based system. In theory it owes direction to the user without specific user request. And to take advantage of the I/O devices possessed by the Symbolics 3620 D, the user interface is mouse-sensitive icon and menu oriented not requiring the user to commit to keyboard input. This provides a user setup of MAESTRO without having to be cognizant of its complex functionality. The FELES sets that up for the user.

5.3.2.2 The LPLMS Functional Philosophy

Management of priorities for a multiple output power distribution system is quite a dynamic problem. Sometimes there is no clear-cut approach to a solution so employing a heuristic is desirable. This is the approach taken by the LPLMS. Also, the updating of priority lists should be maintained on a time-line that is reasonably close to some measure of how often loads may change their energy usage or switch off or on within the system. A figure of 15 minutes was arrived at based upon experts' experiences with previous operational spacecraft power systems. Therefore, the LPLMS reviews the load priority list at least on a regular 15 minute time basis. Also, when contingencies arise the entire priority list may change. So the LPLMS must also review and update the priority list based upon urgent system need. This capability is achieved as a result of a request from FRAMES when an unplanned system condition such as a fault is detected. The request is handled by the LPLMS, and a new load priority list is generated.

5.3.2.3 The FRAMES Functional Philosophy

FRAMES is the heart of the SSMPMAD system. It provides the complex management of the entire system, including setting up the activities of the other knowledge based functions. FRAMES maintains the power system status, tracking each opening and closing of the various power distribution switches. All information flowing up from the LLPs is reviewed for content and meaning from an integrated system point of view within FRAMES. Whenever hard-fault information is transmitted to FRAMES from the LLPs, the affected portions of the system are analyzed in order to diagnose the fault and its cause. FRAMES plans switchgear execution in order to carry out the diagnosis. A basic model of the breadboard must be contained and understood within FRAMES in order to accomplish these activities. FRAMES must update the elements on the Symbolics 3620 D with pertinent information and must inform MAESTRO and LPLMS of the system status with respect to the need of their activity. FRAMES also provides a user interface which is icon and menu driven with user inputs coming from a mouse. The basic screen image is that of the power system topology with enhancements added for displaying switch status.

5.3.2.4 The CAS Functional Philosophy

The CAS functions in three basic ways. First, all lists bound for the LLPs are separated and sent to the appropriate processors. The types of lists sent to the LLPs represent events, priorities, and maximum power values during periods of source reduction. Second, it buffers all data flowing from the LLPs to the Xerox 1186. Third it provides the user-interface during manual system control.

5.3.2.5 The LLF Functional Philosophy

The LLFs exist within two environments; the power distribution control unit (PDCU) and the load center (LC). For both environments they carry out numerous activities. However, functionality is not completely uniform between the two distribution controllers. For both, the scheduled operations of switching activities are carried out. They both acquire and process switch and sensor data in the form of current and voltage (temperature data from the sensors is not currently used), and pass that data to the CAC which is eventually destined for the parent FRAMES. They also perform range testing and fault testing on the data, providing notification to the parent FRAMES in the event anything out of limits or faulted is found. Presently the testing at the PDCU is expanded to handle soft fault detection. Also, both environments calculate and pass the system performance statistics.

5.3.3 Operational Theory Integration

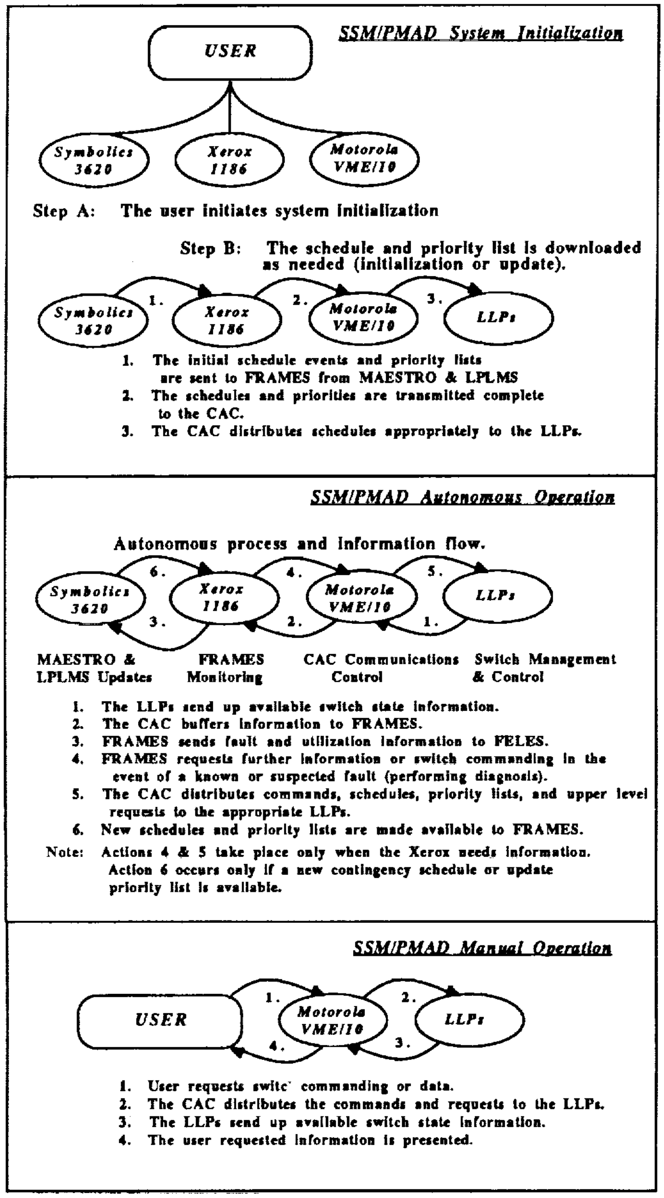
Figure 5.3.3-1 shows the software flow for the SSM/PMAD. The high level data interchange between entities demonstrates the activities which take place in the operation of the breadboard. A single user can operate the entire system. The only critical user functions are to initiate the system's start state. From then on, the system is autonomous in its operation, including contingency management. The user is able to request viewing of the latest available data from the Xerox 1186 user-interface. Detailed information about the schedule of events is available from the Symbolics 3620 D screen.

The user-interface at the Motorola VME/10 should only be exercised from the manual mode.

In the manual mode, all knowledge based components of the system become inactive, and the highest level software functions exist in the Motorola VME/10 for user interface activity. The highest level software switch control functions in the manual mode exist in the LLPs where the actual switch control is commanded based upon the user's request. The CAC still maintains LLP communication management functions. Ultimately, as in more conventional systems, the user is the highest level control function when the SSM/PMAD system is in manual control mode.

A description of what happens with the SSM/PMAD when the system is started and run is as follows (refer to the flow in Figure 5.3.3-1):

1) The user initializes the Symbolics 3620 D; MAESTRO creates a schedule of events list; LPLMS creates an associated priority list. Communications are enabled.
2) The user initializes the Xerox 1186 enabling FRAMES and communications.
3) The user initializes the Motorola VME/10, enabling communications and downloading software to the LLPs.
4) The user initiates system-wide execution and management of the events list.
5) FRAMES monitors all events within the system. Faults which occur are diagnosed. Contingencies are managed. New needed schedules are autonomously initiated, generated, implemented, and managed.
6) System status and statistics are autonomously maintained and reported.



SSM/PMAD System Initialization

Step A: The user initiates system initialization

Step B: The schedule and priority list is downloaded as needed (initialization or update).

1. The initial schedule events and priority lists are sent to FRAMES from MAESTRO & LPLMS
2. The schedules and priorities are transmitted complete to the CAC.
3. The CAC distributes schedules appropriately to the LLPs.

SSM/PMAD Autonomous Operation

Autonomous process and information flow.

MAESTRO & LPLMS Updates

FRAMES Monitoring

CAC Communications Control

Switch Management & Control

1. The LLPs send up available switch state information.
2. The CAC buffers information to FRAMES.
3. FRAMES sends fault and utilization information to FELES.
4. FRAMES requests further information or switch commanding in the event of a known or suspected fault (performing diagnosis).
5. The CAC distributes commands, schedules, priority lists, and upper level requests to the appropriate LLPs.
6. New schedules and priority lists are made available to FRAMES.

Note: Actions 4 & 5 take place only when the Xerox needs information. Action 6 occurs only if a new contingency schedule or update priority list is available.

SSM/PMAD Manual Operation

1. User requests switc' commanding or data.
2. The CAC distributes the commands and requests to the LLPs.
3. The LLPs send up available switch state information.
4. The user requested information is presented.

Note: LLP processing units are Motorola 68010 on 107 cards.

FIGURE 5.3.3-1 The SSM/PMAD Operational Flow

5.4 Resource Scheduling

Resource scheduling in SSM/PMAD is used to schedule activities requiring electrical power as will be needed on Space Station Freedom, making efficient use of available power while recognizing constraints and priorities. This section of the report describes the theory and implementation of the resource scheduler as implemented in MAESTRO. The last part of this section describes the user interface to MAESTRO.

5.4.1 Theory of Operation

The goal of scheduling is to map out when a set of tasks may be completed making efficient use of available resources. Resources in SSM/PMAD are switches, crew, tools, power, etc.

The scheduler, as implemented in MAESTRO, can be defined in four parts. The first part is the representation of activities. Activities are used to represent those types of tasks that are to be performed on Space Station Freedom like operations. The language and representation of constraints defines the second part. There can be constraints on the resources to be used by activities, as well as constraints between activities or the parts of activities. The actual schedule generation process defines the third part. Scheduling in light of contingencies defines the fourth part.

5.4.1.1 Activities

Activities within MAESTRO are represented hierarchically. An activity group is a set of activities representing different ways to accomplish a particular goal. An activity, in turn, is a linear sequence of subtasks which, when performed in the order specified, satisfy that goal. A subtask is a portion of an activity whose resources and conditions requirements do not vary over its duration. Duration can vary, as can delays between subtasks.

5.4.1.2 Constraints

Constraints represent conditions on or between activities. They arise for a variety of reasons. Resource types and availability give rise to rate-controlled and consumable resources. Certain requirements that need to be met for proper operation of a task define conditions on activities. Opportunity window constraints, representationally equivalent to conditions, are constraints not associated with a resource but necessary for the performance of an activity. These constraints, rate-controlled and consumable resources, conditions, and opportunity windows are all performance-controlling constraints.

Rate-controlled resources are those whose availability continues when the subtask using them ends. Examples of this are crew time, thermal rejection, electrical power and equipment. Consumable resources, on the other hand, once depleted, stay depleted until some activity specifically replenishes them. Water, liquid nitrogen and lubricating fluids are examples of this type of constraining resource.

Conditions are states the spacecraft must maintain in order to perform a subtask, and include spacecraft attitude and position, temperature ranges, acceleration, vibration, etc. In general, conditions cannot be consumed by an activity requiring them, which differentiates them from rate-controlled resources. An opportunity window is a performance-controlling constraint not associated with the availability of any resource, but constraining the performance of a subtask just like a condition constraint would. Activities with opportunity window constraints must have appropriate subtasks scheduled to happen within them.

Many of the performance-controlling constraints can be satisfied by more than one resource or condition. An example of this is the case where a subtask could be performed by either of two crew members trained to use a particular piece of equipment, but not by any of the other crew members. This is referred to as a resource disjunction, a case where one resource or another can satisfy a requirement. The existence of a resource disjunction in a subtask description greatly increases the difficulty of finding times during which a subtask can run, as opportunities to perform the subtask depend on which resource is chosen. This can be further complicated by the fact that a resource choice in one subtask can control that in another, e.g. the crew member who performs the calibration of an instrument should be the same one who read the manual at the start of the activity.

Another basic type of constraint on activities is the relational constraint. Constraints of this type relate the start or end of one subtask to that of another, either in the same activity or in another. The current version of MAESTRO only allows uni-directional constraints, in which a constraining subtask can run whether the constrained one can be scheduled while scheduling the constrained one depends on the scheduling of the other. A relational constraint may also constrain activities by relating the start or end of a subtask to some event or absolute time on the timeline.

5.4.1.3 Schedule Generation

MAESTRO creates a schedule by repeatedly executing three steps, referred to as the select-place-update cycle. The first step involves evaluating every activity requested for scheduling with respect to a set of selection criteria, and choosing one activity to put on the schedule next. These criteria include the base priority associated with each activity, the percentage of performances requested that have been scheduled for each (success level), and the relative constraint of each (opportunity). Relative constraint is a rough measure of how many different opportunities each activity has to be placed on the schedule. These criteria are combined using user-selectable weights which reflect the importance of each criterion to the user. An activity chosen will have higher priority, a lower percentage of requested performances scheduled, and/or fewer opportunities to be scheduled than other activities.

Once an activity has been chosen to be scheduled, one instance of it is placed on the schedule. The calculation resulting in the measure of constraint actually determines all allowable start and end times for all subtasks in each activity. This information can be used during placement to position the performance according to soft constraints (preferences) imposed by a user. The user, for example, maximize the data collection subtask, or can schedule the activity as early or as late in the scheduling period as possible. If there is a resource disjunction in a subtask's requirements, a preference can be specified and adhered to, and in fact, a set of possibly contradictory soft constraints can be specified along with an ordering in their importance.

The final step in the scheduling cycle involves updating resource availability profiles to reflect the activity's consumption of resources. The cycle then repeats for as long as the user wishes or until there are no opportunities to schedule any activity. The combination of weights on selection criteria and attention to soft constraints during placement allows the scheduler to be tuned for a variety of scenarios.

5.4.1.4 Contingency Operations

There are a number of situations in which a schedule must be altered in other ways to accommodate various changes. It may become known that resource or condition availabilities will change or have changed, or that an activity not previously known about needs to be added to the schedule. These situations are handled within MAESTRO by a heuristically-guided unscheduling mechanism in concert with a method of altering descriptions of activities already in progress, and aided by the maintenance during scheduling of multiple partial schedules. A change in resource availabilities may result in a projected over-use of a resource. When a resource is found to be overbooked, all activities using that resource during the time it is overbooked are evaluated. The evaluations are done according to a set of criteria designed to determine what activity to alter or unschedule to solve the problem. This should be done with the least impact on the schedule. The criteria include how well the activity's use of the resource fits the amount of overbooking, whether the activity is in progress or not, the activity's priority, amount of crew involvement, use of other resources, further opportunities to be scheduled, success level, and others. These criteria are also weighted to allow flexibility to a user. An activity is selected and unscheduled or selected and altered, then all are again evaluated and another unscheduled or altered, until no resource overbookings remain.

Activities whose condition constraints are violated must be altered or unscheduled; there is no choice as to which to affect. These are all handled the same as those chosen to be perturbed by a resource overbooking. When it is determined that an activity not scheduled must be added to the timeline, the scheduler first tries to find a way to schedule it which will not disturb anything already scheduled. If no opportunity exists, MAESTRO will try to find opportunities which will result in only lower-priority activities being perturbed, and if found, will unschedule or alter one or more of those using the same techniques it uses to handle overbookings. If no lower-priority activities can be found to bump, the scheduler rejects the request (perhaps the activity is not schedulable even in the absence of other activities, or all interfering activities are of a higher priority).

The last thing MAESTRO tries to do after altering the schedule in a contingency is to schedule any activities whose requests have not been fully met, possibly using resources released when some other activity was altered or unscheduled.

5.4.2 Implementation

To schedule activities for the power system the scheduler needs to know about the various resources involved, and needs to have a model of the utility power system. This section describes how the resources are allocated and the utility power system is represented. Following that, scheduling for Space Station Freedom module like power management and distribution is discussed. Finally, operation of the system from the point of view of the scheduler is described.

5.4.2.1 Resource Allocation

Although MAESTRO can handle an unlimited number of resources, several have been allocated for the demonstration of the SSM/PMAD breadboard. This set is representative, and in the case of the power system resources, necessary for the operational scenario presented by MAESTRO. Each of the components of the power system are represented as two types of resources: One, a piece of equipment, and two, a rate-controlled power resource. A rate-controlled resource CREW is allocated and other non-powered equipment General Purpose Handling Tools and Fluid Handling Tools are available. Several pieces of RPC specific test equipment have been defined for ease in setting up a schedule that exercises specific power system components. Each of the allocations listed is for the entire duration of the scheduling period chosen by the user in creating a schedule. The capability to modify and shape these resource availability profiles has not been implemented as part of the Symbolics 3620 D interface for the SSM/PMAD. The resources available are:

All Load Center and Subsystem Testers	18
All RPC testers	6
General Purpose Handling Tools	4
Fluid Handling Tools	4
Crew	2
All other equipment	1
Power system components (equipment)	1
Power system components (rate-controlled)	actual power capability

5.4.2.2 Utility Power

Typical spacecraft built and flown to date have a dedicated power system sized to handle any load the instruments and subsystems on board might normally require. The only limit useful for scheduling with regard to power consumption has been the source output capability (battery management is a complex issue for scheduling, but will not normally be a factor in SSM power distribution or consumption). The SSM's on Space Station Freedom will be unique with respect to PMAD design in that the power system will limit consumption at numerous points along any power path, and these limits are relevant to scheduling. In order to clarify this point it will be useful to describe the SSM/PMAD breadboard under development for Marshall Space Flight Center.

A representative schematic of a portion of this breadboard is shown on the following page: (Figure 5.4.2.2-1)

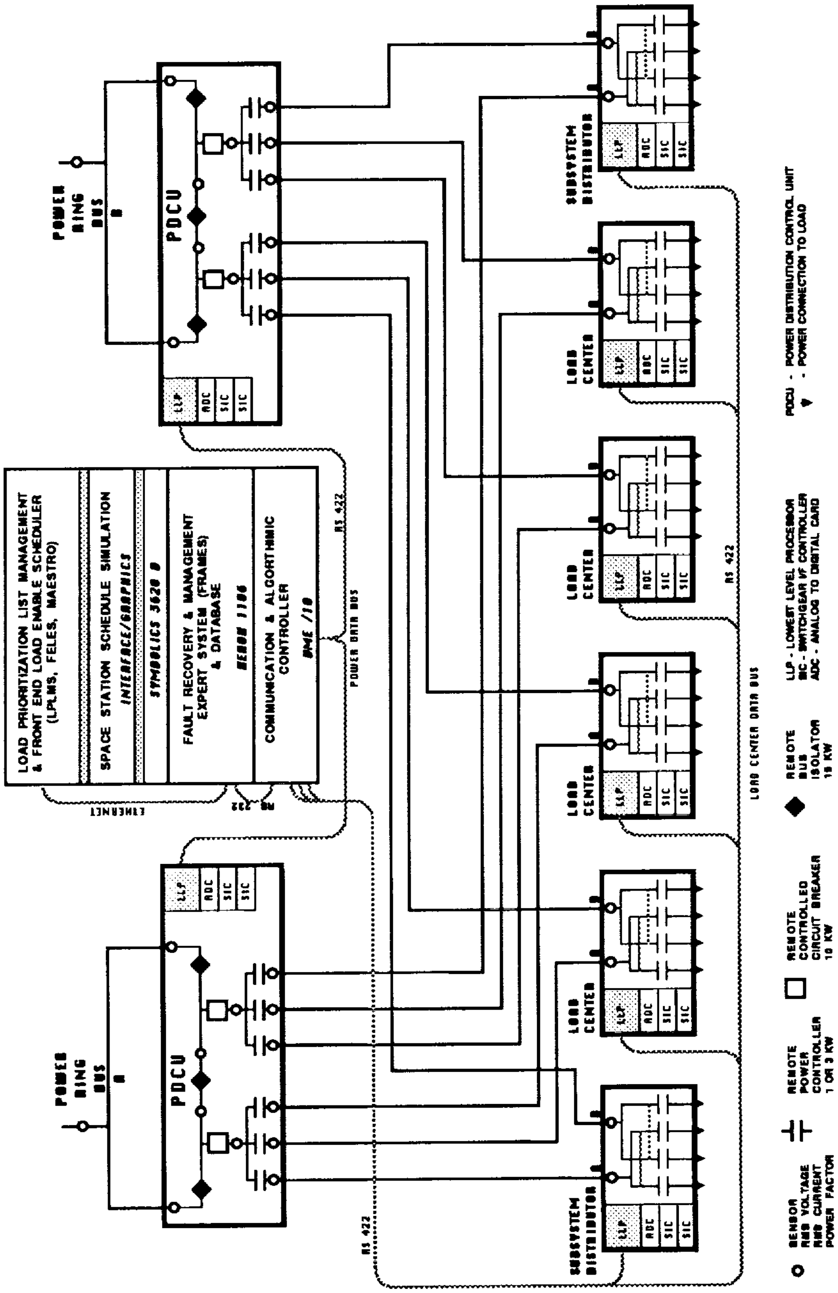

POWER RING BUS B
PDCU
LLP
ADC
SIC
SIC
LOAD PRIORITIZATION LIST MANAGEMENT & FRONT END LOAD ENABLE SCHEDULER (LPLMS, FELES, MAESTRO)
SPACE STATION SCHEDULE SIMULATION
INTERFACE/GRAPHICS
SYMBOLICS 3620 D
FAULT RECOVERY & MANAGEMENT EXPERT SYSTEM (FRAMES) & DATABASE
XEROX 1186
COMMUNICATION & ALGORTHMIC CONTROLLER
DME /10
ETHERNET
RS 232
RS 422
POWER DATA BUS
RS 422
SUBSYSTEM DISTRIBUTOR
LOAD CENTER
LOAD CENTER
LOAD CENTER
LOAD CENTER
SUBSYSTEM DISTRIBUTOR
RS 422
LOAD CENTER DATA BUS
○ SENSOR RMS VOLTAGE RMS CURRENT POWER FACTOR
REMOTE POWER CONTROLLER 1 OR 3 KW
□ REMOTE CONTROLLED CIRCUIT BREAKER 10 KW
◆ REMOTE BUS ISOLATOR 15 KW
LLP - LOWEST LEVEL PROCESSOR
SIC - SWITCHGEAR I/F CONTROLLER
ADC - ANALOG TO DIGITAL CARD
PDCU - POWER DISTRIBUTION CONTROL UNIT
▼ - POWER CONNECTION TO LOAD

Figure 5.4.2.2-1 SSM/PMAD Breadboard Diagram.

Power distribution hardware includes three different types of remotely controllable switches: 1) Remote Bus Isolator (RBI), 2) Remote Controlled Circuit Breaker (RCCB), and 3) Remote Power Controller (RPC).

RPC's provide solid state switching, fast circuit breaking in the case of hard faults, current limiting, I^2t fault tripping, under-voltage tripping and over-temperature tripping. All of the switches provide status reporting and measurement of current through the switch. Other sources of data include sensors located throughout the breadboard, providing current, voltage, frequency, power and power factor. Of note is the fact that more than three, 1-kilowatt RPC's are connected to a single 3-kilowatt RPC.

The functional elements of the PMAD system are depicted in Figure 5.4.2.2-2, the Symbolics 3620 D Interface Processing Architecture. A schedule is made available to the Front End Load Enable Scheduler (FELES), which determines RPC schedules detailing when power will be required of each, and how much. The Load Priority List Management System (LPLMS) uses the input schedule to create an ordered list of RPCs to shed (open) in the event power consumption must be reduced quickly. The RPC schedule and Load Priority List (LPL) are passed to the Fault Recovery and Management Expert System (FRAMES), the Communications and Algorithmic Controller (CAC), and onto the Lowest Level Processors (LLPs). Each LLP executes its portion of the schedule, allowing current through the switches to the amount specified in the schedule. When a fault is detected, FRAMES collects information about it, attempting to isolate it and determine the cause of the problem, then sends information to the FELES.

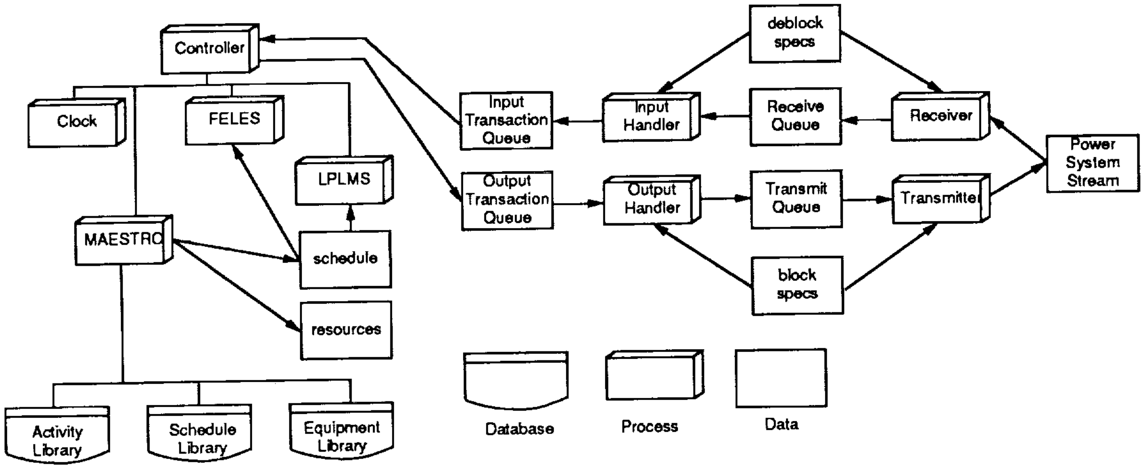


Figure 5.4.2.2-2 Symbolics Interface Processing Architecture

Symbolics 3620 D Interface Processing Architecture

In order to exercise the full potential for automation of this breadboard, the control and fault handling systems on the breadboard had to be interfaced with a scheduling and resource management system, and so MAESTRO has been modified to perform this function. Fault information is passed to MAESTRO, which reschedules such that the revised schedule takes into account the new state of the power system, and makes this schedule available to the FELES.

It is important to note that the power automation software has a very different view of a module and its activities from that of the scheduler. MAESTRO schedules activities which use various resources over durations in time. These resources include equipment which must be powered to operate, and that power is supplied through a set of RPCs within the power system. When a fault occurs, some piece(s) of equipment will be turned off, at least temporarily, so that some activity is interrupted. The power system does not "know" about activities or equipment, so the scheduler must ascertain the effects of some RPC becoming unavailable. To do so it must have a representation of the internals of the power system and the equipment connections to that system.

5.4.2.3 Scheduling for SSM/PMAD

The above power system design presents several challenges for scheduling of activities using power. These are for the most part derived from the fact that a large number of power system components are limiting resources. A number of 1-kilowatt RPCs (or in some cases 3-kilowatt RPCs) will be connected to a single, 3-kilowatt RPC, making that RPC a limiting resource. Thus, instead of scheduling against the availability of electrical power (a single resource), the system must schedule against the availability of as many as 100 power resources. Not only are these RPCs power resources, but they are also pieces of equipment which can become unavailable, raising the number of power related resources to around 200.

When specifying an activity to be scheduled, the user or scheduler must in some way determine which of these power resources will be used to accomplish each subtask, and what power level each resource will supply. Since power must be consumed by some piece of equipment, which itself is a resource represented to the scheduler, location and mode(s) of that equipment can determine paths and levels of power for each subtask, but the associations between equipment and power then must be represented. This increases the complexity of the already complex process of modelling each activity for the scheduler. Furthermore, there may be choices as to where a portable piece of equipment can be powered. Since it is the responsibility of the scheduler to determine resource use, the choice of this location must be made by the scheduler. This leads to a potentially large number of resource disjunctions, the choices between resources utilized as described previously. As was mentioned, these resource disjunctions complicate scheduling immensely. Perhaps the most significant impact on scheduling of this power system design is in the area of contingency operations. These involve a change in the assumptions upon which a schedule is based, causing the scheduler to try to adjust the schedule such that ongoing activities may continue and resources may be used efficiently. In order for the scheduler to perform this adjustment, it must be informed of the specific nature of the changes in assumptions.

Many high priority activities will have redundant power paths assigned to them. The power system is capable of automatically changing the resources used by a subtask, i.e. switching to a redundant path in the event that power cannot be supplied via the primary path, and these changes must be reflected in the description of the subtask. Thus, the scheduler must be able to automatically alter its activity models to account for this type of change.

Power consumption along redundant power paths cannot be scheduled, as this would result in extremely inefficient use of power. Therefore, when a fault occurs which causes the power system to make use of an alternate path to power something, a part of that path may already be fully allocated to one or more other subtasks. This requires that something be immediately turned off, interrupting subtasks not using power on the faulted power path. These interruptions are known as load shedding, and must also be communicated to the scheduler.

The scheduler must react not only to redundancy switching and load shedding, but also to immediate power reductions from outside the module and to user requests to schedule activities not previously requested. These situations all require revising the schedule as quickly as possible to accommodate changes. While the scheduler is making changes, the power system and other subsystems will be trying to continue execution of an old and possibly invalid schedule, which can result in a cascade of faults registered by these subsystems. Timing concerns therefore become a critical aspect of the contingency rescheduling problem. Also, there is a large volume of information to be passed between the scheduler and the various subsystems, necessitating a fairly direct communications path between these systems.

5.4.2.4 System Operation

Before going into a typical system operation scenario, it is necessary to analyze the effects of contingencies from the scheduling point of view. In this section, contingencies are discussed followed by an operational scenario.

5.4.2.4.1 Contingencies

Contingencies may occur after schedule generation in any of the following ways: Changes in resource availabilities; equipment breakdowns; changes in mission objectives; changes in target availabilities; and experiments' requirements alterations. Contingencies may result in invalid schedules with resource overbooking, inefficient schedules with opportunity to do more work, and partially completed activities which require some repair to fulfill mission objectives. Furthermore, there are three places where a contingency may occur with respect to a schedule: 1) Prior to schedule execution; 2) during schedule execution, with ample lag time between notification, and 3) occurrence; and immediate or nearly immediate contingencies.

In the first case, a contingency may occur prior to schedule execution as depicted in Figure 5.4.2.4.1-1 below:

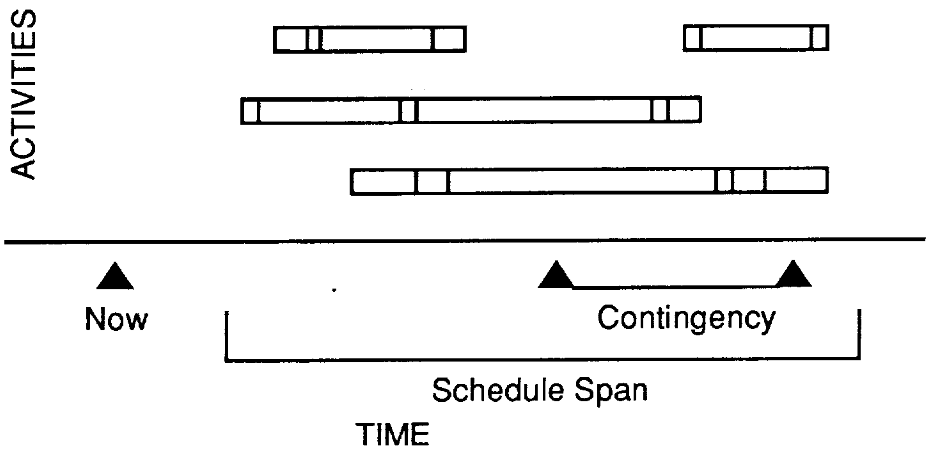


Figure 5.4.2.4.1-1 Contingency 1

There are three things to note about this case: 1) none of the candidate activities have been initiated; 2) notification occurs early enough so no real-time decisions need to be made; and 3) the schedule may be repaired by simply unscheduling and rescheduling selected activities.

In the second case, a contingency may occur during schedule execution as depicted in Figure 5.4.2.4.1-2 below:

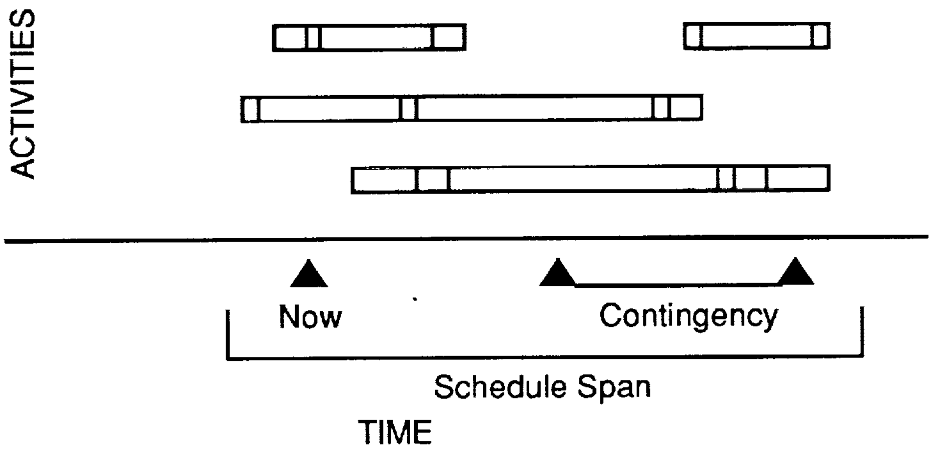


Figure 5.4.2.4.1-2 Contingency 2

When there is ample lag time between notification and occurrence, no immediate reactions are required, there may be a mix of activities, some not yet initiated and some in progress, and activities which are in progress may be restructured in some cases. In activity restructuring there may be a choice of activities to alter. In this case, intelligent decisions need to be made about which activities should be affected. Completed portions of selected activities remain on the schedule with the uncompleted portions removed. Resource profiles are updated to reflect removals. Finally, to restructure an activity any of the following mechanisms may be used: Resource switching, altering subtask or delay durations within the parameters of the normal activity model, and alter activity in the ways specified for contingencies in the activity model, e.g., interrupt, skip, and restart.

Finally, contingencies may occur in real-time or near real-time as depicted below in Figure 5.4.2.4.1-3:

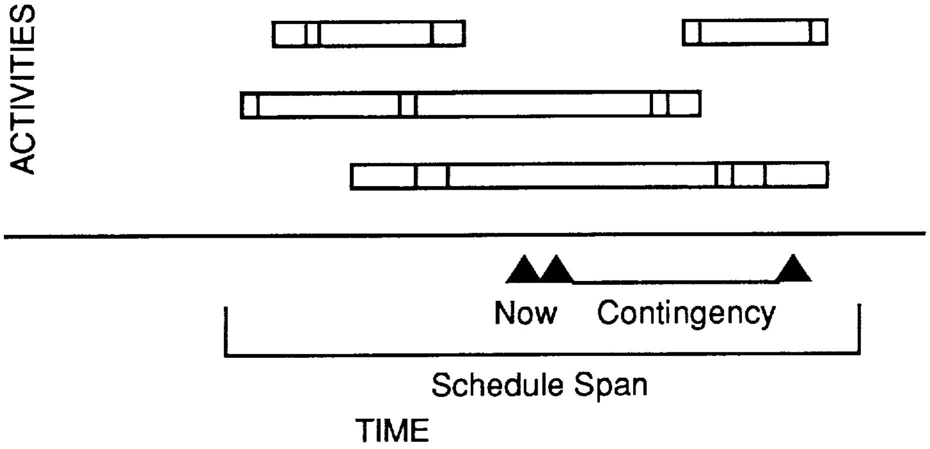


Figure 5.4.2.4.1-3 Contingency 3

In this case, quick reaction is required, there are few choices about which activities to affect and usually the affected activities will be in progress. In addition, changes will be effected on the schedule which are not under the control of the scheduler. The key point in this form of contingency is that coordination with subsystems, the crew, and ground personnel is essential.

5.4.2.4.2 Operation Scenario

5.4.2.4.2.1 Start Up

The user must define a schedule from which the load enablement commands may be generated. This may be accomplished through creating a new schedule or retrieving one from the Schedule Library.

When the schedule is available, the Symbolics 3620 D Interface is ready for starting the operation of the breadboard.

The Controller sends a READY? to FRAMES and CAC and waits for the response that both are READY!. The READY? message to each indicates that the fault diagnosis and algorithmic software should be reset and ready for autonomous breadboard operation.

Upon receiving confirmation that FRAMES and CAC are READY!, the initial event list and priority list are created and transmitted to each. The Controller now waits for the response of INITIALIZED! from each. The INITIALIZED! message indicates that the other software components have received the initial event and priority lists and are now ready to begin operation.

The user is now prompted for when to place the breadboard into action, from 1 to 55 minutes from now. This choice will send each software component the TIME-SYNChronization information, identifying the current time and the time the mission (breadboard operation) should start; the start of mission corresponds to mission time 00:00:00. The automation software is now active and power system commands will begin to be executed. Note that the system has a granularity of one minute.

5.4.2.4.2.2 Normal Operations

The system is now running autonomously. Any user interactions within this mode are only at the user's convenience, as the system management has been fully automated.

Every 15 minutes the LPLMS will generate a new priority list and distribute that to FRAMES and the CAC. The list is generated and transmitted approximately 5 minutes before it becomes effective.

Every 30 minutes the FELES will generate a new event list and distribute that to FRAMES and the CAC. The list is generated and transmitted approximately 5 minutes before its effective time.

5.4.2.4.2.3 Breadboard Fault

An RPC or RCCB trips, registering a fault. Automatically, FRAMES is notified, isolates the fault and determines the corrective action.

FRAMES notifies the Symbolics 3620 D Interface of the contingency situation by sending a CONTINGENCY-START message, indicating that the next set of information sent will be relevant to the current fault. All information sent to the Symbolics 3620 D Interface until a CONTINGENCY-END is received is considered applicable to the contingency, so that MAESTRO will not begin to act until all relevant data are accessible. FRAMES contingency information may consist of LOAD-SHEDs, OUT-OF-SERVICEs, SWITCH-TO-REDUNDANTs, or AVAILABILITY specifications.

During the contingency situation the LLPs should continue executing the load enable schedules for any non-faulted components. This is fully allowed by the segregation of partial FRAMES functionality into the lower level functions.

At this time, the Controller will stop the FELES and LPLMS from generating new lists until the contingency is handled by MAESTRO. The FELES and LPLMS wait to be restarted.

The Controller notifies MAESTRO of the contingency and passes the appropriate fault information. MAESTRO makes changes to the schedule such that all power interruptions are reflected in subtask descriptions as subtask interruptions, some of which can be continued, others restarted, and others halted, and such that subtask descriptions reflect loads switched to redundant sources.

Once MAESTRO has repaired the schedule, the Controller restarts the FELES and LPLMS. New event and priority lists are created and the state of all the components are specified. These are transmitted to FRAMES and the CAC.

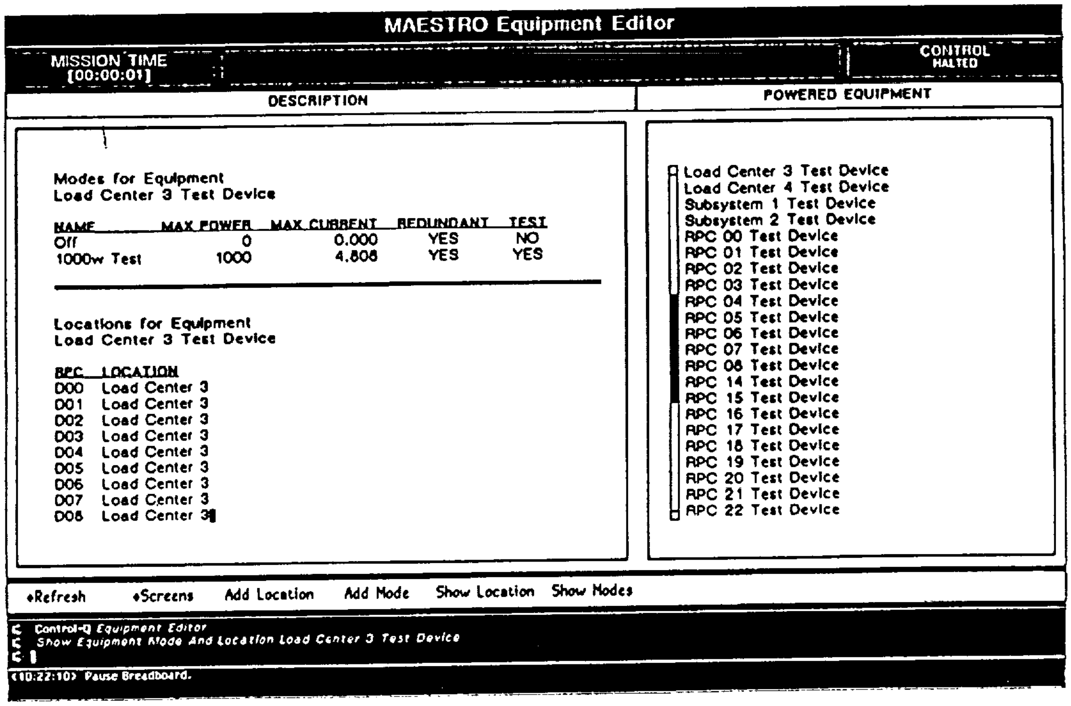
The Controller and the breadboard are now in a stable state and the Controller returns to normal operations. Any new fault situation will prompt the system to begin the same type of activities once again.

5.4.3 MAESTRO User Interface

This section describes the user interface to MAESTRO. It includes the user interfaces to the activity and equipment editors as activities and equipment are important to scheduling. The user interface of the Symbolics 3620 D Interface is described further in Appendix VI. Only the portions referred to will be described here.

5.4.3.1 The Equipment Editor

The Equipment Editor provides the mechanisms for adding modes of operation and locations to equipment. There are two main windows to the Equipment Editor. The Description Window is used for adding modes and locations to equipment, as well as for displaying information about equipment. The Powered Equipment display is a scrollable window of the available equipment. Each line on this display is mouseable. Figures 5.4.3.1-1 and 5.4.3.1-2 which follow reflect a sample display of the Equipment Editor and the Equipment Editor Help screen, respectively.



Modes for Equipment
Load Center 3 Test Device

NAME	MAX POWER	MAX CURRENT	REDUNDANT	TEST
Off	0	0.000	YES	NO
1000w Test	1000	4.808	YES	YES

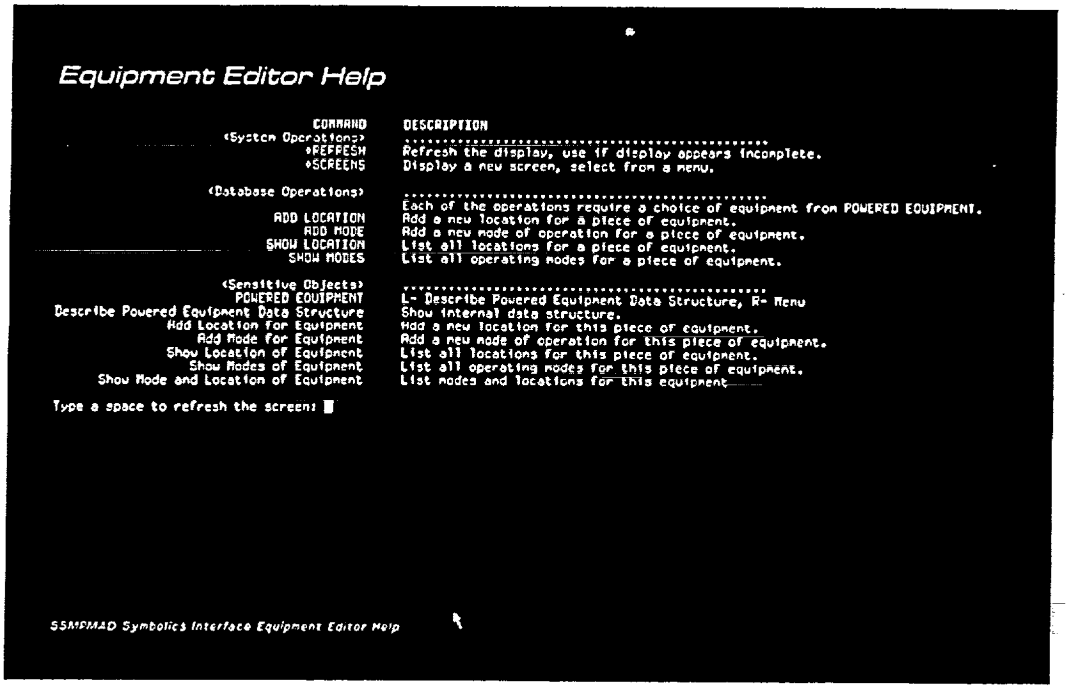
Locations for Equipment
Load Center 3 Test Device

RPC	LOCATION
D00	Load Center 3
D01	Load Center 3
D02	Load Center 3
D03	Load Center 3
D04	Load Center 3
D05	Load Center 3
D06	Load Center 3
D07	Load Center 3
D08	Load Center 3

Load Center 3 Test Device
Load Center 4 Test Device
Subsystem 1 Test Device
Subsystem 2 Test Device
RPC 00 Test Device
RPC 01 Test Device
RPC 02 Test Device
RPC 03 Test Device
RPC 04 Test Device
RPC 05 Test Device
RPC 06 Test Device
RPC 07 Test Device
RPC 08 Test Device
RPC 14 Test Device
RPC 15 Test Device
RPC 16 Test Device
RPC 17 Test Device
RPC 18 Test Device
RPC 19 Test Device
RPC 20 Test Device
RPC 21 Test Device
RPC 22 Test Device

```
C  Control-Q Equipment Editor
C  Show Equipment Mode And Location Load Center 3 Test Device
C
<10:22:10> Pause Breadboard.
```

Figure 5.4.3.1-1 Equipment Editor Screen

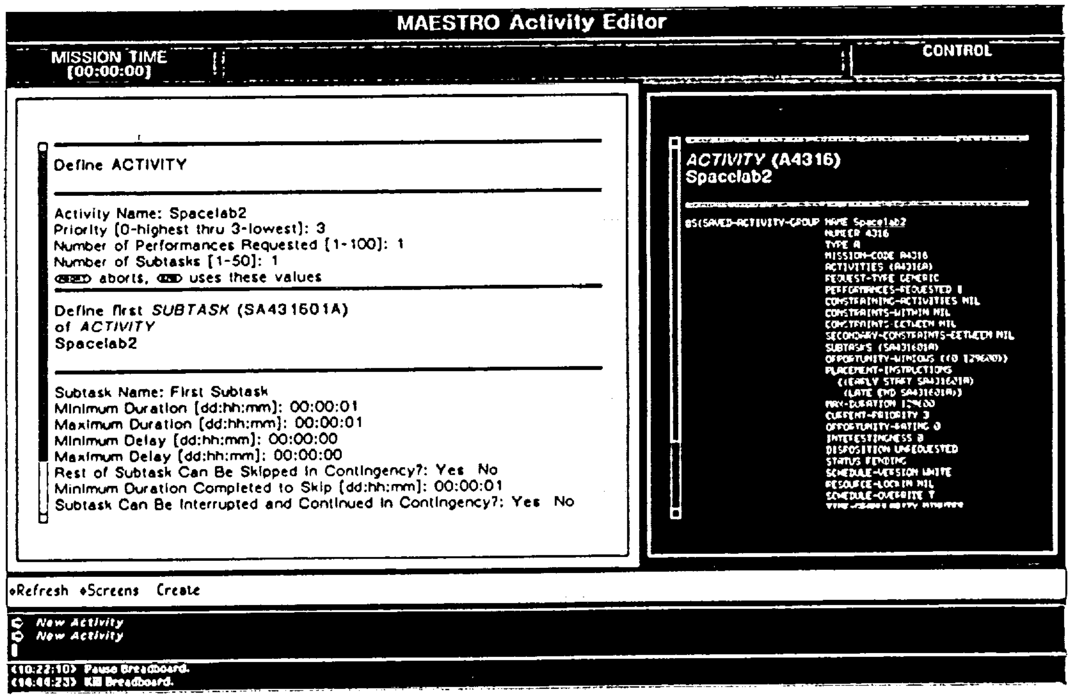


```
Equipment Editor Help

                                COMMAND    DESCRIPTION
                    <System Operations>    .............................................
                               *REFRESH    Refresh the display, use if display appears incomplete.
                               *SCREENS    Display a new screen, select from a menu.

                  <Database Operations>    .............................................
                                           Each of the operations require a choice of equipment from POWERED EQUIPMENT.
                           ADD LOCATION    Add a new location for a piece of equipment.
                               ADD MODE    Add a new mode of operation for a piece of equipment.
                          SHOW LOCATION    List all locations for a piece of equipment.
                             SHOW MODES    List all operating modes for a piece of equipment.

                    <Sensitive Objects>    .............................................
                      POWERED EQUIPMENT    L- Describe Powered Equipment Data Structure, R- Menu
Describe Powered Equipment Data Structure  Show internal data structure.
               Add Location for Equipment  Add a new location for this piece of equipment.
                   Add Mode for Equipment  Add a new mode of operation for this piece of equipment.
              Show Location of Equipment   List all locations for this piece of equipment.
                 Show Modes of Equipment   List all operating modes for this piece of equipment.
        Show Mode and Location of Equipment  List modes and locations for this equipment

Type a space to refresh the screen: █

SSMPMAD Symbolics Interface Equipment Editor Help
```

Figure 5.4.3.1-2 Equipment Editor Help Screen

5.4.3.2 The Activity Editor

The Activity Editor is used to create activities describing tasks to be scheduled. Each activity has a priority and a number of subtasks to be executed sequentially. The main window is used to enter information about the activities and the subtasks associated with that activity. The inverted window is for display purposes only. Figures 5.4.3.2-1 and 5.4.3.2-2 show the Activity Editor and Activity Editor Help screens, respectively.



```
Define ACTIVITY

Activity Name: Spacelab2
Priority [0-highest thru 3-lowest]: 3
Number of Performances Requested [1-100]: 1
Number of Subtasks [1-50]: 1
ABORT aborts, END uses these values

Define first SUBTASK (SA431601A)
of ACTIVITY
Spacelab2

Subtask Name: First Subtask
Minimum Duration [dd:hh:mm]: 00:00:01
Maximum Duration [dd:hh:mm]: 00:00:01
Minimum Delay [dd:hh:mm]: 00:00:00
Maximum Delay [dd:hh:mm]: 00:00:00
Rest of Subtask Can Be Skipped In Contingency?: Yes  No
Minimum Duration Completed to Skip [dd:hh:mm]: 00:00:01
Subtask Can Be Interrupted and Continued In Contingency?: Yes  No
```

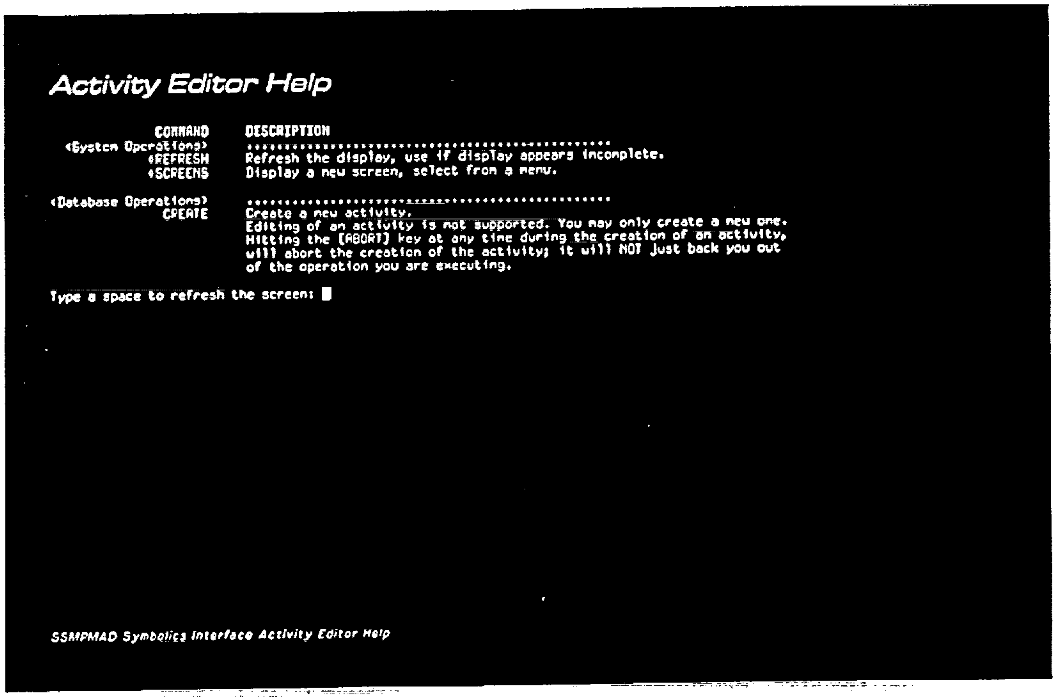
```
ACTIVITY (A4316)
Spacclab2

#S(SAVED-ACTIVITY-GROUP NAME Spacelab2
                        NUMBER 4316
                        TYPE A
                        MISSION-CODE A4316
                        ACTIVITIES (A4316A)
                        REQUEST-TYPE GENERIC
                        PERFORMANCES-REQUESTED 1
                        CONSTRAINING-ACTIVITIES NIL
                        CONSTRAINTS-WITHIN NIL
                        CONSTRAINTS-BETWEEN NIL
                        SECONDARY-CONSTRAINTS-BETWEEN NIL
                        SUBTASKS (SA431601A)
                        OPPORTUNITY-WINDOWS ((0 129600))
                        PLACEMENT-INSTRUCTIONS
                          ((EARLY START SA431601A)
                           (LATE END SA431601A))
                        MAX-DURATION 129600
                        CURRENT-PRIORITY 3
                        OPPORTUNITY-RATING 0
                        INTERESTINGNESS 0
                        DISPOSITION UNREQUESTED
                        STATUS PENDING
                        SCHEDULE-VERSION WHITE
                        RESOURCE-LOCKIN NIL
                        SCHEDULE-OVERWRITE T
                        [illegible]
```

```
New Activity
New Activity
```

```
<10:22:10> Pause Breadboard.
<16:44:23> Kill Breadboard.
```

Figure 5.4.3.2-1 Activity Editor Screen

ORIGINAL PAGE IS
OF POOR QUALITY



Activity Editor Help

```
              COMMAND     DESCRIPTION
     <System Operations>  ...............................................
                *REFRESH  Refresh the display, use if display appears incomplete.
                *SCREENS  Display a new screen, select from a menu.

   <Database Operations>  ...............................................
                  CREATE  Create a new activity.
                          Editing of an activity is not supported. You may only create a new one.
                          Hitting the [ABORT] key at any time during the creation of an activity,
                          will abort the creation of the activity; it will NOT just back you out
                          of the operation you are executing.

Type a space to refresh the screen: █
```

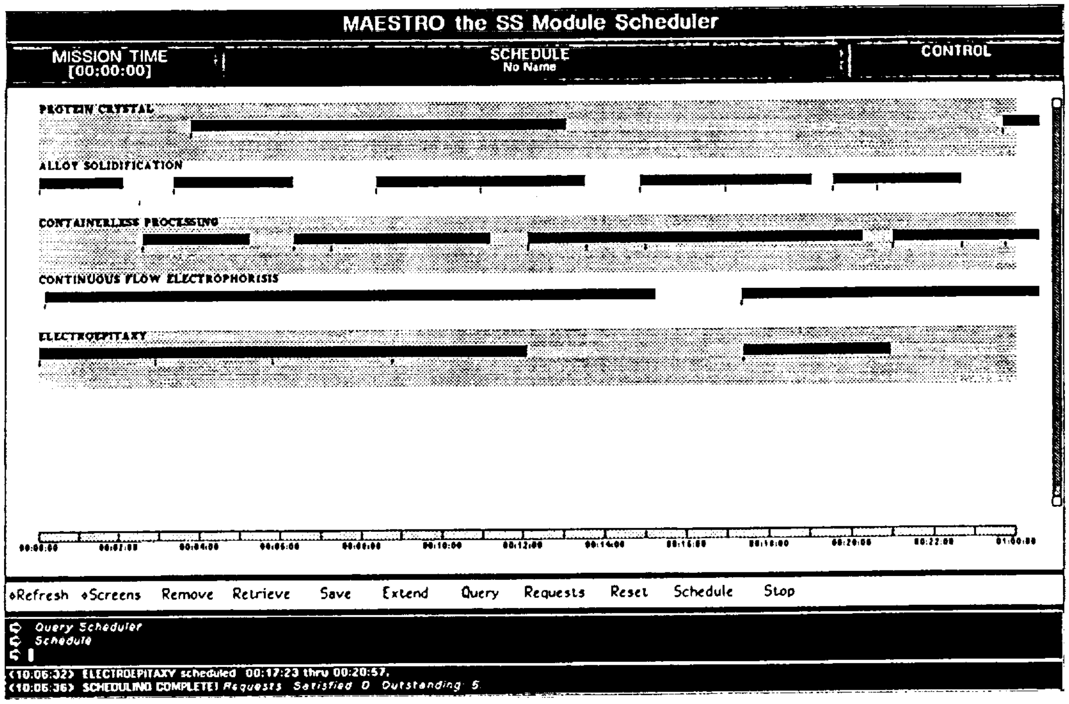
SSMPMAD Symbolics Interface Activity Editor Help

Figure 5.4.3.2-2 Activity Editor Help Screen

ORIGINAL PAGE IS OF POOR QUALITY

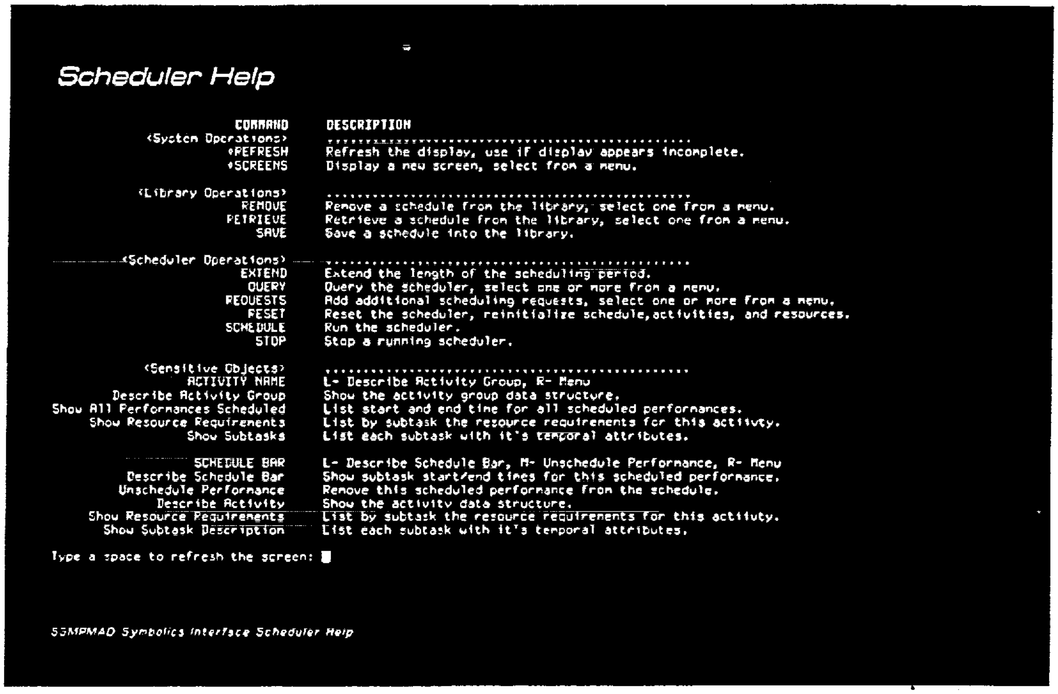
5.4.3.3 <u>The Scheduler</u>

The Scheduler is used for displaying the current schedule, as well as for resetting the scheduler, retrieving a schedule from the schedule library and getting information about the scheduled activities. The help screen for the scheduler provides more detail on the available operations on the Scheduler Screen. Figures 5.4.3.3-1 and 5.4.3.3-2 show the Scheduler and Scheduler Help screens, respectively.



```
⇨ Query Scheduler
⇨ Schedule
⇨ ▌
<10:06:32> ELECTROEPITAXY scheduled 00:17:23 thru 00:20:57.
<10:06:36> SCHEDULING COMPLETE! Requests Satisfied 0 Outstanding 5
```

Figure 5.4.3.3-1 Scheduler Screen



Scheduler Help

```
                     COMMAND     DESCRIPTION
             <System Operations>  ...............................................
                     *REFRESH     Refresh the display, use if display appears incomplete.
                     *SCREENS     Display a new screen, select from a menu.

            <Library Operations>  ...............................................
                       REMOVE     Remove a schedule from the library, select one from a menu.
                     RETRIEVE     Retrieve a schedule from the library, select one from a menu.
                         SAVE     Save a schedule into the library.

          <Scheduler Operations>  ...............................................
                       EXTEND     Extend the length of the scheduling period.
                        QUERY     Query the scheduler, select one or more from a menu.
                     REQUESTS     Add additional scheduling requests, select one or more from a menu.
                        RESET     Reset the scheduler, reinitialize schedule,activities, and resources.
                     SCHEDULE     Run the scheduler.
                         STOP     Stop a running scheduler.

            <Sensitive Objects>   ...............................................
                  ACTIVITY NAME   L- Describe Activity Group, R- Menu
        Describe Activity Group   Show the activity group data structure.
 Show All Performances Scheduled  List start and end time for all scheduled performances.
    Show Resource Requirements    List by subtask the resource requirements for this activty.
                 Show Subtasks    List each subtask with it's temporal attributes.

                  SCHEDULE BAR    L- Describe Schedule Bar, M- Unschedule Performance, R- Menu
          Describe Schedule Bar   Show subtask start/end times for this scheduled performance.
        Unschedule Performance    Remove this scheduled performance from the schedule.
             Describe Activity    Show the activity data structure.
     Show Resource Requirements   List by subtask the resource requirements for this activty.
       Show Subtask Description   List each subtask with it's temporal attributes.

Type a space to refresh the screen: █
```

SSMPMAD Symbolics Interface Scheduler Help

Figure 5.4.3.3-2 Scheduler Help Screen

ORIGINAL PAGE IS
OF POOR QUALITY

5.5 Front End Load Enable Scheduler (FELES)

5.5.1 Theory

The Front End Load Enable Scheduler (FELES) is responsible for translating activity schedules created by MAESTRO into component event lists for the power system and FRAMES. Basically, each item on the schedule must be translated into the appropriate RPC with additional information about the load connected to that RPC. This additional information includes the amount of power allowed through the RPC, whether the load has permission to test, whether the load may switch to a redundant source of power, and the amount of current the load may use.

5.5.2 Implementation

FELES runs as an individual process on the Symbolics 3620 D. The component event list is created periodically at time (- EFFECTIVE-TIME LEAD-TIME) allowing sufficient time to create and transmit the event list to the power system and FRAMES. Each event list created covers a period of time from EFFECTIVE-TIME for a length of (+ SCHEDULING-PERIOD DELTA). As a new event list is created the EFFECTIVE-TIME is incremented by SCHEDULING-PERIOD in preparation for the next event list. The DELTA is used for an overlap period between schedules in the event a contingency arises. Both LEAD-TIME and SCHEDULING-PERIOD may be modified by a LISP programmer for tuning of FELES to the actual performance characteristics of the breadboard and support software. As each event list is created, it is added to the event list keyed by the time it was created for historical record keeping.

Each event list consists of a time stamped list of events that specify actions the breadboard should perform in order to accomplish the objectives of the schedule of experiments/activities. Each event has the following format:

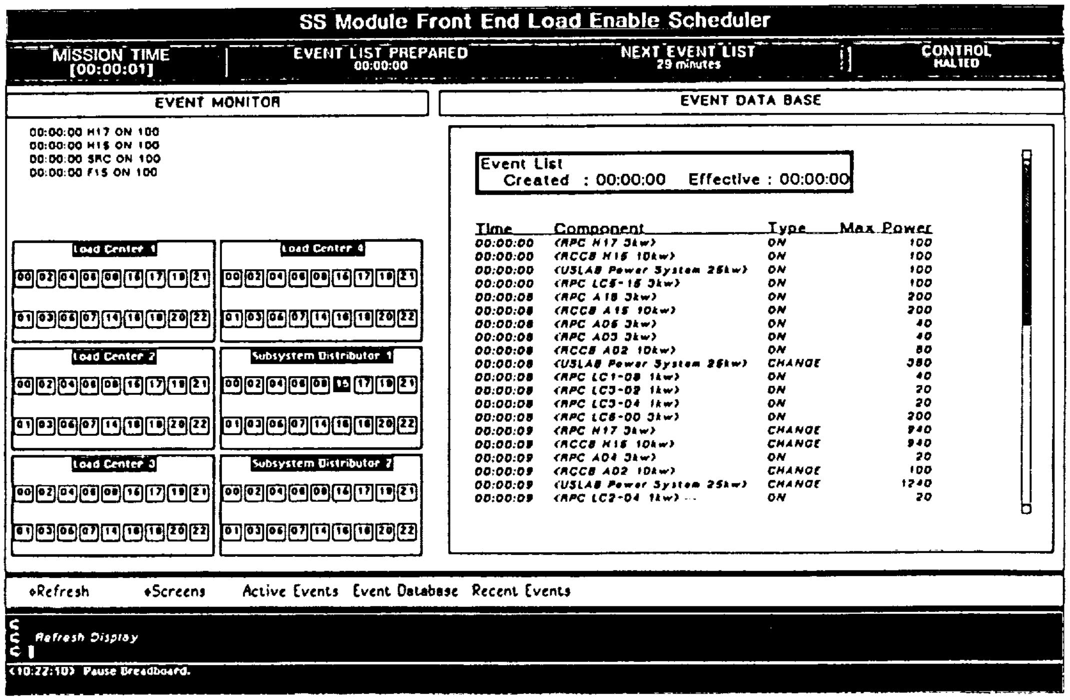
Time of Event	Time the event is to be initiated expressed in minutes from the start of mission
Component	Alphanumeric identifier of the power system component
Event	F-off, disabled O-on, enabled C-change, enabled but expect a different power level
Max Power	Maximum watts projected utilization
Permission to Test	Y-grant permission to open the switch for testing N-no permission to open for testing
Redundancy	Y-there is a redundant power path N-no redundant power path
Switch to Redundant	Y-permission to switch to redundant path N-no permission to switch
Max Current	Maximum current (p=iv) based on module voltage of 208
Min Current	Minimum current projected
Min Power	Minimum watts of projected utilization

5.5.3 User Interface

The Front End Load Enable Scheduler enables the user to browse and monitor the generation of load enablement commands. See Figures 5.5.3-1 and 5.5.3-2 that follow. Please refer to the Console description in the Symbolics 3620 D Environment section for status line monitor definitions.

The Event Monitor is a textual and graphical display indicating what commands are being recognized by the power system. This monitor is continually updated as events are initiated as mission time advances. This monitor effectively shows what should be happening in the power system. The graphical display shows the RPCs within each load center. If the RPC is black, it is enabled. A circle cross below an RPC indicates the RPC is out of service.

The Event Data Base is a textual listing of an event list. When the list was created and when it becomes effective are indicated. The user may choose any list that has been created by the FELES. By mousing on an event line, the user will see what experiments are utilizing the specific component at the time indicated.



SS Module Front End Load Enable Scheduler

MISSION TIME [00:00:01] | EVENT LIST PREPARED 00:00:00 | NEXT EVENT LIST 29 minutes | CONTROL HALTED

EVENT MONITOR

```
00:00:00 H17 ON 100
00:00:00 H15 ON 100
00:00:00 SRC ON 100
00:00:00 F15 ON 100
```

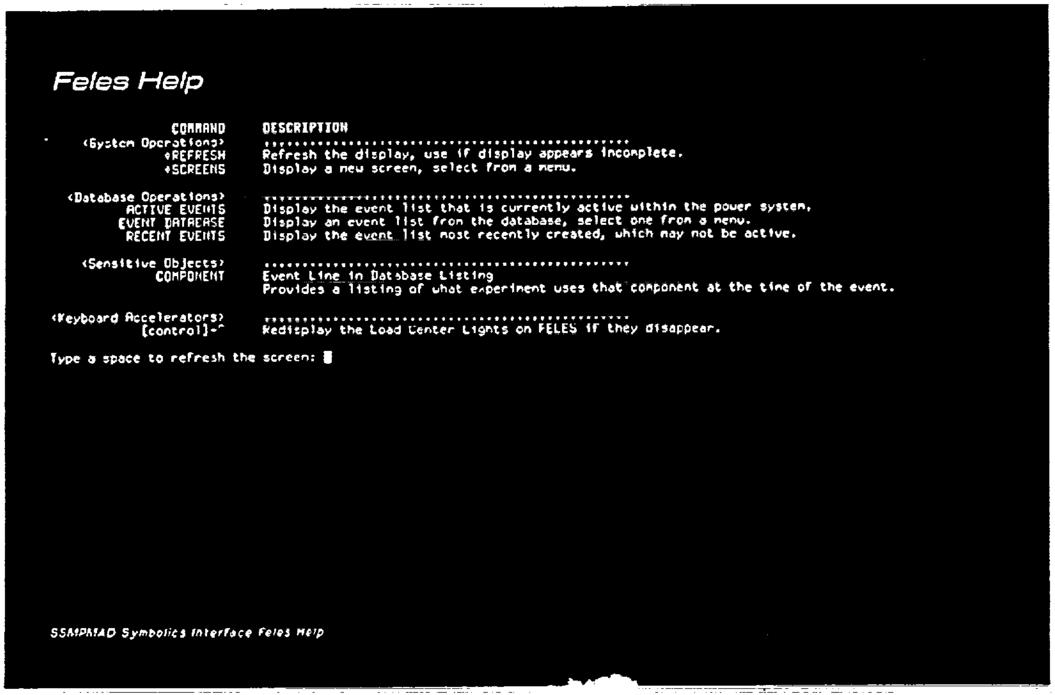
EVENT DATA BASE

```
Event List
   Created  : 00:00:00   Effective : 00:00:00

Time      Component                      Type     Max Power
00:00:00  (RPC H17 3kw)                  ON             100
00:00:00  (RCCB H15 10kw)                ON             100
00:00:00  (USLAB Power System 25kw)      ON             100
00:00:00  (RPC LC5-15 3kw)               ON             100
00:00:08  (RPC A18 3kw)                  ON             200
00:00:08  (RCCB A15 10kw)                ON             200
00:00:08  (RPC A05 3kw)                  ON              40
00:00:08  (RPC A03 3kw)                  ON              40
00:00:08  (RCCB A02 10kw)                ON              80
00:00:08  (USLAB Power System 25kw)      CHANGE         380
00:00:08  (RPC LC1-08 1kw)               ON              40
00:00:08  (RPC LC3-02 1kw)               ON              20
00:00:08  (RPC LC3-04 1kw)               ON              20
00:00:08  (RPC LC6-00 3kw)               ON             200
00:00:09  (RPC H17 3kw)                  CHANGE         940
00:00:09  (RCCB H15 10kw)                CHANGE         940
00:00:09  (RPC A04 3kw)                  ON              20
00:00:09  (RCCB A02 10kw)                CHANGE         100
00:00:09  (USLAB Power System 25kw)      CHANGE        1240
00:00:09  (RPC LC2-04 1kw) ...           ON              20
```

```
<
< Refresh Display
< I
<10:22:10> Pause Breadboard.
```

Figure 5.5.3-1 Front End Load Enable Scheduler Screen

ORIGINAL PAGE IS
OF POOR QUALITY



```
Feles Help

              COMMAND      DESCRIPTION
    <System Operations>    ************************************************
               *REFRESH    Refresh the display, use if display appears incomplete.
               *SCREENS    Display a new screen, select from a menu.

  <Database Operations>    ************************************************
          ACTIVE EVENTS    Display the event list that is currently active within the power system.
         EVENT DATABASE    Display an event list from the database, select one from a menu.
          RECENT EVENTS    Display the event list most recently created, which may not be active.

    <Sensitive Objects>    ************************************************
              COMPONENT    Event Line in Database Listing
                           Provides a listing of what experiment uses that component at the time of the event.

<Keyboard Accelerators>    ************************************************
           [control]-^     Redisplay the Load Center Lights on FELES if they disappear.

Type a space to refresh the screen: █

SSMPMAD Symbolics Interface Feles Help
```

Figure 5.5.3-2 Front End Load Enable Scheduler Help Screen

ORIGINAL PAGE IS OF POOR QUALITY

5.6 Load Priority List Management System (LPLMS)

5.6.1 Theory

The Load Priority List Management System (LPLMS) is responsible for determining and providing a list of lowest level RPCs in reverse order of priority so that the lowest level processors in the power system may quickly shed loads in the event of a power system contingency. For example, in the event of an immediate power change where the scheduler does not have time to compute a new schedule, the LLPs must shed the lowest priority loads to stay within the new amount of power. As another example, an RPC may trip for some reason causing a high priority load below it to be switched to redundant. The effect of switching a load to a redundant supply may force some other loads on the redundant bus to be shed due to limiting power availability on the bus and priority of loads. These actions must be done quickly at the LLP level, yet the power system does not have knowledge of loads and activities, so the Load Priority List (LPL) is computed at a functionally higher level (near the scheduler).

The accuracy of the LPL is dependent upon how fast the LPL can be generated and sent to the LLPs. The schedule has a one minute granularity, yet it is not possible to send a new LPL to the LLPs every minute. A LPL may be generated once every X number of minutes (where X is a programmable variable equal to 15 minutes in this implementation), creating a reasonable approximation of what loads are running over the X minute interval and their respective priorities. At the same time, it is possible that, during operation, a particular load on the LPL is not actually using power sometime during the X minute interval. For this reason it is up to the LLP to continue shedding loads until power constraints are met. An optimal method would be to have an accurate LPL for each minute of operation of the breadboard.

5.6.2 Implementation

The Load Priority List Management System runs as an independent process on the Symbolics 3620 D. The LPL is created periodically at time (- EFFECTIVE-TIME

LEAD-TIME) allowing sufficient time to create and transmit the LPL to the LLPs. Each LPL created spans an interval of time starting at EFFECTIVE-TIME and ending at (+ EFFECTIVE-TIME HOW-OFTEN), where HOW-OFTEN is currently set at 15 minutes. As a new LPL is created EFFECTIVE-TIME is incremented by HOW-OFTEN in preparation of the next LPL. As each LPL is created it is added to the LPL database keyed by the time of creation for historical record keeping.

The criteria used to order the LPL must take into account what each RPC is being used for, how much power it is drawing, and how crucial it is to continue providing power to the task using the RPC. Unfortunately, over a 15 minute time period, an individual RPC may be used sequentially by different tasks. It is impossible to know which task during the 15 minute period might be interrupted if that RPC is turned off, so the influence of all of these tasks must be taken into account. A damage assessment is made of what damage might be done by shedding a particular RPC, and is combined with the benefit of how much power would be dropped if that RPC were switched off to determine where the RPC falls in the list.

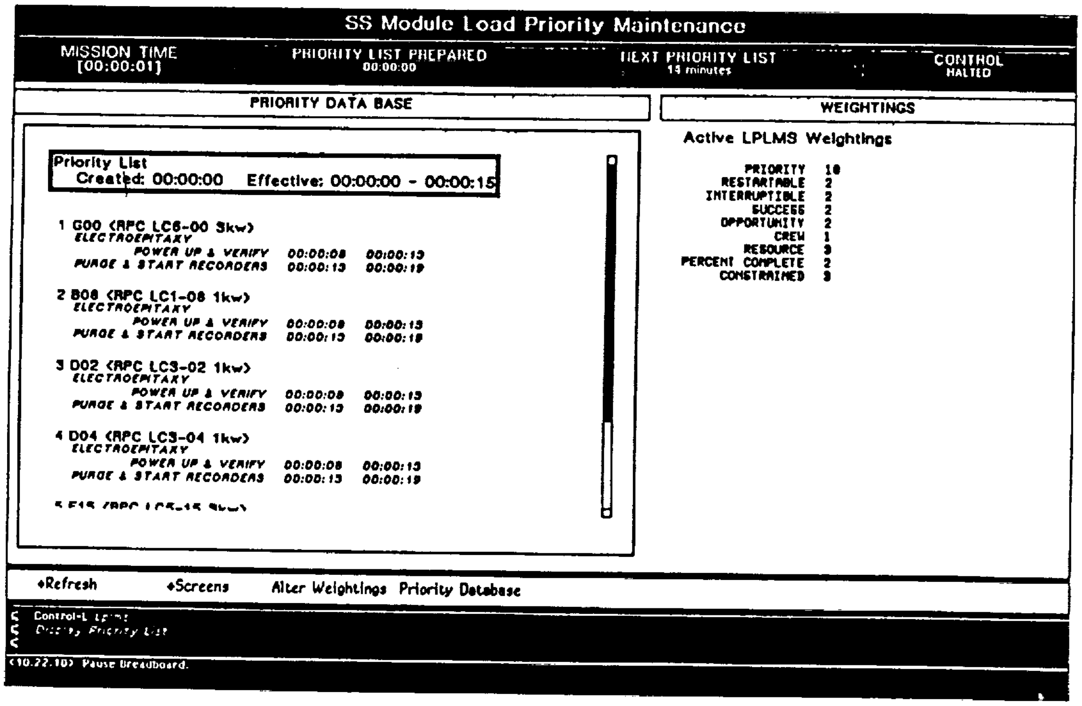
The LPLMS examines the currently active schedule generated by MAESTRO collecting all of the subtasks to be performed during the time interval the LPL will be active. These are used to determine which RPCs will be used. From these subtasks and RPCs, the LPLMS uses the same weighting criteria as the scheduler uses to determine the next subtask to schedule. This generates a priority for the subtask in increasing order. Once this is completed, it only has to be reversed and sent to the LLPs.

5.6.3 User Interface

The Load Priority Maintenance screen provides the user the ability to inspect load priority lists that have been generated by the LPLMS and transmitted to the power system. No updating or modification operations are permitted. See Figures 5.6.3-1 and 5.6.3-2.

The Priority Data Base displays when a priority list was created, when that list is effective, the ordered priority list, and for each component in the priority list the subtasks of the activities that are utilizing that component during the effective time period. The user may display any priority list that has been created by the LPLMS.

The Weightings is an informational display that shows what weighting criteria was used in the development of the priority list and range from 10-highest to 0-lowest.



SS Module Load Priority Maintenance

MISSION TIME	PRIORITY LIST PREPARED	NEXT PRIORITY LIST	CONTROL
[00:00:01]	00:00:00	14 minutes	HALTED

PRIORITY DATA BASE

```
Priority List
  Created: 00:00:00   Effective: 00:00:00 - 00:00:15

1 G00 <RPC LC6-00 3kw>
  ELECTROEPITAXY
           POWER UP & VERIFY   00:00:08   00:00:13
  PURGE & START RECORDERS      00:00:13   00:00:19

2 B08 <RPC LC1-08 1kw>
  ELECTROEPITAXY
           POWER UP & VERIFY   00:00:08   00:00:13
  PURGE & START RECORDERS      00:00:13   00:00:19

3 D02 <RPC LC3-02 1kw>
  ELECTROEPITAXY
           POWER UP & VERIFY   00:00:08   00:00:13
  PURGE & START RECORDERS      00:00:13   00:00:19

4 D04 <RPC LC3-04 1kw>
  ELECTROEPITAXY
           POWER UP & VERIFY   00:00:08   00:00:13
  PURGE & START RECORDERS      00:00:13   00:00:19

5 E15 <RPC LC5-15 5kw>
```

WEIGHTINGS

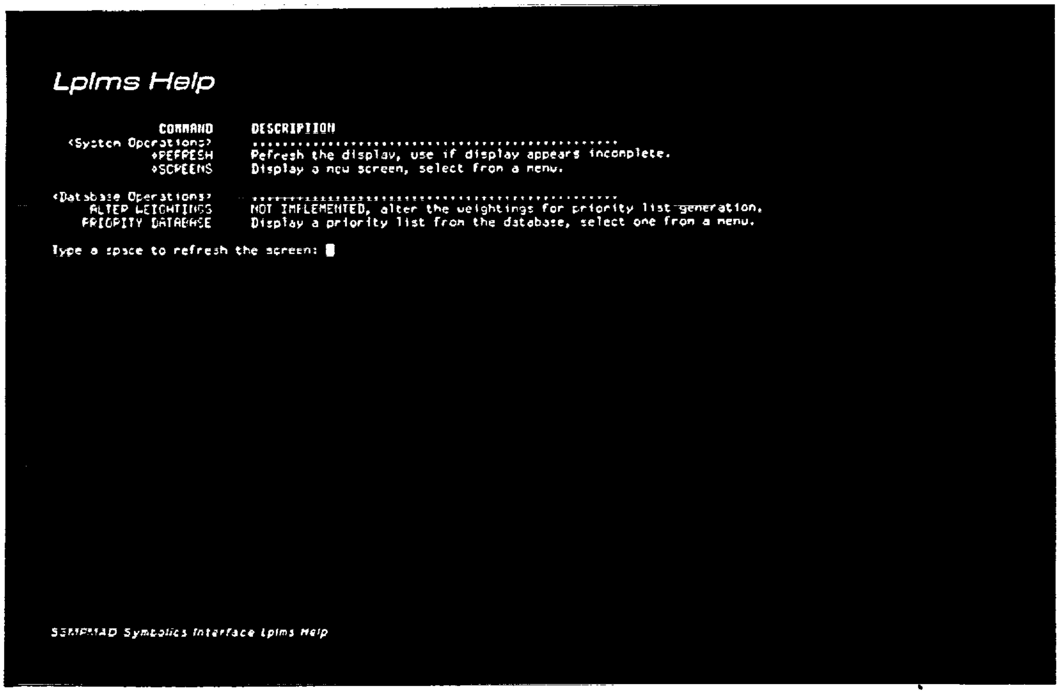
Active LPLMS Weightings

PRIORITY	10
RESTARTABLE	2
INTERRUPTIBLE	2
SUCCESS	2
OPPORTUNITY	2
CREW	1
RESOURCE	3
PERCENT COMPLETE	2
CONSTRAINED	3

```
< Control-L [illegible]
< Display Priority List
<
<10.22.10> Pause Breadboard.
```

Figure 5.6.3-1 Load Priority List Management Screen

ORIGINAL PAGE IS OF POOR QUALITY



Lplms Help

```
              COMMAND     DESCRIPTION
      <System Operations>     .............................................
             *REFRESH     Refresh the display, use if display appears incomplete.
             *SCREENS     Display a new screen, select from a menu.

    <Database Operations>     .............................................
      ALTER WEIGHTINGS    NOT IMPLEMENTED, alter the weightings for priority list generation.
     PRIORITY DATABASE    Display a priority list from the database, select one from a menu.

Type a space to refresh the screen: █
```

SSM/PMAD Symbolics Interface Lplms Help

Figure 5.6.3-2 Load Priority List Management Help Screen

5.7 Fault Management and Recovery

Automation of power management and distribution for SSM/PMAD requires the ability to manage and recover from faults in an autonomous mode. Obviously, without this ability, automation would be severely limited. Therefore, the power system must accomplish comprehensive fault management including detection, recovery, cause identification, system recovery and network protection.

The fault management and recovery aspects of SSM/PMAD are not an isolated part of the automation software, but practically make up most of the automation software. Fault management and recovery requires integration of all components of the automation testbed. Detection of anomalous situations occurs at the switchgear in microseconds. Responses needed in seconds occur at the LLPs. FRAMES diagnoses faults in minutes and MAESTRO recovers from fault situations in tens of minutes. Thus the fault management and recovery aspect of the automation software is functionally apportioned in light of required response times for effective response and management.

As an example, a hard fault will be detected by an RPC when it trips as the first line of defense against network problems. This status change data, along with other sensor data, is transmitted to FRAMES. FRAMES recognizes symptoms in the data, trying to match against known symptom sets (patterns). Depending on the symptom set, the fault diagnosis software will either identify the cause, a set of possible causes, or initiate some diagnostic switch manipulations which will lead to such identification. Switch manipulations are constrained by a supervisory module containing MAESTRO and information about load constraints.

There are three types of faults that may occur in the power network. A hard fault is a situation physically causing a switch to trip, as well as broken components. A soft fault is an illegal use of current not necessarily causing a switch to trip, for example, a resistive short to ground. Finally, an incipient fault is a situation that will become a hard or soft fault in a reasonably short period of time if nothing is done to avert it.

The fault management and recovery software is conceptually divided into three parts. First is the protection of the network by the smart switchgear and subsequent detection of symptoms by the LLPs. Second is the recognition of a fault situation and analysis of the fault by FRAMES. Third is the recovery of the fault situation by MAESTRO. The first two are discussed in detail below. The third, MAESTRO, is discussed in the section on resource scheduling.

Knowledge engineering for SSM/PMAD and particularly FRAMES was done throughout the project. The knowledge engineering resulted in a fault and symptom matrix that was used to develop the FRAMES software. This fault and symptom matrix can be found in Appendix XI.

5.7.1 Lowest Level Processors

From a FRAMES perspective, lowest level processors are identical. LLPs gather switch and sensor data from the power system hardware under their control. The switch information is scanned to determine if switches have tripped. If a switch has tripped, the LLP marks the switch as out of service. Any switching operations on an out of service switch are ignored. The LLP software then informs FRAMES of the hard fault. In diagnosis, FRAMES may require more information from the power system network. If so, FRAMES sends the appropriate LLPs a fault event list which manipulates the switch(es) for fault isolation. An LLP always responds with a fault event list response containing the requested information. Inquiries to the LLPs are made until FRAMES diagnoses the fault. When the fault has been diagnosed, FELES issues a contingency event list and LPLMS issues a new priority list. Upon receipt and implementation of the contingency event list, the LLPs clear all temporarily out-of-service flags and resume normal operations. Although LLP software looks identical from an external point of view, there are some differences between load center (LC) and power distribution control unit (PDCU) LLPs.

LCs and PDCUs handle soft faults in somewhat different ways. A soft fault, as mentioned earlier, is an illegal use of current. When a switch is scheduled it is allotted a maximum current and power. If the allotted current is exceeded, a LC sheds the load because it does not conform to the schedule. In this situation, the LC software informs FRAMES of the illegal use of current and the remedial action taken. Conversely, a PDCU will only inform FRAMES of the illegal use of current. To take remedial action could affect as many as fourteen switches in the load center below the PDCU switch. The PDCU software also looks for illegal uses of current within the PDCU using Kirchoff's Current Law to find discrepancies. LLPs would not be able to handle hard or soft faults without the intelligent power system hardware.

LLPs interface to the power system through a Switchgear Interface Control (SIC) card. The SIC card communicates with fourteen Generic Controller (GC) cards. Each GC card controls the switching operations of a connected switch. The four different types of switches are: One kilowatt Remote Power Controller (RPC), three kilowatt RPC, ten kilowatt Remote Controlled Circuit Breaker (RCCB), and a twenty-five kilowatt Remote Bus Isolator (RBI). The RPCs and RCCBs will current trip if their hardware limits are exceeded. This is done to protect the power system and give the automation system time to correct the problem. The RPCs and RCCBs also return current sensor data to the GC which is digitized and transmitted to the SIC. The SIC communicates with an Analog to Digital (A/D) card which accepts up to sixteen voltage, current, and temperature sensor inputs. The A/D card processes the analog sensor inputs and returns digitized RMS voltage, RMS current, DC voltage, DC current, frequency, average power, instantaneous power, power factor, and temperature data to the SIC card. All digitized data has eight bits of accuracy. Use of this intelligent hardware through the SIC interface facilitates the automation system.

5.7.2. Fault Recovery and Management Expert System

Conceptually, FRAMES is not simply an expert system residing on the Xerox 1186. It is an extended assembly using smart switchgear to identify and classify overstresses and to quickly disconnect them from the network. Local algorithmic controllers, the LLPs, gather the data and sensor values and transmit them to the Xerox 1186 for classification and diagnosis. FRAMES begins at the switches and ends at an interface screen where it reports status outputs. Thus the fault recovery and management software is a large conceptual piece of software. The expert system component for classifying and diagnosing faults resides on the Xerox 1186 and is the component to be discussed here as fault diagnosis.

5.7.2.1 Theory of Operation

There are a number of issues involved in fault diagnosis in general. These include: The computation of symptom sets, model based reasoning, single vs. multiple faults, and how fault isolation is done. Each of these issues will be discussed here.

5.7.2.1.1 Symptom Sets

A symptom set is a set of symptoms that indicate a fault. To put it another way, a fault gives rise to a set of symptoms. In fault diagnosis, these symptom sets may be computed in a number of ways. One may have a model of the power system and dynamically compute the possible symptom sets for any possible fault in the power system. Alternatively, one may analyze all the possible faults in the power system beforehand and save the dynamic computation for memory space instead. The benefit of dynamic computation is to be able to compute symptom sets for unforeseen power system topologies. In the static computation mode, if the power system topology changes significantly, potentially large amounts of work are wasted and need to be redone.

Symptom sets are used for pattern matching in an attempt to determine what fault may have occurred in the power system. Obviously, a symptom or set of symptoms resulting from an actual fault may indicate more than one possible fault. It is then important to isolate the fault from among the various possibilities. Alternatively, it may be possible to analyze a symptom set without pattern matching to all possible symptom sets, letting the symptom set drive out the possible fault scenarios.

In FRAMES, all the reasonable fault scenarios are analyzed and their symptom sets computed. These are used within FRAMES to pattern match against to determine possible fault situations.

5.7.2.1.2 Model Based Reasoning

Similar trade-offs apply here as in the computation of symptom sets. In general, the motivation for using model based reasoning is when all the fault scenarios are not necessarily knowable beforehand. This usually happens when requirements are changing or when the domain of reasoning is a dynamic domain. Model based reasoning is usually used when reasoning from some sort of first principles is required.

For FRAMES this was not considered necessary. An analysis of the power system was done and was also considered static. This analysis gave rise to the possible symptom sets described above. As all the symptom sets were known beforehand, it did not seem necessary to use model based reasoning.

5.7.2.1.3 Single vs. Multiple Faults

The issue of single vs. multiple faults is whether FRAMES will diagnose those faults that occur singly, spaced out from one another, or if FRAMES will diagnose independent and dependent faults occurring at or near the same time. Diagnosing multiple faults is not a well understood problem. The dependent nature among faults greatly increases the complexity of the situation. Furthermore, multiple simultaneous faults were not considered very credible scenarios for the domain under consideration. For FRAMES, single fault diagnosis is utilized. FRAMES also diagnoses certain classes of multiple faults, masked faults. For example, if a switch's current sensor is broken and a short appears below the switch, the switch above will trip on over current. FRAMES will diagnose these kinds of faults.

There is another type of fault situation, cascaded faults, that applies to both multiple faults and single faults. A cascaded fault situation is where a short circuit may arise below a 3k RPC causing it to trip on a fast trip. Consequently all the load center RPCs that were closed will also trip on under voltage. This is a cascade effect. To be accurate, what is really being reported to the expert system on the Xerox is a set of cascaded symptoms arising from a single fault in this example. Most faults will have cascaded symptoms giving some indication of the fault. Thus, when FRAMES diagnoses faults, it also includes those faults with cascaded symptoms.

5.7.2.1.4 Fault Isolation

The issue of fault isolation is how to isolate where a fault occurred. Symptoms may describe a large class of possible faults that could account for them. Obviously, one does not want to hypothesize all the possible faults. Rather, one would like to discriminate further between the possible faults. There are two basic mechanisms to do this. One is to probe for values at various points in the power network. Obviously, in a fully automated system this is not easy to accomplish. The second basic method is to manipulate the switches.

Switch manipulation is performed in FRAMES. Switch manipulation provides for control of the state of faulted areas of the network, allowing testing by opening and closing switches to produce useful results. As switches are opened and closed, data are collected from the results of these operations to further discriminate between possible faults. Switch manipulation proves to be a very useful diagnostic tool in power networks due to the hierarchical topology of the switches, for example, as in Space Station Freedom.

5.7.2.2 Implementation

The SSM/PMAD power breadboard is essentially a radial or linear feed from the RCCBs to the loads (the ring bus lies outside its jurisdiction). The radial feed means that each RPC has exactly one parent, which greatly simplifies fault diagnosis. Note that this would not be true under a more elaborate cross-strapping regime. Note also that

RCCBs have exactly two parents but since they do not trip on under voltage, that fact is less important.

Consider dividing the system with the following levels:

Level 1 – load center switches (RPCs)
Level 2 – PDCU RPCs
Level 3 – PDCU RCCBs
Level 4 – RBIs
Level 5 – Source

Since all the RPCs (but not the RCCBs) will trip on under voltage when its line voltage drops below a specified threshold, a key consideration for diagnosis is what is the highest level (1-5) to report a symptom. In most cases the highest level will report a fast trip, I^2t (over current) or ground fault interrupt (gfi) and any switch below (i.e. closer to the loads) will report tripping on under voltage. (Of course, only those lower switches that were closed before the fault should report an under voltage trip.)

5.7.2.2.1 Components That Can Cause Failures

The diagnostic software considers the following components and subcomponents as capable of causing or having a failure.

5.7.2.2.1.1 Cables

Cables can have a high impedance short to structure or return (I^2t), a low impedance short to structure or return (fast trip), or a high impedance short to ground (gfi).

5.7.2.2.1.2 Switch Input Stub

The cable-like portion of the switch at its input can have the same problems as regular cables.

5.7.2.2.1.3 Switch Output Stub

The cable-like portion of the switch at its output can have the same problems as regular cables.

5.7.2.2.1.4 Current Sensors and Current Comparators in Switches

The current sensors and current comparators in switches control the I^2t, fast trip, and gfi trip mechanisms. Failure in either the sensor or comparator could cause a switch to trip when current was nominal and report I^2t, fast trip, or gfi. Likewise, a failure could cause the switch not to trip on a current anomaly when it should be reading an abnormal current as nominal or by miscalculating the comparisons.

5.7.2.2.1.5 Voltage Sensors and Voltage Comparators in Switches

The voltage sensors and voltage comparators in switches control the under voltage trip mechanism. Failure in either the sensor or comparator could cause a switch to trip on under voltage when voltage was actually nominal or vice versa, not tripping when current was too low. While the latter is assumed to be rare, it would account for a switch reporting an under voltage trip when the sibling and parent switches continue to draw current (a shared cable would also account for this).

5.7.2.2.1.6 Eraseable Programmable Logic Device

The EPLD subsumed numerous functions including some mentioned above. Since actual control of these functions is likely to be separated into individual components, the diagnostic software assumes that either a single EPLD function will fail or the entire chip will. If the entire chip fails, it could cause a switch to trip and report any of the standard trip conditions except fast trip: I^2t, gfi, and under voltage. Since its effects can be so diverse, it would be useful to be able to discriminate a failed EPLD from other possible causes. Fortunately, a failed EPLD will not communicate with the SIC at all. This lack of communication will be found in the first step of manipulating switches. As it turns out, this is the only failure that can be definitively pinpointed to a specific component.

5.7.2.2.1.7 Loads

The loads are assumed to be able to cause any of the current irregularities (and thus, trip conditions) that cables can. Loads are considered to be the responsibility of the load provider rather than the power system proper.

5.7.2.2.2 Numbers of Failures

It has been decided that multiple failures occurring simultaneously are not credible. This leaves two categories of failures: single point failures and two point failures where one is a masked fault. A masked fault is an irregularity in a switch that causes it not to trip in response to an excessive current but instead to propagate the excessive current up to the switch's parent. The parent will then trip (if it did not, three failures would be involved which is assumed to be unrealistic) and the original switch will subsequently trip on under voltage. The resultant set of symptoms will look identical to those produced by, for example, a cable fault above the original switch that creates the same magnitude and type of current excess. The term "masked" is used since this type of failure could happen at any point but will not manifest until the specific, faulted functionality is called for.

5.7.2.2.3 Sources of Additional Information

Beyond the original symptom set, there are two source of additional information, only one of which is presently used.

It is possible that certain switches will not trip at all. Such a switch could propagate an excessive current to its parent and, when its parent trips, also fail to trip on under voltage. The failure to trip on under voltage is an "unreported" symptom. Once the topmost component reporting a symptom has been identified, a quick calculation can determine what all the reported symptoms should have been. (Essentially, any switch below the topmost that was closed/enabled below the fault should report tripping on under voltage.) Comparing these calculated symptoms to those actually reported will reveal any unreported symptoms. There should be no more than one unreported symptom: an unreported symptom reflects a masked fault (probably a complete EPLD failure) and credibility considerations limit the number of masked faults to one, since a masked fault is only revealed by a second fault (typically excessive current propagated to and through the switch with the masked fault). It is not clear that the scenario with unreported symptoms can actually happen (i.e., that the EPLD will malfunction in that way) and the software for handling this contingency is not used.

The second source of additional information is the manipulation of switches. This is the a powerful diagnostic tool used by FRAMES but unfortunately it provides limited information. When the topmost symptom is in a load center, switch manipulation will either reduce the number of possible causes from five to three, for example, or it will not reduce the number at all, depending on the response to the manipulation. When the topmost symptom is in the PDCU, manipulating switches can distinguish between single point failures and a two fault scenario with a masked fault, and, in the case of the latter, potentially determine which lower circuit the faults are in. In only two cases, however, can manipulating switches isolate the cause to an individual component. The entire manipulation process is discussed in detail in subsequent sections.

5.7.2.2.4 Kinds of Failures

The simplest fault scenario occurs when a load center RPC reports tripping on excessive current (I^2t, fast trip, or gfi) and no other component reports any anomaly. The cause is one of the following: high impedance short to structure/return, low impedance short to structure/return or high impedance short to ground in the RPC switch output, the cable connecting the RPC to the load, or in the load itself; or in the current sensor in the RPC or current comparator in the RPC. (Note: the type of short is a function of the type of trip reported, as indicated earlier.)

There are five possible causes and switch manipulation is the only means available to the system to obtain more information. The switch manipulation in all cases follows a set pattern. Since there is only one switch involved here, only the first step of that pattern is used.

An "open" command is sent to the load center RPC. If it does not respond at all, its EPLD is malfunctioning. This is assumed to be the cause of the current trip as well and the diagnostic process ends. If the switch opens, then a "close" command is sent. If the RPC is closed and it trips with the same symptom no further discrimination can be made.

Several scenarios are possible and all five of the possible causes are plausibly implicated. First, even if the load did not resume current consumption, a short in the connecting cable or the switch output cable would retrip the RPC. Likewise, depending on how the EPLD was malfunctioning, it might retrip the RPC even in the absence of current. Similarly, malfunctioning in either the current sensor or comparator could retrip the RPC with or without current flowing. Finally, if the problem was in the load and the load restarted, it would retrip the RPC.

Suppose, on the other hand, the RPC does not retrip when it is reclosed, it can be concluded that the cause is not a short in the switch output cable or in the connecting cable. The other three remain plausible. If the load did not restart (and the SSM/PMAD has no way to monitor the load other than the presence of current across its RPC), then there could still be a problem in either the load itself of the current sensor or comparator.

(Note that if the load restarts, as indicated by current across the RPC, and the RPC does not trip, then there is an ambiguity: all the conditions of the first trip have been replicated but the RPC reacts differently. This may reflect an intermittent failure that FRAMES is not designed to address. Also note that although FRAMES is not designed specifically to diagnose multiple faults and certain other cases, FRAMES does still report the situation at which it could not determine a diagnosis. Switches may still be taken out of service in these situations. In all situations, information is displayed on the FRAMES interface indicating the diagnosis status.)

A more complex scenario entails the topmost symptom being a 3k RPC in the PDCU tripping on excessive current (fast trip, I^2t, or gfi). The single point causes that could cause this are: high impedance short to structure/return, low impedance short to structure/return or high impedance short to ground in the PDCU RPC switch output, the cable connecting the PDCU RPC and a load center RPC, or in a load center RPC switch output, as well as the current sensor or comparator in the PDCU RPC. The two point causes (with masked fault) are: a complete EPLD failure in a load center RPC or a current sensor or comparator failure in a load center RPC coupled with one of the possible shorts of the single point failure. Note that these two point causes are actually classes of causes, each class containing up to 14 members (one for each of the load center RPCs on the bus below the 3k RPC in the PDCU – in the worst case they were all closed before the fault occurred. Thus there are 3x3, or 9 classes of two point failures and 9x14, or 126 individual two point failures. Note further that for single point causes, there are 14 load center RPCs whose switch input cables could have failures, so there are not five, but 18 single point failures. Thus there is a total of 126+18, or 144 possible causes.

Diagnosis proceeds by manipulating switches in accordance with set protocol (the fixed and radial nature of the breadboard obviates the need for an adaptive procedure). First the PDCU RPC and all the load center RPCs are commanded open, partly to assure the breadboard is in a known state for testing and partly to test the EPLD . If any of these fails to respond, all testing stops and that switch is assumed to have a completely failed EPLD which is the single cause. Note: this might not be true if a load center RPC fails to respond it may be the case that there is a second fault, a hard short, below that RPC. The EPLD problem is acting as a masked failure so even if that RPC is replaced, it will trip again when enabled.

If all switches respond to the open commands, the PDCU RPC is commanded closed. This is an important step since it discriminates between single point failures and two point failures with masked faults. If the PDCU RPC retrips with the same symptom when it is closed, the problem is a single point failure. All load center RPCs have been opened so the circuits below them are isolated and cannot propagate an excessive current flow upward (there should, in fact, be no current flow whatsoever in those circuits). Thus, the problem must be in or between the PDCU RPC and the switch input cables of the load center RPCs as listed above (note that this includes the PDCU RPC's current sensor and comparator).

If, on the other hand, the PDCU RPC does not retrip when it is closed, then either the problem is in the PDCU RPC's current sensor or current comparator (which in this case will not show up until a load center switch is closed and a connected load begins to draw current) or it is a two point failure involving a masked fault in one of the 14 load center RPCs and a hard short in the circuit beneath it. (Note: not all 14 load center RPCs may be candidates – only those that were closed before the fault occurred.)

In this circumstance, manipulation proceeds with the load center RPCs. Each one is closed and, if it does not cause retripping, opened in turn. They are tested in isolation to assure that only one RPC and the circuit beneath it are tested at a time. If one of these RPCs causes retripping, one of two things is the case. The problem could be a masked fault in that switch together with a hard short in that RPC's switch output cable, in the cable connecting it to the load, or in the load itself. No further narrowing of possibilities can be made without either manually changing out the switch, cable and/or load or manually testing these components with instruments (the latter might not be possible with the load).

In addition, if this load center RPC is the first to be closed, that results in a current draw of sufficient magnitude (e.g. whose load restarts), an additional hypothesis must be added to the above set. It may be that the current sensor or comparator in the PDCU RPC may be faulted in such a way that it reacts to any non-zero current (which is also presumably above some level, particularly to the fault in question) as an overcurrent. The only way to know whether or not this is a possibility is to monitor for current flow each time a load center RPC is closed. Current software at the LLP level does not collect this data, so FRAMES at the higher level does not track whether or not a load center RPC that caused retripping is the first to cause current draw. Even if it is not the first to draw current however, it may be the first to draw an in-range current that is sufficiently large to cause the tripping.

If none of the load center RPCs causes retripping, three possibilities remain. The first is an "overload": while no individual load center RPC (or, actually, its connected load) was drawing enough current to make it trip, some of them are drawing more than they were scheduled to in such a way that collectively the load center RPCs are drawing enough to trip the PDCU RPC (3.3K worth). FRAMES explores this possibility at this point. All the load center switches are open (each was opened after it was closed in the previous portion of the switch manipulation procedure). All the relevant switches (the ones that were closed before the fault) are closed in sequence. If, at some point during these closings everything trips again in the original pattern, it is assumed that the problem is, in fact, an overload and that some if not all of the contributing switches are in the set that has already been closed.

Finally, if all the relevant load center RPCs have been closed and nothing has retripped, there are two possibilities. One is that some of the loads beneath the switches that were re-enabled where latched, that is, they shut themselves off when current was cut off due to under voltage tripping and they did not restart when the RPC was re-enabled. The other possibility exists only if any of the relevant loads were not restartable (i.e. if the operator did not give permission to test when describing the load/activity to the scheduler). In this case these load circuits were not tested. It is thus possible that any of these RPCs, connecting cables or loads had faults that would cause the original problem. It is also possible that any or all of these loads could have been contributing to an overload.

5.7.2.2.5 Software Configuration

FRAMES is implemented on the Xerox 1186 computer using the LISP programming language and utilizing Portable Common Loops (PCL is an implementation of CLOS and is not related in any way to Xerox LOOPS) for object oriented programming primitives. The structural configuration of FRAMES is given in Figure 5.7.2.2.5-1:

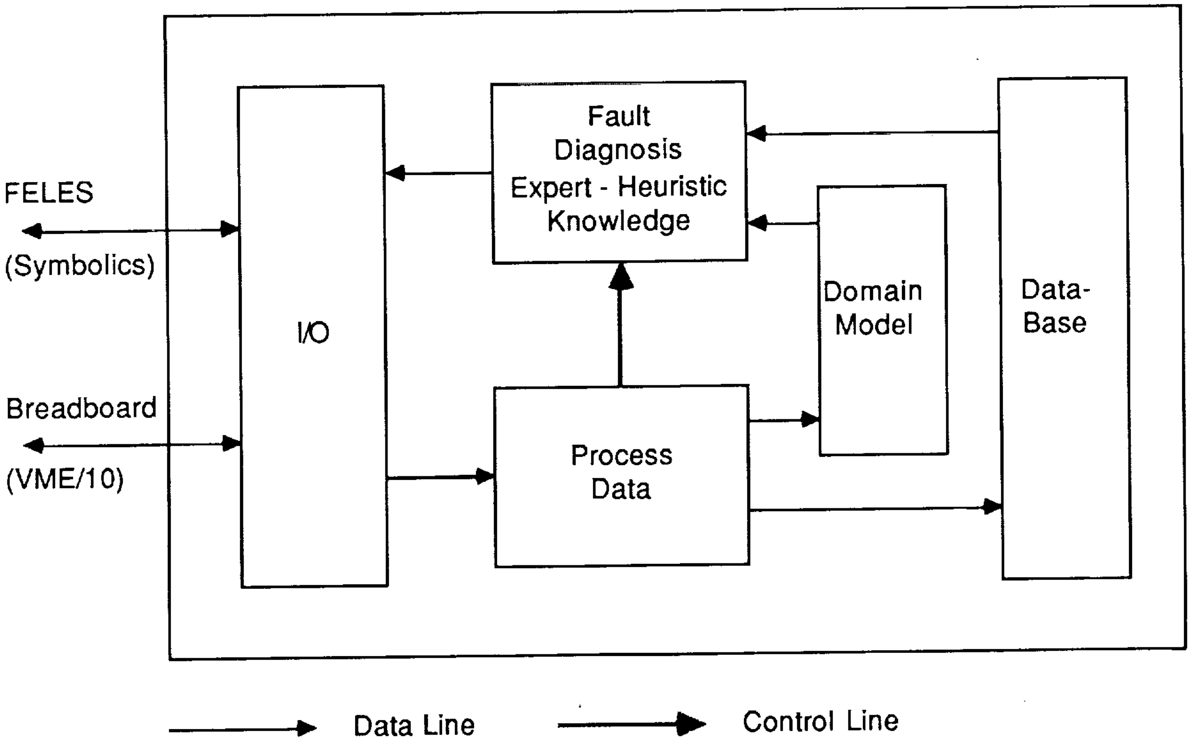


Figure 5.7.2.2.5-1 FRAMES Software Configuration

In the figure there are five main components making up FRAMES. FRAMES interfaces to the rest of the world by using two streams: The FELES Symbolics 3620 D stream and the Motorola VME/10 breadboard stream. All autonomy communication occurs through these streams. Status information, diagnoses, etc. occur through the screen and keyboard as described in the user interface section.

The I/O module is responsible for handling the communications to the other computers. It sets up transactions for transmission and queues them up on an output queue. It also receives transactions and parses them into internal representations. These transactions then get queued up on a receive queue.

The Process Data module receives transactions from the other computers and processes them. This module is entirely data driven. Each transaction invokes a different piece of code for its handling. Transactions from the LLPs mostly consist of switch and sensor data. These data are then stored in the model of the power network and the local database for use by the Fault Diagnosis module. Other transactions, for example, the Ready? and Event List transactions from the Symbolics 3620 D also get processed here.

The Domain Model module is an object oriented representation of the power system network. This is analogous to the visual representation given on the screen. Cables, switches, and loads are all objects and have a variety of slots associated with them for storing fault data, switch status, etc.

The Database module is a simple database used for organizing the data FRAMES uses for operation. There are basic store and retrieve functions associated with this database. When data are stored they are also time stamped. Data are never overwritten and previous elements of data may be accessed by specifying the appropriate time stamp.

Finally, the Fault Diagnosis module is the main portion of the automation software of FRAMES for handling fault situations. This is where the expert/heuristic knowledge is stored. As symptoms from the LLPs are detected FRAMES is triggered to analyze the symptoms and respond appropriately. FRAMES sends down fault event lists to the LLPs when it decides to further isolate the possible location of a fault. When a fault is diagnosed, FRAMES communicates the appropriate information back to the Symbolics 3620 D.

5.7.2.3 User Interface

The user interface to FRAMES is depicted in Figure 5.7.2.3-1. The user interface consists of six distinct windows.

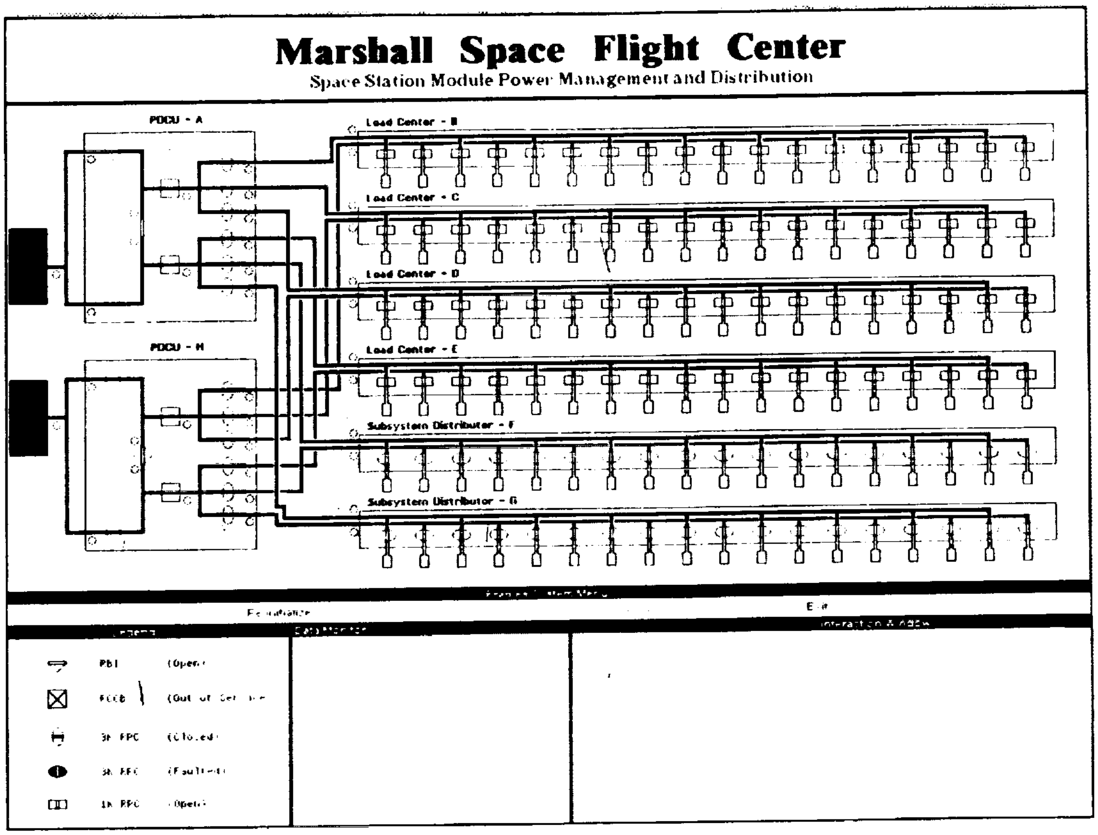


Figure 5.7.2.3-1 FRAMES User Interface

The title window is located at the top of the screen and simply displays the title.

The main window depicts a schematic of the power system being modeled and operated autonomously. A large number of the objects on the schematic are mouse sensitive. Each of the load centers and subsystem distributors may be selected by pressing down and holding the left mouse button to get a menu of operations. The load centers and subsystem distributors may be closed and opened simply for viewing purposes. Each of the sensors may be selected and examined for current and voltage. The switches may also be examined providing current, voltage, switch state, and tripped information. Any recent data transmitted from an LLP about a switch may also be examined from the switch. The cables may also be examined letting the user know if they are powered. There is a second mode to operation of the schematic window. In super-user mode, the various objects may also be inspected and modified. This operation alters the state of the model and thus no longer guarantees correct system operation. To enter super-user mode one types: (super-user t) at the interaction window.

The Menu window, below the schematic window, provides for exiting and reinitializing FRAMES. These options are selected simply by left clicking on them.

The Legend window describes the meaning of the symbols that appear on the Schematic window.

The Data Monitor window is used to display switch data coming from the LLPs. It is simply a monitor window for watching what data the system is receiving. These data are also stored in the database and are also accessible through examination of the switches.

Finally, the Interaction window is where system operations and diagnoses are displayed. As FRAMES recognizes a fault in the system, it displays messages describing what it is currently doing, e.g. opening and closing switches. When FRAMES has made a diagnosis, the diagnosis is also displayed here.

5.8 Power Distribution Management

Power distribution within the SSM/PMAD breadboard is functionally broken down into five separate categories. First, the power distribution control unit which distributes power to the load centers. Second, load centers which distribute power to the loads. Third, switchgear which physically controls the flow of power. Fourth, automation which is comprised of the hardware, interfaces and software required to run a load center or a PDCU. And last, redundancy management which is controlled at the load center level. These areas within power distribution management shall now be discussed.

5.8.1 Power Distribution Control Unit

A power distribution control unit (PDCU) controls power flow to six load centers. At the top level of the power system, a PDCU has three, 25kW remote bus isolators (RBIs) to control power flow from the source. Below the RBIs are two, 10kW remote control circuit breakers (RCCBs). And below each RCCB there are three, 3kW remote power controllers (RPCs). The PDCU also contains thirteen sensors of various power ratings throughout the architecture. These sensors facilitate monitoring the power flow into, within, and out of the PDCU. The function of the PDCU is to provide power to load centers and keep each load center's power independent. Furthermore, each load center may be isolated from the source at the PDCU RPC which feeds it. In this manner, the PDCU monitors, controls, and distributes power to the load centers.

5.8.2 Load Center

A load center monitors and controls attached loads. Load centers come in two varieties, those which contain 1kW RPCs and those which contain 3kW RPCs. A 3kW load center is called a subsystem distributor and a 1kW load center is referenced as just a load center. Both load centers and subsystem distributors perform the same functions. A load center contains twenty-eight, 1kW RPCs and draws power from two PDCUs. Fourteen RPCs draw their power from each PDCU. This means there are two power busses within any load center. At the input of each power bus, a sensor monitors power flow into the load center. This sensor is controlled by the load center. With this sensor and the RPCs, the load center may control whether or not a load receives power and monitor how much current it draws.

5.8.3 Switchgear

Switchgear is a general term for all the power system hardware which takes the commands and gathers data for PDCUs and load centers. This hardware shall now be described in greater detail.

5.8.3.1 Switchgear Interface Control

A switchgear interface control (SIC) card communicates with an LLP, fourteen generic controller (GC) cards, and an analog to digital (A/D) card. The SIC has nineteen different commands which it is designed to handle. The SIC receives these commands from its controlling LLP. The commands are described in the SIC/LLP interface control document in Appendix VII. The SIC communicates with the fourteen GCs for switching and trip information. Each GC may control an RBI, an RCCB, or an RPC. The SIC also communicates with the A/D card and receives sixteen voltage, current, and temperature sensor data inputs.

5.8.3.2 Generic Controller

A generic controller (GC) card controls the switching operation of an RBI, RCCB, or RPC and returns the switch status information to the SIC card. The GC receives command data information from the SIC and commands the switch on or off or does nothing based on the command. Additionally, the GC card processes analog signal information passed to it from the switch and decides whether or not to trip the switch. Conditions which warrant the GC tripping off a switch are Under Voltage, Over Current (I^2t), Surge Current, Ground Fault, or Over Temperature. Moreover, the GC contains an Over Temperature warning, a current limit switch turn on processor, and zero voltage and current crossing detectors.

5.8.3.3 Analog to Digital Card

The analog to digital (A/D) card accepts sixteen voltage, current, and temperature sensor inputs and returns proportional digitized information to the SIC card. The A/D card processes the analog sensor inputs and returns digitized root mean square (RMS) voltage, RMS current, DC voltage, DC current, frequency, average power, instantaneous power, power factor, and temperature data to the SIC card.

5.8.3.4 Remote Power Controller

A remote power controller (RPC) provides 5 amperes (1kW) or 15 amperes (3kW) at 208 Vrms, 20 kHz to any resistive, capacitive, or inductive load. The switch is single pole single throw (SPST) with a main solid state switch, a parallel current limiting switch, and a relay isolator. The RPC provides the GC with analog current, voltage, ground fault, temperature sensor inputs. In addition, the RPC provides the GC with positional information for the solid state switch and the relay isolator. The RPC also contains a self-protection circuit to protect itself from a quick current surge greater than 400% of its normal peak current. The GC commands the RPC on and off.

5.8.3.5 Remote Controlled Circuit Breaker

A remote controlled circuit breaker (RCCB) provides 50 amperes (10kW) at 208 Vrms, 20 kHz to up to 3 fully loaded 3kW RPCs. The switch is SPST and consists of a large relay which switches both the positive and return sides of the 20 kHz power. The RCCB may be switched "hot" and provides the GC with analog current sensor data and relay status information. The GC commands the RCCB on and off.

5.8.3.6 Remote Bus Isolator

A remote bus isolator (RBI) provides 25kW, 208 Vrms at 20kHz to the RCCB switches. The switch is SPST and consists of a large relay which switches both the positive and return sides of the 20kHz power. The RBI may not be switched "hot" and provides the GC with relay status information only. The GC commands the RBI on and off.

5.8.3.7 Switchgear Calibration Algorithms in LLP Software

Within the LLP software, conversion constant values for digitized switch current and sensor information are hard coded. Within the PDCU there are thirteen sensors; these sensors are rated for 15, 50, and 125 amps. The 15 amp sensors are below the 3kW RPCs within the PDCU. The 50 amp sensors are below the RCCBs. The rest of the sensors are of the 125 amp variety. The load centers contain two sensors of the 15 amp type. The RCCBs and RPCs all provide switch current data based on a 10kW, 3kW or 1kW power rating. The LLP software takes the digitized data and produces the appropriate value based on system topology.

At present, the LLPs receive digitized data which at full scale is 200% of the rated current or power. So for the switches, the conversion factors are as follows:

1kW RPC	$0.377\ \frac{dA}{div}$	$\frac{\text{(deciAmps)}}{\text{(division)}}$
3kW RPC	$1.13\ \frac{dA}{div}$	
10kW RBI	$3.77\ \frac{dA}{div}$	

The sensor data are converted as follows:

Vrms	$1.63\ \frac{V}{div}$	
Irms	$.118\ \frac{A}{div}$	(15 A sensor)
	$.392\ \frac{A}{div}$	(50 A sensor)
	$.980\ \frac{A}{div}$	(125 A sensor)

Note: A division is defined as one least significant bit of the returned digitized byte from the SIC card. The unit of deciAmps is used because of the communication protocol with the Symbolics 3620 D. One deciAmp equals one tenth of an Amp.

5.8.4 Automation

Automation as it relates to power distribution covers three topics. First, the hardware platforms under which the algorithmic software executes. Second, the interfaces that connect the hardware platforms with each other, the switchgear, and the high level expert systems. And last, the actual algorithmic software. Although scheduling and fault management and recovery come from the expert systems, this is where the power system is monitored and controlled.

5.8.4.1 Automation Hardware

One of the two platforms for automation hardware is the Motorola VME/10. The Motorola VME/10 is a Motorola 68000 based, 32 bit machine, with multi-tasking capability and running the VersaDos operating system. The Motorola VME/10 algorithmic software is written in Pascal. This computing platform contains a VME bus backplane into which extra VME cards may be added. In the back of the automation Motorola VME/10 there are 2 MVME-331 intelligent communications controllers, 2 MVME-705 6 port serial communications cards, an extended memory card, and an MVME-400 dual port RS-232 serial communications card. The communications algorithmic controller (CAC) software is based on the Motorola VME/10.

The other platform is the lowest level processor (LLP). The LLP is comprised of three VME bus cards and a VME bus rack mount chassis. The first card is an MVME-107 68010 based single board computer with 512k Bytes of on board random access memory. The processor on this card gives the LLP the computing horsepower of a 32 bit processor. The second card is an MVME-331 intelligent communications controller. This card communicates with the MVME-107 over the VME bus and directly to the MVME-705 card. Last, the MVME-705 6 port serial communications card is controlled by the MVME-331 card. The load center and PDCU software running on the LLP platforms was also developed in Pascal under PDOS.

5.8.4.2 Interfaces

Several interfaces exist within the automation system which permit data transactions. The CAC software on the Motorola VME/10 communicates with FRAMES on the Xerox 1186 over an RS-232 communication link through the MVME-400 card. The CAC software also transacts with the LLPs running load center on PDCU software. This communications link is RS-422 and passes through MVME-331 and MVME-705 cards at both ends. In addition, the LLPs have two more RS-422 links using four wire operation to the switchgear. To conclude, there are three data system interfaces between the switchgear and FRAMES.

5.8.4.3 Software

Algorithmic software may be broken down into three distinct subsystems. First, there is the communications algorithmic controller software on the Motorola VME/10. Second, the LLPs running load center algorithmic software. Last, the LLPs running power distribution control unit software. The functionality of these different algorithmic software systems shall now be discussed.

5.8.4.3.1 Communications Algorithmic Controller Software

The communications algorithmic controller (CAC) software runs on the Motorola VME/10 and operates primarily as a parser and server for the LLPs. All messages from FRAMES, LPLMS, and FELES directed to the LLPs must pass through the CAC software before distribution to the LLPs. These messages have a global frame of reference and the CAC must parse out each message for every LLP. In this manner, each LLP only receives messages with information pertinent to its local frame of reference. Conversely, when an LLP sends a message to FRAMES, the CAC software inserts the global address of the LLP at the beginning of the message. The message FRAMES receives has local information but is globally located. To conclude, the CAC software provides an interface between local LLP software and global expert systems.

5.8.4.3.2 Load Center Algorithmic Software

The load center (LC) software commands and monitors the switches within its domain. The LC algorithmic software is initially downloaded to the LLPs from the CAC and sent an event list, priority list, and time list. At this point the LC software enters its control loop. First, the software updates the event list and priority list, if necessary, and performs any scheduled operations in the event list. Next, the software strobes the topology hardware for all switch and sensor data. Then, the software inspects the switch and sensor data for hard faults and anomalous conditions. The software also computes short term statistics from the data. If switches have been commanded on or off, or if an anomalous condition or hard fault has occurred, the LC software informs FRAMES and receives any new instructions. If a new message has come down from the CAC, it is processed. At this point, the cycle repeats. The LC software performs as an intelligent slave commanding and monitoring the topology hardware for the upper level expert systems.

5.8.4.3.3 Power Distribution Control Unit Algorithmic Software

The power distribution control unit (PDCU) software commands and monitors the switches within its domain. The PDCU algorithmic software is initially downloaded to the LLPs from the CAC and sent an event list and time list. At this point the PDCU software enters its control loop. First, the software updates the event list, if necessary, and performs any scheduled operations in the event list. Next, the software strobes the topology hardware for all switch and sensor data for hard faults and anomalous conditions. The software also computes short term statistics from the data and searches for soft faults based on Kirchoff's Current Law. If an anomalous condition or hard fault has occurred, the PDCU software informs FRAMES and receives any new instructions. If a new message has come down from the CAC, it is processed. At this point, the cycle repeats. The PDCU software performs an intelligent slave commanding and monitoring the topology hardware for the upper level expert systems.

The load center and power distribution control unit software discussed above contains many procedures and functions. Some of these functions have been flowcharted with visual control logic representations (VCLRs). Many of the VCLRs are the same for load center and PDCU software. Those which do differ, are so labeled. The following VCLRs are included in Appendix IX:

1. LCMAIN - Load center main program.
2. PDCU MAIN - Power distribution control unit main program.
3. ALGORITHMS - Check switches and sensors for trips and anomalous conditions.
4. CURRTIME - Returns present mission time in seconds.
5. CVTDAYANDSEC - Returns Julian day and seconds from date and time.
6. CVTTIME - Return numerical values from system date and time.
7. GETTIME - Get system date and time.
8. SETCLOCK - Set system time and date and store start of mission time and date.
9. CALCENERGY - Compute power system performance statistics.
10. DOSCHEDULE - Implement new event and priority lists and execute events.
11. GETALLDATA - Get switch and sensor data from switchgear.
12. MANUAL MODE - Translate Motorola VME/10 commands to SIC card.
13. UPDATE CONTINGENCY LIST - Updates schedule with a contingency list.
14. UPDATE PRIORITIES - Updates priority list.
15. UPDATE SCHEDULE - Buffers new event list and sets implementation time.
16. CHANGE SCHEDULE - Implements new event list.
17. INIT_BUF_QUEUE - Initializes event list buffer queue.

18. INSERT_SCHED - Inserts an event list in buffer queue.
19. REMOVE_BUFFER - Removes a buffer from the queue.
20. CONFIGURE_SIC PORTS - Configures SIC ports.
21. SEND/RECEIVE SIC DATA - Sends and receives SIC data.
22. CMNDSWITCH - Turns on/off switches and sets trip flags accordingly.
23. LOADSHED - Load shedding based on priority.
24. REDSW - Redundant switching.
25. CHANGE SWITCH STATE - Change power limits on a switch.
26. SWITCH-ON - Turn on switch.
27. UNCONDITIONAL OFF - Turn off switch.
28. DECODE - Decrypts Motorola VME/10 communication incoming.
29. ENCODE - Encrypts Motorola VME/10 communication outgoing.
30. GET Motorola VME/10 DATA - Assigns VME/10 input next data space in buffer.

5.8.5 Redundancy Management

Certain loads require redundant sources of power to make sure they keep operating. The load centers power redundant loads with one switch off each power bus within the load center. If a switch trips with a load which is redundant and the LLP has permission to switch to redundant, it attempts to power the redundant switch. If sufficient power is available on the redundant bus, the redundant switch is powered. If lower priority loads may be shed to provide enough power for the redundant switch, the loads are shed and the redundant switch is powered. If there is not enough power available, the redundant switch is not powered. This method of redundant switching allows for failure of a power bus, since the redundant switch is on the redundant power bus. Load centers are in control of redundant switching and providing redundant power to loads.

5.9 Operation Considerations

The operation of the ACM/PMAD involves a complex set of user and system interactions, both with the hardware and the software. The current implementation provides quite an open system architecture. Careful consideration should be exercised in operating the system in order to assure personnel safety and system integrity. The system is built around a 20 kHz, 208 volt source which can be dangerous if not operated properly.

The present system has the capability for extension to handle multiple faults. It currently handles only single faults with predictable outcomes. When setting up system operation regimes, single fault scenarios should be used. The lowest level hardware activated switching should work under multiple faults, but FRAMES may try to perform exotic switch testing activities for which there needs to be knowledge base enhancement for reasonable and assured performance.

Communication of switch activity arrives at the Xerox 1186 in what may appear to be an extended amount of time. However, the system is reacting to the presence of data in a reasonable amount of time when the integration of the information at the CAC and the FRAMES is considered.

5.10 Breadboard Timing Considerations

When automating the breadboard, certain timing considerations must be addressed. First, switching operations within any event list are treated separately. This leads to events being processed sequentially and therefore events scheduled for the same time are not executed simultaneously. Because the automation system is distributed, the LLPs do not switch at the same time or necessarily in the same order. Hence, a load center switch could switch before its corresponding switch in the PDCU giving erroneous under voltage faults. For this reason, the PDCU switches are initially powered on and PDCU event lists only change the value of scheduled power through a given switch. Last, each access to the SIC card can take as much as 2 1/4 seconds (the timeout). As a result, each additional access to the switchgear increases the control loop cycle time. Timing within a distributed automated processing system can have serious consequences if not considered and addressed ahead of time.

There are also automation system timing constraints which must be considered. On the average it takes seven to ten seconds for a change in switch position within an LLP to filter back up to the FRAMES interface. However, FRAMES may only process one transaction at a time and this could lengthen the time it takes for LLP data to be processed. It takes approximately ten seconds for FRAMES to process a transaction from the LLPs and be ready for the next transaction. When FRAMES probes the LLPs for fault isolation data it could take up to a minute and one-half before the SIC data are returned. This time too, could be extended if FRAMES is busy processing a transaction from another LLP. Naturally, with more LLPs, the probability of FRAMES receiving intermediary transactions increases. All data presented in this section is empirical in nature from testing performed on the breadboard.

5.11 Manual Override

It is desirable in an automated system for human beings to be able to take over control of the system. In the event that the automated system fails, it is necessary for a user to be able to take control. This control on SSM/PMAD comes through a manual override interface. When in manual mode of operation, the user may access the entire breadboard. This means the user may turn on or off and have control over all switches manually. Moreover, the user must have working knowledge of the power system because switching PDCU switches can have major consequences on load centers. The user also has access to all breadboard sensors. This form of manual intervention removes automation from the breadboard, but permits a user to directly control the breadboard.

5.11.1 Operation

The manual mode of operation interface is accessed at the Motorola VME/10. When a command is entered at the interface, the Motorola VME/10 converts the command to the appropriate four byte command string and sends it to the appropriate LLP. The four byte commands are defined in the SIC/LLP interface control document in

Appendix VII. Upon receipt of the command string at the LLP, the LLP forwards the command to the SIC card and waits for a response. When the response comes, the LLP returns the data to the Motorola VME/10. The Motorola VME/10 takes this data and displays it to the user interface screen. Manual mode of operation directly accesses the SIC card and interprets the responses.

5.11.2 Implementation

The manual mode interface was implemented with ease of use in mind. The entire user interface is menu-driven. Each menu gives the user the ability to change the LLP and SIC being accessed. Each sub menu defaults to returning to the main menu. Each menu gives the user the option of shutting off all switches on the currently designated LLP and SIC. The manual mode user interface implementation was created to facilitate user access to the breadboard.

5.11.3 User Interface

The manual mode user interface may be accessed after the initial event list, priority list, and time list have been received at the LLPs. The interface is initiated by pressing a carriage return at the Communications Algorithmic Controller (CAC) when in normal operation. The following menu will appear on the CAC monitor upon entry into manual mode:

MANUAL OVERRIDE MENU

1. Switchgear Interface Card (SIC) RESET
2. Generic Card (GC) SELECT
3. Switch, Power Sensor, & Temp Sensor DATA
4. Temperature Sensor MENU
5. Switch MENU
6. Power Sensor MENU
7. Select LLP
8. Select SIC
9. KILL all Switches
10. QUIT

SELECT A FUNCTION (1 TO 10)

The user must then select an LLP before performing any operations on that LLP. This selection process notifies the LLP software to enter its manual mode procedure and notifies the CAC of which LLP to query. The CAC will also inform the selected LLPs of exit from manual mode so they may recycle their software.

The user interface also contains three sub-menus which will now be discussed. First, the temperature sensor menu gives the user access to all temperature data from the selected LLP and SIC. The menu that appears on the CAC monitor is as follows:

TEMPERATURE SENSOR MENU

1. All temperature sensors DATA
2. MONITOR all temperature sensors
3. KILL all Switches
4. Select LLP
5. Select SIC
6. Manual Override Menu RETURN (default)

SELECT A FUNCTION (1 TO 6)

Second, the user might select the switch menu which furnishes the following menu:

SWITCH MENU

1. Switch(es) RESET
2. Conditional Switch(es) ON
3. Conditional Switch(es) OFF
4. UNconditional Switch(es) ON
5. UNconditional Switch(es) (OFF)
6. SELECTED Switch DATA N Times
7. All Switches DATA N Times
8. CONTINUOUSLY switch switch
9. MONITOR selected switch
10. Select LLP
11. Select SIC
12. KILL all Switches
13. Manual Override Menu RETURN (default)

SELECT A FUNCTION (1 TO 13)

This menu allows the user to turn on or off any switch or group of switches on the selected LLP and SIC. Likewise, the user may monitor the state of any switch on the selected LLP and SIC. Last, the user could select the power sensor menu. This menu allows the user to monitor the power sensors and get the power factor data for any sensor. This menu looks as follows:

POWER SENSOR MENU

1. SELECTED Sensor DATA N Times
2. ALL Sensors DATA
3. Power Factor DATA
4. MONITOR selected power sensor
5. KILL all Switches
6. Select LLP
7. Select SIC
8. Manual Override Menu RETURN (default)

SELECT A FUNCTION (1 TO 8)

All of the menus have the option of selecting a new LLP and a new SIC, providing easy manipulation of the breadboard.

5.12 ACMPMAD Test Plan

This is the Task III Test Plan.

5.12.1 GENERAL

5.12.1.1 Purpose of the Test Plan

The Test Plan for ACMPMAD, contract NAS 8-36433, program is written to:

1. Provide guidance for the management and technical effort necessary throughout the test period.

2. Establish a comprehensive test plan and communicate to the user the nature and extent of the tests to provide a basis for evaluation of the system.

5.12.2 Project References

The documents utilized by this contract are:

1. Contract Agreement, NAS 8-36433, dated June 25, 1985.

2. Software Development Plan for ACM/PMAD, Revision A, dated February 1987.

3. SSM/PMAD Breadboard Usage Plan, MCR-88-624, dated September 1988.

4. LLP/SIC Interface Document, dated August 1988.

5.12.3 Acronyms

AI	Artificial Intelligence
ACMPMAD	Automation of Common Module Power Management and Distribution
CM/PMAD	Common Module/Power Management and Distribution
FRAMES	Fault Recovery and Management Expert System
LLP	Lowest Level Processor
LPL	Load Priority List
LPLMS	Load Priority List Maintenance System
MB	Megabyte
MSFC	Marshall Space Flight Center
RPC	Remote Power Controller
SIC	Switchgear Interface Card
SSM/PMAD	Space Station Module Power Management and Distribution

5.12.4 Equipment Requirements

The hardware and software required to run the tests are identified in this section.

5.12.4.1 Hardware

The SSM/PMAD communications architecture (Figure 5.12.4.1-1) is identified below:

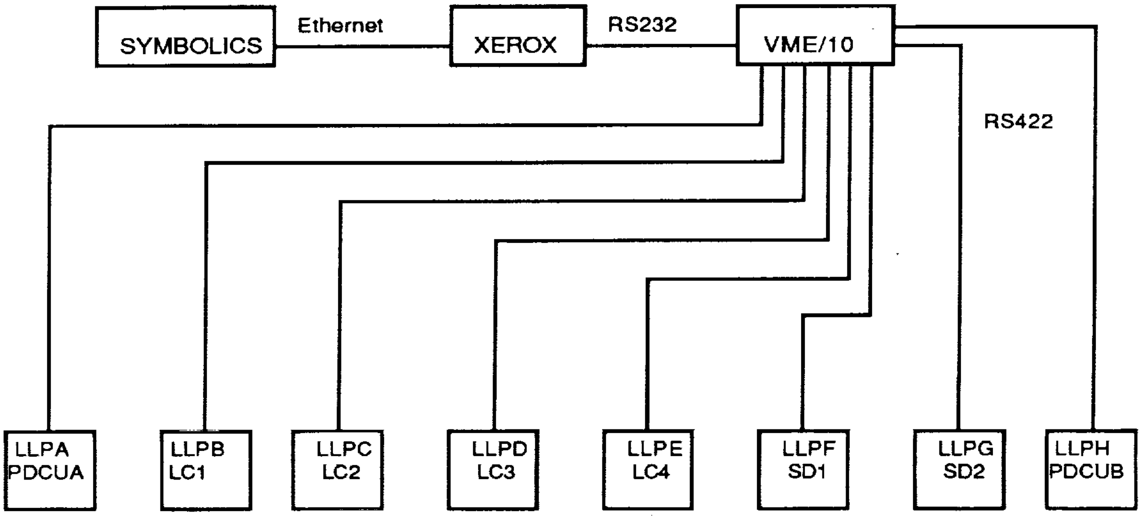


Figure 5.12.4.1-1 SSM/PMAD Communications Architecture

1. Symbolics 3620 D 3640.

2. Xerox 1186 AI workstation with CLOS development environment with 3.5 MB RAM, 80 MB Hard Disk and IBM PC Floppy Output Capability.

1. Motorola VME-10 development system.

The SSM/PMAD breadboard block diagram is shown in Figure 5.12.4.1-2 below:

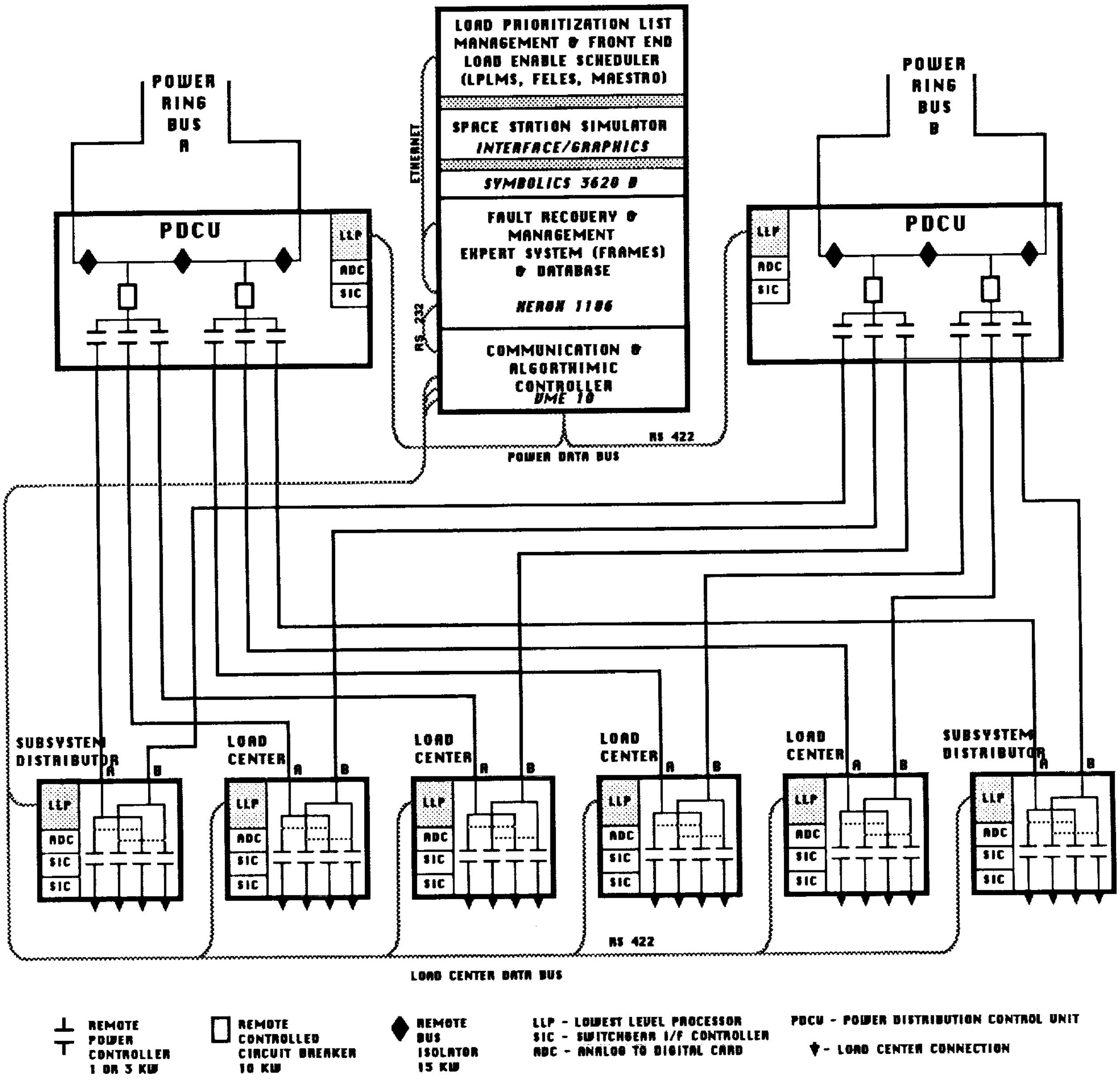

LOAD PRIORITIZATION LIST MANAGEMENT & FRONT END LOAD ENABLE SCHEDULER (LPLMS, FELES, MAESTRO)
SPACE STATION SIMULATOR INTERFACE/GRAPHICS
SYMBOLICS 3620 D
FAULT RECOVERY & MANAGEMENT EXPERT SYSTEM (FRAMES) & DATABASE
XERON 1186
COMMUNICATION & ALGORTHIMIC CONTROLLER
VME 10
ETHERNET
RS 232
RS 422
POWER DATA BUS
POWER RING BUS A
POWER RING BUS B
PDCU
LLP
ADC
SIC
SUBSYSTEM DISTRIBUTOR
LOAD CENTER
A
B
RS 422
LOAD CENTER DATA BUS
REMOTE POWER CONTROLLER 1 OR 3 KW
REMOTE CONTROLLED CIRCUIT BREAKER 10 KW
REMOTE BUS ISOLATOR 15 KW
LLP - LOWEST LEVEL PROCESSOR
SIC - SWITCHGEAR I/F CONTROLLER
ADC - ANALOG TO DIGITAL CARD
PDCU - POWER DISTRIBUTION CONTROL UNIT
- LOAD CENTER CONNECTION

Figure 5.12.4.1-2 Breadboard Block Diagram

5.12.4.2 Software

The configuration of the software is identified below:

1. Versados 4.51 Object, 849CMP60004

2. Versados VMELAN 2.1, 849CMP60005

3. Motorola VME/10 Graphics Server and Demo 1.01, 849CMP60002

4. Pascal 2.3, 849CMP60001

5. Xerox Lisp: Lyric Release for the 1186 - Lisp.Sysout, 849CMP71001

6. Xerox Lisp: Lyric Release for the 1186 - Lyric-Patch-1, 849CMP71002

7. Xerox Lisp: Lyric Release for the 1186 - Lyric-Library, 849CMP71003

8. Xerox Lisp: Lyric Release for the 1186 - Installation Utility, 849CMP71004

9. Xerox Lisp: Lyric Release for the 1186 - System Files, 849CMP71005

10. Xerox Offline Diagnostics Master Disk 1.3e #1, Rev. Lyric, 849CMP71006

11. Xerox Offline Diagnostics Boot Diagnostics Master Disk 1.3e #2, Rev. Lyric, 849CMP71007

12. PCL-CLOS, 849CMP50002 version dated 8/27/87,

13. Symbolics Operating System Genera 7.1

14. TCP/IP Protocol

15. VME-10 Algorithmic Software, 849CMP10000

16. SSMPMAD Symbolics Interface V1.0 Software, 849CMP20000

17. Xerox FRAMES Software, 849CMP15000

5.12.5 Test Description

This test is designed to demonstrate the capabilities of the ACMPMAD system. It is intended to satisfy the requirements of Activity (2) of Task II, Activity (2) of Task III, Activity (5) of Task III and Activity (1) of Task IV. All test requirements are satisfied by demonstration.

5.12.5.1 Test Data

5.12.5.1.1 Input Data

Input data are controlled by this test in order to demonstrate system capabilities. When the word "type" is used, a Return is implied unless specified otherwise.

5.12.5.1.2 Output Data

Output data will consist of schedule data.

5.12.5.2 Test Procedures

5.12.5.2.1 Test Setup

5.12.5.2.1.2 SSMPMAD Interface Software - Symbolics 3620 D

	ACTION	RESPONSE
1.	Turn on power	Screen appears.
2.	Place the SSMPMAD Symbolics Interface tape in the Symbolics 3620 disk drive.	No observable reaction.
3.	Load the SSMPMAD Symbolics Interface tape by typing "(tape:carry-load)"	A list of directories appear and the user is prompted whether to load all of the files or just selected ones.
4.	Type "Q" or "S" for selective load	Prompt appears.
5.	Type "y" to 'pmad:si;ssmpmad. translations' only	pmad:si; ssmpmad. translations is loaded.
6.	Edit the translation file to change the physical host to *SSMPMAD-PHYSICAL-HOST*	Physical host is changed
7.	Evaluate the buffer. Type "META SHIFT-E"	Prompt appears.
8.	Save the file	File is saved.

9.	Type "(tape:carry-load)"	A list of files appear and the user is prompted to load all of the files or just selected ones.
10.	Type "Q" or "S" for selective load	Prompt appears.
11.	Type "y" to the following files: pmad:si;*.lisp pmad:ui;*.lisp pmad:pmad;*.lisp pmad:lplms;*.lisp pmad:developer;*.lisp pmad:schedule-library;*.* pmad:library;*.* pmad:si;*.bin pmad:ui;*.bin pmad:pmad;*.bin pmad:lplms;*.bin pmad:scheduler;*.*	The specified files are loaded.
12.	To load the system into the Symbolics 3620 D Environment, type "(load "host:> ssmpmad>load")	The system is loaded into the Symbolics Environment.

13. Type <SELECT> "A"	Screens are displayed and everything is initialized.

Q____

5.12.5.3 FRAMES Software - Xerox 1186

1.	Turn on power	Screen appears with row of buttons displayed on bottom of screen.
2.	Press "F1" key (boot from local hard disk)	After short interval of time, Xerox is booted; normal display is seen on screen.
3.	Display the background menu by holding down the Right Mouse Button while over the "Background".	Background Menu is displayed.
4.	Select the File Browser by highlighting "File Browser" in the Background Menu with the cursor and releasing the previously held down button.	A default empty window shape is displayed for the user to shape to the preferred size.
5.	Select the window position and shape by moving mouse to the preferred position, holding down the Left Mouse Button, dragging the right-bottom corner	A File Browser window is displayed. It is waiting for input as to what location to display files from.

of the window to size it, and release the mouse button

6. Insert the FRAMES Disk 1 into the floppy disk drive and close the floppy door.

   No observable reaction.

7. Type "{floppy}".

   The contents of the disk in the floppy drive is displayed.

8. Select all the files on the disk by positioning the cursor over the topmost file entry and clicking the Left Mouse Button followed by positioning the cursor over the bottom-most file entry and clicking the Right Mouse Button.

   Each file has a ">" pointing to it.
   On the top-right portion of the File Browser Window, a menu of actions is displayed.

NOTE: To get to the bottom-most file entry may require scrolling the window. To scroll the window, slowly move the cursor to the left of the window and stop when the scroll bar appears. Clicking the Left Mouse Button will display the current line (next to the cursor) at the top of the window. When the bottom-most file is displayed, discontinue this action.

9. Click the Left Mouse Button on the "Copy" selection. pathname to copy the files to.

The top of the File Browser Window queries for a

10. The destination pathname is variable. It always starts with "{dsk}".
To figure out the rest of it, look at the locations of the files on the floppy as the hard disk displayed in the File Browser. The rest of the data name should correspond to the greatest level of common pathname of the files on floppy

For example, if all the files on the floppy are under "<lispfiles>comm>" that should be the completion to arrive at: "{dsk}<lispfiles>comm>". If there were files under both "<lispfiles>comm>" and "<lispfiles>data>", the greatest completion will be "<lispfiles>" to arrive at "{dsk}<lispfiles>". Once the pathname is entered, type a Return.

The selected files are copied from the floppy to

The idea files behind this is to copy the files on the floppy to the same relative pathnames on the hard disk

When copying is completed, the copy selection on the File Browser is not highlighted.

The user is prompted whether to retain the subdirectories.

11.	Type <Return>	
12.	Remove the floppy from the floppy drive and insert the next floppy in its place.	No observable reaction.
13.	Display the contents of the floppy by clicking the Left Mouse Button on the "Recompute" option of the File Browser.	The files of the current floppy are displayed in the File Browser Window
14.	Repeat Steps 8 through 13 until all the files from all the floppies are copied to the hard disk.	All the files are copied to the hard disk
15.	Move the cursor to the top portion of the File Browser Window. Hold down the Right Mouse Button and highlight the "Close" option of the window menu. Release the mouse button.	The File Browser window is no longer visible.
16.	Type "(load '{dsk}<lispfiles> frames>init>load-all.lisp)"	The corresponding file is loaded.
17.	Type "(load-all)"	The files for FRAMES are loaded into the Lisp World. A display window appears showing the files being loaded.

18.	When the Display Window inverts, move the cursor into the window and click the Left MouseButton.and closing.	Numerous things occur, including windows opening When the
	Then press the Space-Bar twice in quick succession. Interaction Window returns further to scroll old data off the top of the window.	This indicates that the user does not want to be queried with a normal interaction prompt ("XXX>" where XXX is some integer), FRAMES is finished loading. NOTE: It takes approximately 30 minutes to load the system.
19.	To logout, type "(il:logout)"	The screen reverts to the status of Step 1.
20.	Turn off power.	System is turned off.

Q_____

5.12.5.4 Test Initialization

There are four component types to initialize: Symbolics 3620 D, Xerox 1186, Motorola VME/10 and LLPs.

The simplest procedure (until familiarity with how the system works is acquired) is to initialize the four component types in the following strictly linear order:

5.12.5.5 Power Up the Symbolics 3620 D, Xerox 1186 and Motorola VME/10

1.	Turn on the monitor (switch in back) on the Symbolics 3620 D.	Monitor is powered up.
2.	Turn on the main power on the Symbolics 3620 D.	The user is prompted with a "FEP Command:" prompt.
3.	Type "hello" on the Symbolics 3620 D.	The user is prompted with a "FEP Command:" prompt.
4.	Type "FEP0:>PMAD.BOOT.1" on the Symbolics 3620 D.	The system boots up with the operating system HERALD. NOTE: The user may be prompted to enter date and time. Time and date will be entered as necessary. A <Return> will be pressed if the date and time is correct. The user is prompted to log in.
5.	Turn on the power to the Xerox 1186.	The screen appears with a row of buttons displayed near the bottom of the screen.

6.	Press the "F1" key on the Xerox 1186.	The system boots up. The screen looks like it was the last time a user logged out.
7.	Power up Motorola VME/10.	VME/10 is powered up.
8.	Power up Motorola VME/10 Monitor and Wyse terminal.	VME/10 monitor and Wyse terminal are powered up. "Waiting for disk to spin up" message appears on VME/10 monitor. "Power Up Test Complete" message appears on VME/10 monitor. System automatically boots up and configures all the ports on the VME/10. "=" prompt appears on the VME/10 monitor. Wyse terminal completes power up self test.

Q____

5.12.5.6 Initialize the Symbolics 3620 D

1. Type ":login PMAD"

   The PMAD user's initialization file is loaded enabling the Symbolics to communicate with other computers.
   "Do you wish to load SSMPMAD patches?" prompt appears.

2. Type "Y"

   Symbolics loads patches. User is prompted with "(Y, P or N)".

3. Type "P"

   Symbolics proceeds. User is prompted with "(Y, P or N)".

4. Type "P"

   Symbolics proceeds.

5. Press <SELECT> "A".

   The SSMPMAD interface is started.

   When it is finished, the user sees the console screen of the SSMPMAD interface.

6. Press <CONTROL> S.

   Maestro the SS Module Scheduler screen appears.

7.	Select "Reset" to reset the scheduler	Select Length of Scheduling Period menu appears.
8.	Select "8 Hour"	Select Activities for Scheduling menu appears.
9.	Schedule the demo activities by selecting: Demo1 Part1 Demo1 Part2 Demo1 Part3 Demo1 Part4 Demo1 Part5	The Schedule is ready and the FELES is ready to be started.
10.	Click Left Mouse Button on "Highlighted"	The activities are displayed.
11.	Select "Schedule"	The activities are scheduled.
12.	Press <SELECT> L.	Lisp Listener appears.
13.	Type "(set-feles-lead-time 2)	Feles lead time is set to 2. This is an optimization for responding to contingency situations faster.

14. Press <SELECT> A.

Maestro the SS Module Scheduler screen appears.

Q_____

5.12.5.7 Initialize the Xerox 1186

1. Position the cursor over the Background Menu and hold down the Right Mouse Button.

The Background Menu appears.

2. Position the cursor over the "FRAMES" selection of the Background Menu and hold Right Mouse Button (over the small triangle) to highlight the "Initialize FRAMES" option. Release the mouse button.

The FRAMES interface is displayed and initialized. NOTE: This will take 2 minutes at the most.

FRAMES is now ready for normal operations.

Q_____

5.12.5.8 Starting the Schedule

1. Press <META> S on the Symbolics 3620D.

Starts and updates control box. (No visible action).

Q_____

5.12.5.9 Initialize the Motorola VME/10 and the LLPs

NOTE: If the Motorola VME/10 has timed out, press "Reset" on the Motorola VME/10 chassis and wait for the system to reboot.

"=" prompt appears on the VME/10.

Type "CLRBK" on the Motorola VME/10.

"Turn on all LLPs and Press Their Reset Buttons" prompt appears.

2. Turn on LLP's 10, 12 and 13 and press their Reset Buttons (The Red Button).

LLPs are turned on.

Load Center	LLP
A	10
C	12
D	13

3. Press <Return> on the Motorola VME/10

Errors may occur from the ports being cleared which can be ignored.

"=" prompt appears on the VME/10.

4. Type "Off" on the Motorola VME/10.

User is logged out.

5. Press <BREAK> on the Motorola VME/10.

"Enter user no. =" prompt appears.

6.	Type "200.MSFC" on the Motorola VME/10.	"=" prompt appears.
7.	Type "ALGO" on the Motorola VME/10.	"Do you wish to use real Xerox communications?" prompt appears.
8.	Type "Y" on the Motorola VME/10.	"Is CN 10 present?" prompt appears.
9.	Type "Y" to LLPs 10, 12 and 13 and type "N" LLPs 11, 14, 15, 20 and 21 on the Motorola VME/10.	The LLPs selected are to displayed on the VME/10.
		"Do you wish to go back and try again?" prompt appears.
10.	Type "N" on the Motorola VME/10.	"Do you wish to download to CN10?" prompt appears.
11.	Type "Y" to all download questions on the Motorola VME/10 pressing "Reset" on the LLPs unless the download starts immediately on the screen.	Each LLP will be downloaded.
		NOTE: LLP downloads each take approximately 8 minutes.

		"Do you wish to go back and try again?" prompt appears on the VME/10.
12.	Power up power section of the breadboard including 20kHz, 208, nominal and housekeeping power.	Power section of the breadboard is powered up.
13.	Type "N" on the Motorola VME/10.	"Do you wish to use the debug statements?" prompt appears on the VME/10.
14.	Type "Y" on the Motorola VME/10.	"Ensure there is a terminal..."
15.	Press <Return> on the Motorola VME/10.	Wyse terminal will scroll with information. Menu for start of mission time appears on the Symbolics.

Q______

5.12.5.10 Test Steps

5.12.5.10.1 Schedule 1

C1	C5	D1	D5	TIME
ON	OFF	ON	OFF	0
OFF	OFF	ON	ON	2
OFF	ON	OFF	ON	4
ON	ON	ON	ON	6
OFF	OFF	OFF	OFF	38

1. Select a "one minute" delay to start the schedule on the Symbolics 3620 D.

The schedule will begin execution at that time.

At mission time 0, the light bulb at switches C01 and D01 lights. Shortly thereafter, the FRAMES interfaces reflects the power to the load at switches C01 and D01.

At mission time 2, the light bulb at switch C01 is turned off and the light bulb at switch D05 is turned on. Again the Frames interface reflects the power to the load at switches C01 and D05.

At mission time 4, the light bulb at switch C05 lights and the light bulb at switch D01 is turned off. Again the FRAMES

	interface reflects the power to the load at switches C05 and D01.
	At mission time 6, the light bulb at switches C01 and D01 are turned back on. Again the FRAMES interface reflects the power to the load at switches C01 and D01.
2. Inject a short at point A at mission time 9 (Figure 5.12.5.10.1-1).	A04 will trip.
	C01 and C05 will subsequently trip on under voltage.
	Both LLPs will send data to FRAMES indicating these trips.
	When FRAMES receives this data, it will send a message to both LLPs to turn off all the switches from A04 and below.

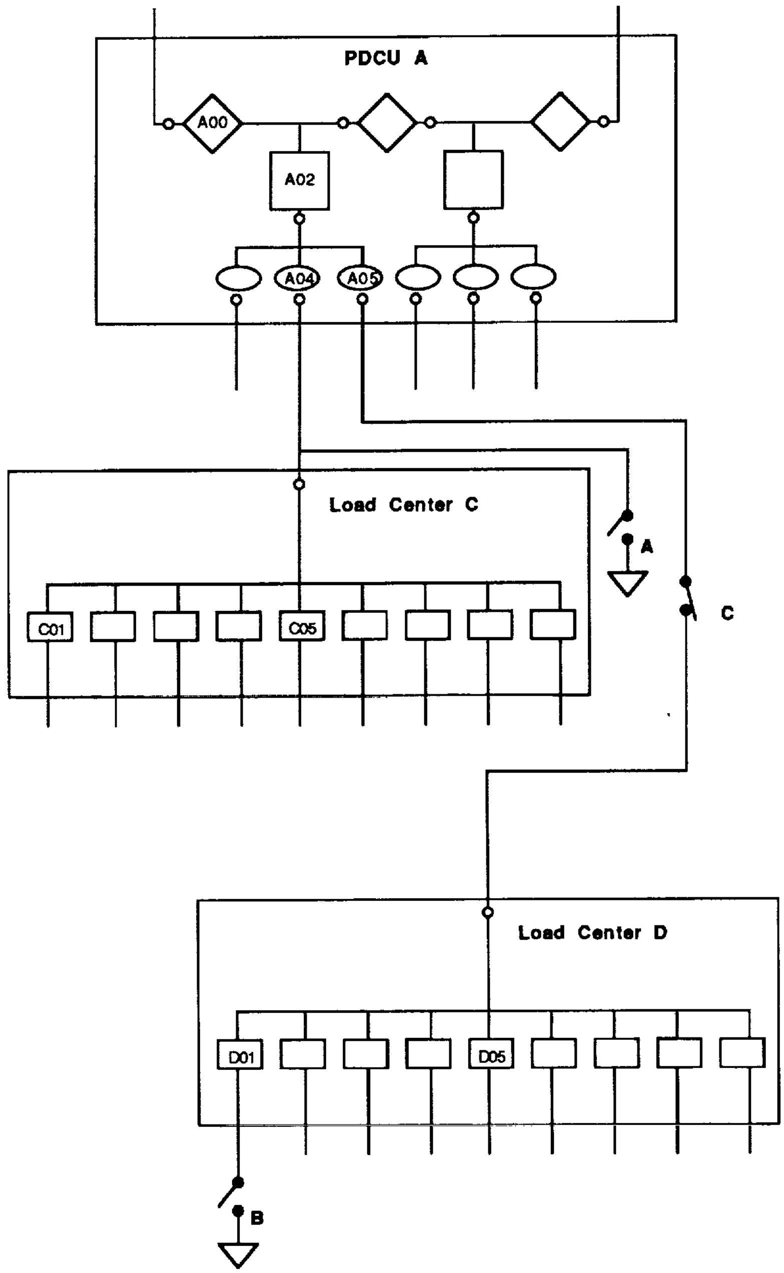


Figure 5.12.5.10.1-1 Test Plan Switch Configuration

		When the LLPs respond (also indicating that the operation was performed without any errors) FRAMES will command A04 to flip.
		When PDCU A closes A04, it will trip again.
		This data will be sent back to FRAMES.
		FRAMES will then diagnose the fault and send a message to FELES that A04 and all the switches below it are out of service.
3.	Wait approximately 5 minutes for Contingency to go into effect.	Contingency goes into effect.
4.	Inject a short at point B (Figure 5.12.5.10.1-1).	D01 will trip.
		Load Center D will send data to FRAMES indicating the situation.

FRAMES can be observed to receive the information and will send a message to Load Center D to open D01.

Load Center D will open D01 and report back that nothing unusual occurred in doing so.

FRAMES will then send a message to flip D01 (close and then open D01).

When Load Center D closes D01, D01 will trip again.

This will be reported back to FRAMES and FRAMES will diagnose the fault.

FRAMES will indicate the switch is out of service on the interface.

FRAMES will send a message to FELES indicating that D01 is no longer in service.

		FELES will have MAESTRO reschedule the tasks and FELES will send a contingency list down to FRAMES and the VME/10. Once the contingency list goes into effect, all unnecessary switches will be turned off. This change can be observed on the FRAMES interface.
5.	Wait for the schedule to finish.	All lights are turned off.
6.	Remove shorts from switches.	Shorts are removed.

Q______

5.12.5.11 Schedule 2

1.	Press <META> K on the Symbolics 3620D.	The current schedule is killed.
2.	Select "Re-initialize" on the Xerox 1186.	"Done initializing" message appears.
3.	Press <Return> on the Motorola VME/10.	"Do you wish to enter override?" prompt appears.

4.	Type "Y" on the Motorola VME/10.	"Breadboard is now in Manual Mode. Press Return." prompt appears.
5.	Press <Return> on the Motorola VME/10.	Manual Override Menu appears.
6.	Type "7" on the Motorola VME/10.	"Enter desired LLP using..."
7.	Type "A" on the Motorola VME/10.	"...press Return to continue"
8.	Type "7" on the Motorola VME/10.	"Enter desired LLP using..."
9.	Type "C" on the Motorola VME/10.	"...press Return to continue"
10.	Type "7" on the Motorola VME/10.	"Enter desired LLP using..."
11.	Type "D" on the Motorola VME/10.	"...press Return to continue"
12.	Press <Return> on the Motorola VME/10.	Manual Override Menu appears.
13.	Type "10" on the Motorola VME/10.	"Press the Break key to exit" prompt appears.

14.	Press <BREAK> on the Motorola VME/10.	"Enter user no.=" prompt appears.
15.	Type "200.MSFC" on the Motorola VME/10.	"=" prompt appears.
16.	Type "ALGO" on the Motorola VME/10.	"Do you wish to use real Xerox communications?" prompt appears on the VME/10.
17.	Type "Y" on the Motorola VME/10.	"Is CN10 present? prompt appears on the VME/10.
18.	Type "Y" to each of the LLPs that are applicable (10, 12 and 13) and type "N" to the LLPs that will not use (11, 14, 15, 20 and 21).	"Do you wish to go back and try again?" prompt appears on the VME/10.
19.	Type "N" on the Motorola VME/10.	"Do you wish to download to CN10?" prompt appears on the VME/10.
20.	Type "N" on the Motorola VME/10.	"Do you wish to go back and try again?" prompt appears on the VME/10.
21.	Type "N" on the Motorola VME/10.	"Do you wish to use the debug statements?" prompt appears on the VME/10.

22.	Type "Y" on the Motorola VME/10.	"Ensure there is a terminal..." prompt appears on the VME/10.
23.	Press <Return> on the Motorola VME/10.	The Wyse terminal scrolls with information. Menu for start of mission time appears on the Symbolics.
24.	Select "Reset" on the Symbolics 3620 D.	Select Length of Scheduling Period menu appears on the Symbolics.
25.	Select "8 Hour" on the Symbolics 3620 D.	Select Activities for Scheduling menu appears on the Symbolics.
26.	Select the following activities: Task 3 A Task 3 B Task 4 A Task 4 B Task 5 A Task 5 B Task 6 Task 7 Task 8 Task 9	The Schedule is ready.

27.	Click Left Mouse Button on "Highlighted" on the Symbolics 3620 D.	Activities are displayed on the Symbolics.
28.	Select "Schedule" on the Symbolics 3620 D.	The activities are scheduled.
29.	Press <META> S on the Symbolics.	Starts and updates control box (No visible action).
30.	Select a "one minute" delay to start the schedule on the Symbolics 3620 D.	C01, C05, D01, and D05 lights will turn on when the schedule starts.
31.	Inject a fault at Point C (open circuit) at mission time 3 (Figure 5.12.5.10.1-1).	D01 and D05 turn off (under voltage).
		D sends data to FRAMES.
		FRAMES waits for data from A, B & C.
		FRAMES diagnoses situation and sends information to FELES.
		MAESTRO processes contingency and sends contingency list down.

		A05 is turned off when contingency list takes effect. This will be observable on FRAMES interface.
32.	At mission time 20, click Left Mouse Button on "Screens" on the Symbolics 3620 D.	Select a Screen Menu appears.
33.	Select "Resource Manager" on the Symbolics 3620 D.	Resource Manager Window appears.
34.	Select "Immediate Power Change" on the Symbolics 3620 D.	"Power available to the module (watts)" prompt appears.
35.	Type "800" on the Symbolics 3620 D.	"How long effective" prompt appears.
36.	Type "0:0:10" on the Symbolics 3620 D.	Starts processing.
		Control will be in contingency state.
		On the Xerox, either C01 or C05 will go out.
		"Contingency Power System Fault" appears.

37.	Wait for contingency event list to take effect.	A consistent state of the world is present.
38.	Let schedule finish.	All lights are turned off.

Q______

Shutdown

5.12.5.12 Motorola VME/10

1.	Turn off power to Motorola VME/10 monitor.	Monitor is turned off.
2.	Turn off power to Motorola VME/10.	VME/10 is turned off.
3.	Turn off power to terminal connected to Motorola VME/10 port 2.	Terminal is turned off.
4.	Turn off power to the LLPs.	LLPs are turned off.
5.	Turn off power to 20kHz, 208V, nominal and housekeeping power.	Power is turned off.

Q______

5.12.5.13 Xerox 1186

1.	Select "Exit" with the Right Mouse Button.	Screens disappear.
2.	Type "(il:logout)"	Screen will blank out and Xerox can be turned off.

3.	Turn off Xerox 1186.	Xerox is turned off.

Q______

5.12.5.14 Symbolics 3620 D

1.	Selected "Screens"	Select a Screen Menu appears.
2.	Select "Console"	Console Menu appears.
3.	Select" Menu"	Select an Operation Menu appears.
4.	Select "Kill"	Everything is reset.
5.	Press <SELECT> L.	Lisp Listener appears.
6.	Type "logout"	User is logged out and Symbolics can be turned off.
7.	Type ":Halt Machine"	"Do you really want to halt the machine?" prompt appears on the Symbolics.
8.	Type "Yes"	The machine is halted.
9.	Turn off the Symbolics 3620 D.	The Symbolics is turned off.

Q______

[1] On December 14, 1988 a NASA Change Request specifying 120 V dc source power was put into effect.

6.0 TASK IV

Task IV for the SSM/PMAD was completed in October of 1985. The fundamental activity of Task IV was to study various host computer candidates and to make recommendations based on conclusions reached within the study.

Requirements were established as a result of the Task IV study. Also, trades in cost and performance were done. Recommendations were made based upon the specified and derived requirements and the results of the trades. The Motorola VME/10 was recommended as the host computer of choice and has since been purchased and successfully integrated into the SSM/PMAD as such.

Detailed results of Task IV are provided in the Task IV Study Report, included as Appendix III within this document.

7.0 SUMMARY

Martin Marietta's team for the SSM/PMAD, under contract to NASA/MSFC, has produced a working multi-agent knowledge based system. Its purpose is two-fold. First, it provides automation for scheduling and managing power enabling activities. Second, it functions to perform fault analysis and management for the test bed in near real-time conditions. These activities are brought about by the cooperative efforts of many independent software entities. These are:

1) Fault Recovery and Management Expert System (FRAMES)
2) Front End Load Enable Scheduler (FELES; peripherally including MAESTRO)
3) Load Priority List Management System (LPLMS)
4) Communications and Algorithmic Software (CAS)
5) Lowest Level Functions (LLFs).

Each high level software grouping, for example, LPLMS, exists within one or more hardware processing environments. These being:

1) FRAMES - Xerox 1186 and Lowest Level Processors (LLPs)
2) FELES & MAESTRO - Symbolics 3620 D
3) LPLMS - Symbolics 3620 D
4) CAS - Motorola VME/10 Communications and Algorithmic Controller (CAC)
5) LLFs - Motorola 107 Card Lowest Level Processors (LLPs).

The development effort has been on-going since 1985. The development process was organized around four major tasks. Task IV was completed first in 1985 and focused on selecting a host processing environment; the Motorola VME/10 was chosen. Task I was completed next in 1986 and provided an overall partitioning of the system by functions. Task II, which defined the roles of the various components in both the AI realm and the deterministic functionality, was completed in 1987. And, as a finish to the initial system activity, the Task III system was delivered to NASA/MSFC in 1988.

The completed development of the initial system does not mark the termination of analysis, development, and refinement within the SSM/PMAD system environment. Work is now in progress to improve the system from both the hardware and software perspectives. The present SSM/PMAD power automation system is a firm foundation upon which to base these continuing efforts. Important advantages in power automation have been gained .

The advantages which have been gained in the implemented design for the SSM/PMAD power automation system exist in the areas of modularity and integration.

The activities of Task I enabled the understanding of the needed relations between power hardware and the automated control of that hardware. Definitions of the controlling entities were made in Task I and functions were partitioned, providing a basis upon which to achieve system-wide modularization in a top-down decomposing flow.

The modularization of the breadboard provided the capability to partition functions into software entities which could be allocated to appropriate hardware processing environments. This allowed for the approach which was taken in Task II, defining what activities belonged to knowledge base functions and how those functions would in turn be allocated.

The implementation which was achieved in Task III provided a working set of AI and deterministic functions. These functions are supple representations of the actual hardware in the power automation breadboard, and of an expert's view of how the system should behave under the given breadboard conditions. The suppleness of the system is allowing for flexible growth of the knowledge and the associated knowledge processing functions. As well, the deterministic modularization makes it possible for integrated system growth within that process environment. In all, the SSM/PMAD power automation system is ready to achieve new evolutionary goals.

7.1 What's Next?

The SSM/PMAD possesses a rich and flexible architecture providing considerable opportunity for growth and enhancement. Both NASA and Martin Marietta have identified a set of tasks for enhancing the system. The enhancements under consideration should make the system more robust, increase the usability of it by crew members and/or other people, and provide for extendability and integration with other components and systems, etc. The following subsections outline a number of areas for improvement.

7.1.1 Knowledge Base Rule Grouping

The need for understanding interactions among multiple expert systems in a near real-time environment such as the SSM/PMAD is imperative. In order to attain the goal of this understanding, two tasks must be accomplished. First, the rules which are used within the total expert system environment must be organized as sets of complete and complementary entities. Otherwise, rules and their execution within one expert system may negate or deadlock rules or their effects as viewed from the knowledge bases of other expert systems (e.g., FRAMES may be depending upon decision information from a planning expert system which is in turn dependent upon FRAMES for an accurate system configuration status. Which rules belong where, and what is the individual rule content ?). The rules controlling the activities of the SSM/PMAD should be grouped into families which would be represented by their intentions and actions. The second task which must be accomplished is a completely uniform management of the multiple knowledge bases. This is discussed in the next paragraph.

7.1.2 Knowledge Base Management System

The current implementation of FRAMES is coded in LISP on a Xerox 1186 computer. The expert system rules are implemented in-line in the LISP language. This was done in order to achieve an initial measure of the comparative performance of FRAMES in its simplest form within the overall system. The knowledge processing within the system does not raise performance issues. However, the management of the knowledge within the overall system proves difficult under these circumstances.

Interactions among FRAMES, FELES, and LPLMS, as well as their impact on the user-interface, can only be understood by properly managing the knowledge and its varying inferences which occur as multi-variate functions. Uniform knowledge management combined with the proper rule grouping representations will provide a cohesive picture of these multiple interacting agents.

Currently, work defining and implementing a Knowledge Base Management System that will allow for easy modification of rules and diagnoses, is being performed. This will allow for the viewing of knowledge which currently exists in FRAMES and will extend it to incorporate more of the data gathering task in a meaningful manner as well. Finally, organizing the rules and their patterns of execution will provide the basis for further development in verifying and validating the various interacting agent knowledge bases for consistency and completeness.

7.1.3 Model Based Causal Reasoning

FRAMES currently only uses a model to keep track of the representation of the power system network. FRAMES should be enhanced to allow for the use of a model to reason over the possible faults and symptoms that can occur in the power system network. Motivation for this currently exists when diagnosing soft faults. Depending upon the configuration of RBIs in the power system network, the node equations used to test for soft faults are different. This is model based reasoning.

One of the drawbacks of model based reasoning is that computation is pushed to the real time situations where it is not wanted. This does not necessarily have to be the case though. Models can be set up at initialization so that compilations can then be done before actual operation of the system. Also, the depth of needed modeling is variable and can be combined into a hybrid approach using model based reasoning only where it is relevant. This would allow alternative topologies to be examined without major rewrites of software. This would also allow the system to be applied to other areas of power system automation.

7.1.4 Levels of Automation and User Interaction

Another area of enhancement is the addition of intermediate levels of automation. This allows the user to dynamically adjust schedules and have the adjustments be reflected at the scheduler. It also allows manipulation of switches via the FRAMES interface so that FRAMES knows what positions switches should be in in case of fault situations. The user-engaged automation level works in harmony with user-interface enhancements. The result is that when a user increases the amount of manual control on the SSM/PMAD, knowledge is added at the user-interface, avoiding the need for the user knowing how the automation expert systems accomplish their tasks. Therefore, the knowledge based activities take on an additional but related role. They must understand who is guiding the system execution, and the flow of knowledge must be regulated between the automation activities and the user-interface. And, at the user's demand, the expert systems must completely withdraw from execution within the system, leaving only a total manual mode.

7.1.5 Multiple Fault Diagnosis

Although multiple faults are not seen as very likely in the Space Station Freedom module power management and distribution system environment, this does not mean that they should not be taken into account. There was the recent example of a space shuttle launch in which a possible bug in the software was known to exist. The bug was supposed to be very unlikely to occur (e.g. .001 probability) in a launch situation, yet it did. This entailed a fix of the bug for proper operation. Therefore, experience establishes that although multiple faults are not very likely, it is still quite important to allow for them and to handle them properly.

7.1.6 Data Acquisition and Analysis

Enhancing the system in terms of better acquisition and analysis of data in a longer term fashion is also needed. There is some need to do this for incipient fault analysis. It would be nice to analyze data on a long term bases to characterize load performances, and the power system network performance in general. The current hardware implementation is not robust enough to handle increased levels of data analysis and acquisition.

7.1.7 User Interface

Currently, three separate user interfaces must be operated to run the system. These should be integrated into one common type of user interface. The computer upon which this resides will then control the initialization and operation of the other computers. This enhances usability, as well as understandability of the system.

7.1.8 Computer Hardware

To make progress on these enhancements requires more robust and extendable computer platforms. The system is currently using a Symbolics 3620 D, Xerox 1186, Motorola VME/10, and VME bus 68000 microprocessors for the LLPs. The computer hardware has added a lot of constraints on current operations that prohibit a number of the enhancements which should be made. Moving to a general purpose workstation, such as a SUN based platform, perhaps in conjunction with 80386 type processors for the LLPs to provide robustness, flexibility, and performance.

7.1.9 ADA

Additionally, it would be desirable for an SSM/PMAD system to fly on the Space Station Freedom. To do this, ADA is seen as a probable target software environment. Initial investigations into available ADA platforms and the the feasibility of moving to an ADA implementation for the SSM/PMAD are currently in progress.

7.1.10 Fault Injection

Extensive experience with the SSM/PMAD breadboard provides a strong basis for understanding the problems encountered in both managing it and in planning activities to be used in its analysis. From this experience it is seen that there is a strong need to develop a software fault injection capability. Fault injection, from a simulation approach, would provide an immediate means to exercise many of the breadboard components and capabilities, which otherwise may not be able to be accomplished without a full mock-up capability for the Space Station Freedom. Software fault injection requires a strong modeling capability within the overall breadboard architecture. Therefore, the model based causal reasoning capability will work hand in hand with fault injection, and this provides a needed first step towards knowledge base validation and verification.

7.1.11 Knowledge Base Validation and Verification

Knowledge base validation and verification is an important and critical activity which must be accomplished in order to get expert systems into space. NASA's strong concern and commitment to the validation of software in general and expert systems in particular is recognized. Much needs to be done in this area to further the investigation leading to specific implementations for verifying expert systems. The expert systems within SSM/PMAD should be verified as to consistency and completeness at a minimum.

7.1.12 Power System Simulation

In conjunction with a number of the above ideas, a simulation tool is needed for simulating the power system which is being modeled and automated. This would allow for exercising the automation and fault diagnosis software without having to rely on physical hardware being present and available. A simulation capability would provide a strong environment for a fault injection capability, and it would appear as a natural outgrowth of any model based causal reasoning capability which existed.

7.1.13 120 Volt DC Source Power

On December 14, 1988 a Change Request specifying 120 Volt dc source power went into effect. The SSM/PMAD system software needs to be modified to handle this new type power source.

8.0 REFERENCES

Adams, Thomas L., "Model-Based Reasoning for Automated Fault Diagnosis and Recovery Planning in Space Power Systems," American Chemical Society, 1985.

Britt, Daniel L., Gohring, John R., and Geoffroy, Amy L., "The Impact on the Utility Power System Concept on Spacecraft Activity Scheduling," Intersociety Energy Conversion Engineering Conference, 1988.

Davis, Randall, "Diagnostic Reasoning Based on Structure and Behavior," Artificial Intelligence 24(1-3):347-410, 1984.

DeKleer, Johan, "An Assumption Based TMS," Artificial Intelligence 28(2):127-162, 1986.

Drew, D.L. and Yoshimoto, G.M., "Space Station Coordinator, An Application of Intelligent Process Control," AIAA/NASA Symposium on Automation, Robotics and Advanced Computing for the National Space Program, 1985.

Ford, D., Weeks, D., "Intelligent Data Reduction: A Preliminary Investigation" in the 23rd IECEC Proceedings, Volume 3, Denver, CO 1988.

Freeman, Kenneth A., Walsh, Rick, and Weeks, David J., "Concurrent Development of Fault Management Hardware and Software in the SSM/PMAD," Intersociety Energy Conversion Engineering Conference, 1988.

Freeman, Kenneth A., and Pistole, Carl O., "A Concept for Standard Load Center Automation," Intersociety Energy Conversion Engineering Conference, 1987.

Genesereth, Michael R., "The Use of Design Descriptions in Automated Diagnosis," Artificial Intelligence 24(1-3):411-436, 1984.

Geoffroy, Amy L., Britt, Daniel L., Bailey, Ellen A., and Gohring, John, "Power and Resource Management Scheduling for Scientific Space Platform Applications," Intersociety Energy Conversion Engineering Conference, 1987.

Hanscher, Walter, and Davis, Randall, "Issues in Model Based Troubleshooting," MIT Industrial Liaison Program Report 11-34-87, 1987.

Iwasaki, Yumi, and Simon, Herbert A., "Causality in Device Behavior," Carnigie Mellon University Report CMu-CS-85-118, 1985.

Kirkwood, N., Weeks, D., "Diagnosing Battery Behavior with an Expert System in Prolog" in the 21st IECEC Proccedings, San Diego, CA 1986.

Lee, S.C., and Lollar, Louis F., "Development of a Component Centered Fault Monitoring and Diagnosis Knowledge Based System for Space Power System," Intersociety Energy Conversion Engineering Conference, 1988.

Lollar L., Weeks, D., "Going for the Long Term (Automating Spacecraft Power Systems)" in IEEE Potentials, Volume 7, Number 1, October 1987.

Miller, William D., and Jones, Ellen F., "Automated Power Management within a Space Station Module," Intersociety Energy Conversion Engineering Conference, 1988.

Miller, William D., and Jones, Ellen F., "Automated Space Power Distribution and Load Management," Intersociety Energy Conversion Engineering Conference, 1987.

Nguyen, T.A., Perkins, W.A., Laffey, T.J., and Pecora, D., "Checking an Expert System Knowledge Base for Consistency and Completeness," Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 375-378, 1985.

Reiter, R., "A Theory of Diagnosis from First Principles," Artificial Intelligence 32(1):57-96, 1987.

Riedesel, Joel, "Consistency and Completeness: An Exercise in Knowledge Base Validation," M.S. Thesis, University of Illinois at Urbana Champaign, 1988.

Stachowitz, Rolf A., and Combs, Jacqueline B., "Validation of Expert Systems," Proceedings of the Twentieth Annual Hawaiin Conference on System Sciences, 686-695, 1987.

Suwa, W., Scott, A.C., and Shortliffe, E.H., "An Approach to Verifying Completeness and Consistency in a Rule-Based Expert System," AI Magazine, 16-21, 1982.

Talukdar, Sarosh, and Cardozo, Eleri, "Patchwork Synthesis and Distributed Processing for Power System Diagnosis," Intersociety Energy Conversion Engineering Conference, 1987.

Walsh, R., Freeman, K., Weeks, D., "Concurrent Development of Fault Management Hardware and Software in the SSM/PMAD" in the 23rd IECEC Proceedings, Volume 3, Denver, CO 1988.

Walsh, Rick, "Applications for Power Control within a Space Station Module," Intersociety Energy Conversion Engineering Conference, 1987.

Weeks, D., "Autonomously Managed High Power Systems" in the 21st IECEC Proceedings, San Diego, CA 1988.

Weeks, D., "Artificial Intelligence Approaches in Space Power Systems Automation at Marshall Space Flight Center" in the Proceedings of The First International Conference on Industrial & Engineering Applications of Aritficial Intelligence & Expert Systems, Tullahoma, TN 1988.

Weeks, D., "Automation of the Space Station Core Module Power Management and Distribution System" in the Proceedings of SOAR '88, Dayton, OH 1988.

Weeks, David J., "Expert Systems in Space," IEEE Potentials, Volume 6, Number 2, Mar, 1987.

Weeks, D., "Space Power System Automation Approaches at the George C. Marshall Space Flight Center" in the 22nd IECEC Proceedings, Philadelphia, PA 1987.

Weeks, D., "Artificial Intelligence and Space Power Systems Automation" in the Proceedings of the Third Conference on Artificial Intelligence for Space Applications, Huntsville, AL 1987.

Weeks, D., "Expert Systems for MSFC Power Systems" in the Proceedings of the Conference on Artificial Intelligence for Space Applications, Huntsville, AL 1986.

Weeks, D., "Application of expert systems in the Common Module electrical power system" in SPIE Volume 580 Space Station Automation, Cambridge, MA 1985.

Weeks, D., Bechtel, R., "Autonomously Managed High Power Systems" in the 20th IECEC Proceedings, Miami, FL 1985.

Weeks, D., Kish, J., Doice, J., "Space Station Power System Automation Demonstration" in the Proceedings of the SPIE Space Station Automation IV Conference, Cambridge, MA 1988.

Wong, C., Weeks, D., et al "Cooperating Expert Systems for Space Station: Power/Thermal Subsystem Testbeds" in the 23rd IECEC Proceedings, Volume 3, Denver, CO 1988.

Wong, Carla M., Weeks, David J., Sundberg, Gale R., Healey, Kathleen L., and Dominick, Jeffry S., "Cooperating Expert Systems for Space Station: Power/Thermal Subsystem Testbeds," Intersociety Energy Conversion Engineering Conference, 1988.

9.0 GLOSSARY

Activity An activity defines a task consisting of a set of subtasks to be executed sequentially. Activities are scheduled by MAESTRO.

Anomaly An anomaly indicates an unexpected event. A switch tripping due to excess current is an anomaly.

Artificial Intelligence Assuming "intelligence" is defined: The faculty of thinking, reasoning, and acquiring and applying knowledge – as exhibited by people. Artificial Intelligence, then, is the mimicking of natural intelligence; furthermore, the artificial intelligence is exhibited by an artifact. For example, analyzing and implementing the knowledge of an expert to create an expert system describes the process of ascribing some artificial intelligence in the domain of the said expert to the expert system as implemented in some artifact (i.e. a computer).

Autonomy The condition or quality of self-operating.

Breadboard The hardware required to monitor, distribute, and control power flow to the loads. This hardware consists of SICs, GCs, A/Ds, RBIs, RCCBs, RPCs, and sensors.

Bus A nodal point of a power distribution network.

Causal Reasoning Reasoning from causes to effects. Causal reasoning attempts to describe how components of the domain are causally related to one another. Then, as these components are analyzed, causal reasoning can make use of the causal relationships to understand what has happened and predict what will happen.

Common LISP Object System The object oriented programming paradigm currently under consideration of the standards committee (made up of individuals from the Common LISP community) as the standard for object oriented programming in LISP.

Component In general, a component is some entity in the domain that is being modelled and reasoned about. For example, a switch may be a component.

Contingency A possibility that must be prepared against; future emergency.

Controller The functional module of the automation software residing controlling the scheduling and related processes. On the Symbolics, the Controller maintains a state transition network to determine what it should do in any of the defined events.

Current Voltage ÷ resistance.

Database A collection of data arranged for ease and speed of retrieval.

Equipment Various items of hardware that need power to operate. An individual item includes information about it indicating various modes of operation.

Event List A transaction consisting of a list of events. An event in the list indicates a switch that should be turned on or off, how much power is allotted to it, etc.

Exception Condition A condition that does not conform to normal expectations.

Expert System A program that mimics the knowledge of an expert such that the program is as expert as the expert is in the domain of the expert.

Fault A defect in a circuit or wiring caused by imperfect connections, poor insulation, grounding, or shorting.

Fault Isolation The act or process of isolating a fault given information of the symptoms of the fault.

Fault Management and Recovery The act or process of managing and recovering from new faults so that autonomy is maintained.

First Principles The basic axioms of a system or model. First principles are used in some reasoning programs for analyzing existing situations and predicting future situations.

Hard Fault A fault causing a switch to physically trip.

Incipient Fault A fault that is beginning to exist or appear; leading to a hard or short fault.

Knowledge Base The part of a knowledge base system that contains codified knowledge and heuristics used to solve problems.

Knowledge Base System A problem solving system that uses a knowledge base to reason about data in a database and the external world.

LLP/SIC ICD The interface control document defining the data formats and commands between the LLP and the SIC.

Load A device or the resistance of a device to which power is delivered.

Load Center The physical box at which a number of loads may be connected. A load center contains up to twenty-eight one kilowatt RPCs to which loads may be connected.

Load Priority List An ordered list of switches. This list is used for shedding loads (opening switches) in the event that available power is reduced.

Load Shed The act of opening a switch such that the load can no longer use power through the switch.

Lowest Level Processor The computer that is responsible for commanding switches and collecting data from switches. Each LLP is a self contained processing unit responsible for either a load center, power distribution control unit, or subsystem distributor.

Masked Fault A fault that cannot be observed except by the presence of another fault. For example, a switch having a broken current sensor won't trip on overcurrent, while the switch above will.

Model A description of a system or theory that accounts for all of its known properties (or the properties that are important to the model builder).

Model Based Reasoning A method by which reasoning uses a model for drawing conclusions about the domain being studied.

Motorola VME/10 The computer on which the CAC resides. Controls and processes communications between the LLPs and FRAMES.

Multiple Fault A situation where more than one dependent or independent faults occur within delta time of one another. Delta time used here is the time it takes to recover from a single fault.

Parent Switch The switch hierarchically connected immediately above another switch.

Portable Common Loops The most common implementation of CLOS. This implementation was originally created by Xerox PARC and is implemented for a large number of computers. Portable Common Loops has no relationship to Xerox LOOPS.

Power Electrical energy dissipated within circuits or components.

Power Factor Ratio of average power to apparent power.

Power Distribution Control Unit The physical box which controls the distribution of power to load centers.

Power Hardware All the components within the breadboard which monitor or control the flow of electrical current.

Power System Network The topology of switches and cables making up a network from source to loads.

Resource An object used by a piece of equipment. For example, crew time is a resource, a switch is a resource.

Resource Scheduling The process of scheduling a set of activities to make the most efficient use of available resources.

Schedule The object that indicates when what activities are to be executed. The schedule includes descriptions of what resources the activities need as well.

Sensor A device which responds to a voltage, current, or temperature and provides an analog measurement used in monitoring the breadboard.

Sibling Switches Those switches that immediately below the parent switch of the switch except for the switch itself. Exactly analogous to the siblings of a person in a family.

Single Fault A hard, soft, or incipient fault which occurs as a singleton.

Soft Fault An illegal use of current. Hard faults may also have an illegal use of current; soft faults may be distinguished in that they don't necessarily cause a hard fault.

Source Power Reduction The act of specifying a reduced amount of power available from the source.

Subsystem Distributor A load center containing three-kilowatt RPCs instead of one-kilowatt RPCs, capable of distributing power to lower level distributors.

Switch A device which permits current to flow when closed. RBIs, RCCBs, and RPCs are all switches.

Switch Manipulation The act or process of opening and closing switches by the diagnostic routines of FRAMES for the purpose of isolating a fault.

Switchgear A general term referring to all breadboard power hardware involved with switching activities.

Switchgear Interface Controller The element of power hardware which communicates with LLPs, switches, and sensors.

Symbolics 3620D A LISP machine on which MAESTRO, FELES, and LPLMS resides.

Symbolics 3640 A LISP machine on which MAESTRO, FELES, and LPLMS resides.

Symptom A tripped switch or current reading, etc. providing an element of information about a fault.

Symptom Set A set of symptoms indicating a fault. A fault directly maps to a set of symptoms. The object is to take a set of symptoms (possibly indicating many different faults) and determine which fault may have produced the symptom set.

Transaction A message that is transmitted anywhere between the automation software.

Voltage Electric potential or potential difference expressed in volts.

VME/10 A computer workstation on which the CAC resides. Also known as the CAC.

Xerox 1186 A LISP machine on which FRAMES resides.

APPENDIX I – TASK I DATA

MCR-86-583
Contract No. NAS8-36433

Task I
Study
Report

July 1986

SPACE STATION AUTOMATION OF COMMON MODULE POWER MANAGEMENT AND DISTRIBUTION

W. D. Miller
D. Pottruff
J. Tietz
D. Britt

MARTIN MARIETTA
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201

FOREWORD

This report was prepared by Martin Marietta Denver Aerospace, for the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, in response to Contract NAS8-36433, and is submitted as the Task I Summary Report, as specified in the contract data requirements list.

CONTENTS

Page

INTRODUCTION 1

1.0 TASK-I STUDY SUMMARIES 2
1.1 Government-Furnished CM/PMAD Design Review 2
1.1.1 General Review and Evaluation of GFP Designs 8
1.1.2 General Review and Evaluation of GFP Designs 10
1.2 CM/PMAD Function Partitioning 12
1.2.1 Partitioning Approach 12
1.2.2 Definition of Controlling Entities 12
1.2.3 Partitioning Rules 13
1.2.4 Functional Decomposition of the CM/PMAD Task 14
1.2.5 Estimate Relevant to Software Development of Function 3.0 . 17
1.2.6 Results of Partitioning 17
1.3 Expert Systems Role Evaluation 34
1.3.1 ES/KBS Applicability Criteria 35
1.3.2 Applications within Power Network Control 36
1.3.3 System Involving Power Control or Maintenance 38
1.3.4 Application as Onboard Experiments 40
1.3.5 Recommendations 41
1.4 Automation Architecture Issues 42
1.4.1 Distributed vs Centralized Automation Analysis 42
1.4.2 Fault Isolation Analysis 48
1.4.3 Load Management Analysis 55
1.4.4 Space Station Power and Data Bus Interface Analysis 61
1.4.5 Data and Power Ground Separation Analysis 63
1.4.6 Local Energy Storage Analysis 64
1.4.7 Crew Interface Analysis 66
1.4.8 Autonomy Sensing Techniques and Requirements Analysis . . . 78
1.5 Fiber Optics Evaluation 81
1.6 Data Bus and Microprocessor Selection 82
1.6.1 Maximum Data Rates 82
1.6.2 Data Bus Selection 88
1.6.3 Microprocessor Evaluation 89
1.7 Computer Language Selection 92
1.7.1 Conventional Software 92
1.7.2 Artificial Intelligence (AI) Software 93
1.8 Automation Approach/Architecture Definition 101
1.9 Time Shared Use of Space Station Bus vs CM/PMAD Dedicated Bus 103
1.10 Work Package-04 Data Exchange Identification 104
1.10.1 Data from WP-04 to CM/PMAD 104
1.10.2 Data from CM/PMAD to Work Package-04 105

2.0 SUMMARY 107

3.0 REFERENCES 109

A APPENDIX A - FUNCTIONAL BREAKDOWN OF THE CM/PMAD TASK . . . A-1

B APPENDIX B - CREW INTERFACES RELEVANT TO FUNCTIONS OF THE CM/PMAD TASK B-1

CONTENTS - cont'd

Figure Page

1.1-1 Power Distribution Network Schematic (1) 3
1.1-2 Power Distribution Network Schematic (2) 4
1.1-3 Power Distribution Network Schematic (3) 5
1.1-4 Power Distribution Network Schematic (4) 6
1.1-5 Power Distribution Network Schematic (5) 7
1.1.1-1 Number of Measurements per Design 11
1.1.1-2 Cost Comparisons 11
1.2.4-1 Power Management and Distribution Top-Level Functional Breakdown 15
1.2.4-2 Functional Breakdown of CM/PMAD Power Network Control 16
1.6.1-1 Maximum Bus Rates for Fully Loaded Module as Baseline . . . 84
1.6.1-2 Maximum Bus Rates for Common Module as Baseline 85
1.6.1-3 Maximum Bus Rates for Hierarchically Arranged Subsystem . . 86
1.8.1 CM/PMAD Automation Architecture 102

Table Page

1.1-1 Five Government-Furnished Designs Summary 2
1.1.1-1 Automation Software Relative Sizing 9
1.2.3-1 Rule for function Partitioning of CM/PMAD 18
1.2.5-1 Preliminary Estimates, Assuming S/W Control of Function 3.0 20
1.2.6-1 Partitioning of Functions 1.0 and 2.0 26
1.2.6-2 Partitioning of Function 3.0 27
1.4.1-1 APSM Factors in Selecting Distributed vs Central Computer Architecture 45
1.4.6-1 Local Energy Storage Functional Impact Analysis 65
1.7.2-1 Adjustments to Weighting Factors for CM/PMAD Evaluation Criteria 97
1.7.2-2 Evaluation Matrix for CM/PMAD Language for Pre-implementation Experiments 99
1.7.2-3 Evaluation Matrix for CM/PMAD Language for AI Flight Software 100

GLOSSARY

ac	Alternating Current
AI	Artificial Intelligence
AMPS	Autonomously Managed Power System
APSM	Automated Power Systems Management
AgZn	Silver-Zinc
BIU	Bus Interface Unit
BLES	Baseline Load Enable Schedule
CM	Common Module
CM/PMAD	Common Module/Power Management and Distribution
dc	Direct Current
EMC	Electromagnetic Compatibility
EMES	Energy Management Expert System
ES/KBS	Expert Systems/Knowledge-Based Systems
FDDI	Fiber-Distributed Data Interface
FOC	Final Operating Configuration
HADS	Hierarchically Arranged Distributed Subsystem
IOC	Initial Operational Configuration
IR&D	Independent Research and Development
LAN	Local Area Network
LESD	Load Enable Schedule Display
$Li/SOCl_2$	Lithium-Thionyl Chloride
MUX/DEMUX	Multiplexer/Demultiplexer
NiCd	Nickel-Cadmium
NiH_2	Nickel-Hydrogen
ORU	On-orbit Replaceable Unit
PCU	Power Control Unit
PLES	Preliminary Load Enable Schedule
RBI	Remote Bus Isolators
RCCB	Remote-Controlled Circuit Breakers
RPC	Remote Power Controllers
UPS	Uninterruptible Power Supply

INTRODUCTION

This document is in response to the Summary Report requirements of Task I of the Statement of Work for Automation of the Common Module Power Management and Distribution (CM/PMAD). Task I, CM/PMAD System Automation Plan Definition, includes the review, with respect to automation, of Government provided candidate network topologies. In addition, Task I includes CM/PMAD functions definition, function partitioning and evaluation of any expert systems role in those functions. Task I also includes study of the issues involved in CM/PMAD automation as well as investigations of hardware and software approaches. Additionally, efforts include requirements definition for CM/PMAD data exchange with other elements of Space Station. Overall, Task I efforts, summarized in this report, provide a data base of information for the selection of an automation approach as well as definition of an automation approach for the Space Station CM/PMAD.

1.0 TASK 1--STUDY SUMMARIES

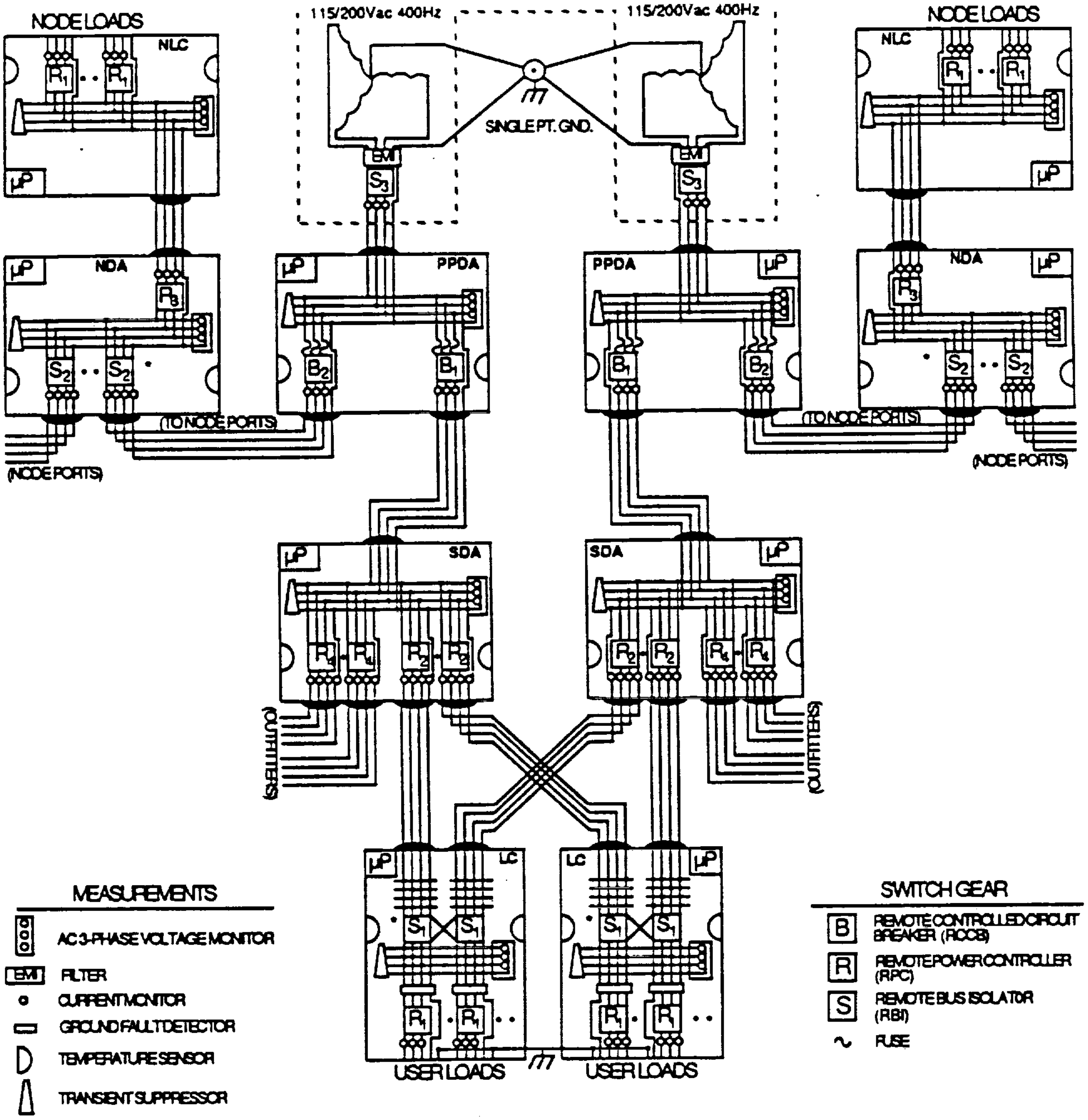
1.1 GOVERNMENT-FURNISHED CM/PMAD DESIGN REVIEW

Government-furnished conceptual network designs for the common module electrical power management and distribution system were to be reviewed and evaluated with respect to the subtasks of Task 1. Five designs (Fig. 1.1-1 through 1.1-5) were furnished that identified candidate distribution networks; each defining power types, converters, switches, and circuit breakers. The five designs (Table 1.1-1) provided a matrix of candidates involving two input power types and various distribution schemes. All furnished designs use remote power controllers (RPC), remote-controlled circuit breakers (RCCB), and remote bus isolators (RBI). Additionally, designs two through five incorporate intramodule bulk power conversion for distribution as shown in Table 1.1-1.

Table 1.1-1 Five Government-Furnished Designs Summary

Design	Input Power Type	Distributed Power Type			
		Nodes		Outfitters	Users
		Loads	Ports		
1	115/200 Vac, 400 Hz, 3 Phase	Input	Input	Input	Input
2	115/200 Vac, 400 Hz, 3 Phase	Input	Input	150 Vdc	150 Vdc
3	115/200 Vac, 400 Hz, 3 Phase	Input & 150 Vdc	Input & 150 Vdc	150 Vdc	150 Vdc
4	115/200 Vac, 400 Hz, 3 Phase	Input	Input	Input & 150 Vdc	Input & 150 Vdc
5	150 Vdc	Input	Input	Input & 115/200 Vac, 400 Hz, 3 Phase	Input & 115/200 Vac, 400 Hz, 3 Phase

Figure 1.1-1
POWER DISTRIBUTION NETWORK SCHEMATIC (1)

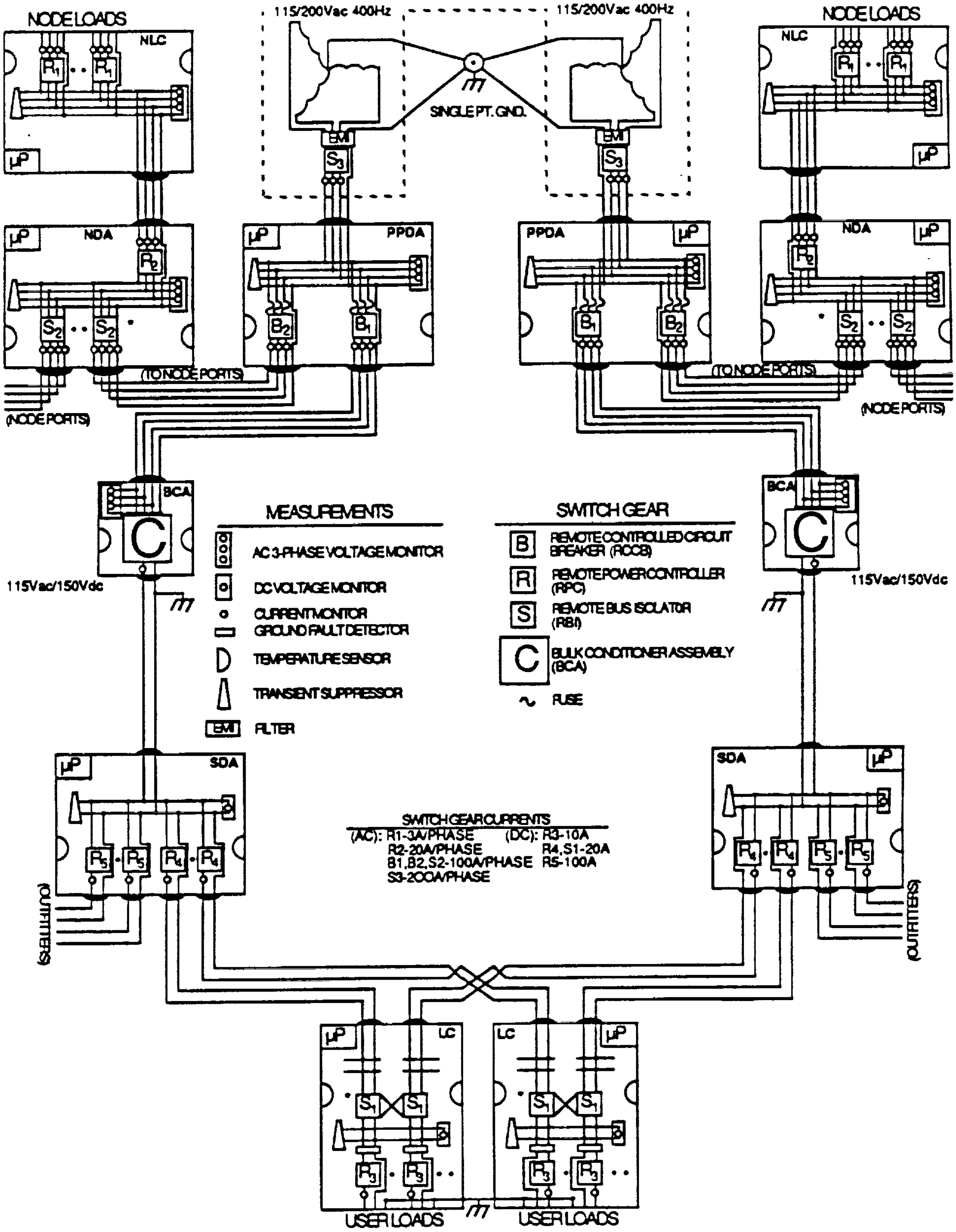


SWITCHGEAR CURRENTS

(AC): R1-3A/PHASE
S1,R2-10A/PHASE
R3-20A/PHASE
R4-50A/PHASE
B1,B2,S2-100A/PHASE
S3-200A/PHASE

¹LOCKOUT SCHEME MUST BE USED TO ENSURE ONLY ONE SWITCH ACTIVE AT ANY GIVEN TIME TO PREVENT SHORTING OF BUSSES

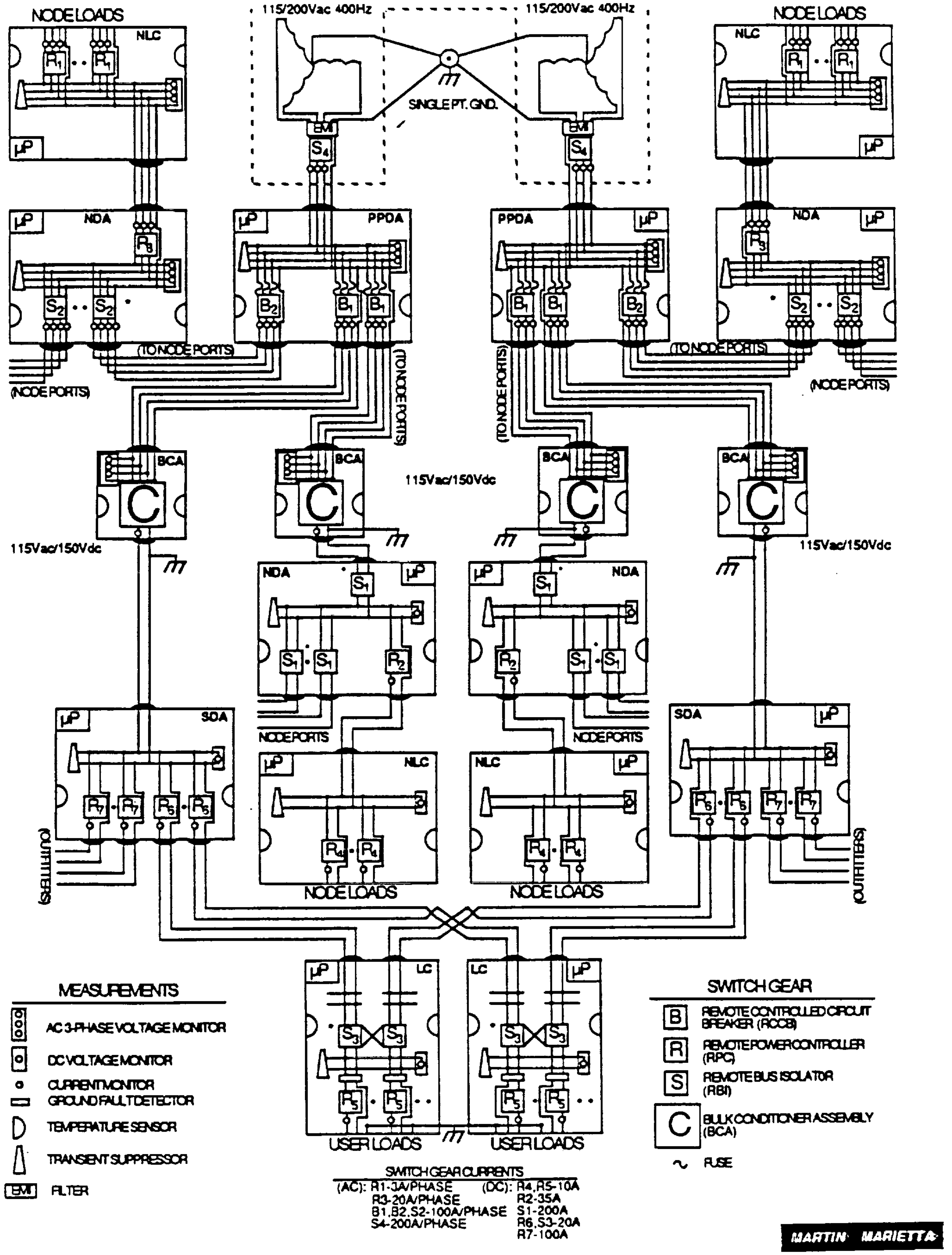
Figure 1.1-2
POWER DISTRIBUTION NETWORK SCHEMATIC (2)



[1] LOCKOUT SCHEME MUST BE USED TO ENSURE ONLY ONE SWITCH ACTIVE AT ANY GIVEN TIME TO PREVENT SHORTING OF BUSSES

MARTIN MARIETTA

Figure 1.1-3
POWER DISTRIBUTION NETWORK SCHEMATIC (3)



SWITCH GEAR CURRENTS

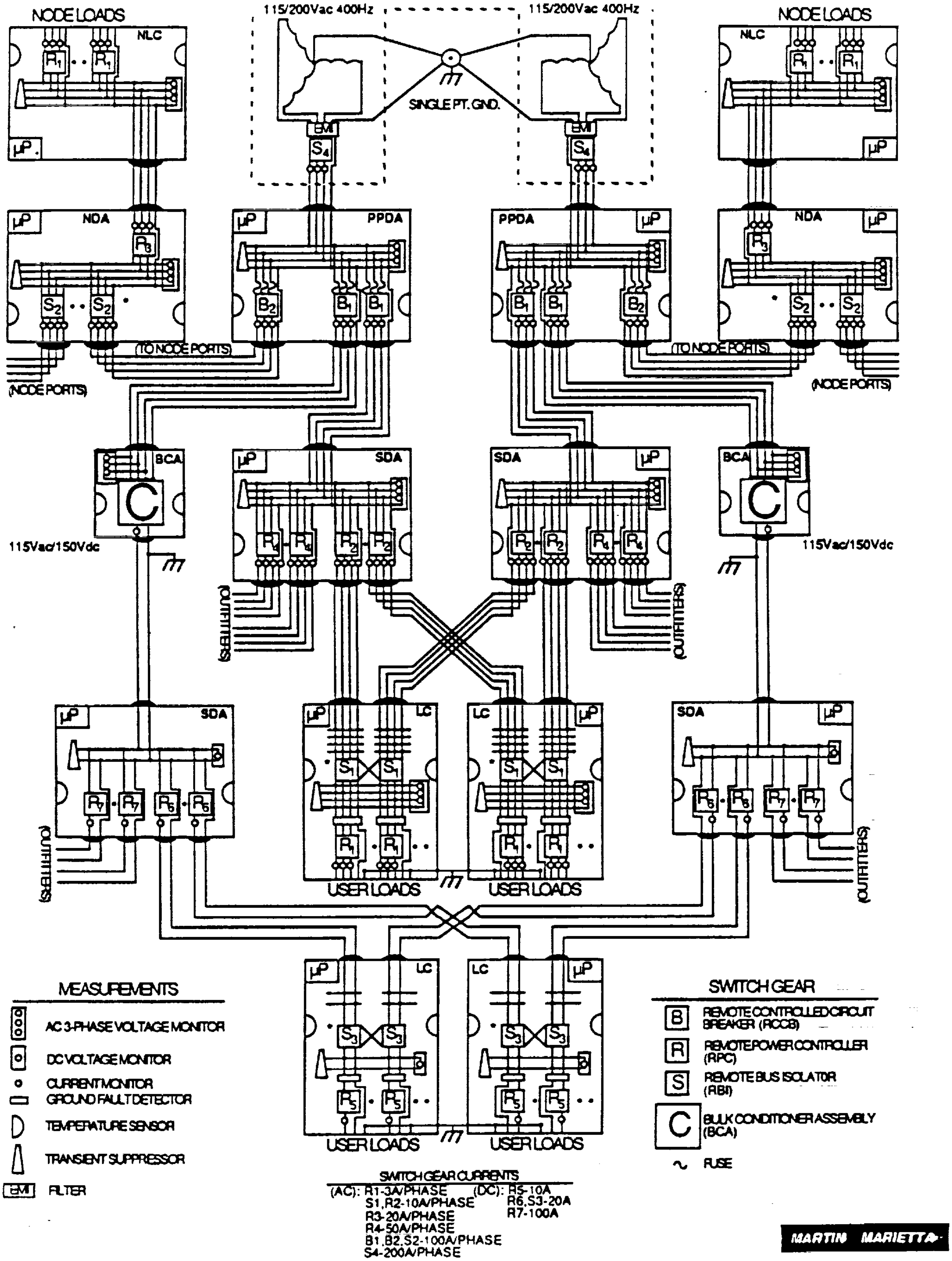
(AC):	(DC):
R1-3A/PHASE	R4,R5-10A
R3-20A/PHASE	R2-35A
B1,B2,S2-100A/PHASE	S1-200A
S4-200A/PHASE	R6,S3-20A
	R7-100A

MARTIN MARIETTA

*LOCKOUT SCHEME MUST BE USED TO ENSURE ONLY ONE SWITCH ACTIVE AT ANY GIVEN TIME TO PREVENT SHORTING OF BUSSES

Figure 1.1-4

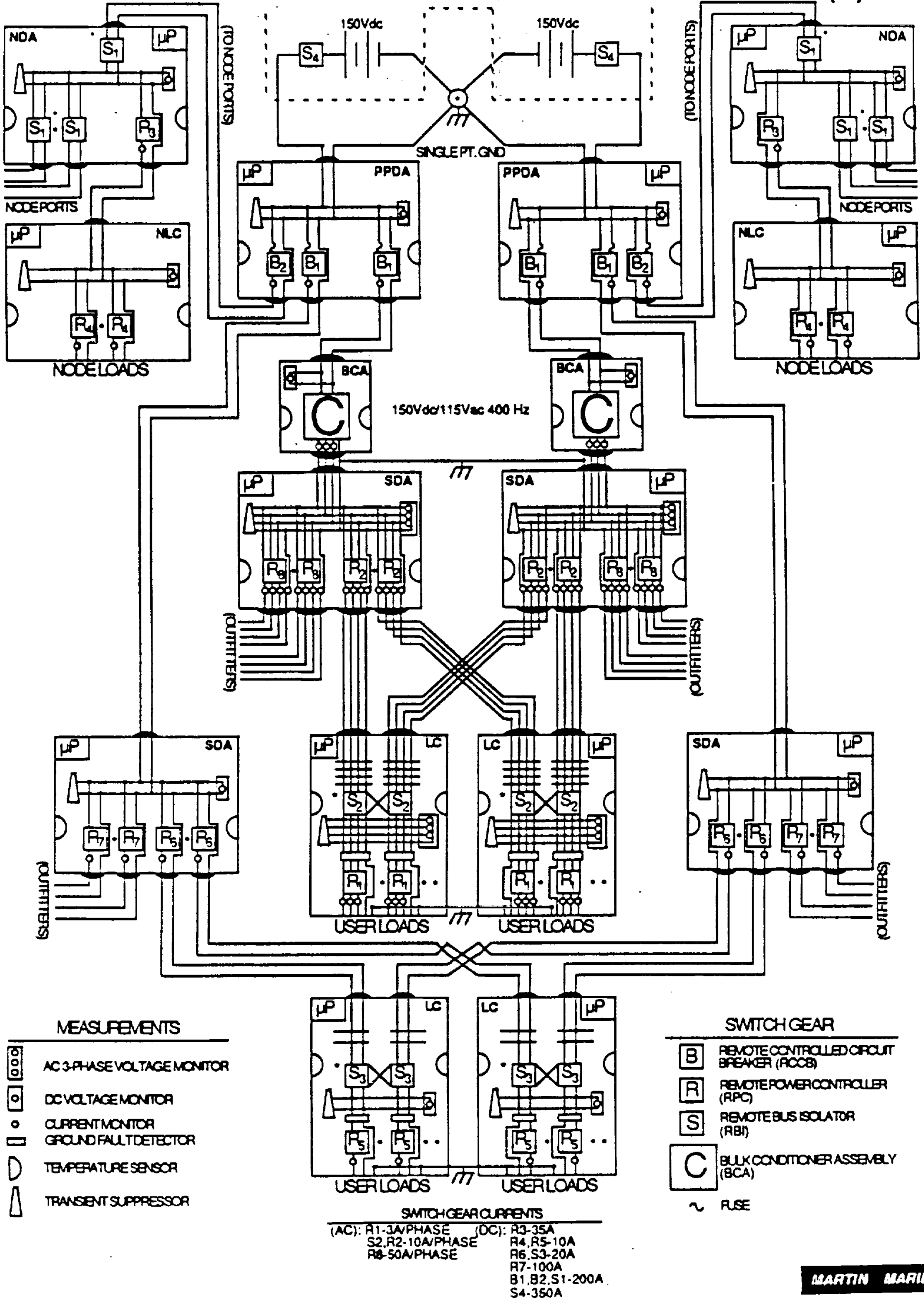
POWER DISTRIBUTION NETWORK SCHEMATIC (4)



¹LOCKOUT SCHEME MUST BE USED TO ENSURE ONLY ONE SWITCH ACTIVE AT ANY GIVEN TIME TO PREVENT SHORTING OF BUSSES

Figure 1.1-5

POWER DISTRIBUTION NETWORK SCHEMATIC (5)



MARTIN MARIETTA

*LOCKOUT SCHEME MUST BE USED TO ENSURE ONLY ONE SWITCH ACTIVE AT ANY GIVEN TIME TO PREVENT SHORTING OF BUSSES

1.1.1 General Review and Evaluation of GFP Designs

None of the candidate designs use direct current (dc) only for both input and distribution power type. An all dc system greatly reduces the number of sensors, effectors and software complexity; and therefore, reduces risk and cost with respect to automation software development, test, and integration. However, it must also be noted that there are hardware considerations to be studied, such as the electrical isolation issue, which becomes more difficult to resolve in an all dc system. In addition, load requirements and total power-type use must also be a major consideration when selecting distribution power types. The hardware trade studies involved with respect to an all dc system are not repeated or evaluated in this task and may offset the automation software advantages of a dc system supposed and investigated here.

The total number of sensors, effectors, and implemented function complexity are not the only considerations in sizing an automation task; but they are considered significant keys in any automatic control and data acquisition system. In this sense, the automation task of these designs is typical. Using this as a comparison tool in this case is valid because as the total number of sensors and effectors increase, the complexity of functions implemented also increases.

The number of sensors, and therefore, data requiring handling, are plotted versus furnished design for various total numbers of loads in Figure 1.1.1-1. The parameters include an all dc system approach for its comparison. Groundrules in the baseline number of loads include assuming loads and power availability points as follows:

1) Node Loads--12 per node load center;

2) Node Ports--Five per node distribution assembly;

3) Outfitters--37 per secondary distribution assembly;

4) User Loads--30 per load center.

Additionally, where two power types are available to a class of electrical loads, it is assumed that half of each load class is allocated one of the power types. Two electrical power types available to a single load point are assumed not to exist. Finally, all other measurements, such as temperature, are considered the same for each design, and therefore, are ignored.

The expected results show the significant differences in total measurements in approaches, where there are 192 total measurements for an all dc approach and 668 for an all three phase distribution approach. Converting the total number of measurements into absolute automation software effect estimates is only debatably possible at the present stage of software definition. However, insight-lending comparisons are made by estimating the automation software for an all dc system from 20,000 to 40,000 lines of high-level code. This estimate comes from comparison of a similar automatic control and data acquisition system* and from a review of the functions to be implemented, developed, and discussed in Appendix A.

Using the baseline estimate of software code and factors of both 25% and 50% increase per 100% increase in total number of measurements allows conversion from measurements to software sizing. The results in the relative software sizing are presented in Table 1.1.1-1.

Table 1.1.1-1 Automation Software Relative Sizing

Design	Relative Software Sizing Range (Lines of Code: Minimum - Maximum)	
	25% Factor	50% Factor
dc	20,000 - 40,000	20,000 - 40,000
1	37,000 - 75,000	55,000 - 110,000
2	28,000 - 56,000	36,000 - 73,000
3	28,000 - 57,000	37,000 - 73,000
4	33,000 - 67,000	47,000 - 94,000
5	30,000 - 61,000	41,000 - 81,000

*Martin Marietta's Battery Development and Test Facility, developed in 1982-1984, comprises a network of three HP 1000 computer systems coupled with data acquisition and control hardware. The entire system contains approximately 2000 points of measurement and 1600 effectors. It is a real-time battery test facility, containing approximately 75,000 lines of code (excepting the operating system), implementing many functions similar to a CM/PMAD.

Using $400 per line of code as a comparison in dollars and using the midpoints of Table 1.1.1-1 entries yields the estimates shown in Figure 1.1.1-2. Clearly, the software advantages can be seen as the use of dc distribution is increased. The estimates given are not intended to be used in an absolute sense, but for comparisons. They are, however, also intended as a basis for deciding what, if any, significant effect automation software has on power-type selections. An estimated savings of $2.5 million is possible in automation software development by selecting an all dc distribution network scheme. The second best approach, with respect to automation software development cost and risk, is to use two wire distribution when possible while still meeting system requirements.

The total number of measurements and parameters is not the only effect on power management and distribution software. An alternating current (ac) system requires more complex software data conditioning with respect to peak voltages, root-mean-square voltages, frequency accounting, and power factors. In addition, any power conversion also requires control, redundancy assessment, and fault management. Additionally, in the case of three-phase systems, phase angles and load balancing also are factors in an increase of overall automated management hardware and software complexity.

1.1.2 Subtask Review and Evaluation of GFP Designs

In addition to the above review of the general approaches, review and evaluation of the automation of the candidate networks are presented in each of the applicable subtasks of Task 1. The differences in candidate designs, when reviewed in Subtask 1.2 Function Partitioning, result in including power conversion management and table maintenance in the top-level functional decomposition for designs two through five. Subtasks 1.4 Automation Architecture Issues, and 1.8 Automation Approach/Architecture Definition, are also affected directly by the differences in these distribution approaches. These effects are presented in the respective sections.

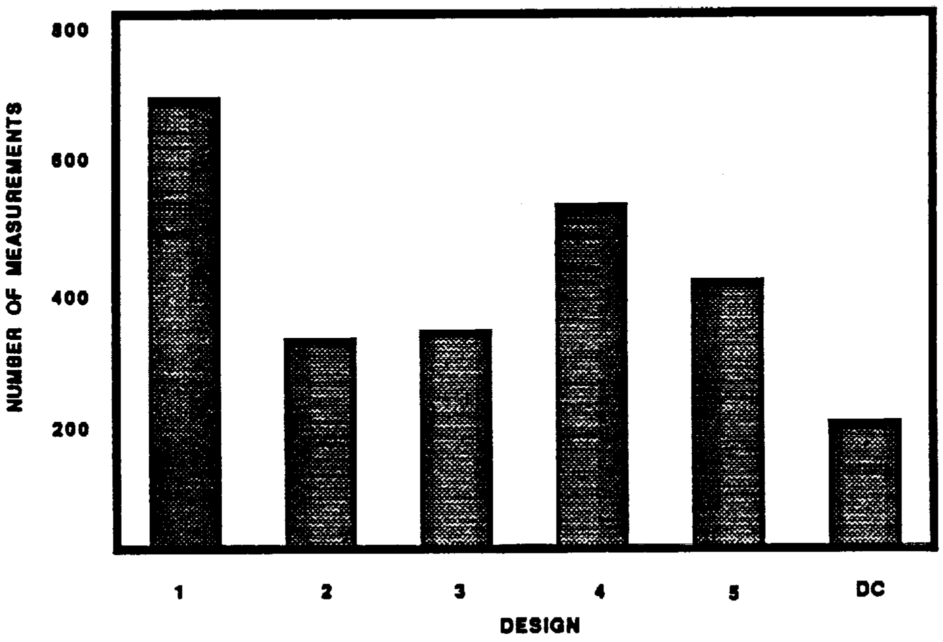


FIGURE 1.1.1-1 NUMBER OF MEASUREMENTS PER DESIGN

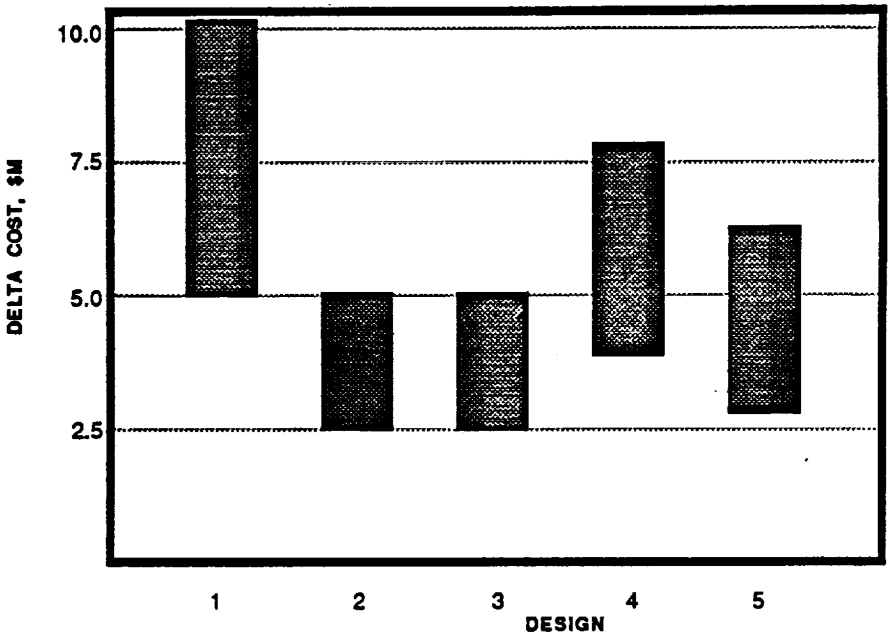


FIGURE 1.1.1-2 COST COMPARISONS

1.2 CM/PMAD FUNCTION PARTITIONING

Electrical power is an indispensable resource in a space vehicle with a human crew. If it failed, few devices on the vehicle would operate, and the safety of the crew would soon be at risk. Thus, management and distribution of common module electrical power is a vital task; yet, ironically, it is an inherently tedious one. Because the task is tedious, it would seem logical to automate as much of it as possible. However, because the task is vital, it would seem unwise to delegate all of it to machinery. Obviously, humans and machinery must manage and distribute electrical power in the common module. The questions are (1) which functions of the task should humans perform; (2) which functions should machinery perform; and (3) what kinds of machinery are to be used. In answering those questions, we partition the task of common module power management and distribution (CM/PMAD); i.e., we assign to each function of the task a specific type of controlling entity. The partitioning done in this section applies to the CM/PMAD task as it will exist in the IOC.

1.2.1 Partitioning Approach

The approach used to partition the CM/PMAD task is divided into four major steps:

1) List and define potentially useful types of controlling entities for functions in the CM/PMAD task;

2) Develop rules and guidelines to partition functions;

3) Define the task of CM/PMAD to a sufficient level of functional detail to allow partitioning of a single controlling entity to each function;

4) Partition the controlling entities and definitions of step one to each of the functions in the functional decomposition of step three, using the rules and guidelines of step two.

1.2.2 Definitions of Controlling Entities

1) Hardware Partition--In this report, any function partitioned to hardware is to be controlled entirely by hardware. The hardware may be settable (for example, the remote power controllers may have settable trip levels), but not programmable in the usual sense.

2) Algorithmic Software Partition--Algorithmic software is also called conventional software. Any function of the CM/PMAD task that is partitioned to algorithmic software is to be controlled by algorithmic software plus sufficient hardware to allow the software to be effective (e.g., software needs memory to reside in, a microprocessor to be executed in, etc).

3) Expert System Partition--Any function of the CM/PMAD task that is partitioned to an expert system is to be controlled by expert software (sometimes called artificial intelligence), plus sufficient hardware to allow the software to be effective. Expert software incorporates analogs of the knowledge and experience of one or more human experts and is designed to mimic their intelligent approach to solving complex problems.

4) Crew Partition--Any function of the CM/PMAD task that is partitioned to the crew is to be controlled by one or more persons on the space station or on the ground plus sufficient hardware and software to allow their actions to be effective. These persons are assumed to be technically trained, but not necessarily expert in the function to be controlled.

5) Expert-Aided Crew Partition--This partition has the same definition as crew partition, above, except that the crew person(s) will be aided in their controlling function by an expert. The expert may be an expert system (above), or a person who is not part of the day-to-day crew.

1.2.3 Partitioning Rules

The second step in partitioning CM/PMAD is to list rules for function partitioning. The rules of this step are in Table 1.2.3-1. As each function of the CM/PMAD task is identified, these rules are consulted to determine which of the controlling entities is the most practical one to perform that function. Function types, as referenced in the rules, are coarsely categorized as follows:

1) Simple -- Functions/processes which are well understood (i.e., "common knowledge"), usually involving simple mathematics/logic, with predictable inputs and outputs.

2) Complex -- Functions which are technically understood using knowledge available from accepted text books or procedures, but which involve advanced scientific skills or special training to implement.

3) Expert -- Functions which are usually understood only by recognized experts, and require the knowledge and judgement of an expert to fulfill defined requirements.

1.2.4 Functional Decomposition of the CM/PMAD Task

The third step in partitioning is to define the task of CM/PMAD to a sufficient level of detail to allow partitioning of the functions involved. A useful way of doing this is to map a functional breakdown of the task. Figure 1.2.4-1 shows the upper levels of the current functional breakdown of CM/PMAD. The major function labeled "power network control" is broken down to a finer level of functional detail in Figure 1.2.4-2. The two figures constitute the functional breakdown of CM/PMAD to the level of detail necessary to complete partitioning.

The detailed description of the functional breakdown of the CM/PMAD task is in Appendix A. The description refers to Figure 1.2.4-1 and follows it from left to right until it gets to the major function labeled "power network control"; then the description refers to Figure 1.2.4-2 and follows it from left to right. The description defines, for convenience, the term "CM computer" to mean any part of or all of the distributed information processing in the common module (CM), whether S/W (including expert systems) or logic H/W. Similarly, the term "space station main computer" is taken to mean any part of or all of the distributed information processing in the space station outside of the modules.

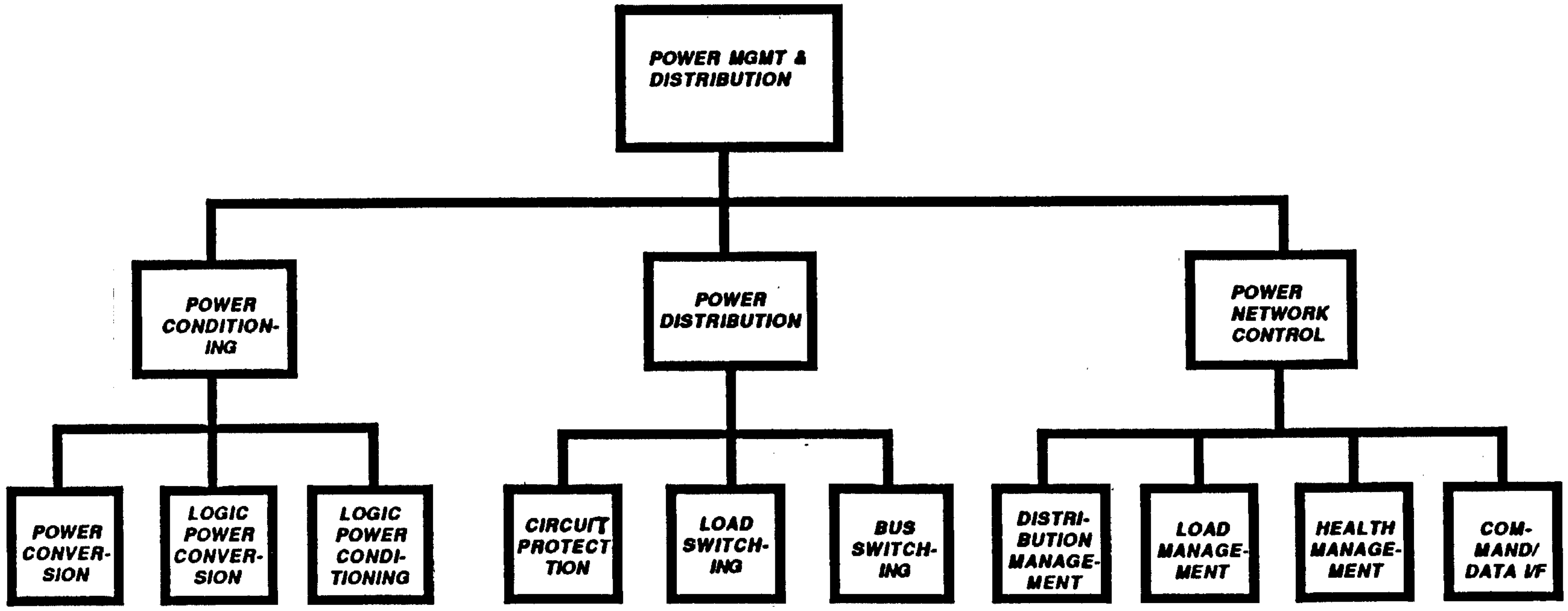
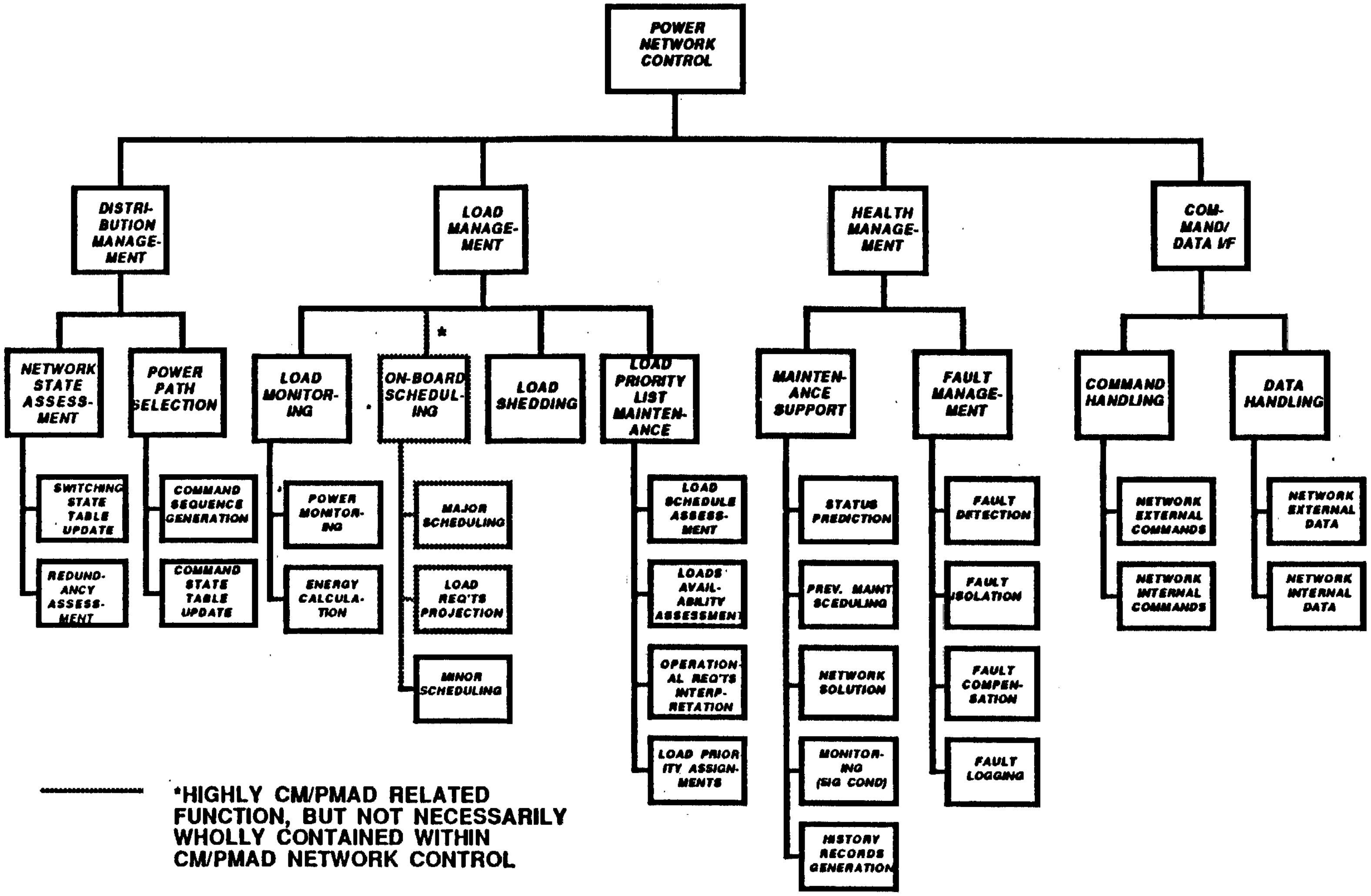


FIGURE 1.2.4-1 POWER MANAGEMENT AND DISTRIBUTION TOP LEVEL FUNCTIONAL BREAKDOWN



*HIGHLY CM/PMAD RELATED FUNCTION, BUT NOT NECESSARILY WHOLLY CONTAINED WITHIN CM/PMAD NETWORK CONTROL

FIGURE 1.2.4-2 FUNCTIONAL BREAKDOWN OF CM/PMAD POWER NETWORK CONTROL

At this point in the CM/PMAD partitioning, it is obvious that most of Functions 1.0 (Power Conditioning) and 2.0 (Power Distribution) will probably be controlled by hardware, while most of Function 3.0 (Power Network Control) will probably be controlled by some mix of software types.

1.2.5 Estimates Relevant to Software Development of Function 3.0

The next step in the functional partitioning of CM/PMAD is to consider the (provisionally assumed) software development of Function 3.0, Power Network Control. Begin by determining which subfunctions of Function 3.0 could be controlled entirely by a single type of controlling entity. Next, estimate a rough necessary capability (or complexity) of each subfunction. Then make rough estimates of the difficulty of developing the necessary software to control the subfunctions. The results are summarized in Table 1.2.5-1. Table 1.2.5-1 is arranged to follow the functional breakdown of Figure 1.2.4-2 from left to right.

1.2.6 Results of Partitioning

The final step is to perform the actual partitioning of the entire CM/PMAD task, Functions 1.0, 2.0, and 3.0. Table 1.2.6-1 shows the final partitioning of Functions 1.0 (Power Conditioning) and 2.0 (Power Distribution). Table 1.2.6-1 is arranged to follow the functional breakdown of Figure 1.2.4-1 from left to right. Table 1.2.6-2 shows the final partitioning of Function 3.0 (Power Network Control). Table 1.2.6-2 is arranged to follow the functional breakdown of Figure 1.2.4-2 from left to right. The Applicable Partitioning Rules referred to in Tables 1.2.6-1 and 1.2.6-2 are the rules presented in Table 1.2.3-1.

Table 1.2.3-1 Rules for Function Partitioning of CM/PMAD

1. Functions should be implemented by the most life-cycle cost-effective approach.

2. The implemention of a function must be capable of meeting all requirements specified for the function.

3. It is usually life-cycle cost-effective to implement simple functions in algorithmic software, where required response times are greater than a few milliseconds.

4. A function that requires a response in the microsecond range should be implemented in hardware.

5. A function that is responsible for last-line-of-defense crew safety should be initiated by hardware or by crew action.

6. Hardware should not be used to implement a complex function that allows a response time in the minutes or longer range.

7. Complex functions cannot require a response time equal to or less than a few milliseconds.

8. Complex functions that require responses in a few seconds should be implemented in software.

9. Complex functions that require responses in equal to or greater than tens of seconds may be implemented in algorithmic software, expert systems or crew.

10. Functions that require an expert's knowledge and judgment should be implemented by expert systems or expert aided crew.

11. Expert functions that are required to respond in seconds or less are sufficiently small to allow implementation in algorithmic software.

12. Complex functions should be implemented in algorithmic software, expert systems or crew.

13. A function that is responsible for last-line-of-defense equipment safety must be implemented by hardware, crew, or algorithmic software.

14. A function that is responsible for last-line-of-defense of experimental equipment safety or experiment data safety may be implemented by hardware, algorithmic software, expert systems, or crew.

15. Functions that occur predictably and periodically on a less than or equal to weekly basis are usually most life-cycle cost-effective when implemented in algorithmic software or expert systems.

16. Expert functions that occur predictably and periodically on a less than or equal to weekly basis are usually most life-cycle cost-effective when implemented in expert systems.

Table 1.2.3-1 (concl)

17. Simple and complex functions that occur predictably and periodically on a greater than or equal to monthly basis are usually most life-cycle cost-effective when implemented by algorithmic software, expert system, or crew action.

18. Expert functions that occur predictably and periodically on a greater than or equal to monthly basis are usually most life-cycle cost-effective when implemented by expert-aided crew.

19. A complex function that has totally predictable and absolute ranges of inputs is usually most life-cycle cost-effective when implemented in algorithmic software.

20. A function that has no predictability of inputs should not be considered for automation.

21. Expert functions that occur unpredictably but periodically less than or equal to a monthly basis are usually most life-cycle cost-effective when implemented in expert systems.

22. Expert functions that may occur unpredictably and require a response in less than fractional hours are usually most life-cycle cost-effective when implemented in expert systems.

23. Any function routinely and historically implemented in software is probably most life-cycle cost-effective when implemented in software.

24. A function that is on-line and in the real-time control loop for the initial operating configuration should not be implemented in an expert system.

25. Any function routinely and efficiently implemented in hardware is probably most life-cycle cost-effective when implemented in hardware.

26. Expert functions that may occur unpredictably and require a response in hours may be life-cycle cost-effective when implemented in expert systems.

TABLE 1.2.5-1 PRELIMINARY ESTIMATES, ASSUMING S/W CONTROL OF FUNCTION 3.0

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
DISTRIBUTION MANAGEMENT							
NETWORK STATE ASSESSMENT							
- SWITCHING STATE TABLE UPDATE	X			X			LESS THAN 10 SECONDS
- REDUNDANCY ASSESSMENT	X			X			LESS THAN 10 SECONDS
POWER PATH SELECTION							
- COMMAND SEQUENCE GENERATION	X			X			LESS THAN 10 SECONDS
- COMMAND STATE TABLE UPDATE	X			X			LESS THAN 10 SECONDS

TABLE 1.2.5-1 (cont)

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
LOAD MANAGEMENT							
LOAD MONITORING							
- POWER MONITORING	X			X			LESS THAN 10 SECONDS
- ENERGY CALCULATION	X			X			LESS THAN 10 SECONDS
ON-BOARD SCHEDULING							
- MAJOR SCHEDULING			X			X	APPROXIMATELY 0.5 HOURS OR LESS
- LOAD REQUIRE-MENTS PROJECT-IONS		X				X	LESS THAN 1 MINUTE
- MINOR SCHEDULING			X			X	5 MINUTES OR LESS
LOAD SHEDDING	X				X		LESS THAN 10 SECONDS

TABLE 1.2.5-1 (cont)

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
LOAD MANAGEMENT (con't)							
LOAD PRIORITY LIST MAINTENANCE							15 TO 20 MINUTES
- LOAD SCHEDULE ASSESSMENT	X				X		15 TO 20 MINUTES
- LOADS AVAILABILITY ASSESSMENT	X			X			15 TO 20 MINUTES
- OPERATIONAL REQUIREMENTS INTERPRETATION			X			X	15 TO 20 MINUTES
- LOAD PRIORITY ASSIGNMENT	X			X			15 TO 20 MINUTES

TABLE 1.2.5-1 (cont)

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
HEALTH MANAGEMENT							
MAINTENANCE SUPPORT							
- STATUS PREDICTION			X			X	1 WEEK TO 1 MONTH
- PREVENTIVE MAINTENANCE SCHEDULING		X				X	LESS THAN 1 DAY
- NETWORK SOLUTION		X			X		10 TO 60 SECONDS
- MONITORING		X			X		LESS THAN 10 SECONDS
- HISTORY RECORDS GENERATION	X					X	LESS THAN 10 SECONDS

TABLE 1.2.5-1 (cont)

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
HEALTH MANAGEMENT (con't)	▨	▨	▨	▨	▨	▨	▨
FAULT MANAGEMENT	▨	▨	▨	▨	▨	▨	▨
FAULT DETECTION	▨	▨	▨	▨	▨	▨	▨
- R/T PORTION	X			X			LESS THAN 10 SECONDS
- BACKGROUND PORTION		X			X		10 TO 60 SECONDS
FAULT ISOLATION	▨	▨	▨	▨	▨	▨	▨
- R/T PORTION		X			X		LESS THAN 10 SECONDS
- BACKGROUND PORTION		X				X	10 TO 60 SECONDS
FAULT COMPENSATION	X				X		LESS THAN 10 SECONDS
FAULT LOGGING	X				X		LESS THAN 10 SECONDS

TABLE 1.2.5-1 (cont)

FUNCTION	NECESSARY CAPABILITY			ESTIMATED DIFFICULTY OF DEVELOPMENT			DESIRABLE RESPONSE TIME
	SIMPLE	COMPLEX	EXPERT	LOW	MODERATE	HIGH	
COMMAND/DATA I/F		X				X	LESS THAN 10 SECONDS

TABLE 1.2.6-1 PARTITIONING OF FUNCTIONS 1.0 AND 2.0

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
POWER CONDITIONING		
- LOGIC POWER CONVERSION	HARDWARE	25
- LOGIC POWER CONDITIONING	HARDWARE	25
POWER DISTRIBUTION		
- CIRCUIT PROTECTION	HARDWARE	4,25
- LOAD SWITCHING	HARDWARE	25
- BUS SWITCHING	HARDWARE	25

TABLE 1.2.6-2 PARTITIONING OF FUNCTION 3.0

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
DISTRUBITION MANAGEMENT	▨	▨
NETWORK STATE ASSESSMENT	▨	▨
- SWITCHING STATE TABLE UPDATE	ALGORITHMIC SOFTWARE	3,23,24
- REDUNDANCY ASSESSMENT	ALGORITHMIC SOFTWARE	3,23,24
POWER PATH SELECTION	▨	▨
- COMMAND SEQUENCE GENERATION	ALGORITHMIC SOFTWARE	3,23,24
- COMMAND STATE TABLE UPDATE	ALGORITHMIC SOFTWARE	3,23,24

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
LOAD MANAGEMENT		
LOAD MONITORING		
- POWER MONITORING	ALGORITHMIC SOFTWARE	3,23,24
- ENERGY CALCULATION	ALGORITHMIC SOFTWARE	3,23,24
ON-BOARD SCHEDULING		
- MAJOR SCHEDULING	EXPERT SYSTEM	10,22
- REQUIREMENTS PROJECTION	ALGORITHMIC SOFTWARE	6,9,12,19
- MINOR SCHEDULING	EXPERT SYSTEM	10,22
LOAD SHEDDING	ALGORITHMIC SOFTWARE	3,24

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
LOAD MANAGEMENT (con't)		
LOAD PRIORITY LIST MAINTENANCE		
- LOAD SCHEDULE ASSESSMENT	ALGORITHMIC SOFTWARE	3,24
- LOADS AVAILABILITY ASSESSMENT	ALGORITHMIC SOFTWARE	3,24
- OPERATIONAL REQUIREMENTS INTERPRETATION	EXPERT SYSTEM	10,22
- LOAD PRIORITY ASSIGNMENT	ALGORITHMIC SOFTWARE	3,24

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
HEALTH MANAGEMENT		
MAINTENANCE SUPPORT		
- STATUS PREDICTION	EXPERT SYSTEM (WEEKLY)	10,15,16
	EXPERT-AIDED CREW (MONTHLY)	10,18
- PREVENTIVE MAINTENANCE SCHEDULING	ALGORITHMIC SOFTWARE (PART OF TIMELINE GENERATION)	12,19
- NETWORK SOLUTION	ALGORITHMIC SOFTWARE	9,12,19,23,24*
- MONITORING	ALGORITHMIC SOFTWARE	8,9,12,19,23,24
- HISTORY RECORDS GENERATION	ALGORITHMIC SOFTWARE	3,23,24

* rule 24 applies only if the solution can be done within one control cycle (10 sec)

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
HEALTH MANAGEMENT (con't)		
FAULT MANAGEMENT		
FAULT DETECTION		
- R/T PORTION	ALGORITHMIC SOFTWARE	3,23,24
- BACKGROUND PORTION	ALGORITHMIC SOFTWARE	9,12,19,23*

*rule 23 applies particularly in the case of microprocessor self-test

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
HEALTH MANAGEMENT (con't)		
FAULT MANAGEMENT (con't)		
FAULT ISOLATION		
– R/T PORTION	ALGORITHMIC SOFTWARE	8,9,12,19,24
– BACKGROUND PORTION	ALGORITHMIC SOFTWARE	9,12,19,23*
FAULT COMPENSATION	ALGORITHMIC SOFTWARE	3,23*,24
FAULT LOGGING	ALGORITHMIC SOFTWARE	3,23,24

*rule 23 applies particularly in the case of distributed microprocessor control

TABLE 1.2.6-2 (cont)

FUNCTION	PARTITION	APPLICABLE PARTITIONING RULES
COMMAND/DATA I/F	ALGORITHMIC SOFTWARE	8,9,12,19,23,24

1.3 EXPERT SYSTEMS ROLE EVALUATION

As a part of the work to automate CM/PMAD on space station, the role that expert systems/knowledge-based systems (ES/KBS) might play has been evaluated. This role evaluation consisted of gathering information about various functions that an autonomous power system must perform, then evaluating each of these with respect to a set of expert systems applicability criteria. The evaluation covered not only explicit CM/PMAD functions, but also possible functional implementations beyond IOC.

It must be noted that the terms "expert system," "knowledge-based system," and "rule-based system" are not synonymous. Within the context of this report, the term expert system will mean any software system that performs a complex, well-defined task using the same input information and problem-solving strategies as a human expert and that has the capability to make accessible to a user the reasoning it uses to perform the task. Expertise within such a system has its origins in the experience one or more persons have accumulated while performing the task. The term "knowledge-based system" refers to a software system that implements problem-solving knowledge that may have come from a human expert, textbooks, or other knowledge sources. This distinction is important because there are no human experts who have experience managing a power system like that which will be on the common modules, as none has ever been built. Certain aspects of this problem will be similar enough to existing tasks that the experience of humans can serve as the basis for a software system, but in other cases this will not be true.

The third term, "rule-based systems", refers to a specific method of implementing a knowledge-based system or an expert system, and there are other methods that may be used. Rule-based programming has proven to be an effective vehicle for capturing a human expert's knowledge of a problem.

A primary source of information about the functions evaluated was the CM/PMAD function partition matrix, section 1.2, that identifies those functions that should be automated intelligently and those that should be handled by conventional means. Other sources include Weeks (ref 1), Weeks and Bechtel (ref 2), the ATAC report on automation (ref 3), and an article by Prerau (ref 4), that details criteria used to select an appropriate domain for an expert system.

A number of functions meet the criteria well enough to warrant the application of ES/KBS techniques to their implementation. Of these, some are specific to the CM/PMAD system as it is defined, while others are less limited in scope or are intended as explorations into new technology applications, proof-of-concept exercises, etc.
Applications for ES/KBS specific to the CM/PMAD system include load priority list maintenance, and status estimation and prediction for system health maintenance. More general applications include dynamic load scheduling, fault management, and maintenance procedures advising.

Certain nonessential functions can be implemented as isolated expert systems considered experimental and having the status of other payloads. One of these is a fault analysis system specific to a hardware device with which a crew member would interact to diagnose a program with that device but which would have no controlling hardware connection to it. Another is an expert system that would monitor the operation of a payload and attempt to operate it in a power-efficient manner.

This paper discusses these functions and their evaluation with respect to ES/KBS applicability criteria. Section 1 explains some of these criteria. Section 2 describes specific CM/PMAD applications, and Sections 3 and 4 examine more general applications and experimental applications, respectively. Section 5 provides some general comments and recommendations concerning the development of these systems.

1.3.1 ES/KBS Applicability Criteria

1.3.1.1 General Criteria--The applicability of ES/KBS techniques to a task depends on problem complexity and scope, availability of alternative approaches, decomposability of the task, and the nature of reasoning processes used to perform it. A good task will be difficult but not inordinately so; it will require a few minutes to a few hours for a person to perform. Conventional approaches to the task will not be satisfactory. The task will be well defined, with clearly specified ranges of inputs and outputs. It will not attempt to be expert in an entire field, but only in a limited subdomain within that area.

The task will require the use of heuristics that efficiently partition a large problem space or that allow decisions to be based on uncertain or incomplete information. The knowledge in the system will be domain-specific; weak problem-solving methods (i.e., deduction from first principles) are not required, nor is deep reasoning based on causal analysis.

Decomposability is essential. A large problem must be broken into a set of subtasks that can be attacked independently. Within each subtask a basic approach must be feasible that can be refined later so that special cases need not be treated immediately. The program can in this way be prototyped easily, and changes made to the approach before the methods initially chosen are set in stone. A decomposable task will more easily be gradually phased into operation, with those subtasks in which there is confidence allowed control earlier than others. The task must allow the program to be tested in the environment for which it is designed without being given control until it is trusted.

1.3.1.2 Criteria Specific to Expert Systems--An expert system implements a problem solution based on the experience a person has. Therefore, a human expert must be available to the project, and must be willing and able to give his knowledge to the program designer, who will translate this into a representation interpretable by software. The expert must be recognized as such by others in the field, or the program will not be trusted. The expert must be patient enough to put up with the knowledge engineer's initial lack of domain expertise. He will be asked questions about issues that seem trivial or irrelevant. The knowledge-engineering process and subsequent software development will require a substantial amount of the expert's time, so there must be a commitment to the project on the part of management.

An expert system can be thought of as an archive of expert problem-solving knowledge specific to some domain. If this knowledge is scarce, the system development will be looked on much more favorably than if there are many experts, who may look at the program as a competitor. It is important that these and other domain area personnel support, or at least do not hinder, program development.

A task that is frequently performed but requires real expertise is an especially good candidate for automation, as experts tend to become bored by repetition, preferring to go on to new and more challenging problems.

1.3.2 Applications within Power Network Control

1.3.2.1 Load Priority List Maintenance

1.3.2.1.1 Description--One of the CM/PMAD tasks detailed in the function partition is generating a load priority list. This list will control the selection of loads to be disconnected in the event of a reduction in available power, and the choice of loads to be connected if an unexpected increase in power availability occurs. Algorithmic software can be used to generate a list using a static set of specifications, but there are occasions where a special mission requirement may affect the load priorities, or a complex set of circumstances may require the application of seldom-used criteria.

The task of load priority list maintenance is to assess the priority list and determine if there are circumstances indicating that a change is needed in the list. Priority list assessments will be made whenever the list is updated, every 15-20 minutes.

This function does not include generating the criteria used in decision-making. Some of these criteria will depend on the space station configuration; others will change more frequently, and these changes must be communicated to the list maintenance function regularly.

1.3.2.1.2. ES/KBS Applicability--This function should be implemented as a knowledge-based system, incorporating the list assessment criteria as rules that an interpreter can apply when appropriate. The complexity of the task, as well as its data-driven nature and frequency of performance, indicates that it is a prime candidate for intelligent automation of this type. The criteria will vary over time, and this implementation approach will accommodate changes in criteria without necessitating restructuring of the software that performs the task. Rapid prototyping will be made easier because only a small subset of the rules need be generated for the system to be executable and testable.

1.3.2.2 Health Status Prediction

1.3.2.2.1 Description--In a system as complex as CM/PMAD, it will not be enough to handle component failures as they occur. The health of the system must be continually monitored and an attempt made to predict its future status to enable replacement of elements before they fail. Engineering data will be accumulated on the ground and analyses and predictions made on a weekly basis, with more thorough analyses done monthly. In this way, the number of system failures can be reduced and the necessary replacement units made available before they are needed.

1.3.2.2.2 ES/KBS Role--This function should be implemented as an expert system. The difficult aspect of the task is that it requires the analysis not only of single variable values or value trends but of interactions between these that can indicate that failure is imminent even though no single value or trend would. Recognizing these circumstances requires a high degree of power systems expertise. Also, many CM/PMAD system components will degrade in a more-or-less continuous manner, as opposed to failing all at once, and judgments must be made as to when a system or interaction is such that a component must be replaced. The task will involve large amounts of data, and only some of them will be relevant to any particular prediction. These factors, combined with the frequency and regularity of task performance, indicate that an expert system would be the most cost-effective method of function implementation.

1.3.3 Systems Involving Power Control or Maintenance

1.3.3.1 Rationale--It is essential to consider applications that are not specifically limited to CM/PMAD functions, especially with FOC space station in mind. Intelligent programs that control a subsystem on the common modules need information about other systems that interact with and are integrated with that subsystem, just as a human expert would. A typical example might be a situation where a temperature sensor goes out of range. This may be caused by a power system failure or a mechanical failure such as loss of lubrication in a bearing, or a blockage in a duct that causes a pump to work harder and overheat, etc. An optimally autonomous common module will require expert systems that span subsystem boundaries, interacting with other expert systems and managing the interactions between subsystems or the resources used by or involved with more than one subsystem. The following applications are involved with power management but are not specific to CM/PMAD.

1.3.3.2 Dynamic Load Scheduling

1.3.3.2.1 Description--The task of dynamic load scheduling involves creating and modifying load activation schedules subject to constraints on resource availability, load characteristics, mission priorities, etc, that vary continuously. An optimal schedule will achieve mission goals in the shortest time possible without violating power availability and other constraints. On the common modules, it is expected that power requests will total many times that which is available, so this is an important and ongoing problem. Included in this task is the problem of efficiently handling situations where a schedule must be abandoned unexpectedly and a contingency schedule developed on short notice.

1.3.3.2.2 ES/KBS Applicability--Load scheduling is a costly, human-intensive task that requires a great deal of experience. It belongs to a class of problems whose complexity rises exponentially with the number of operations, and it requires heuristics developed through years of scheduling experience to deal with this complexity. The task is well defined and to an acceptable extent decomposable, though not optimally so. An expert system could be developed that deals with some subset of the constraints human schedulers handle, with more information added incrementally. As confidence in the schedules the system creates increases, its use can be gradually phased into operation.

The payoff associated with development of a program such as this will be large, because there are many similar applications that this system could handle well, with only modifications to the domain-specific parts of the knowledge base. The program could include other functions such as the analysis of power consumption trends to determine actual power availability, and provision of interrogation options to allow crew members to determine power system status and the reasons for scheduling choices or conditions that led to contingency scheduling, for instance.

1.3.3.3 Fault Management

1.3.3.3.1 Description--Fault management in the common modules consists of detecting an abnormal state in a system, isolating the fault(s) that caused the abnormal state, and suggesting (or initiating) an action that brings the system back to an operationally sound state. A typical example would be a short circuit that causes an increase in current through a breaker. The breaker is tripped, shutting off current through the circuit. A fault management system would monitor that circuit, recognize the current shutoff, isolate the cause of it, and suggest that an alternate circuit configuration be selected. The program must have information about all of the common module subsystems because a problem in one system may cause a recognizable fault in another. The types of faults dealt with must not be such that a real-time response is required.

1.3.3.3.2 ES/KBS Applicability--Fault management systems have been built for a variety of applications, and though the common modules are potentially more complex than in previous projects, a knowledge-based approach appears to be a cost-effective alternative. The program will need to interface with various hardware systems and possibly with other software, such as the contingency payload scheduler as well. It can provide a focus for automation in several subsystems. Monitoring these and their interactions will be time intensive and will require more intelligence and flexibility than algorithmic software can provide. Also, it is assumed that the hardware aboard the common modules will change over time, and a knowledge-based approach will make it easier to keep the fault management system up to date.

1.3.3.4 Maintenance Procedures Advising

1.3.3.4.1 Description--The various subsystems aboard the common modules will be continuously monitored, and when a fault occurs, it will be necessary to come up with a short-term workaround and to replace the unit responsible for the fault. Maintenance procedures advising involves giving the crew (or perhaps an autonomous robot at FOC) the information needed to carry out service procedures, both unscheduled, as above, and routine maintenance such as filter replacement, parts lubrication, etc. Information the crew member needs includes the location of the unit, the states that various systems must be in for the operation to be carried out, and step-by-step instructions for the procedure, including contingency information to handle foreseeable abnormalities.

1.3.3.4.2. ES/KBS Applicability--This task is a good candidate for a knowledge-based approach. The number of interacting subsystems on the common modules will be large, and crew members will be rotated too frequently to allow sufficient time for them to become expert at maintaining these changing systems. The task is amenable to the kind of incremental development and phase-in appropriate to ES/KBS techniques. Until the systems to be maintained take on more definition, there will be no expert at this task, so the knowledge base will be developed by analyzing subsystem construction and operation.

1.3.4 Applications as Onboard Experiments

1.3.4.1 Fault Analysis--There may be hardware devices on the common module that will be sufficiently complex that they will require expert troubleshooting, but which are not amenable to routine transport back to Earth. An example of such a system on Earth is the PROFS workstations used by the National Weather Service. These workstations are distributed throughout the country and require an expert technician to travel to remote sites to repair them--at great expense. A diagnostic expert system a la MYCIN could be used to assist a crew member in deciding how to find a faulty component in a device of this sort, how to replace the defective part, and how to carry out routine testing and maintenance. Such a system would free an expert on the ground and the communication channel he would use to help the crew member repair the device. The program could have a minimal hardware interface with the device but would more likely carry on a dialogue with the crewman, requesting information and presenting diagnoses and explanations on the screen.

In deciding whether to implement such an expert system, a number of factors must be considered. For example, how critical is rapid repair of the hardware; what is the cost of developing the expert system; what is the cost of keeping a human expert on hand; what is the cost of the computer used for the expert system (including expenses in weight, power, and maintenance of the computer itself; and how likely is it that a human expert would have to remain on call anyway? Other factors include how common the expertise is and the ability of the crew to make repairs once they know the problem.

1.3.4.2 Payload Power Monitoring--Many of the payloads aboard the common modules will be capable of a variety of modes of operation, differing in the amount of power they consume. An expert system specific to a particular experiment could monitor its operation, analyzing its power use trends to communicate to the payload scheduler or health status monitor, and operating the payload (or suggesting an operational regime) that would be most power efficient and would accomplish the mission goals for that payload. Included in this system could be fault detection software that would make certain the payload could not draw more power than it would under normal operating conditions.

1.3.5 Recommendations

Among applications within power network control, the load priority maintenance system is the best candidate for current implementation. Because this system will actually control hardware (by deciding with no human intervention which loads will be shed), it will be necessary to test and refine the system incrementally over time. This will assure that the system is modifiable and can be trusted.

Work on health status prediction could start now, using expertise developed from experience with other power systems. Current power systems test bed projects such as the Autonomously Managed Power System (AMPS) could become a proving ground for this type of expert system. Testing (and raising confidence in) the system will be a major issue. Prediction problems typically are such that only good statistical analysis can prove a program useful, and this requires a large number of system tests with real data.

All of the applications described earlier that are not limited to CM/PMAD can be approached now. Dynamic load scheduling will require a great deal of work, because within certain contexts the general scheduling problem is still a difficult research issue, but this work will result in enormous benefit in terms of the number of man-hours saved and the range of applicability of the solutions devised. Temporal modeling is a hard problem, as is the size of the task in this domain. The work done here on the Energy Management Expert System (EMES) has provided some insights into the problem.

The fault management and maintenance procedures advising tasks are more dependent on specific common module systems definitions, but these systems are sufficiently defined to allow work to begin in knowledge acquisition and representation. Both systems will be large and complex, and work on them must begin early if they are to play a role in initial operational configuration (IOC) space station.

The ES/KBS applications as onboard experiments are comparatively small, self-contained systems and can be considered for development whenever the associated devices or payloads become available.

1.4 AUTOMATION ARCHITECTURE ISSUES

Under this subtask we addressed and analyzed the issues of distributed versus centralized automation, fault isolation, load management, interfaces to space station power and data buses and their returns, separation of data and power grounds, local energy storage, crew interfaces, sensing requirements, and sensing techniques as they pertain to the automation of the CM/PMAD system.

1.4.1 Distributed versus Centralized Automation Analysis

Automation is defined as "the use of machines to control and/or carry out processes in a predefined or modeled set of circumstances without human intervention." Centralized automation is defined as a central computing element for processing of data resulting in control decisions and command issuance. Distributed automation is defined as several or more computing elements either arranged single level, or hierarchically with several levels. The significance in the difference between centralized and distributed is that distributed automation allows both preprocessing of data (hierarchical arrangement) and processing of data in smaller amounts, in parallel fashion on each level. For this report, all automation approaches are within the power subsystem, i.e.:

1) Centralized refers to a single processing element in the space station power subsystem;

2) Distributed refers to multiple processing elements on a single level, i.e., one power subsystem processor for each module or element;

3) Hierarchically arranged distributed system (HADS) refers to a hybrid approach, using both central processor(s) and distributed processors arranged in hierarchical fashion.

Processing, although not totally interchangeable with automation, is an important part of automation with respect to comparisons in the distributed versus centralized question. Distributed processing has recently been a favorite candidate for functions and tasks that were related through processing resources available and not related through dependence on a large amount of shared task-dependent data. Complexity of those distributed systems greatly increases when the amount of shared dependent data increases. However, improvements in communication links, e.g., local area networks (LAN), between processors have significantly reduced the complexity factor and, therefore, risk. Conversely, a processing system must be increased in required size, power, and processing speed as more functions are allocated to a central computer.

The analyses and comparisons performed here pertain to the CM/PMAD subsystem. However, the overall station power system, including source, conversion, storage, and distribution, is briefly considered for completeness. Experience gained from the automated power systems management (APSM) (Ref 5) summarized in Table 1.4.1-1, indicates the HADS approach is preferred for the overall station power system. Additionally, Autonomous Spacecraft Design and Validation Methodology Handbook, Issue 2, (Ref 6) indicates a hybrid between centralized and distributed approaches is preferred for spacecraft that require a high level of processing complexity. Major keys in our recommendation of the HADS approach are risk, growth capability, subsystem performance, modularity, and adaptability to change. The overall modular approach selected in the reference station configuration lends itself particularly well to the distributed approach in overall power system automation. In the HADS approach, the shared data between the station-level managers of power and the CM/PMAD manager of power are simplified to power and energy requirements and resource allocations. Using the HADS approach, each station module has maximum flexibility in local power subsystem use. This approach most closely resembles the power network of a typical commercial power company, a desirable goal in that the power subsystem of space station is more of a "utility" than ever before in a space project. The HADS automation approach down to the module level in turn requires processor capability at the top CM/PMAD level as a minimum.

The major advantage of HADS is that the approach uses the strong points of both the centralized and distributed approaches. The specific advantages of HADS from an overall general view in the comparison of distributed versus central power management and distribution control are:

1) Increased testability (distributed strong point);

2) Higher growth capability and flexibility (distributed strong point);

3) Increased speed of information transfer (distributed strong point);

4) Lower development risk (centralized strong point)

A centralized approach for the entire power system at the space station level would be impractical if not impossible to ground test before flight without building the entire station power system and related loads or without a very large amount of simulation. Either testing approach for the centralized power system control would be very expensive and possibly render the testing results questionable in the case of widely used simulation. Alternatively, the distributed approach with its modularity lends itself particularly well to verification testing, both in development and preflight. This especially pertains to distributing control to at least the module level. The control and operation of each power system element (i.e. power generation and energy storage, distribution points, and module power management and distribution) could be checked out separately before flight, accommodating buildup of the station. Within the module, testing is also simplified in the distributed approach, but the factor is not as significant.

Table 1.4.1-1
APSM Factors in Selecting Distributed vs Central Computer Architecture

APSM Requirement	Central Approach	Distributed Approach
1. Computational Rates	Time Sequenced, Timing Criticality	Concurrent Processing of Critical Processes
2. Hardware Cost	Initially Lower	Initially Higher
3. Packaging	Complex, Heavy Wire Harness	Simple System Interconnect
4. Software Complexity	Complex Executive Software, Software Module Interaction	Simple Executive Software, Independent Partitioned Software Modules
5. Software Development Cost	Equivalent (Complex Interaction)	Equivalent (More Software To Be Developed)
6. Reliability	Complete Loss Due Tc Processor Failure. More Expensive To Add Redundancy	Graceful Degradation Cheaper To Add Redundancy
7. Development Risk	Greater	Less
8. Adaptive to Change	More Expensive	Less Expensive
9. Growth Capability	Hard to Expand	Easy to Expand
10. Spacecraft Interface	Equivalent	Equivalent
11. Subassembly and System Testing	Complex, Expensive	Simpler, Lower Cost
12. Future Data System Trend	Decreasing	Increasing

Growth capability and overall flexibility are major advantages of a distributed system, either single or multiple level. As long as the interfaces are carefully maintained, the particular functions in the distributed processors are easily modified without major testing of the overall system being necessary. Conversely, in a centralized system, the addition, deletion, or modification of a function would cause validation testing of the entire system to assure changes have not violated timing constraints or degraded overall system performance to an unacceptable level.

The speed of information transfer is also an important consideration for comparing distributed to centralized power system control. Ideally, the exact requirements for CM/PMAD automation would be specified and always remain unchanged. The functions can be well defined, partitioned, and allocated, but the automation requirements with respect to speed of reaction, for example, depend heavily on electrical loads and their requirements. The nature of space station and its projected long-time operational life precludes defining these detailed load requirements with a guarantee of no change (or even no major change). The speed of reaction to an event is very important to the trade between centralized and distributed control of the power management and distribution system. How long can a load be disconnected (or connected) or a failure or fault go undetected before the controlling section of the power system must actually be apprised and react? An answer given to this question can be as fast as is technically feasible, but within reasonable hardware and software constraints.

The worst case data rates and required processing time for centralized, distributed and HADS approaches, as determined in Section 1.6 of this report, show clearly and not surprisingly that the HADS approach reacts fastest of all approaches where it is assumed that all relevant data must be transmitted to the decisionmaker for processing. Under the assumptions of Section 1.6, the centralized approach (using a total of six modules with the common module as a baseline) is limited to hundreds of milliseconds reaction time, while the distributed is less than a hundred milliseconds. The HADS approach can react in less than ten milliseconds. Where the required reaction time and required subsystem performance has not yet been totally specified, HADS represents the least risk approach for meeting those eventual requirements that may change during the design phase of space station.

Finally, the HADS approach lends itself to overall commonality in the power subsystem and perhaps across other subsystems as well. Additionally, the HADS approach has a high degree of modularity, thus reducing development risk. Software programs can be smaller and more manageable; however, it should be noted that for the HADS approach, software interfaces increase and must be well thought out early in the design phase, keeping the interfaces as simple as possible to maximize the advantages of the HADS approach.

It must be noted that the availability of reliable, cost-effective, fault tolerant processors, and supporting integrated circuits having low weight and power characteristics, i.e., VLSI and CMOS processors was assumed.

In the HADS approach, as many functions and subfunctions as possible are allocated to the lowest level physical elements as defined by the ability to perform the decision process with a minimum of external data required. The approach is to make the lowest level controllers as small as is reasonably possible while maintaining the required capability for the allocated functions. Primarily, lowest level functions include the following:

1) Data acquisition;

2) Data conditioning;

3) Data synchronization (time stamping for analysis in system solution);

4) Data compression;

5) Limit checking;

6) Local fault handling;

7) Short term data storage;

8) Requested data transfer;

9) Effectors control.

The major functional responsibilities of overall distribution, load ranking and scheduling, and subsystem health management are allocated to the top-level CM/PMAD controller in the HADS approach.

1.4.2 Fault Isolation Analysis

In 1985, Martin Marietta Denver Aerospace successfully demonstrated FIES-II, an expert system computer program that diagnoses hardware faults in a breadboard dc power system. This program, developed under contract to NASA's Marshall Space Flight Center, demonstrates that artificial intelligence software can effectively locate faults in a power system by examining telemetry data.

FIES-II demonstrates the handling of faults of moderate complexity, but it is limited to finding a single failure. However, this failure may be any of several types, including a remote power controller stuck open or closed, a bus shorted to ground, an overload, or a resistive path where none should exist. Because the power system breadboard used in the demonstration had multiple buses, these faults could change the interconnections between sources and loads, resulting in numerous subtle problems.

An actual space station power system might exhibit faults beyond what FIES-II could analyze. For one thing, the space station's power system is not a simple dc system but a three-phase ac system. This means that faults can develop on any of the three phases. Further, an ac system has more parameters than voltage and current to examine for faults--a problem may appear as an incorrect frequency or phase, low-power factor, or high distortion. In addition, multiple faults, failed sensors, and fault propagation must be considered as possibilities. Some faults may require causal reasoning for diagnosis.

Other faults in the power system may be easier to examine. For example, at the low end of the spectrum of fault complexity, circuit breakers have been used for many years to clear simple electrical overloads.

Between these extremes are numerous possibilities that must be considered.

Under this task, the range of possible faults that might occur in the space station power system were investigated, and findings and recommendations for approaches to detecting and isolating each kind are presented.

1.4.2.1 Fault Type Analysis--The term "fault" as used here embraces the full spectrum of malfunctions mentioned previously, and ranges from minor deviations from expectations to catastrophic failures. Faults in components that are part of CM/PMAD or are connected to it are considered here. In this context, the term "computer" will mean all of, or any part of, the distributed intelligence in the common module and will include conventional software as well as any expert systems or other artificial intelligence software.

1.4.2.1.1 Problem External to Common Module--First, any automated power management and distribution system must be aware of the quality of power delivered from outside the common module. Problems in this area may not be caused by anything inside the common module, and there may be little that a system in the common module can do to isolate or correct such a fault. Nevertheless, failure to recognize such problems could lead to incorrect problem analysis and inappropriate action.

Faults in this category include low voltage, incorrect frequency, improper phase relationship between the three phases, and high harmonic distortion. For any of these problems, CM/PMAD has little choice of action: (1) switching to the alternate bus if that bus has better quality power; (2) recording the problem in a log; (3) notifying a higher authority; and (4) remembering the problem to guide future actions. None of these actions would require complex decisions beyond the capability of conventional software. On the other hand, attempting to diagnose the cause of such problems is not a suitable task for CM/PMAD, because this would require a great deal of information from outside the common module.

1.4.2.1.2 Open or Short Circuit--A second fault type is a short or open circuit in the network. In general, shorts will result in excessive current and will therefore be cleared by circuit protectors before any kind of computer could begin an analysis. This means that the computer analyzing the fault will need to recognize an open remote power controller that should be closed and deduce that there is a fault downstream from it; the fault itself may or may not appear in the telemetry the computer examines.

Our experience with FIES-II has convinced us that either conventional software or rule-based or expert system software could diagnose shorts and opens and take corrective action. However, either implicitly or explicitly, both types of software would need to include heuristics to guide the search for the fault, recognize the implications of an open remote power controller, or find an appropriate corrective action. We recommend rule-based software that may use frame-based representations because heuristics are most easily expressed in this form and because heuristics are considerably more difficult to recognize, modify, and debug in conventional software. In addition, rule-based or frame-based software will simplify the modifications that must be made as the space station grows and experiments are changed, because it minimizes the amount of code that must be changed and encapsulates information that is likely to change, isolating it from code that will not need modification.

1.4.2.1.3 Faulty Sensor--A third type of failure is a faulty sensor. This type of fault may be easy to diagnose or extremely difficult. The key factor that determines the level of difficulty is the amount of redundancy in the sensor configuration. For example, if there is no redundancy, a failed sensor will be very difficult to identify, because one problem can produce the same measurements as another.

The complexity of this problem is compounded by the number of ways a sensor can fail; sensors may be noisy, intermittent, stuck at one output value, or have an incorrect scale factor. In addition, if sensor data are multiplexed on a bus, one sensor's data may be mistaken for another's. Furthermore, all data passing through one multiplexer or bus interface may be garbled by a single failure in the multiplexer or interface.

We do not believe that the possibility of faulty sensors will greatly complicate the software for fault detection and isolation. However, making such software practical will require some constraints on design and will have some effect on operations.

Specifically, redundancy should be provided in critical measurements, particularly in bus voltage measurements. Without this redundancy, the software may be unable--as indeed a human might be--to distinguish among possible failure modes without conducting such experiments as switching a load from one bus to another, turning loads off or on, or having crew members change out an ORU, or report an observation. Although redundancy will minimize the chances for the software to ignore a failure or misdiagnose a problem, we believe that full redundancy is unnecessary. In fact, we expect the law of diminishing returns to limit sensor redundancy to a small fraction of the total number of sensors. This will, however, require allowing the software to conduct experiments to locate and isolate a fault.

1.4.2.1.4 Failed RPC--A fourth fault type is a failed RPC. These devices might fail in any of a number of ways. For example, an RPC may change state when no command is given or fail to change state when commanded to do so. This could result from a failure in circuitry that decodes the RPC's address or command from the control bus, or it could be a failure in a switching element. In either case, the problem is a subset of the short-or-open failure mode discussed previously as long as the software does not conduct experiments.

However, additional complexity results from an RPC that responds to the wrong address--performs a function another RPC was commanded to perform--or one that misunderstands a command. If the software is attempting to conduct experiments to isolate a fault, the malfunctioning RPC could take actions and send status data that would be very misleading. We believe that the software can cope with this malfunction, but it will have to be explicitly considered as a failure mode during software design.

Two other RPC failure modes would be difficult to detect unless an overload caused the RPC to trip. In the first of these modes, the RPC trips at a different current level than the one for which it was set. Because of the rarity of overloads and the length of the measurement interval, the software might have no way of determining that this type of failure has occurred. We believe that this type of failure is best examined by fault-tolerant design, a built-in self-test function, or periodic testing of the equipment rather than by the fault-isolation software.

The second difficult-to-detect failure mode is a changed trip delay. Specifically, circuit protectors typically are rated to carry a specified overload for a specified amount of time before tripping. If the delay characteristics change, there will generally be no way for the software to determine that they have. We believe that this type of failure is also best handled outside of the fault-isolation software.

A final RPC failure mode is a resistive closure, i.e., excessive voltage drop across the RPC when it is closed. The effects of this failure are distinctive and should be easily detected.

1.4.2.1.5 Faulty Power Conditioner--At this time, the role of power conditioners in the common module, if any, has not been determined. However, the failure modes typical of power conditioners--no output, incorrect frequency, low efficiency, high distortion, improper phasing, etc--produce symptoms that are readily traced to the conditioner. We therefore believe that fault isolation software will be able to properly diagnose this type of fault.

1.4.2.1.6 Power Control Unit (PCU) Failure--The PCU is assumed to be the highest-authority computer with algorithmic software in the CM/PMAD. Computer failures present the most complex set of possibilities, because, at least in principle, a computer malfunction can result in any of thousands of inappropriate actions. In practice, however, most computer malfunctions result in very obvious symptoms, e.g., the computer stops doing anything at all that is detectable from outside the machine, sprays random characters onto the terminal screen continuously, or prints an error message and halts. None of these failure modes would interfere with space station operability or place the common module closer to a life-threatening situation. Other failure modes might, if the computer is given too much control authority. For example, if it is possible for the PCU to turn off life-support systems, a malfunction could cause it to do so.

An error in PCU software could produce the same range of problems as hardware faults can produce, including sending misleading information to the fault-diagnosis computer if diagnosis is done in a separate computer. Such software and hardware techniques as self-test programs, watchdog timers, and similar handshakes with external equipment, error-correcting memory, and built-in consistency checks can reduce the likelihood of serious failures, but preventing such failures will require reducing the PCU's authority. This can be done by adding hardware that prevents the PCU from issuing certain dangerous commands without an enabling signal from an independent authority. This authority could be the crew, a second computer, or special hardware. In this case, the fault diagnosis and isolation would be provided by the independent authority.

Another factor to consider is that the PCU is assumed to control the RPCs, so if the PCU fault is to be isolated, it must be disconnected from the control bus and its functions taken over manually or by another computer. An architecture that allows such a switchover would introduce other equipment that would also have failure modes.

1.4.2.1.7 Fault-Diagnosis Computer--If fault diagnosis and isolation are done by a computer separate from the PCU, this computer will be subject to the same kinds of problems as the PCU is. To some extent these computers can check each other, but full checking would require embedding the programs of each in the other. Even then it would be unknown which computer had malfunctioned. A third computer could be added with a majority vote arrangement; but we do not believe this level of complexity is warranted in IOC.

1.4.2.1.8 Microprocessors Subsidiary to the PCU--It is assumed that microprocessors will be embedded in various subassemblies of the CM/PMAD to minimize the amount of logic circuitry required to interface sensors to the data/command bus, decode commands, perform self testing, provide status information to the PCU, etc. These microprocessors and their associated memories and support circuitry can fail in a variety of ways, the most probable of which is to stop performing any detectable function at all. This failure will be readily identified, because the fault isolation computer will be unable to communicate with it.

Certain other failures will be more difficult to isolate. For example, a microprocessor may babble on the data bus, making it impossible to communicate with any of the microprocessors. Or it may appear to function normally but report erroneous data or status information, change the state of an RPC or ignore a command to do so, or respond to commands directed to a different microprocessor.

Some of these problems may be misdiagnosed as a faulty RPC, but this would probably be acceptable, because the microprocessor likely would be housed in the same ORU.

Other failures of microprocessors may have to be isolated through experimentation. Failures that disable the data bus can be minimized by providing a redundant bus.

1.4.2.1.9 Load Faults—Fault isolation software for power management and distribution would generally view loads as "black boxes," i.e., it would have very little information about them. It would be beyond the scope of such software to attempt to troubleshoot problems internal to a load. However, where the load produces externally observable symptoms of malfunction, the software might be able to signal that there is a fault or log the fault. Specific examples include:

1) Current to the load is out of normal operating range but not high enough to trip the RPC;

2) The load's power factor is out of normal operating range;

3) The distribution of power consumption among the three phases is abnormal.

In addition, the software might be able to identify some cases where conducted emissions (electrical interference with the power system) from the load exceed allowable levels.

1.4.2.2 Recommendations--We recommend that experimental fault-isolation software be flown on IOC. This software would monitor the data coming from sensors and status indications coming from RPCs. It would also monitor the commands sent from the PCU to RPCs and maintain an internal software model of the state of the power system in the common module. It would then continuously look for indications of faults and notify the crew when one is detected. Finally, it would attempt diagnosis of the problem and report its findings to the crew. In diagnosing the problem, it may request such crew actions as toggling an RPC or reporting visual clues, e.g. whether an indicator light is on. We do not recommend giving this program control authority on IOC; it will be used to test the fault isolation concept and to gather performance data so that an enhanced version can safely be given some control authority by FOC. However, even at FOC we foresee a need for some human override capability.

The IOC version of this software should be able to diagnose shorts and opens, RPCs that are stuck open or closed, most sensor failures (with some redundancy in sensors and allowing for some experimentation), faulty power conditioners, and most types of failures of the PCU and microprocessors subsidiary to it. It would also be able to notify the crew of loads consuming too much or too little power. Such a program should be able to determine why an RPC tripped, although it may have to conduct some experiments to isolate the malfunction to the ORU level.

Switching to redundant hardware can sometimes be done without the level of intelligence required to diagnose a fault. For example, when bus voltage drops to zero because of a tripped RPC, conventional software could take corrective action to restore power to the bus before (or while) software using artificial intelligence techniques analyzes the details of the problem. In general, we recommend using conventional software for all tasks that are commonly and effectively handled by conventional software, particularly where it is important to rapidly get to a safe condition, even though this condition may not be optimum.

1.4.3 Load Management Analysis

Historically, load management has been a human-intensive activity based on the ground. It consists of scheduling the operation of the various electrical loads on a spacecraft power system to most effectively use the power available. To date the most practical approach to automating this activity has been automatic load shedding, i.e., having an onboard computer turn off the lowest priority loads in sequence until the power consumption has been reduced to an acceptable level.

This approach has not been fully satisfactory for several reasons:

1) Priorities change during a mission. For example, a low-cost experiment may initially have very low priority, but its priority may increase greatly if it has gathered 99 hours of data out of an intended 100 hours and must be completely restarted if it is turned off. Similarly, a load may require a considerable investment in power, crew time, or other resources to prepare for operation. After it is prepared, its priority may be very high although initially it could not be operated at all. Another example is a science experiment for which data-gathering opportunities are very rare. Such an experiment would usually have very low priority, but when conditions are right, it would have a very high priority;

2) One set of loads may be more useful than another set although it contains more lower-priority loads than the other set contains. This may happen, for example, when one high-priority load duplicates some of the functions of another high-priority load or when two loads working together produce much more benefit than the sum of their individual benefits;

3) The lowest-priority load that is on may consume a large amount of power. If this load must be turned off because of a power shortage, it may be possible to turn on several still lower-priority loads without exceeding the power budget. A simple numerical priority scheme would not detect this opportunity.

Because of such factors, it is highly desirable for any load management software for the common module to reason about more factors than a single numerical priority rating. On the other hand, reaction time constraints may make it necessary to minimize the reasoning done in shedding loads.

Under contract to Marshall Space Flight Center, Martin Marietta Denver Aerospace wrote a load-management computer program known as the EMES. This program considers a large number of factors for each load, including power consumption, requirements for operation in daylight or during eclipse, pointing requirements, whether the load can be restarted after interruption, or whether it should not be interrupted. The program can plan operation of the loads for a specified number of hours, replan for changed conditions while attempting to minimize schedule changes, or do emergency replanning when equipment failure suddenly reduces available power.

Although EMES served its purpose as a test bed for demonstrating that software can reason about some of the subtleties of load management, it would require a number of changes to make it suitable for managing loads on a real space station.

First, EMES was too slow. This problem has been examined in EMES-II, developed under independent research and development (IR&D) funding at Martin Marietta Denver Aerospace. Making such software fast with a rule-based design approach requires a well thought out mixture of production rules and algorithmic code. For example, the original version of EMES typically used "fired" rules 70 times to update a table of power availability versus time. Algorithmic code could have updated the table thousands of times faster. Many such inefficiencies are obvious only after considerable time has been expended on software design.

Second, EMES "knows" about loads by maintaining a set of parameters for each load. These parameters specify power consumption and a number of constraints on operation of the load, but they do not specify all the factors that human schedulers might consider. Furthermore, it is likely that the set of relevant factors will change considerably during the life of the space station, as will the values assigned to these factors. For example, the addition of an experiment may place a new and unanticipated constraint on the operation of other loads. Designers of load management software will have to be very careful to allow for the addition of new types of information about loads, new constraints, and new rules for using this new information without introducing errors or inconsistencies into the program.

A related problem is that any software with limited, parameterized knowledge of loads will probably miss constraints imposed by unanticipated events during a mission—equipment failures, emergencies, political factors, etc—to which human planners could readily adapt.

On the other hand, allowing for the fact that EMES was written in 1983-1984 when the design of the space station had not progressed very far, EMES produces schedules competently. Furthermore, EMES-II is faster than human planners and, unlike humans, does not suffer from fatigue, get careless when under pressure, get sick, or forget details. Such software would permit more frequent replanning as situations change, which might be more beneficial than having a better plan to start with and being unable to change it.

We believe that load management software similar in capability to EMES should be used for IOC, but it should be regarded as an experiment, not a controller. Such a program could be used to give a new priority list to conventional load-shedding software periodically, perhaps once every few minutes. Although, for reasons mentioned previously, this would not result in optimum load shedding, it would be a great improvement over any fixed-priority scheme, and would generally be superior to what human schedulers could do, because the priority list would always be fresh, reflecting the current situation. This approach is preferred over giving direct control to the software because it gives human observers a chance to modify its priority lists or replace them altogether. In addition, the conventional software that implements the load shedding based on this list can perform consistency checks by comparing it to a partially ordered list of prearranged priorities. For example, Load A may always have higher priority than Load B despite changes in either's numerical rating.

Using intelligent load-management software as an experiment will provide several benefits:

1) Better power use can be expected because of frequent revision of the priority list;

2) Observing the behavior of the software will provide the data needed to refine the scheduling rules, load-definition parameter set, and parameter values;

3) Using the software as an experiment will develop confidence in the program's competence without risk to mission success.

The recommended eventual approach to load management is described as a scenario in these following paragraphs. The description assumes that the schedule being composed is to cover all of the next crew shift period. Usually, the load scheduling function will be used for that specific purpose. It should be noted, however, that under extraordinary circumstances (loss of a solar panel, for example), the function may be invoked to reschedule the remainder of the present crew shift.

"Space station manager" refers to the software entity that manages that portion of the space station not contained within the modules.

Shortly before the beginning of each crew shift, the space station manager will inspect that portion of the ground-generated timeline that covers the coming shift. Using that information and using what it "knows" about the present status of stationwide resources, the space station manager will compute preliminary estimates of how much of each stationwide resource to allow each module to use during the coming shift. At the end of this process, the manager will have composed for each module a preliminary resource allowances list. These lists cover only available stationwide resources; the space station manager will not have direct information about resources available within a given CM.

At this point, the space station manager will alert the major scheduling functions of each module and will present to each module its individual list of preliminary resource allowances for the coming shift.

Aboard the CM, the major scheduling function will now compose a preliminary schedule of CM activities that would cover the coming crew shift. It is estimated that it will take about a half hour for the major scheduler to compose the schedule. Input data to this function will include, but not be limited to, the preliminary resource allowances list from space station, the CM portion of the ground-generated timeline, the CM/PMAD Redundancy Assessment Record of Function 3.1.1.2 (Appendix A), and various sensor measurements.

One of the most important products of this preliminary schedule of CM activities will be the preliminary load enable schedule. The preliminary load enable schedule will be a compact, chronological sequence of CM/PMAD events that would occur during the coming crew shift. Each event entry will contain the following information:

1) A load designator (identifying number or other label);

2) Whether that load is to be enabled (connected to electrical power) or disabled (disconnected from electrical power);

3) The time when this event is to occur.

As it composes the preliminary load enable schedule, the major scheduler will assign to each CM load a load class number. Load class is not the same as load priority. A load priority number exactly defines a load's CM-wide priority over a moderate time interval (about 15 minutes or less). No two loads in a given CM will have the same load priority number. A load class number coarsely defines a load's space station-wide priority over a wider time interval (a fraction of a crew shift period or more). Many loads may have the same load class number.

Each module on the space station will use the same algorithm to assign load class numbers to its own loads. By this means, the CM will be able to assign rough, stationwide priorities to each of its loads, even though it knows nothing about individual loads in other modules or about individual loads on the station structure.

The preliminary load enable schedule is composed, and load class numbers are assigned, and CM/PMAD now has enough information to make load requirements projections to cover the coming crew shift. A description of these projections is given in Function 3.2.2.2 (Appendix A). The projections are forwarded to the space station manager as "electrical resource requests" broken down by load class.

Ideally, the load requirement projection "requests" of the CM/PMAD would closely match the preliminary resource allowances originally let by the space station manager. But perhaps a load originally scheduled in the ground-generated timeline failed an hour before, so the major scheduler decided not to put it in the preliminary load enable schedule at all. In this case, the CM/PMAD "request" would be less than the preliminary allowance, and the space station manager could decide to give the excess resource to another module. Alternatively, one of the scheduled loads may have started drawing more power than usual during a previous shift, and the major scheduler "knows" this. In that case, the "request" would be larger than the preliminary allowance, and the space station manager would need to decide whether to draw some resource from other modules to make up the difference.

The space station manager will review all of the load requirements projections from all of the modules and will use an empirical algorithm to decide how to allocate electrical resources among the modules. The algorithm must be empirical, because the space station manager will not have detailed information about individual loads within modules. The manager will not have detailed information because its interfaces to the modules must be kept relatively simple and generic. The interfaces must be kept relatively simple and generic so that, if one module is taken off the station and replaced with an improved version, it won't be necessary to make significant S/W and H/W changes to the other modules or to the station.

After reviewing all the load requirements projections, the space station manager will issue to each module a list of final resource allowances.

In the CM, the minor scheduler will inspect the final resource allowances and will make whatever small adjustments are necessary to the schedule of CM events covering the coming crew shift period. The function will do its work within a few minutes.

The schedule of CM activities is now ready for final crew or ground personnel modifications, if any.

If the crew or ground personnel decide to remove loads from the schedule, this can be accomplished immediately. After the crew has approved the change(s), the minor scheduler would take a few more minutes to schedule enough additional loading that the excess energy of the removed load(s) would be used efficiently. In doing this, the minor scheduler would be limited to adding loads that do not require crew monitoring or assistance. To do otherwise would require the minor scheduler to schedule a stationwide resource, i.e. crew shift time. Such a decision could not be made at the CM level.

The crew and ground personnel should be discouraged from making final load additions to the schedule; electrical energy that can be gathered and stored by the space station is limited, and close to 100% of it will have been accounted for by the original ground-generated timeline, the major scheduler, and the minor scheduler. If a last-minute load must be added, the crew or ground personnel would specify which load, when it is to be added, and how long it must be supplied with power. The minor scheduler would quickly determine which other loads would have to be shed to accommodate the new Load and would so notify the crew and ground personnel. It would shed as few loads as possible from the lowest load classes. The crew and ground personnel could decide to accept the recommendations of the minor scheduler, to control all loads manually during the time the added load would be in operation, or to cancel the original request.

1.4.4 Space Station Power and Data Bus Interface Analysis

The objective of this task is to analyze and define the CM/PMAD power and data bus interfaces to space station for each of the candidate government-furnished networks, shown in Section 1.1 as Figures 1.1-1 through 1.1-5.

1.4.4.1 CM/PMAD Interfaces to Space Station Power--The CM/PMAD is unique in the sense that it must distribute power to its own controllers. In the recommended approach, PCU, the main controller for CM/PMAD, receives its power directly from the main power bus. Connection is made on the CM main transformer output side in the case of designs one through four. PCU power for design five is from the dc primary input bus directly. The interface, in either case, will be to the local conditioning in the PCU. The PCU conditioning is required to provide transformer isolation. The local or midlevel processors receive power from the power input of the device within its command which is electrically nearest the CM power input. Similarly, lowest level device controllers interface to power at the power input of the device being controlled. Isolation is again accomplished at the local power conditioning required in the device controllers. In all cases, the local power conditioning provides isolation that allows a local grounding scheme to chassis.

Alternatively, bulk power conditioning could provide power for a CM/PMAD controller power bus. However, we recommend against this approach. The main factors in recommending against this approach are:

1) Single-point failure causes inoperability of all controllers and processors on the power bus. Providing redundant buses to match the dual redundancy approach in the candidate design would still be susceptible to a single-point failure problem in the user load center where cross strapping is used;

2) Line regulation to the required levels would be difficult;

3) Local power conditioning would still be required for isolation;

4) Flexibility after IOC is reduced.

1.4.4.2 Space Station Data Bus Interfaces--The data bus interfaces considered include the CM data management network, space station management network, and the CM/PMAD power control unit. The selection of central versus distributed power control has significant effect on power subsystem data interfacing to the Space Station data bus(es). Therefore, the approach to data bus interfacing is defined for both central and distributed approaches. A centralized system is defined as a single control processor for the entire space station power system, even though it may be termed a distributed subsystem with respect to the overall space station. A distributed approach is defined as a HADS. A HADS is further defined, for the purposes of investigating interfaces, to be a master-slave arrangement. While networking is possible in a HADS, that case is treated under the centralized system.

Interface hardware grounding constraints are discussed in Section 1.4.5, Data and Power Ground Separation Analysis.

1.4.4.2.1 Data Bus Interfaces in Distributed Approach--The master-slave arrangement in the HADS requires a data link from each slave to the master. The arrangement either must comprise a minimum of a shielded twisted wire pair (or coaxial wire, depending on the selected bus) or a fiber-optic link. In the case of electrical wire, the transmit/receive electronics must provide isolation through optical coupling, transformer isolation, or provide differential drivers or receivers with high common mode rejection to minimize possible ground loop currents. The shield, in the case of electrical wire, should be tied to chassis ground at the receiver or source, but not both. In the case of a HADS where there are multiple drops, the best approach is optical isolation or transformer isolation with the shield tied to a single chassis ground.

The candidate designs are all well suited to the HADS approach. Hierarchically above the power control unit, there is no difference in data bus interfaces for the candidate designs. The differences in the internal bus interfaces per candidate design are simply in number and location.

1.4.4.2.2 Data Bus Interfaces In Central Approach--In the centralized approach, the interfaces between the overall power system controller and the space station data management network is through a bus interface unit (BIU) to the CM network and then through a packet switch or bridge to the space station management network. The bridge and BIU are each a set of electronics with an electrical wire or fiber optic interface to the CM LAN. As in the case of the HADS interfacing, these sets of interfacing electronics must also be isolated as above.

For the central approach, the devices within the CM/PMAD are interfaced through multiplexer/demultiplexers (mux/demux) that provide the data interface to the central control processor. The mux/demux also must provide electrical isolation as above. In addition, the central approach requires additional cabling design or packaging so that the interface from a mux/demux group to the central control processor must become part of a cable assembly.

1.4.4.2.3 Software Interfaces--Software interfacing to the CM/PMAD in the central approach is an integral part of the overall software package. The software interface in the central approach, while it is easily definable and straightforward in design, it is address and data intensive at the highest level, requiring an address for each device in the CM/PMAD. Alternatively, the HADS approach, using the master-slave approach, gains the advantage of the centralized system in software interfacing while gaining the modularity and flexibility of a distributed system.

1.4.5 Data and Power Ground Separation Analysis

The local power conditioning, either ac-dc or dc-dc, of each processor and controller, as discussed in the previous section, provides transformer isolation from the main power bus. This prevents dc and low-frequency current returns through the structure. The design must provide at least 10 megohms dc isolation from the power supply inputs to the signal ground of each box. Also, signal ground must be isolated from chassis ground by at least 10 megohms. It is noted, however, that the internal grounding scheme for each processor or controller must also be designed to the electro-magnetic interference constraints given in an eventual space station electromagnetic compatibility (EMC) document.

Prevention of ground loops between the various CM/PMAD elements can be accomplished by using optical coupling data or by the transformer isolation provided in the Manchester coding approach as used in the MIL-STD 1553B data bus. A base-band standard wire bus using differential line drivers and receivers would also provide good insulation. The small ground loop current that would result would not affect signals on the bus because of the high common mode rejection ratio of the differential receiver. A base-band wire data bus would require the use of a twisted shielded pair for each signal wire of the bus. The shield would be tied to chassis ground at the receiver or the source, but not both.

1.4.6 Local Energy Storage Analysis

The local energy storage, if used, can be either primary or secondary. Applications can range from low-rate, long-duration use (such as would be used for maintenance of a safe haven condition) to providing high-rate, short-duration power (such as would be used for an uninterruptible power supply). A typical and most probable use of local energy storage is in the adaption of a common module to provide a logistics support module that must have internal electrical power during transition periods.

Seven* electrochemical couples for local energy storage are considered for functional effects on the CM/PMAD control network:

Primary

1) Silver-Zinc (AgZn);
2) Lithium-thionyl chloride ($Li/SOCl_2$);

Secondary

3) Nickel-Cadmium (NiCd);
4) Nickel-Hydrogen (NiH_2);

Secondary - molten salts

5) Sodium-Sulfur;
6) Lithium-metal sulfide;

Regenerative fuel cell

7) Hydrogen-Oxygen.

Each couple and type was analyzed for effect to the functional decomposition with results given in Table 1.4.6-1. Each of the functions checked is a delta to a CM without any local energy storage. If local energy storage is added to the CM, using the hydrogen-oxygen fuel cell or the silver-zinc battery provides the least effect to CM/PMAD automation.

*There are certainly more electrochemical couples possible that may have been considered for a local energy storage effect analysis, but these were considered representative of the functional effects of all types.

TABLE 1.4.6-1 LOCAL ENERGY STORAGE FUNCTIONAL IMPACT ANALYSIS

ENERGY SOURCE TYPE	PRIMARY		SECONDARY				FUEL CELL
COUPLE / FUNCTIONS	AgZn	LiSOCL2	NiCd	NiH2	NaS	LiMS	H-O
CHARGE MAINTENANCE	●		●	●	●	●	
RECHARGE CONTROL			●	●	●	●	
DISCHARGE CONTROL (CURRENT LIMIT)	●	●	●	●	●	●	●
DISCHARGE CONTROL (OVER DISCHARGE)	●	●	●	●	●	●	
STORAGE ENERGY MANAGEMENT			●	●	●	●	●
RECONDITIONING MANAGEMENT			●	●	●	●	
FAULT DETECTION	●	●	●	●	●	●	●
SINGLE POINT STATUS				●			●
STATE OF CHARGE COMPLEX	●	●	●		●	●	
HAZARDOUS MATERIAL DETECTION		●			●	●	
TREND ANALYSIS	●	●	●	●	●	●	●
REGENERATION MANAGEMENT							●
THERMAL MANAGEMENT	●	●	●	●	●	●	●

1.4.7 Crew Interface Analysis

1.4.7.1 Introduction

1.4.7.1.1 Purpose--It is intended that the CM/PMAD task be as automated as is practical. It is also recognized that some human (crew or ground) monitoring of and intervention in the task will be necessary. This section examines possible types of interfaces between CM/PMAD and the crew and recommends those types considered practical. Possible interfaces between CM/PMAD and ground personnel are not considered here.

1.4.7.1.2 Scope--At this writing, much remains to be resolved in the larger details of the designs of the CM power network and of the CM data network. Much also needs to be resolved in defining the elements of the mission of the space station. Given these facts, and observing the rapid development in microcomputer technology, it is impossible to predict with confidence what interface types, existing or projected, would be appropriate for the entire operational life of the station, which is estimated to be between 10 and 30 years. And so, unless otherwise noted, types of interfaces recommended in this report will be assumed to apply only to common modules within the IOC of the space station.

1.4.7.1.3 Definitions--This paragraph defines several short-hand terms that are used for convenience in this section.

The term "CM computer" means any part of or all of the distributed information processing in the CM, whether S/W (including expert systems) or logic H/W. Similarly, the term "space station main computer" is defined as any part of or all of the distributed information processing in the space station outside of the common module.

The "CM/PMAD subsystem" consists of the CM electrical power network and that portion of the "CM computer" that is devoted to the CM/PMAD task.

1.4.7.2 Crew Interfaces Relevant to CM/PMAD Functions--In Appendix B, Crew Interfaces Relevant to Functions of the CM/PMAD Task, each function of the CM/PMAD task is examined to determine which functions require interfaces between CM/PMAD and the Crew. For easy cross-referencing, Appendix B has the same paragraph arrangement as Appendix A, Functional Breakdown of the CM/PMAD Task. (Appendix A presents a detailed description of the CM/PMAD task and its functions.)

In Appendix B, the two major CM/PMAD displays are mentioned, but not described in detail: the load enable schedule display, and the CM/PMAD block diagram display. These displays are described below in paragraphs 1.4.7.3.1.2.1 and 1.4.7.3.1.2.2, respectively.

1.4.7.3 Conclusions

1.4.7.3.1 Recommended Interfaces between CM/PMAD and the Crew--The following recommended interfaces between CM/PMAD and the crew are concluded to be practical and cost-efficient for the CM/PMAD task only. They are not intended to preclude the development of other technologies thought useful to support other CM subsystems. It is stressed that these recommendations apply only to the IOC.

1.4.7.3.1.1 Hardware

1.4.7.3.1.1.1 CM Computer Console--Some sort of CM computer console should be available to allow the crew to monitor and control elements not only of CM/PMAD but of other CM subsystems as well.

1.4.7.3.1.1.1.1 Video Screens--It is recommended that the CM computer console have at least two identical video screens of 14-in. diagonal measurement or larger to support crew intervention in the CM/PMAD task.

A touch-screen capability is recommended for each video screen. This would allow a crew member easy selection of interactive control options on low- or medium-density displays.

Color graphics capability is recommended. Because two-dimensional graphics should be fully capable of supporting the CM/PMAD task (paragraph 1.4.7.3.2.2.2), color capability need not be sophisticated. Eight colors including red, yellow, green, blue, black, and white should be sufficient.

A screen resolution of 640-by-400 pixels or better is recommended to allow display of complex block diagrams.

1.4.7.3.1.1.1.2 Keyboard--The CM computer console should include a keyboard. A keyboard would allow the crew to transmit precise commands of considerable complexity to the CM/PMAD task. The keyboard keys should include all letters, numbers, and punctuation marks found in a standard QWERTY arrangement. All American astronauts having a technical education sufficient to control CM/PMAD will be familiar with this arrangement. Other key arrangements said to be more efficient than the QWERTY system are not nearly as well known and would, therefore, require special training of many of the crew members.

An accessory numerical keypad should be included on the keyboard. This keypad would feature the numbers 0 through 9 and + and - sign keys in an adding machine cluster arrangement. Most of these keys should have alternate cursor control functions. In some operating modes (such as changing a control quantity on a block diagram), these keys would have numerical entry functions only. In other modes (such as modifying a tabular display for control purposes, or in editing), these keys would have cursor control functions only.

It is likely that special function keys will have some use for controlling the CM/PMAD task. Without a detailed crew interface S/W design, it is difficult to predict how many special function keys will be useful, but it is recommended that at least five be included on the keyboard.

1.4.7.3.1.1.1.3 Trackball Assembly--In a high-density graphic display (such as the CM/PMAD block diagram display described in paragraph 1.4.7.3.1.2.2), touch-screen control may not be practical simply because a crew member's finger is too big to place the control cursor accurately. For accurate control of high-density displays, a trackball assembly (the ball plus a push-button switch) is recommended. An appropriately designed trackball assembly need not weigh more than a mouse interface. A mouse requires an open, flat surface against which it can be pushed, while a trackball does not. Also, in weightlessness, a mouse would require a bracket in which it could be stored when not in use; a trackball assembly, being permanently mounted, would have no such problem. Furthermore, a mouse would trail a cable that could become entangled in the close working environment of the CM; a trackball assembly could be mounted at the CM computer console without using an exposed cable.

Where should the trackball assembly be mounted? If it were mounted on one side of the CM computer keyboard, it would be inconvenient for crew members of opposite-handedness to operate. Mounting the ball on the axis of symmetry of the console would not be as space-efficient as a side mount, but that arrangement would not require special training for some crew members. It is recommended that the assembly be mounted on the axis of symmetry of the CM computer console, and between the keyboard and main video screen. In this position, the assembly would not be in the way of a crew member using both hands to work the keyboard. It is further recommended that the trackball assembly be built onto a pivoted platform that can be rotated left or right in small angular steps (much like a rotary switch) to any angle convenient to the user.

1.4.7.3.1.1.2 Dead-Face Switch--A double-throw dead-face switch should be provided that would enable a crew member to cut all electrical power in the CM, except emergency lighting and other emergency subsystems. The switch should be absolutely fail-safe.

The purpose of the dead-face switch would be three-fold: (1) it would be the crew's last line of defense in case of an electrical fire; (2) it would be a rapid and sure means of disconnecting runaway equipment in the CM which threatened immediate crew injury or equipment damage; and (3) it would be a convenient means of safeguarding CM circuitry while power cables were being connected to or disconnected from the CM.

The dead-face switch and its surroundings should be distinctly colored and adequately lit by regular or emergency lighting circuits.

When used to open the dead-face circuit, the operation of the dead-face switch should be intuitive and easily accomplished from any angle in weightlessness. When used to open or to close the dead-face circuit, the operation of the switch should not require foot restraints.

It would be permissible for the CM/PMAD subsystem also to control the dead-face circuit by digital command. However, the design should be such that manual opening of the dead-face switch must inhibit all computer commands to the dead-face circuit.

1.4.7.3.1.2 Software

1.4.7.3.1.2.1 Load Enable Schedule Display--This should be the display/control most often used by the crew to intervene in the CM/PMAD task. A crew member need not be specially trained in the CM/PMAD subsystem to use this display and its controls effectively.

The load enable schedule display would allow the crew visibility into and limited control over the on-board scheduling function of CM/PMAD. Before proceeding further in this section, we recommend that the reader review Appendix A, Function 3.2.2, On-Board Scheduling. There, the reader will find definitions and explanations of terms used in the discussion that follows.

In the absence of human intervention, the formal script for CM/PMAD activities will be the baseline load enable schedule (BLES). The BLES will be a compact, coded file of Load enable/disable instructions suitable for direct use by CM/PMAD S/W. As such, it will not be readily understandable by people. The crew, to understand and control the BLES, will need a user-friendly interface tool: the load enable schedule display (LESD).

The LESD would show the BLES in chart form and with English labels. Information shown for each Load would include the following:

(1) Load name (such as X-ray telescope);

(2) S/W code number of the RPC that directly feeds that Load;

(3) Load class number;

(4) Load priority number (if one is presently assigned);

(5) Indications showing what time(s) the Load is scheduled "enabled" and what time(s) it is scheduled "disabled,"

Loads would be sorted on the chart by load class with the higher classes toward the top and the lower classes toward the bottom. Because there would generally be more loads scheduled than could conveniently be shown on a single video screen, a means would be provided to allow the user to scroll the chart up and down.

At user option, the LESD would show any one of the following: (1) the present BLES, (2) a BLES evolved by S/W for the next scheduled period, or (3) a BLES that was used in the past.

If showing the present BLES, the LESD would include a prominent, moving line that would divide the chart into two parts: (1) scheduled events that have already happened, and (2) scheduled events that have yet to happen. If any Loads originally scheduled in the present BLES have been disabled by load shedding (Appendix A, Function 3.2.3), those Loads would be flagged with a distinctive color. Also, if any Loads originally scheduled in the present BLES are recommended to be descheduled by minor scheduling (Appendix A, Function 3.2.2.3) in response to a Load addition by the crew, those Loads would be flagged with a distinctive color.

As mentioned earlier, the LESD not only would show the form of the baseline load enable schedule, but also would provide controls for a crew member to modify that schedule. At one end or corner of the LESD, there should be a special window through which a crew member and the CM/PMAD could communicate interactively via the keyboard. CM/PMAD would use this window to advise the crew member of display options and to prompt the crew member for the information necessary to change the BLES. The crew member would reply with keyboard entries and touch-screen entries, as appropriate.

Keyboard communication between a crew member and the LESD should be, as much as possible, in natural language. With this aid, a crew member not specially trained in the CM/PMAD Subsystem could still control it effectively through the LESD. For example, the S/W should be able to recognize that "enable the Upper Atmosphere GCMS", "turn on the chromatograph", and "energize the gas chromatography experiment" probably all mean: "connect electrical power to the input of the upper atmosphere gas chromatograph and mass spectrometer". If there is more than one type of GCMS, the S/W should be able to ask: "Do you mean the upper atmosphere GCMS, or the heavy ion GCMS?".

1.4.7.3.1.2.2 CM/PMAD Block Diagram Display--The CM/PMAD block diagram Display would show the general state of the subsystem H/W. At crew option, the display could also be used as a control tool. This option would invoke a module of emergency S/W that would bypass much of the automatic CM/PMAD S/W, allowing the crew direct control of subsystem H/W at the individual RPC level.

The CM/PMAD block diagram display would be a video display that could be called up by a crew member at the CM computer console. The display would show all of or any selected part of the CM/PMAD subsystem hardware in block diagram form. The diagram would be a line drawing representing elements of the subsystem. Elements that were presently connected to electrical power would be drawn in a distinctive color (perhaps white); those elements not connected to power would be drawn in black.

The finest available resolution of the CM/PMAD block diagram display would be to a level of detail sufficient: (1) to allow the crew to intelligently control all interfaced H/W elements of CM/PMAD, and (2) to effectively support the crew (should the automatic S/W be unable to do so) in preventive maintenance and repair operations. At all levels of resolution any element for which a fault had been detected and isolated by CM/PMAD S/W would be flagged in red. Density of the display permitting, notes would be shown nearby briefly describing the nature of the fault. If the faulty element were still connected to electrical power, the red flag would flash; if the element had been disconnected, the flag would dim and would not flash. CM/PMAD elements that had been predicted to fail would be displayed using a strategy similar to that just described, except that elements predicted to fail would be flagged in yellow.

When first called up, the display would show the entire CM/PMAD subsystem. At this point, the general detail of the display would necessarily be simplified to show the whole subsystem. In this simplified form, three-phase elements would be shown with one line or node representing all three phases plus the neutral path and the grounding path. The display would show the sensed switching states of RBIs or RCCBs connecting the major buses to one another and connecting major buses to the inputs of the load centers For three-phase devices, the switching states would be assumed to apply equally to all three phases, i.e., either all three phases would be connected, or all three phases would be disconnected. The display would also show major bus voltages and major feeder currents. For three-phase elements, the voltage and current values displayed would be the averages for all three phases.

If more detail about a particular region of the display were needed, a crew member could position a window over the area in question and zoom-in for a closer look. As the crew member zoomed-in, more details and measurements would be displayed for elements in the window; details would be added in order of importance. The display would show the part number of and the schematic boundary of each H/W module in CM/PMAD. It would also show the part number of each interconnecting cable between modules. The module and cable part numbers would be used by the crew to locate actual CM/PMAD devices during preventive maintenance and repair operations (actual modules and cables should be clearly marked or labeled). When the crew member had zoomed-in toward a particular CM/PMAD element (an RPC, for example) to the maximum limit of resolution, the display would show for that element every phase-independent quantity that was presently measured by sensors at that element or that was computed for that element based on the sensor measurements. At maximum resolution, three-phase devices would be represented by one line or node representing all three phases, plus one line or node representing the neutral path, and one line or node representing the grounding path. Where it is deemed appropriate and practical, some three-phase devices would be shown at maximum resolution with a separate line or node for each phase. At that level of detail, sensor measured quantities would be shown for each phase separately instead of averaging for all three phases.

Usually, numerical and logical quantities shown on the CM/PMAD block diagram display would be the values most recently measured by sensors or computed from sensor measurements. The crew would have the option, however, to view CM/PMAD quantities as they had been measured just prior to the most recent trip of an RPC or RCCB. The crew could use this information to check whether the trip was appropriate.

At crew option, the CM/PMAD block diagram display could be used as a control tool. This option would invoke a module of emergency S/W that would bypass much of the automatic CM/PMAD S/W, allowing the crew direct control of subsystem H/W at the individual RPC level. The option would allow the crew direct control of all H/W elements in the CM/PMAD power network, and would allow coarse control of many H/W elements in the CM/PMAD data network.

There should be two general types of control: (1) control of bilevel states (on/off, connect/disconnect, enable/disable, etc.), and (2) control of states with which numbers are associated (trip level of RPC, regulator voltage adjustment, mode number, etc). For three-phase devices, it is recommended that any applied control should apply to all phases simultaneously and equally (e.g., a connect command should connect all three phases at the same time).

It is specifically recommended that any control that the crew applies to a given three-phase device should apply simultaneously and equally to all phases of the device. If crew control were not limited in this way, loading imbalances could occur which could open RPC's or similar devices in the power network. Such loading imbalances could conceivably damage elements of the CM/PMAD Subsystem.

Bilevel states would be controlled by the crew in the following manner. First, a crew member would zoom-in (if necessary) to the part of the CM/PMAD block diagram display showing the state to be controlled. Second, the crew member would position the cursor at this indication using the trackball assembly. Third, the crew member would press the push-button switch on the trackball assembly to signal the computer of his or her desire to change to the other state. At this point, the display would show the other state, but flagged in a distinctive color to indicate that this was a "commanded" and not yet an "actual" state. When the color flag disappeared, this would indicate that the "commanded" state had become an "actual" state of the device. At any time during this sequence, the crew member should be able to cancel the state change by entering a single convenient input.

Control states with which numbers are associated would be controlled by the crew in the following manner. First, a crew member would zoom-in (if necessary) to the part of the CM/PMAD block diagram display showing the value to be controlled. Second, the crew member would position the cursor at this indication using the trackball assembly. Third, the crew member would press the push-button switch on the trackball assembly to signal the computer that this number is now to be changed. At this point, the display would flag the number in a distinctive color to indicate readiness to accept a new value. The crew member would then enter the new value for the number via the keyboard. This number, too, would be flagged in a distinctive color to indicate that this is a "commanded" value and not yet an "actual" value. When the color flag disappeared, this would indicate that the "commanded" value had become an "actual" state of the device. At any time during this sequence, the crew member should be able to cancel the number change by entering a single, convenient input.

1.4.7.3.1.2.3 Other Video Displays/Controls--Each of the following displays would also double as a control tool for the crew to use. Descriptions of the displays may be found in Appendix B under the function number cited.

* Load Priority List Display (Function 3.2.4.4).

* Interactive Self-Test Displays (Function 3.3.2.1).

* Interactive S/W Comparison Displays (Function 3.3.2.1).

Each of the following would be for display purposes only. Descriptions of the displays may be found in Appendix B under the function number cited.

* History Records Displays (Function 3.3.1.5).

* Fault Report Displays (Function 3.3.2.4).

* Audio alarm indicating significant CM/PMAD fault (Function 3.3.2.4).

1.4.7.3.2 Other Technologies that Were Considered--Described here are other potential crew interface technologies that were considered while preparing this section, but which are not specifically recommended. The reasons why they are not recommended are listed.

1.4.7.3.2.1 Hardware: Control via Mouse--A mouse can be used to perform the same valuable services as a trackball assembly (paragraph 1.4.7.3.1.1.1.3). Also, a mouse can be used with equal facility by left- or right-handed crew members. Unfortunately, a mouse requires an open, flat surface against which it can be pushed. With limited space available, and considering weightlessness, open, flat surfaces will have little other use in the CM. A mouse also trails a cable that can become entangled in a close working environment. Furthermore, in weightlessness, a mouse would require a bracket where it could be stored when not in use. It seems that a trackball assembly would be a better choice than a mouse to help the crew to monitor or control high-density displays in CM/PMAD.

1.4.7.3.2.2 Software

1.4.7.3.2.2.1 CM/PMAD Schematic Display--This would have been a display similar to the CM/PMAD block diagram display described above in paragraph 1.4.7.3.1.2.2, except that it would have shown far more detail. It would have been similar to a detailed electrical schematic in form, showing all three phases of three-phase devices (even showing connector and pin numbers). In addition, it would have shown every sensor measurement of all elements in the power network. Finally, it would have shown details of faults that had been detected and isolated in S/W, and would have shown them on a phase-by-phase basis. A CM/PMAD schematic display is specifically not recommended, because it would be inappropriate for it to do any of these. A discussion follows.

Detailed electrical schematics are commonly used as aids for trouble shooting of the system depicted. Trouble shooting is done so that preventive maintenance or repair may be performed. It is practical for the crew to do preventive maintenance or repair, but only to a limited extent.

Coarse maintenance and repair (specifically, device replacement) could be done at the modular level, down to an individual RPC or to a particular electrical cable. The level of detail necessary for this type of operation would be provided by the recommended CM/PMAD block diagram display.

Finer maintenance or repair could be done by replacing an individual card within a multicard module. This could be done in weightlessness with simple tools, but how would the crew know which card to replace? The level of sensor measurement now contemplated does not go lower than module level, so Fault Management (Function 3.3.2, Appendix A) would not be able to locate a particular faulty card in a module. It is conceivable that the crew could use detachable test equipment to find a faulty card, but this would mean either removing and disassembling the module and then performing manual tests, or connecting test equipment to special connectors on the module case. Neither of these operations is recommended; the first, because of the crew time involved; the second, because of the weight of the special connectors and their associated wiring.

Because maintenance or repair at a level finer than that of individual modules doesn't seem practical, and because detail sufficient to support module-level maintenance would already be available in the CM/PMAD block diagram display, a CM/PMAD schematic display cannot be justified for maintenance or repair.

If the CM/PMAD schematic display were made sufficiently detailed to show all phases of the three-phase devices, it could be made sufficiently detailed to display every sensor measurement taken in the power network. This would be a huge amount of data. Meaningful correlation of this data would be a difficult task even for a person specially trained in the CM/PMAD subsystem. Because most crews are expected to contain no such experts, it does not seem useful to design a CM/PMAD schematic display to display all sensor data.

If the CM/PMAD schematic display were made sufficiently detailed to show all phases of the three-phase devices, it could be made sufficiently detailed to display comprehensive fault data, flagged in color, on a phase-by-phase basis. This would be useful if a crew member were allowed control over individual phases; he or she could simply recalibrate or disconnect the faulty phase. However, it is specifically recommended that any control the crew could apply to a three-phase device should apply to all phases simultaneously and equally. If control were not limited in this way, loading imbalances would be possible that could open RPCs or similar devices in the power network. Such loading imbalances could conceivably damage elements of the CM/PMAD Subsystem. Because, under this limitation, a fault on any single phase could only be compensated by disconnecting the entire device, and because sufficient display detail to decide to do this would be available on the recommended CM/PMAD block diagram display, it is not necessary to have a CM/PMAD schematic display to show detailed fault data.

In conclusion, there seems to be no practical use for a CM/PMAD schematic display to support crew interface with the CM/PMAD task.

1.4.7.3.2.2.2 Perspective Graphics and Three-Dimensional Graphics—The displays recommended for crew interface with the CM/PMAD task are of four general types: (1) chart form, (2) block diagram form, (3) text form, and (4) measurement-versus-time graphic form. Perspective graphics and three-dimensional graphics have been used to augment displays of mechanical layouts or of mechanical interaction. In none of the four general types of displays mentioned will there be any mechanical layout information or mechanical interaction information to be augmented.

It is possible that perspective graphics or three-dimensional graphics could show the locations of CM/PMAD elements needing replacement. However, any complete servicing instructions would also have to give step-by-step directions on how to remove access covers, how to safely disconnect modules from the subsystem, etc. It would seem more practical to simply put numbered labels on all access covers, modules, cables, etc, and refer to these in any servicing instructions. The servicing instructions would be generated under Preventive Maintenance Scheduling (Appendix A, Function 3.3.1.2) or under Fault Logging (Appendix A, Function 3.3.2.4).

In conclusion, there seem to be no useful roles for perspective graphics or three-dimensional graphics in a crew interface with CM/PMAD.

1.4.7.3.2.2.3 Voice Recognition/Speech Synthesis--The idea was examined that the CM computer console should have provisions for voice recognition and speech synthesis. This equipment would have allowed a crew member located in any part of the CM to conduct a command conversation with CM/PMAD. One serious difficulty exists with this approach.

The difficulty is that most of the replies given by S/W in answer to crew commands would have to be fairly detailed, making it difficult for the crew member to remember all his options while judging which was best. For example, suppose the crew member wanted to change the baseline load enable schedule on the load enable schedule display (paragraph 1.4.7.3.1.2.1). This would be the simplest common crew input to the CM/PMAD task. The crew member could keep his or her modification request fairly straightforward:

"Enable the UV Telescope at 06:00:00, U.T. for 25 minutes."

The S/W might reply:

"I can do that. However, to support your Load, I would have to disable the following Loads:

(1) The upper atmosphere GCMS at 06:02:10, which would be 14 minutes early after a 3-hour and 46-minute run;

(2) The antifungus spraying sequence at 06:03:40, which would be 2 hours and 6 minutes early after a 54-minute run; and finally;

(3) The Quenton Medical electrophoresis experiment at 06:15:50, which would be 1 hour and 20 minutes early after a 14-hour run.

"After your Load was disabled, I could reconnect only the antifungus spraying sequence at 06:25:00. Is all of this acceptable?"

This would be a fairly typical reply; replies involving many more than three recommended disabled Loads are possible. The complexity of these replies could be done away with if CM/PMAD reserved an "electrical resources pad" big enough that, if an unscheduled Load was added, other Loads would not have to be shed. Such a pad was used on Skylab. On space station, however, we recommend using modern software to keep our pad as close to zero as possible; to waste as little of the available energy as possible.

The only way most people could efficiently evaluate a S/W reply such as the one above would be to view it in graphic form, perhaps as a timeline chart with horizontal bars plotted on a time grid showing when the various Loads of the command and reply would be enabled and disabled. Without such a display, the crew member would have to ask the S/W to repeat its reply one or more times while considering whether the S/W recommendation was appropriate. An appropriate graphic display showing the kind of S/W reply described above would be available in the recommended load enable schedule display. To view this display while making his or her decision, the crew member would have to come to the CM computer console. If he or she has to do this, the prime advantage of voice command (control of CM/PMAD from anywhere in the CM) has been neutralized.

A second illustration of the complex reply problem may be shown by supposing that voice were to be used to control the other major crew interface, the CM/PMAD block diagram display (paragraph 1.4.7.3.1.2.2). The information on that display would be densely packed and highly symbolic. Anyone who has tried to describe details of a complex block diagram to another person while relying on language alone soon recognizes the difficulty in doing so. Usually, the person doing the describing must resort to a blackboard or similar aid (such as a video screen).

The other recommended displays were examined for possible voice control, but the problem was the same: unwieldy amounts of data would have to be remembered by a commanding crew member while judging whether a given S/W reply was appropriate.

In conclusion, voice command of the CM/PMAD task seems to be too cumbersome to be practical.

1.4.8 Autonomy Sensing Techniques and Requirements

To automate common module power management and distribution, the control computer or computers will need to sense the quantity of power used in different parts of the network and the quality of that power. In addition, it will need to sense the status of each remote power controller, and temperature measurements will prove useful in diagnosing malfunctions.

Although any power system can be expected to require sensing these factors, the type of power (ac or dc, single-phase or polyphase, etc) will determine the physical quantities to be measured and the appropriate techniques. The discussion that follows examines three different types of power—dc, 400 Hz three-phase ac, and 20 kHz single-phase ac—because at the time of the study no firm decision had been made about the type of power that will be used in the common module.

1.4.8.1 Quantity of Power—For a dc system, the quantity of power can be measured by sensing voltage and current and multiplying these quantities in the control computer. Voltage is readily sensed with a voltage divider, which reduces the voltage to a level appropriate for an analog-to-digital converter and provides current limiting in the event of a failure in the instrumentation electronics.

Direct current is more difficult to sense. Although a current-sensing resistor (meter shunt) in series with the line could be used, this approach has two significant drawbacks. First, if dc is used, it will likely be 270 volts. The circuitry required to sense a few hundredths of a volt of drop across the resistor at 270 volts above ground is more complex than alternative approaches. Second, the power dissipation in the current-sensing resistor may be as high as three watts in a feeder and, therefore, may contribute unnecessarily to packaging problems.

Unfortunately, the alternatives have undesirable features also. First, Hall-effect devices can be used to measure current by measuring the magnetic flux density in the core of a one-turn inductor placed in series with the line. A similar approach can be used if the Hall-effect device is replaced with a resistor made of magnetoresistive material.

For accuracy with this approach, the flux-density measurement is not used directly; nonlinearities, hysteresis, and sensor sensitivity drift severely degrade accuracy. Instead, the signal is amplified and used to drive a second (feedback) winding to drive flux density to zero. The current in this feedback winding is forced to be proportional to the sensed current, and a voltage proportional to it is readily obtained as the sensor output signal. The feedback approach is not effective with magnetoresistive sensors because they are not sensitive to polarity.

A dc transformer is another alternative. This type of sensor requires an oscillator and can be expected to be more complex than the Hall-effect approach.

Despite the complexity of the feedback from the Hall-sensor approach, it still appears better for the application than the alternatives. It has the additional advantages of proven technology and isolation. The technique is commonly used for oscilloscope current probes.

In summary, we recommend a voltage divider for dc voltage sensing and a feedback Hall-sensor approach for dc current sensing.

For ac systems, measuring power quantity becomes more complex, because the power factor must be considered. It is possible to build a sensor with an output proportional to power. For example, an analog multiplier could be driven from a voltage divider on one input and from a current transformer on the other input. The output of the multiplier could then be low-pass filtered to provide the desired signal. A similar effect could be achieved with a Hall-effect device. The bias current would be provided by simply connecting it to the line voltage through a series resistance. The magnetic field would be provided by placing the device in a gapped inductor core as with the current-sensing scheme described previously. Because a Hall device's output is proportional to the product of bias current and magnetic flux density, this arrangement would make the output proportional to power.

We do not recommend these techniques for two reasons. First, we doubt that the desired accuracy would be achieved, and second, we believe the automation computer will need power factor information for other purposes.

We recommend measuring voltage, current, and the phase angle between them instead. The computer can calculate power from these quantities.

Voltage is easily measured with a voltage divider as with dc. Integrated circuits are readily available to convert the reduced-amplitude ac signal to a dc signal proportional to its RMS value.

Current measurements can be handled similarly, except that a current transformer is used in place of a voltage divider.

Power factor is most readily measured indirectly by measuring the phase angle between voltage and current. This approach has the added benefit of determining whether current is leading or lagging voltage. One simple technique is to convert sinusoidal voltages and current signals—from the same voltage divider and current transformer described previously if convenient—to logic-level square waves. This requires two comparators, one for each signal. The low-to-high transition of the signal representing voltage can start a counter that counts pulses from an oscillator. The low-to-high transition of the current signal latches the count in a register. The second appearance of the voltage signal's low-to-high transition freezes the count in the counter. The count in the counter is now proportional to the period of the voltage waveform, and frequency can be calculated by simply computing its reciprocal. The ratio of the register count to counter count is proportional to the phase difference, and power factor is readily computed from this.

Major benefits of the technique are that frequency measurement is a byproduct, that the component count is small, and that the measurement is produced in digital form directly, so no analog-to-digital conversion is required.

Although all these techniques are proven for low frequencies, we have not seen them applied to 20-kHz power. This frequency approaches the performance limits of some commonly available true-RMS-to-dc converters, and careful design would be required to achieve accuracy in phase angle measurements. Nevertheless, we see no fundamental problem in using the same techniques at 400 Hz or 20 kHz.

1.4.8.2 Quality of Power--To detect problems and diagnose their solution, the automation computer(s) must know something about the quality of the power. For a dc system, the minimum information needed is the voltage. In contrast, an ac system would also require measurements of power factor and perhaps, at the point where power enters the common module, also frequency and phase angles between the phases of a three-phase system. Techniques suitable for all these measurements have been discussed.

Another measurement that could be of benefit is noise. However, we do not see noise measurements being useful enough to warrant the cost of the instrumentation.

1.4.8.3 Status of Remote Power Controllers--To evaluate problems, the control computer(s) must know the commanded state of each remote power controller and its actual state, because a controller may fail to respond to a command or may self-diagnose a fault. We believe that a minimum of six status messages should be provided for:

1) "Normal and closed,"

2) "Normal and open,"

3) "Malfunctioning and shorted,"

4) "Malfunctioning and open,"

5) "Other malfunction,"

6) "Tripped."

It may also be useful to distinguish between a slow trip from a small overload and a fast trip from a gross overload that required current limiting.

1.4.8.4 Temperature--Temperature measurements will prove useful in detecting failure, because overheating is an easily measured symptom of a number of potential failure modes. Temperature can be measured many ways--thermistor, thermocouple, semiconductor junction, platinum wire, etc--but thermistors are inexpensive and easily provide sufficient accuracy for this purpose.

1.5 FIBER OPTICS EVALUATION

We do not recommend using fiber optics internal to CM/PMAD unless it becomes required for high data rate performance or electrical isolation. The internal data bus requirements in the CM/PMAD current architectural control approach do not overcome the operational issues involved in using fiber optics. While the performance characteristics of a fiber optic bus are very attractive, operational issues such as onboard maintenance, environmental effects, total dose radiation effects, outgassing, and connector mating lessen the attractiveness.

In performance characteristics, optical cables have significantly smaller mass and bulk than the coaxial cable or twisted wire pairs required by electrical networks, and being nonconductive, they provide complete electrical isolation between all bus units. They are insensitive to, and do not produce, electromagnetic interference. Electrical networks with throughput sufficient for power control applications include Ethernet and MIL-STD-1553. Each of these networks is a bus having a single tapped cable (coaxial for Ethernet and twisted wire pair for MIL-STD-1553). Integrated bus interface units are available for each.

A bus topology is possible with fiber optics networks, but insertion losses in taps limit the total number of nodes to a handful (10-15 typical). Fiber optics usually use a star configuration where transmitter power is divided equally among all of the receivers, or a ring where information rotates around and is regenerated at each node. A star network uses a passive optical coupler, and thus, may have simple transceiver electronics. A failure in one node will not disable the entire network. Its disadvantages are that the coupler represents a single point of failure, and that the amount of optical fiber cable required to reach each node from the coupler location can be prohibitive. If there is a large number of nodes, it may be necessary to use laser sources to assure an adequate optical power margin. The ring is essentially a loop of point-to-point links. Because each transmitter sends to only one receiver, large power margins may be maintained with inexpensive LED sources. The ring requires less cabling than the star, as fibers are placed only between nodes. Disadvantages of the ring configuration are that messages must be regenerated at each node and then removed from the network after reception; and that a node failure will open the loop. Techniques for eliminating the latter problem include optical bypass devices and dual-ring architectures that provide redundant signal paths.

Commercial networks using both star and ring configurations exist, and some, such as the Siecor LAN (star network) emulate Ethernet and other popular electrical protocols. One promising candidate where extremely high data rates are required, is the fiber distributed data interface (FDDI), a 100-megabit per second ring-based LAN with ANSI standardization, and which is soon expected to have integrated electronics available.

1.6 DATA BUS AND MICROPROCESSOR SELECTION

The objective of this task is to determine the maximum data rates, the most appropriate microprocessor(s) for the CM/PMAD system, and the data bus to be used. In particular, we evaluated the use of the MC 68000 microprocessor. Critical issues that influenced the selection included the amount of distributed automation, functions to be automated, power system sensing techniques, total system data, computer language selection, and the overall approach or architecture for CM/PMAD automation.

1.6.1 Maximum Data Rates

Functions of Section 1.2 were reviewed to identify those functions within PMAD network control that would be factors in calculating maximum data rates within CM/PMAD. Specifically, functions of interest are those functions that require significant data transfer. Distribution management was not considered a key with respect to required data rates. Distribution management, in both the centralized (one power system controller for all space station) and distributed (distributed processing to the CM/PMAD level) is table driven; and therefore, requires transmission of only that data affecting a portion of the relevant table. However, for a centralized power system, load management and health management can require all PMAD system data to be transmitted to the decisionmaker. It is noted that a distributed approach does not necessarily require all CM/PMAD data to be transmitted for the load management function.

Having found that there is reason to transmit all CM/PMAD data, it is key to define the minimum time required for reaction to an event. The question to answer in the definition is: "How much time can CM/PMAD take before it must correct an anomaly?" Inspecting the types of loads reveals three categories exist with respect to required speed of reaction to loss of power or power quality outside of specifications but within the remote power controller's set points.

Category one loads cannot have any power interruptions, e.g, computers, etc. Category one loads must have additional hardware, capacitor bank or battery driven, to provide uninterruptible power during the reconfiguration period.

Category two loads can have interrupted power, but only for the tens of milliseconds range. Category two loads include category one loads where a capacitive approach is used for the uninterruptible power supply (UPS). These loads must be hardware handled using an analog-sensing circuit on the input and a lock-out switching scheme to the redundant bus. As shown in subsequent paragraphs, it is possible to handle category two loads with software interaction only by using a hierarchically arranged distributed system down to a manageable number of loads.

Category three loads include those loads that may have input power interrupted and require service within several seconds. Many loads fall into this category, even though they may be critical loads, such as the environmental control and life support subsystem and the thermal subsystem. Category three loads are the only software serviceable loads for both space station power subsystem centralized and CM/PMAD centralized approaches.

The worst-case approach with respect to category three loads is taken to determine the required bus rates. The following assumptions are made in the analysis:

1) All sensor data are required at the decisionmaker.

2) There are six modules on space station (growth).

3) There is a possibility of 28 load centers, each with 20 possible loads in a fully loaded module.

4) On the fully loaded module and the CM, there are 41 subsystem loads, totaling two load centers.

5) There are seven load centers, each with 20 possible loads, in a CM.

6) There is one current measurement per phase per load.

7) There is one voltage measurement per phase per load center.

8) Temperature, power factor, etc, do not require rapid measurement.

9) A measurement word comprises 16 bits.

10) A three-phase system requires a 30% data increase for commands and occasional data.

11) A dc system requires a 100% data increase for commands and occasional data.

12) Required processing time is 50 microseconds per word including memory stuffing overhead. (Derived from 100 lines assembly for approximately 500 machine cycles at a microprocessor clock rate of 10 mHz.)

13) Intelligent fault detection and isolation is a background task.

14) No DMA controller for communication.

Using these assumptions, the required data rate is plotted versus required reaction time. Using parameters, we calculated the data rate for both a fully loaded module baseline, plotted in Figure 1.6.1-1, and a CM baseline, plotted in Figure 1.6.1-2, each with the following:

1) Three-phase, six-module, space station power subsystem centralized;

2) Dc, six-module, space station power subsystem centralized;

3) Three-phase, single-module (distributed to at least the module);

4) Dc, single-module (distributed to at least the module).

The equation for the curves is as follows:

$$Dr = \frac{Dt}{Tr - Tppb * Dt}, \text{ where:}$$

Dr = Data rate (bits/second)

Dt = Total data (bits)

Tr = Reaction Time (seconds)

Tppb = Processing time per bit (seconds/bit)

Interpreting the curves, the "steep" portion reflects processing time limited, while the "dotted" lines represent transmission time limitations. Decreasing the processing time "moves" the "steep" portions to the left, while increasing the processing time significantly increases the reaction time (at the "knee" of the curves). The effect of changing the total data handled is shown by comparing the four curves. The curves show that for both CM and fully loaded module approaches, and each using a distributed system (at least to the module level), a 1-megahertz data bus is adequate (with the listed assumptions) for about 100 milliseconds of reaction time. This would be satisfactory for all category three loads. •

Using a HADS significantly decreases required data bus rates and reaction times. Local decisionmaking at the load center level results in the data for 20 loads required to be transmitted. A level up to the CM/PMAD controller is roughly the same with each load center considered as a load. The data rates in the HADS approach are plotted (Fig. 1.6.1-3) for comparison. As expected, the HADS approach is far superior to other approaches with respect to reaction time. The plot also shows, with a HADS approach, category two loads can be software-serviced. Additionally, using a 1-megahertz data bus, available processing time is increased. For instance, with HADS and with a 1-megahertz bus and at 100-milliseconds required reaction time, approximately 98.7 milliseconds is available as processing time, even in the three-phase system.

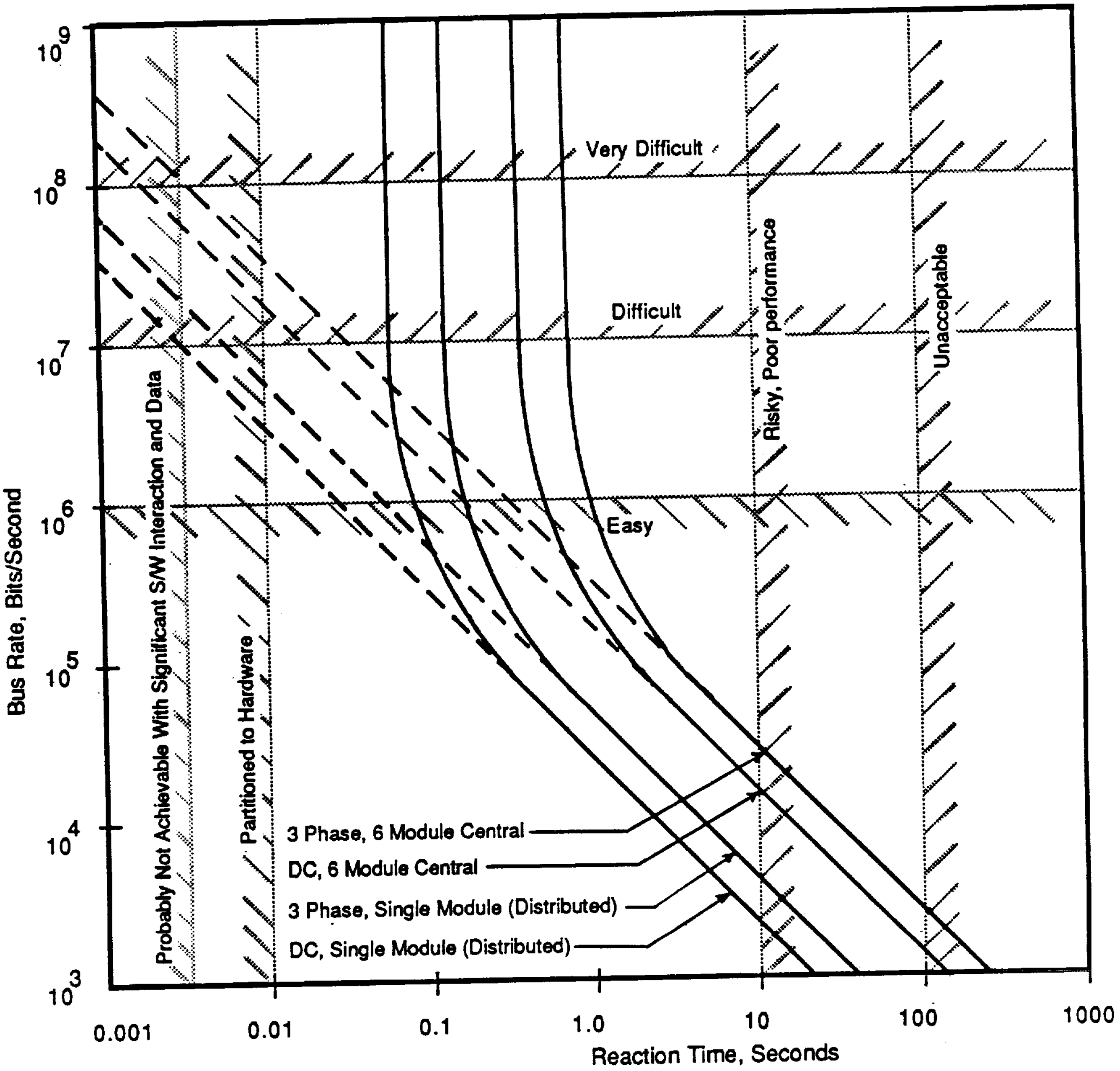


Figure 1.6.1-1 Maximum Bus Rates For Fully Loaded Module As Baseline

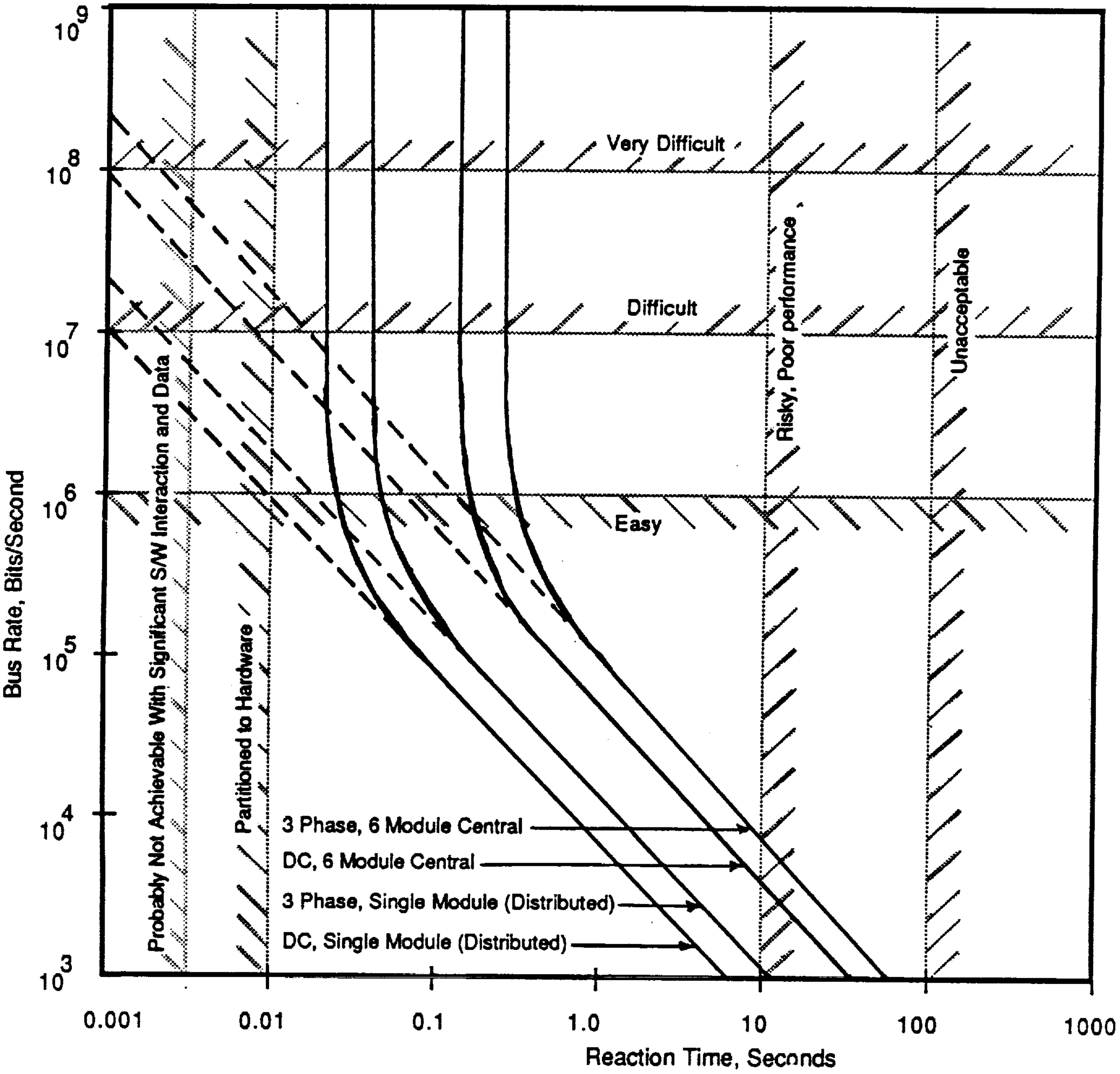


Figure 1.6.1-2 Maximum Bus Rates For Common Module As Baseline

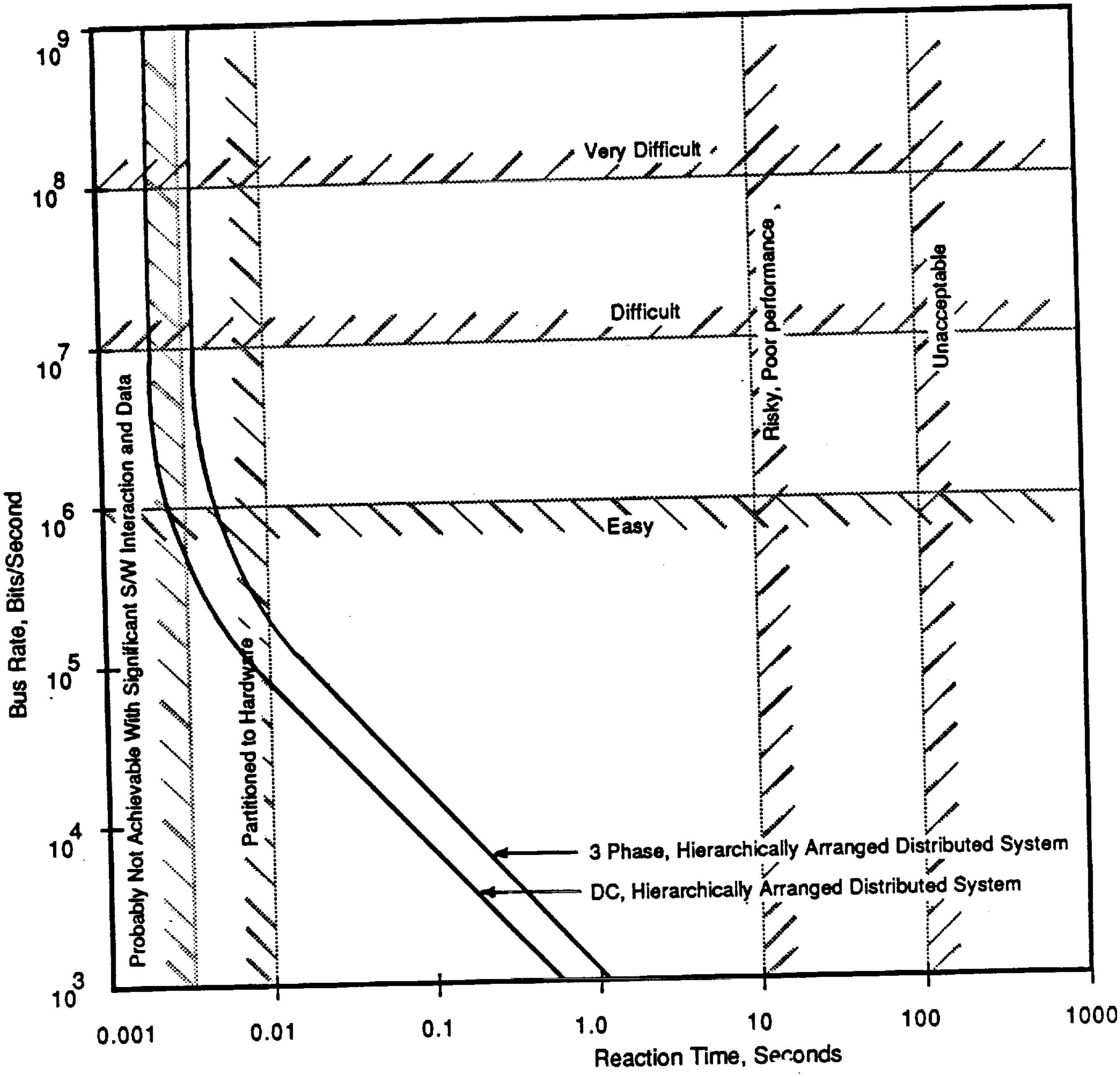


Figure 1.6.1-3 Maximum Bus Rates For Hierarchically Arranged Subsystem

1.6.2 Data Bus Selection

There are two buses or data interfaces in CM/PMAD in the HADS approach. There is a data interface between the CM/PMAD controller and the load centers. There is also a data interface between the load centers and digital logic controlling a group of six to twelve remote power controllers. We recommend that the same approach be used for each data interface, simply for commonality, to reduce risk and complexity.

Selecting 1 megahertz as a bus data rate allows design flexibility as a 1-megahertz rate satisfies the 100-millisecond distributed (to CM) and the 10-millisecond HADS approaches. The MIL-STD 1553 serial data bus has overwhelming advantages when compared to other standards for this application. The advantages are as follows:

1) Military standard—military qualified;

2) Protocol well defined, well understood--very little development risk;

3) Serial--two wire interface (shielded twisted pair);

4) Transformer isolation--Manchester coding;

5) Over 40 db common mode rejection (with transformer isolation);

6) Handles up to 31 remotes (load centers)—32 without broadcast;

7) Supported by off-the-shelf hardware;

8) Block transfer--up to 32 data words per transfer;

9) Twenty bit word includes three synchronization bits and one parity bit.

Other approaches considered were:

1) RS232C--20 kilobits/second, serial interface;

2) RS423--100 kilobits/second, serial interface;

3) RS422A--10 megabits/second, serial interface;

4) RS449--2 megabits/second, serial interface;

5) IEEE 488--4 megabits/second, parallel data bus;

6) IEEE 802.3, Ethernet--10 Megabits/second, LAN;

7) IEEE 802.4--token passing bus, LAN;

8) IEEE 802.5--token passing ring, LAN.

1.6.3 Microprocessor Evaluation

Critical issues that influence the selection include the amount of distributed automation, functions to be automated, role of expert systems, power system sensing techniques, data rates, computer language selection, commonality with other subsystems, and the overall approach to CM/PMAD automation. It is noted that if this selection were done ten years ago, the selection would have been limited to four or five eight-bit processors, all fairly limited by today's standards. Screening resulted in five processor families. General guidelines for the coarse screening were:

1) Flight qualifiable;

2) Minimum of 16 bit data bus size;

3) Minimum of 5 Megahertz clock rate;

4) Minimum address range of 1 megabyte.

The processors passing coarse screening are as follows:

1) Zilog Z8001,

2) Intel M8086,

3) Harris 80C86,

4) Motorola 68000 family,

5) Fairchild F9450.

Any of the above processors can be made adequate through development programs, however, the Fairchild F9450 is an easy choice for the following reasons:

1) MIL-STD-1750A processor--standardized and well understood, reducing any development risk;

2) Screening to BCQPL;

3) Ada cross assembler available;

4) 20-megahertz clock rate.

The Fairchild F9450 is a 16-bit processor fabricated with I^3L technology. It can perform bit, byte, word, double word, single precision (32 bit), and double precision (48 bit) floating-point numbers. Arithmetic operations include multiply and divide. The processor can directly address two megawords of memory, expandable to 16 megawords and has ten addressing modes. The F9450 is intended for real-time processing using two on-chip programmable timers and 16 levels of vectored interrupts. The architecture of the F9450 is organized into five sections: data processor, address processor, interrupt and fault processor, microprogrammed control, and a timing unit. The data processor is 16-bits wide and is responsible for all data processing in the CPU. The address processor includes an instruction counter and a memory address register that determines the addresses for all instructions and operands. All faults and interrupts, whether generated internally or externally, are handled by the interrupt and fault processor. The microprogrammed control section governs all operations of the CPU with two levels of pipelining. The timing unit generates the internal and external strobes required for internal CPU operation and the different bus transactions. It has 24 user-accessible registers. Multiprocessing is supported by a flexible bus arbitration scheme and process synchronization (test and set) instructions.

The F9450 instruction set is optimized for complex real-time applications. It implements the complete MIL-STD-1750A instruction set architecture on a single chip. It has 141 commands. Commands include signed and unsigned multiply and divide operations, program and timer control instructions, extensive interrupt and fault control operations, and multiple function instructions. Comprehensive software support for the F9450, including assemblers, loaders, simulators, and compilers, is provided by Fairchild and other sources. Software development can be performed using the Fairchild System-1 development system or the VAX 11/7XX computers using the VMS operating system.

In addition, we specifically examined the MC680XX family. The MC68000, which is representative of the family, performs operations on bit, BCD, byte, ASCII, 16-bit word, and double-word data. This 16-bit processor is actually a 32 bit machine internally, and has full 32-bit wide registers. The processor can directly address 2^{14} bytes of memory, and is organized so that it has a separate data and address bus (16-line data bus and 23-line address bus).

The architecture resembles that of a mainframe computer. All internal registers are 32-bits wide and the ALU is also 32 bits. The program counter is 24 bits, of which 23 go directly to the address bus and the 24th bit generates special strobe signals that can be externally recombined to form the 24th address bit. Only external clock, memory, and I/O circuits are needed for operation.

The instruction set of the MC68000 contains 61 basic commands, but almost all can take advantage of the 14 addressing modes. Commands include signed and unsigned multiply and divide operations, special trap instructions, powerful subroutine call and return functions, and multiple-function instructions.

Software features of the instructions set include a structure that supports high level languages such as Pascal, Basic, Cobol, and Fortran, and more recently, Ada. Each instruction operates on bytes, words, and double words, and can use any of the 14 addressing modes. The large addressing range and the powerful subroutine link and unlink instructions permit reentrant codes to be easily generated. Special trap instructions make the code easier to test with the 16-trap vectors available to the programmer.

Disadvantages of the MC68000, when compared to the F9450, are that it is not currently scheduled for flight qualification development, and it is not a MIL-STD-1750A processor. The disadvantages manifest themselves mostly in commonality and standardization.

1.7 COMPUTER LANGUAGE SELECTION

The objective of this task was to determine the most appropriate computer language for CM/PMAD consistent with NASA guidelines on computer languages for use on space station. At the time the study began, the guidelines simply favored high-level languages, and there were many possibilities. However, NASA has now standardized on Ada for all operating software, except possibly for artificial intelligence software.

Because of the new guidelines, we focused on two questions:

1) Is there any compelling reason not to use Ada for CM/PMAD conventional software?

2) What language or expert system shell is most appropriate for the artificial intelligence software?

1.7.1 Conventional Software

We found no convincing argument for avoiding using Ada in the conventional CM/PMAD software. Ada has all the capabilities needed for real-time control, and few other languages do. For instance, Ada includes multitasking and interrupt servicing as a standard feature. And while Jovial, Forth, and Modula-2 also provide multitasking, they do not do it any better than Ada. Other languages, if they support tasking at all, do so through nonstandard operating system calls or language extensions.

Similarly, we examined:

1) Costs associated with software growth and maintenance;

2) Readability of code;

3) Support of structured and modular coding concepts;

4) Standardization and commonality with other space station software;

5) Error checking;

6) How well it fits the problem, which affects the volume of code, its clarity, and the time required in development;

7) Compiler cost and availability;

8) Code size limitations;

9) Speed of compiled code;

10) Overall software development and life-cycle costs, including the development environment cost.

In this examination, we compared Ada to Modula-2, Forth, Fortran, Pascal, Assembly Languages, "C", Algol, Jovial, Basic, and PL/M. Some of these languages offered a few advantages. For example, some compilers for various languages generate code that runs faster than Ada code does. Part of this difference is built into Ada because of its error-checking capability, but part is because of the immaturity of current Ada compilers. The next generation of compilers can be expected to narrow this gap.

Likewise, according to a TRW presentation at SIGAda, Ada software tends to require a longer development schedule than some of the other languages. However, TRW concluded that Ada's error checking can result in a net reduction in cost. In short, none of the other languages offered advantages strong enough to warrant requesting a waiver from the use of Ada for conventional software in CM/PMAD.

1.7.2 Artificial Intelligence (AI) Software

Because the field of AI is changing so rapidly, to select a language for AI in the automation of CM/PMAD, it was necessary to make some assumptions about languages for AI. These assumptions are in some cases currently untrue, but which may be true in the future. It is assumed that a language for these applications will meet the following CM/PMAD AI software requirements which we deem it essential that any language intended to support CM/PMAD conform to. The language must:

1) Support powerful and flexible means of modeling rich and complex problem domains dealing with many levels of abstraction, allowing arbitrary data structures to be created and modified as desired. These data structures may include numbers, symbols, tables, relations, rules, functions, networks, etc.

2) Provide structures and functions that facilitate sophisticated search methods over large problem spaces.

3) Be supported by a strong interactive development environment that facilitates incremental program development and testing.

4)* Provide a mechanism for executing a program to create or load another program and integrate it with the presently executing program to begin executing the loaded program in a meaningful way (Ref 7).

* Not absolutely essential for ACMPMAD, but included here for completeness, as many AI applications require this mechanism.

The only language that meets these requirements is LISP. Based on the assumption that future work would allow other languages to do so, we have evaluated several languages for use in the AI portions of ACMPMAD software, including Forth, Ada, Common Lisp, LOOPS, LISP with flavors, KEE, ART, Knowledge Craft, OPS-83, OPS-5, PROLOG, and Common LISP with KEE or ART. There was no significant difference found between Ada and two versions of LISP in our analysis, summarized in the following paragraphs. However, because Ada does not meet the above AI software requirements, and in fact would only be modified to do so at costs well exceeding $50 million (Ref 7), we recommend LISP as the language for AI within the ACMPMAD.

The comparison of languages for ACMPMAD AI tasks was accomplished by an assembly of eight Martin Marietta employees with relevant backgrounds. The results presented are from group discussions as opposed to a statistical compilation of eight individual analyses. We started the investigation by defining evaluation criteria and assigned numerical ratings to indicate the importance of each. A rating of 0 indicates negligible importance; a rating of ten indicates a serious concern, hard to overcome. The criteria and their ratings are:

1) Does the use of this language or shell represent a business risk?

   Specifically, some shells, e.g., KEE and ART, are proprietary products of small companies. Should those companies fail, there would be no support for the shells. Because of the fiercely competitive nature of the AI software business at this time, it is not unlikely that some small companies will get pushed out of the market. A related problem is that shells are rapidly evolving. Even if a company survives, some of its products may become unsupported "orphans." Similarly, where standards are loose, as in Forth or LISP with Flavors, the code may reflect one vendor's dialect of the language so much that it is difficult to convert to another vendor's compiler. We assign this criterion a rating of four for pre-implementation experimentation and five for flight software.

2) Does the language or shell support large software projects?

   This question involves two issues. First, some languages, including the best-known implementations of Forth, limit program size. They will not allow large programs without such "tricks" as swapping program segments and data between disk and memory. The other issue is support of modern techniques for managing the complexity of large software projects with many programmers. These techniques include structured code, modular design, information hiding, maximizing module strength, and minimizing intermodule coupling. A rating of ten is assigned for both experimental software and flight software.

3) Are interfaces to the external world and to other types of software available?

The concern here is with providing direct control of hardware and automatic acquisition of the data needed for making decisions. In addition, some of the software may involve intensive numerical computation that may be difficult to express in the formal language or shell. Access to subroutines written in other languages may be useful if not absolutely essential. We give this criterion weights of four and six.

4) Does the language or system support validation and verification?

The main issue here is testing, but a number of subordinate issues are involved. For instance, if a proprietary shell accounts for the greater part of the code, should this shell code be verified, and will that be practical? Can the rules, frames, inheritance mechanisms, etc, in an expert system's knowledge base be verified? A language or shell that makes validation difficult may have an effect on reliability and cost. We assign this criterion weights of six and eight.

5) Does the language or system have general use for a large variety of AI software?

Practical considerations argue for minimizing the number of languages used in space station flight software, so it is desirable that one language be selected for all AI software. Some of the shells are not very flexible for use in a general AI context; they are specialized for expert systems or even a class of expert systems. Such shells would be difficult to use for a robotic path planner or a natural language data base interface. Weights of eight and ten are assigned to this criterion.

6) Is there sufficient hardware support?

In other words, can the language or shell be used with hardware that can be flown on space station? If not, can it be modified readily to do so, or can hardware be built to run it in space? The ratings are three and five.

7) Are support tools adequate?

For instance, shells for developing expert systems typically provide tools for tracing rule firings and other functions that minimize the labor of developing and testing a new expert system. Special editors, error checking compilers, and similar aids were also considered. Ratings are five and four.

8) Is the system or language suitable for space?

This consideration is closely related to verifiability. A language like Ada, for which compilers have passed extensive validation suites, would rate highly. A proprietary language that has never been examined in detail by anyone other than the vendor would get a low score. We assign weights of four and eight to this criterion.

9) Are robust implementations available?

Although a language may be good, it may be implemented poorly. For example, existing compilers for the language may fail to detect certain syntax errors, may generate incorrect code for certain constructions, or may produce few run-time error checks. These problems could result in bugs that escape detection during testing or an excessively long testing period. This criterion is assigned weights of five and ten.

10) Is there flexibility and user control of the end product?

Some shell developers make it easy to develop an expert system by providing major portions of the end product, e.g., windowing facilities, monitoring of variables, etc. While this reduces development time, it also forces the expert system into the shell vendor's mold, limiting options for such things as display appearance, hardware interfaces with external equipment, and conflict resolution strategies. This could cause problems in developing an embedded system for use in space. Ratings of four and eight are given to this criterion.

11) Does the language or shell support clean representation for the problem domain(s)?

Frames, flavors, and rules are very useful ways to represent knowledge, but they are not universally the best ways. Does the language restrict the software developer to perhaps inappropriate paradigms? This criterion is given weights of ten and eight.

After cataloging these criteria, we grouped them into categories to prevent overemphasis in certain areas. For example, several of the considerations are of concern primarily because of their effect on cost. Without grouping, cost might be overemphasized at the expense of reliability or performance. Table 1.7.2-1 summarizes how the considerations were grouped and the effect this grouping has on their weights. Essentially, what is done adjusts the weights proportionally so that the total of the weights for the criteria in each group equaled the group weight.

Table 1.7.2-1 Adjustments to Weighting Factors For CM/PMAD Evaluation Criteria

Group	Group Weights		Criteria	Original Weights		Adjusted Weights	
	Experiments	Flight		Experiments	Flight	Experiments	Flights
Lifecycle Costs	10	8	1	4	5	1.4	1.5
			2	10	10	3.4	3.0
			7	5	4	1.7	1.2
			11	10	8	3.5	2.3
Reliability	7	10	4	6	8	2.8	3.1
			8	4	8	1.8	3.1
			9	5	10	2.4	3.8
Functionality and Performance	7	8	3	4	6	1.5	1.7
			5	8	10	2.9	2.7
			6	3	5	1.1	1.4
			10	4	8	1.5	2.2

Under life-cycle cost we grouped business risk (criterion one), support for large software projects (criterion two), adequacy of support tools (criterion seven), and support for clean representation for the problem domains (criterion 11). Although some of these criteria also have reliability and performance ramifications, their primary effect was cost, either during the initial software development or later during maintenance.

Under reliability we grouped support for validation and verification (criterion four), suitability for space (criterion eight), and implementation robustness (criteria nine). Again, it can be argued that these criteria affect the other categories as well, but they seemed to be primarily reliability issues.

We grouped the remainder of the criteria under functionality and performance. These criteria included interfaces (criterion three), general utility (criterion five), hardware support (criterion six), and flexibility (criterion ten).

The various languages and shells were evaluated against each criterion, and weighted scores were computed (Tables 1.7.2-2 and 1.7.2-3) It was expected that the shells for expert system development would fare much better than they did. However, the unique constraints of flight software (and of minimizing the cost of using experimental results in flight software) tipped the balance in favor of general purpose languages. The shells were particularly weak in provision for interfaces to instrumentation and actuators, in being nonstandard and proprietary, and in making formal verification of the complete system difficult. They also were narrowly focused toward expert systems, making them difficult to use for other AI applications as planners and natural language database interface.

Table 1.7.2-2 EVALUATION MATRIX FOR CM/PMAD LANGUAGES FOR PRE-IMPLEMENTATION EXPERIMENTS

Criterion Number	Adjusted Weight	FORTH		ADA		COMMON LISP		LOOPS		LISP with Flavors		KEE		ART		Knowledge Craft		OPS-83		OPS-5		PROLOG		KEE or ART with C LISP	
		Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd
1	1.4	5	7.0	9	12.6	7	9.8	4	5.6	5	7.0	1	1.4	1	1.4	1	1.4	4	5.6	5	7.0	4	5.6	1	1.4
2	3.4	0	0.0	8	27.2	8	27.2	6	20.4	8.3	28.2	2	6.8	2	6.8	6	20.4	2	6.8	2	6.8	6	20.4	2	6.8
3	1.5	9	13.5	9	13.5	1	1.5	1	1.5	1	1.5	1	1.5	1	1.5	1	1.5	9	13.5	5	7.5	1	1.5	1	1.5
4	2.8	0	0.0	9	25.2	6	16.8	1	2.8	5	14.0	1	2.8	1	2.8	1	2.8	6	16.8	8	22.4	6	16.8	1	2.8
5	2.9	2	5.8	4	11.6	7	20.3	7.3	21.2	8	23.2	6	17.4	6	17.4	7.3	21.2	5	14.5	5	14.5	4	11.6	7	20.3
6	1.1	10	11.0	10	11.0	8	8.8	8	8.8	8	8.8	5	5.5	5	5.5	7	7.7	8	8.8	8	8.8	8	8.8	8	8.8
7	1.7	5	8.5	5	8.5	7	11.9	8.3	14.1	8	13.6	9	15.3	9	15.3	8	13.6	1	1.7	2	3.4	4	6.8	9	15.3
8	1.8	5	9.0	10	18.0	6	10.8	1	1.8	1	1.8	1	1.8	1	1.8	1	1.8	1	1.8	1	1.8	1	1.8	1	1.8
9	2.4	8	19.2	6	14.4	7	16.8	2	4.8	5	12.0	3.7	8.9	3.7	8.9	3.7	8.9	3	7.2	3	7.2	3	7.2	3.7	8.9
10	1.5	10	15.0	10	15.0	8	12.0	5	7.5	7	10.5	2	3.0	2	3.0	5	7.5	8	12.0	8	12.0	8	12.0	2	3.0
11	3.5	3	10.5	2	7.0	8	28.0	9	31.5	9	31.5	5	17.5	5	17.5	8.3	29.1	2	7.0	2	7.0	2	7.0	8.3	29.1
SUM		▨	99.5	▨	164.0	▨	163.9	▨	120.0	▨	152.1	▨	81.9	▨	81.9	▨	115.9	▨	95.7	▨	98.4	▨	99.5	▨	112.3
Normalized Score		4.1		6.8		6.8		5.0		6.3		3.4		3.4		4.8		4.0		4.1		4.1		4.7	

Table 1.7.2-3 EVALUATION MATRIX FOR CM/PMAD LANGUAGES FOR AI FLIGHT SOFTWARE

Criterion Number	Adjusted Weight	FORTH		ADA		COMMON LISP		LOOPS		LISP with Flavors		KEE		ART		Knowledge Craft		OPS-83		OPS-5		PROLOG		KEE or ART with C LISP	
		Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd	Raw	Wtd
1	1.5	5	7.5	9	13.5	7	10.5	4	6.0	5	7.5	1	1.5	1	1.5	1	1.5	4	6.0	5	7.5	4	6.0	1	1.5
2	3.0	0	0.0	8	24.0	8	24.0	6	18.0	8.3	24.9	2	6.0	2	6.0	6	18.0	2	6.0	2	6.0	6	18.0	2	6.0
3	1.7	9	15.3	9	15.3	1	1.7	1	1.7	1	1.7	1	1.7	1	1.7	1	1.7	9	15.3	5	8.5	1	1.7	1	1.7
4	3.1	0	0.0	9	27.9	6	18.6	1	3.1	5	15.5	1	3.1	1	3.1	1	3.1	6	18.6	8	24.8	6	18.6	1	3.1
5	2.7	2	5.4	4	10.8	7	18.9	7.3	19.7	8	21.6	6	16.2	6	16.2	7.3	19.7	5	13.5	5	13.5	4	10.8	7	18.9
6	1.4	10	14.0	10	14.0	8	11.2	8	11.2	8	11.2	5	7.0	5	7.0	7	9.8	8	11.2	8	11.2	8	11.2	8	11.2
7	1.2	5	6.0	5	6.0	7	8.4	8.3	10.0	8	9.6	9	10.8	9	10.8	8	9.6	1	1.2	2	2.4	4	4.8	9	10.8
8	3.1	5	15.5	10	31.0	6	18.6	1	3.1	1	3.1	1	3.1	1	3.1	1	3.1	1	3.1	1	3.1	1	3.1	1	3.1
9	3.8	8	30.4	6	22.8	7	26.6	2	7.6	5	19.0	3.7	14.1	3.7	14.1	3.7	14.1	3	11.4	3	11.4	3	11.4	3.7	14.1
10	2.2	10	22.0	10	22.0	8	17.6	5	11.0	7	15.4	2	4.4	2	4.4	5	11.0	8	17.6	8	17.6	8	17.6	2	4.4
11	2.3	3	6.9	2	4.6	8	18.4	9	20.7	9	20.7	5	11.5	5	11.5	8.3	19.1	2	4.6	2	4.6	2	4.6	8.3	19.1
SUM		▨	123.0	▨	191.9	▨	174.5	▨	112.1	▨	150.2	▨	79.4	▨	79.4	▨	110.7	▨	108.5	▨	110.6	▨	107.8	▨	93.9
Normalized Score		4.7		7.4		6.7		4.3		5.8		3.1		3.1		4.3		4.2		4.3		4.1		3.6	

1.8 AUTOMATION APPROACH/ARCHITECTURE DEFINITION

Reviewing all previous sections results in recommending a HADS approach for CM/PMAD as shown in Figure 1.8-1. The selected approach has maximum flexibility and growth capability while maintaining a high degree of automation. Additionally, the approach meets all the derived possible requirements described in the previous sections. It is noted that the approach assumes there is a common module data management network and a space station-level data management network. The CM/PMAD PCU interfaces to the common module data management network via a bus interface unit. The PCU interfaces to the lower level processors via a MIL-STD 1553 bus that is transformer isolated at each drop point. The lower level processors interface to the remote switches and sensing devices through a controller buffer. There are approximately ten remote devices, e.g., remote power controllers, etc, storing data and accepting commands from a controller buffer. The interface between the controller buffer and the remote devices is synchronous, located on the same card as the remote devices.

The major functional responsibilities of the PCU are CM/PMAD top-level internal command and data handling, overall distribution management, load prioritization and scheduling, and subsystem health management. The functions allocated to the lower-level processors are as follows

1) Data acquisition,
2) Data conditioning,
3) Data synchronization (time stamping for analysis in system solution),
4) Data compression,
5) Local limit checking,
6) Short-term data storage,
7) Requested data transfer,
8) Effectors control.

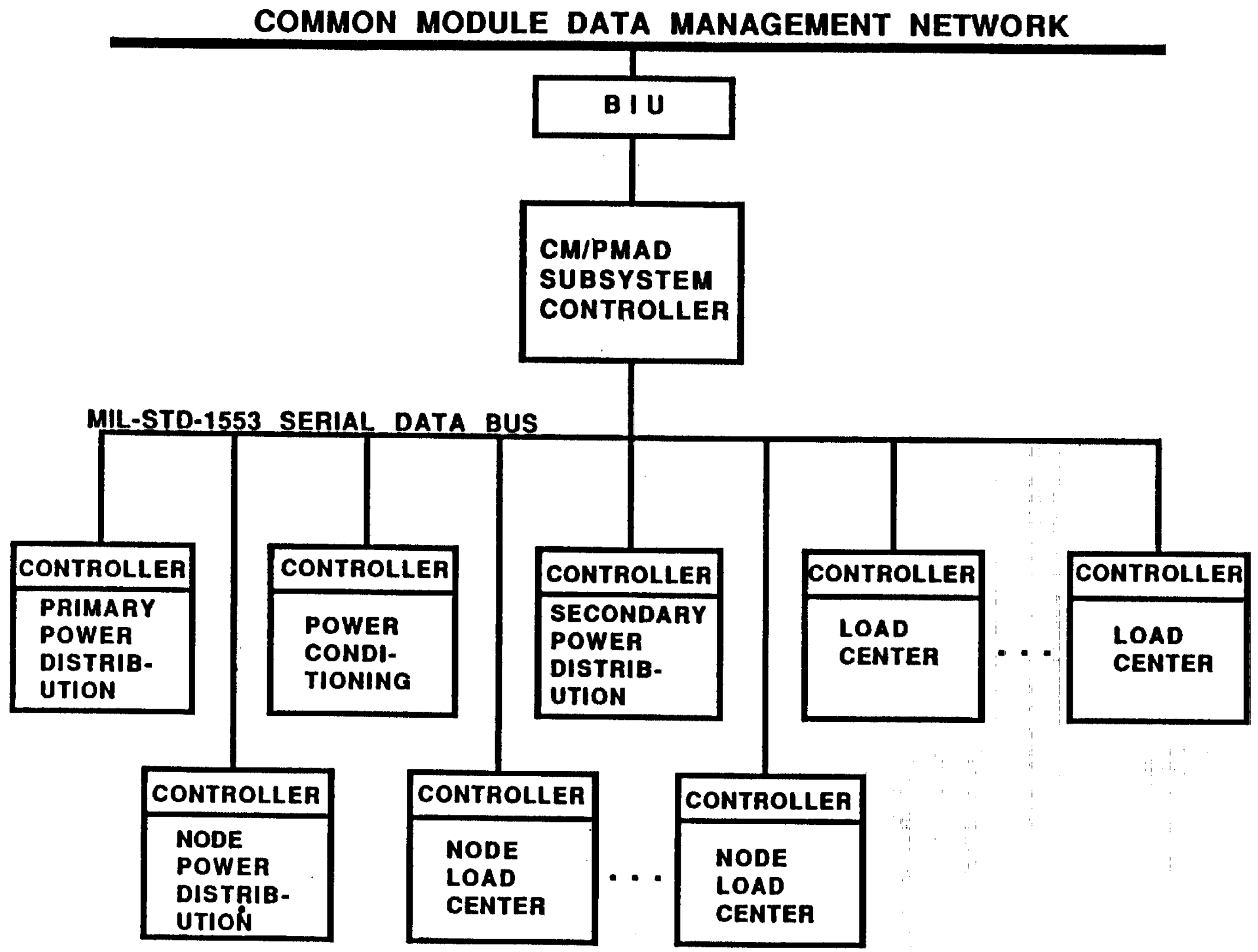


FIGURE 1.8-1 CM/PMAD AUTOMATION ARCHITECTURE

1.9 TIME SHARED USE OF SPACE STATION BUS VS CM/PMAD DEDICATED BUS

Time sharing an overall space station network data bus versus using a dedicated CM/PMAD bus is highly dependent on the power subsystem control architecture approach as well as the maximum required data rates. Reviewing the centralized approach versus a distributed approach (Section 1.4.1) and the maximum data rates (Section 1.6) possible in worst-case situations, we recommend a dedicated CM/PMAD data bus. The advantages of the HADS in reduced protocol overhead, development risk, growth, and testability would be significantly reduced by sharing an overall space station data bus network.

Additionally, another major drawback in time sharing the space station network data bus is the possibility of effecting the space station bus during a fault condition. During a major fault condition, there will be a flood of data from all subsystems while each subsystem is attempting rectification through local redundancy. Additional complexities result if the fault condition is power related. With the recommended architecture, a minimum of data is required between CM/PMAD and other space station subsystems; therefore, there would be little to gain by time sharing the space station network.

1.10 WORK PACKAGE-04 DATA EXCHANGE IDENTIFICATION

This section describes the data that must flow between the CM/PMAD task and Work Package--04.

1.10.1 Data from WP-04 to CM/PMAD

1.10.1.1 Total Power Allowance Versus Time--When necessary, Work Package -04 must be able to send a total power allowance versus time projection to CM/PMAD. Total power is defined as the magnitude (in kVA) of instantaneous complex power that could be delivered to the CM at a given moment without causing a circuit breaker or similar device to open in WP-04. The time resolution of the projection will be limited; if the projection were plotted as a graph, it would look like a stair-step function, with each stair-step representing a discrete time interval. The value of total power assigned to each stair-step should be the maximum allowable value predicted for that interval.

The projection period should probably be no shorter than one orbital day (or, when the station is in continuous sunlight, no shorter than one orbit) and should probably be no longer than one crew shift period. The time resolution of the projection should be no narrower than one CM/PMAD control cycle (about 10 seconds) and should be no wider than 5% of the projection interval.

In determining allowed values of total power, WP-04 must assume worst-case power factors for the CM based on information contained in the Ground-generated timeline or on experience. It cannot compute worst-case power factors, because it will have no detailed knowledge of the CM/PMAD power network or its Loads.

Work Package--04 must be able to send such projections to CM/PMAD during the station-wide resource bargaining portions of formal onboarJ scheduling operations (Appendix A, Function 3.2.2, Onboard Scheduling). When called to do so, it should be able to transmit the entire projection within 1 minute.

1.10.1.2 Energy Consumption Allowance Versus Time--When necessary, Work Package--04 must be able to send an energy consumption allowance versus time projection to CM/PMAD. The time resolution of the projection will be limited; if the projection were plotted as a graph, it would look like a stair-step function, with each stair-step representing a discrete time interval. The value of energy consumption assigned to each stair-step should be the total allowable value predicted for that interval only.

The period and time resolution for this projection should be the same as those for the total power allowance versus time projection described above.

Work Package--04 must be able to send such projections to CM/PMAD during the station-wide resource bargaining portions of formal onboard scheduling operations (Appendix A, Function 3.2.2, Onboard Scheduling). If WP-04 detects a new, significant fault in the energy conversion or energy storage subsystems, it should quickly be able to send a revised projection covering the remainder of the present projection period. When called on to send an energy consumption allowance projection, it should be able to transmit the entire projection within 1 minute.

1.10.2 Data from CM/PMAD to Work Package--04

1.10.2.1 Total Power Request vs. Time--When necessary, CM/PMAD must be able to send total power request versus time projections to Work Package--04. Total power is defined as the magnitude (in kVA) of instantaneous complex power that CM would require to support a preliminary load enable schedule evolved during formal scheduling operations (Appendix A, Function 3.2.2, Onboard Scheduling). A projection must be sent for each load class (see discussion of load class in Appendix A, Function 3.2.2.1, Major Scheduling). The time resolution of the projections will be limited; if the projections were plotted as graphs, they would look like stair-step functions, with each stair-step representing a discrete time interval. The value of total power assigned to each stair-step should be the maximum required value predicted for that interval for that load class.

The projection periods should probably be no shorter than one orbital day (or, when the station is in continuous sunlight, no shorter than one orbit) and should probably be no longer than one crew shift period. Meanwhile, the time resolution of each projection should be no narrower than one CM/PMAD control cycle (about 10 seconds) and should be no wider than 5% of the projection interval.

In determining requested values of total power, CM/PMAD must assume nominal major input bus voltages, and worst-case power factors for delivered power based on information contained in the ground-generated timeline or on experience. It cannot compute worst-case power factors, because it will have no detailed knowledge of the other module power networks or their loads or of the space station power network and its loads.

CM/PMAD must be able to send such projections to WP-04 as soon as it is able to evaluate its preliminary load enable schedule during the station-wide resource bargaining portions of formal onboard scheduling operations (Appendix A, Function 3.2.2, Onboard Scheduling). When ready to send its projections, it should be able to transmit them all within "n" minutes, where "n" is the number of load classes.

1.10.2.2 Energy Consumption Request vs. Time--When necessary, CM/PMAD must be able to send energy consumption request versus time projections to WP-04. The time resolutions of the projections will be limited; if the projections were plotted as graphs, they would look like stair-step functions, with each stair-step representing a discrete time interval. The value of energy consumption assigned to each stair-step should be the total requested value predicted for that interval only.

The period and time resolution for this projection should be the same as those for the total power request versus time projection described above.

2.0 SUMMARY

Five Government-furnished power distribution candidate topologies were reviewed and compared. The candidates included distributing an input power type of 115/200 Vac, 400 Hz three-phase with and without internal bulk power conditioning. Also included was an input power type of 150 Vdc with bulk conversion to 115/200 Vac, 400 Hz three-phase power. An all DC system was also considered for completeness in comparison. A comparison and review of the candidate power distribution topologies indicated significant automation software impact where three phase power was utilized. The impact was due to three phase requiring an increase in total number of sensors required for automation and the increased complexity of a network solution used in fault diagnosis and isolation. The increased number of sensors results in an increase in the automation software effort because of the additional internal data requiring handling. In addition, an AC system requires more complex software data conditioning with respect to peak voltages, root-mean-square voltages, frequency accounting, and power factors. Additionally, in the case of three-phase systems, load balancing is also a factor in an increase of overall automated management hardware and software complexity. A range of possible savings in millions of dollars was estimated by including the all DC distribution system. The automation software was estimated utilizing the functions defined in this study.

The CM/PMAD major power network control functions were defined as: 1) Distribution Management; 2) Load Management; 3) Health Management; and 4) Command/Data Interface. These functions were further decomposed into levels sufficient to perform function partitioning. Function partitions were defined as: 1) Hardware; 2) Algorithmic Software; 3) Expert System; 4) Crew; and 5) Expert-aided Crew. In addition, rules for partitioning were developed and used in the function partitioning. The major control functions of distribution management and command/data interface were partitioned to algorithmic software, while portions of load management and health management were partitioned to expert systems.

In addition to function definition and partitioning, automation issues were addressed. Issues included distributed versus centralized automation, fault isolation, load management, Space Station power and data bus interfaces, data and power ground separation, local energy storage impacts, crew interfaces, and sensing techniques.

Required data bus rates were found to be highly dependent on the overall automation approach, central versus distributed. The central approach required very high data rates while the recommended hierarchically arranged distributed approach can be supported by data rates less than megabits per second.

Automation hardware was evaluated having addressed issues and data rates. CM/PMAD internal hardware recommended included the military standard 1750A chip set, with a Military Standard 1553 data bus. The use of fiber optics for a data bus internal to CM/PMAD was found with attractive advantages in performance characteristics in electrical isolation and electromagnetic interference considerations. However, with the use of fiber optics not being absolutely required, and with operational issues such as onboard maintenance, environmental effects, total dose radiation effects, outgassing, and connector mating, it was not recommended.

Computer languages internal to CM/PMAD were also evaluated. Software studies resulted in the identification of no compelling reason to avoid the use of Ada as the recommended computer language for algorithmic software within CM/PMAD. However, argument was found for the use of LISP computer language in expert systems development for Space Station IOC.

The overall approach recommended utilizes an hierarchically arranged distributed system for CM/PMAD control. A power subsystem processor interfaces to the Common Module data management network through a bus interface unit. The power subsystem processor communicates with and directs the lower level power subsystem processors through an internal dedicated power subsystem data bus. The lower level processors, in turn, manage the lowest level data with immediate control of switchgear through digital logic interfaces. The lower level processors are responsible for data acquisition, conditioning, compression and synchronization as well as local limit checking, short term data storage and effectors control.

The recommended approach requires certain data between the Space Station power production element and the CM/PMAD. The CM/PMAD power subsystem processor must receive the total power allowance versus time and the energy consumption allowance versus time. In addition, the CM/PMAD power subsystem processor must provide to the power production element the total power request versus time and the energy consumption request versus time.

3.0 REFERENCES

The following documents and technical publications are referenced in this report:

1. D. J. Weeks, "Application of Expert Systems in the Common Module Electric Power System," NASA/MSFC, August 1985.

2. D. J. Weeks and R. T. Bechtel, "Autonomously Managed High Power Systems," NASA/MSFC, August 1985.

3. ATAC Technical Report, "Advancing Automation and Robotics Technology for the Space Station and for the U.S. Economy," NASA Technical Memorandum 87566, March 1985.

4. D. S. Prerau, "Selection of an Appropriate Domain for an Expert System," AI Magazine, Summer 1985.

5. M. S. Imamura, et. al., "Design and Development of Automated Power Systems Management: Phase 1 Technical Report," Martin Marietta Report MCR-78-539, April 1978.

6. P. R. Turner, et. al., "Autonomous Spacecraft Design and Validation Methodology Handbook: Issue 2," Jet Propulsion Laboratory Report 7030-4, April 1984.

7. D. O. Christy and W. E. Loper, "Use of AI for Ada Development: Appendix A - Using Ada for AI Development," NOSC, September 1984.

APPENDIX II – TASK II DATA

MCR-87-548
Contract No. NAS8-36433

Task II
Study
Report

August 1987

SPACE STATION AUTOMATION OF COMMON MODULE POWER MANAGEMENT AND DISTRIBUTION

W. D. Miller
R. Walsh

MARTIN MARIETTA
DENVER AEROSPACE
P.O. Box 179
Denver, Colorado 80201

FOREWORD

This report was prepared by Martin Marietta Denver Aerospace, for the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, in response to Contract NAS8-36433, and is submitted as the Task II Summary Report, as specified in the contract data requirements list.

CONTENTS

1.0 INTRODUCTION 3

2.0 TASK II STUDY SUMMARIES 7

2.1 Definition of Knowledge Base and Deterministic Algorithms . . . 7

2.1.1 Knowledge Base Definition. 7

2.1.2 Knowledge Base Management System Definition 10

2.1.3 Inference Procedures Definition 11

2.1.4 Deterministic Algorithms 11

2.2 Program Plan 12

2:3 Fault Management Approach. 14

2.4 Hardware Development Definition. 25

3.0 SUMMARY. 26

4.0 REFERENCES 27

1.0 INTRODUCTION

This document is in response to the Summary Report requirements of Task II of the Statement of Work for Automation of the Core Module Power Management and Distribution (CM/PMAD). Task II, Definition of Hardware and Software Elements of Automation, includes defining those elements of hardware and software necessary for implementation of the automation approach defined in Task I of this contract. Elements of software include knowledge base, knowledge base management systems, inference procedures, as well as deterministic algorithms. Efforts, with respect to software, also include an overall approach to development of non-standard software for use in CM/PMAD automation. Task II also includes the development of an overall approach to fault management for CM/PMAD. Hardware definition includes hardware requiring development for implementing the automation approach.

During Phase I, we performed characterization studies in defining the plan for an overall approach to automation of a CM/PMAD. The CM/PMAD functions were defined as shown in Figure 1-1 with power network control defined as: 1) Distribution Management, 2) Load Management, 3) Health Management and 4) Command/Data Interface. Further decomposition of the power network control function resulted in the functional breakdown shown in Figure 1-2. Study of those functions as well as partitioning efforts resulted in identification of several areas where the use of artificial intelligence techniques (specifically knowledge based systems) is warranted. The major control functions of distribution management and command/data interface were partitioned to algorithmic software, while portions of load management and health management were partitioned to expert systems.

In addition to function definition and partitioning, automation issues were addressed. Issues included distributed versus centralized automation, fault isolation, load management, Space Station power and data bus interfaces, data and power ground separation, local energy storage impacts, crew interfaces and sensing techniques. In Task I, required data bus rates were found to be highly dependent on the overall automation approach, central versus distributed. The central approach required very high data rates while the recommended hierarchically arranged distributed approach can be supported by data rates less than megabits per second.

In Task I, computer languages internal to CM/PMAD were also evaluated. Software studies resulted in the identification of no compelling reason to avoid the use of Ada as the recommended computer language for algorithmic software within CM/PMAD. However, argument was found for the use of LISP computer language in expert systems development for Space Station IOC.

The overall approach recommended in Task I utilized an hierarchically arranged distributed system for CM/PMAD control. A power subsystem processor interfaces to the Core Module data management network through a bus interface unit. The power subsystem processor communicates with and directs the lower level power subsystem processors through an

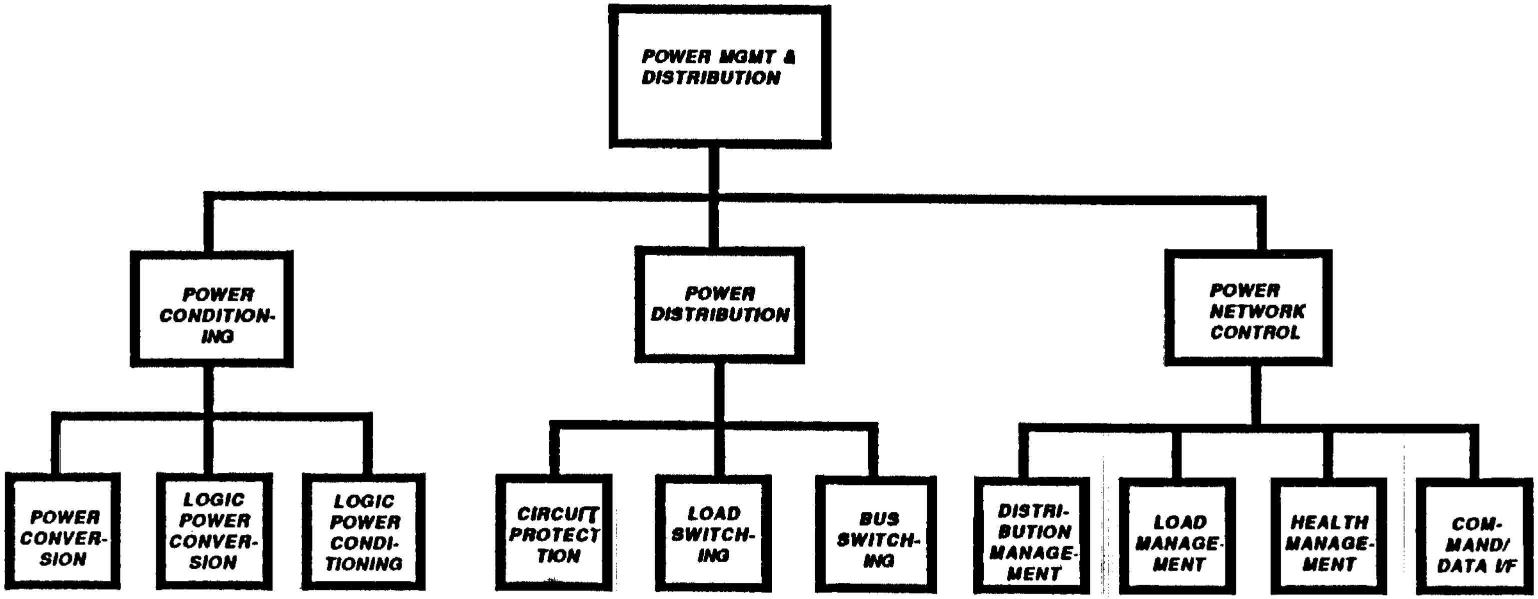
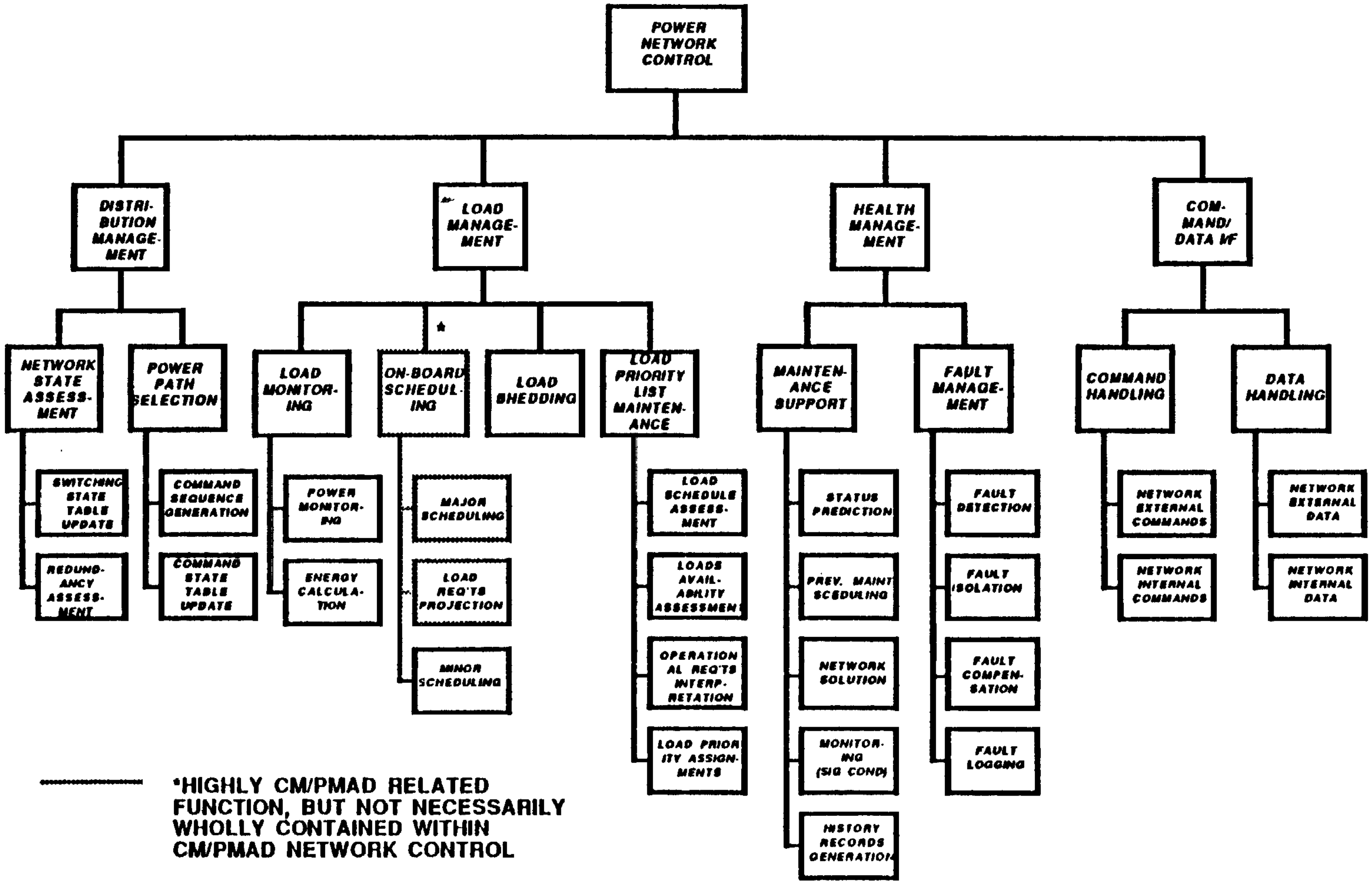


FIGURE 1-1 POWER MANAGEMENT AND DISTRIBUTION TOP LEVEL FUNCTIONAL BREAKDOWN



┈┈┈┈┈ *HIGHLY CM/PMAD RELATED FUNCTION, BUT NOT NECESSARILY WHOLLY CONTAINED WITHIN CM/PMAD NETWORK CONTROL

FIGURE 1-2 FUNCTIONAL BREAKDOWN OF CM/PMAD POWER NETWORK CONTROL

internal dedicated power subsystem data bus. The lower level processors, in turn, manage the lowest level data with immediate control of switchgear through digital logic interfaces. The lower level processors are responsible for data acquisition, conditioning, compression and synchronization as well as local limit checking, short term data storage and effectors control.

2.0 TASK II STUDY SUMMARIES

2.1 DEFINITION OF KNOWLEDGE BASE AND DETERMINISTIC ALGORITHMS

2.1.1 Knowledge Base Definition

A knowledge base typically contains the procedural, often heuristic information necessary for solving a complex problem. This general information, similar to the understanding an expert in the field might have, is used in conjunction with specific factual data to handle a particular problem. Defining knowledge bases for the CM/PMAD first requires identifying functional areas they will address. Drawing on the partitioning in Task I, these functional areas are load scheduling, contingency planning and aspects of both fault prediction during normal operations and fault management during anomalous conditions. While these categories are not strictly sequential or temporal in nature, they do describe the phases or states of the subsystem. By augmenting the algorithmic execution of the conventional software, the knowledge based systems provide CM/PMAD with an ability to react "intelligently" and with a higher level of autonomy to changes in mission requirements, power subsystem status, related subsystems' status, and available resources.

Automated on board final load scheduling, which reduces human interaction, must result in an executable time-line or load schedule. The knowledge base for this function deals with temporal and spatial constraints, as well as power subsystem hardware availability constraints. Concisely developing this knowledge base also requires understanding how loads are described, including load interrelationships, constraints, and requirements. These load descriptions are the data that a scheduler uses heuristically to create a schedule based on its understanding of how to balance constraints. Consistent with Task I results, a large part of the information for this function will be transmitted in the coarse or multi-day mission uplinked time-line, in which electrical loads will be time "windowed" with any required constraints. Example load constraints in the knowledge base include:

1) earliest and/or latest task start time,
2) latest completion time,
3) minimum/maximum duration,
4) physical location,
5) set-up time or initialization period,
6) load interdependencies (other loads/scheduled resources),
7) required resources each profiled, e.g., power, crew interaction, thermal, experiment specific resources,
8) maintenance periods,
9) load class assignment for relative importance (includes mission requirements),
10) overall qualitative "level loading" requirements of any specified resources,

11) resources availability (versus maximum load duration which may be longer than period being "scheduled"), and
12) load continuity requirements.

General load descriptions on which the knowledge base is used include:

1) unique identifier,
2) load owner,
3) location (rack, power switch number, etc.),
4) dependence on other subsystems and other loads,
5) resource requirements profile (with maximum and minimum) resolved to minutes,
6) tolerance to interruption,
7) power usage description, e.g.:

a) level constant load,
b) periodic with constant peak and valley usage values,
c) aperiodic with constant peak and valley usage values,
d) periodic with variant peak and valley usage values,
e) aperiodic with variant peak and valley usage values,
f) power factor profile, and
g) in-rush current surge magnitude and duration,

8) redundancy methodology,
9) crew serviceable components, and
10) load priority statement (load class designator and general time based profile of load importance relative to importance at load start.

Contingency planning, a second area for knowledge based systems, is required for the graceful shedding of loads when there is a required change in loading due to a significant decrease in resources available, either identified by the system or directed by the Space Station manager. This planning is embodied in the load priority list management function described in Task I. The knowledge base, best implemented in production rules, consists largely of heuristics patterned after those used by experts in determining the dynamic relative importance of loads. Example rules would be:

1) Life critical loads are of the highest importance.
2) Equipment safety is of high importance.
3) Overall mission requirements are probably more important than individual experiment requirements.
4) If a load is dependent on another load, both loads should have close relative priorities.
5) Non-repeatable experiments should have higher priority than easily repeatable experiments.
6) A load requiring redundant power inputs probably has a higher priority than a load not requiring redundancy.
7) A load with less tolerance to interruption generally has more priority than a load with greater tolerance to interruption.

8) A load near completion generally has a higher priority than a newly started load in the same class.
9) Loads of a higher class should have relatively higher priority than loads of a lower class.
10) A load priority may increase as a result of the failure of a similar load, e.g., the last working light in an area.
11) A failed or unavailable load has no priority.
12) Loads requiring perishable resources generally have higher priority than loads in the same class that do not.
13) Loads requiring many resources generally have higher priority than loads in the same class that do not.
14) Note: Apparent relative priority (defined by its shedding or lack of shedding during a resource reduction or anomaly) for a load can be increased if its shedding will not aid crisis resolution.

Fault management, also requiring a knowledge base, is discussed here in a general sense with respect to Space Station power management and distribution. Section 2.3, Fault Management, presents approaches for its implementation. Types of credible faults, reference Task I Summary Report, that must be considered are as follows:

1) faults external to CM/PMAD (external to the subsystem or external to the element),
2) open or short circuit faults,
3) faulty sensor,
4) failed switch (at any hierarchical level),
5) faulty local power conditioner,
6) subsystem control unit failure,
7) microprocessors subsidiary to the subsystem control unit, and
8) load faults.

The knowledge base required to detect, diagnose, isolate and provide correction or recommended action consists of knowledge of the configuration of the power system, principles in its operation under normal conditions and heuristics for locating and handling failures. Configuration knowledge includes subsystem architecture, redundancy methodology, sensor type/location and component operation. Example required component operation knowledge is summarized for the lowest level switch, a remote power controller (RPC):

1) Current limiting capability,
2) Data acquisition constraints,

    a) measurement type,
    b) accuracy, resolution and repeatability,
    c) speed,
    d) dependence on environment,
    e) format including status,

3) Current switching capability level,
4) Circuit protection parameters,
5) Power stage dependence on environment, and
6) Typical failure mode.

The operation of the subsystem with its components is best modeled using an object oriented approach to describe the components' interconnections and causal interactions. This can be used to generate expected values to be compared to measured values for fault identification and location. Additionally, principles governing the subsystem operation together with the above mentioned heuristics provide the basis for resolving fault location. Example heuristics would be:

1) More than 3% continuous voltage drop across a switch indicates a serious problem in the switch.
2) Widely varying current readings in a switch or sensor while all other readings appear normal indicate a faulty sensor.
3) If a switch has recently changed state, dependent asynchronous measurements may not be valid for comparison.

The detection of incipient faults, or fault prediction, also a part of health management, also requires a knowledge base. The function utilizes historical data combined with configuration and component knowledge as well as trend analysis techniques to detect the incipient faults. Working as a background task on the data base, including fault and event historical records, the prediction process shares a knowledge base with the fault management process. In addition to what the latter requires, the fault prediction knowledge base must include an expert's knowledge on the importance of subtle changes in CM/PMAD overall operation as well as individual component operation. The knowledge would not just be a set of "soft limits" of operation, but would more importantly include the knowledge of component interaction under stress and the resultant effect on future operation.

2.1.2 Knowledge Base Management System Definition

The development of knowledge based systems (KBS) in the prototype phases of space station could be profitably based on a number of existing "expert system building tools" (typically with a "production system" at its core) or, as they are also called, knowledge base management systems (KBMS). An extant KBMS provides a variety of advantages. As a very high level language, it can be easy to learn and can spare the developer entanglement with low level coding. Many KBMSs available offer considerable flexibility in representational techniques for information, allowing a good fit between data and representation, different representations for different types of data and, equally important, the opportunity for changing or experimenting with the representation to find the most appropriate fit. In addition, many KBMSs provide both good applications interfaces, useful for clear concept presentation during demonstrations, and good development interfaces, aiding the system developer in orchestrating the knowledge in different parts of the system.

For the implementation of on-board knowledge base systems, however, the current generation of tools by their very generality and their interface capabilities have far too much overhead in both storage and computation time. They do not currently provide a practical and efficient implementation choice for deployed knowledge based systems. A more appropriate approach would would be to use such tools where they are suitable for prototyping and initial development. Deployment development could then proceed in at least two ways, depending on several factors including need for flexibility. If little flexibility is required, then the knowledge based system could be "hard-coded" in a suitable language such as Lisp or Ada. A more likely route, depending on the need for flexibility or other special capabilities, would be to implement application specific, highly tailored tools in a language like Lisp or Ada and then use those tools to implement the KBS. Done properly, this latter approach could combine the efficiency of "hard-coding" with the power of the more general tools.

2.1.3 Inference Procedures Definition

Inference procedures for fault management and status prediction are expected to be forward chaining intermixed with causal reasoning. Scheduling and load priority list management will primarily be based on constraint balancing techniques coupled with temporal reasoning.

2.1.4 Deterministic Algorithms

Functions that are usually understood only by recognized experts, and require the knowledge and judgement of an expert to fulfill defined requirements are best performed by experts or knowledge based systems. However, functions ranging from simple to complex can be accomplished with deterministic algorithms. We define complex as functions that are technically understood using knowledge available from accepted text books or procedures, but that involve advanced scientific skills or special training to implement. In addition to performing many functions in their entirety, deterministic algorithms can be integrated with some knowledge based systems for an increase in overall function performance efficiency. Fault management is an example of this approach. The fault detection function in the space station growth configuration is shared by hardware, deterministic algorithms, and a knowledge based system, each with its own responsibilities. Primarily, however, deterministic algorithms are recommended for use according to the guidelines developed in Task 1.

From our function partitioning studies of Task 1, the deterministic algorithm approach should be used for the following CM/PMAD functions, reference Figure 1-2:

1) distribution management,
2) load monitoring and load shedding, within load management,
3) signal conditioning within maintenance support,
4) fault detection (limit verification and condition exception handling) within fault management, and
5) the entire command and data interfacing function (communications as well as data compression, data reporting, and execution of commanded switch control.)

In addition, deterministic algorithms should be used for initialization of the subsystem for initial use and after any periods of non-use.

2.2 PROGRAM PLAN

Developing the software systems described in paragraph 2.1 above, particular attention to the AI techniques is warranted. There are three basic themes that typify much of AI, including that above, which necessitate a development approach somewhat different than in conventional software development. These considerations are as follows:

1) The use of knowledge representation techniques,
2) the use of heuristic search methods which render large problem spaces manageable, and
3) the use of informative, interactive user friendly interfaces.

In addition, there are issues that must be dealt with when describing an approach to expert systems development and testing:

1) Does the development effort address the correct problem statement?
2) Is there an expert who solves the problem sufficiently accessible to the project?
3) Does the knowledge in the program emulate that of the expert in the chosen domain, including capability to interact with a non-expert?
4) Is the program sufficiently robust and free of coding errors?
5) Can the program be modified to incorporate more expertise, or to handle a related or analogous problem area?

In order to obtain satisfactory results with respect to the concerns mentioned above, expert systems designers have developed a methodology for program development based primarily on empirical analysis. Using a technique known variously as iterative refinement or rapid prototyping, designers develop an initial prototype system intended to elucidate essential problem aspects, then develop successive versions of the program (which may or may not incorporate any of the previous versions) as more is learned about the problem and the most promising approaches to its solution.

As each prototype version is developed, it must be evaluated in several ways. The authors of the prototype must ensure that it performs as expected, i.e., the code as written expresses their intent. The experts must evaluate the prototype, verifying that it contains their approaches to solving the problem and that it applies their knowledge appropriately. The customer must make certain that program will perform a useful task, solving the problem they have in mind, and that the interface(s) to the program ensure its usefulness within their operating regime. This frequent evaluation is reflected in the progression of steps in the expert system development approach, as listed below.

PHASE 1: Initial Problem Analysis and Prototyping

1) Initial problem definition
2) Identification, location and selection of experts
3) Initial knowledge acquisition, knowledge representation study, and initial interface design
4) Development tools selection and prototype hardware selection
5) End user technical review
6) Knowledge acquisition, representation and coding, interface design, initial coding and function level testing
7) Prototype expansion and expert evaluation
8) End user demonstration and evaluation

PHASE 2: Second Generation Prototyping and Knowledge Refinement

1) Analysis of prototype with respect to end user system evaluation
2) Second generation prototype preliminary design
3) Second generation tools and hardware selection and/or development
4) Second generation prototype design refinement
5) End user/system technical review
6) Knowledge refinement with experts
7) Second generation prototype coding and documentation
8) Prototype domain coverage expansion, prototype expert evaluation, and internal function level testing
9) End user/system demonstration, review and performance evaluation

PHASE 3: Deliverable System Development

1) Problem restatement
2) Preliminary system design
3) Experts and end user design review
4) Detailed design, including interface(s)
5) Experts and end user detailed design review
6) System coding and documentation
7) Function level testing
8) Element level integration and verification testing
9) System level integration and verification testing
10) End user use and evaluation

These three phases - initial problem analysis and prototyping, second generation prototyping and knowledge refinement, and deliverable system development - are characterized by level of attention to considerations such as efficiency and verifiability. These considerations have relatively low importance initially as the system concept is established and explored. Over the course of development these gradually become overriding factors. The final phase is very similar to the development cycle for a conventional program, the prototyping having resulted in a sufficiently thorough understanding of the best methods to solve the problem that attention to program efficiency and integrity dominates.

2.3 FAULT MANAGEMENT APPROACH

The discussion on fault management in this section draws on the MSFC power management and distribution breadboard that has been the focus of our Task III efforts. Fault management on the MSFC breadboard facilitates illustration of fault management techniques applicable in the Space Station power management and distribution system. The efforts on the breadboard with its set of data acquisition capabilities have shown that for the fault management system to be non-trivial, it should handle multiple simultaneous failures as well as failure in different types of components (e.g., sensors, switches and cables). If only single point failures or only failures in one type of component are considered then any given set of symptoms can be caused by only one failure.

It appears that the best approach to managing the more complex fault scenarios during Space Station development - and very likely operation as well - would incorporate causal reasoning in an object oriented paradigm. The power system configuration is expected to change and evolve throughout Space Station development as different configurations are examined and new requirements imposed. As discussed below, if diagnostic knowledge is hard-coded, a configuration change can result in substantial re-coding with the concommitant re-testing.

A causal reasoning approach obviates the bulk of this problem. It would require only that the changes in the configuration be reflected in the software ("causal") model of the configuration that is used to reason with. Additionally, changing components (e.g., using a switch with different trip characteristics) would only require changing the description of that component in the model. In the course of subsequent analyses, the system would deduce the effect of that change on system behavior.

A similar situation arises when analyzing or diagnosing a previously degraded system. A traditional, hard-coded diagnostic system would simply fail when considering the degraded parts of the network. (It should be noted that there are some more or less elaborate workarounds for this).

A causal based diagnostic system would take the situation in stride. If it had identified the degradations while troubleshooting, it would have updated its model of the network appropriately. When it next needed to examine system operation, it would base its considerations on the up-to-date model, using it in the same way it had previously used the original model of the 'healthy' network. The causal system would of course know that the network was degraded: what is important is that it would reason on the basis of what is (as reflected in its model) rather than from pre-compiled assumptions about what should be.

The essential point of this section is that the most flexible and adaptable diagnostic system for Space Station power network control is one that uses a significant amount of causal reasoning during the actual diagnosis (during "runtime" Space Station operation rather than in advance), to adequately characterize the problem, determine its possible causes and evaluate those possible causes.

2.3.1 Aspects of Diagnosis

This section discusses the three general steps in the automated diagnosis. The next section, 2.3.2, presents five options for performance of these steps, while section 2.3.3 delineates the advantages and disadvantages for each of the options. Finally, section 2.3.4 examines the fifth option in detail - the option indicated in the preceding paragraph that uses causal reasoning to perform the analyses at runtime for diagnosis. This is the chosen approach for the Fault Recovery and Management Expert System (FRAMES) of the MSFC breadboard.

When a human diagnoses a power system problem manually, he or she uses a great deal of sensory data that will not (at least initially) be available to an automated diagnostic mechanism on board Space Station. Not having such direct information (such as a burned wire, etc.) available, an automated troubleshooter must structure its analysis differently. It must use available sensor data to determine if a problem exists, what the problem is, identify possible causes and then evaluate those possible causes to determine if any actually did cause the problem.

The existence of any symptom indicates that there is a problem to ba diagnosed. Defining that problem is, in effect, identifying all the symptoms that have occurred. A symptom is any anomalous state in the network (or equivalently, any anomalous response by a component to an input or change in the network status). Most anomalous states will be reported by the lowest level processing units - a tripped switch is flagged as anomalous data and sent for analysis to FRAMES which is at a higher hierarchical level. Some anomalous states will not be identified at the lowest level - for example, when a switch should trip but does not, or in a soft fault scenario, when a sensor with its circuitry has been consistently reporting slightly erroneous data. The higher level must have capability to ascertain such cases and add them to the reported anomalous data to form the 'symptom set'. Defining the problem is establishing this symptom set.

It is important for understanding the assessment of the pre-compiled approaches in the succeeding sections to look at how symptom sets can be grouped together into classes. Different symptom sets could result from the same fault depending on the pre-fault status of the network. For example, in Figure 2-1, if the PPDA RCCB trips (due to, e.g., overcurrent from a hard cable short), all previously closed switches below it (the PPDA RPCs and the LC RPCs) should trip on under-voltage. (One of the characteristics of RPCs and RCCBs is that they will trip open on under-voltage as well as over-current.) The switches reported as having symptoms are those that were closed, so the number of symptoms, and hence the symptom sets, vary in accordance with which switches were open or closed.

It is thus possible to group together into a 'class' all symptom sets that could result from a given fault (or set of faults) in accord with the different possible pre-fault states of the network. Note that some classes could be caused by more than one fault. In addition, a given symptom set could result from several different types of faults and thus in general be a member of several different classes. At run time in the pre-compiled approaches, the diagnostic process would have to consider all the causes of all the classes that the identified symptom set was a member of.

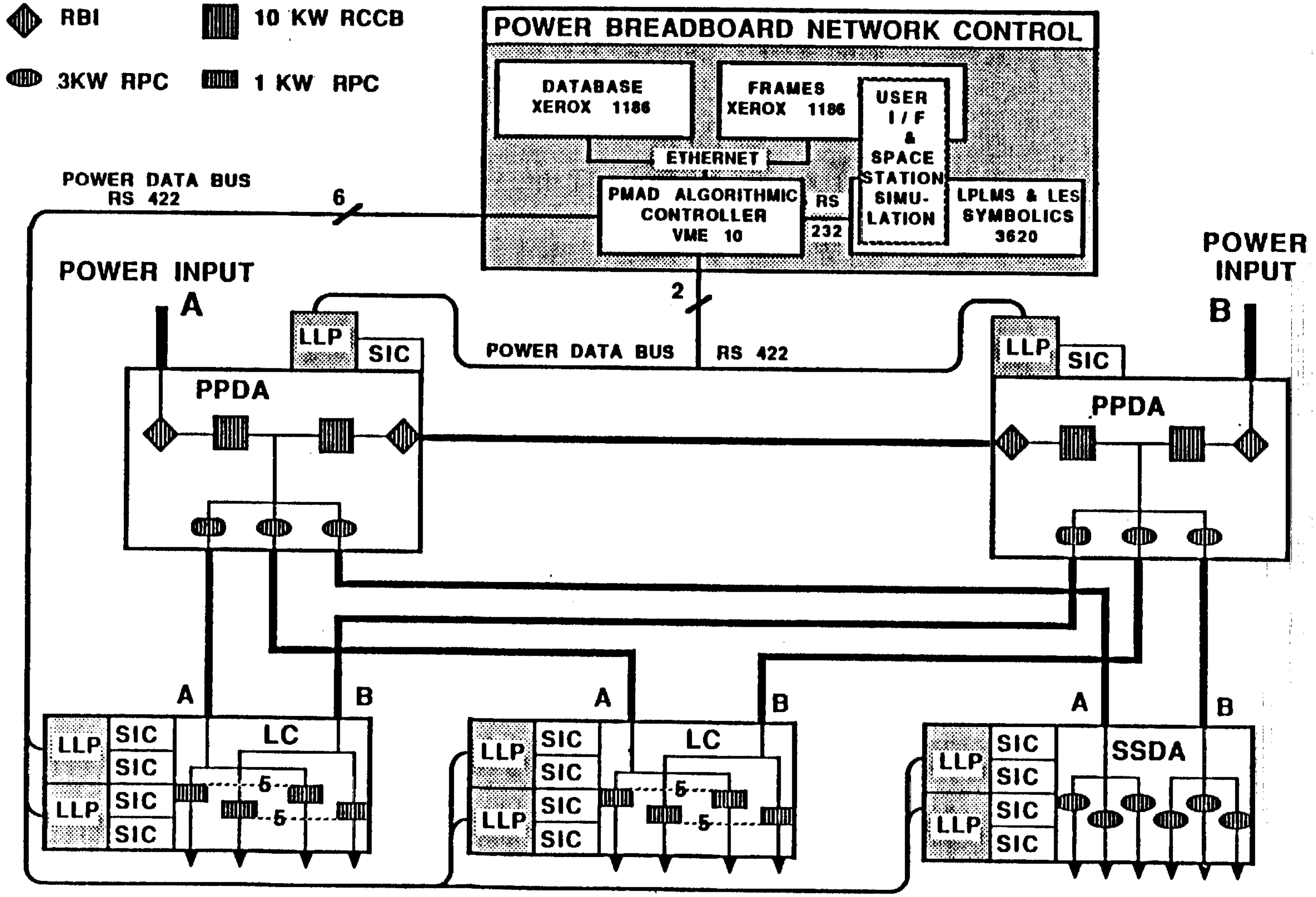


FIGURE 2-1 BREADBOARD BLOCK DIAGRAM FOR AUTOMATION APPROACH VERIFICATION

Once the symptoms have been identified, an automated troubleshooter must determine what kinds of failures could have caused those symptoms. The complexity of this analysis depends on the complexity of diagnosis being done. Power system failures could be in the power control, communications or processing area. Considering only power control, failures could arise in sensors, switches or cabling. If the troubleshooter is only considering one of these types, e.g., switch failures, rather than all three, then any given set of symptoms that could arise in the MSFC breadboard could have been caused by one and only one type of fault. Similarly, if only single point failures are addressed, then any set of symptoms could have resulted from one and only one type of fault. These two situations would make for a rather simplified system.

In contrast, if multiple failures are addressed and all three types of control faults considered, then a symptom set in the MSFC breadboard could have been caused by up to eleven (11) distinct types of faults (and combinations of faults). Thus, FRAMES is intended to handle multiple faults and combinations of sensor, switch and cable faults. FRAMES will address both hard and soft faults and at some later point will examine trending conditions and incipient failures (which can either be or or not be the preludes to soft faults). Causes can also be considered to be grouped into classes. A class might be, for example, that there was a hard short below an LC RPC and the LC RPC failed to trip on overcurrent (this obviously involves two independent failures). The class does not refer to which LC RPC is involved, only that one is. The individual causes that are members of this class would each refer to a different LC RPC. One member might be: "there was a hard short under LC RPC-1 and LC RPC-1 failed to trip". Other members would specify the other LC RPCs on the involved bus.

Once the set of possible causes (or, more accurately, the set of classes of causes) has been identified, evaluation of them should be relatively straightforward. FRAMES' primary means of evaluation is by manipulating switches.* The following simplified example illustrates an evaluation:

> Assume a shorted condition in the load and a failed LC RPC such that overcurrent does not "trip" the LC RPC, but that under-voltage does "trip" it as a result of the PPDA RPC tripping on an overcurrent condition. The symptoms would be the same as if there was a hard short in the cabling between the PPDA RPC and the LC RPC. Evaluating the possible causes(s), in this example case, FRAMES would turn on the PPDA RPC (with the lower level affected RPCs in the off condition). If the PPDA RPC does not trip (it would not in our example case because the shorted condition is "below" an RPC in the "off" condition) then a shorted cable between the PPDA and LC can be discounted.

It is not clear that switch manipulation by a fault manager will always be appropriate or even always allowed during operation of the Space Station. There are many considerations, including personnel and

* In a number of soft fault scenarios, FRAMES will also draw on first principles. While the bulk of the discussion and examples presented here focus on hard faults, the critical elements of the approach transfer readily to soft faults.

equipment safety, required load operational scenarios (an electrical load may not be restartable), etc., in the allowance of switch manipulation. However, a mature fault management system can incorporate governing rules and guidelines as well as accept "permission" from a "higher authority" to manipulate particular switches necessary for evaluation.

If no switch manipulation is allowed, however, the diagnostic process could, as a minimum, report the classes of possible causes it has identified which could account for the symptom set. It could prioritize them with respect to likelihood but it could not provide further discrimination. For purposes of this discussion, it is assumed that some switch manipulation is permissible and the process of evaluating possible causes described below uses it. This discussion draws a distinction between calculating how to evaluate a particular possible cause (and thus testing them one at a time) and determining what each step of a standardized (tree-) search procedure would reveal about the possible causes. If intermediate steps need not be presented to the user, the latter approach appears to be more appropriate unless it is decided that the power network might become (in the graph theoretic sense) cyclic or non-hierarchical in significant ways. This is addressed in section 2.3.4.3.

2.3.2 Options For Performing Diagnostic Processes

There are five (5) main options for performing the key diagnostic analyses. Advantages and disadvantages are discussed in section 2.3.3.

(1) Perform all the analyses ahead of time by hand.

- o Enumerate all symptoms and all causes, not just their classes. Enumerate all possible symptom sets.
- o For each symptom set, numerate all possible causes.
- o For each cause identify how to evaluate that cause (which switches to manipulate to get evidence for or against).

At runtime then, it would only be necessary to identify all the symptoms. From that, all possible causes would immediately be known, e.g., via table look-up, as would all the means of discriminating between possible causes.

(2) Perform all analyses ahead of time by machine.

- o Same as (1) but the analyses are automated.

Runtime behavior would be the same except under degraded conditions where the automated version might be able to handle the situation and the manual very likely could not. See comment in next section.

(3) Perform the analyses by class ahead of time by hand.

- o Enumerate all possible classes of symptom sets.
- o For each class, enumerate all possible classes of causes. For each class of causes, identify how to evaluate the class and when necessary how to find the appropriate member within it.

Runtime processing would be somewhat different than (1), since the pre-compiled analysis deals with classes, not individual symptom sets or causes. The notion of classes is introduced largely as a way of reducing the exhaustive enumeration of (1) and (2) above, shifting some of the work to runtime. During actual diagnosis, it would be necessary to identify all the symptoms and then determine what class that symptom set belongs to. Once the class of symptom sets has been determined, the classes of possible causes would be available via look-up from pre-compiled analyses. Depending on how the evaluation analysis was done, it may be possible to search through the class of causes or it may be necessary to search through the individual members of the classes.

(4) Perform the analyses by class ahead of time by machine.

- o Same as (3) but analysis is automated.

Runtime behavior would be the same as (3) except that it would likely be much easier for the automated version to handle degraded conditions than the manual. It is not clear that the manual version could - see comment in following section.

(5) Perform all the analysis during actual diagnosis.

- o At runtime (obviously), establish the symptom set.
- o From the symptom set (rather than its class), determine the possible causes. Note that this does not require finding the class of the symptom set or the class(es) of causes. This process works by a causal analysis of the specifics rather than using the general classes
- o Determine how to evaluate the causes and do so. In many cases this will mean simply executing a standard evaluation procedure and when anomalies appear, determine what cause is being implicated.

2.3.3 Advantages and Disadvantages of Options

The five options described in the previous section each have various advantages and disadvantages. These focus on such issues as complexity of pre-runtime analyses, speed at runtime, data management at runtime and flexibility both in moving to new network configurations and in adapting to failures or runtime changes in configuration.

(1) Perform all analyses ahead of time by hand.

ADVANTAGES:

o Fast runtime execution (depending on how diagnostic information generated by analysis is handled at runtime).

DISADVANTAGES:

o Possibility of overlooking possible symptoms or causes due to the enormity of the analysis.

o Possible significant problem managing all diagnostic information generated in analyses. The amount of this information can be prohibitively large. If the network had 15 LC RPCs per bus, some of the 35 classes of symptom sets can have on the order of 10^{13} members. Effectively managing this amount of information on a time critical fashion is basically unfeasible.

o Analysis must be re-done (or re-verified) completely if move to a different network (or configuration).

o Analysis must be re-done (or re-verified) after a fault has occurred. To do this by hand while Space Station is operating is not feasible except under trivial failures.

(2) Perform all analyses in advance by machine.

ADVANTAGES

o Fast execution time (depending on how diagnostic information is handled).

o Pre-runtime analysis is much faster and more likely to be complete and reliable than (1).

DISADVANTAGES

o Possible significant problem managing all the diagnostic information generated in analyses. See comments under (1).

o Computation to perform analysis is substantial and must be repeated if move to different network (or configuration).

o Computation to perform analysis is substantial and may need to be repeated (off-line while Space Station continues to operate) after every failure.

o Again, there may be programmatic solutions to the re-computation, e.g., it would be possible to restrict the re-computation to address only areas effected by the failure. These may or may not be sufficient.

(3) Perform analyses by class ahead of time by hand.

ADVANTAGES

o Fast execution time.

o Depending on how the fault diagnosis information is handled in (1) and (2), execution time for (3) and (4) could be close to or much slower than execution time in (1) and (2).

o Execution time for (3) and (4) may or may not be slightly faster than for (5).

DISADVANTAGES

- Possibility of overlooking classes of possible symptoms or causes.
- Still requires runtime classification of symptoms and searching within cause classes for individual causes. These two steps are not part of (5), which "adds" to the relative speed of (5).
- Overlapping class problem - symptom sets may belong to more than one class, multiplying the search space and slowing execution time unless handled with finesse.
- Analyses must be re-done (or re-verified) completely if move to a different network (or configuration).
- Analyses must be re-done (or re-verified) after a fault has occurred. See comments under (1).

(4) Perform analyses by class ahead of time by machine.

ADVANTAGES

- Fast execution time. See notes under (3).
- Pre-runtime analyses much faster and more likely to be complete and reliable than (3).

DISADVANTAGES

- Still requires runtime classification of symptoms and searching within cause classes for individual causes. See note under (3).
- Overlapping class problem. See comments under (3).
- Must repeat computation if move to different network (or configuration).
- May need to repeat computation after every failure. See comments under (2).

(5) Perform all analyses necessary during actual diagnosis.

ADVANTAGES

- Greatest flexibility of all options, both in adapting (automatically) to identified runtime failures and in moving to new networks (or configurations).
- More likely to identify all possibilities and therefore find 'obscure' faults - than (1) or (3).
- Minimizes problem of managing diagnostic information - can be concisely formulated in an object oriented paradigm.
- Substantive explanation is easier than under the other approaches.

DISADVANTAGES

- May be slower than pre-compiled approaches at runtime.

The relative speeds of the five options are a function of:

(a) How the enormous volume of diagnostic information is handled at runtime in (1) and (2).

(b) How fast the classification and declassification processes are in (3) and (4).

(c) How much the heuristics will reduce computation time in (5).

Given the substantially greater flexibility along with these trade-offs in execution time, it seems prudent to pursue option (5).

2.3.4 The Use of Option Five in FRAMES

As discussed earlier, the runtime process consists of three main parts:

1) establishing the symptom set,
2) identifying possible causes for the symptom set, and
3) evaluating the possible causes.

The following subsections examine how option 5 is used in FRAMES.

2.3.4.1 Establishing the Symptom Set

A proposed procedure for establishing the symptom set is:

(1) Gather data
(2) Identify the highest level in the network with a reported anomaly
(3) Predict what other symptoms should be present given 2, reasoning from the pre-fault network configuration
(4) Compare the predictions in 3 with the reported anomalies in 1
(5) The symptom set is the reported anomalies in 1 plus the differences from 4

Gathering data is an algorithmic task allocated to the lowest level processes. It is assumed that the lowest level processes which control the switches identify any tripped switch. All these are flagged and reported to the higher level control, FRAMES. Using the structural information about the network represented in its object oriented causal model, FRAMES does a standard search to find the flagged component that is closest to the power source ("highest" in the network). Typically this might be a switch reporting tripped on overcurrent. Using the same structural information along with information about current flow, voltage dynamics and the behavior of network components, FRAMES might reason as shown in the following simplified example:

> A tripped switch means that there should be zero voltage on the load side of the switch. This zero voltage condition should hold along the cable beneath the switch down to the next level of subordinate components, in this case switches. Each of these switches should have zero voltage at their inputs which should cause all of them to trip on under-voltage.

It would then add these switch "tripped" conditions to its predicted symptoms and continue in the same fashion moving downward through the network through each of these switches. The comparison in 4 and set union in 5 are straightforward.

Note that the FRAMES only needs to know about how a component, be it a cable, switch or sensor, responds to an input - that is, how it changes state and what its output will be. A significant advantage to this is that changes in the network are noted in the model by changes in the behavior of components and their connections - all of which are merely data for the process that is propagating effects through the network and looking at global activity/behavior. If an old or faulty switch is replaced by a switch with different trip characteristics, for example, FRAMES will at the appropriate time consider whether that new switch will trip under a particular set of circumstances, independent of whether any other switch will or not.

2.3.4.2 Identifying Possible Causes

Once the symptom set is identified, possible causes can be established using a form of causal reasoning known as constraint suspension. In this method, the healthy functioning of each aspect of a component is considered a constraint on the system behavior. A sensor, for example, imposes at least two such constraints: It measures (and reports) properly, and it does not significantly alter system function (e.g., it does not significantly reduce current flow or cause a voltage drop). A switch imposes several: It permits no (or negligible) current flow when open, it trips on current flow above a certain level, it trips on voltage below a certain level, etc.

The various constraints "imposed" on the system by its components can be suspended, singly or in sets, to see if that malfunction could cause the identified symptoms. This can be done to different degrees. For example, the constraint that a cable flows current from its input to its output without any intermediary shunt could be suspended completely, creating a hard short condition, or only partially, creating a soft fault situation.

Beginning with the highest component in the network to evidence an anomaly (e.g., a tripped switch or a consistently high sensor reading), FRAMES would look at the constraints imposed by that component. It would select one, suspend it, and then propagate the effects through the network model, "tripping" switches and "reporting" sensor data in an internal simulation. (Note: both the action and depth of propagation could be controlled to minimize computation.) If the effects generated in this simulation match the actual symptom set identified in the previous step (i.e., the symptoms that resulted when the actual fault occurred), that suspended constraint is added to the set of possible causes. If the effects do not match, that constraint is not added. In either case, the analysis goes on to the other constraints of that component, then to the constraints imposed by subordinate components in a breadth first, partially constrained search. Further, if the effects are only a subset of the symptom set, that constraint is given consideration as one of a set of multiple faults and (at the appropriate time) FRAMES searches for other constraints that would non-trivially complement this one and together, in a multiple fault scenario, account for the symptom set.

An analysis by constraint suspension could be done poorly using a brute force method and result in gross inefficiencies. It might, for example, suspend constraints at locations in the network model unrelated to the symptom-reporting area - in an extreme case, it might look at bus "B" when all symptoms are on bus "A". As another example of such inefficiency, it might suspend a constraint that could in no way produce any of the symptoms in question.

The analysis needs to be guided. Some heuristics for this include:

- o start at the highest level with identified anomalies and work away from the source,
- o search in the location of symptoms (first),
- o examine faults in groups of equal liklyhood, e.g., evaluate single point causes before identifying multiple causes, and
- o depth of search (in terms of levels of subordinate switches) is equal to the number of faults being considered.

2.3.4.3 Evaluating Possible Causes

Assuming for this discussion that manipulation of switches is allowed in order to evaluate possible causes, the underlying analysis for evaluation is closely related to the causal analysis used in identifying possible causes. The relationship is sufficiently close that under certain circumstances, the identification and evaluation might happen at the same time. Switches would thus be manipulated and, based on the results, the nature of the fault or cause would be deduced, rather than vice versa. Such cases should be exploited since execution time as discussed in 2.3.3 would be considerably reduced and would make option 5 much closer in speed to the other options. As noted in 2.3.3, however, this would remove the possibility of cleanly presenting information concerning possible causes.

Assuming, however, that evaluation is done after possible causes have been identified, the problem is: given a possible cause, how can evidence be obtained concerning it? If all raw data analysis has already been incorporated, evidence can be obtained only by bringing power to the locale of suspected failure. Thus: what switches need to be closed to connect this locale to the power source and, at the same time, what other switches need to be opened to isolate that connection and minimize unintended interactions?

This question can be addressed by a fairly straightforward bounded search using the structural connections reflected in FRAMES' causal model of the network. Because this is straightforward and since the maximum depth of search will not be more than a handful of levels (3 or 4 in most network topologies), the process will take little computation time. Actual switch manipulation time is considered constant across the five options.

2.4 HARDWARE DEVELOPMENT DEFINITION

Hardware requiring development for the automation of CM/PMAD may include sensors for 20 kHz range voltage, current, frequency, power, and power factor (phase angle). Specific sensor development efforts may be required in the areas of accuracy, resolution, and repeatability for the specified types and range. We feel this development may be required to mitigate risks associated with sensing parameters at 20 kHz. We purposely excluded temperature sensors in identification of hardware requiring development in that we believe the technology is well developed for the ranges and uses projected by Task 1 studies. Sensor requirements are highly related to the techniques used to carry out the CM/PMAD identified functions. The development or definition of those requirements may change as the MSFC breadboard development and evaluation efforts of Task III provide data. Our studies currently show that barring 20kHz integration risks, data accuracy in the one percent range will be sufficient for CM/PMAD operation. Furthermore, excepting temperature sensors, the approach considered utilizes voltage and current sensors as the only physical sensing hookup to the power circuitry. Output from these sensor types will also be used in derivation of frequency, power, and power factor values. Hardware development with respect to these additional measurements will be required in the sensing and data conditioning circuitry. We view this situation as typical of any flight hardware requiring development from the initial requirements definition phase. Also included in this category are the following hardware areas:

1) digital support circuitry for switchgear, including interface logic, data acquisition, and command/control logic,

2) microprocessor based load center controllers, and

3) microprocessor based power distribution control unit controllers.

Required development of hardware within the power network control function that supports Space Station growth configurations and that also require advances in technology consists of developing:

1) flight suitable multi-megabyte random access memory capable of the extended mission duration in the space environment with respect to total dose radiation,

2) flight suitable multi-100 megabyte data storage devices also capable of operation in space environments, and

3) a flight suitable computer capable of taking full advantage of the AI techniques used in overall CM/PMAD operation as well as qualified for an extended mission in space environments.

3.0 SUMMARY

Knowledge bases, knowledge base management systems, inference procedures, as well as deterministic algorithms to implement the approach as defined in Task I were defined. Application areas requiring a knowledge base approach included scheduling, load priority management, fault prediction, fault management. Currently available knowledge base management systems prove useful in the early stages of the development of an expert system, but have far too much overhead in both storage and computation time to be used in the final flight version systems. The recommended approach for development of these final versions would be to implement application specific tools in Lisp or Ada and then use these tools to implement the knowledge based system. Roles for deterministic algorithms were also defined. The deterministic algorithm approach was recommended for distribution management, load monitoring and shedding, signal conditioning, lowest level fault detection, and communication functions.

A methodology for producing the above software requiring unique development was presented. The approach consisted of three phases - initial problem analysis and prototyping, second generation prototyping and knowledge refinement, and deliverable system development.

Fault management studies identified five options for approaching the diagnosis problem. The most flexible and adaptive of these, and the one with the greatest potential for sophisticated explanation, has been chosen for use in FRAMES. This approaches uses a heuristically guided causal reasoning system embedded in an object oriented paradigm.

Fault management studies resulted in recommending a knowledge based system in an advisory role for IOC with deterministic algorithms at the lowest level for condition exception handling and limits verification. The approach provides the capability for growth in the fault management function for Space Station growth configurations. The approach also facilitates incorporation of new technologies, e.g. AI hardware and software, as they become available.

Required hardware development necessary for implementation of the automation approach of Task I was defined. Results divided the hardware development into two general areas. The first was development typical of programs with flight hardware. Sensors, data acquisition and control circuitry, and digital interfacing logic to effectors were grouped into this category. Also in this category we identified microprocessor based controllers for both the load center and the power distribution control unit. The second area was defined as hardware development that requires advances in a technology. Flight suitable multi-megabyte random access memory, flight suitable multi-megabyte data storage devices, and a flight suitable computer capable of taking full advantage of AI techniques were identified as requiring development in the second category.

4.0 REFERENCE

This Task II effort was preceded by the referenced Task I, CM/PMAD System Automation Plan Definition, under the same MSFC contract NAS8-36433. Those Task I efforts in characterizations and plan definitions served as an embarkation point for Task II. Full documentation for a more detailed understanding of those Task I efforts and results can be found in the following report:

W.D.Miller, et. al., "Space Station Automation of Common Module Power Management and Distribution: Task I Study Report,"
Martin Marietta Report MCR-86-583, July 1986.

APPENDIX III – TASK IV DATA

FOREWORD

Martin Marietta Denver Aerospace is submitting this report to NASA's George C. Marshall Spaceflight Center in fulfillment of the requirements of contract NAS8-36433, Task IV, item 2. The report details the results of our evaluation of the government-proposed requirements, hardware, software support packages, and operating system for the two computer systems that will interface with the common module power management and distribution (CM/PMAD) controllers.

CONTENTS

		Page
I.	TASK-IV STUDY CONCLUSIONS	I-1
II.	REQUIREMENTS ANALYSIS	II-1
A.	Both Systems	II-1
B.	Primary Computer	II-3
C.	Stimulus Computer	II-3
III.	RECOMMENDATIONS	III-1

Table		
I-1	GDI System Components	I-2
I-2	VME/10 System Components	I-4

I. TASK-IV STUDY CONCLUSIONS

We do not recommend the computer system defined by General Digital Industries, Inc., for the common module power management and distribution (CM/PMAD) breadboard. Instead, we recommend an alternate configuration based on the Motorola VME/10 development system.

We reached this conclusion in completing task IV, item one, of the contract statement of work. Under this task we evaluated the government-proposed requirements, hardware, software support packages, and operating system for the two computer systems that are to interface with the CM/PMAD controllers. The system we evaluated was defined in a study conducted by General Digital Industries (GDI) and was documented in their report Space Station Power System Control Study Final Report.

The primary reason we recommend a different configuration is cost: the GDI version costs $120,747 with the modifications we found necessary (Table I-1). The VME/10 system (Table I-2) costs only $65,585. This is an important consideration, because the contract has only $53,000 allocated for the system, and purchase of the GDI system would require additional funding or rescoping of other tasks.

However, there are additional reasons for recommending a different configuration. Specifically, although the GDI system appears to meet the major requirements for the breadboard, it is incomplete. For example, for the 18 Opto-22 circuit cards, no provision is made for a cable to connect the cards to the computer or for a card cage or other housing for the cards or a power supply to power the cards. Similarly, although Ethernet hardware was specified, the required Towernet software was not.

A related problem is that NCR has changed its product line since GDI wrote its report. The model 1632 is no longer manufactured (although used equipment is available), and the features NCR now includes with the basic Tower XP differ from what the report describes.

Further, some of the parts do not appear to work well together. For example, the Unix-V operating system of the GDI configuration is designed for an office environment -- accounting, spread sheets, data base management, etc. -- not for real-time control. GDI therefore specifies the polyFORTH language. However, according to Ms. Sheree Krawetz at FORTH, Inc., the polyFORTH software provides its own operating system with its own file structure, utility software, I/O drivers and device handlers. These are not compatible with Unix. In fact, once polyFORTH has written on the Winchester disk, it cannot be used for Unix without being erased first. The result is that the entire Unix environment will be rendered useless for developing the automation software. And because the Ethernet software runs under Unix, it will be very difficult to use Ethernet in the control software.

Table I-1 GDI System Components

PRIMARY COMPUTER

Quantity	Manufacturer	Model	Description	Price	Total
1	NCR	Tower XP	Computer, 1MB Memory, 1 Flexible-Disk Drive, 1 Streaming Tape Drive, 1 Winchester Disk, 1 High-Performance RS-232C Interface, and 1 Printer Interface	$16545	$16545
1	NCR	CWD509-2001-00FD	Unix Operating System	755	755
1	NCR	CWD509-2008-00FD	FORTRAN Compiler	650	650
1	NCR	CWD509-2009-00FD	Pascal Compiler	700	700
1	FORTH, Inc.	pF32/NCR	FORTH Software	3200	3200
1	FORTH, Inc.		License to Download to Controllers	800	800
1	NCR		Towernet Software	995	995
1	Excelan	EXOS 201, Model 3	Ethernet controller	2095	2095
1	Excelan		Cable for Ethernet	155	155
1	Excelan		Ethernet Transceiver	495	495
1	Data Translation	DT712-64DI-PGH	64 Diff. A/D Channels	1780	1780
1	Data Translation	DT724	4-Channel D/A	730	730
1	Data Translation	EP067	4 Cables	260	260
1	Data Translation	DT705	Termination Panel	330	330
1	Intel	iSBC519	72 Chan TTL I/O	660	660
2	Opto-22	PB24Q	Relay Driver Socket Bd	108	216
6	Opto-22	ODC5Q	Quad Relay Driver	35	210
6	Opto2	IDC5BQ	Quad Relay Sensor	45	270
1	Tektronix	4125P+Opt 19	Computer Display	20850	20850
1	Tektronix	4691	Color Graphics Copier	12950	12950
1	Tektronix	4510+Opt 30	Graphics Rasterizer	4495	4495
1	Hayes Microcomputer Products	Smartmodem 2400	2400-baud Modem	899	899
1	TBD	TBD	Cables	200	200
			TOTAL		70240

Table I-1 GDI System Components (Cont.)

STIMULUS COMPUTER

Quantity	Manufacturer	Model	Description	Price	Total
1	NCR	Tower XP	Computer, 1MB Memory, 1 Flexible-Disk Drive, 1 Streaming Tape Drive, 1 Winchester Disk, 1 High-Performance RS-232C Interface, and 1 Printer Interface	$16545	$16545
1	NCR	CWD509-2001-00FD	Unix Operating System	755	755
1	NCR	CWD509-2008-00FD	FORTRAN Compiler	650	650
1	NCR	CWD509-2009-00FD	Pascal Compiler	700	700
1	FORTH, Inc.	pF32/NCR	FORTH Software	2400 *	2400
1	Data Translation	DT724	4 Channel D/A	730	730
1	Intel	iSBC519	72 Chan TTL I/O	660	660
1	Opto-22	AC7	RS-232C to RS-422 Conv.	85	85
16	Opto-22	PB16MS	Multiplexer	260	4160
256	Opto-22	ODC5	Relay	7.90	2022
1	Tektronix	4125P+Opt 19	Computer Display	20850	20850
1	TBD	TBD	Power Supply, 5vdc	75	75
1	TBD	TBD	Pwr Supply, +/- 12 vdc	75	75
1	TBD	TBD	Cables	200 est	200
1	TBD	TBD	Card Cage, Housing, Connectors for Relays and Multiplexers	600 est	600
			TOTAL		50507

TOTAL, BOTH COMPUTERS: $120,747

* Second-system price break shown.

NOTE: Prices are based on distributor price lists, data from the GDI study, vendor advertisements and catalogs, and informal quotations received by telephone. They do not include Martin Marietta G&A, shipping/freight charges, or receiving inspection. Actual cost may therefore differ from the amount shown.

Table I-2 VME/10 System Components

PRIMARY COMPUTER

Quantity	Manufacturer	Model	Description	Price	Total
1	Motorola	M68K102D1	Computer, O/S	$16530	$16530
1	Motorola	MVME330-VX	Ethernet	3300	3300
1	Excelan		Cable for Ethernet	155	155
1	Excelan		Ethernet Transceiver	495	495
1	Motorola	MVME222-1	1 MB Memory	1750	1750
1	Motorola	MVME400	Dual RS-232C	395	395
1	Motorola**	MVME350	Streaming Tape I/face	1450	1450
1	Archive **	FT60	Streaming Tape Drive	1295	1295
1	Archive **	SC199-2	Streamer Controller	375	375
1	Diablo	C150	Color Inkjet Printer	998	998
1	Motorola	M68KVMPRTCE	Printer Cable	125	125
1	Motorola	MVME605	4-Channel D/A	675	675
1	Motorola	MVME600	8-Chan A/D Master	750	750
2	Motorola	MVME601	8-Chan A/D Expander	350	700
1	Motorola	MVME410	Printer Port	350	350
1	Motorola	MVME340	TTL I/O, Timer	1125	1125
3	Motorola	MVME625	Relay Driver	340	1020
3	Motorola	MVME620	Digital Input	325	975
1	Motorola	M68VVXBPASCAL	Pascal Compiler	995	995
1	Motorola	MVME922	Expansion Backplane	290	290
1	US Robotics	Password	Modem	450	450
			TOTAL		34198

** MVME350 is not available until Nov/Dec. An alternate configuration using MVME319 and Cipher "Floppy Tape" is now available at a similar price but requires changing tapes to archive a full 40 MB disk.

Table I-2 VME/10 System Components (Cont.)

STIMULUS COMPUTER

Quantity	Manufacturer	Model	Description	Price	Total
1	Motorola	M68K102D1	Computer, O/S	16530	16530
1	Motorola	MVME350	Streaming Tape I/face	1450	1450
1	Archive	FT60	Streaming Tape Drive	1295	1295
1	Archive	SC199-2	Streamer Controller	375	375
1	TBD	TBD	Cables	200 est	200
1	Motorola	MVME410	Printer Port	350	350
1	Motorola	MVME605	4-Channel D/A	675	675
1	Motorola	MVME340	TTL I/O, Timer	1125	1125
1	Motorola	MVME202	512 KB Memory	1395	1395
1	Motorola	M68VVXBPASCAL	Pascal Compiler	995	995
1	Opto-22	AC7A	I/face w/ Pwr Supply	190	190
16	Opto-22	PB16MD	Relay Multiplexer	260	4160
256	Opto-22	ODC5 *	Relay, 60vdc, 3A	7.90	2022 *
1	TBD	TBD	Power Supply, 5vdc	75	75
1	TBD	TBD	Card Cage, Housing, Connectors for Relays and Multiplexers	550	550
			TOTAL		31387

TOTAL, BOTH COMPUTERS: $65,585

* Relay part number will depend on power type. Part number shown allows comparison with GDI system.

NOTE: Prices are based on distributor price lists, data from the GDI study, vendor advertisements and catalogs, and informal quotations received by telephone. They do not include Martin Marietta G&A, shipping/freight charges, or receiving inspection. Actual cost may therefore differ from the amount shown.

Moreover, a few of the items in the GDI system seem to be inconsistent or inappropriate. In particular, GDI specifies two flexible disk drives on each computer to allow copying disks. The second drive is not necessary for copying disks, however, because information can be copied to the Winchester disk and then back to as many flexible disks as required from the Winchester. In addition, GDI specifies two DMA channels with a minimum of 10 MB/s transfer rate but specifies no use for them. We assume they are intended for Ethernet and Winchester interfaces. If this is the intent, it is sufficient to state that the system will provide Ethernet communications and Winchester disk storage. Similarly, GDI has specified streaming tape backup capability for only one of the two computers. Both have large-capacity Winchester drives. If it is reasonable to have backup capability on one, it should be reasonable for the other as well.

Finally, the GDI configuration does not appear to meet all the "mandatory" requirements set forth in the report. For example, the report calls for 150 ms flexible-disk access time; the NCR computer provides no better than 181 ms. The report calls for at least 40 MB of Winchester disk space; the NCR computer provides 39 MB. The report requires D/A output impedance under 1000 ohms; the selected devices provide 4000 ohms. The report calls for a library of graphics routines, but no such package is provided in the software specified. And, although such a package is available from a third party (Precision Visual Graphics), it is designed to work only with FORTRAN programs. The polyFORTH system is incompatible with this software and would require a separate graphics package. Mr. Ed Boykin, the Denver, Colorado, NCR representative, was not aware of any support for graphics under FORTH. Similarly, the report states as a requirement that the RS-232C ports be configured by means of "DIP" switches. The NCR computers do not have switches; they are configured through software. In most cases, these differences are inconsequential. For example, the access time for the flexible disk need not be specified at all, because the flexible disks will not be used during breadboard operation. They are provided only as a means for getting software onto and off of the Winchester disk.

We assume from these discrepancies that the term "mandatory" is not to be taken literally, and in view of the current uncertainty about exactly how these computers will be used, this appears to be a reasonable assumption. In addition, we assume that phrases that appear to be extracted from a particular vendor's sales literature are soft requirements, e.g. that the screen editor must have "powerful search, replace, and formatting commands."

Based on these assumptions, we have identified the previously mentioned VME/10 system. This system appears to meet all essential requirements for CM/PMAD at a considerably lower cost. An analysis of how this system meets requirements is presented in Chapter II.

The primary differences between the two systems are physical appearance, operating system software, graphics resolution, and expansion bus structure. Specifically:

1) Physical Appearance. The GDI system is based on the NCR Tower computers, which stand on the floor. The VME/10 system is a desk-top unit. The shapes and dimensions of the components of the systems differ, and although total volume is similar, the GDI system is somewhat larger. The VME/10 system has a smaller screen (14 inches vs. 19 inches for the GDI system);

2) Operating System Software. The GDI system uses Unix and polyFORTH; the VME/10 system uses VersaDOS. VersaDOS has a hierarchical file system that resembles that of Unix, but it is less flexible and does not allow for directories within directories with nesting to any depth as Unix does. Unlike Unix, VersaDOS is designed for real-time applications and provides high-level language (Pascal) support for such multi-tasking functions as intertask communication, interrupt-service routines, setting of task priorities, suspending tasks, spawning new tasks, etc;

3) Graphics Resolution. The GDI system has approximately 50% better resolution, but both systems appear adequate for the CM/PMAD application. The VME/10 system has a resolution of 600 x 800 pixels;.

4) Expansion Bus Structure. The expansion bus of the NCR computers is Multibus-I; the VME/10 system uses a VME bus. There is no clear superiority of either over the other. Multibus-I is an older standard and therefore has a wider variety of products available for it.

The VME/10 system has the advantage of being usable for compiling programs for read-only memories in the controllers of a distributed-intelligence architecture. Such software development on the GDI system would require each controller to have at least partial polyFORTH support, because FORTH code is not compiled entirely to machine code and requires run-time interpreters from the polyFORTH system.

Both systems are based on the Motorola 68010 microprocessor; both provide high-resolution color graphics terminals and color inkjet printers. Neither system is better than the other in all respects. However, we believe the VME/10 system provides all necessary features at a much lower cost than the GDI system.

II. REQUIREMENTS ANALYSIS

Listed below are what we believe to be the real requirements for the two computers. In most cases, these agree with those given in the GDI report. We have deviated from GDI's requirements only where GDI's own system did not meet the requirement, where a significant cost saving would result from a small requirement change, or where we saw no basis for a requirement.

A. BOTH SYSTEMS

1. Graphics Terminal. Both systems should have high-resolution color graphics suitable for displaying schematic diagrams for viewing not only by the operator but also others standing nearby. The display should have at least eight colors, and the terminal should be equipped with a detachable keyboard and an interface to the computer.

The VME/10 system has a built-in color graphics terminal with a fourteen-inch screen and 600x800 resolution with eight colors. The detachable keyboard and interface are included in the basic VME/10 system.

2. Color Printer. The two systems should share a high-resolution (at least 80 dots/inch) color inkjet printer. This printer should be capable of printing the terminal screen with a one-keystroke command from the terminal. A page should be printed in less than five minutes.

The Diablo C150 color inkjet printer has a resolution of 120 dots/inch and prints a page in 4.5 minutes. Although the maximum page size is smaller than the 11x17-inch area provided by the Tektronix printer GDI selected, it will draw on a full 8.5x11-inch page, and the cost is $16465 less than the Tektronix printer. Further, the C150 is not restricted to printing what is on the terminal screen as the Tektronix unit is and does not require an expensive terminal as the Tektronix unit does.

Printing the screen with a one-keystroke command is not automatic with the Diablo printer, but according to Clint Bauer, a Motorola technical assistance representative, the VME/10 system can be programmed to read the screen memory, format the information, and send it to a printer in response to a control character or function key.

3. Software Commonality. Both systems should be of the same type and use the same operating system and development software.

This requirement is satisfied by using two VME/10 systems.

4. Modem. The systems should share a modem for transmitting and receiving data over a telephone line. An RS-232C port should be provided to support the modem.

A US Robotics Password modem is specified, and ports are provided in each VME/10 configuration for the modem. Although the modem specified does not have the 2400-baud speed of the Hayes modem GDI specified, it costs approximately half as much. Furthermore, its 1200-baud speed is as high as is commonly supported in systems the breadboard is likely to be connected to.

The MVME400 card provides two RS-232C ports. In the stimulus computer, one of these is used for the Opto-22 relay control equipment. In the primary computer there is one spare RS-232C port.

5. Support Software. Development software provided should include a screen-oriented editor, a high-level language, a file manager, a linker (if needed for the language), and support for real-time multi-tasking software.

These items are provided in the VersaDOS software that comes with the VME/10 system and the Pascal compiler specified. The GDI system does not provide a screen-oriented editor; the polyFORTH editor is line oriented.

6. Real-time Clock. A real-time clock should be provided that has the ability to interrupt the computer at specified intervals and provide the time of day.

A time-of-day clock is built into the VME/10, and three timers with interrupt capability are provided on the MVME340 cards, one of which is in each computer.

7. Synchronization Link. A synchronization link should be provided between the computers.

The purpose of this link and its exact function are not yet identified. However, a spare TTL I/O line from the MVME340 card in each computer should suffice for this link.

8. Disks. A hard disk unit should be provided with each computer. The capacity of each should be at least 40 megabytes. A flexible-disk drive should also be provided in each computer.

These items are built into the VME/10 system specified.

9. Backup Storage. Each computer should be equipped with bulk tape storage for backup storage and archiving.

The MVME350, FT60, and SC199-2 provide a streaming tape backup capability. The entire 40 megabyte capacity of each Winchester disk can be backed up on a single tape with this system.

B. PRIMARY COMPUTER

The following requirements are unique to the primary computer.

1. TTL Output. Twenty-four TTL outputs should be provided for the solar array simulator.

These are provided by the MVME340 card, which actually provides 64 parallel I/O lines, along with interrupt capability.

2. Analog Output. One analog output is required for the solar array simulator.

This output is provided by the MVME605 card, which provides three spare channels.

3. Relay Interface. Twenty-one relay control outputs and 21 relay sense inputs are required for the load-center interface.

The three MVME620 cards provide 4 channels for 10-60 vdc signal monitoring with 2500 v input isolation and protection for input overvoltage and transients. The three MVME625 cards provide 24 outputs of 10-60 vdc with 2500 v isolation and suppression of inductive load transients. Overcurrent protection is provided for 2 A, maximum.

4. Analog Input. Twenty differential analog inputs are required.

The MVME600 and MVME601 boards provide 24 channels of analog input.

5. Ethernet. An Ethernet interface is required.

This interface consists of the MVME330-VX card and the Excelan cable and transceiver. Support software is provided and is included in the cost of the MVME330-VX card.

6. Memory. A minimum of one megabyte of memory should be provided.

The MVME222-1 card provides one megabyte. Additional memory is provided in the basic VME/10 system, but approximately half of it is used for graphics support.

C. STIMULUS COMPUTER

1. Relay Outputs. Discrete digital outputs should be provided for 256 relays.

These outputs are provided by the Opto-22 equipment specified. All 256 relays are controlled through a single RS-232C port.

2. Memory. A minimum 512 kbytes of memory should be provided.

We have specified a 512-kbyte board (MVME202). Additional memory is provided in the basic VME/10 system, but approximately half of it is used for graphics support.

3. Analog Output. One channel of analog output is required.

This output is provided by the MVME605 card. The card provides three spare outputs.

4. TTL Outputs. A minimum of 24 TTL outputs should be provided.

These outputs are provided by the MVME340 card, which includes 40 spare outputs.

III. RECOMMENDATIONS

We recommend purchase of the VME/10 system described in this report. However, we believe it would be wise to defer ordering the Ethernet interface and the relays until the controllers and other portions of the system are better defined. The reason is that either of these items could be found to be inappropriate, and both are expensive.

Specifically, the Ethernet interface with the required support software costs over $3000 for either the VME/10 system or the GDI system. Ethernet was designed for an office environment, not for a real-time control environment, and has characteristics that may not be desirable for CM/PMAD. These include the fact that messages sent over Ethernet may be delayed an unknown amount of time, the weight and size of the coaxial cable required, the high cost and complexity of the interfaces, and the vulnerability of the network to being disabled by a babbling transmitter or a shorted controller interface.

Similarly, the relays may be found to be inappropriate. The relays recommended in the GDI report are rated at 60 vdc and 3 A. Such relays will not be suitable for directly inserting faults in either an AC or DC breadboard because of the low voltage and current ratings. They may or may not be suitable for operating contactors. Postponing the purchase of these parts will allow tailoring the relay drive circuitry to the requirements of the breadboard.

Both systems provide analog and TTL outputs to the solar array simulator. The functions these outputs perform are not yet specified. These interfaces should be defined as soon as possible, certainly before software development begins.

Finally, we recommend addition of a small dot-matrix printer to either system for such functions as printing program listings during software development and rapid printout of data. We believe this will be beneficial because color inkjet printers print very slowly -- over four minutes per page for either model mentioned in this report. The cost of a typical small dot-matrix printer is less than $400.

APPENDIX IV – FUNCTION DEFINITIONS

13.0 FUNCTION DEFINITIONS

13.1 Power Conditioning

Of the types of CM input power considered, none is directly suitable for CM/PMAD distributed logic circuitry (microprocessors, RAM modules, etc). Clearly, some power conditioning must be performed as part of the CM/PMAD task.

13.1.1 Logic Power Conversion

Logic power conversion includes voltage transformations and ac/dc conversion.

13.1.2 Logic Power Conditioning

Logic power conditioning includes dc voltage regulation and filtering.

13.2 Power Distribution

In performing this function, CM/PMAD provides the actual electrical paths for power to flow from place to place within the CM, or physically prevents power from flowing.

13.2.1 Circuit Protection

Circuit protection is the actual, physical defense of the mechanical and electrical integrity of CM/PMAD circuit elements.

13.2.2 Load Switching

Load switching includes the connection of loads to or disconnection of loads from nearby electrical buses.

13.2.3 Bus Switching

This is the connection of electrical buses to or disconnection of electrical buses from one another.

13.3 Power Network Control

The equipment that will perform Function 13.1, Power Conditioning, will probably be self-regulating. The equipment that will perform Function 13.2, Power Distribution, will need to be told what to do and when to do it; Function 13.3, Power Network Control, will provide that guidance.

13.3.1 Distribution Management

It is assumed that before this function is performed, Function 13.3.2.2, Onboard Scheduling has first determined which loads are to be supplied with electrical power. The function of distribution management is to determine specifically how to get electrical power from the CM power input to the loads in question. When this function is complete, all information necessary to format suitable H/W commands will have been generated. However, the actual formatting and routing of commands will not be done by this function; it will be done under Function 13.3.4.1.2, Network Internal Commands.

13.3.1.1 Network State Assessment

13.3.1.1.1 Switching State Table Update

Each line of the switching state table will represent the actual state (connected/disconnected) of a particular switching device in CM/PMAD. The table will be updated once every control cycle so that both commanded state transitions and, if they occur, uncommanded state transitions will be registered.

13.3.1.1.2 Redundancy Assessment

This function will keep and update as necessary a record of the present state of availability (available, failed, predicted soon to fail, etc.) of all CM/PMAD elements. The relative efficiency of redundant elements of CM/PMAD will also be included. Hereafter, this record will be referred to as the redundancy assessment record.

13.3.1.2. Power Path Selection

13.3.1.2.1 Command Sequence Generation

Using the data in the baseline load enable schedule (Function 13.3.2.2, On-board Scheduling), plus the output of Function 13.3.1.1, Network State Assessment, this function will compose an appropriate time sequence of switching state transitions. The sequence will be designed to minimize constructive interference between switching transients. It will also use the most efficient of the redundant elements available.

13.3.1.2.2 Command State Table Update

The command state table will contain in compact form all information necessary to format suitable commands to execute the switching time sequence composed by the previous function. The table will be updated at those times that one or more switching states are changed. Formatting of the commands will be done in Function 13.3.4.1.2, Network Internal Commands.

13.3.2 Load Management

13.3.2.1 Load Monitoring

13.3.2.1.1 Power Monitoring

Using sensor measurement or computation, the power monitoring function will find the total electrical power flow (in kVA) to the CM and will update it every control cycle. This function will support trend analysis, On-board Scheduling (Function 13.3.2.2), and Load Shedding (Function 13.3.2.3), as appropriate.

13.3.2.1.2 Energy Calculation

The total amount of electrical energy consumed by CM since some appropriate starting point will be computed in each control cycle by this function. This function will support trend analysis, On-board Scheduling (Function 13.3.2.2), and Load Shedding (Function 13.3.2.3), as appropriate.

13.3.2.2. On-board Scheduling

It is likely that a ground-generated timeline of some complexity will be transmitted to the space station. It is assumed that the Timeline will contain information on all space station activities, including projected activities in the CM. It is further assumed that the timeline will be uplinked no more often than about once per week, though a limited number of short uplink modifications may be transmitted at irregular intervals. This timeline will be one of the data inputs to the onboard scheduling function.

In addition, it is assumed that there will be a S/W entity that manages that portion of the space station not contained in the modules. In this report, the term space station manager will be used to refer to that entity. The onboard scheduling function will bargain with the space stationmanager for station-wide resources. Station-wide resources are those which must be shared between modules; they include electrical energy, thermal radiation capacity, crew member shift time, etc.

The main purpose of the onboard scheduling function is to produce and modify as necessary as baseline load enable schedule (BLES) for CM/PMAD. The BLES will cover a specific period in the near future and Function 13.3.1.2.1, Command Sequence Generation. Each entry in the BLES will contain the following information:

(1) A load designator (identifying number or other label);

(2) Whether that load is to be enabled (connected to electrical power) or disabled (disconnected form electrical power);

(3) The time when this event is to occur.

13.3.2.2.1 Major Scheduling

This function will be the most advanced tool in onboard scheduling. Input data to the major scheduling function will include the CM portion of the ground-generated timeline, preliminary station-wide resource allowances from the space station manager, the redundancy assessment record of Function 13.3.1.1.2, and various sensor measurements.

The major scheduling function will produce a preliminary load enable schedule (PLES) that is a precursor to the desired baseline load enable schedule.

In producing the PLES, the major scheduling function will try to use efficiently 100% of the preliminary electrical resource allowances from the space station manager. The difficulty of this task depends on whether any significant equipment faults, either within the CM or outside of it, have occurred since the original timeline was generated on the ground. If new faults have not occurred, the task of the major scheduling function is relatively simple, and it will produce a PLES that will very much resemble the CM portion of the timeline. But if significant new faults have occurred, they would affect the balance of electrical resources all over the station. In this latter case, the major scheduling function would either have to make do with lower-than-expected allowances, or would have to find ways to use as much as possible of an unexpected surplus of electrical resources. If significant new faults have occurred, then the task of the major scheduling function has become more complex. Under such circumstances, it is estimated that this function could take up to, roughly, a half hour to compose the PLES.

In addition to composing the PLES, this function will also assign to each CM/PMAD load a load class number. A load class number is a rough, station-wide priority designation. Many CM loads may have the same load class number. Because there will be a limited number of load classes station-wide (it is recommended that there be 10 or fewer), the bargaining of station-wide resources between the onboard scheduling function and the space station manager can be conducted over a simple, generic interface, thus preserving the modularity of the CM design.

13.3.2.2.2 Load Requirements Projections

Using the data in preliminary load enable schedule generated in the previous function and other data available in the CM, and using portions of the software that supports Function 13.3.3.1.3, Network Solution, this function will compose projections (predictions) of various CM/PMAD load requirements versus time.

There will be two types of projections: total power projections and energy consumption projections. A projection of each type will be composed for each load class (load class numbers are generated in the previous function). Thus, if there are "n" load classes, this function will provide "2n" load requirements projections.

It is assumed that there will be one or more intermediate levels of software-controlled management tasks between the CM/PMAD task and the space station manager task. This function will forward its projections through those intermediate tasks to the space station manager task as electrical resource requests necessary to support the preliminary load enable schedule.

13.3.2.2.2.1 Total Power Projections

This function will compose a total power request versus time projection for each load class in CM/PMAD. Total power is defined as the magnitude (in kVA) of complex power.

Each projection will cover the period of the preliminary load enable schedule. The time resolution of each projection will be limited; if the projections were plotted as graphs, they would look like stair-step functions with each stair-step representing a discrete time interval. The value of total power assigned to any given stair-step will be the requested maximum value expected during that interval.

13.3.2.2.2.2 Energy Consumption Projections

This function will compose an energy consumption request versus time projection for each load class in CM/PMAD.

Each projection will cover the period of the preliminary load enable schedule. The time resolution of each projection will be limited; if the projections were plotted as graphs, they would look like stair-step functions with each stair-step representing a discrete time interval. The value of energy consumption assigned to any given stair-step will be the estimated energy to be consumed during that interval only.

13.3.2.2.3 Minor Scheduling

The space station manager will examine the load requirements projections of Function 13.3.2.2.2 and will produce final station-wide resource allowances for each of the modules. Those final allowances that are intended for CM will be forwarded to this function. The minor scheduling function will consider these allowances, any final modifications requested by the crew or ground personnel, and other data available in the CM, and will modify the PLES generated in Function 13.3.2.2.1 to produce a BLES. The BLES is a compromise schedule that the timeline, the space station manager, the onboard scheduling function, the crew, and the ground all agree with at the end of formal scheduling activities. If there wereinaccuracies in the various scheduling processes on the ground and aboard the station, if no new equipment faults appeared (either in the CM or outside it) which effected the CM/PMAD task, and if neither the crew or the ground requested further loading changes, the BLES would be the final schedule for the CM/PMAD activities.

In producing the BLES, the minor scheduling function will try to use efficiently 100% of the final electrical resource allowances from the space station manager. Because the minor scheduling function has the PLES to begin with, and because the final allowances from the space station manager will not be greatly different from the preliminary allowances, the task of minor scheduling is much less difficult than that of major scheduling (Function 13.3.2.2.1). The minor scheduling function should be able to perform its task within a few minutes.

To this point, all of the description of the onboard scheduling function and its attendant subfunctions has been concerned with formal scheduling activities. It is estimated that formal scheduling as described above (excluding final crew or ground modifications) will take up to 30 or 40 minutes to accomplish. Because formal scheduling will include bargaining with the space station manager for station-wide resources, and because station-wide resources must be shared between modules, it seems reasonable that formal scheduling will occur on all modules simultaneously. Also, because crew shift time is a station-wide resource, all of the crew should be given the opportunity to approve or modify the formal schedules for the various modules. This final approval, even by an experienced crew, could take as much as 15 or 20 minutes. When this time is added to the time already taken by S/W, the total for formal scheduling activity comes to about an hour. Because of the time taken by S/W, because all of the modules must be involved simultaneously, and because of the time taken away from the crew, it is obvious that formal onboard scheduling will not be invoked frequently. In fact, it will probably not be done any more than once in each crew shift.

There will be times when the crew or ground will wish to modify the BLES of a particular CM without having to go through a formal scheduling process.

Suppose that a crew member became suddenly ill three hours into the schedule and could not monitor several loads assigned to him. Because other crew members' shift time had already been scheduled, they could not monitor his loads either. If the crew decided not to invoke a formal rescheduling, they would simply tell the CM/PMAD task which loads to remove. Minor scheduling (this function) would remove the ill crew member's loads from the BLES and would replace them with other loads that would not require human assistance. This load-filling process would assure that as few as possible of the CM's final resource allowances were wasted. Any informal change to the BLES that involved the net removal of loading would be compensated by this load-filling process.

What about informal changes to the BLES that involve adding a load or loads? The crew or ground should be discouraged from making informal changes that add loads; electrical energy that can be gathered and stored by the space station is limited, and close to 100% of it will have been accounted for by the original ground-generated timeline and by the major and minor scheduling functions, all of which contributed to the BLES. For example, suppose that an astronomy team in Arizona has just concluded that the solar flare they are observing will become the biggest one recorded since 1947. Suppose further that the space station is now on the night side of Earth but will be able to observe the sun in 20 minutes, not enough time for a formal scheduling. In a case like this, the crew or ground would tell the CM/PMAD task which load is to be added (the solar instrument platform), when it is to be added, and how long it is to be supplied with electric power. Minor scheduling (this function) would then quickly determine which other loads would have to be descheduled from the BLES to accommodate the new load and would so notify the crew/ground. The function would recommend the descheduling of as few loads as possible from the lowest load classes. If the crew or ground approved these recommendations, this function would modify the BLES accordingly.

13.3.2.3 Load Shedding

Ordinarily, the BLES produced in onboard scheduling (Function 13.3.2.2) would require no further modification. However, further modification by rapid load shedding may be required if one or more of the following things happen:

(1) If there has been a significant error in one of the scheduling functions, on the ground or in orbit, which underestimates the electrical resources required by scheduled loads;

(2) If a scheduled load suddenly begins drawing significantly more power than usual;

(3) If CM/PMAD is notified by the space station manager that there must be a sudden reduction in the electrical resource allowances scheduled for the present period;

(4) If CM/PMAD is notified by the CM thermal subsystem that more heat is being generated in CM than can be safely dumped.

Should load shedding become necessary, this function will use the load priority list (Function 13.3.2.4) as a guide for the rapid disabling of low-priority loads. After it has shed a load, this function will not re-enable it. It will communicate with minor scheduling (Function 13.3.2.2.3). This latter function may reschedule the shed load to be enabled at some later time or may replace it with a less demanding load.

13.3.2.4 Load Priority List Maintenance

As a background task, this function will continuously maintain a load priority list. If the CM/PMAD subsystem is being started up for the first time or after a general power failure, this function will have the capability of generating a completely new load priority list in about 15 minutes and continuously maintaining it. In the list, each CM/PMAD load scheduled in the baseline load enable schedule (Function 13.3.2.2.3) as enabled or soon-to-be-enabled will have a unique CM-wide load priority number.

If load shedding (Function 13.3.2.3) decides that one or more loads must be quickly disabled, it will use the load priority list as a guide to determine which loads should be shed first and which should be kept powered to the end. The load priority list may also be used as a guide for some strategies of fault isolation (Function 13.3.3.2.2).

Ordinarily, the BLES will be comprehensive enough that the load priority list will seldom need to be invoked. The primary use of the list will be to help maintain effective load control in the event of the sudden onset of a significant equipment fault, either within CM or outside it, that reduces CM load capacity.

13.3.2.4.1 Load Schedule Assessment

This function will assess the importance to load priority of elements of the BLES of Function 13.3.2.2.3.

13.3.2.4.2 Loads Availability Assessment

This function will inspect the redundancy assessment record of Function 13.3.1.1.2, and will determine whether a given load can be connected to CM electrical power. If a load is not available, this function will direct that it be stricken from the load priority list, or if the load is not already on the list, that it not be placed there.

13.3.2.4.3 Operational Requirements Interpretation

There will probably be a body of operational requirements, mission rules, and other similar documentation that could affect the assignment of load priority. This function will interpret that body of documentation (or a database equivalent).

13.3.2.4.4 Load Priority Assignments

As the functions described above are being performed, they will direct the Load Priority Assignments function. This latter function will perform the actual updating of the load priority list.

When the BLES (Function 13.3.2.2.3) directs that load "X" be disabled, this function will take its priority number away and remove load "X" from the list entirely. As long as a load remains scheduled as disabled, it will not be on the list and will have no priority number. This function will then fill in the resulting gap in the list by upgrading the priority numbers of those loads which had had a lower priority than load "X".

When the BLES shows that load "Y" is scheduled soon-to-be-enabled, this function will assign to load "Y" a provisional priority number and will place load "Y" on the list in the appropriate place. Loads having a priority number equal to or less than load "Y"'s number will have their priorities reduced by one so that no two loads will have the same number. Soon after when load "Y" is enabled, this function, in accordance with the directions of Functions 13.3.2.4.1 through 13.3.2.4.3, will move load "Y"'s priority number up within the list until load "Y" arrives at its enabled priority level. Thereafter, as long as load "Y" is scheduled as enabled, its priority will be reviewed about every 15 minutes and changed as appropriate.

13.3.3 Health Management

This function manages the "health" of elements within CM/PMAD only.

13.3.3.1 Maintenance Support

13.3.3.1.1 Status Prediction

In this function, accumulated engineering data from the CM/PMAD subsystem will be analyzed and predictions made regarding the future reliability of various CM/PMAD elements.

13.3.3.1.2 Preventive Maintenance Scheduling

Based on the results of the previous function, this function will develop preventive maintenance schedules for CM/PMAD. These schedules will eventually be incorporated into the mission timeline generated on the ground.

13.3.3.1.3 Network Solution

This function will compute periodically a general solution of the CM/PMAD power network state. Inputs to this network solution will include electrical measurements at the CM power input, information from the switching state table (Function 13.3.1.1.1), and a database of operational characteristics of various loads and CM/PMAD elements. The network solution will provide computed values to check against actual sensor measurements around the CM power network.

The network solution will be used by fault management (Function 13.3.3.2) to detect and log subtle changes in the operating characteristics of various CM/PMAD elements. Detecting these subtle changes will be useful in managing soft faults in the CM power network. Logging of the changes will provide some of the input data for status prediction (Function 13.3.3.1.1).

Fault management will also use the network solution to spot malfunctioning network sensors.

Portions of the networksolution software will be used to support Function 13.3.2.2.2, Load Requirements Projections.

13.3.3.1.4 Monitoring

(1) Collect CM/PMAD engineering data with various transducers;
(2) Convert the data (if necessary) to electrical signals;
(3) Condition the electrical signals in hardware;
(4) Convert the resulting analog signals to digital values;
(5) After each digital value is transported to one of the CM/PMAD microprocessors (Function 13.3.4.2.2, Network Internal Data), the monitoring function will condition the value in S/W, converting it into a mathematical or logical analogue of the original engineering datum.

13.3.3.1.5 History Records Generation

Various records of the operating history (or predicted operating future) of CM/PMAD will be kept by this function. The function will initiate the recording of most of these records into peripheral storage (disk or similar) at the CM computer. The function will also initiate the transmission of some of the records to the space station telemetry subsystem for ultimate transmission to the ground.

13.3.3.2 Fault Management

13.3.3.2.1 Fault Detection

This function will detect faults, either in H/W or in S/W, within the CM/PMAD Subsystem.

13.3.3.2.2 Fault Isolation

After a fault has been detected, this function will determine its location within the CM/PMAD subsystem.

13.3.3.2.3 Fault Compensation

When the fault has been precisely located, this function will restore the CM/PMAD subsystem to as normal an operating state as is possible.

13.3.3.2.4 Fault Logging

The fault logging function will do two or more of the following:

(1) Compose a fault report consisting of: (a) a description of the fault, (b) a time-sequence log of its detection, isolation, and compensation, and, if appropriate, (c) instructions for the manual replacement of the failed component;

(2) Cause to be written to peripheral storage (disk or similar) a permanent record of the fault report;

(3) If necessary, update the redundancy assessment record (Function 13.3.1.1.2) to show whether the affected CM/PMAD element is failed or predicted to fail;

(4) Initiate the display of the fault report on a video screen at the CM computer console;

(5) Initiate transmission of the fault report to the space station telemetry subsystem;

(6) Initiate an audible/visible alarm within the CM.

How many of these steps that are performed by the function will depend on the severity of the fault.

13.3.4 Command/Data Interfacing

This function is performed in every hardware system that is primarily computer controlled but also monitored by persons. The techniques required to perform this function are well understood and have been well developed by systems programmers;

therefore, they will not be described in detail here. However, broad classes of command handling and data handling functions are listed below.

13.3.4.1 Command Handling

13.3.4.1.1 Network External Commands

13.3.4.1.2 Network Internal Commands

13.3.4.2 Data Handling

13.3.4.2.1 Network External Data

13.3.4.2.2 Network Internal Data

APPENDIX V – CREW INTERFACES

14.0 APPENDIX V: CREW INTERFACES

14.1 Power Conditioning

14.1.1 Logic Power Conversion

This function includes voltage transformation and ac/dc conversion of power for the distributed logic circuitry in the CM.

Display to Crew:

Data specific to the module that performs this function would appear on the CM/PMAD block diagram display. This display would show data sufficient to allow the crew to decide whether it was necessary to connect an alternate module.

Control by Crew:

Not appropriate. Applications similar to this are routinely, rapidly, precisely, efficiently, and reliably controlled by hardware.

14.1.2 Logic Power Conditioning

This function includes dc voltage regulation and filtering of power for the distributed logic circuitry in the CM.

Display to Crew:

Data specific to the module that performs this function would appear on the CM/PMAD block diagram display. This display would show data sufficient to allow the crew to decide whether it was necessary to connect an alternate module.

Control by Crew:

Not appropriate. Applications similar to this are routinely, rapidly, precisely, efficiently, and reliably controlled by hardware.

14.2 POWER DISTRIBUTION

14.2.1 Circuit Protection

Display to Crew:

Data specific to any given RPC or RCCB that performs this function would appear on the CM/PMAD block diagram display. Included would be data showing whether the device tripped and when it tripped. For a device which had tripped, the crew would be able to select whether to view data which applied to the device just prior to the moment it tripped, or to view data describing the present state of the device. The display would also show data relevant to the state of the CM dead-face switch (described below).

Control by Crew:

Trip levels may be set remotely on RPCs or RCCBs. If so, they would be adjustable via the CM/PMAD block diagram display. For devices controlling three-phase circuits, trip levels should only be adjustable for all three phases simultaneously and equally.

It is specifically recommended that any control that the crew applies to a given three-phase device should apply simultaneously and equally to all phases of the device. If crew control were not limited in this way, loading imbalances could occur that could open RPCs or similar devices in the power network. Such loading imbalances could conceivably damage elements of the CM/PMAD subsystem.

A double-throw dead-face switch would be provided that would enable a crew member to cut all electrical power in the CM, except emergency lighting and other emergency subsystems. The purpose of the dead-face switch would be three-fold: (1) it would be the crew's last line of defense in case of an electrical fire, (2) it would be a rapid and sure means of disconnecting runaway equipment which threatened immediate crew injury or equipment damage, and (3) it would be a convenient means of safeguarding CM circuitry while power cables were being connected to or disconnected from the CM.

14.2.2 Load Switching

Display to Crew:

Data specific to any given RPC that performs this function would appear on the CM/PMAD block diagram display. Included would be data from the command state table (Function 14.3.1.2.2) showing the commanded switching state of the RPC, and data taken from the switching state table (Function 14.3.1.1.1) showing the actual switching state of the RPC.

Control by Crew:

RPCs that perform this function would be controllable via the CM/PMAD block diagram display. They could be commanded by the crew to connect or to disconnect. For RPCs controlling three-phase circuits, connect/disconnect commands should apply to all three phases simultaneously and equally (i.e., either all phases would be connected, or all phases would be disconnected).

14.2.3 Bus Switching

Display to Crew:

Data specific to any given remote controlled circuit breaker (RCCB) or remote bus isolator (RBI) that performs this function would appear on the CM/PMAD block diagram display. Included would be data from the command state table (Function 14.3.1.2.2) showing the commanded switching state of the device in question, and data taken from the switching state table (Function 14.3.1.1.1) showing the actual switching state of that device.

Control by Crew:

RCCBs or RBIs that perform this function would be controllable via the CM/PMAD block diagram display. They could be commanded by the crew to connect or to disconnect. For devices controlling three-phase circuits, connect/disconnect commands should apply to all three phases simultaneously and equally (i.e., either all phases would be connected, or all phases would be disconnected).

14.3 POWER NETWORK CONTROL

14.3.1 Distribution Management

14.3.1.1 Network State Assessment

14.3.1.1.1 Switching State Table Update

Display to Crew:

Data from the updated switching state table would be featured on the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. Sufficient redundancy should be built into the CM/PMAD S/W and its supporting H/W that manual control of this function would not be necessary. The crew must rely on software to display the switching states, because traditional hardware displays (e.g., pilot lamps or mechanical flags) will not be used on the CM.

14.3.1.1.2 Redundancy Assessment

This function will keep and update as necessary a record of the present state of availability of and relative efficiency of all CM/PMAD elements. Hereafter, this record will be referred to as the redundancy assessment record.

Display to and Control by Crew:

The present state of availability of CM/PMAD elements, as kept in the redundancy assessment record, would be shown on the CM/PMAD block diagram display. Elements that were recorded as failed would be flagged in red. Elements that were recorded as having been predicted soon to fail would be flagged in yellow. Ordinarily, the redundancy assessment record would be updated automatically by CM/PMAD S/W; this would in turn drive the placement of the colored flags on the CM/PMAD block diagram display. However, the crew would have the option to reverse this sequence by manually entering flags in to the CM/PMAD block diagram display. The status implied by such manually-entered flags would then be automatically incorporated into the redundancy assessment record.

A crew member could elect to place a red flag on an element in the CM/PMAD block diagram display. This would prompt the S/W to change the entry for that element in the redundancy assessment record, listing the element as failed. In effect, the element in question would have been permanently disabled so that the S/W would not connect it into the subsystem again.

Alternatively, if a crew member had reason to believe that a particular element of the CM/PMAD subsystem would fail in the near future, he or she could place a yellow flag on the element in the CM/PMAD block diagram display. This would prompt the S/W to change the entry for that element in the redundancy assessment record, listing the element as predicted to fail. Thereafter, whenever the CM/PMAD S/W established a power path that could involve the flagged element, it would select the unflagged alternate instead. The software would not use the yellow-flagged element unless the alternate element subsequently failed.

14.3.1.2 Power Path Selection

14.3.1.2.1 Command Sequence Generation

This function will compose appropriate time sequences of commands for the purpose of establishing or breaking electrical power paths in the CM/PMAD power network. Each sequence would be designed to minimize constructive interference between switching transients, and would use the most efficient redundant elements available. This function does not encompass the formatting of or actual execution of the commands.

Display to Crew:

If this function were being performed by CM/PMAD S/W, it would proceed too rapidly for the crew to follow. If the crew elected to perform this function, a display of the composing of a sequence would be unnecessary (see "Control by Crew", below). Of course, the crew would be able to see the results of any command sequence as it was being executed by viewing the CM/PMAD block diagram display.

Control by Crew:

A crew member electing to intervene in the CM/PMAD task should move cautiously. He or she should only give one manual switching command to the subsystem at a time and should wait to see the results. Therefore, a crew member would perform the command sequence generation function in his or her mind and only one step beyond the most recent manual command. Because of the relative slowness of human reflexes and the need for caution, the crew member would be initiating successive manual commands too slowly to cause significant constructive interference between switching transients.

14.3.1.2.2 Command State Table Update

Display to Crew:

Updated data from the command state table would be available to the crew via the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. The command state table will be a compact coded record suitable for use only by other CM software. While it is conceivable that the crew could control CM/PMAD elements by manually updating the command state table, such a task would be time consuming and not at all intuitive.

14.3.2 Load Management

14.3.2.1 Load Monitoring

14.3.2.1.1 Power Monitoring

Display to Crew:

Data specific to this function would appear on the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. Though this is a very simple function, it will have to be performed in every control cycle. This would, therefore, not be a practical task for a crew member.

14.3.2.1.2 Energy Calculation

Display to Crew:

Data specific to this function would appear on the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. Though this is a very simple function, it will have to be performed in every control cycle. This would, therefore, not be a practical task for a crew member.

14.3.2.2 On-board Scheduling

14.3.2.2.1 Major Scheduling

Display to Crew:

Data generated by this function would be available to the crew in the load enable schedule display.

Control by Crew:

Not appropriate. The manual performance of this function would require the review of large amounts of data and the expenditure of a great deal of time. If this function were to fail, the entire on-board scheduling function would become unreliable. In such an event, the crew would probably elect to control CM/PMAD manually through the CM/PMAD block diagram display. Such control would be on an ad hoc basis, and would not really be scheduling in the spirit of this function.

14.3.2.2.2 Load Requirements Projections

Display to Crew:

Data generated by this function would not be directly available to the crew in real time, though it would be implied in the load enable schedule display. Records of this data would be available through Function 14.3.3.1.5, History Records Generation.

Control by Crew:

Not appropriate. The manual performance of this function would require the review of large amounts of data and the expenditure of a great deal of time. If this function were to fail, the entire onboard scheduling function would become unreliable. In such an event, the crew would probably elect to control CM/PMAD manually through the CM/PMAD block diagram display.

14.3.2.2.3 Minor Scheduling

Display to Crew:

Data generated by this function would be available to the crew in the load enable schedule display.

Control by Crew:

Not appropriate. The manual performance of this function would require the expenditure of too much time to be practical. If this function were to fail, the entire onboard scheduling function would become unreliable. In such an event, the crew would probably elect to control CM/PMAD manually through the CM/PMAD block diagram display. Such control would be on an ad hoc basis, and would not really be scheduling in the spirit of this function.

14.3.2.3 Load Shedding

Displays to Crew:

The performance of this function would be directly evident to the crew in the load enable schedule display and would be implied in changes in the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. In order to be effective, this function would need to be performed far more rapidly than human reflexes would permit.

14.3.2.4 Load Priority List Maintenance

14.3.2.4.1 Load Schedule Assessment

Displays to Crew:

The crew would consult the load enable schedule display and the load priority list display (Function 14.3.2.4.4) to assess the effect of the schedule on the list.

Control by Crew:

Strictly speaking, a crew member would control this function in his or her mind. Of course, the results of this function would be applied by the crew to the load priority list display (Function 14.3.2.4.4).

14.3.2.4.2 Loads Availability Assessment

Displays to Crew:

The crew would consult the CM/PMAD block diagram display (looking for red- or yellow-flagged elements) to assess whether a given load were available and, therefore, whether a priority assignment were appropriate.

Control by Crew:

Strictly speaking, a crew member would control this function in his or her mind. Of course, the results of this function would be applied by the crew to the load priority list display (Function 14.3.2.4.4).

14.3.2.4.3 Operational Requirements Interpretation

Displays to Crew:

Operational requirements, mission rules, and other similar documentation are likely to be extensive. It is unlikely that complete hard copies of all of it would be stored aboard the space station. It is equally unlikely that it would be convenient for a crew member to skim through this large body of documentation via video display. The most useful display would seem to be an audio conversation with the ground personnel who are familiar with the documentation and can refer to it directly.

Control by Crew:

The crew should certainly be allowed to participate with the ground in controlling this function if human intervention becomes necessary. Strictly speaking, a crew member would control this function in his or her mind. Of course, the results of this function could be applied by the crew to the load priority list display (Function 14.3.2.4.4).

14.3.2.4.4 Load Priority Assignments

Using data generated by Functions 14.3.2.4.1 through 14.3.2.4.3, this function performs the actual updating and maintenance of the load priority list.

Display to Crew:

The crew would be able to view the load priority list at any time by consulting the load priority list display. The display is described below.

Control by Crew:

The load priority list would ordinarily be maintained automatically by CM/PMAD S/W. It is recommended that the crew not be allowed to modify the list while the automatic function is running. However, the crew would be allowed to disable the automatic function. The crew would then be able to modify the list via the load priority list display. This tabular display would show (in English) the form of the compactly coded list. Each entry in the display would cite a specific Load and would show its priority number (1 would be the highest priority, 2 the second highest, etc.) The crew would be able to modify the list by modifying the display using simple keyboard or touch-screen entries as appropriate. After modifying the list, the crew could elect to restart the automatic function if appropriate.

14.3.3 Health Management

14.3.3.1 Maintenance Support

14.3.3.1.1 Status Prediction

In this function, accumulated engineering data from the CM/PMAD subsystem will be analyzed, and predictions will be made regarding the future reliability of various CM/PMAD elements. Because this is a rather complex function, it will probably be performed by experts and expert systems on the ground. The results of this function will be uplinked to the CM/PMAD task in a timely manner. These results will also drive preventive maintenance scheduling (Function 14.3.3.1.2).

Display to Crew:

Elements of CM/PMAD that have been predicted soon-to-fail would be flagged in yellow on the CM/PMAD block diagram display. Also included would be time estimates to failures. This information would be uplinked from the ground to the redundancy assessment record of Function 14.3.1.1.2. CM/PMAD S/W would use data in the redundancy assessment record to determine which (if any) elements to flag in yellow on the CM/PMAD block diagram display.

Control by Crew:

Not appropriate. This is a rather complex function which will probably be performed by experts and expert systems on the ground. It is doubtful that the crew could spare the time necessary to perform this function.

14.3.3.1.2 Preventive Maintenance Scheduling

Based on the results of status prediction (Function 14.3.3.1.1), a preventive maintenance schedule for CM/PMAD will be developed and eventually incorporated in the ground-generated timeline. Also contained within the timeline may be codes referring to canned maintenance procedures which may be kept in high-density storage devices (laser disk or similar) aboard the CM. This function will probably be performed on the ground.

Display to Crew:

Preventive maintenance schedules will be a part of the timeline. As such, they will be considered by onboard scheduling (Function 14.3.2.2) in producing the BLES. The BLES will contain instructions to disable appropriate CM/PMAD devices and loads in support of preventive maintenance or repair operations. Thus, preventive maintenance scheduling will be visible to the crew via the load enable schedule display.

It is assumed that CM S/W not connected with the CM/PMAD task will note the codes referring to the canned maintenance procedures mentioned above. Subsequent production of hard copies of those procedures for use by the crew is not now assumed to be part of the CM/PMAD task.

Control by Crew:

Not appropriate. To do this function properly, it would first be necessary to perform the previous function (Function 14.3.3.1.1., Status Prediction), and the crew will not be doing that.

14.3.3.1.3 Network Solution

Displays to and Control by Crew:

It is conceivable that expected data from the automatically generated network solution could be available for display at crew option on the CM/PMAD block diagram display. Of course, actual data measured by sensors or computed from sensor measurements would already be available on that display. However, meaningful comparison of the expected and actual data would be a difficult task even for a person thoroughly trained in the CM/PMAD subsystem; useful control of the function would be even more difficult. Because most crews are expected to contain no such experts, it does not seem useful to design a crew interface to support this function. It should be noted that a tool on the ground similar to that conjectured here might be useful to an expert examining the automatic performance of this function.

14.3.3.1.4 Monitoring

Display to Crew:

Sensor data collected and processed by the monitoring function would be available on the CM/PMAD block diagram display.

Control by Crew:

The monitoring function, as defined, will perform the following services for CM/PMAD:

(1) Collect CM/PMAD engineering data with various transducers;
(2) Convert the data (if necessary) to electrical signals;
(3) Condition the electrical signals in H/W;
(4) Convert the resulting analog signals to digital values;
(5) After each digital value is transported to one of the CM/PMAD microprocessors (Function 14.3.4.2.2, Network Internal Data), the Monitoring function will condition the value in S/W, converting it into a mathematical or logical analogue of the original engineering datum.

It is plain from the above definition that direct control of the Monitoring function by the crew would not be appropriate.

14.3.3.1.5 History Records Generation

Displays to Crew:

Various records of the operating history of CM/PMAD (various sensor measurements or computed quantities versus time) will be kept by this function. The function will initiate the recording of most of these records into peripheral storage (disk or similar) at the CM Computer. All such on-board records would be available to the crew in user-friendly do-it-yourself video displays, the forms of which the crew could compose as needed. The forms most likely to be of use by the crew would be columnar tables of related variables sampled at the same time intervals, and measurement(s) versus time graphs. These displays would be available through the CM computer console.

Control by Crew:

Not appropriate. Doing this function manually would take a great deal of time. It is doubtful that the crew could spare the time necessary to perform this function.

14.3.3.2 Fault Management

14.3.3.2.1 Fault Detection

Displays to and Control by the Crew:

Ordinarily, fault detection (this function) and fault isolation (Function 14.3.3.2.2) would be performed automatically by CM/PMAD S/W. If this part of the S/W should fail and if a significant fault should also appear in H/W, the crew should notice that something is wrong. For example: a load is unpowered when the load enable schedule display clearly shows it should be enabled. Another example: a powered load is acting strangely (panel light dims, device continually cycles on and off, module lighting flickers, etc.), suggesting that the quality of power delivered to its input has eroded. In situations such as these, where it appears that the fault management S/W is not doing its job, the crew should have some means of detecting and isolating the H/W fault. To manually search for H/W faults in the CM/PMAD power network, the crew would examine the subsystem data available in the CM/PMAD block diagram display. In principle, any H/W fault in the power network could be detected and isolated by this means, though the more subtle ones might escape notice of the crew. To manually detect and isolate a fault in the CM/PMAD data network, the crew would, via interactive displays (keyboard plus video), initiate self-test programs. The crew would also use interactive displays to run S/W comparisons between programs in CM/PMAD microprocessors and their recorded equivalents in peripheral storage (disk, tape, or similar) of the CM computer.

14.3.3.2.2 Fault Isolation

Displays to and Control by the Crew:

See comments under Fault Detection (Function 14.3.3.2.1).

14.3.3.2.3 Fault Compensation

Displays to Crew:

Any fault compensation that had been applied automatically by CM/PMAD S/W would be a part of the fault report (see Fault Logging, Function 14.3.3.2.4). The results of fault compensation, whether applied by the S/W, through crew intervention, or by ground command, would be shown on the CM/PMAD block diagram display.

Control by Crew:

No special control tool dedicated solely to this function is necessary. The crew would be able to apply whatever fault compensations were deemed appropriate by manipulating one or more of the following: the load enable schedule display, the CM/PMAD block diagram display, or any of the control tools described elsewhere in this appendix.

14.3.3.2.4 Fault Logging

Displays to Crew:

As part of its usual operation, this function will automatically generate fault reports. Each fault report will consist of (1) a description of the fault, (2) a time-sequence log of its detection isolation, and compensation, and, if appropriate, (3) instructions for the manual replacement of the failed element. Fault reports will be written to peripheral storage (disk or similar) at the CM computer. The crew would be able to read these records by calling them up on the video screen of the CM computer console. If a given fault were mild, this would be the only way the crew would see the associated fault report. If the fault were a little more severe, the Fault Logging function would automatically cause the fault report to be written to a video screen, as well as to peripheral storage. If the fault were still more severe, the fault logging function would also cause an alarm to be sounded in the CM. Regardless of the severity of the fault, its detection, isolation, and compensation would be implied in the data available in the CM/PMAD block diagram display. In that display, faulty components would be flagged in red, components that were predicted soon-to-fail would be flagged in yellow, and disconnected circuits in general would be evident. Furthermore, as display density permitted, notes would be displayed near flagged elements briefly describing the fault or predicted fault.

Control by Crew:

The crew would be able to generate manual fault reports, if desired, via keyboard entries at the CM computer console. The crew should be inhibited from modifying already existing fault reports, particularly those which were automatically generated by the fault logging function.

14.3.4 Command/Data Interfacing

This function will need to handle large amounts of digital data rapidly and precisely. Consequently, this would not be an appropriate function for human intervention.

APPENDIX VI – SYMBOLICS ENVIRONMENT

15.0 APPENDIX VI: SYMBOLICS ENVIRONMENT

15.1 The Symbolics Interface Processing Architecture

The Symbolics Interface (SI) to the Space Station Module Power Management and Distribution (SSM/PMAD) system provides software for the User Interface (UI), Front End Load Enable Scheduler (FELES), Load Priority List Management System (LPLMS), Scheduling (MAESTRO), and TCP/IP communications.

The architecture provides for independent processes operating in an asynchronous real time environment shown in Figure 15.1-1:

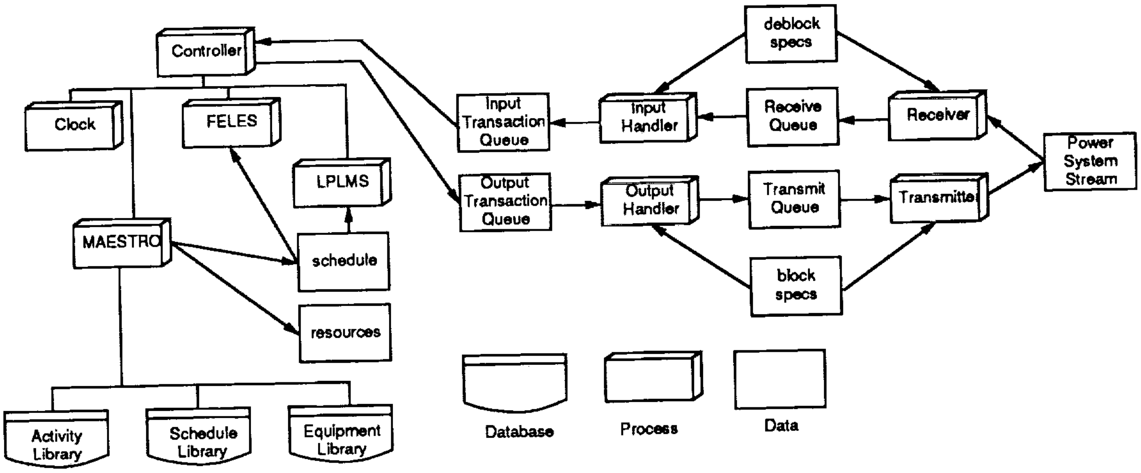


Figure 15.1-1 Symbolics Interface Processing Architecture

The Power System Stream is the stream connection to the network. All communications that flow to/from the Symbolics Interface and the SSMPMAD breadboard are processed through this data object. This stream insulates the transaction processing software from the specific communications medium.

The Receiver listens for input on the Power System Stream then reads a transaction. If the historical log is active, the transaction is recorded there, as well as being deblocked and placed on the queue of input transactions, the Receive Queue.

The Receive Queue is a first in first out queue of transactions obtained by the Receiver. It serves as an interface between the Receiver and the transaction Input Handler.

The Deblock Spec library is a set of deblock specifications for each of the blocked transactions that the system may process.

The Input Handler is responsible for obtaining messages from the Receive Queue, deblocking if necessary, formatting into an application transaction, then dispatching the transaction to the Input Transaction Queue or any other processing entity.

The Input Transaction Queue is a 3 level priority queue (high, normal, low) that obtains application specific transactions from the Input Handler. It provides the input to the real time Controller.

The Controller is responsible for the real time operations of the SSM/PMAD system. It is the main driver that provides for the initialization, time synchronization, event list distribution, priority list distribution, and contingency operations of the SSM/PMAD breadboard.

The mission Clock is the time driver for the operations of the the Symbolics Interface. Time granularity in the system is 1 minute. All time derived processes utilize this for time synchronization.

MAESTRO is the scheduling component of the system which provides the capabilities for scheduling spacecraft activities. The schedule from which the breadboard is driven is developed by MAESTRO. MAESTRO also provides the rescheduling capabilities necessary in the event of a real time fault in the power system breadboard.

The Activity Library is a library of spacecraft activity models that may be chosen for scheduling.

The Equipment Library is a library of powered equipment models, specifying where each piece of equipment may be connected in the power system and the various modes of operation of those pieces of equipment.

The Schedule Library provides a repository for previously generated schedules.

The Front End Load Enable Scheduler is the interface between MAESTRO based schedules and commands that effect the operation of the breadboard components. It is responsible for periodically sending component event lists to the breadboard for switch operations.

The Load Priority List Management System is responsible for periodically notifying the breadboard software of the relative priorities of each of the loads.

The Output Transaction Queue is a first in first out queue that provides an interface between commands generated by the Controller and the Output Handler that is to process those transactions.

The Output Handler is responsible for obtaining forms from the Output Transaction Queue, performing any required reformatting, performing required blocking, and adding an entry to the Transmit Queue for transmission of the transaction.

The Transmit Queue is a first in first out queue of transactions to be transmitted to other processing components in the power system network.

The Transmitter is responsible for the transmission of transactions along the network. It operates as its own process, sleeping until something is on the Transmit Queue. The transaction is removed from the queue, formatted for transmission on the Power System Stream, then if the historical log is active, the transaction is added to it.

15.2 Transaction

15.2.1 Transaction Format

The following table describes the basic transaction format:

Message Start	x	Start of message indicator Control-A (ascii 1)
Destination	x	Address of unit where message is being sent
Source	x	Address of unit sending the message
Message Type	p	Type of message
Message Block	?	Contains message data bytes The format of this varies with each message type
Message End	x	End of message indicator CR (ascii 13)

15.2.1.1 Notes on Formatting

All messages will consist of a sequence of ASCII data bytes. Except for Message Start and Message End all bytes will be printable ASCII characters.

PACKED numerical values are represented as a single ASCII character offset from 48 (zero) through 126 (tilde) which provides a range 0-78.

PACKED79 numerical values are represented as two ASCII packed numerical values describing a base 79 number. The first byte is in increments of 79, the second are units. This provides a numeric range of 0-6240.

NUMERIC values that are not PACKED or PACKED79 will be passed as a group of ASCII bytes (i.e. 300 would be 3 0 0).

Unit addresses for source and destination:

V	CAC (VME/10)
X	FRAMES (Xerox)
S	SI (Symbolics)
P	both FRAMES & CAC
llp	a particular LLP (A-H)

15.2.2 Transactions

15.2.2.1 01 Sync Time

SYNC TIME provides the timing parameters for time synchronization of the breadboard software components. SI will distribute this to the CAC and FRAMES. Now represents the current time. Start of Mission provides an actual calendar/clock time from which to base all scheduling offsets. It corresponds to mission time 00:00:00 (dd:hh:mm). All time based schedule data will be represented as minutes offset from Start of Mission.

15.2.2.2 02 Event List

SI will distribute the events for the load enable schedule to the CAC and FRAMES for operation of the breadboard.

15.2.2.3 03 Load Priority List

Whenever a new load priority list is created, SI will distribute the list to FRAMES and the CAC.

15.2.2.4 06 Component Switch to Redundant

Whenever FRAMES is notified that a switch has been switched to its redundant supply, FRAMES notifies the SI of the information using this transaction.

15.2.2.5 07 Load Shed

FRAMES will notify the SI anytime loads are shed, so that appropriate scheduling decisions may be made.

15.2.2.6 08 Contingency Events

When SI has completed its reaction to a contingency situation CAC and FRAMES will be notified of the new enable schedule, as well as what state the power system components should be in to successfully execute the new set of events. All state entries will be listed before all event entries. Both state and event entries have the same format, the difference being that all "time of state"s are filled with zeros.

15.2.2.7 09 Out of Service

Whenever a component is known to be out of service FRAMES will notify SI so that appropriate scheduling decisions may be made.

15.2.2.8 10 Utilization

In order to report/graph actual power utilization of components vs. available and/or scheduled power, FRAMES must notify SI of those measurements.

15.2.2.9 11 Ready?

READY? is sent by SI to tell FRAMES and the CAC to initialize and be prepared for the initial EVENTS and PRIORITIES. When the initialization has occurred, CAC and FRAMES will notify SI with the declarative I'm READY! message.

15.2.2.10 12 Ready!

The declarative message READY! is sent by FRAMES and the CAC to the SI as a notification that they are ready to receive the initial EVENTS and PRIORITIES.

15.2.2.11 13 Initialized

After receiving the initial EVENTS and PRIORITIES, FRAMES and the CAC send this message to the SI, this notifies SI that the other breadboard computer system components are ready for operation.

15.2.2.12 14 Source Power Change

The SOURCE POWER CHANGE is a simulated Space Station message, i.e. someone/thing of authority has notified the module that there will be change in the availability of power to the module.

15.2.2.13 15 Contingency Start

An anomalous condition has been recognized in the power system. FRAMES is working the situation and tells SI so via this transaction. Any transactions received by SI between the CONTINGENCY-START and CONTINGENCY-END messages are considered pertinent information to the contingency situation.

15.2.2.14 16 Contingency End

The contingency situation has been handled by FRAMES and all pertinent information has been sent to SI. SI should now handle implications to the schedule.

15.3 Controller State Transitions

QUIET	send <READY? 11> --> READY-WAIT

READY-WAIT	if <READY! 12> from CAC
	--> FRAMES-READY-WAIT
	else
	--> CAC-READY-WAIT

FRAMES-READY-WAIT	if <READY! 12> from FRAMES --> READY
CAC-READY-WAIT	if <READY! 12> from CAC --> READY
READY	start FELES start LPLMS send <EVENT-LIST 02> send <PRIORITY-LIST 03> --> INITIALIZE-WAIT

INITIALIZE-WAIT	if <INITIALIZED 13> from CAC
	--> FRAMES-INITIAL-WAIT
	else
	--> CAC-INITIAL-WAIT

FRAMES-INITIAL-WAIT	if <INITIALIZED 13> from FRAMES --> INITIALIZED
CAC-INITIAL-WAIT	if <INITIALIZED 13> from CAC --> INITIALIZED
INITIALIZED	send <START-OF-MISSION 01> --> NORMAL
NORMAL	when <CONTINGENCY-START 15> halt FELES halt LPLMS --> CONTINGENCY
CONTINGENCY	when <CONTINGENCY-END 16> handle contingency MAESTRO send <CONTINGENCY-EVENTS 08> send <PRIORITY-LISt 03> --> NORMAL

15.4 User Interface

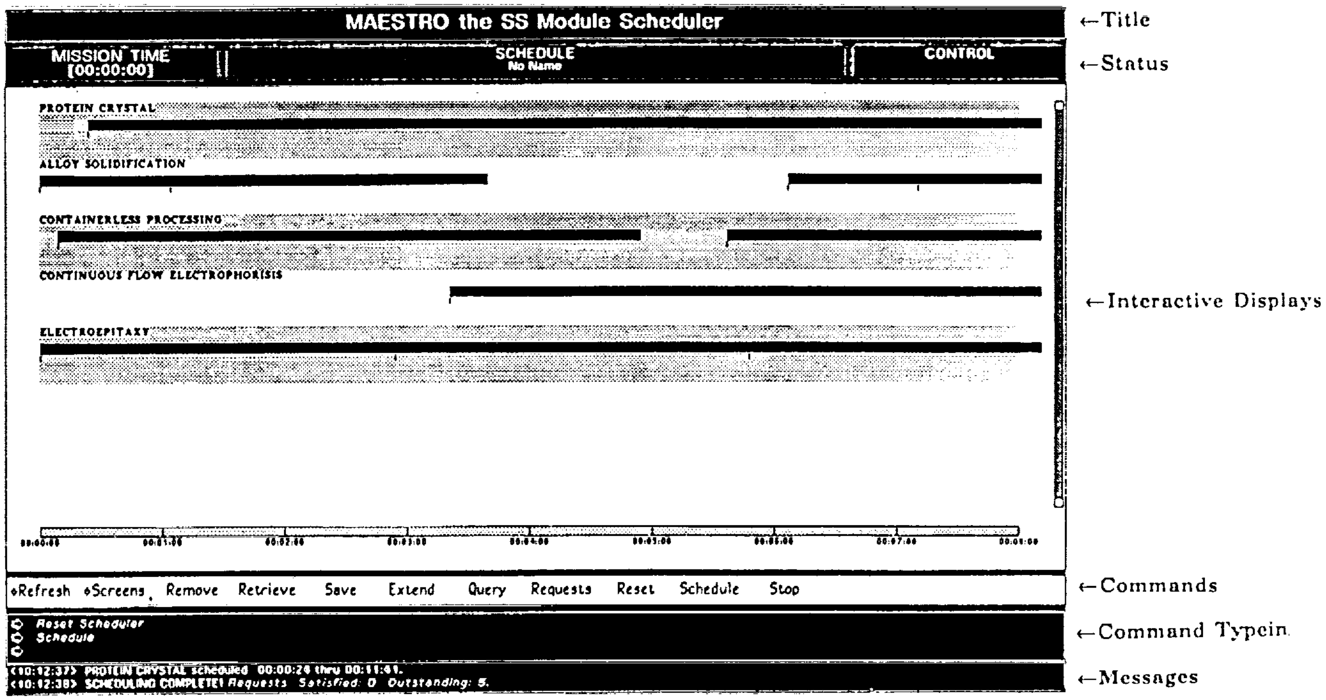
The User Interface (UI) for the Symbolics Interface of the SSM/PMAD software provides capabilities for developing, inspecting, and initiating operational scenarios for the power system breadboard. Through the use of multiple windows and configurations, the UI assists the user in the operation and monitoring of the mission definitions to stimulate the use of the breadboard.

The UI is command driven; each of the commands may be initiated through the use of the mouse, keyboard, or in some cases single key accelerators. The interface is designed to minimize keyboard input and promote use of the mouse.

The following 8 screen configurations are available:

Console
Front End Load Enable Scheduler
Load Priority List Management System
Scheduler
Resource Manager
Communications
Activity Editor
Equipment Editor

Each Screen is organized in a similar fashion. Figure 15.4-1 outlines how a typical screen is partitioned. Please note that it is nearly always the case that mouse sensitive objects appear in light windows while reverse video windows are information displays.



```
⇨ Reset Scheduler
⇨ Schedule
⇨
<10:12:37> PROTEIN CRYSTAL scheduled 00:00:24 thru 00:11:41.
<10:12:38> SCHEDULING COMPLETE! Requests Satisfied: 0 Outstanding: 5.
```

←Title
←Status
←Interactive Displays
←Commands
←Command Typein
←Messages

Figure 15.4-1 Example Symbolics Screen

15.4.1 Console

The Console provides overall system status information. It consolidates many of the status monitors that appear on other screens. The Console is the first screen displayed when the user initiates the system. See Figures 15.4.1-1 and 15.4.1-2 for the Console and its Help Screen.

The Mission Time window monitors the simulation mission time. The time is represented in a DD:HH:MM format. If the time is surrounded with square brackets [] the breadboard is stopped or halted. When starting the breadboard an alternate representation will appear, a countdown of how long until the start of the mission will be displayed and continuously updated.

The Control monitor displays the status of the system, that is, whatever status the real time Controller currently maintains.

The Schedule monitor displays the name of the schedule that is active.

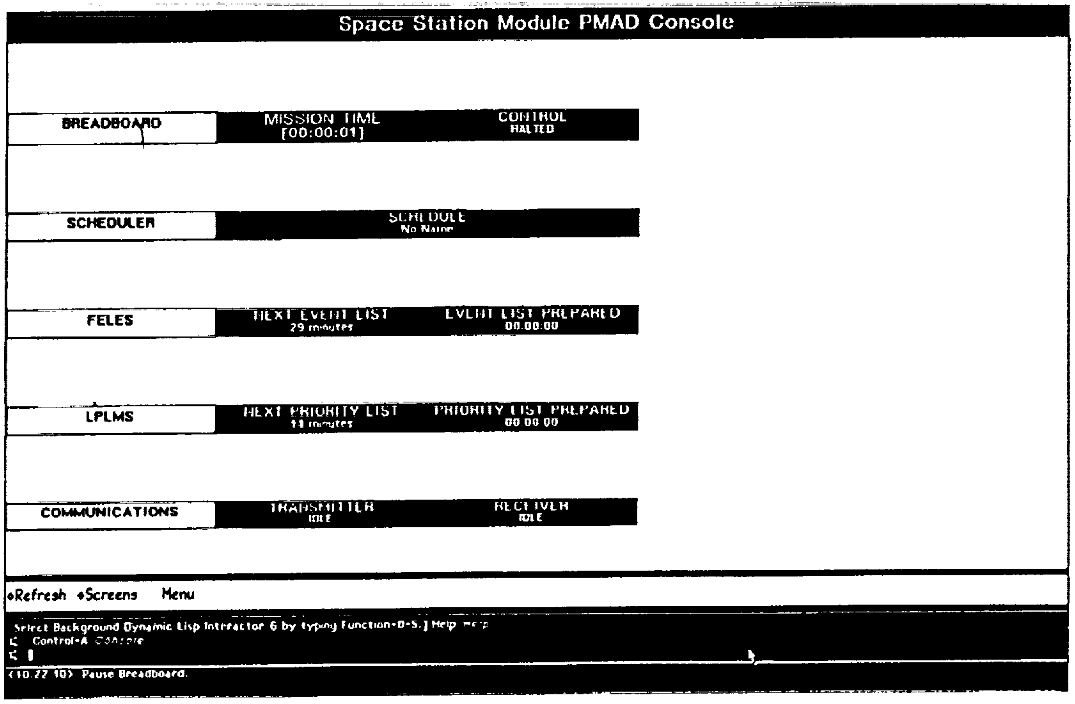
The Next Event List monitor displays how long until the next event list is created.

The Event List Prepared monitor displays when the last event list was created.

The Next Priority List monitor displays how long until the next priority list is created.

The Transmitter monitor displays the type of transaction transmitted.

The Receiver monitor displays the type of transaction received.



Space Station Module PMAD Console

BREADBOARD — MISSION TIME [00:00:01] — CONTROL HALTED

SCHEDULER — SCHEDULE No Name

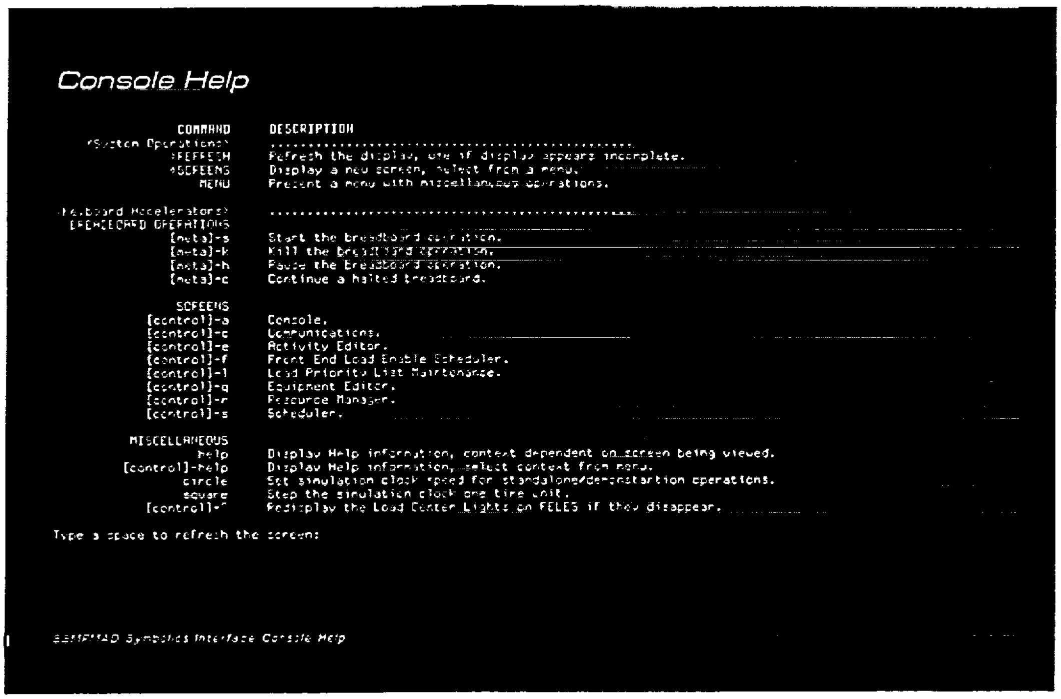
FELES — NEXT EVENT LIST 29 minutes — EVENT LIST PREPARED 00:00:00

LPLMS — NEXT PRIORITY LIST 11 minutes — PRIORITY LIST PREPARED 00:00:00

COMMUNICATIONS — TRANSMITTER IDLE — RECEIVER IDLE

```
Select Background Dynamic Lisp Interactor 6 by typing Function-0-S.] Help Help
⊊  Control-A Console
⊊ ▮
<10.22.10> Pause Breadboard.
```

Figure 15.4.1-1 Console Screen



Console Help

```
                COMMAND    DESCRIPTION
  <System Operations>      ..............................................
             *REFRESH      Refresh the display, use if display appears incomplete.
             *SCREENS      Display a new screen, select from a menu.
                 MENU      Present a menu with miscellaneous operations.

<Keyboard Accelerators>    ..............................................
 BREADBOARD OPERATIONS
             [meta]-s      Start the breadboard operation.
             [meta]-k      Kill the breadboard operation.
             [meta]-h      Pause the breadboard operation.
             [meta]-c      Continue a halted breadboard.

              SCREENS
          [control]-a      Console.
          [control]-c      Communications.
          [control]-e      Activity Editor.
          [control]-f      Front End Load Enable Scheduler.
          [control]-l      Load Priority List Maintenance.
          [control]-q      Equipment Editor.
          [control]-r      Resource Manager.
          [control]-s      Scheduler.

        MISCELLANEOUS
                 help      Display Help information, context dependent on screen being viewed.
       [control]-help      Display Help information, select context from menu.
               circle      Set simulation clock speed for standalone/demonstartion operations.
               square      Step the simulation clock one time unit.
          [control]-^      Redisplay the Load Center Lights on FELES if they disappear.

Type a space to refresh the screen:
```

SSM/PMAD Symbolics Interface Console Help

Figure 15.4.1-2 Console Help Screen

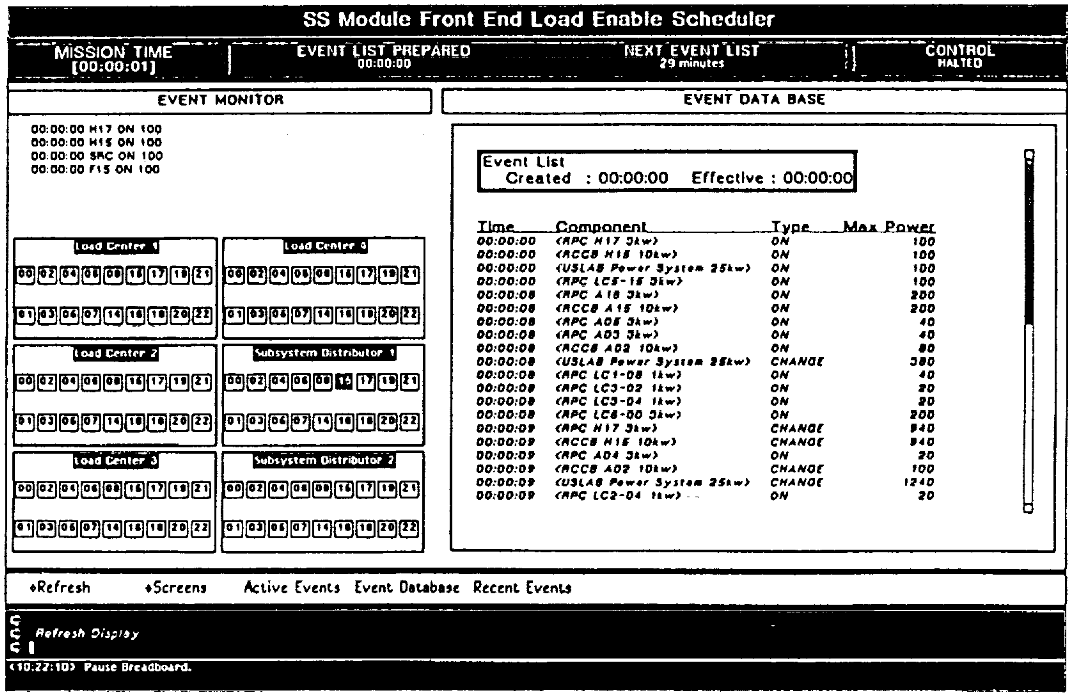
ORIGINAL PAGE IS
OF POOR QUALITY

15.4.2 Front End Load Enable Scheduler

The Front End Load Enable Scheduler enables the user to browse and monitor the generation of load enable commands. See Figures 15.4.2-1 and 15.4.2-2 for the FELES and Help Screen. Please refer to the Console description in the Symbolics Environment section for status line monitor definitions.

The Event Monitor is a textual and graphical display indicating what commands are being recognized by the power system. This monitor is continually updated as events are initiated as mission time advances. This monitor effectively shows what should be happening in the power system. The graphical display shows the RPCs within each load center. If the RPC is black it is enabled. A circle cross below an RPC indicates the RPC is out of service.

The Event Data Base is a textual listing of an event list. When the list was created and when it becomes effective are indicated. The user may choose any list that has been created by the FELES. By mousing on an event line the user will see what experiments are utilizing the specific component at the time indicated.



SS Module Front End Load Enable Scheduler

MISSION TIME [00:00:01]

EVENT LIST PREPARED 00:00:00

NEXT EVENT LIST 29 minutes

CONTROL HALTED

EVENT MONITOR

```
00:00:00 H17 ON 100
00:00:00 H15 ON 100
00:00:00 SRC ON 100
00:00:00 F15 ON 100
```

EVENT DATA BASE

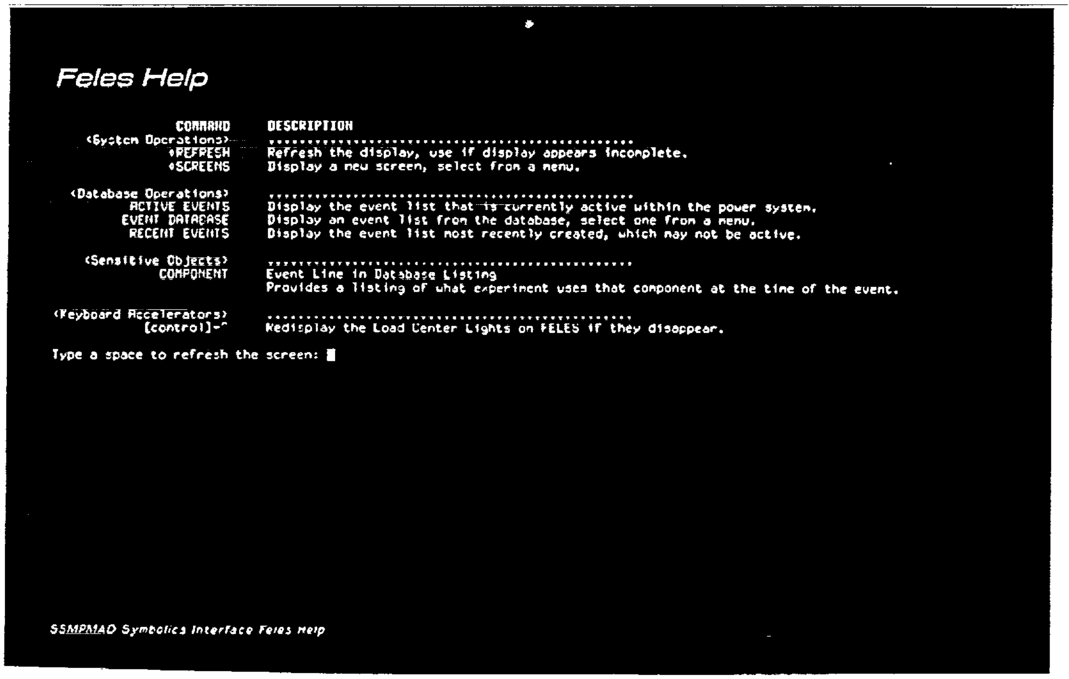
Event List
Created : 00:00:00 Effective : 00:00:00

Time	Component	Type	Max Power
00:00:00	<RPC H17 3kw>	ON	100
00:00:00	<RCCB H15 10kw>	ON	100
00:00:00	<USLAB Power System 25kw>	ON	100
00:00:00	<RPC LC5-15 3kw>	ON	100
00:00:08	<RPC A18 3kw>	ON	200
00:00:08	<RCCB A15 10kw>	ON	200
00:00:08	<RPC A05 3kw>	ON	40
00:00:08	<RPC A03 3kw>	ON	40
00:00:08	<RCCB A02 10kw>	ON	80
00:00:08	<USLAB Power System 25kw>	CHANGE	380
00:00:08	<RPC LC1-08 1kw>	ON	40
00:00:08	<RPC LC3-02 1kw>	ON	20
00:00:08	<RPC LC3-04 1kw>	ON	20
00:00:08	<RPC LC6-00 3kw>	ON	200
00:00:09	<RPC H17 3kw>	CHANGE	940
00:00:09	<RCCB H15 10kw>	CHANGE	940
00:00:09	<RPC A04 3kw>	ON	20
00:00:09	<RCCB A02 10kw>	CHANGE	100
00:00:09	<USLAB Power System 25kw>	CHANGE	1240
00:00:09	<RPC LC2-04 1kw>	ON	20

Refresh Display

<10:22:10> Pause Breadboard.

Figure 15.4.2-1 FELES Screen



Feles Help

```
              COMMAND    DESCRIPTION
      <System Operations>    ............................................
                 •REFRESH    Refresh the display, use if display appears incomplete.
                 •SCREENS    Display a new screen, select from a menu.

    <Database Operations>    ............................................
            ACTIVE EVENTS    Display the event list that is currently active within the power system.
           EVENT DATABASE    Display an event list from the database, select one from a menu.
            RECENT EVENTS    Display the event list most recently created, which may not be active.

      <Sensitive Objects>    ............................................
                COMPONENT    Event Line in Database Listing
                             Provides a listing of what experiment uses that component at the time of the event.

 <Keyboard Accelerators>     ............................................
             [control]-^     Redisplay the Load Center Lights on FELES if they disappear.

Type a space to refresh the screen: █
```

SSMPMAD Symbolics Interface Feles Help

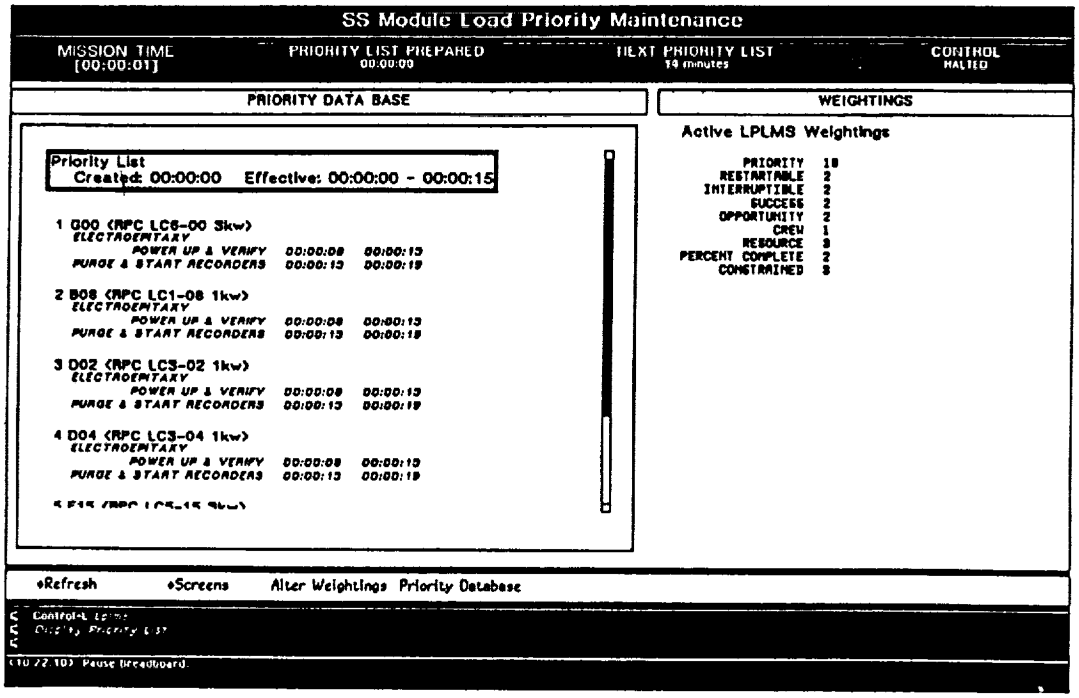
Figure 15.4.2-2 FELES Help Screen

15.4.3 Load Priority List Management System

The Load Priority Maintenance screen provides the user the ability to inspect load priority lists that have been generated by the LPLMS and transmitted to the power system. See Figures 15.4.3-1 and 15.4.3-2 for the LPLMS and its associated Help Screen. No updating or modification operations are permitted.

The Priority Data Base displays when a priority list was created, when that list is effective, the ordered priority list, and for each component in the priority list the subtasks of the activities that are utilizing that component during the effective time period. The user may display any priority list that has been created by the LPLMS.

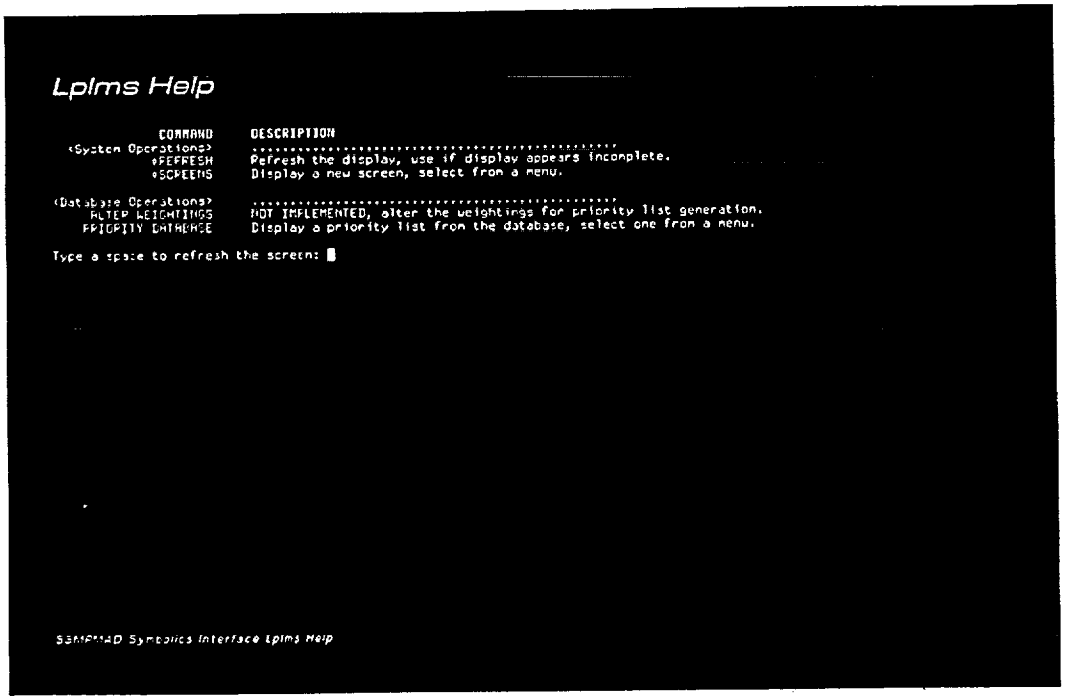
The Weightings is an informational display that shows what weighting criteria was used in the development of the priority list. These ratings are non-modifiable and represent 10-highest to 0-lowest.



```
SS Module Load Priority Maintenance

MISSION TIME      PRIORITY LIST PREPARED      NEXT PRIORITY LIST      CONTROL
[00:00:01]        00:00:00                    14 minutes              HALTED

PRIORITY DATA BASE                                      WEIGHTINGS

Priority List                                           Active LPLMS Weightings
  Created: 00:00:00   Effective: 00:00:00 - 00:00:15
                                                                 PRIORITY  10
 1 G00 (RPC LC6-00 3kw)                                       RESTARTABLE  2
   ELECTROEPITAXY                                           INTERRUPTIBLE  2
           POWER UP & VERIFY       00:00:08   00:00:13            SUCCESS  2
   PURGE & START RECORDERS         00:00:13   00:00:19        OPPORTUNITY  2
                                                                     CREW  1
 2 B08 (RPC LC1-08 1kw)                                          RESOURCE  3
   ELECTROEPITAXY                                        PERCENT COMPLETE  2
           POWER UP & VERIFY       00:00:08   00:00:13        CONSTRAINED  3
   PURGE & START RECORDERS         00:00:13   00:00:19

 3 D02 (RPC LC3-02 1kw)
   ELECTROEPITAXY
           POWER UP & VERIFY       00:00:08   00:00:13
   PURGE & START RECORDERS         00:00:13   00:00:19

 4 D04 (RPC LC3-04 1kw)
   ELECTROEPITAXY
           POWER UP & VERIFY       00:00:08   00:00:13
   PURGE & START RECORDERS         00:00:13   00:00:19

 5 E15 (RPC LC5-15 3kw)

•Refresh      •Screens      Alter Weightings  Priority Database

< Control-L Lcme
< Display Priority List
<
(10 72.10) Pause Breadboard.
```

Figure 15.4.3-1 LPLMS Screen

ORIGINAL PAGE IS OF POOR QUALITY



Lplms Help

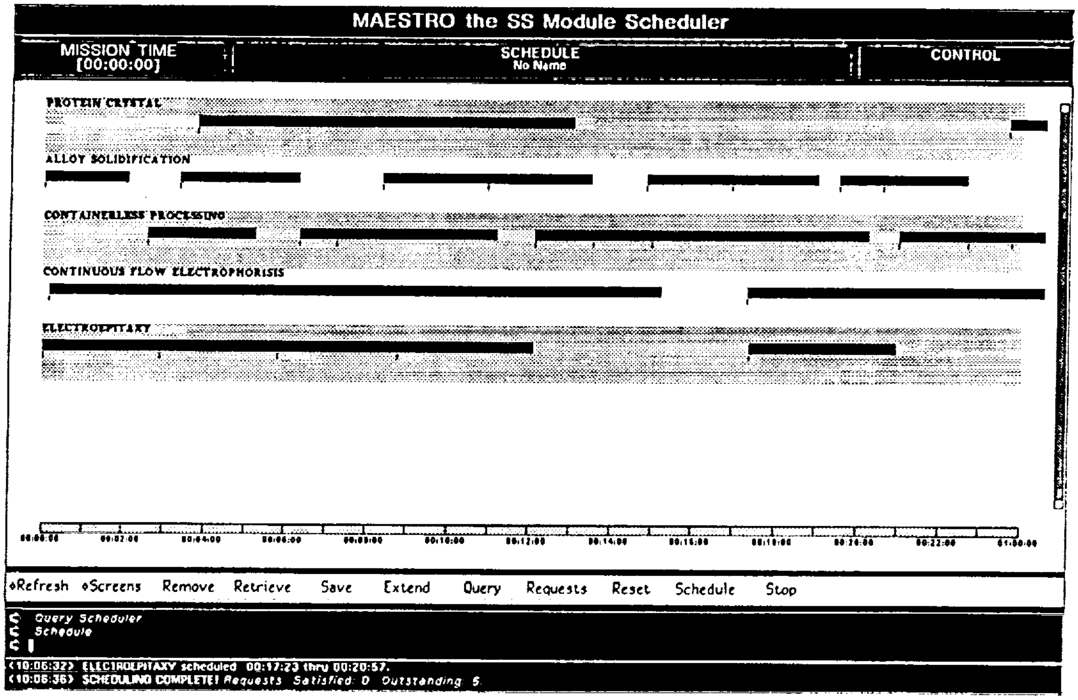
```
              COMMAND     DESCRIPTION
    <System Operations>   ..................................................
               *REFRESH   Refresh the display, use if display appears incomplete.
               *SCREENS   Display a new screen, select from a menu.

  <Database Operations>   ..................................................
       ALTER WEIGHTINGS   NOT IMPLEMENTED, alter the weightings for priority list generation.
      PRIORITY DATABASE   Display a priority list from the database, select one from a menu.

Type a space to refresh the screen: █
```

SSMPMAD Symbolics Interface Lplms Help

Figure 15.4.3-2 LPLMS Help Screen

15.4.4 Scheduler

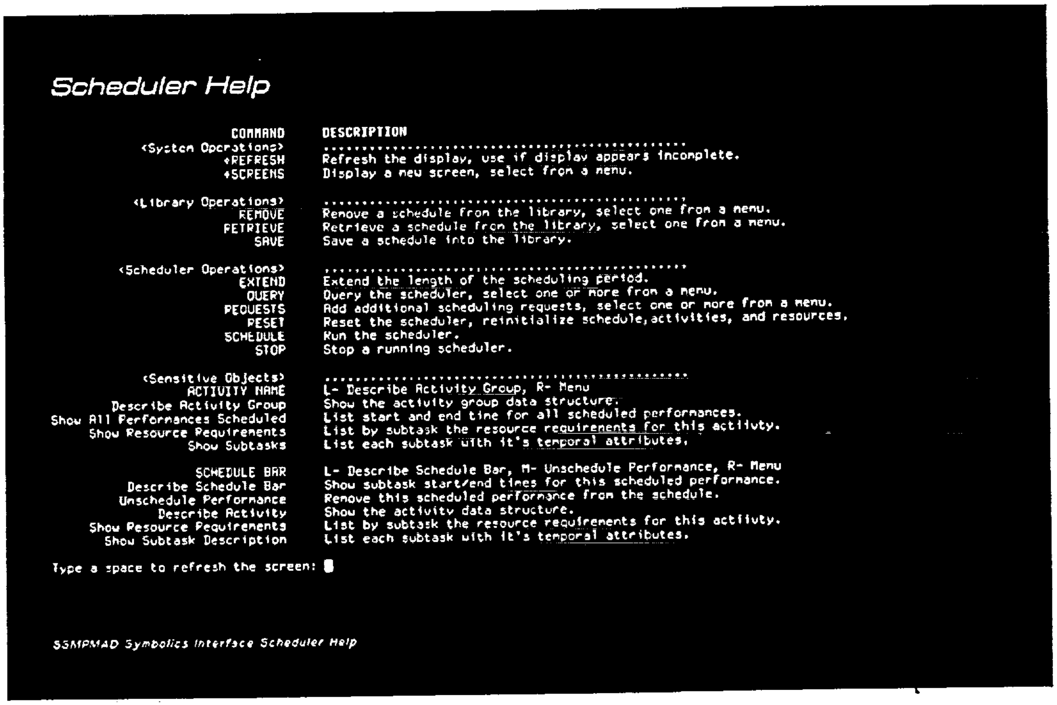
The Scheduler is used for displaying the current schedule as well as for resetting the scheduler, retrieving a schedule from the schedule library and getting information about the scheduled activities. The help screen for the scheduler provides more detail on the available operations on the Scheduler Screen. The following two figures, 15.4.4-1 and 15.4.4-2 reflect the Scheduler and Scheduler Help screens respectively.



```
  Query Scheduler
  Schedule

<10:06:32> ELECTROEPITAXY scheduled 00:17:23 thru 00:20:57.
<10:06:36> SCHEDULING COMPLETE! Requests Satisfied: 0 Outstanding: 5.
```

Figure 15.4.4-1 Scheduler Screen

ORIGINAL PAGE IS
OF POOR QUALITY



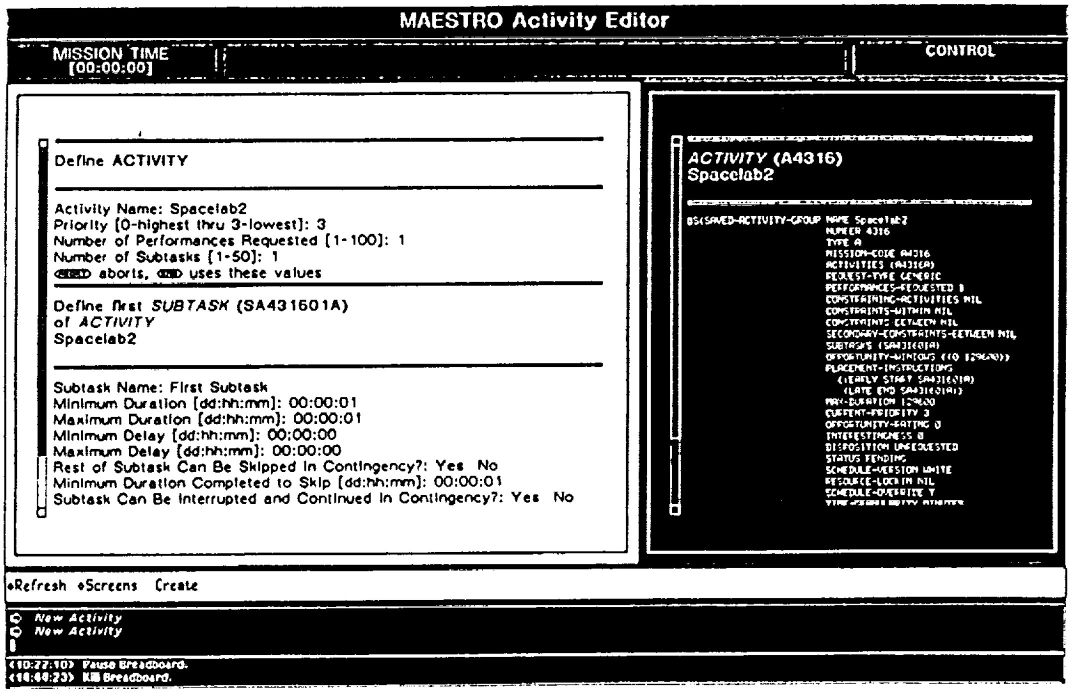
```
Scheduler Help

                          COMMAND    DESCRIPTION
              <System Operations>    ................................................
                         *REFRESH    Refresh the display, use if display appears incomplete.
                         *SCREENS    Display a new screen, select from a menu.

             <Library Operations>    ................................................
                           REMOVE    Remove a schedule from the library, select one from a menu.
                         RETRIEVE    Retrieve a schedule from the library, select one from a menu.
                             SAVE    Save a schedule into the library.

           <Scheduler Operations>    ................................................
                           EXTEND    Extend the length of the scheduling period.
                            QUERY    Query the scheduler, select one or more from a menu.
                         REQUESTS    Add additional scheduling requests, select one or more from a menu.
                            RESET    Reset the scheduler, reinitialize schedule,activities, and resources.
                         SCHEDULE    Run the scheduler.
                             STOP    Stop a running scheduler.

              <Sensitive Objects>    ................................................
                    ACTIVITY NAME    L- Describe Activity Group, R- Menu
          Describe Activity Group    Show the activity group data structure.
    Show All Performances Scheduled  List start and end time for all scheduled performances.
       Show Resource Requirements    List by subtask the resource requirements for this activity.
                    Show Subtasks    List each subtask with it's temporal attributes.

                     SCHEDULE BAR    L- Describe Schedule Bar, M- Unschedule Performance, R- Menu
            Describe Schedule Bar    Show subtask start/end times for this scheduled performance.
          Unschedule Performance     Remove this scheduled performance from the schedule.
                Describe Activity    Show the activity data structure.
       Show Resource Requirements    List by subtask the resource requirements for this activity.
         Show Subtask Description    List each subtask with it's temporal attributes.

Type a space to refresh the screen: █


SSMPMAD Symbolics Interface Scheduler Help
```

Figure 15.4.4-2 Scheduler Help Screen

ORIGINAL PAGE IS
OF POOR QUALITY

15.4.5 Activity Editor

The Activity Editor is used to create activities describing tasks to be scheduled. Each activity has a priority and a number of subtasks to be executed sequentially. The main window is used to enter information about the activities and the subtasks associated with that activity. The inverted window is for display purposes only. Figures 15.4.5-1 and 15.4.5-2 show the Activity Editor and Activity Editor Help screens respectively.



Define ACTIVITY

Activity Name: Spacelab2
Priority [0-highest thru 3-lowest]: 3
Number of Performances Requested [1-100]: 1
Number of Subtasks [1-50]: 1
aborts, uses these values

Define first *SUBTASK* (SA431601A)
of *ACTIVITY*
Spacelab2

Subtask Name: First Subtask
Minimum Duration [dd:hh:mm]: 00:00:01
Maximum Duration [dd:hh:mm]: 00:00:01
Minimum Delay [dd:hh:mm]: 00:00:00
Maximum Delay [dd:hh:mm]: 00:00:00
Rest of Subtask Can Be Skipped In Contingency?: Yes No
Minimum Duration Completed to Skip [dd:hh:mm]: 00:00:01
Subtask Can Be Interrupted and Continued In Contingency?: Yes No

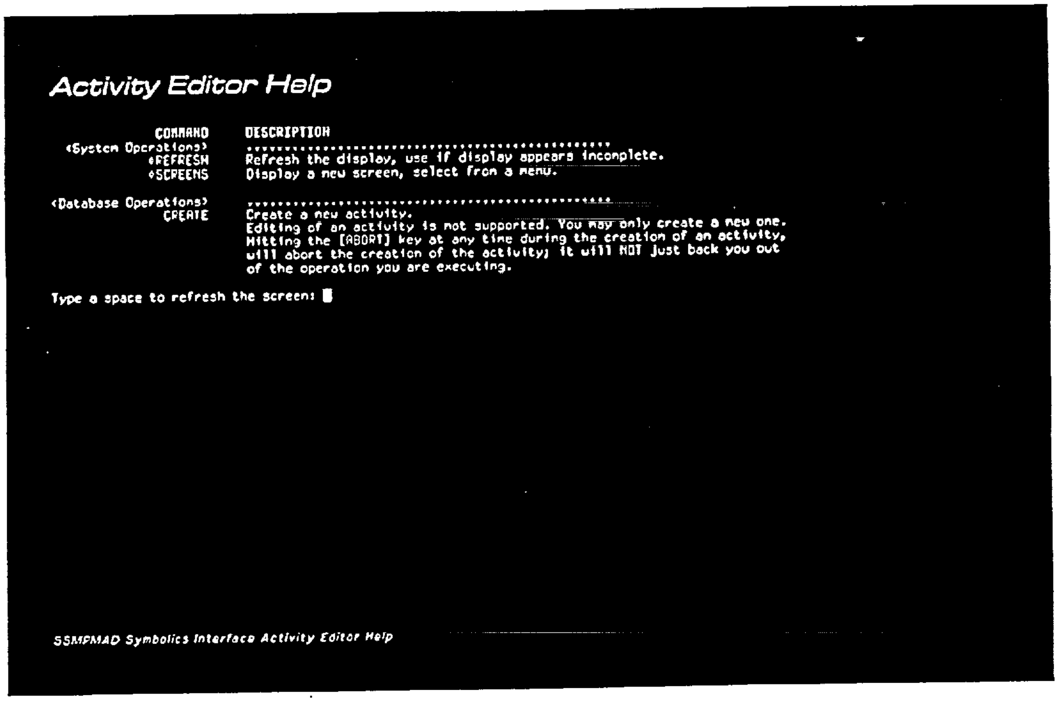
ACTIVITY (A4316)
Spacclab2

```
OS(SAVED-ACTIVITY-GROUP NAME Spacelab2
                        NUMBER 4316
                        TYPE A
                        MISSION-CODE A4316
                        ACTIVITIES (A4316A)
                        REQUEST-TYPE GENERIC
                        PERFORMANCES-REQUESTED 1
                        CONSTRAINING-ACTIVITIES NIL
                        CONSTRAINTS-WITHIN NIL
                        CONSTRAINTS-BETWEEN NIL
                        SECONDARY-CONSTRAINTS-BETWEEN NIL
                        SUBTASKS (SA431601A)
                        OPPORTUNITY-WINDOWS ((0 129600))
                        PLACEMENT-INSTRUCTIONS
                          ((EARLY START SA431601A)
                           (LATE END SA431601A))
                        MAX-DURATION 129600
                        CURRENT-PRIORITY 3
                        OPPORTUNITY-RATING 0
                        INTERESTINGNESS 0
                        DISPOSITION UNREQUESTED
                        STATUS PENDING
                        SCHEDULE-VERSION WHITE
                        RESOURCE-LOCKIN NIL
                        SCHEDULE-OVERRIDE T
                        [illegible]
```

```
New Activity
New Activity
<10:27:10> Pause Breadboard.
<18:48:23> Kill Breadboard.
```

Figure 15.4.5-1 Activity Editor Screen

ORIGINAL PAGE IS
OF POOR QUALITY



Activity Editor Help

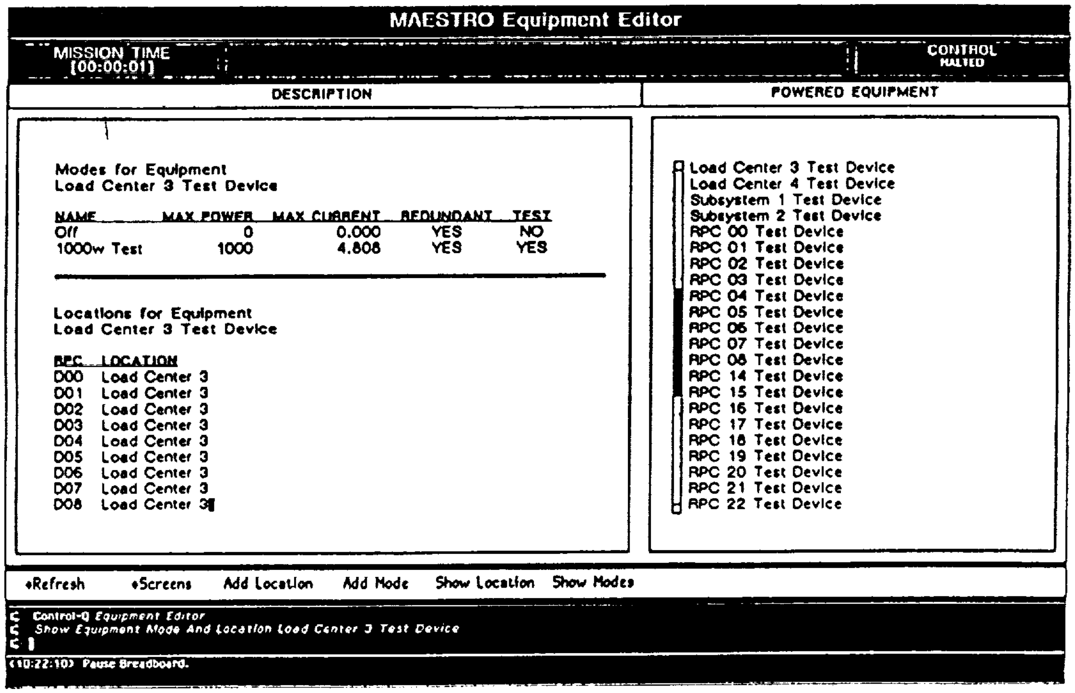
```
              COMMAND    DESCRIPTION
   <System Operations>   ********************************************
              *REFRESH   Refresh the display, use if display appears incomplete.
              *SCREENS   Display a new screen, select from a menu.

 <Database Operations>   ********************************************
                CREATE   Create a new activity.
                         Editing of an activity is not supported. You may only create a new one.
                         Hitting the [ABORT] key at any time during the creation of an activity,
                         will abort the creation of the activity; it will NOT just back you out
                         of the operation you are executing.

Type a space to refresh the screen: █
```

SSMPMAD Symbolics Interface Activity Editor Help

15.4.5-2 Activity Editor Help Screen

15.4.6 Equipment Editor

The Equipment Editor provides the mechanisms for adding modes of operation and locations to equipment. There are two main windows to the Equipment Editor. The Description Window is used for adding modes and locations to equipment, as well as for displaying information about equipment. The Powered Equipment display is a scrollable window of the available equipment. Each line on this display is mouseable. The following two figures, 15.4.6-1 and 15.4.6-2, show a sample display of the Equipment Editor and the Equipment Editor Help screen respectively.



Modes for Equipment
Load Center 3 Test Device

NAME	MAX POWER	MAX CURRENT	REDUNDANT	TEST
Off	0	0.000	YES	NO
1000w Test	1000	4.808	YES	YES

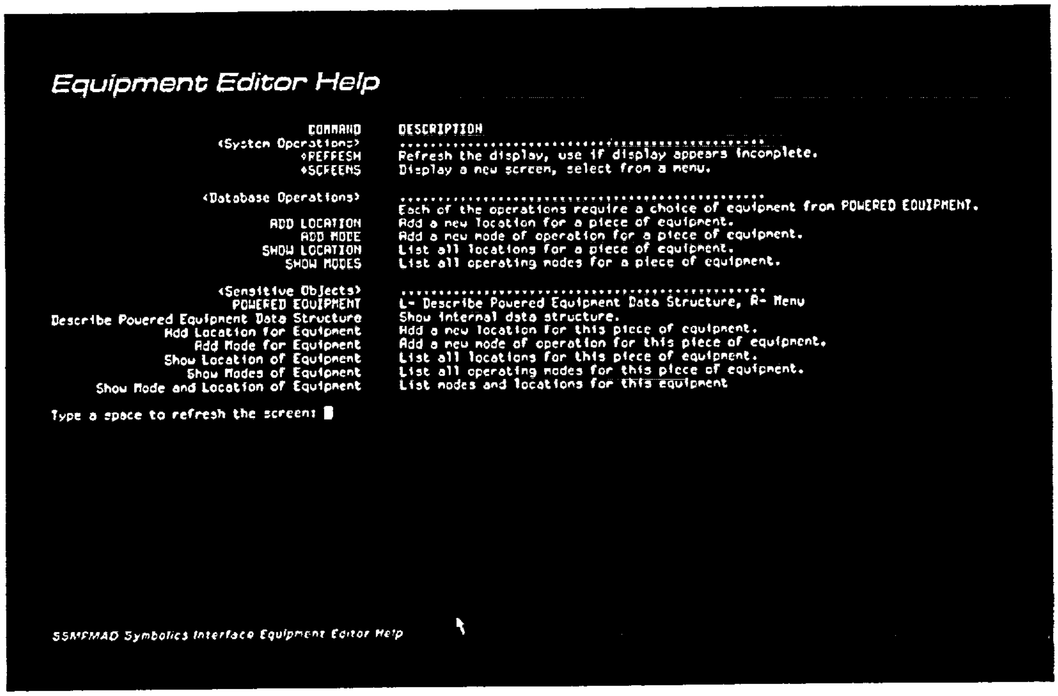
Locations for Equipment
Load Center 3 Test Device

RPC	LOCATION
D00	Load Center 3
D01	Load Center 3
D02	Load Center 3
D03	Load Center 3
D04	Load Center 3
D05	Load Center 3
D06	Load Center 3
D07	Load Center 3
D08	Load Center 3

Load Center 3 Test Device
Load Center 4 Test Device
Subsystem 1 Test Device
Subsystem 2 Test Device
RPC 00 Test Device
RPC 01 Test Device
RPC 02 Test Device
RPC 03 Test Device
RPC 04 Test Device
RPC 05 Test Device
RPC 06 Test Device
RPC 07 Test Device
RPC 08 Test Device
RPC 14 Test Device
RPC 15 Test Device
RPC 16 Test Device
RPC 17 Test Device
RPC 18 Test Device
RPC 19 Test Device
RPC 20 Test Device
RPC 21 Test Device
RPC 22 Test Device

```
C  Control-Q Equipment Editor
C  Show Equipment Mode And Location Load Center 3 Test Device
C
<10:22:10> Pause Breadboard.
```

Figure 15.4.6-1 Equipment Editor Screen



```
Equipment Editor Help

                             COMMAND   DESCRIPTION
                  <System Operations>  ...............................................
                            +REFRESH   Refresh the display, use if display appears incomplete.
                            +SCREENS   Display a new screen, select from a menu.

                <Database Operations>  ...............................................
                                       Each of the operations require a choice of equipment from POWERED EQUIPMENT.
                        ADD LOCATION   Add a new location for a piece of equipment.
                            ADD MODE   Add a new mode of operation for a piece of equipment.
                       SHOW LOCATION   List all locations for a piece of equipment.
                          SHOW MODES   List all operating modes for a piece of equipment.

                  <Sensitive Objects>  ...............................................
                   POWERED EQUIPMENT   L- Describe Powered Equipment Data Structure, R- Menu
Describe Powered Equipment Data Structure   Show internal data structure.
            Add Location for Equipment   Add a new location for this piece of equipment.
                Add Mode for Equipment   Add a new mode of operation for this piece of equipment.
           Show Location of Equipment   List all locations for this piece of equipment.
              Show Modes of Equipment   List all operating modes for this piece of equipment.
     Show Mode and Location of Equipment   List modes and locations for this equipment

Type a space to refresh the screen: █

SSMFMAD Symbolics Interface Equipment Editor Help
```

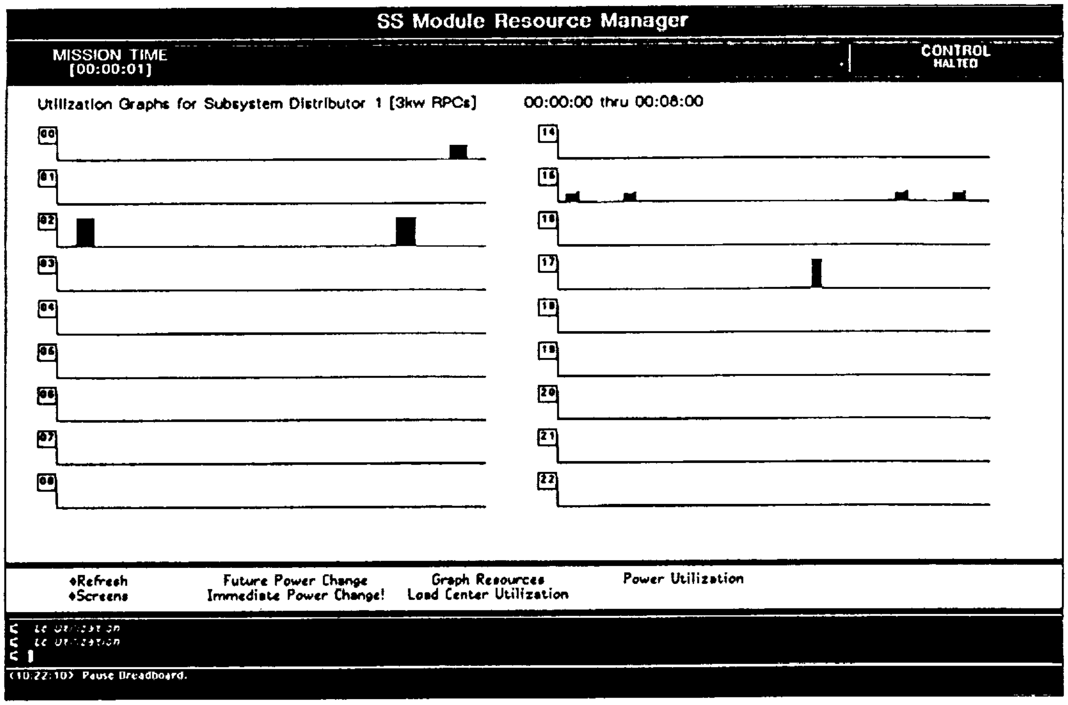
Figure 15.4.6-2 Equipment Editor Help Screen

15.4.7 Resource Manager

The Resource Manager screen provides graphical and textual information related to power resource utilization. The user may display power utilization graphs for the entire system, by load center, or individual components. Another feature of the interface is the specification of a change to the source power in order to effect the breadboard during operations. See Figures 15.4.7-1, 15.4.7-2 and 15.4.7-3 for the Resource Manager, Utilization Graph and Help Screen.

The Interactive Display region of the screen provides two functions: 1) display of power utilization graphs, and 2) input region for Source Power Changes. Graphs displayed are always mouse sensitive providing the capability to display its parent component utilization, a textual list of utilization, or a list of activities that are scheduled to utilize the component.

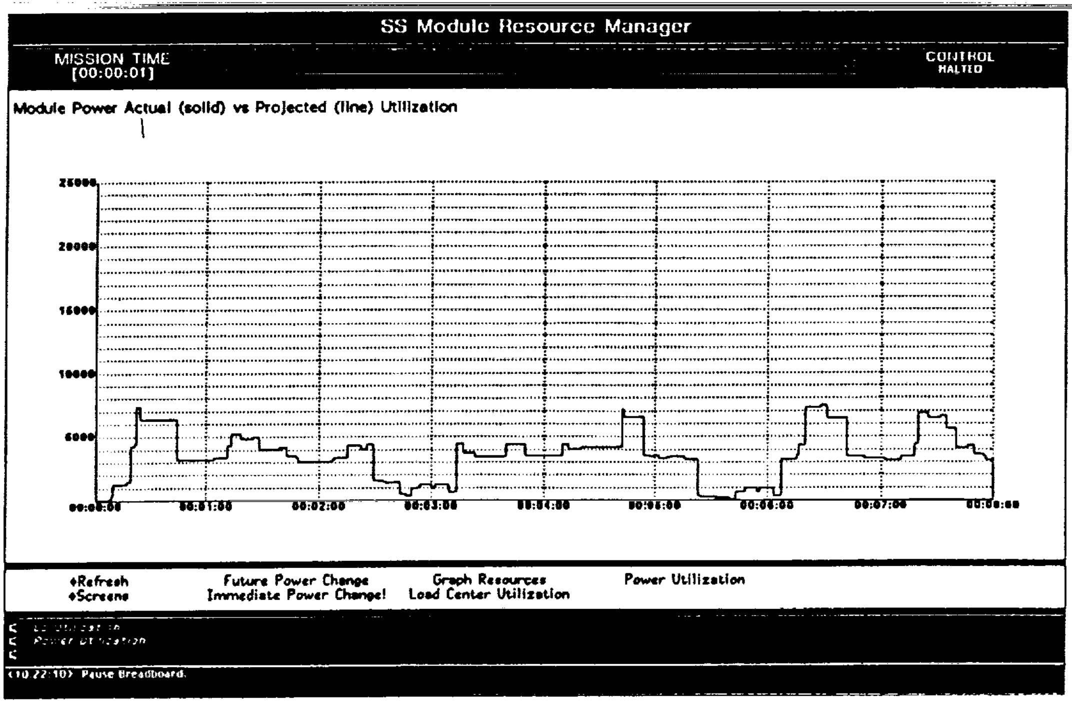
Source power change specifications utilize this area for input of the change in power and the duration of that change.



```
< Lc Utilization
< Lc Utilization
<
```

```
<10:22:10> Pause Breadboard.
```

Figure 15.4.7-1 Resource Manager Screen



SS Module Resource Manager

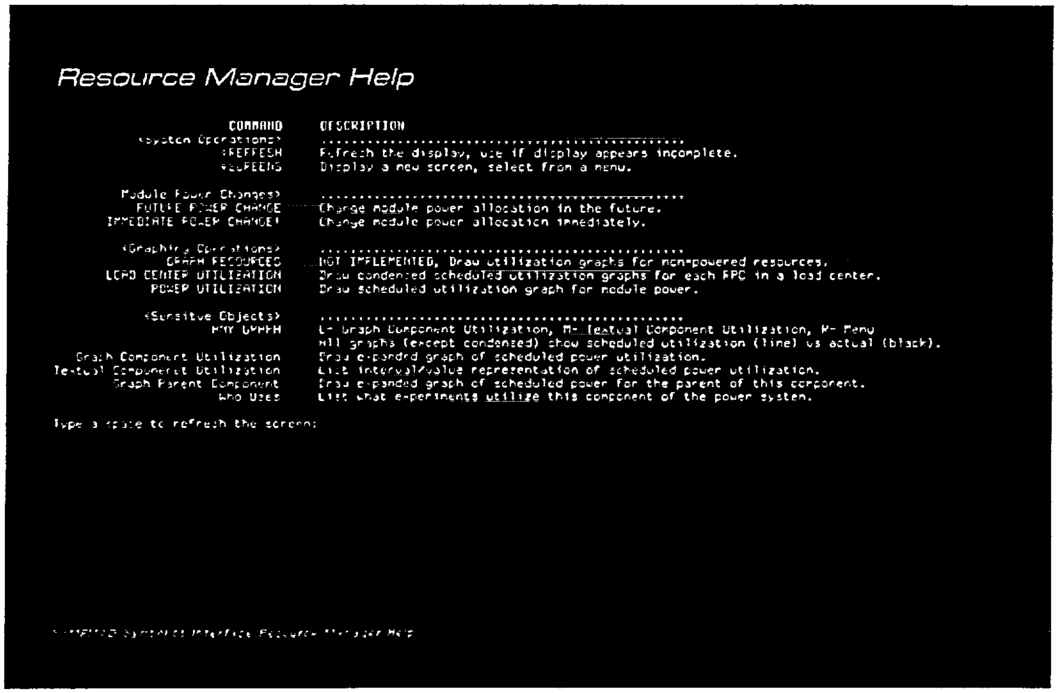
MISSION TIME
[00:00:01]

CONTROL
HALTED

Module Power Actual (solid) vs Projected (line) Utilization

```
<  LC Utilization
<  Power Utilization
<
<10:22:10> Pause Breadboard.
```

Figure 15.4.7-2 Resource Manager Screen



Resource Manager Help

```
                    COMMAND    DESCRIPTION
        <System Operations>    ............................................
                    REFRESH    Refresh the display, use if display appears incomplete.
                    SCREENS    Display a new screen, select from a menu.

     <Module Power Changes>    ............................................
        FUTURE POWER CHANGE    Change module power allocation in the future.
     IMMEDIATE POWER CHANGE    Change module power allocation immediately.

      <Graphing Operations>    ............................................
            GRAPH RESOURCES    NOT IMPLEMENTED, Draw utilization graphs for non-powered resources.
    LOAD CENTER UTILIZATION    Draw condensed scheduled utilization graphs for each RPC in a load center.
          POWER UTILIZATION    Draw scheduled utilization graph for module power.

        <Sensitive Objects>    ............................................
                  ANY GRAPH    L- Graph Component Utilization, M- Textual Component Utilization, R- Menu
                               All graphs (except condensed) show scheduled utilization (line) vs actual (black).
Graph Component Utilization    Draw expanded graph of scheduled power utilization.
Textual Component Utilization  List interval/value representation of scheduled power utilization.
     Graph Parent Component    Draw expanded graph of scheduled power for the parent of this component.
                  Who Uses     List what experiments utilize this component of the power system.

Type a space to refresh the screen:
```

[illegible] Symbolics Interface Resource Manager Help

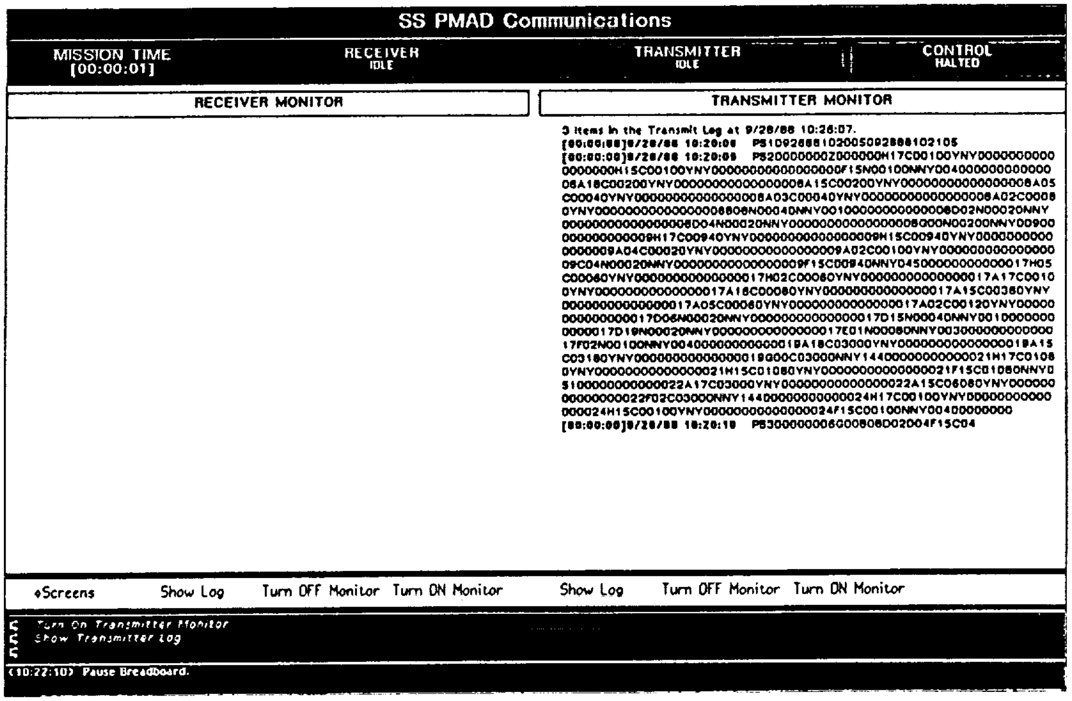
Figure 15.4.7-3 Resource Manager Help Screen

15.4.8 Communications

The Communications screen displays raw transactions that are received by the Receiver or transmitted by the Transmitter. When the monitors are on, as each transaction is processed it is displayed; if the monitors are off, no display of the transactions will occur until the monitor is turned on. See Figures 15.4.8-1 and 15.4.8-2 for the Communications and its associated Help Screen.

The Transmitter Monitor displays each transactions that is transmitted over the network by the SI. The actual time and mission time are displayed along with each transaction's "raw" contents. By mousing on a transaction the user may see the deblocked version of the transaction.

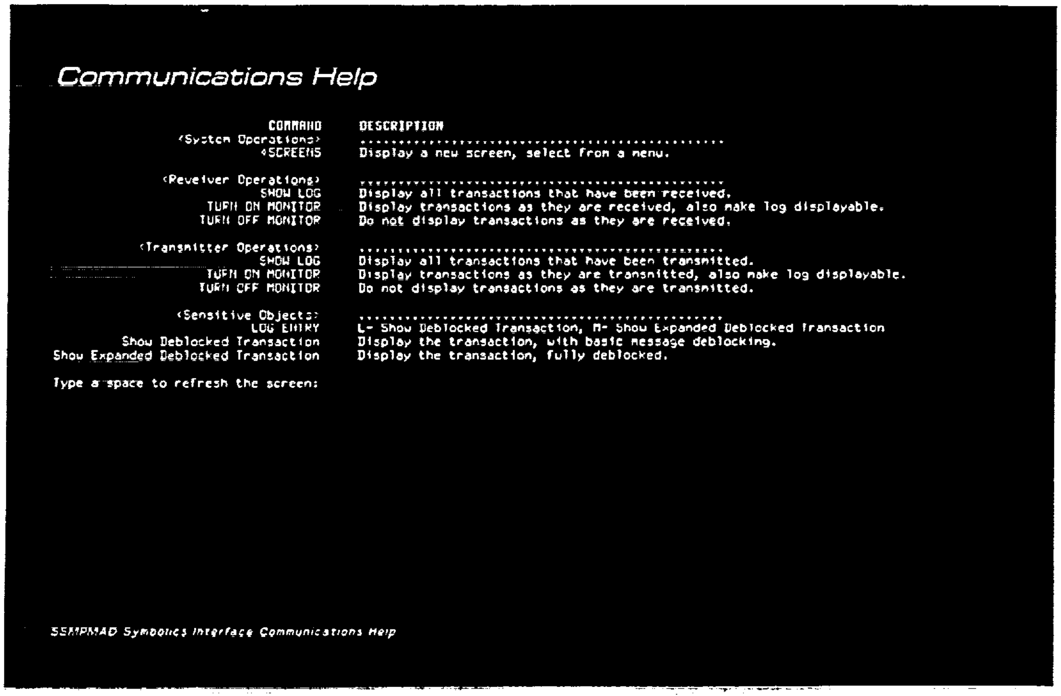
The Receiver Monitor displays each transaction that is received over the network by the SI. The actual time and mission time are displayed along with each transaction's "raw" contents. By mousing on a transaction the user may see the deblocked version of the transaction.



```
3 Items in the Transmit Log at 9/28/88 10:26:07.
[00:00:00]9/28/88 10:20:06   PS1092888102005092888102105
[00:00:00]9/28/88 10:20:09   PS200000000Z000000H17C00100YNY0000000000
0000000H15C00100YNY000000000000000000F15N00100NNY004000000000000
06A18C00200YNY000000000000000006A15C00200YNY000000000000000006A05
C00040YNY000000000000000006A03C00040YNY000000000000000006A02C0008
0YNY000000000000000006B06N00040NNY001000000000000006D02N00020NNY
000000000000000006D04N00020NNY00000000000000000B00N00200NNY00900
000000000009H17C00940YNY00000000000000009H15C00940YNY0000000000
0000009A04C00020YNY000000000000000009A02C00100YNY000000000000000
09C04N00020NNY000000000000000009F15C00940NNY045000000000000017H05
C00060YNY000000000000000017H02C00060YNY000000000000000017A17C0010
0YNY000000000000000017A18C00060YNY000000000000000017A15C00360YNY
000000000000000017A05C00060YNY000000000000000017A02C00120YNY00000
0000000000017D06N00020NNY000000000000000017D15N00040NNY0010000000
0000017D19N00020NNY000000000000000017E01N00080NNY003000000000000
17F02N00100NNY004000000000000019A18C03000YNY000000000000000019A15
C03180YNY000000000000000019G00C03000NNY144000000000000021H17C0108
0YNY000000000000000021H15C01080YNY000000000000000021F15C01080NNY0
51000000000000022A17C03000YNY000000000000000022A15C06080YNY000000
0000000022F02C03000NNY144000000000000024H17C00100YNY0000000000
000024H15C00100YNY000000000000000024F15C00100NNY0040000000
[00:00:00]9/28/88 10:20:10   PS300000006G00808D02004F15C04
```

```
⊊   Turn On Transmitter Monitor
⊊   Show Transmitter Log
⊊
(10:22:10) Pause Breadboard.
```

Figure 15.4.8-1 Communications Screen



Communications Help

```
                               COMMAND    DESCRIPTION
                      <System Operations>    ............................................
                                 <SCREENS    Display a new screen, select from a menu.

                     <Reveiver Operations>    ............................................
                                 SHOW LOG    Display all transactions that have been received.
                           TURN ON MONITOR    Display transactions as they are received, also make log displayable.
                          TURN OFF MONITOR    Do not display transactions as they are received.

                  <Transmitter Operations>    ............................................
                                 SHOW LOG    Display all transactions that have been transmitted.
                           TURN ON MONITOR    Display transactions as they are transmitted, also make log displayable.
                          TURN OFF MONITOR    Do not display transactions as they are transmitted.

                       <Sensitive Objects:    ............................................
                                LOG ENTRY    L- Show Deblocked Transaction, M- Show Expanded Deblocked Transaction
               Show Deblocked Transaction    Display the transaction, with basic message deblocking.
      Show Expanded Deblocked Transaction    Display the transaction, fully deblocked.

Type a space to refresh the screen:
```

Figure 15.4.8-2 Communications Help Screen

APPENDIX VII – ICD

16.0 APPENDIX VII: ICD

The following are SIC (Switchgear Interface Card) to LLP (Lowest Level Processor) commands, formats, and expected responses. The COMMANDS are messages from the LLP to the SIC. The RESPONSE is the actual data returned from the SIC in response to a command. The LLP will wait for a RESPONSE from the SIC after each command is sent. If no RESPONSE is received within 2 seconds, the SIC card will be considered nonfunctional. All COMMANDS sent to the SIC card will end with a CR (Carriage Return) which flags end of transmission to the firmware on the MVME331 card (intelligent communications controller). All RESPONSES from the SIC will also end with a CR for the same reason. The MVME331 card removes the CR before transmission from the SIC to LLP and from the LLP to the SIC.

NOTES:

The dip switch configuration for SIC is as follows:

Switch 1 - switch open (off) - bit0 high
Switch 2 - switch open (off) - bit1 high
Switch 3 - switch open (off) - bit2 high
Switch 4 - switch open (off) - bit3 high

The SIC port configuration is as follows:

Baud rate - 9600
Data bits - 8
Stop bits - 1
Parity - even

Status Format:

```
*************************************************************
 *    byte1    *       byte2      *       byte3      *    byte4    *
*************************************************************
```

```
where:  byte1  -> $30 -- status OK
               -> $31 -- status NOT OK

        byte2  -> cc  -- copy of command received
                         with MSB bit always set to 1

        byte3  -> $80 -- status OK
               -> $FF -- unknown command
               -> $81 -- first  byte not a command byte
               -> $82 -- did not receive first data byte
               -> $83 -- first data byte msb not high
               -> $84 -- did not receive second data byte
               -> $85 -- second data byte msb not high
               -> $86 -- switch already on
               -> $87 -- switch already tripped when
                         tried to turn it on
               -> $88 -- switch already off
               -> $89 -- switch already tripped when
                         tried to turn it off
               -> $8A -- GC Data Valid error when
                         getting switch data
```

<u>NOTE</u>: If the following statuses are received, do not 'download' switch settings

```
               -> $8B -- continous buffer overflow
                         (reset continous buffer)
               -> $8C -- once buffer overflow
                         (redo once buffer)
```

<u>NOTE</u>: If the following statuses are received, the SIC card must be reset or must use the redundant SIC

-> $A1 -- SIC character buffer overrun
-> $A2 -- character overwritten (OE)
-> $A4 -- parity error from UART (PE)
-> $A6 -- OE and PE
-> $A8 -- framing error (FE)
-> $AA -- FE and OE
-> $AC -- FE and PE
-> $AE -- FE and OE and PE
-> $F7 -- SIC internal memory parity error

byte4-> $0D -- end of status

Command Word Format:

```
************************************************************
 *    byte1    *      byte2      *      byte3     *    byte4    *
************************************************************
```

where: byte1 -> cc -- command
byte2 -> dd1 -- first byte of data word
byte3 -> dd2 -- second byte of data word
byte4 -> $0D -- end of command

Switchword Format:

	bit14=0 (switch not tripped)	bit14=1 (tripped)
bit0	current (1)	trippedsurge current H
bit1	current (1)	tripped fast trip H
bit2	current (1)	spare
bit3	current (1)	spare2

bit4	current (1)	tripped overcurrent (i^2t) H
bit5	current (1)	tripped undervoltage H
bit6	current MSB (1)	tripped grnd fault H
bit7	always 1	always 1
bit8	current overrange H (1)	tripped overtemp latched H
bit9	S2 solid state swtch on H	S2 solid state swtch on H
bit10	S1 mech switch on H	S1 mech switch on H
bit11	overtemperature H	overtemperature H
bit12	off control input H (2)	off control input H (2)
bit13	on control input H (2)	on control input H (2)
bit15	always 1	always 1

(1) RMS current

(2)

bit13	bit12	RPC command
0	0	on (error in hardware)
0	1	on
1	0	off
1	1	no change

GC Data Valid word format:

bit0 -> GC Data Valid switch 7 H
bit1 -> GC Data Valid switch 8 H
bit2 -> GC Data Valid switch 9 H
bit3 -> GC Data Valid switch 10 H
bit4 -> GC Data Valid switch 11 H
bit5 -> GC Data Valid switch 12 H
bit6 -> GC Data Valid switch 13 H
bit7 -> always 1
bit8 -> GC Data Valid switch 0 H
bit9 -> GC Data Valid switch 1 H
bit10 -> GC Data Valid switch 2 H

bit11 -> GC Data Valid switch 3 H
bit12 -> GC Data Valid switch 4 H
bit13 -> GC Data Valid switch 5 H
bit14 -> GC Data Valid switch 6 H
bit15 -> always 1

NOTE: L - data valid
H - data not valid

Sensorword Format:

bit0 -> sensor data bit 0
bit1 -> sensor data bit 1
bit2 -> sensor data bit 2
bit3 -> sensor data bit 3
bit4 -> don't care
bit5 -> don't care
bit6 -> don't care
bit7 -> always 1
bit8 -> sensor data bit 4
bit9 -> sensor data bit 5
bit10 -> sensor data bit 6
bit11 -> sensor data bit 7
bit12 -> don't care
bit13 -> don't care
bit14 -> don't care
bit15 -> always 1

A current/voltage sensorword_set consists of 9 sensorwords of the above format for a given current/voltage sensor. The 9 sensorwords will be of the following order:

V rms

I rms

V offset

I offset

V instantaneous

I instantaneous

P instantaneous

P real

frequency

In this document the notation sensorword_set_n will mean the 9 sensorwords of the described sensorword format in the described order for a given voltage/current sensor "n" where n can be sensor/voltage sensor 0 to 15

1) COMMAND: execute SIC firmware reset (does not reset actual set configuration)

FORMAT:
cc --> $24
dd1 --> $80
dd2 --> $80

RESPONSE:
- set up 2 sec timeout
- four bytes of data plus the status as described in the NOTES where the first two bytes give the following data:

bit 0 -> 0 if GC7 connected, 1 if not
bit 1 -> 0 if GC8 connected, 1 if not
bit 2 -> 0 if GC9 connected, 1 if not
bit 3 -> 0 if GC10 connected, 1 if not
bit 4 -> 0 if GC11 connected, 1 if not
bit 5 -> 0 if GC12 connected, 1 if not
bit 6 -> 0 if GC13 connected, 1 if not
bit 7 -> always 1

bit 8 -> 0 if GC0 connected, 1 if not
bit 9 -> 0 if GC1 connected, 1 if not
bit 10 -> 0 if GC2 connected, 1 if not
bit 11 -> 0 if GC3 connected, 1 if not
bit 12 -> 0 if GC4 connected, 1 if not
bit 13 -> 0 if GC5 connected, 1 if not
bit 14 -> 0 if GC6 connected, 1 if not
bit 15 -> always 1

the third byte gives the following data:

bit 0 -> current SIC switch0 setting
bit 1 -> current SIC switch1 setting
bit 2 -> current SIC switch2 setting
bit 3 -> current SIC switch3 setting
bit 4 -> 0 if A/D connected, 1 if not
bit 5 -> don't care
bit 6 -> don't care
bit 7 -> always 1

the fourth byte gives the following data:

bit 0 -> don't care
bit 1 -> don't care
bit 2 -> don't care
bit 3 -> don't care
bit 4 -> don't care
bit 5 -> don't care
bit 6 -> don't care
bit 7-> always 1

2) COMMAND: reset switch

FORMAT: cc --> $22

dd1 --> $80 + j

j -- 7 bits corresponding to the switches as follows:

bit 0 -> switch 0
bit 1 -> switch 1
bit 2 -> switch2
bit 3 -> switch 3
bit 4 -> switch 4
bit 5 -> switch 5
bit 6 -> switch 6

dd2 --> $80 + k

k -- 7 bits corresponding to the switches as follows:

bit 0 -> switch 7
bit 1 -> switch 8
bit 2 -> switch 9
bit 3 -> switch 10
bit 4 -> switch 11
bit 5 -> switch 12
bit 6 -> switch 13

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

3) COMMAND: command switch on checking switch on or tripped status first; if any of the above conditions exist, the switch command for that particular switch or switches is not executed

FORMAT:
```
cc   --> $2E
dd1 --> $80 + j  (j is defined in (2))
dd2 --> $80 + k  (k is defined in (2))
```

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

4) COMMAND: command switch off checking switch off or tripped status first; if any of the above conditions exist, the switch command for that particular switch or switches is not executed

FORMAT:
```
cc   --> $2F
dd1 --> $80 + j  (j is defined in (2))
dd2 --> $80 + k  (k is defined in (2))
```

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

5) COMMAND: command switch on immediately even if already on or tripped

FORMAT:
```
cc   --> $21
dd1 --> $80 + j  (j is defined in (2))
dd2 --> $80 + k  (k is defined in (2))
```

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

6) COMMAND: command switch off immediately even if already off or tripped

FORMAT: cc --> $20
dd1 --> $80 + j (j is defined in (2))
dd2 --> $80 + k (k is defined in (2))

RESPONSE:- set up 2 sec timeout
- status as described in the NOTES

7) COMMAND: get data for one specified switch a specified number of times

FORMAT: cc --> $2C
dd1 --> $80 + j (j is defined as 1 to $7F depending on the number of times data is specified to be taken -- input buffer must be taken into account)
dd2 --> $80 + k (k is defined as 0 to $D depending on the switch specified)

RESPONSE: -set up 2 sec timeout
- data defined as:
j number of 16-bit switchwords plus the status as described in the notes

8) COMMAND: get data for one specified sensor a specified number of times

FORMAT: cc --> $2D
dd1 --> $80 + j (j is defined as 1 to $EF depending on the number of times data is specified to be taked)

dd2 --> $80 + k (k is defined as 0 to $F depending on the sensor specified)

RESPONSE: - set up 2 sec timeout
- data defined as:
j number of sensorword_set_n for the specified sensor plus the status as described in the NOTES

9) COMMAND: get data for all fourteen switches a specified number of times.

FORMAT: cc --> $30
dd1 --> $80 + j (j is defined as 1 to $7F depending on the number of times data is specified to be taken, input buffer size must be taken into account)
dd2 --> $80

RESPONSE: - set up 2 sec timeout
- data defined as:
(j times (fourteen switchwords plus GC Data Validword set)) plus the status as described in the NOTES

10) COMMAND: get data for all sixteen sensors one time
FORMAT: cc --> $31
dd1 --> $80
dd2 --> $80

RESPONSE: - set up 2 sec timeout
- data defined as:
sixteen sensorword_set_n plus status
as described in the NOTES

11) COMMAND: select GC (all GC select codes will be set to zero)
FORMAT: cc --> $23
dd1 --> $86
dd2 --> $85

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

12) COMMAND: reset continuous buffer
FORMAT: cc --> $25
dd1 --> $80
dd2 --> $80

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

13) COMMAND: fill continuous buffer (First use reset continous buffer then use this command to download code that is to be continuously executed. Code will start executing as soon as the download is started. Up to 80 of these commands may be concatenated before the buffer space is overrun.)

FORMAT: cc --> $26
ee1 --> $80 + q (q is defined as higher 4 bits of
8-bit code(see sensorword))

ee2 --> $80 + r (r is defined as lower 4 bits of
8-bit code (see sensorword))

At the end of the command is appended a $26 until the last command, then a $0D is appended.

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

14) COMMAND: fill once buffer (This command is used to download code that is to be executed only once. Code execution is started by the trigger once buffer command. Up to 80 of these commands may be concatenated before the buffer space is overrun.)

FORMAT: cc --> $27
ee1 --> $80 + q (q is defined as (13))
ee2 --> $80 + r (r is defined in (13))
ee3 --> as defined in (13)

At the end of the of the command or commands is appended a $0D.

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

15) COMMAND: trigger once buffer
FORMAT: cc --> $2A
dd1 --> $80
dd2 --> $80

RESPONSE: - set up 2 sec timeout
- status as described in the NOTES

16) COMMAND: get buffered data
FORMAT: cc --> $29
dd1 --> $80 + v (v is defined as:

bit0 -> buffer0
bit1 -> buffer1
bit2 -> buffer2
bit3 -> buffer3
bit4 -> don't care
bit5 -> don't care
bit6 -> don't care)

dd2 --> $80

RESPONSE: - set up 2 sec timeout
- data of the following format and status as described in NOTES
HEADER - $20

$ssssss - three bytes of status
$8F - dip switch setting for SIC card (if not $8F, SIC card not installed)
$nnnn - position in loop counter
$kk - times through loop counter
$mm - breakpoint
$22 - start of data

--> 14 switchwords plus 1 GC Data Valid word

NOTE: TM is temperature multiplexed, TC is temperature common (TM is not useful)

temperature sensorwords 0TM,
0TC, 1TM, 1TC, 2TM,
2TC, 3TM, 3TC
frequency sensorword 0
sensor_word_set_0
frequency sensorword 1
sensor_word_set_1
frequency sensorword 2
sensor_word_set_2
frequency sensorword 3
sensor_word_set_3
---> $22 - end of buffer
repeat arrowed sections for sensors
4 to 7, 8 to 11, and 12 to 15

17) COMMAND: get power factor and sign (To calculate the power factor use pf1=[Pavg1/[Vrms1*Irms1]. Use the same calculation to determine pf2 using Pavg2, Vrms2, and Irms2; if pf2 < pf1 denotes capacitive loading; if pf2>=pf1 denotes inductive loading; ie, voltage leading current)

FORMAT:
cc --> $2B
dd1 --> $80 + j (j is defined as 0 to $F depending
on sensor pair used)
dd2 --> $80

RESPONSE:
- set up 2 sec timeout
- data defined as six sensor words for the specified in the following order plus status as described in the NOTES.

V rms1
I rms1
P real1
V rms2
I rms2
P real2

18) COMMAND: get all 16 temperature sensor readings one time

FORMAT:
cc --> $32
dd1 --> $80
dd2 --> $80

RESPONSE:
- set up 2 sec timeout
- 16 * 2 sensorwords for the temperature sensors and the status as described in the NOTES

19) COMMAND: get all 16 power factors and signs
(To calculate the power factors see (17))

FORMAT:
cc --> $33
dd1 --> $80
dd2 --> $80

RESPONSE:
- set up 2 sec timeout
- data defined as 16 * (six sensor words for each sensor in the following order) plus the status as described in the NOTES.

V rms1

I rms1

P real1

V rms2

I rms2

P real2

APPENDIX VIII – BREADBOARD USAGE

MCR-88-624
Contract No. NAS8-36433

Post-Installation Test and Usage Plan for SSM/PMAD

January 1989

SPACE STATION AUTOMATION OF COMMON MODULE POWER MANAGEMENT AND DISTRIBUTION

K. A. Freeman

MARTIN MARIETTA ASTRONAUTICS GROUP
SPACE SYSTEMS COMPANY
P.O. BOX 179
DENVER, COLORADO 80201

FOREWORD

This report was prepared by Martin Marietta Space Systems Division for the National Aeronautics and Space Administration, George C. Marshall Space Flight Center, in response to Contract NAS8-36433, and is submitted as the Post-Installation Test and Usage Plan as required in the contract data requirements list.

TABLE OF CONTENTS

		Page
	INTRODUCTION......................................	1
I.	Purpose...	2
II.	Scope...	2
III.	Proposed Uses of the SSM/PMAD Breadboard..........	2
III.A.	Overview..	2
III.B.	Concept Evaluation Facility.......................	4
III.C.	Expert System Test Bed............................	5
III.D.	Function Simulator and Real-Time Software Test Bed	5
III.E.	System Autonomy Demonstration Program Test Bed....	5
III.F.	Development Toward EPS Hardware Test Bed Usage....	5
III.G.	Development Toward Astronaut Training Facility....	6
IV.	Exercising the SSM/PMAD...........................	6
IV.A.	Producing Fault Conditions in Hardware............	6
IV.A.1.	Load Operations and Faults in a 1kW Load Circuit..	6
IV.A.1.1	Load Simulation and Load Bank Concept.............	6
IV.A.1.2	Load Circuit Faults...............................	13
IV.A.2.0	1kW Remote Power Controller (RPC).................	18
IV.A.2.1	RPC Operation.....................................	18
IV.A.2.1.a	Starting Loads....................................	18
IV.A.2.1.b	Supplying Loads...................................	18
IV.A.2.1.c	Stopping Loads....................................	18
IV.A.2.2	Forcing the RPC to Trip...........................	18
IV.A.3	Current Sensor....................................	23
IV.A.4	Cable...	23
IV.A.5	3kW Remote Power Controller.......................	24
IV.A.6	Cable...	24
IV.A.7	10kW Remote Controlled Circuit Breaker............	24
IV.B.	Breadboard Power Revision.........................	24
IV.C.	Forcing a Critical Load to Change to a Redundant Power Channel..............................	25

LIST OF FIGURES

Figure		Page
1.	MSFC Testbed Block Diagram........................	3
2.	Sample Faults and Symptom Sets....................	7
3.	Fault Relay Installation Guidelines...............	14
4A.	1kW Fault Relay Layout............................	15
4B.	3kW Fault Relay Layout............................	16
4C.	10kW Fault Relay Layout...........................	17
5A.	Remote Power Controller Functions.................	19
5B.	Remote Power Controller Basic Schematic...........	20

LIST OF TABLES

Table		Page
1.	Load Resistance Specifications....................	8
2.	Series Inductive Load Circuit.....................	9
3.	Parallel Capacitive Load Circuit..................	10
4.	Theory and Formulae...............................	11-13

INTRODUCTION

This document is in response to the derived Test Plan Development requirement under Task 3 of the Statement of Work for the Automation of the Common Module Power Management and Distribution (ACM/PMAD) contract.

I. Purpose

The purpose of this document is to make recommendations regarding the appropriate purpose and use of the SSM/PMAD breadboard power system at Marshall Space Flight Center. This document is produced under NASA contract NAS 8-36433, and fulfills a contractual requirement.

II. Scope

The recommendations contained in this document are limited in application to the SSM/PMAD breadboard. They should not be construed as being applicable to any other Space Station system breadboard or to any other power system breadboard, nor should they be applied to any other breadboard.

III. Proposed Uses of the SSM/PMAD Breadboard

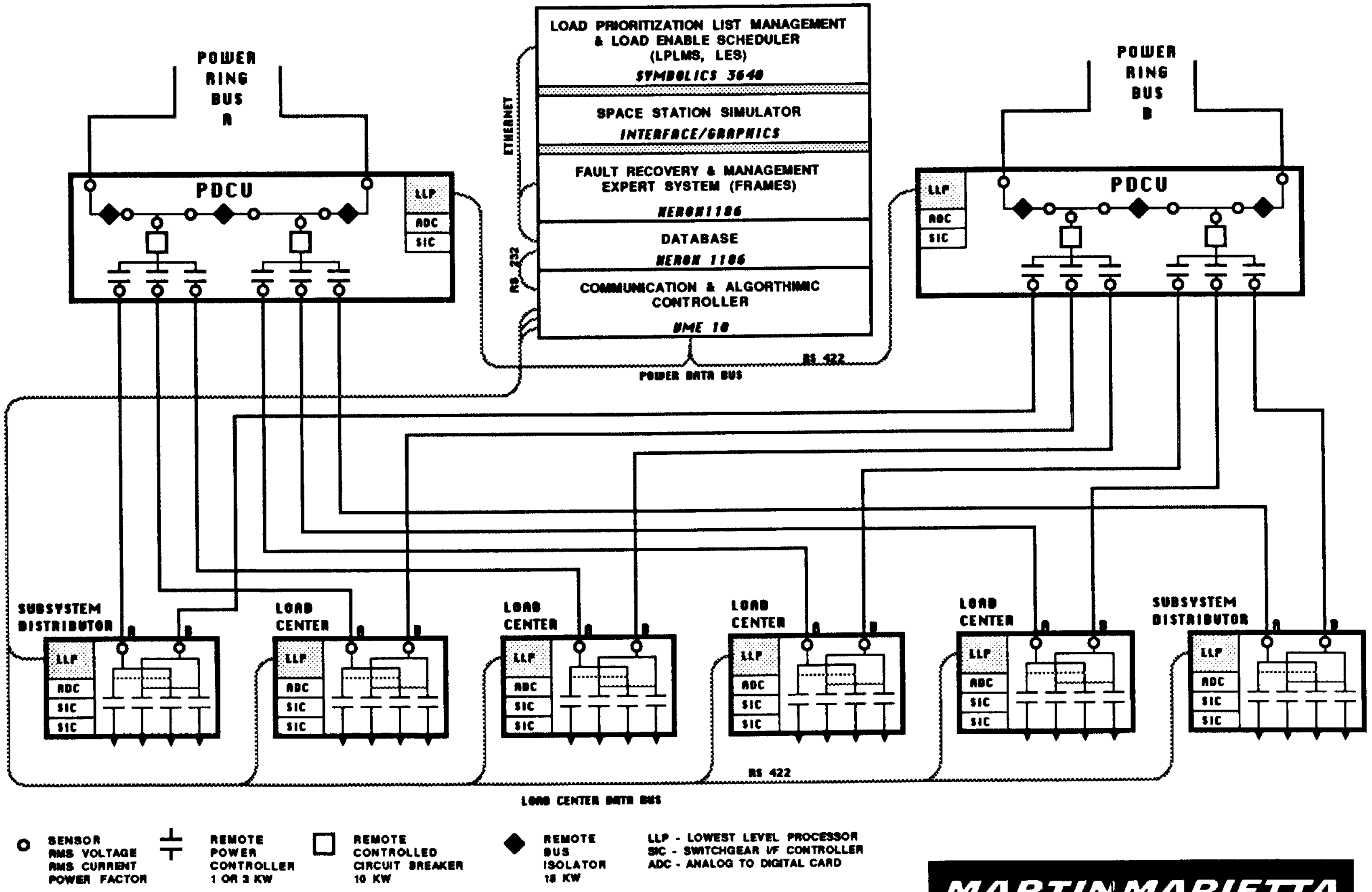
A. Overview

The SSM/PMAD Breadboard is composed of many components, each of which must be verified individually. In turn, each subassembly must undergo testing to be sure it functions properly as a unit. Finally, the entire breadboard must be tested as a system in order to verify that it performs in accordance with the intent of its designers and meets all of its requirements. These levels of testing are not addressed in this document. What is addressed is the question of what utility the SSM/PMAD has once it is verified as a system.

The SSM/PMAD is a dual channel 20kHz power system large enough to operate a substantial number of realistically sized loads simultaneously and autonomously. Its architecture and functionality are based upon the requirements of a Space Station Core Module, but implemented in a ground-based laboratory. Topology is as shown in Figure 1, and is clearly reflective of that directed in JSC 30263 (January 15, 1987 issue), "Architectural Control Document for Electrical Power System (for Space Station)." Vendor-supplied components were chosen for their suitability to a research and development environment and their ability to provide required performance in the appropriate areas rather than their similarity to flight hardware or nature as flight hardware. Components and software items that were produced specifically for this breadboard were specified to implement functions that are expected to be requirements on Space Station. The physical implementations of hardware and the required platforms for the software are not of flight quality but can be considered functional emulations of flight hardware.

Therefore, simulation should remain on a functional basis. The traditional hardware-level system performance tests can be performed and should be performed, but the data may not be applicable to the Space Station or any other power

MARSHALL SPACE FLIGHT CENTER TESTBED


POWER RING BUS A
PDCU
LLP
ADC
SIC
LOAD PRIORITIZATION LIST MANAGEMENT & LOAD ENABLE SCHEDULER (LPLMS, LES)
SYMBOLICS 3640
SPACE STATION SIMULATOR
INTERFACE/GRAPHICS
FAULT RECOVERY & MANAGEMENT EXPERT SYSTEM (FRAMES)
XEROX1186
DATABASE
XEROX 1186
COMMUNICATION & ALGORTHIMIC CONTROLLER
VME 10
ETHERNET
RS 232
RS 422
POWER DATA BUS
POWER RING BUS B
PDCU
SUBSYSTEM DISTRIBUTOR
LOAD CENTER
LOAD CENTER
LOAD CENTER
LOAD CENTER
SUBSYSTEM DISTRIBUTOR
A
B
RS 422
LOAD CENTER DATA BUS
SENSOR RMS VOLTAGE RMS CURRENT POWER FACTOR
REMOTE POWER CONTROLLER 1 OR 3 KW
REMOTE CONTROLLED CIRCUIT BREAKER 10 KW
REMOTE BUS ISOLATOR 15 KW
LLP - LOWEST LEVEL PROCESSOR
SIC - SWITCHGEAR I/F CONTROLLER
ADC - ANALOG TO DIGITAL CARD

MARTIN MARIETTA

FIGURE 1.

system. In the case of EMI characterization, the breadboard system cables and their couplings to nearby metallic bodies are not typical of the flight system, since the flight cables and installation guidelines have not been specified. The power source used with the breadboard and the impedance between it and the SSM/PMAD are not intended to be representative of the flight system since the flight equipment is not yet designed. Hence, regulation studies, transient response measurements, and fault reaction and coordination studies may be performed, but the results may not be valid if extrapolated to Space Station. They will contribute to evaluation of the switchgear concepts. Such studies should be done to characterize the breadboard.

B. Concept Evaluation Facility

The SSM/PMAD functionality provides a basis of the broad functional capability needed by Space Station. Power is delivered via a dual ring bus controlled by remote bus isolators (RBIs). Remote controlled circuit breakers (RCCBs) control the inputs to redundant 10kW buses, which each supply three 3kW remote power controllers (RPCs), housed in a power distribution and control unit (PDCU). Load centers, each representing one rack or double rack, contain two 3kW buses each, traceable to different PDCUs on different channels. Each bus supplies nine 1kW RPCs. Loads requiring redundant power supply are connected to two RPCs of different channels, and loads requiring single supply are connected to one RPC. Two load centers out of six are designated subsystem distribution assemblies (SDAs), and are equipped with 3kW RPCs instead of 1kW RPCs.

By applying loads of differing type, size, and profile to the load centers and SDAs and coordinating their schedules the system engineer may simulate a wide variety of situations and observe interactions in real time, with real loads, and with real power. The breadboard includes instrumentation, remote control, and fault protection hardware that allow both monitoring of system behavior and observation of the monitoring and control operations themselves. Application concepts in many areas may be set up and evaluated. Suitable applications include:

a. distribution system management and control
b. switchgear hardware usage concepts
c. protective device operation
d. protection system coordination
e. automated fault management
f. automated redundancy management
g. autonomous operation
h. person-machine interactions
i. autonomous schedule implementation
j. inter-subsystem behavior
k. power interruption and reconnection
l. load converter evaluation.

It is anticipated that such studies will generate much useful information, and that the best ways to accomplish critical on-orbit tasks will be elucidated.

C. Expert System Test Bed

The availability of crew time is one of the most restrictive factors in activity planning, both on-orbit and on the ground. This provides a strong motivation to reallocate functions from the crew to the lowest subsystem levels possible, and to create expert systems to implement as many functions as possible. Many ground-based and orbital functions are manpower-intensive and amenable to performance by expert systems. The SSM/PMAD provides a test bed equipped with sensors and effectors that can be used to host expert systems designed to perform these types of functions.

D. Function Simulator and Real-Time Software Test Bed

The SSM/PMAD is equipped with sensors, data acquisition equipment and software, and command execution software to provide traditional automation and remote control capabilities. These permit operation and observation from a single computer terminal, which provides both a manual-mode human interface and an interface to higher level automation computers or autonomous controllers. The switchgear and sensors included are fully compatible with the data acquisition and command execution software, providing a hardware test bed for any real-time automation software that may be required for future development. The switchgear control, protection, and status reporting characteristics conform to current Space Station requirements (JSC 30263) and enable the breadboard to provide a functional simulation of a Space Station module power management and distribution system to which appropriate software may be interfaced.

E. System Autonomy Demonstration Program Test Bed

It is currently planned to utilize the SSM/PMAD breadboard in the System Autonomy Demonstration Program 1990 demonstration. It will be utilized in the Power System Autonomy Demonstration to show interaction between expert controllers at the overall Space Station power subsystem level and at the module distribution and management level. It may also be used in the Power-Thermal Autonomy Demonstration to support interactions between the overall power and thermal subsystem controllers. These activities will exercise all of the functions included in the SSM/PMAD.

F. Development Toward Hardware Test Bed Usage

The SSM/PMAD can be upgraded with flight or prototype flight hardware as it becomes available. It can be developed into a ground simulation that will be useful in flight system evaluation. Not only can the hardware be updated, but the software can be refined and modified due to the use of industry-standard languages, interfaces, and systems. Therefore, any changes in functional requirements can be implemented within the constraints of available compatible hardware. The SSM/PMAD will be a useful tool for Space Station power system development for many years to come due to its flexibility and standardization.

IV. Exercising the SSM/PMAD

A. Producing Fault Conditions in Hardware

The fault conditions and operations discussed below are chosen with reference to the set of faults that FRAMES is designed to diagnose. A selection of these and the symptom sets by which they will be detected is displayed in Figure 2.

1.0 Load Operations and Faults in a 1kW Load Circuit

1.1 Load Simulation and Load Bank Concept

Test loads are based on a resistive component of load that dissippates a percentage of the circuit capability. Table 1 details the resistor values, performance to be expected with actual resistor values, and maximum inductive components for several levels of load simulation, including fault injection. Assumptions incorporated include the following:

a. The distribution system will be monitored by measurement channels with a 1.5% accuracy capability. Therefore, the loads must be known to 1 part in 1000, with the error budget divided between resistors and reactive elements. Power resistors will have to be known to at least 0.1% at operating temperature. This does not affect the way resistances are specified, but does affect the way they are measured.

b. The inductive component of the load impedance must be less than this allowed error. The corner frequency of the series L-R network must therefore be greater than 200kHz. This drives the requirement relating to purchase of non-inductive resistors.

c. The voltage is always assumed to be 208 volts rms.

d. All currents are rms values, and all power values are average rather than peak.

It is also necessary to vary the load power factor over a range encompassing faulty leading power factor, the permissible operational range, and faulty lagging power factor. It is recommended that the ability to simulate power factors up to and including 0.7 leading and lagging be incorporated. Leading power factors may be simulated by adding series inductance to the resistors specified above, and lagging power factors may be

SAMPLE FAULT AND SYMPTOM LIST

ITEM	ELEMENT TYPE AND LOCATION	HARDWARE FAULTS	SYMPTOM SET
1	UPPER SWITCH (RCCB)	Contactor failed open	1kW and 3kW RPCs trip due to undervoltage, sensors below switch read low voltage.
2		Contactor failed closed	No symptoms if closed, but switch will not close when commanded.
3		Contacts resistive	Sensors below show low voltage if switch is closed, no symptoms otherwise.
4		Overload sensor failed	No immediate symptoms, could contribute to a masked fault.
5	LOAD CENTER INPUT CABLE	Short circuit to structure	3kW RPCs trip due to ground fault, 1kW RPCs trip due to undervoltage.
6		Open circuit	3kW RPC current reads zero, all 1kW RPCs below it trip due to undervoltage.
7		Series resistive	Load Center sensor reads low voltage, may not be recognized as a fault.
8		Shunt resistive	Sensors above cable read excessive current.
9	LOWEST SWITCH (1kW RPC)	Power stage failure - miscellaneous	No immediate symptoms, could contribute to a masked fault.
10		Input short to return	3kW above fast trips, other1kW RPCs on the same bus trip on undervoltage.
11		Output short to return	1kW fast trips, sensors above it read reduced current.
12		Main switch failed open	RPC current reads low if on, no symptoms otherwise, load underpowered.
13		Main switch transistors fail to saturate	RPC trips due to overtemperature.
14		Isolation relay fails open	RPC current reads zero, may not be recognized as a fault, no symptoms if open.
15		Isolation relay contacts resistive	RPC current is too low, may not be recognized as a fault, no symptoms if open.
16		Overload detector fails	No immediate symptoms, could contribute to a masked fault.
17		Short circuit detector fails	No immediate symptoms, could contribute to a masked fault.
18		Overheat sensor fails	No immediate symptoms, could contribute to a masked fault.
19		Undervoltage sensor fails	No immediate symptoms, could contribute to a masked fault.
20		Ground fault sensor fails	No immediate symptoms, could contribute to a masked fault.
21	LOAD CIRCUIT	Short circuit to structure	RPC trips due to ground fault.
22		Open circuit	RPC current is too low, may be recognized as a symptom if switch is closed.
23		Series resistive	RPC current is too low, may be recognized as a symptom if switch is closed.
24		Shunt resistive	RPC current is higher than scheduled when load draws its scheduled maximum.
25		Load overcurrent	RPC trips on overcurrent.
26		Load short to return	RPC trips on fast trip.

FIGURE 2.

TABLE 1 LOAD RESISTANCE SPECIFICATIONS

PARAMETERS NOMINAL LOAD LEVELS (LLN)	NOMINAL LOAD RESISTANCE (Rnom)	ACTUAL LOAD RESISTANCE (Ract)	ACTUAL POWER DISSIPPATION (Pact)	ACTUAL CURRENT (Iact)	ACTUAL LOAD LEVEL (LLA)	MAXIMUM PARASITIC INDUCTANCE (Lact)
25%	173Ω	175Ω	247W	1.19A	24.70%	139μH
50%	86.5Ω	90Ω	481W	2.31A	48.10%	71.7μH
100%	43.3Ω	45Ω Parallel 2-90Ω	961W	4.62A	96.10%	35.8μH
125%	34.6Ω	35.8Ω Parallel 2-90Ω and 1-175Ω	1210W	5.81A	121.00%	28.7μH
200%	21.6Ω	22.5Ω Parallel 4-90Ω	1920W	9.24A	192.00%	17.9μH

TABLE 2 SERIES INDUCTIVE LOAD CIRCUIT

NOMINAL LOAD LEVEL POWER FACTOR	25% Ract = 175Ω	50% Ract = 90Ω	100% Ract= 45Ω	125% Ract= 35.8Ω	200% Ract= 22.5Ω
1.00	L= 0 /Z/= 175Ω I= 1.19A P= 247W VA= 247VA	L= 0 /Z/= 90Ω I= 2.31A P= 480W VA = 480VA	L= 0 /Z/ = 45Ω I= 4.62A P= 960W VA= 960VA	L= 0 /Z/= 35.8Ω I = 5.81A P= 1210W VA=1210VA	L = 0 /Z/= 22.5Ω I= 9.24A P= 1920W VA=1920VA
0.95	L= 458μH /Z/= 184Ω I= 1.13A P= 223W VA= 235VA	L= 235μH /Z/= 94.7Ω I= 4.39A P= 867W VA= 913VA	L= 118μH /Z/= 47.4Ω I= 4.39A P= 867W VA= 913VA	L= 94.2μH /Z/= 37.7Ω I= 5.52A P= 1090W VA=1150VA	L= 58.9μH /Z/= 23.7Ω I= 8.78A P= 1730W VA=1830VA
0.9	L= 673μH /Z/= 194Ω I= 1.07A P= 200W VA= 223VA	L= 346μH /Z/= 100Ω I= 2.08A P= 389W VA= 433VA	L= 173μH /Z/= 50.0Ω I= 4.16A P= 779W VA= 865VA	L= 138μH /Z/= 39.8Ω I= 5.23A P= 979W VA=1090VA	L= 86.5μH /Z/= 25.0Ω I= 8.32A P= 1560W VA=1730VA
0.8	L= 1.04μH /Z/= 219Ω I= .950A P= 158W VA= 198VA	L= 536μH /Z/= 113Ω I= 1.84A P= 305W VA= 383VA	L= 268μH /Z/= 56.3Ω I= 3.69A P= 613W VA= 769VA	L= 213μH /Z/= 44.7Ω I= 4.65A P= 774W VA= 967VA	L= 134μH /Z/= 28.1Ω I= 7.40A P= 1230W VA=1540VA
0.7	L= 1.42μH /Z/= 250Ω I= .832A P= 121W VA= 173VA	L= 729μH /Z/= 129Ω I= 1.61A P= 233W VA= 335VA	L= 364μH /Z/= 64.3Ω I= 3.23A P= 469W VA= 672VA	L= 291μH /Z/= 51.2Ω I= 4.06A P= 590W VA= 844VA	L= 182μH /Z/= 32.1Ω I= 6.48A P= 945W VA=1350VA

TABLE 3 PARALLEL CAPACITIVE LOAD CIRCUIT

NOMINAL LOAD LEVEL POWER FACTOR	25% Ract= 175Ω	50% Ract= 90Ω	100% Ract= 45Ω	125% Ract= 35.8Ω	200% Ract= 22.5Ω
1.00	C= 0 /Z/= 175Ω I= 1.19A P= 247W VA= 247VA	C= 0 /Z/= 90Ω I= 2.31A P= 481W VA= 481VA	C= 0 /Z/= 45Ω I= 4.62A P= 961W VA= 961VA	C= 0 /Z/= 35.8Ω I= 5.81A P= 1210W VA=1210VA	C= 0 /Z/= 22.5Ω I= 9.24A P= 1920W VA=1920VA
0.95	C= 14.9µF /Z/= 166Ω I= 1.25A P= 247W VA= 247VA	C= 29.0µF /Z/= 85.5Ω I= 2.43A P= 481W VA= 505VA	C= 29.0µF /Z/= 42.7Ω I= 4.87A P= 961W VA=1010VA	C= 72.9µF /Z/= 34.0Ω I= 6.12A P= 1210W VA=1270VA	C= 116µF /Z/= 21.4Ω I= 9.73A P= 1920W VA=2020VA
0.9	C= 22.0µF /Z/= 157Ω I= 1.32A P= 247W VA= 275VA	C= 42.7µF /Z/= 81.0Ω I= 2.57A P= 481W VA= 535VA	C= 85.4µF /Z/= 40.5Ω I= 5.14A P= 961W VA=1070VA	C= 107µF /Z/= 32.2Ω I= 6.46A P= 1210W VA=1340VA	C= 171µF /Z/= 20.2Ω I= 10.3A P= 1920W VA=2140VA
0.8	C= 34.0µF /Z/= 140Ω I= 1.49A P= 247W VA= 310VA	C= 66.1µF /Z/= 72.0Ω I= 2.89A P= 481W VA= 601VA	C= 132µF /Z/= 36.0Ω I= 5.78A P= 961W VA=1200VA	C= 166µF /Z/= 28.7Ω I= 7.25A P= 1210W VA=1510VA	C= 265µF /Z/= 18.0Ω I= 11.6A P= 1920W VA=2410VA
0.7	C= 46.3µF /Z/= 122Ω I= 1.70A P= 247W VA= 354VA	C= 90.0µF /Z/= 63.0Ω I= 3.30A P= 481W VA= 686VA	C= 180µF /Z/= 31.5Ω I= 6.60A P= 961W VA=1370VA	C= 226µF /Z/= 25.1Ω I= 8.29A P= 1210W VA=1720VA	C= 360µF /Z/= 15.7Ω I= 13.2A P= 1920W VA=2750VA

TABLE 4 THEORY AND FORMULAE

TABLE 1.

Ⓐ $$\text{POWER} = \frac{(\text{VOLTAGE})^2}{\text{RESISTANCE}}$$

(X % OF FULL LOAD) = LLN

FULL LOAD = 1000W

$$\therefore \quad \text{POWER} = (1000\text{W}) \left(\frac{\text{LLN}}{100} \right) = \frac{(\text{VOLTAGE})^2}{\text{RESISTANCE}}$$

RESISTANCE = RNOM

$$\therefore \quad \boxed{\text{RNOM} = \frac{(\text{VOLTAGE})^2}{10 \times \text{LLN}}}$$

Ⓑ ACTUAL LOAD RESISTANCES (RACT) ARE MADE UP OF 175 Ω AND 90Ω RESISTORS IN COMBINATION.

Ⓒ $$\text{PACT} = \frac{(\text{VOLTAGE})^2}{\text{RACT}}$$

Ⓓ $$\text{IACT} = \frac{\text{VOLTAGE}}{\text{RACT}}$$

Ⓔ LLA= ACTUAL % OF FULL LOAD

$$\boxed{\text{LLA} = \frac{(\text{VOLTAGE})^2}{\text{RACT}} + 1000 \text{ WATT} \times 100\%}$$

Ⓕ LACT= MAXIMUM PERMITTED PARASITIC INDUCTANCE IN THE LOAD

$$\boxed{\text{LACT} = \frac{\text{RACT}}{2\Pi \times 200{,}000 \text{ HERTZ}}}$$

TABLE 4. (continued)

TABLE 2.

Ⓐ THE POWER FACTOR (PF) OF A LOAD IS DEFINED AS FOLLOWS:

$$\text{PF} = \text{COS [ARCTAN} \left(\frac{\text{VARS}}{\text{WATTS}}\right)] = \frac{\text{WATTS}}{[(\text{WATTS})^2 + (\text{VARS})\ \ ]^{2\ 1/2}}$$

FOR A SERIES R-L LOAD.

$\text{VARS} = I\ \ X^2 L$

$\text{WATTS} = I\ \ R^2$

SUBSTITUTING AND SOLVING FOR L,

$$\boxed{L = \frac{\text{RACT}}{2\Pi\ f} \left(\frac{1}{\text{PF}^2} - 1\right)^{1/2}}$$

Ⓑ $/Z/ = [(2\Pi\ fL)^2 + R^2\ \ ]^{1/2}$

Ⓒ $I = \frac{\text{VOLTAGE}}{/Z/}$

Ⓓ $\text{PACT} = I^{\ 2} \times \text{RACT}$

Ⓔ APPARENT POWER = VOLTAGE x I

TABLE 4. (continued)

TABLE 3.

Ⓐ THE POWER FACTOR IS DEFINED AS IN TABLE 2. FOR A PARALLEL R-C LOAD.

$$\text{VARS} = \frac{(\text{VOLTAGE})^2}{X_C}$$

$$\text{POWER} = \frac{(\text{VOLTAGE})^2}{\text{RACT}}$$

USING THESE RELATIONS AND THE DEFINITION OF POWER FACTOR,

$$C = \frac{1}{2\Pi f\ (\text{RACT})} \left(\frac{1}{\text{PF2}} - 1\right)^{1/2}$$

Ⓑ $$/Z/ = \frac{1}{\left[\frac{1}{(\text{RACT})^2} + (2\Pi f C)^2\right]^{1/2}}$$

Ⓒ $$I = \frac{\text{VOLTAGE}}{/Z/}$$

Ⓓ $$\text{PACT} = \frac{(\text{VOLTAGE})^2}{\text{RACT}}$$

Ⓔ APPARENT POWER = VOLTAGE x I

simulated by adding shunt capacitance to them after removing the inductors. Table 2 identifies the value of inductance required to produce these power factors at each load level and the load circuit performance to be expected. Table 3 provides values of capacitance required and circuit performance to be expected. Theory and formulae used in these calculations are documented in Table 4. Operation with capacitive loads should only be attempted if each RPC has a 5 microhenry inductor installed in series with its "hot" 20kHz terminal.

Loads of varying time profile in either level or power factor may be simulated using an electronic load simulator or the load bank described above. This requires that the load bank be controlled by relays that may be switched hot at the expected stress levels. A load profile of essentially arbitrary shape may be programmed with either of these devices.

1.2 Load Circuit Faults

Load circuits below the 1kW RPC level are susceptible to several simple hardware faults that can be modelled using relays and impedances, as shown in Figures 3 and 4. A similar set of fault injection relays installed below the 10kW RCCB and 3kW RPCs, as also shown in Figure 4, will be useful as well. The relays shown may be used to inject the following types of faults:

a. open circuit - open the series relay while operating a load.
b. supply line short to return - install a shorting bar in series with the shunt relay, then close the relay while operating a load.
c. supply short to structure - close the relay to ground while operating a load.
d. series resistance in cable - transfer the series DPDT relay to the resistive position while operating a load.
e. shunt resistance in cable - install a large resistor (large enough to produce a measurable increase in current, but not so large as to produce an overload) in series with the shunt relay, then close the relay while operating a 25% or 50% load.

2.0 1 kW Remote Power Controller (RPC)

The Remote Power Controllers (RPC) are required to enable and disable power to the load circuits, isolate failed loads from the Load Center (LC) bus, control switchover to redundant supply, measure load current, and interrupt the load current in the event of one of several types of fault. When loads do not contain internal on - off control the RPCs will be called upon to control them. RPCs are therefore required to start, supply, and stop power to loads over a range of power levels and power factors. In the course of starting loads inrush currents may reach damaging levels, and the RPC must protect the LC bus from voltage sag or possible damage. Topology and functions are shown in Figures 5a and 5b.

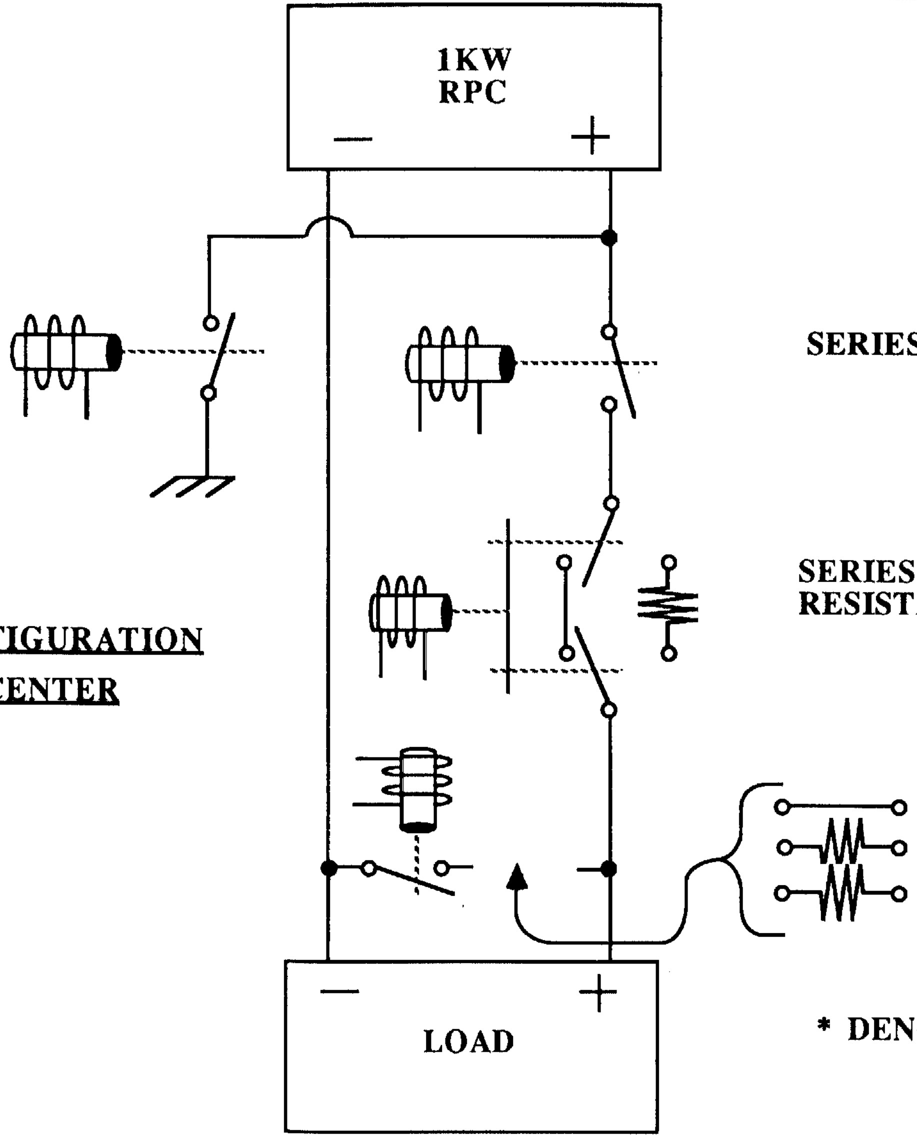
FAULT RELAY INSTALLATION GUIDELINES

- FAULT RELAY INSTALLATION ONLY REQUIRED IN TWO LOAD CIRCUITS ON TWO LOAD CENTERS, AT LOAD CIRCUIT LEVEL.
- BUS AND CABLING BETWEEN 1kw RPCs AND 3kw RPCs CAN BE HANDLED IN ONE SECTION OR TWO, DIVIDED BY SENSOR #6 (S6).
- TO CONSERVE EARLY-PHASE INTEGRATION EFFORT AND HARDWARE, RECOMMEND CONSIDERING THAT CABLE TO BE ONE SEGMENT.
- SINCE ADJACENT 3kw CIRCUITS (PDCU OUTPUTS) ARE INDEPENDENT IN 2-FAULT SCENARIOS, IT IS ONLY NECESSARY TO EQUIP THE ONE 3kw OUTPUT CIRCUIT THAT FEEDS THE FAULTABLE 1kw CIRCUITS.
- BOTH FAULTABLE 3kw CIRCUITS SHOULD BE ON A FAULTABLE 10kw CIRCUIT.
- IMPEDANCE OF FAULTABLE CIRCUITS SHOULD BE ANALYZED TO DETERMINE THE EFFECT OF ADDING FAULT INJECTION RELAYS. FAULTABLE CIRCUITS MAY HAVE TO BE COMPENSATED TO RESTORE IDENTICAL PERFORMANCE.

FIGURE 3

MARTIN MARIETTA

1 KW FAULT RELAY LAYOUT



CABLE SHORT
TO
STRUCTURE
*

SERIES OPEN

SERIES
RESISTANCE - CABLE OR CONTACT
DEGRADED

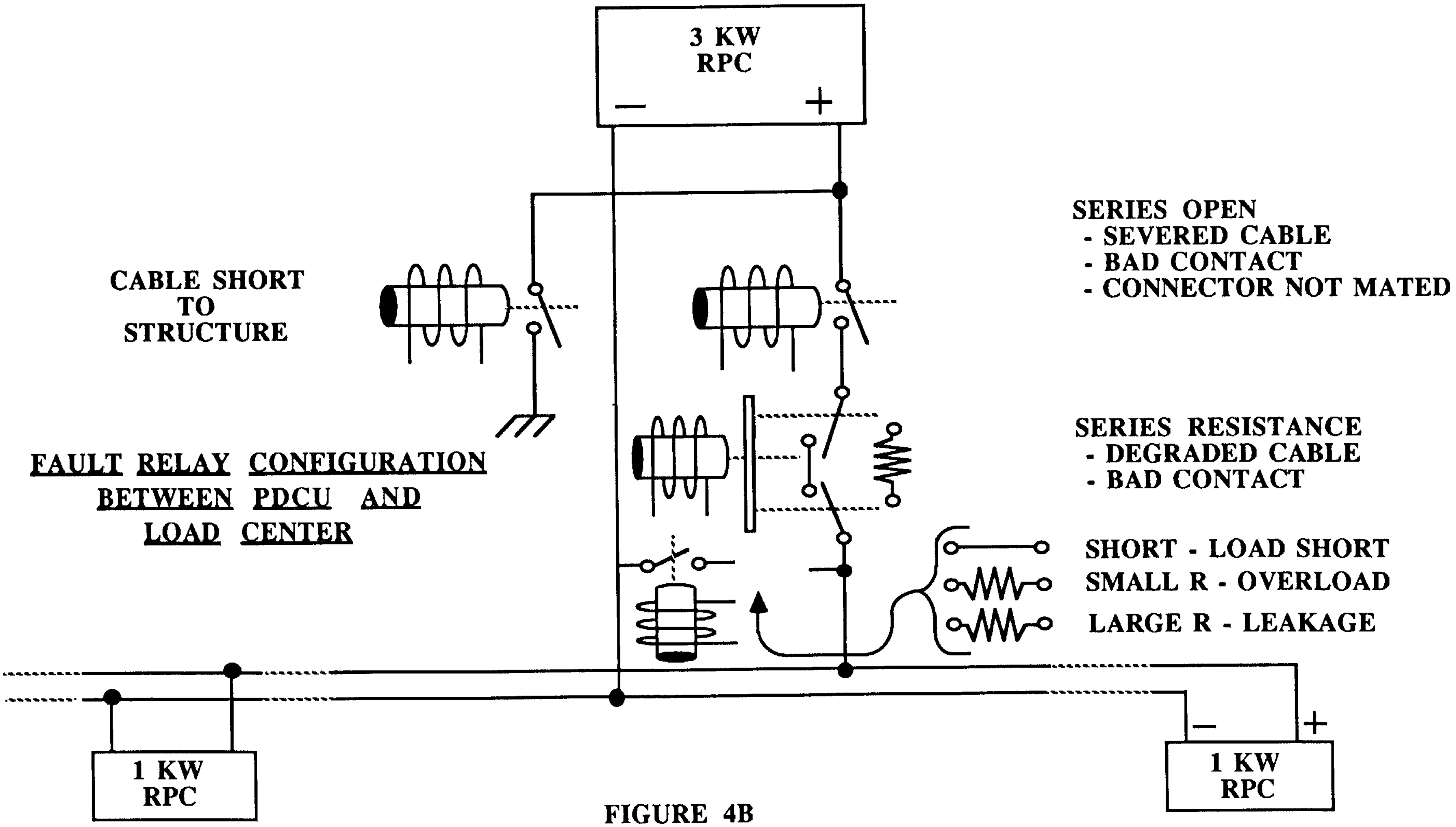
SHORT - LOAD SHORT *
SMALL R - OVERLOAD *
LARGE R - LEAKAGE

FAULT RELAY CONFIGURATION BETWEEN LOAD CENTER AND LOAD

* DENOTES SWITCHGEAR TEST

FIGURE 4A

3KW FAULT RELAY LAYOUT



CABLE SHORT
TO
STRUCTURE

FAULT RELAY CONFIGURATION BETWEEN PDCU AND LOAD CENTER

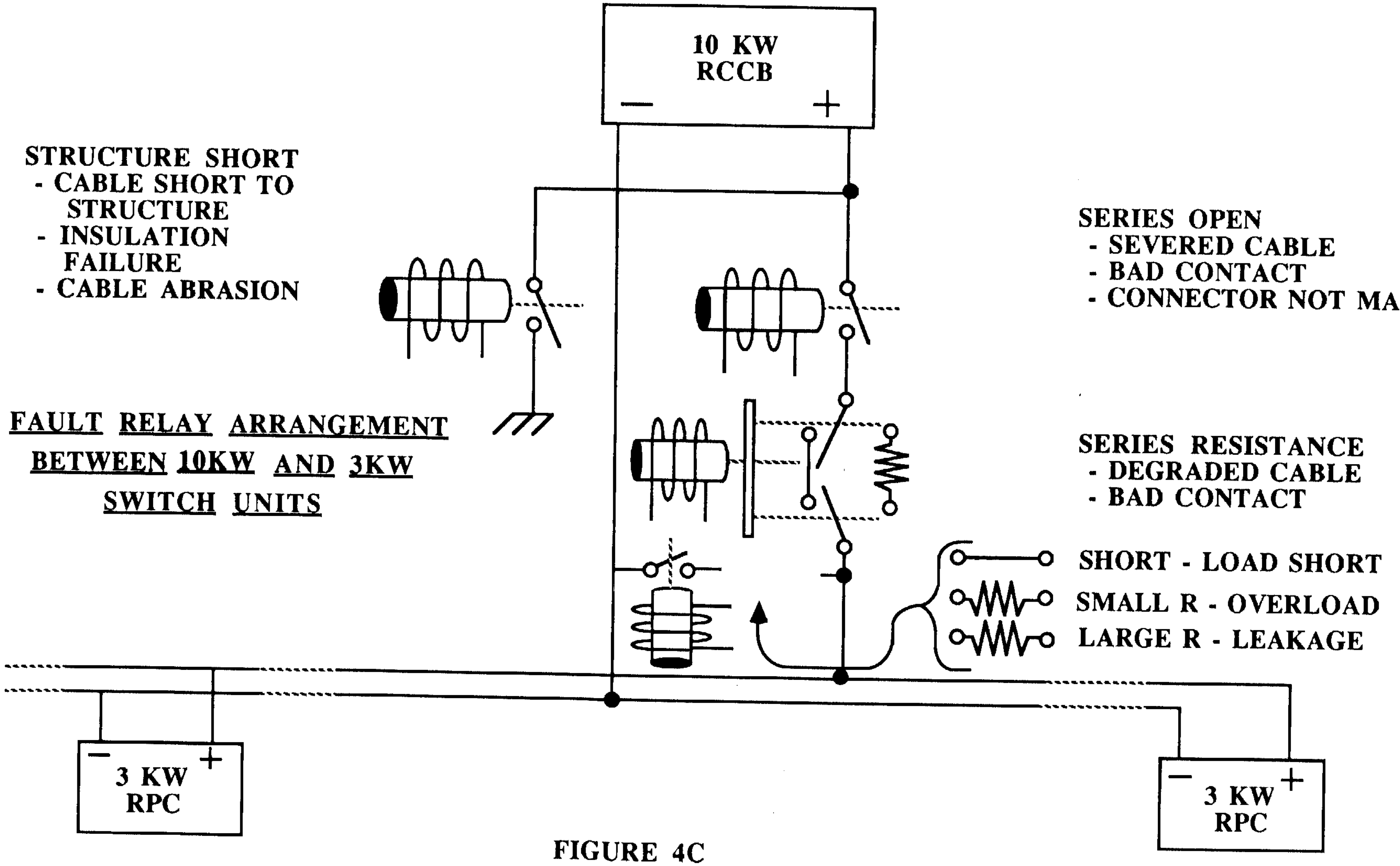
SERIES OPEN
- SEVERED CABLE
- BAD CONTACT
- CONNECTOR NOT MATED

SERIES RESISTANCE
- DEGRADED CABLE
- BAD CONTACT

SHORT - LOAD SHORT
SMALL R - OVERLOAD
LARGE R - LEAKAGE

FIGURE 4B

10 KW FAULT RELAY LAYOUT



STRUCTURE SHORT
- CABLE SHORT TO STRUCTURE
- INSULATION FAILURE
- CABLE ABRASION

<u>FAULT RELAY ARRANGEMENT BETWEEN 10KW AND 3KW SWITCH UNITS</u>

SERIES OPEN
- SEVERED CABLE
- BAD CONTACT
- CONNECTOR NOT MATED

SERIES RESISTANCE
- DEGRADED CABLE
- BAD CONTACT

SHORT - LOAD SHORT
SMALL R - OVERLOAD
LARGE R - LEAKAGE

FIGURE 4C

MARTIN MARIETTA

2.1 RPC Operation

Simulating load operations requires use of load impedance elements. Load impedances may be provided by a load bank as specified in Tables 1, 2, and 3, individual load impedances, or an electronic load simulator. The load bank and the load simulator will afford much more convenient operation over the full ranges of load level and load power factor than will individual load impedances.

2.1a Starting loads

The load starting capability may be exercised by connecting the desired load to the output of an open RPC, and closing the RPC. Once a load is started, it may be run, shutdown, or faulted to further exercise the system. With a load bank the load may be varied once it is started to apply a load profile to the control system.

2.1b Supplying loads

If no excessive inrush current trips the RPC on startup, the load is being supplied if the procedure of Paragraph 2.1a above has been completed.

2.1c Stopping loads

If the load connected has been started and can be run, it can be stopped by opening the RPC. If a load bank or an electronic load simulator is used, the load level or power factor can be changed and the start-supply-stop sequence repeated.

2.2 Forcing the RPC to trip

A 1kW RPC may be made to trip by injecting an actual fault condition into the breadboard or by hardwiring certain control signals on the Generic Controller Card (GC). This discussion only deals with injected faults. The fault injection relays may be used with a load bank, a special variable load, or an electronic load simulator to produce overstresses in a controlled fashion. The goal is to produce overstress conditions to which an RPC is designed without reaching destructive levels.

The following connections and operations will inject the specified hardware faults into the 1kW RPC load circuit:

a. trip on overload - increase the load level to greater than 120% at any power factor, or at a 100% level reduce the power factor to 0.7 or less lagging. The load level changes could be applied in a load bank or in the shunt fault injection relay, after a properly low resistance (low enough to represent an overload but not so low that a fast trip is created) is installed in series with it.

REMOTE POWER CONTROLLER FUNCTIONS

PROTECTION

- FAST (SURGE) TRIP
- I^2T (OVERLOAD) TRIP
- OVERTEMPERATURE TRIP (SELF-PROTECTION)
- UNDERVOLTAGE TRIP
- GROUND FAULT TRIP
- LIMITS INRUSH CURRENTS
- TRIPS ON EXCESSIVE INRUSH

SWITCHING

- SOLID STATE SWITCHING
- TURNS ON AT ZERO VOLTAGE
- TURNS OFF AT ZERO CURRENT
- CONTAINS ALL SENSORS NECESSARY
- RELAY ISOLATES LOAD WHEN OFF

SENSING - INTERNAL UNLESS NOTE

- VOLTAGE SENSOR
- CURRENT SENSOR - CURRENT VALUE REPORTED TO CONTROLLER
- TEMPERATURE SENSOR
- ZERO-CROSSING DETECTOR
- ZERO-CROSSING DETECTOR CAN BE TUNED FOR DIFFERENT POWER FACTORS

LOW DISSIPPATION - 1% LOSS

SUPPORTS HARDWARE-LEVEL REDUNDANCY CONTROL

FIGURE 5A

MARTIN MARIETTA

REMOTE POWER CONTROLLER BASIC SCHEMATIC

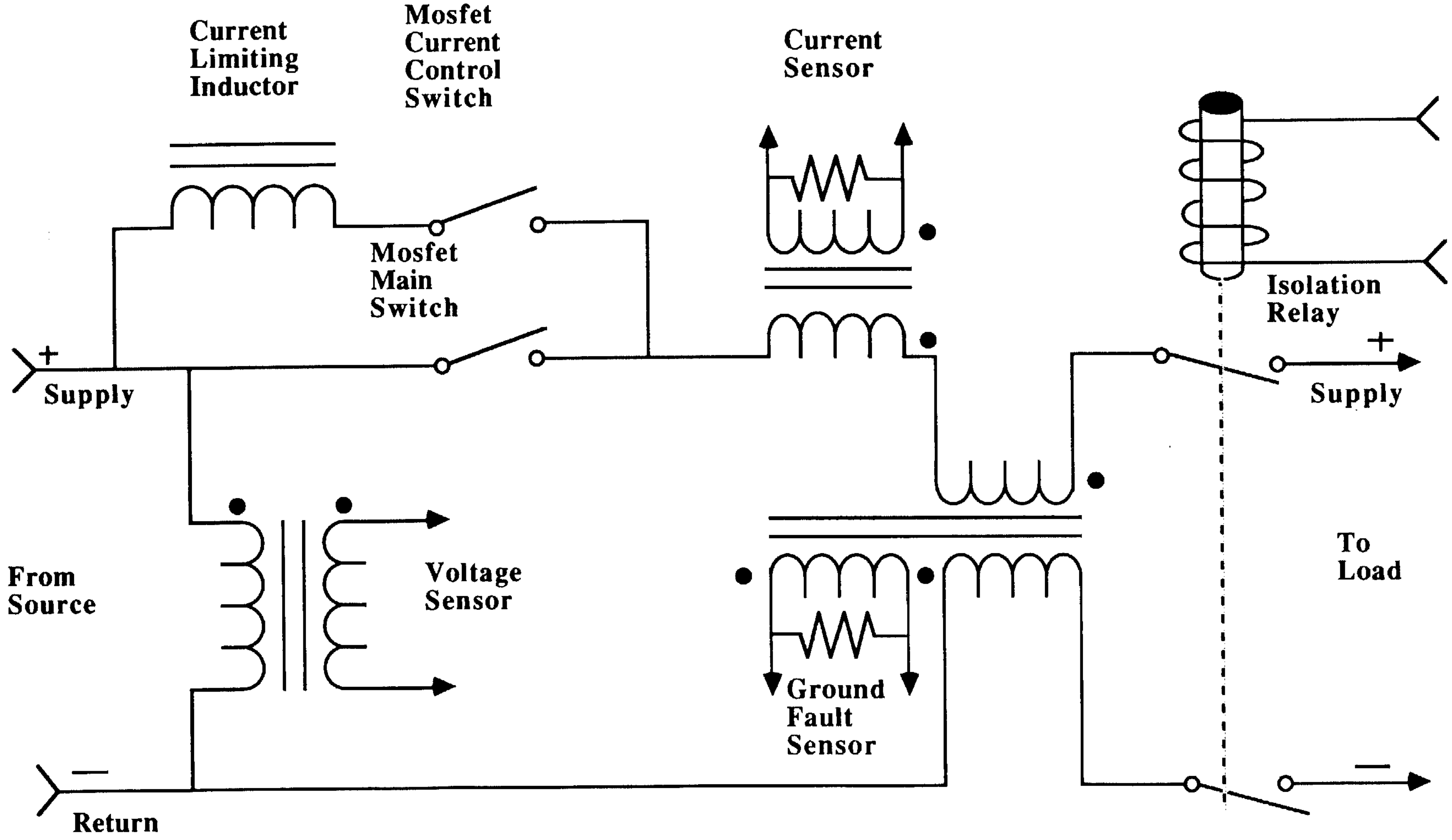


FIGURE 5B

MARTIN MARIETTA

b. trip on fast trip - install a shorting bar or a resistor that will draw 400% or more of the rated load in series with the shunt fault injection relay. With the shunt relay open and a load being supplied, close the shunt relay. The RPC will trip via its fast trip mechanism.

c. trip on ground fault - install a large resistor (large enough to draw 50 milliamps) in series with the short to ground relay. With the RPC supplying a load, close the relay.

d. trip on undervoltage - while the RPC is supplying a load, either open the 3kW RPC supplying it or the series fault injection relay above it (interrupt the current supply) or short the 3kW circuit above it using a shorting bar and the shunting fault injection relay.

f. trip on inrush - load an RPC with a 43.3 ohm resistor in parallel with a 2 microfarad capacitor, and turn on the RPC.

g. masked overload failure - referring to the Generic Controller schematic (849NWT12001, Rev. A, page 3), short U20A pin 2 to U20A pin 12, reduce the load resistance to between 35.8 ohms and 22.5 ohms for a 1 kW RPC (see Table 1), and close the RPC. All manipulations of a Generic Controller card must be done under strict observance of Electrostatic Discharge guidelines and safeguards, and by personnel who are fully trained and certified in those procedures. Failure to observe this caution may result in destruction of some or all of the semiconductor devices installed on the GC.

h. isolator failed open - referring to the Generic Controller schematic (849NWT12001, Rev. A, page 4), ground the drain of Q2.

Then command the RPC to close. These manipulations of the GC card must be performed with excellent workmanship by a qualified engineer or technician to avoid damage to or destruction of the printed wiring board assembly.

i. main switch failed open - on the AC RPC modification card, connect the banded end of diode D102 to the positive end of capacitor C102, then command the RPC to close. The precautions mentioned in paragraphs g and h immediately above are required in this operation, as well.

j. input short to return - install a shorting bar in series with the shunt fault injection relay located above the 1kW RPC. After the system is operating normally, close the shunt fault injection relay.

k. output short to return - install a shorting bar in series with the shunt fault injection relay located below the 1kW RPC. After the system is operating normally, close this shunt fault injection relay.

3.0 Sensors

Sensor groups are placed at every input and output of a PDCU and at every branching point, or bus, in the system. Each sensor group consists of a voltage transformer, a current transformer and a burden resistor. Voltage and current waveforms are transmitted to an AD card. In addition to sampling and digitizing, the AD card contains a voltage-to-frequency converter, RMS converters for voltage and current, integrators and squaring circuits to calculate dc offsets and average power, and a phase shifter used to determine the direction of the power factor. Power factor magnitude is calculated by software in the LLP. All of this data is digitized and transmitted to the LLP via the SIC. Total failure of either a current sensor or a voltage sensor may be simulated by disconnecting the twisted pair wires to the sensor in question. Failures of calibration may be inserted by use of voltage dividers designed to produce any desired degree of error.

4.0 Cable

Cables serve to transport energy from one power system element to the next. The SSM/PMAD uses conventional twisted pairs as an initial configuration. If in the course of extended system operation this type of cable is found to be unsuitable, other configurations should be considered. Though cables are often thought to be not credible as sources of system failure modes, that may not be the case on Space Station or any other vehicle with a 30 year lifetime.

Cable failures may be injected into the breadboard as follows:

a. short circuit from power to structure - with the system operating, close the cable short to structure fault injection relay.

b. resistive path between cable conductors - connect a resistor in series with the shunt fault injection relay, operate the system. Close the shunt fault injection relay when it is desired to inject the fault.

c. series resistive cable or contact degradation - with the system operating, transfer the dual series relay from the non-resistive position to the resistive position.

5.0 3kW Remote Power Controller (RPC) - similar to 2.0

6.0 Cable - similar to 4.0

7.0 10 kW Remote Controlled Circuit Breaker (RCCB)

RCCBs are required to protect the ring bus from short circuits and heavy overloads that involve an entire radial distribution tree.

They are required to stop the largest fault currents of any protective devices and to isolate the fault from the ring bus until it can be resolved. Hence, they are only provided with an overload protection capability.

RCCB faults may be injected as follows:

a. failure of the overload trip capability - temporarily disable the overcurrent detection circuit on the Generic Controller card by the method described in paragraph 2.2g above. Observe all cautions mentioned in that paragraph.
b. contacts resistive - same procedure as in paragraph 4c above.

B. Breadboard Power Revision

The SSM/PMAD must manage its operation such that it does not consume more power than it is allowed to. A higher level resource manager or similar system will allocate a specific amount of power to the SSM/PMAD. The Load Enable Scheduler (LES) will then schedule as many tasks as it can within that constraint, among others. Many situations in the Space Station, such as damage to solar arrays by meteorites or reduction of solar dynamic capability due to scheduled servicing may require revision of the power allocation to a specific module. If a revised allocation is received by the LES, it will reschedule if required and implement that new schedule. The new schedule may require addition or removal of loads.

Power allocation changes are made at the Symbolics Interface (SI). They may be entered while a schedule is being executed. The breadboard response should be automatic.

C. Forcing a Redundant Load to Change to an Alternate RPC

Assume that a load requiring redundant power is installed, initially operating from channel A. The load is connected to channel B, but the channel B RPC is open, being inhibited by the fact that the channel A RPC is closed.

The next step is to disable channel A while it is turned on, which may be done in two ways: the "A" RPC may be forced to open on undervoltage by causing its input voltage to go to zero, or it may be shorted and forced to open on fast trip. In the first case the channel A RPC will open on undervoltage, releasing the inhibition on the corresponding channel B RPC. The channel B RPC will then turn on, resuming the operation of the load.

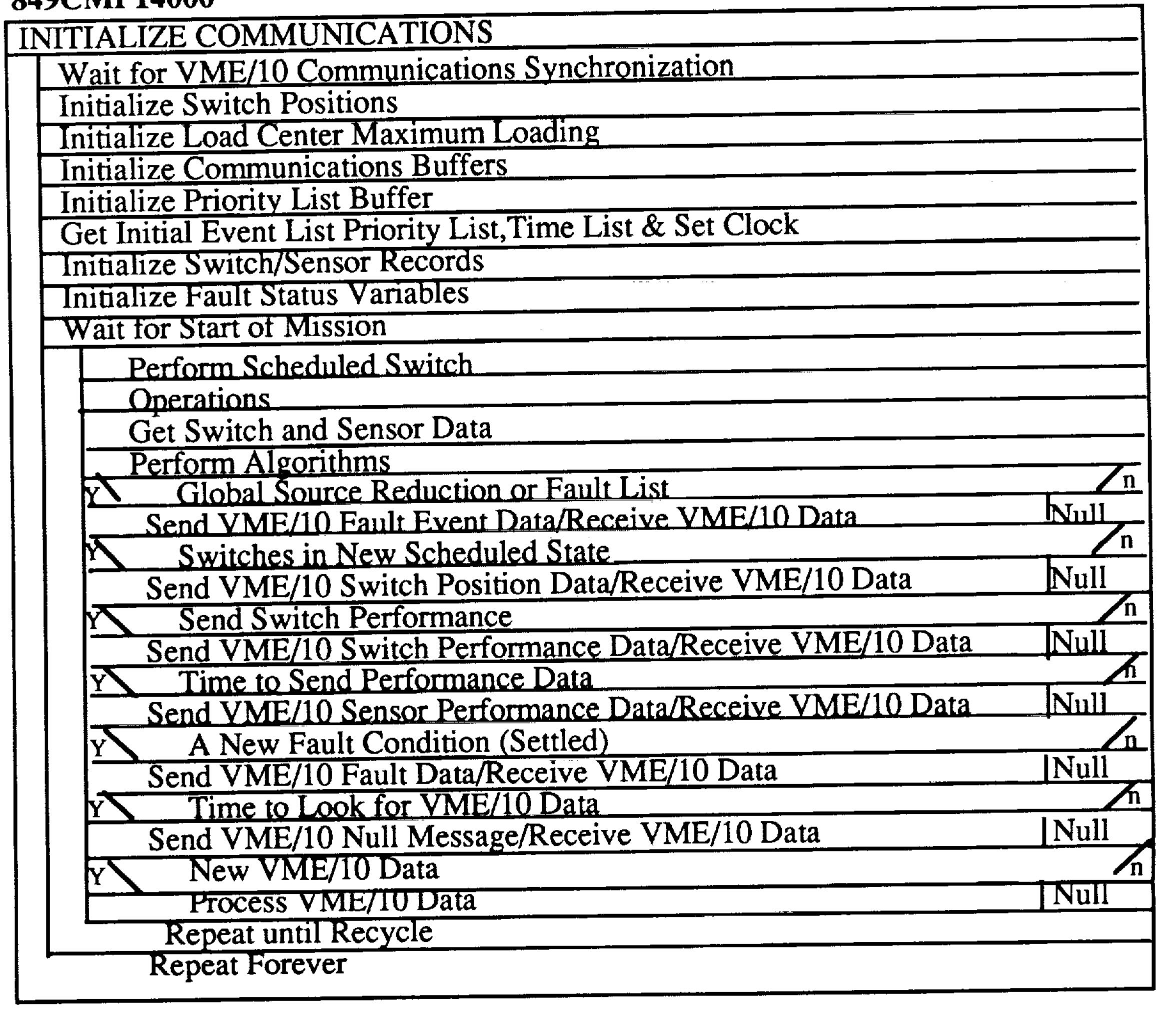
In the second case the load circuit is shorted from supply to return, and it should open on fast trip, reclose, open again, and stay open. After the channel A RPC has opened for the second time, the channel B RPC will enter a turn-on sequence. Since the applied short circuit is still present, a high inrush current will be detected through the channel B RPC, and it will trip open and stay open.

APPENDIX IX – VCLRs

18.0 APPENDIX IX: VISUAL CONTROL LOGIC REPRESENTATIONS

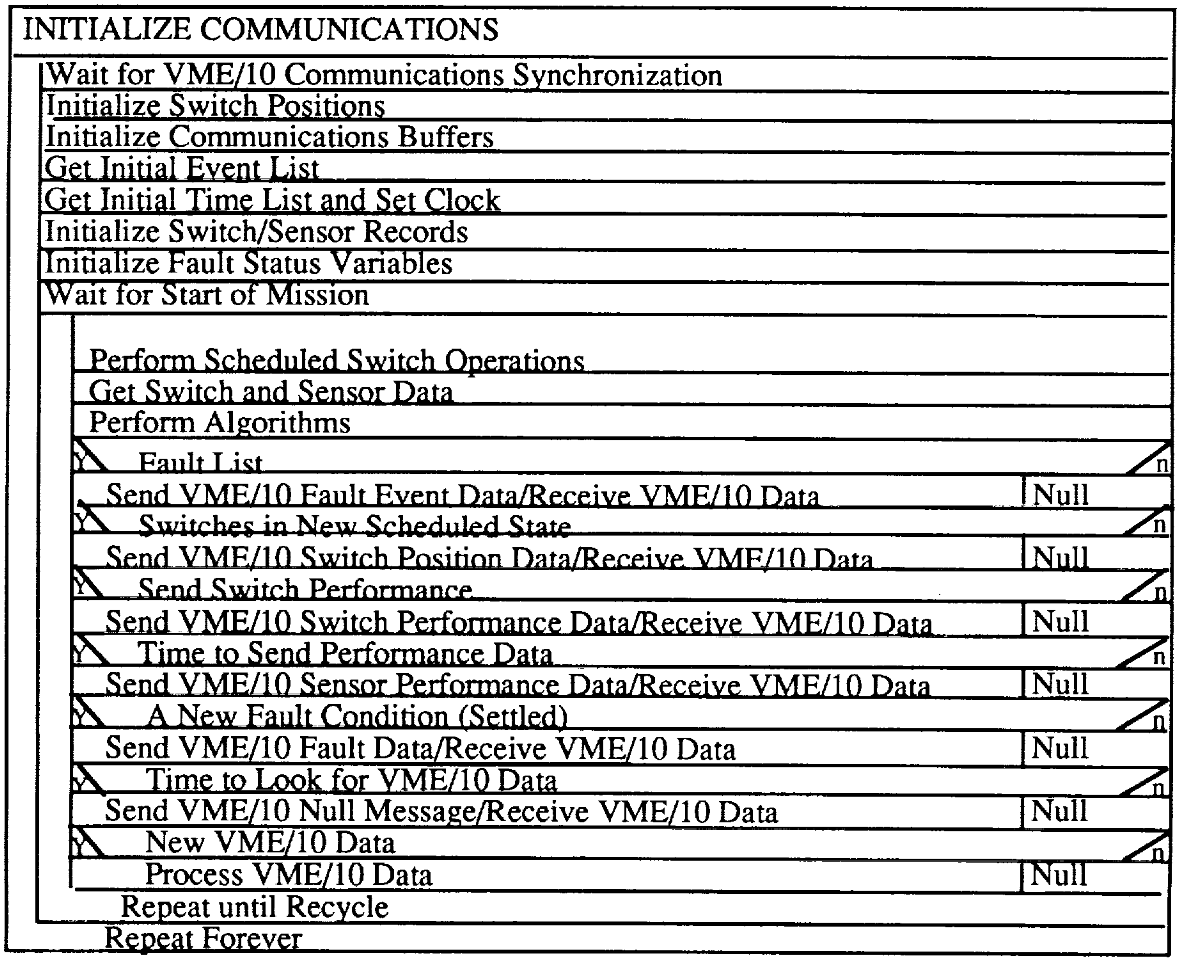
This appendix contains Visual Control Logic Representations (VCLRs) for the Lowest Level Processor (LLP) functions.

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14000
LC MAIN



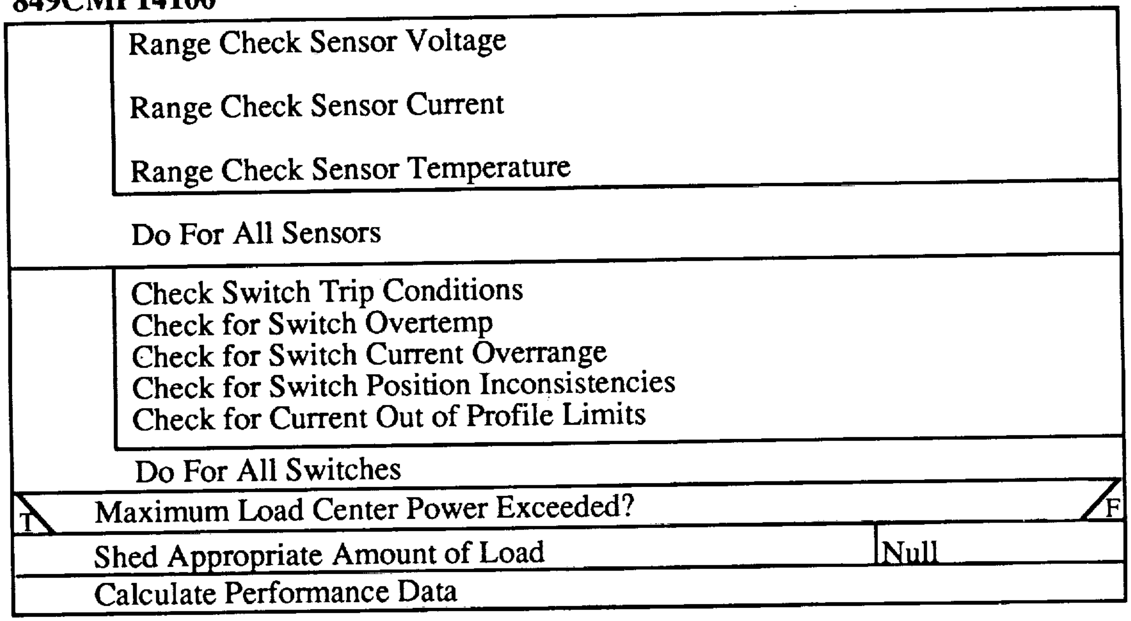
```
INITIALIZE COMMUNICATIONS
    Wait for VME/10 Communications Synchronization
    Initialize Switch Positions
    Initialize Load Center Maximum Loading
    Initialize Communications Buffers
    Initialize Priority List Buffer
    Get Initial Event List Priority List,Time List & Set Clock
    Initialize Switch/Sensor Records
    Initialize Fault Status Variables
    Wait for Start of Mission
        Perform Scheduled Switch
        Operations
        Get Switch and Sensor Data
        Perform Algorithms
        Y   Global Source Reduction or Fault List                         n
              Send VME/10 Fault Event Data/Receive VME/10 Data            Null
        Y   Switches in New Scheduled State                               n
              Send VME/10 Switch Position Data/Receive VME/10 Data        Null
        Y   Send Switch Performance                                       n
              Send VME/10 Switch Performance Data/Receive VME/10 Data     Null
        Y   Time to Send Performance Data                                 n
              Send VME/10 Sensor Performance Data/Receive VME/10 Data     Null
        Y   A New Fault Condition (Settled)                               n
              Send VME/10 Fault Data/Receive VME/10 Data                  Null
        Y   Time to Look for VME/10 Data                                  n
              Send VME/10 Null Message/Receive VME/10 Data                Null
        Y   New VME/10 Data                                               n
              Process VME/10 Data                                         Null
        Repeat until Recycle
    Repeat Forever
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14000P **PDCU MAIN**



```
INITIALIZE COMMUNICATIONS
  Wait for VME/10 Communications Synchronization
  Initialize Switch Positions
  Initialize Communications Buffers
  Get Initial Event List
  Get Initial Time List and Set Clock
  Initialize Switch/Sensor Records
  Initialize Fault Status Variables
  Wait for Start of Mission
    Perform Scheduled Switch Operations
    Get Switch and Sensor Data
    Perform Algorithms
    Y  Fault List  n
       Send VME/10 Fault Event Data/Receive VME/10 Data  | Null
    Y  Switches in New Scheduled State  n
       Send VME/10 Switch Position Data/Receive VME/10 Data  | Null
    Y  Send Switch Performance  n
       Send VME/10 Switch Performance Data/Receive VME/10 Data  | Null
    Y  Time to Send Performance Data  n
       Send VME/10 Sensor Performance Data/Receive VME/10 Data  | Null
    Y  A New Fault Condition (Settled)  n
       Send VME/10 Fault Data/Receive VME/10 Data  | Null
    Y  Time to Look for VME/10 Data  n
       Send VME/10 Null Message/Receive VME/10 Data  | Null
    Y  New VME/10 Data  n
       Process VME/10 Data  | Null
    Repeat until Recycle
  Repeat Forever
```

SSM/PMAD CONVENTIONAL SOFTWARE (LC) 849CMP14100 ALGORITHMS



Range Check Sensor Voltage

Range Check Sensor Current

Range Check Sensor Temperature

Do For All Sensors

Check Switch Trip Conditions
Check for Switch Overtemp
Check for Switch Current Overrange
Check for Switch Position Inconsistencies
Check for Current Out of Profile Limits

Do For All Switches

Maximum Load Center Power Exceeded?

Shed Appropriate Amount of Load

Null

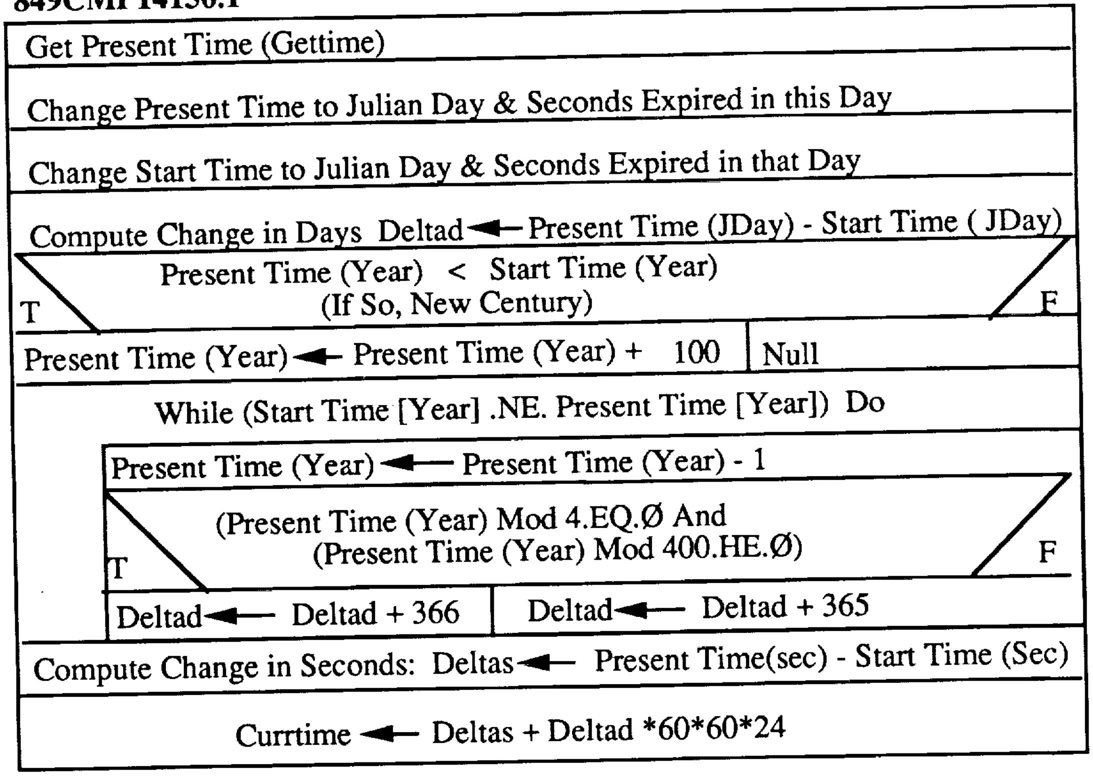
Calculate Performance Data

SSM/PMAD CONVENTIONAL SOFTWARE (PDCU)
849CMP14100P ALGORITHMS

```
    Range Check Sensor Voltage
    Range Check Sensor Current
    Range Check Sensor Temperature
Do For All Sensors
    Check Switch Trip Conditions
    Check for Switch Overtemp
    Check for Switch Current Overrange
    Check for Switch Position Inconsistencies
    Check for Current Out of Profile Limits
Do For All Switches
Calculate Performance Data
Detect Soft Faults With Kirchoff's Current Law
Based Upon RBI Configuration
```

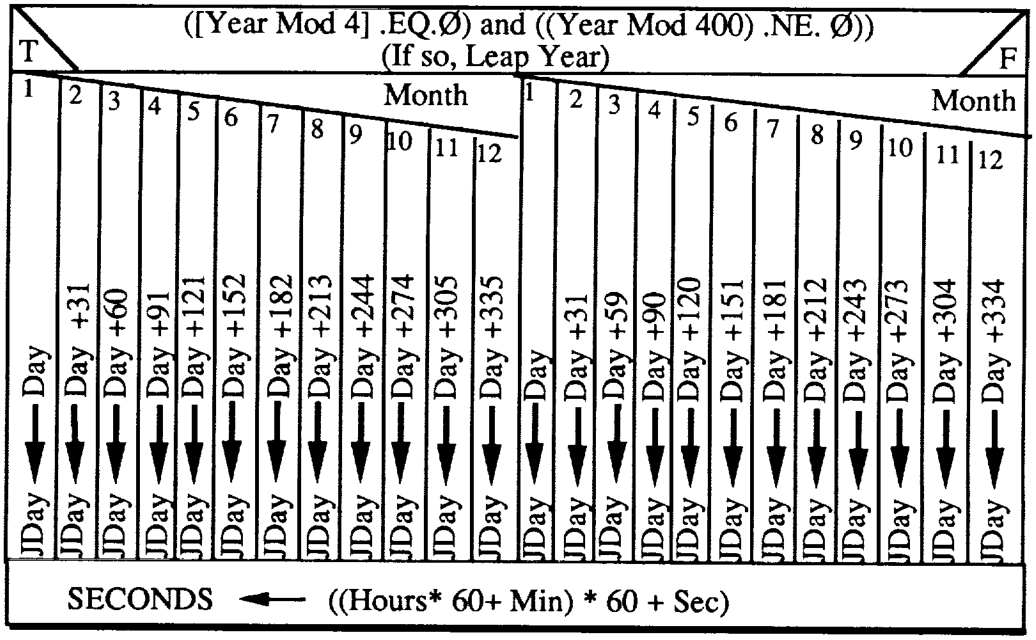
SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14150.1

CURRTIME



```
Get Present Time (Gettime)
Change Present Time to Julian Day & Seconds Expired in this Day
Change Start Time to Julian Day & Seconds Expired in that Day
Compute Change in Days  Deltad ← Present Time (JDay) - Start Time ( JDay)
Present Time (Year)  <  Start Time (Year)
      (If So, New Century)
T   Present Time (Year) ← Present Time (Year) +  100
F   Null
While (Start Time [Year] .NE. Present Time [Year]) Do
    Present Time (Year) ← Present Time (Year) - 1
    (Present Time (Year) Mod 4.EQ.Ø And
          (Present Time (Year) Mod 400.HE.Ø)
    T   Deltad ← Deltad + 366
    F   Deltad ← Deltad + 365
Compute Change in Seconds:  Deltas ← Present Time(sec) - Start Time (Sec)
Currtime ← Deltas + Deltad *60*60*24
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14150.2 **CVT DAY AND SEC**



SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14150.3

CVT TIME

Day ← 10* (Ord (Date [4]) -Ord('Ø')) + Ord (Date [5]) -Ord('Ø')
Month ← 10* (Ord (Date[1]) -Ord ('Ø')) + Ord (Date[2]) -Ord('Ø')
Year ← 10* (Ord (Date[7]) -Ord('Ø')) + Ord (Date[8] -Ord('Ø')
Hour ← 10* (Ord (Time[1]) -Ord ('Ø')) + Ord (Time[2] - Ord ('Ø')
Min ← 10* (Ord (Time[4]) -Ord ('Ø')) + Ord (Time[5]) -Ord('Ø')
Sec ← 10* (Ord (Time[7]) -Ord ('Ø')) + Ord (Time[8]) -Ord ('Ø')

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14150.4 **GETTIME**

Get Operating System Clock Time (XRTM)
Get Operating System Clock Date (XRDT)
Convert Time Format (Cvttime)

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14150.5

SETCLOCK

Convert Month of Time Now to Integer Form

Convert Day of Time Now to Integer Form

Convert Year of Time Now to Integer Form

Convert Hour of Time Now to Integer Form

Convert Minutes of Time Now to Integer Form

Convert Seconds of Time Now to Integer Form

Set The Operating System Date

Set the Operating System Time

Convert Month of Start of Mission Time

Convert Day of Start of Mission Time

Convert Year of Start of Mission Time

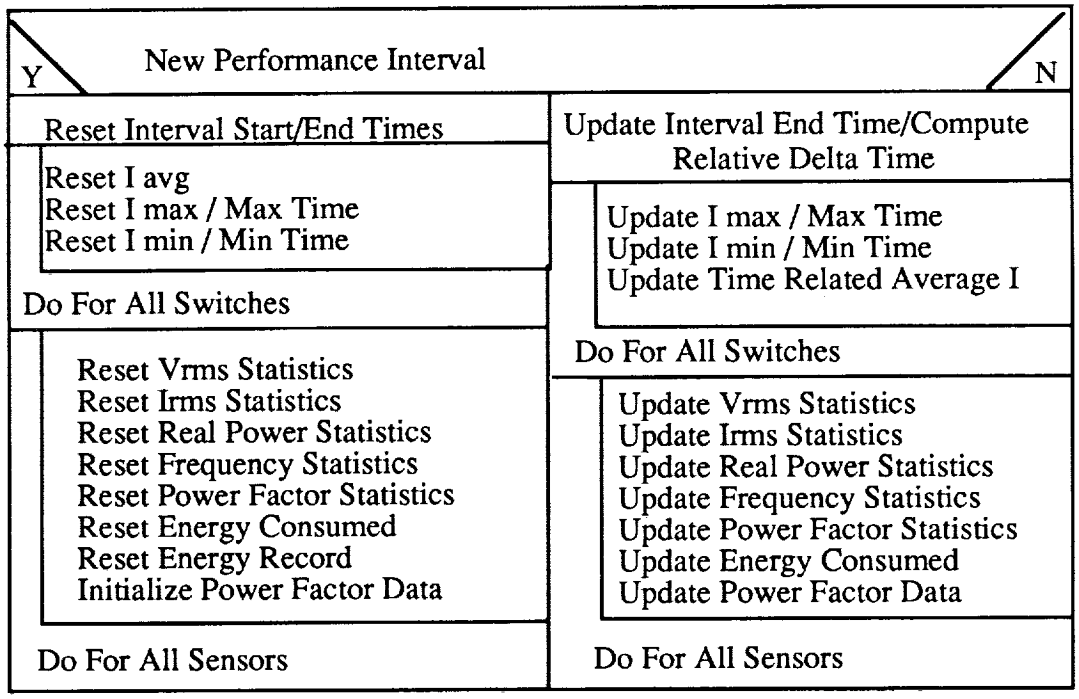
Convert Hour of Start of Mission Time

Convert Minutes of Start of Mission Time

Convert Seconds of Start of Mission Time

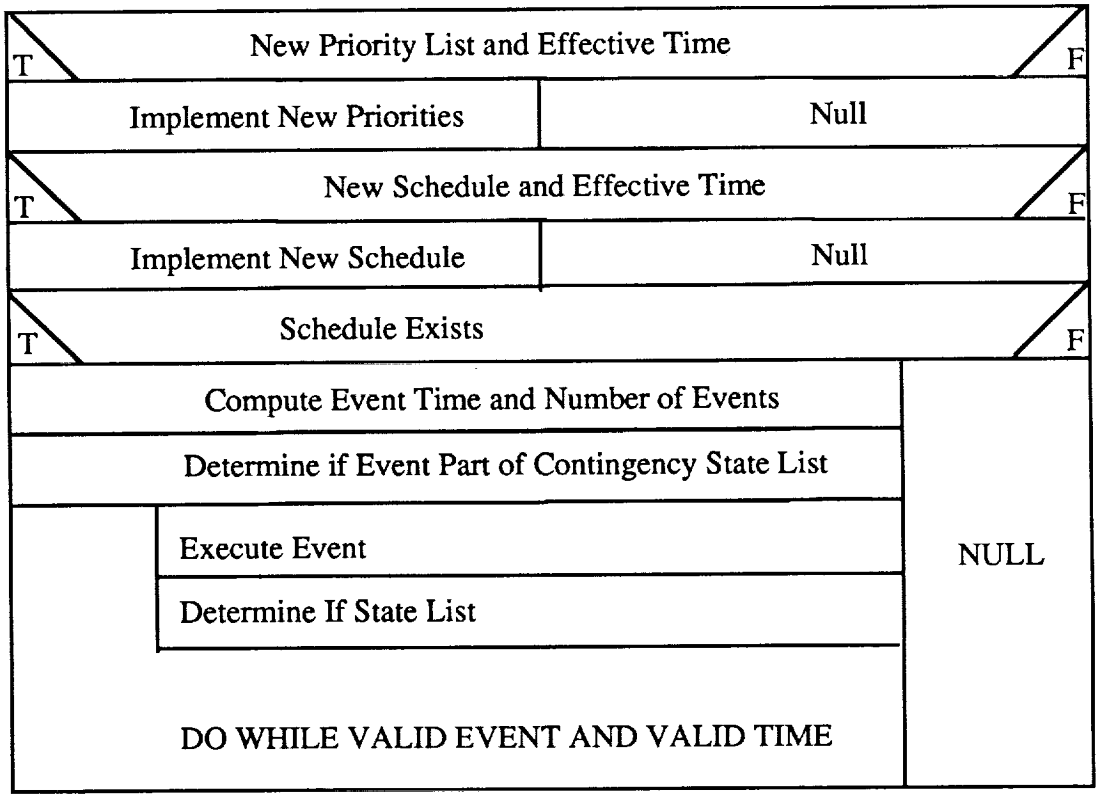
Store the Start Time

SSM/PMAD CONVENTION SOFTWARE (LC/PDCU)
849CMP14400.1 CALCENERGY



```
New Performance Interval
Y:
    Reset Interval Start/End Times
    Do For All Switches
        Reset I avg
        Reset I max / Max Time
        Reset I min / Min Time
    Do For All Sensors
        Reset Vrms Statistics
        Reset Irms Statistics
        Reset Real Power Statistics
        Reset Frequency Statistics
        Reset Power Factor Statistics
        Reset Energy Consumed
        Reset Energy Record
        Initialize Power Factor Data
N:
    Update Interval End Time/Compute
        Relative Delta Time
    Do For All Switches
        Update I max / Max Time
        Update I min / Min Time
        Update Time Related Average I
    Do For All Sensors
        Update Vrms Statistics
        Update Irms Statistics
        Update Real Power Statistics
        Update Frequency Statistics
        Update Power Factor Statistics
        Update Energy Consumed
        Update Power Factor Data
```

SSM/PMAD CONVENTIONAL SOFTWARE (LC/PDCU)
849CMP14400.2 **DOSCHEDULE**


New Priority List and Effective Time
T
F
Implement New Priorities
Null
New Schedule and Effective Time
T
F
Implement New Schedule
Null
Schedule Exists
T
F
Compute Event Time and Number of Events
Determine if Event Part of Contingency State List
Execute Event
Determine If State List
DO WHILE VALID EVENT AND VALID TIME
NULL

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14400.3

GET ALL DATA

```
Get Bus A Switch Information
Get Time Stamp
  Move Currents To Frames Data Table & Time Stamp
Get Bus B Switch Information
Get Time Stamp
  Move Currents To Frames Data Table & Time Stamp
Get Sensor Data
Get Time Stamp
  Move Sensor Info To Frames Data Table & Time Stamp
Get Temperature Data
Get Time Stamp
  Move Temperatures To Frames Data Table & Time Stamp
Get Bus Power Data (For Use Later In Power Factor Computations)
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14400.4 **MANUAL MODE**

```
Write VME/10 CONFIRMATION MESSAGE OF MANUAL MODE
RECYCLE ← FALSE

WHILE NOT RECYCLE DO
    GET VME/10 COMMAND
    COMMAND < > EXIT
    Y
        SEND COMMAND TO PROPER SIC CARD
        GET SIC RESPONSE
        SEND SIC RESPONSE TO VME/10
    N
        RECYCLE ← TRUE
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14400.5 UPDATE CONTINGENCY LIST

```
Contingency Flag ← True

Processed Fault Flag ← False

Update Schedule (849CMP14400.8)
```

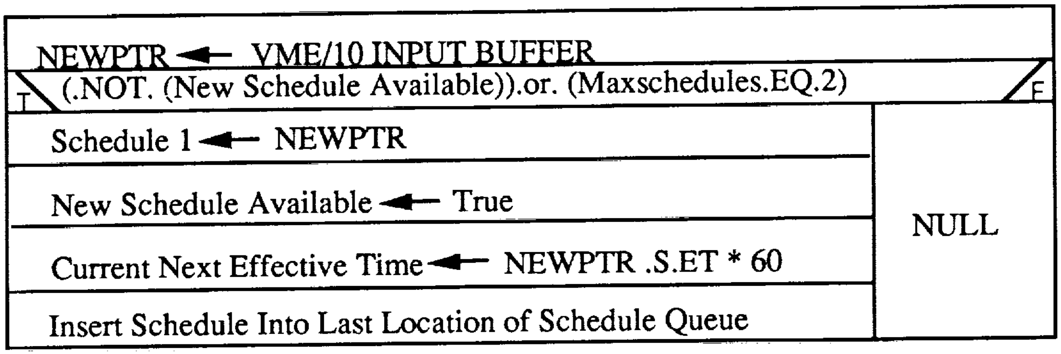
SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14400.6

UPDATE PRIORITIES

```
NEWPTR ← VME/10 INPUT BUFFER
For I ← Ø to 27
    Priority [I] ← Ø
For I ← Ø to Number of New Priorities
    Priority [New Priority . Switch] ← New Priority . Priority
```

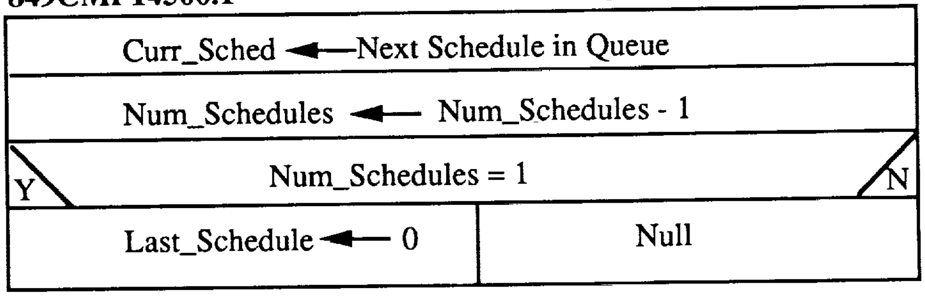
SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14400.8

UPDATE SCHEDULE



```
NEWPTR ← VME/10 INPUT BUFFER
(.NOT. (New Schedule Available)).or. (Maxschedules.EQ.2)
  T:
    Schedule 1 ← NEWPTR
    New Schedule Available ← True
    Current Next Effective Time ← NEWPTR .S.ET * 60
    Insert Schedule Into Last Location of Schedule Queue
  F:
    NULL
```

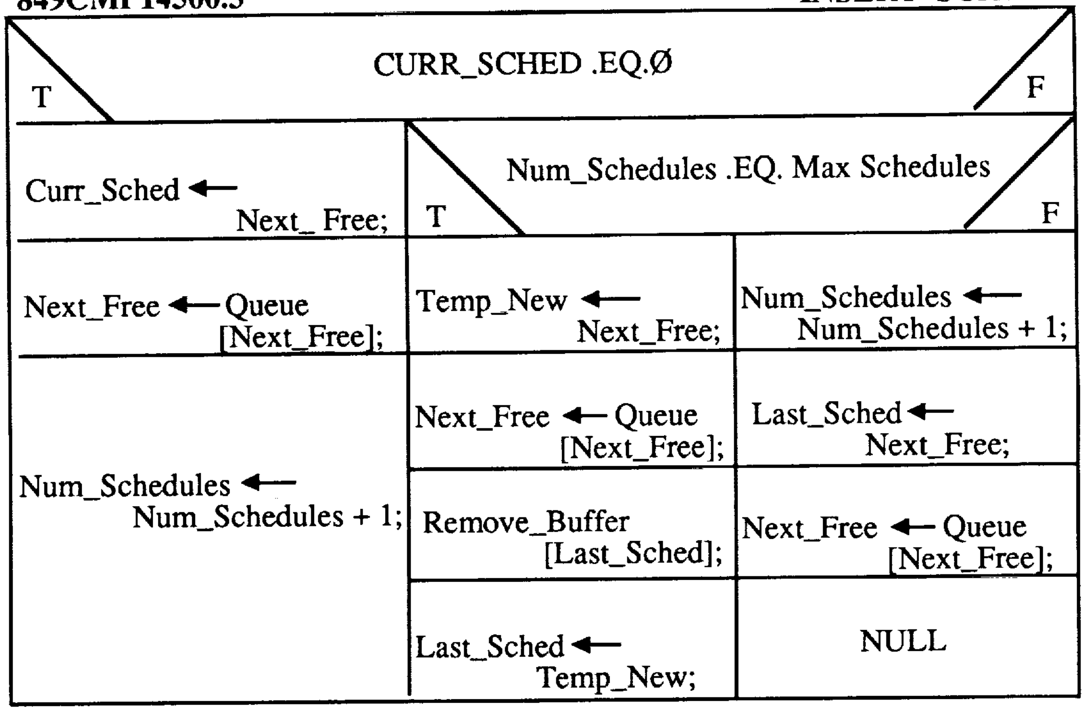
SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14500.1 **CHANGE SCHEDULE**


Curr_Sched ◄— Next Schedule in Queue
Num_Schedules ◄— Num_Schedules - 1
Num_Schedules = 1
Y
N
Last_Schedule ◄— 0
Null

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14500.2 **INIT_BUF_QUEUE**

```
CNTR ← 1
While (CNTR.LE. MAX BUFFERS) Do
    Back Q [MaxBuffers-Cntr +1] ← MaxBuffers-Cntr;
    Queue [Cntr] ← Cntr+1;
    New (BuffQ [Cntr])          {Allocate Space for BuffQ}
    Cntr ← Cntr +1;
BackQ [1] ← MaxBuffers;
Queue [MaxBuffers] ← 1;
Num_Sched ← Ø;
Curr_Sched ← Ø;
Last_Sched ← Ø;
Next_Free ← 1;
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14500.3 **INSERT SCHED**



```
CURR_SCHED .EQ.Ø
T
    Curr_Sched ← Next_ Free;
    Next_Free ← Queue [Next_Free];
    Num_Schedules ← Num_Schedules + 1;
F
    Num_Schedules .EQ. Max Schedules
    T
        Temp_New ← Next_Free;
        Next_Free ← Queue [Next_Free];
        Remove_Buffer [Last_Sched];
        Last_Sched ← Temp_New;
    F
        Num_Schedules ← Num_Schedules + 1;
        Last_Sched ← Next_Free;
        Next_Free ← Queue [Next_Free];
        NULL
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14500.4 **REMOVE_BUFFER**

```
BACKQ [QUEUE [WHICH]] ←—BACKQ [WHICH];
QUEUE [BACK Q[WHICH]]←—QUEUE [WHICH];
QUEUE [WHICH] ←— NEXT FREE;
BACKQ [WHICH] ←— BACKQ [NEXT-FREE];
BACKQ [NEXT-FREE] ←— WHICH;
QUEUE [BACK Q [WHICH]] ←— WHICH;
NEXT-FREE ←—WHICH;
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14700.1 **CONFIGURE SIC PORTS**

```
CALL CONFIGURE PORT WITH:

    Timeout = 2 seconds
    Read Line Terminator = 16# ØØØØØØØD
    Writeline Terminator = 16# ØØØØØØØD
    Parity Even
    Don't Flush Receive Queue
    Don't Flush Transmit Queue
      9600 BAUD
     PORT NUMBER 1

Wait for Interrupt From 331 Board

Call Intr and Check The Error Flag

CALL CONFIGURE  PORT WITH:

    Timeout = 2 seconds
    Read Line Terminator = 16# ØØØØØØØD
    Writeline Terminator = 16# ØØØØØØØD
    Parity Even
    Don't Flush Receive Queue
    Don't Flush Transmit Queue
      9600 BAUD
     PORT NUMBER 2

Wait for Interrupt From 331 Board

Call Intr and Check The Error Flag
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14700.2

SEND AND RECEIVE DATA TO/FROM SIC

SELECT GENERIC CARD PRIOR TO SENDING COMMAND
WAIT FOR INTERRUPT FROM 331 BOARD/CALL INTR
READ RESPONSE TO SELECT GENERIC CARD
WAIT FOR INTERRUPT FROM 331 BOARD/CALL INTR
CHECK FOR ERROR
SEND COMMAND TO SIC CARD
WAIT FOR INTERRUPT FROM 331 BOARD/CALL INTR
READ RESPONSE TO COMMAND
WAIT FOR INTERRUPT FROM 331 BOARD/CALL INTR
CHECK FOR ERROR

SSM/PMAD CONVENTIONAL SOFTWARE (LC & PDCU)
849CMP14800.1 **CMNDSWITCH**

SELECT SIC CARD FOR OPERATION

SELECT COMMAND (ON/0FF)

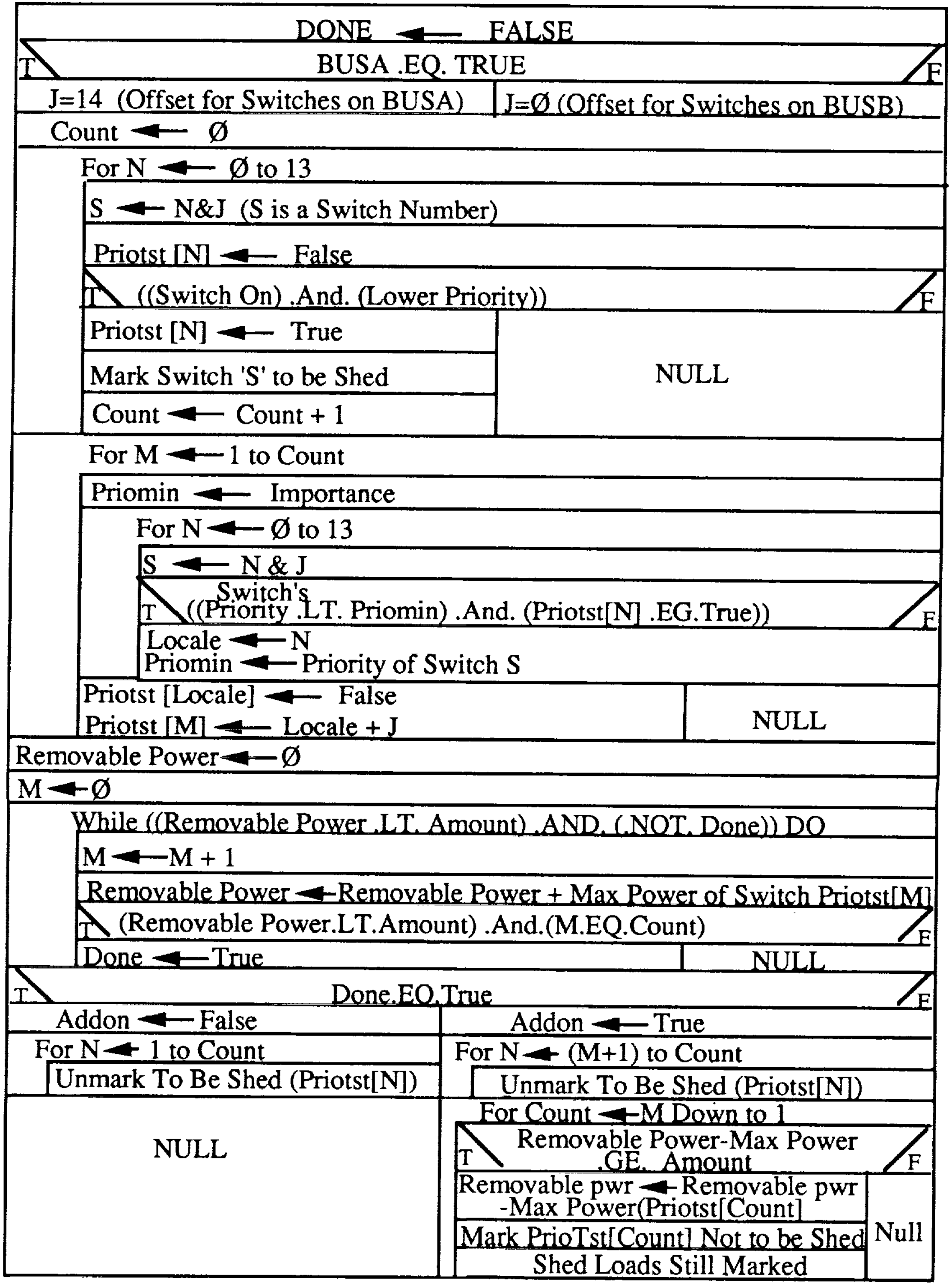
ESTABLISH BIT DEFINED COMMAND WORD

SEND OPERATION TO SIC CARD

SET LOCAL AND FRAMES SWITCHWORDS/BASED ON RESPONSE

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14800.2

LOAD SHED



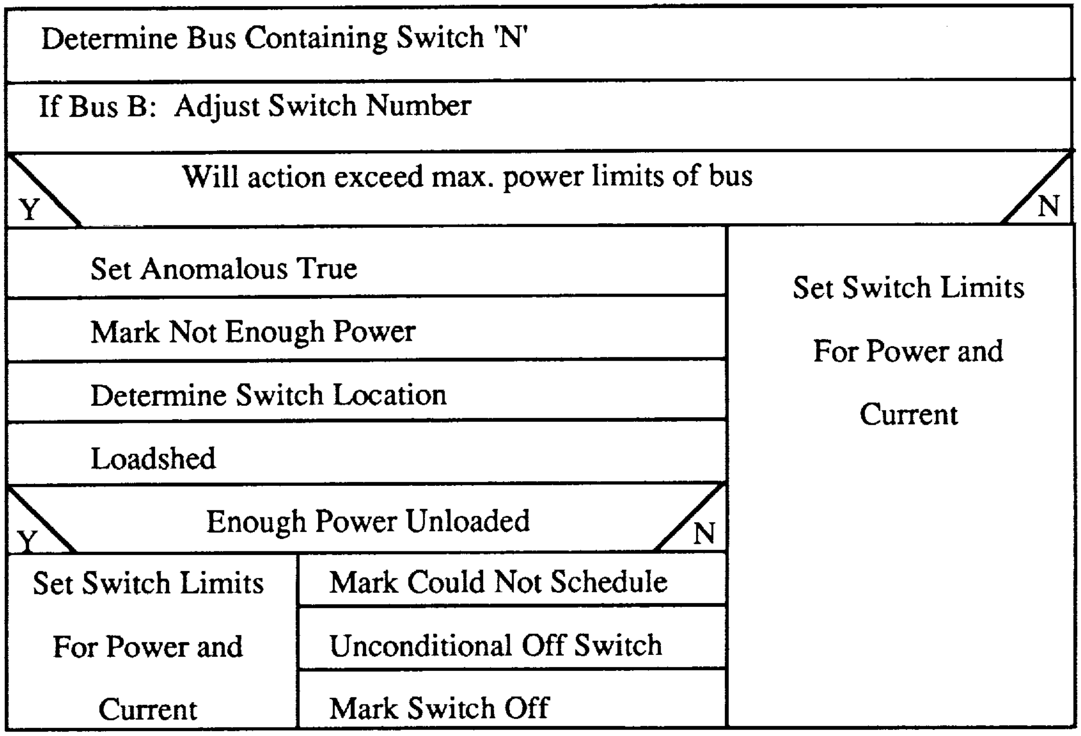
```
DONE ← FALSE
T  BUSA .EQ. TRUE  F
  T: J=14  (Offset for Switches on BUSA)
  F: J=Ø (Offset for Switches on BUSB)
Count ← Ø
For N ← Ø to 13
    S ← N&J  (S is a Switch Number)
    Priotst [N] ← False
    T  ((Switch On) .And. (Lower Priority))  F
      T: Priotst [N] ← True
         Mark Switch 'S' to be Shed
         Count ← Count + 1
      F: NULL
For M ← 1 to Count
    Priomin ← Importance
    For N ← Ø to 13
        S ← N & J
        T  ((Switch's Priority .LT. Priomin) .And. (Priotst[N] .EG.True))  F
          Locale ← N
          Priomin ← Priority of Switch S
    Priotst [Locale] ← False
    Priotst [M] ← Locale + J
                                        NULL
Removable Power ← Ø
M ← Ø
While ((Removable Power .LT. Amount) .AND. (.NOT. Done)) DO
    M ← M + 1
    Removable Power ← Removable Power + Max Power of Switch Priotst[M]
    T  (Removable Power.LT.Amount) .And.(M.EQ.Count)  F
      T: Done ← True
      F: NULL
T  Done.EQ.True  F
  T: Addon ← False
     For N ← 1 to Count
         Unmark To Be Shed (Priotst[N])
     NULL
  F: Addon ← True
     For N ← (M+1) to Count
         Unmark To Be Shed (Priotst[N])
     For Count ← M Down to 1
         T  Removable Power-Max Power .GE. Amount  F
           T: Removable pwr ← Removable pwr -Max Power(Priotst[Count]
              Mark PrioTst[Count] Not to be Shed
              Shed Loads Still Marked
           F: Null
```

SSM/PMAD CONVENTION SOFTWARE
(LC)
849CMP14800.3 **REDSW**

T — REDUNDANT SWITCH AND PERMISSION	F
DETERMINE REDUNDANT SWITCH SET PRIORITY OF REDUNDANT SWITCH RESET PRIORITY OF SWITCH ESTABLISH SINGLE EVENT SCHEDULE TO TURN ON REDUNDANT IF POWER AVAILABLE TURN ON REDUNDANT	NULL

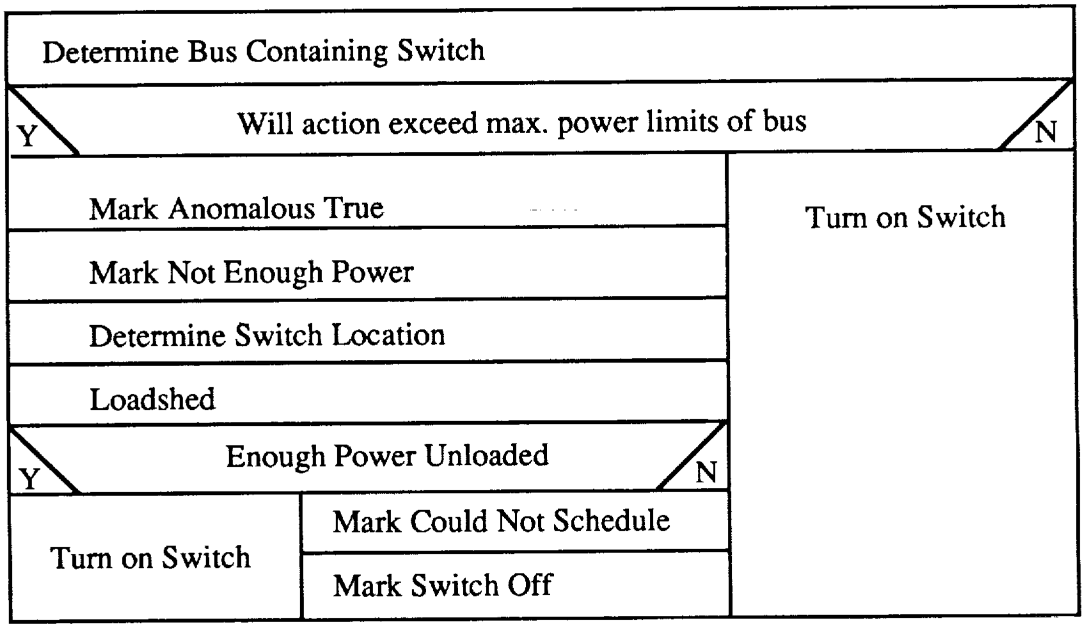
SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14800.4

Change State of Switch 'N'



```
Determine Bus Containing Switch 'N'
If Bus B:  Adjust Switch Number
Will action exceed max. power limits of bus
  Y:
    Set Anomalous True
    Mark Not Enough Power
    Determine Switch Location
    Loadshed
    Enough Power Unloaded
      Y:
        Set Switch Limits For Power and Current
      N:
        Mark Could Not Schedule
        Unconditional Off Switch
        Mark Switch Off
  N:
    Set Switch Limits For Power and Current
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14800.6 **SWITCH ON**



SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14800.7 **UNCONDITIONAL OFF**

Determine if Switch is On Bus A or Bus B

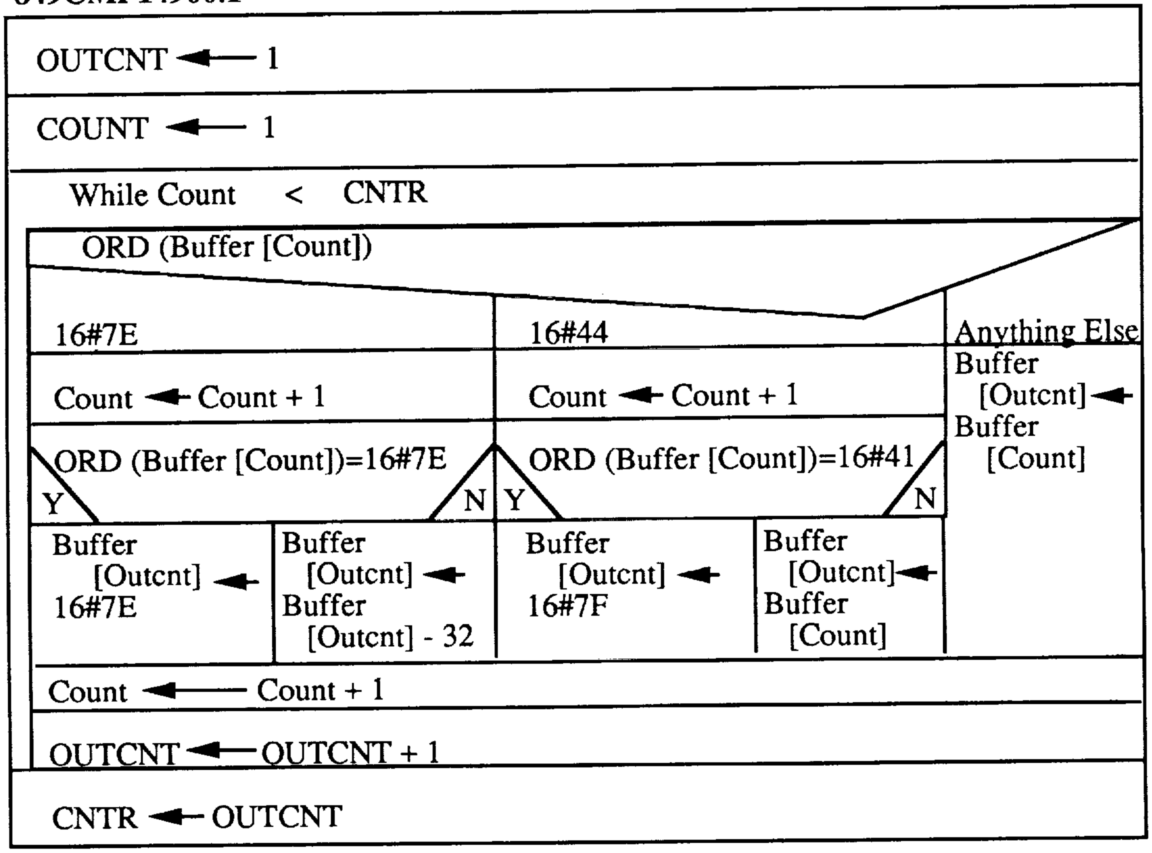
If Switch is On Bus B Adjust Switch Number

Determine Which Byte of Command Has Switch Indictors

Send Command to Turn Off Switch

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14900.1

DECODE



```
OUTCNT ← 1
COUNT ← 1
While Count < CNTR
    ORD (Buffer [Count])
        16#7E:
            Count ← Count + 1
            ORD (Buffer [Count])=16#7E
                Y: Buffer [Outcnt] ← 16#7E
                N: Buffer [Outcnt] ← Buffer [Outcnt] - 32
        16#44:
            Count ← Count + 1
            ORD (Buffer [Count])=16#41
                Y: Buffer [Outcnt] ← 16#7F
                N: Buffer [Outcnt] ← Buffer [Count]
        Anything Else:
            Buffer [Outcnt] ← Buffer [Count]
    Count ← Count + 1
    OUTCNT ← OUTCNT + 1
CNTR ← OUTCNT
```

SSM/PMAD CONVENTIONAL SOFTWARE
849CMP14900.3 **ENCODE**

```
J ← 1
I ← 1 to Length
    ORD (INBUF [I]) = ?
        1 thru 31
            OUTBUF[1] ← 16#7E
            J ← J + 1
            OUTBUF[J] ← CHR(ORD(INBUF [I]+ 16#20)
        16#7E
            OUTBUF[1] ← 16#7E
            J ← J + 1
            OUTBUF[J] ← 16#7E
        16#7F
            OUTBUF[1] ← 16#44
            J ← J + 1
            OUTBUF[J] ← 16#41
        16#44
            OUTBUF[1] ← 16#44
            J ← J + 1
            OUTBUF[J] ← 16#44
        ELSE
            OUTBUF[I] ← INBUF[I]
            NULL
    J ← J + 1
OUTBUF [J] ← 16#ØD
OUTBUF [J+1] ← 16#ØØ
OUTBUF [J+2] ← 16#ØØ
OUTBUF [J+3] ← 16#ØØ
Length ← J + 3
```

SSM/PMAD CONVENTIONAL SOFTWARE 849CMP14900.4

CONVERT VME/10 INPUT TO APPROPRIATE FORMAT

Assign a Pointer to Next Data Space in Buffer Queue
Assign Length of Pointer Type Pointed At
Call READVME10 (Read a Buffer of Data)
Assign GETVDATA to Pointer to Buffer

APPENDIX X – SSM/PMAD EXPOSITORY AND ACTIVITY PLAN

19.0 APPENDIX X: SSM/PMAD EXPOSITORY AND ACTIVITY PLAN

19.1 Preface

This document describes a large portion, if not all, of the tests required to demonstrate the operational capabilities of SSM/PMAD. It is divided into a number of parts for testing various aspects of the system. In the first part,the tests for dynamic rescheduling are described. The second part describes testing for source power changes. The third part describes tests for redundancy in the power system. And finally, the fourth part describes the tests for FRAMES diagnoses.

In addition, where there are known bugs or conceptual flaws in the system, these will be pointed out where appropriate.

19.2 Dynamic Rescheduling Testing

To test this, set up a fairly simple schedule should be set up on the scheduler. Define a number of short, 15 minute activities that all use the same switch, e.g. c05. All these activities will then have to be scheduled sequentially.

Each of these activities should require, for example, 800 watts.

Then impose a future source power change for, say, 20 minutes, of a maximum available power of 500 watts. Some of the activities should be taken off of the schedule. If the activities are interruptible or reschedulable, they should be rescheduled after the source power change is no longer in effect.

If an immediate source power change is done, the same type of interaction should occur. If activities are taken off of the schedule permanently it is probably a small bug in FRAMES.

19.3 Redundancy Testing

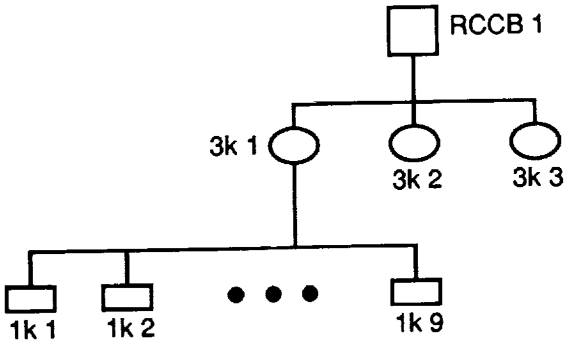
To test redundancy a number of problems must be overcome. FRAMES redundancy handling mechanisms do not currently work correctly. If a load is switched to redundant during a fault scenario, FRAMES will think that some symptoms were reported when they should not have been or that not all were reported that should have been. FRAMES will not try to turn on the switch that the load was on after it has been switched to redundant ,though.

If a switch to redundant occurs as a result of a source power change, FRAMES will handle it correctly. Unfortunately ,this is highly unlikely.

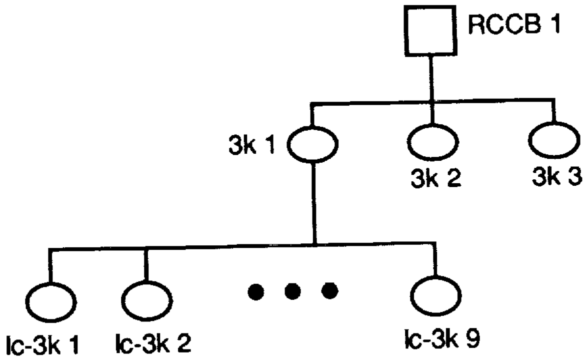
Therefore, redundant switching can only be tested at the LLP level at this point. To do this, a schedule should be set up so that a load that can be switched to redundant is currently operating on c05. Another load that cannot be switched to redundant should be operating on c22. Assume that only these two loads, at 400 watts each, are operating. Then after the schedule has begun executing, apply a short to c05. The load should then be switched to c19 and finish its execution there. Ignore whatever happens at FRAMES. The scheduler will not be notified of the load being switched to redundant either. Once the short has been applied, FRAMES could then be taken out of service except that the CAC is dependent upon FRAMES..

19.4 FRAMES Diagnosis Testing

This section describes the tests for generating FRAMES diagnoses. Each test consists of the diagnosis, its name in the code and its English output, followed by some notes about the diagnosis and how to test the diagnosis. Each diagnosis refers to the following figure:



There are two diagnoses that deal specifically with subsystem distributors instead of load centers as depicted by the following figure, (we will refer to this figure as Figure 2):



Each of the diagnoses are accompanied by a short description of the test that should produce the diagnosis. As most of the tests are actually quite complex and involve applying shorts to the right locations at specific times, we do not depict them on the figures. This would just make the figures that much more confusing. Instead the figures are provided for the purpose of reference in the diagnoses and the test descriptions to get a better idea of what the diagnosis is about.

A number of the diagnoses describe the symptoms that occurred and end with a statement: "This is not a fault scenario that is currently addressed." What this indicates is that FRAMES has encountered a scenario where more than a single fault has occurred. There are some cases where FRAMES can diagnose multiple dependent faults, but in most cases multiple faults are not diagnosed. These diagnoses are included so that FRAMES can end a line of reasoning without halting the system. In general, in these cases a number of switches will be taken out of service and autonomous operation will continue. FRAMES will not have diagnosed the fault per se, however.

Items enclosed in '<' and '>', (e.g. <this item>) are virtual slots that are filled in by actual switches and faults, etc., when the diagnosis is actually determined by FRAMES. We are using virtual slots because the same diagnosis can be used in multiple cases. In cases where multiple values are given, e.g., <fast-trip, i2t, gfi>, any of the values can be inserted. Where multiple values occur in different places in the diagnosis, a respective ordering is kept.

19.4.1 diagnose-format--new-top-or-symp-with-lower-uv

Diagnosis:

<RCCB 1> tripped on <fast-trip, i2t, gfi>.
During testing the switches below <RCCB 1> the following symptoms
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario RCCB 1 trips on either a fast-trip, i^2t, or ground fault. When testing the 3k switches below RCCB 1 we get an under voltage without the corresponding RCCB 1 retrip. This means we seem to be getting an entirely different fault than the first time.

To test this we need to initially insert one of the shorts below RCCB 1 (above the 3ks). This fault is a temporary fault. Then when the 3k switches are being tested, retrip RCCB 1 with a different fault.

19.4.2 diagnose-format--found-and-double-masked-fault

Diagnosis:

<RCCB 1> tripped on <fast-trip, i2t, gfi>.
During testing of the load center switches below <RCCB 1>, <RCCB 1> retripped with <fast-trip, i2t, gfi>.

POSSIBLE CAUSES:
A <low-impedance, high-impedance, supply-to-ground> short below
A faulty <fast-trip, i2t, gfi> sensor on both the <1k> and <3k>.

Description:

To test this disable the appropriate sensor on a 1k and the 3k above the 1k, e.g., 1k 2 and 3k 1. Then inject the short below the 1k and let it be a permanent short. All loads should have permission to test.

19.4.3 diagnose-format--RCCB-errors-during-closing-of-3ks

Diagnosis:

<RCCB 1> tripped on <fast-trip, i2t, gfi>.
During testing of the switches below <RCCB 1>, the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

Originally, RCCB 1 trips on one of the mentioned trips. First RCCB 1 is tested and tests ok. Then the 3ks are flipped one by one without a fault. Then the 3ks are sequentially closed. At this point something trips again. During the testing, all of the 1k switches have been previously opened. The point is that if a 3k will cause a trip again during closings but not flips, something unusual is occuring.

To test this, apply a short below RCCB 1 to cause it to trip. Remove the short right away. Then during closing of the 3k switches apply another short to one of the 3ks below RCCB 1. To do this after the 3ks have been flipped, quickly set a short on 3k 1. Of course you must make sure that the 3k you use is one that was being used originally.

19.4.4 diagnose-format--lower-current-trip-during-closes

Diagnosis:

<3k 1> tripped on <fast-trip, i2t, gfi>.
<1k 2> tripped on a current trip during closing of the switches below <3k
This should not be possible, if it was possible, <1k 2> should have tripped during the flipping of the switches below <3k 1>.

This is not a fault scenario that is currently addressed.

Description:

This is exactly analogous to diagnose-format--RCCB-errors-during-closing-of-3ks.

To test this, apply a short below 3k 1 (above the 1ks). Let this be a temporary short. Then, during the closing of the 1ks, put another short below one of them. To do this, set the short after flipping of the 1ks.

19.4.5 diagnose-format--overload

Diagnosis:

<3k 1> tripped on <i2t, gfi>.
<1k 5> tripped on under voltage during closing of the switches below <3k 1>.

POSSIBLE CAUSES:
An overload situation where the lower switches are drawing too much current for <3k 1>.

Description:

To have this scenario, there must be at least four 1k switches operating. Each of these switches should be allocated such that the sum total of their power is 3k watts. To simulate this, apply an i2t trip below 3k 1 and above the 1ks. Remove the short and reapply it when the fourth 1k switch is being turned on. This test requires that all the 1k switches have permission to test.

19.4.6 diagnose-format--lower-uv-during-closes-upper-fast-trip

Diagnosis:

<3k 1> tripped on <fast-trip>.
<1k 5> tripped on under voltage during closing of the switches below <3k 1>.

POSSIBLE CAUSES:
A low impedance short below <1k 5> and
A faulty fast-trip sensor on <1k 5>.

Description:

To test this, disable the fast trip sensor on 1k 5 and apply a permanent fast-trip below it.

19.4.7 diagnose-format--new-top-problem-and-uv

Diagnosis:

<3k 1> tripped on <fast-trip, i2t,gfi>.
When testing the lower switches, the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario a 3k trips. During testing of the 1ks, we get under voltages at the 1ks as expected but a different 3k trips.

To test this, apply a temporary short below 3k 1. Then, during testing of the 1k switches apply a short below 3k 2 and an open circuit below 3k 1.

19.4.8 diagnose-format--found-3k-race-uv

Diagnosis:

<3k 1> tripped on <fast-trip, i2t,gfi>.
During testing of the lower switches, <lc-3k 2>, a 3k rpc, tripped with under voltage.

POSSIBLE CAUSES:
A <low-impedance, high-impedance, supply-to-ground> fault below <lc-3k 2>.

Description:

This diagnosis refers to Figure 2.

To test this we first apply a temporary short below 3k 1. Then during testing we apply the same short again exactly when the load center 3k is being flipped. This could actually be done by disabling the appropriate current sensor below one of the load center 3ks and applying a permanent short below that switch.

19.4.9 diagnose-format--u-v-during-flips-w-out-upper-trip

Diagnosis:

<3k 1> tripped on <fast-trip, i2t,gfi>.
When opening <3k 1> the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

First, this diagnosis is wrong. It should suggest the problem as the following: First 3k 1 trips. During flips of the 1ks, we get under voltages back from them. But the 3k does not report tripping again.

To test this, apply a temporary short below 3k 1. Then after testing 3k 1 and before testing the 1ks, open the circuit between 3k 1 and the 1ks.

19.4.10 diagnose-format--found-3k-race

Diagnosis:

<3k 1> tripped on <fast-trip, i2t,gfi>.
During testing of the lower switches, <lc-3k 2>, a 3k rpc, tripped with <fast-trip, i2t, gfi>.

POSSIBLE CAUSES:
A <low-impedance, high-impedance, supply-to-ground> fault below <lc-3k 2>.

Description:

This diagnosis refers to Figure 2.

This is the same test as in 4.8. The difference here is that first 3k 1 trips, then lc-3k 2 trips with the same symptom. To do this, apply a temporary short below 3k 1. Then, after opening all the switches and right before testing of the load center 3ks, apply the same short below lc-3k 2.

19.4.11 diagnose-format--new-top-not-under-v

Diagnosis:

<RCCB 1> tripped on <fast-trip, i2t,gfi>.
When opening <3k 1>, the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, RCCB 1 trips on a current trip. During testing of the 3k switches, RCCB 1 retrips on a different symptom. This indicates a different problem than originally expected and could be multiple faults.

To test this, first apply a short below RCCB 1. During testing of the 3ks, apply a different short below RCCB 1.

19.4.12 diagnose-format--too-many-independent-failures-when-opening

Diagnosis:

<switch> tripped on <fast-trip, i2t,gfi>.
When opening <switch>, the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, one of the switches trips on a current trip. When first opening all the relevant switches, new, unexpected symptoms occur.

To test this, apply a short below 1k 1. After detecting the fault indication at FRAMES, also apply a short at 1k 2.

19.4.13 diagnose-format--found-and-single-masked-fault

Diagnosis:

<3k 1, RCCB 1> tripped on <fast-trip, i2t,gfi>.
When we flipped <lower switch>, <3k 1, RCCB 1> retripped on <fast-trip, i2t, gfi>.

POSSIBLE CAUSES:
A failure in the <fast-trip, i2t, gfi> sensor of <lower switch> and
A <low-impedance, high-impedance> fault below <lower switch>.

Description:

To test this, disable the fast trip sensor at 1k 1 or 3k 1. Then, apply a fast trip below 1k 1 or 3k 1. Let this be a permanent short.

19.4.14 diagnose-format--multiple-tops-during-lower-flips

Diagnosis:

<3k 1, RCCB 1> tripped on <fast-trip, i2t,gfi>.
When reclosing <3k 1, RCCB 1> the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

First, this diagnosis is wrong.

The RCCB or a 3k first trips on a current trip. Then, when flipping a lower switch, multiple faults occur. This is an unexpected situation and is not diagnosable.

To test this, apply a temporary short below 3k 1. Then, when flipping of the 1ks is to occur, apply a short to both 1k 1 and 1k 2.

19.4.15 diagnose-format--failure-to-close-top

Diagnosis:

<3k 1, RCCB 1> tripped on <fast-trip, i2t,gfi>.
When reclosing <3k 1, RCCB 1> the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario the switch is first flipped without retripping it. Then, when it is closed for subsequent testing of the lower switches, it trips again.

To test this, apply a temporary short below 3k 1. Then, after 3k 1 is flipped reapply the short.

19.4.16 diagnose-format--no-retrip-on-reclose-of-lc-switch

Diagnosis:

<1k 2> tripped on <fast-trip, i2t,gfi>.
It did not trip again when it was reclosed during testing.

POSSIBLE CAUSES:
Most Likely:
Short circuit in cable to lad that was burned clear or was otherwise removed.
Similar temporary short in load on that circuit.
Less Likely:
Intermittent RPC control failure involving current sensor, current discriminator, or EPLD.

Description:

To test this, apply a temporary short below 1k 2.

19.4.17 diagnose-format--with-back-rush

Diagnosis:

<3k 1, RCCB 1> tripped on <fast-trip>.
These lower switches tripped on fast trip: <switches>
These lower switches tripped on under voltage: <switches>

CAUSES of <3k 1, RCCB 1> trip:
Short circuit in the cable below <3k 1, RCCB 1>.
Short circuit in the switch output of <3k 1, RCCB 1>.
Short circuit in the input of one of the following switches: <switches below <3k 1, RCCB 1>>.
CAUSE of <fast trip switches below> fast trip:
Energy storing load on each of these switches discharged into the short circuit.

Description:

To test this, apply a short below 3k 1 and get 3k 1 and 1k 1 both to fast trip at once.

Note: This will probably not work properly in the case of back rush where the short is below RCCB 1. In this case, some 1ks will fast trip and RCCB 1 will fast trip. This diagnosis expects fast trip symptoms from the level immediately below.

19.4.18 diagnose-format--multiple-retrip-on-single-reclose

Diagnosis:

<switch> tripped on <fast-trip, i2t, gfi>.
When <switch> was reclosed, the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, any switch trips on a current trip. When the switch is closed, multiple symptoms occur.

To test this, apply a short below 3k 1. Then, after FRAMES receives the data, apply an additional short under 3k 2. Both of these should be permanent shorts.

19.4.19 diagnose-format--new-top-symptom-when-manipulating-lower-lc-switches

Diagnosis:

<RCCB 1> tripped on <fast-trip, i2t, gfi>.
<RCCB 1> tripped on <i2t, gfi, fast-trip, u-v> when <3k 1> was reclosed.

This is not a fault scenario that is currently addressed.

Description:

In this scenario RCCB 1 first trips on a current trip. When the 3ks are being tested, RCCB 1 retrips only with a different symptom.

To test this, first apply a temporary fast trip below RCCB 1. Then, during testing of the 3ks, apply an i2t short below RCCB 1.

19.4.20 diagnose-format--retrip-on-reclose-with-new-top

Diagnosis:

<switch> tripped on <fast-trip, i2t, gfi>.
When <switch> was reclosed <switch 2> tripped on <fast-trip, i2t, gfi, u-v>.

This is not a fault scenario that is currently addressed.

Description:

In this scenario, any switch trips on a current trip. When the switch is reclosed, an entirely new symptom is found.

To test this, apply a temporary fast trip to 1k 1. Then after FRAMES opens the switch, before further testing, apply a fast trip to 1k 2.

19.4.21 diagnose-format--retrip-on-reclose-with-new-symptom

Diagnosis:

<switch> tripped on <fast-trip, i2t, gfi>.
When <switch> was reclosed it tripped on <i2t, fast-trip, gfi, u-v>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, any switch trips on a current trip. When the switch is reclosed, the switch trips on a different symptom.

To test this, apply a temporary fast trip to 1k 1. Then, after FRAMES opens the switch, before further testing, apply an i2t trip to 1k 1.

19.4.22 diagnose-format--lc-retrip-on-reclose

Diagnosis:

<1k 1>, an 1k rpc in a load center, tripped on <fast-trip, i2t, gfi>. When it was reclosed during testing, it tripped again with the same symptom.

POSSIBLE CAUSES:
Most Likely:
Short Circuit supply to return in the cable below <1k 1>.
Short Circuit in load equipment on this circuit.
Less Likely:
Short circuit in RPC output wiring.
RPC control failure involving current sensor, current discriminator, or EPLD.

Description:

To test this, simply apply a permanent short below one of the 1k switches.

19.4.23 diagnose-format--retrip-on-reclose

Diagnosis:

<3k 1, RCCB 1> tripped on <fast-trip, i2t, gfi>.
When it was reclosed during testing, it tripped again on <fast-trip, i2t, gfi>.

POSSIBLE CAUSES:
Most Likely:
Short circuit in cable below <3k 1, RCCB 1>.
Short circuit in the output of <3k 1, RCCB 1>.
Short circuit in the input of one of the following switches: <switches below tripped switch>.
Less Likely:
Failure of current sensor, current comparator, or EPLD of <3k 1, RCCB 1>.
(if it was 3k 1 and it was an i2t trip and there were more than 3 1ks on, then:
Overload resulting from improper current increase in a heavily loaded load center.)

Description:

To test this, simply apply a permanent short below one of the 3k switches or below RCCB 1.

19.4.24 diagnose-format--single-top-with-under-v-no-multiple-possible-and-no-current-above

Diagnosis:

<switch> tripped on under voltage.
The other <1k, 3k, rccb>s were not in a position to trip when <switch> tripped.
<switch above> the <3k, rccb, rbi> above, registers no current flow.

POSSIBLE CAUSES:
Most Likely:
An open circuit below <switch above>.
Less Likely:
Failure of the voltage sensor of <switch>.
Failure of the voltage comparator of <switch>.
Failure of the EPLD of <switch>.
(if the switch was an RCCB:
It is also possible that the power has not been turned on.)

Description:

To test this, apply an open circuit fault between 3k 1 and the 1ks below it.

19.4.25 diagnose-format--single-top-with-under-v-multiple-possible-and-no-current-above

Diagnosis:

<switch> tripped on under voltage.
<x> of the <1k, 3k, rccb>s were closed when <switch> tripped but did not trip.
<switch above>, the <3k, rccb, rbi> above, registers no current.

This is not a fault scenario that is currently addressed.

Description:

In this scenario, a switch trips on under voltage. Some of the siblings of the switch were closed when the switch tripped but did not trip themselves. In addition, the switch above has no current flow.

To test this, disconnect 1k 1 from the power bus and cause the current sensor of 3k 1 to always read 0. Also, have some other loads connected in the load center below 3k 1.

19.4.26 diagnose-format--single-top-with-under-v-no-multiple-possible-and-current-above

Diagnosis:

<switch> tripped on under voltage.
<siblings of switch> were not in a position to trip when <switch> tripped.
<switch above> the <3k, rccb, rbi> above, registers a positive current.

POSSIBLE CAUSES:
Most Likely:
The current sensor, current comparator, or EPLD of <switch above> may be faulty AND
one of:
Supply to return fault in the cable above <switch>.
Supply to return fault in the switch output of <switch above>.
Supply to return fault in the switch input of one of the <1k, 3k, rccb>s.
(if the switch was a 1k:
Less Likely:
In addition to the above:
The current sensor, current comparator, or EPLD of <switch above> may be faulty.

Description:

In this scenario, one of the switches trip on under voltage. None of its siblings are in a position to trip. Also, the switch above has current flowing through it but does not trip. Since RCCBs cannot trip on under voltage, this diagnosis should always be ok.

To test this, set up one load on 1k 1. Then, disable the current comparator of 3k 1 and apply a short below 3k 1.

19.4.27 diagnose-format--single-top-with-under-v-multiple-possible-and-current-above

Diagnosis:

<switch> tripped on under voltage.
<x> of the <1k, 3k, rccb>s were closed when <switch>tripped but did not trip.
<switch above> the <3k, rccb, rbi> above, registers a positive current.

POSSIBLE CAUSES:
Most Likely:
Failure in the voltage sensor of <switch>.
Failure in the voltage comparator of <switch>.
Failure in the EPLD of <switch>.

Description:

In this scenario, we can suppose 1k 1 trips on under voltage. 1k 2 also has a load running but does not trip. Also 3k 1 has current flowing through it.

To test this, disconnect 1k 1 from the power bus during operation. Also, have additional loads running in the load center.

19.4.28 diagnose-format--single-(1k,3k)-top-with-under-v-no-multiple-possible-and-current-over-limit-above

Diagnosis:

<1k, 3k> tripped on under voltage.
<siblings of switch> were not in a position to trip when <1k, 3k> tripped.
<3k, rccb>, the <3k, rccb> above, registers a current over limit condition but does not report tripping.

POSSIBLE CAUSES:
Most Likely:
The current sensor, current comparator, or EPLD of <3k, rccb> may be faulty AND

One of:
Supply to return fault in the cable above <1k, 3k>.
Supply to return fault in the switch output of <3k, rccb>.
Supply to return fault in the switch input of one of the <1k, 3k)s.
Less Likely:
In addition to the above:
The current sensor, current comparator, or EPLD of <3k, rccb> may be faulty.

Description:

In this scenario, we can suppose 1k 1 trips on under voltage. 3k 1 has a current over limit condition and there were no other siblings of 1k 1 that could have tripped. Thus, there should be a fault and a bad current sensing device on the switch above (3k 1).

To test this, have only one load in the load center active. Then, fake out the sensor on 3k 1 to think the current is over limit so it trips. Keep the sensor faked out.

19.4.29 diagnose-format--single-(1k,3k)-top-with-under-v-multiple-possible-and-current-over-limit-above

Diagnosis:

<1k 1, 3k 1> tripped on under voltage.
<3k 1, RCCB 1>, the <3k, rccb> above, did not report tripping but does register a current over limit condition.

POSSIBLE CAUSES:
Most Likely:
A fault in the switch input of <1k 1, 3k 1> with a failure of the trip mechanism of <3k 1, RCCB 1>.
Less Likely:
A short somewhere below <1k 1, 3k 1> with a failure of <1k 1, 3k 1> to recognize it and a failure of <3k 1, RCCB 1> to trip.
A failure in the voltage sensor of <1k 1, 3k 1>, a failure in the current sensor of <3k 1, RCCB 1> and <3k 1, RCCB 1> not tripping.

Description:

This is a complex scenario. Simply put, suppose 1k 1 trips on under voltage. There are siblings of 1k 1 that are also operating loads but do not trip. Additionally, 3k 1 reports a current over limit condition but does not trip.

To test this, unhook 1k 1 from the power bus during operation. Also, at the same time, fake the sensor of 3k 1 into thinking that the current is over limit and disable the trip mechanism so it cannot trip or report tripping.

19.4.30 diagnose-format--multiple-different-rccb-top & diagnose-format--multiple-rccb-top-not-under-v

Diagnosis:

The following symptoms occurred:
<symptoms>

This is a fault scenario that is not currently addressed.

Description:

In this scenario, both RCCBs in a load center trip either both on under voltage or else on different symptoms.

To test this, simultaneously apply a fast trip below one RCCB and an i2t trip below the other.

19.4.31 diagnose-format--multiple-different-rccb-top-under-voltage

Diagnosis:

The following symptoms occurred:
<symptoms>
It is possible that the power was not turned on.

This is a fault scenario that is not currently addressed.

Description:

In this scenario, both RCCBs in a load center trip on under voltage.

Unfortunately, this is an impossible scenario.

19.4.32 diagnose-format--multiple-3k-top-not-identical

Diagnosis:

<3k 1> tripped on <fast-trip, i2t, gfi>.
<3k 2> tripped on <i2t, gfi, fast-trip>.

This is a fault scenario that is not currently addressed.

Description:

In this scenario, two 3k rpcs trip with different symptoms.

To test this, have 3k 1 and 3k 2 both trip with different symptoms.

19.4.33 diagnose-format--multiple-3k-top-crossed

Diagnosis:

The following 3k RPCs tripped on fast-trip:
3k 1, 3k 2.
The third 3k RPC was also closed but did not report tripping.

POSSIBLE CAUSE:
A short circuit between the output cables of the two 3k RPCs.

Description:

To test this, try crossing the output cables of two 3k RPCs.

19.4.34 diagnose-format--multiple-3k-top-under-voltage-not-all-that-could

Diagnosis:

The following 3k RPCs tripped on under voltage:
3k 1, 3k 2.
The third 3k RPC was also closed and could have tripped bud did not.

This is not a fault scenario that is currently addressed.

Description:

To test this, unhook both 3k 1 and 3k 2 both at the same time from the power bus during system operation.

19.4.35 diagnose-format--multiple-different-3k-top

Diagnosis:

The following symptoms occurred:
3k 1 tripped on <fault 1>
3k 2 tripped on <fault 2>
3k 3 tripped on <fault 3>

This is not a fault scenario that is currently addressed.

Description:

To test this, apply three faults to the 3k switches where not all the faults are the same.

19.4.36 diagnose-format--multiple-3k-top-not-under-v

Diagnosis:

The following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, two or three 3k switches trip with identical faults (not under voltage).

To test this, apply two or three faults to the 3k switches. If two faults are applied, do not do fast-trip as this will look like crossed output cables.

19.4.37 diagnose-format--multiple-3k-top-under-v

Diagnosis:

The following 3k RPCs tripped on under voltage:
3k 1, 3k 2, 3k 3
<RCCB 1>, the rccb above, did not report tripping.

POSSIBLE CAUSES:
A resistive contact in the <RCCB 1>
An open or disconnected cable below <RCCB 1> and above the 3k RPCs.
A low impedance fault below <RCCB 1> and above the 3k RPCs with <RCCB 1> failing to trip on i2t.

Description:

In this scenario, all the 3k switches trip on under voltage.

To test this, apply an open circuit below the rccb.

19.4.38 diagnose-format--more-than-two-1k-not-fast-trip-or-under-v

Diagnosis:

The following load center RPCs tripped:
<symptoms>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, multiple 1k switches tripped with different faults or not all fast trip or under voltage.

To test this, apply multiple faults at the same time to more than one 1k switches.

19.4.39 diagnose-format--multiple-1k-not-all-that-could-under-voltage

Diagnosis:

The following load center RPCs tripped on under voltage:
<switches>
The following load center RPCs did not trip on under voltage and were in a position to do so:
<more switches>

This is not a fault scenario that is currently addressed.

Description:

In this scenario, multiple 1k switches tripped with under voltage. In addition, more of the load center switches were also active when the trips occur.

To test this, apply multiple under voltages at the same time to more than one 1k switches.

19.4.40 diagnose-format--multiple-1k-all-that-could-under-voltage

Diagnosis:

The following load center RPCs tripped on under voltage:
<switches>
<3k 1>, the 3k RPC above did not report tripping.

POSSIBLE CAUSES:
Most Likely:
A resistive contact in <3k 1>
An open or disconnected cable below <3k 1> and above the load center RPC.

A resistive cable below <3k 1> and above the load center RPC.
Less Likely:
A low impedance fault below <3k 1> and a failed overload current sensor of <3k 1>.

Description:

To test this, simply apply an open circuit below 3k 1 during operation.

19.4.41 diagnose-format--more-than-two-1k-fast-trip-no-permission-to-test

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
None of these switches have permission to test.
Further isolation of the fault is not possible.

POSSIBLE CAUSES:
A short circuit in or below any of these load center RPCs.

Description:

In this scenario, multiple 1k switches trip on fast trip. No testing can be done as none of the loads have permission to test.

To test this, set up the loads so that they do not have permission to test. Then, apply a fast trip to at least two of the 1ks at the same time.

19.4.42 diagnose-format--too-many-and-unequal-retrips-on-multiple-lc-tops

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
During testing the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed:

Description:

In this scenario, multiple 1k switches trip on fast trip. During testing, more than two trips occur again and the number of retrips is not equal to the number of original trips.

This is not straightforward to test. First, multiple 1ks need to trip on fast trip. Then, during testing multiple 1ks (a different number) need to trip as well.

19.4.43 diagnose-format--too-many-retrips-on-multiple-lc-tops

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
During testing the following symptoms occurred:
<symptoms>

This is not a fault scenario that is currently addressed:

Description:

In this scenario, multiple 1k switches trip on fast trip. During testing, more than two trips occur again and the number of retrips is equal to the number of original trips. Additionally, the number of trips is greater than two.

This is not straightforward to test. First, multiple 1ks need to trip on fast trip. Then, during testing the same 1ks should be tripped again, all at the same time.

19.4.44 diagnose-format--no-retrip-on-multiple-lc-top

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
During testing, none of these switches re-tripped.
(if some of the switches did not have permission to test:
The following switches did not have permission to test:
<switches>

POSSIBLE CAUSES:
A low impedance fault in the switch output, cable, or load below one of these switches.
A short between the output cables of two or more of these.)

Description:

In this scenario, multiple 1k switches trip on fast trip. If some of the switches have no permission to test we get one case, otherwise we get the other.

To test this, apply a temporary fast trip to multiple 1k switches at the same time.

19.4.45 diagnose-format--output-cables-short-in-lc

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
During testing, both <1k 1> and <1k 2> re-tripped on fast trip.

POSSIBLE CAUSES:
Most Likely:
A short between the output of <1k 1> and <1k 2>.
(if some switches did not have permission to test:
However, the following switches did not have permission to test:
<switches>)

Description:

In this scenario, multiple 1k switches trip on fast trip. When tested, only two of them retripped (during closes).

To test this, apply a short between the output cables of two load center switches during operation and keep it there.

19.4.46 diagnose-format--back-rush-in-lc

Diagnosis:

The following load center RPCs tripped on fast trip:
<switches>
During testing, <1k 2> re-tripped on fast trip.

POSSIBLE CAUSES:
Most Likely:
A low impedance short in the cable below <1k 2>.
A low impedance short in the switch output of <1k 2>.
A low impedance short in the load below <1k 2>.
Cause of other switches tripping:
Backrush due to energy storage in the loads.
(if some switches did not have permission to test:
Less Likely:
A short between the output cables of some of the switches--the following switches did not have permission to test <switches>.)

Description:

In this scenario, multiple 1k switches trip on fast trip. When tested, only one switch retrips. This indicates a possible back rush situation.

To test this apply a temporary fast trip to at least two load center switches at once. Only leave one of the fast trips in place.

19.4.48 diagnose-format--single-top-under-v-with-failure-to-open-subordinate

This diagnosis is not reachable in the code.

19.4.49 diagnose-format--single-lc-top-under-v-with-failure-to-open-non-top

This diagnosis is not reachable in the code.

19.4.50 diagnose-format--no-permission-to-close-top

Diagnosis:

<1k 2> tripped on <fast-trip, i2t, gfi>.
<1k 2> does not have permission to close.

POSSIBLE CAUSE:
A <fast-trip, i2t, gfi> fault in some subcomponent of the power network headed by <1k 2>.

Description:

In this scenario a 1k switch trips on a current trip. Additionally, it does not have permission to test.

To test this, set up a load without permission to test on 1k 2. Then, apply a short below 1k 2.

This diagnosis has a probable bug in it as well.

19.4.51 diagnose-format--single-top-with-failure-to-open-subordinate

Diagnosis:

The initial critical symptom in this situation was <3k 1, RCCB 1> tripping on <fast-trip, i2t, gfi>.
During routine precautionary switch opening, <1k, 3k> failed to respond to an open command.
This indicates an Altera chip (PLD) failure in <1k, 3k>, which could have caused it not to respond to a <fast-trip, i2t, gfi> condition in the circuit beneath it.
The fault then propagated up to <3k 1, RCCB 1> above, with the <1k, 3k> below tripping on under voltage.
It is not possible to test this further since the results of any relevant testing could be attributed to the failed Altera chip in <1k, 3k>.

Description:

To test this situation, fake out the current sensor in 1k 2 and apply a short below 1k 2. Also, disable 1k 2s opening mechanism.

This diagnosis is probably incorrectly coded and may result in an error condition.

19.4.52 diagnose-format--single-top-with-failure-to-open-top

Diagnosis:

<switch> tripped on <fast-trip, i2t, gfi>.
When commanded to open <switch> did not respond.

POSSIBLE CAUSES:
A failure in the EPLD of <switch>.

Description:

To test this, apply a short below 1k 2. Then, disable 1k 2s opening mechanism.

19.4.53 diagnose-format--no-permission-to-test-in-load-center

This diagnosis is not reachable in the code.

19.4.54 diagnose-format--no-permission-to-test-in-level-below

Diagnosis:

<3k 1> tripped on <fast-trip, gfi>.
Testing is not permitted in the switches below <3k 1>.
The fault is only isolated to a position in or below the switches below <3k 1>.