

Contract NAS8-36433
MCR-94-1315

Final Report

May 1994

SPACE STATION MODULE POWER MANAGEMENT AND DISTRIBUTION SYSTEM (SSM/PMAD)

(NASA-CR-193991) SPACE STATION
MODULE POWER MANAGEMENT AND
DISTRIBUTION SYSTEM (SSM/PMAD)
Final Report (Martin Marietta
Corp.) 591 p

N95-12819

Unclas

G3/20 0019791

MARTIN MARIETTA

SSM/PMAD Final Report

Report Compiled by

William Miller

Daniel Britt

Michael Elges

Chris Myers

(This Page Intentionally Left Blank)

Foreword

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 Final Report, as specified in the contract data requirements list. In particular, the work was performed for the Power Systems Branch (EB72) at NASA/MSFC.

Readers of this document are referred to:

1. NASA Contractor Report 4260 Space Station Automation of Common Module Power Management and Distribution.
2. NASA Contractor Report 4273 Knowledge Management: An Abstraction of Knowledge Base and Database Management Systems.
3. Space Station Automation of Common Module Power Management and Distribution, Interim Final Report, Volume II; MCR-89-516.
4. Space Station Module Power Management and Distribution Operational Study Report; MCR-92-1414.

~~PREVIOUS PAGE BLANK NOT FILMED~~

(This Page Intentionally Left Blank)

Contents

1	Introduction	1
1.1	Purpose	1
1.2	History	1
1.3	Summary	6
2	System Description	7
2.1	Overview	7
2.2	Requirements	8
2.2.1	Background	8
2.2.2	System Requirements	9
2.3	Top Level Configuration	21
2.4	Operational Theory	25
2.4.1	Overview	25
2.4.2	Function Partitioning in Brief	25
2.4.3	Initialization	25
2.4.4	Schedule Creation	28
2.4.5	Schedule Execution	28
2.4.6	Fault Handling	28
2.4.7	User Schedule Change	29
2.4.8	Manual Control	30
2.5	Hardware Description	32
2.5.1	Hardware Description	32
2.6	Software Description	34
2.6.1	Knomad	34
2.6.2	MAESTRO	36
2.6.3	Front-End Load Enable Scheduler (FELES)	39
2.6.4	Load Priority List Management System Update	40
2.6.5	The KANT Run-Time Planning System	42
2.6.6	The Fault Recovery and Management Expert System	54
2.6.7	The SSM/PMAD Seamless User Environment Interface	89

2.6.8	The Transaction Software	119
2.6.9	Lowest Level Functions	129
2.6.10	Intermediate Levels of Autonomy	129
2.6.11	Soft and Incipient Faults in the <i>SSM/PMAD</i> Test Bed	149
3	Future Directions	159
3.1	Testbed Enhancements	159
3.2	Technology Transfer Possibilities	160
3.2.1	Data Management Environment	160
3.2.2	Robotics Environment	161
3.2.3	Control Processing Environment	163
3.2.4	Marine Environment	163
A	Relevant Papers	169
B	Knomad Reference Manual	235
B.1	Introduction	235
B.2	The Database Management System	236
B.2.1	Tuples and Views	236
B.2.2	Database Constraints	238
B.2.3	Facts and Frames	239
B.2.4	Assertions and Retrievals	242
B.2.5	Locks	243
B.2.6	Initialization	243
B.3	The Distributed Database	244
B.3.1	Using the Distributed Database	244
B.3.2	Locking and Unlocking Concepts	245
B.3.3	Error Checking	246
B.4	KNOMAD Tools	247
B.4.1	The Rule Management System	247
B.4.2	The Backward Chaining System	257
B.4.3	The Constraint System	263
B.5	Miscellaneous Utilities	273
B.5.1	Queues	273
B.5.2	Events	274
B.5.3	Semaphores	276
B.5.4	Clocks	277
B.5.5	The KNOMAD Unifier	278
B.6	Object-Oriented Programming in KNOMAD	278
B.6.1	Objects and Inheritance	278

B.6.2	Methods	279
B.7	KNOMAD BNF Syntax	280
B.7.1	Definitions	280
B.7.2	Rule Management System	280
B.7.3	Frames	282
B.7.4	Database Assertions	282
B.7.5	Integrity Constraints	282
B.8	Glossary	283
C	MAESTRO Internals Reference Manual	285
C.1	Introduction	285
C.2	MAESTRO Data Objects	286
C.3	Schedule Generation	287
C.4	Non-Real-Time Contingency Handling	287
C.5	Real-Time Contingency Handling	289
D	LLP / FRAMES Interface Control Document	291
E	Bugs and Their Disposition	313
F	SSM/PMAD Test Plan	355
G	SSM/PMAD User Manual	499

(This Page Intentionally Left Blank)

List of Figures

1.1	The SSM/PMAD System Block Diagram	3
1.2	The Previous SSM/PMAD Automation Structure	4
1.3	The Present SSM/PMAD Automation Structure	5
2.1	SSM/PMAD User Interface Screen Layout	14
2.2	SSM/PMAD System Flow Window	17
2.3	SSM/PMAD Workbox	18
2.4	SSM/PMAD User Interface Single Selection Menu	19
2.5	SSM/PMAD User Interface Multiple Choice Menu	20
2.6	The Previous SSM/PMAD Automation Structure	22
2.7	The Present SSM/PMAD Automation Structure	23
2.8	The SSM/PMAD System Block Diagram	24
2.9	SSM/PMAD Software Flow	26
2.10	SSM/PMAD Control and Timing Flow	27
2.11	The Ordering of Priorities by Hierarchical Categories	41
2.12	SSM/PMAD Test Bed Autonomous Operational Flow	43
2.13	A User's View Into The Autonomously Executing System	45
2.14	The KANT Planning Activity Interface	46
2.15	SSM/PMAD Power to Loads	47
2.16	SSM/PMAD Distributed Control and Process Management	49
2.17	The SSM/PMAD Process Flow with KANT	52
2.18	KANT Interface Relations	53
2.19	SSM/PMAD User Interface Layout	90
2.20	Activity Editor Window	97
2.21	Activity Editor Status Line Category I	98
2.22	Activity Editor Status Line Category II	98
2.23	Activity Editor Status Line Category III	99
2.24	Activity Editor Edit Menu	99
2.25	Activity Editor Activity Pull Right	101
2.26	Activity Editor Expansion of a Subtask	103
2.27	Activity Editor Subtask Options Menu	104

2.28	Activity Editor Subtask Options Menu	105
2.29	Activity Editor Availability Edit Workbox	108
2.30	Activity Editor Updated Availability List	110
2.31	Activity Editor Scheduling Menu	112
2.32	Activity Editor Scheduling Changes	113
2.33	Schedule Changes due to Delete	115
2.34	Delete Subtask Pull Right Menu	116
2.35	Subtask Deletion Information	117
2.36	SSM/PMAD User Environment ILA Representation	120
2.37	System Overview	132
2.38	Switch Grouping	134
2.39	Seizure Select Menu	136
2.40	ILA Seizure Information Menu	139
2.41	Updated ILA Seizure Information Menu	140
2.42	Switch Seizure Workbox	141
2.43	Switch Description Lines	144
2.44	Switch Description Line Reference	145
2.45	Add Menu with Load Center Pull Right	147
2.46	Manual Release Menu	148
2.47	Present <i>SSM/PMAD</i> Test Bed Power Management Process	151
2.48	Power Management Process Allocating Unused Power	152
2.49	The <i>SSM/PMAD</i> Testbed Fault Type Hierarchy	153
2.50	The <i>SSM/PMAD</i> User Interface Screen Layout	155
2.51	The Management and Control Allocation Constraints	157
3.1	Data Management Environment	160
3.2	Robot A Environment	162
3.3	Robot B Environment	162
3.4	Marine Environment	163

Acronyms

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
COTR	Contracting Office Technical Representative
DBMS	Database Management 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
ILA	Intermediate Levels of Autonomy
JSC	Johnson Space Center
KANT	Knowledge Augmentation and Negotiation Tool.
KBMS	Knowledge Based Management System
KHz	KiloHertz
KVA	KiloVolt Amps
L-R	Inductive-Resistive
LC	Load Center
LES	Load Enable Scheduler
LISP	List processing computer language
LLF	Lowest Level Function
LLP	Lowest Level Processor
LPL	Load Priority List
LPLMS	Load Priority List Management System
MAESTRO	Scheduling system ("MAESTRO" is not an acronym)
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	Transmission Control Protocol / Internet Protocol
UI	User Interface
VCLR	Visual Control Logic Representation

Chapter 1

Introduction

1.1 Purpose

The Space Station Module Power Management and Distribution (SSM/PMAD) testbed located at the NASA George C. Marshall Space Flight Center has undergone significant enhancement over the past two years. The purpose of this Final Report is to provide an overview of the SSM/PMAD system and to provide detailed information on these enhancements.

1.2 History

The Technical Proposal for the SSM/PMAD contract was submitted in May, 1985. Since that time there have been several major software, hardware and documentation deliveries made to NASA. The first of these were the deliverables for the four original contract tasks. Task I was the Common Module power management and distribution system automation plan definition. That study report was delivered in July, 1986. Task II was the definition of hardware and software elements of automation. Task III was the design, implementation and delivery of the hardware and software making up the original SSM/PMAD system. These two tasks were reported on in Volume I of the Interim Final Report (IFR), delivered in February, 1989. Task IV was the definition and development of the MSFC host breadboard computer environment. That task was completed early in the contract performance, being a prerequisite to the other tasks.

Prior to delivery of the IFR Volume I, a contract extension was negotiated between Martin Marietta and NASA, resulting in follow-on work performed during 1989 and 1990. Changes to the system included:

- 20 kHz, 400 Vac Ring Bus changed to 120 Vdc Star Bus configuration

- Rehosting of FELES and FRAMES from Xerox 1186 to Solbourne 5/501 workstation
- New Lowest Level Processor (LLP) hardware (Intel 80386)
- Use of UNIX instead of Xerox operating system on workstation
- New communications protocol and hardware (ethernet TCP/IP)
- New integrated database/knowledge management system (KNOMAD)
- Rule reorganization in FRAMES
- Handling of multiple faults
- Significant improvement in Lowest Level Function transactions and fault management
- Significant improvement in the User Interface
- Intermediate Levels of Autonomy enhancements
- Faulted switch management

The Draft Volume III Interim Final Report, MCR-92-1309 was delivered in February, 1992. The basic operating structure and architecture of the testbed did not change significantly after the June 1990 delivery. The system block diagram shown in Figure 1.1 is the same as the one reported in Volume II of the IFR except for the addition of software elements to the Solbourne workstation and the removal of software elements from the previously employed Symbolics workstation.

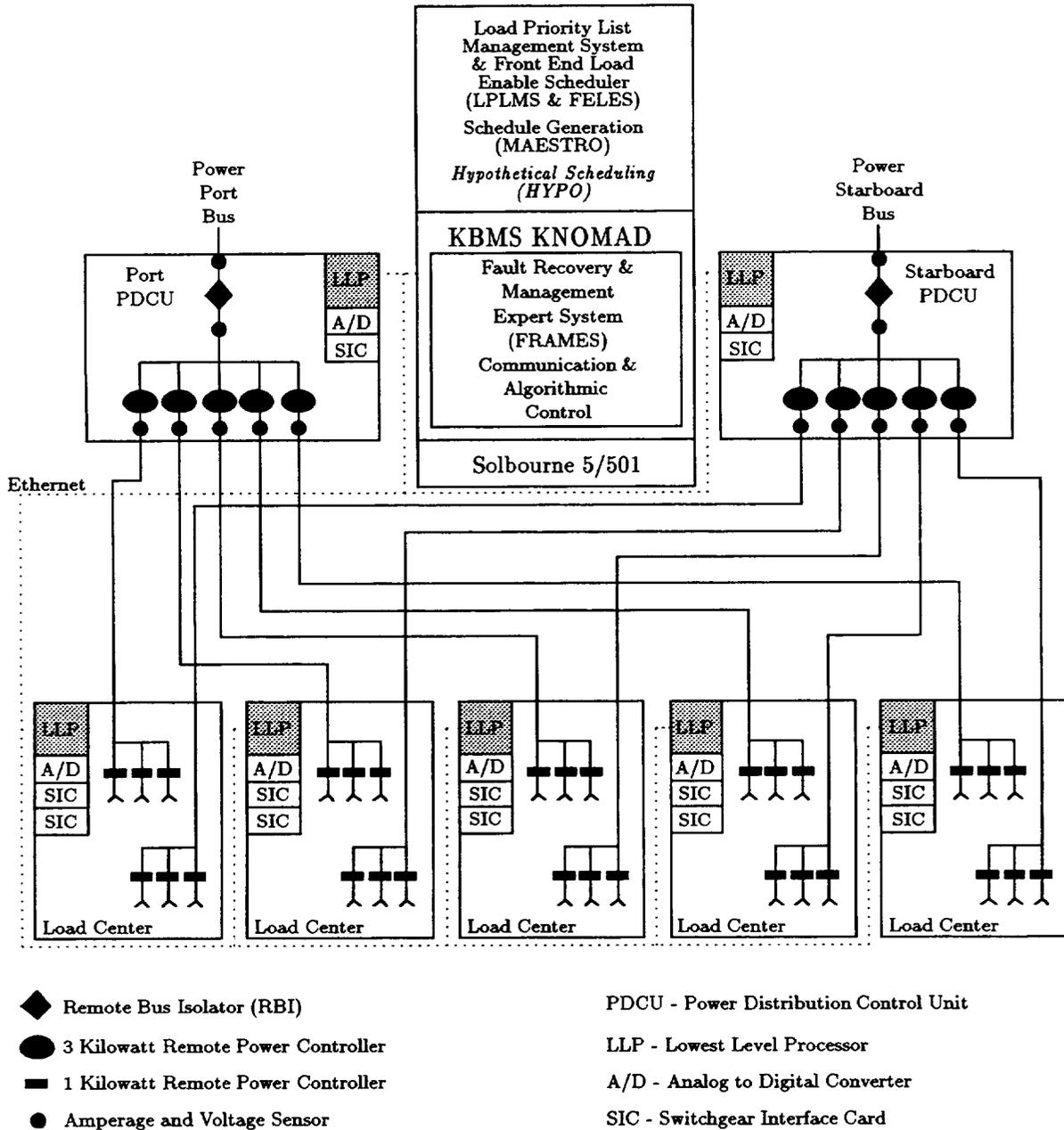


Figure 1.1: The SSM/PMAD System Block Diagram

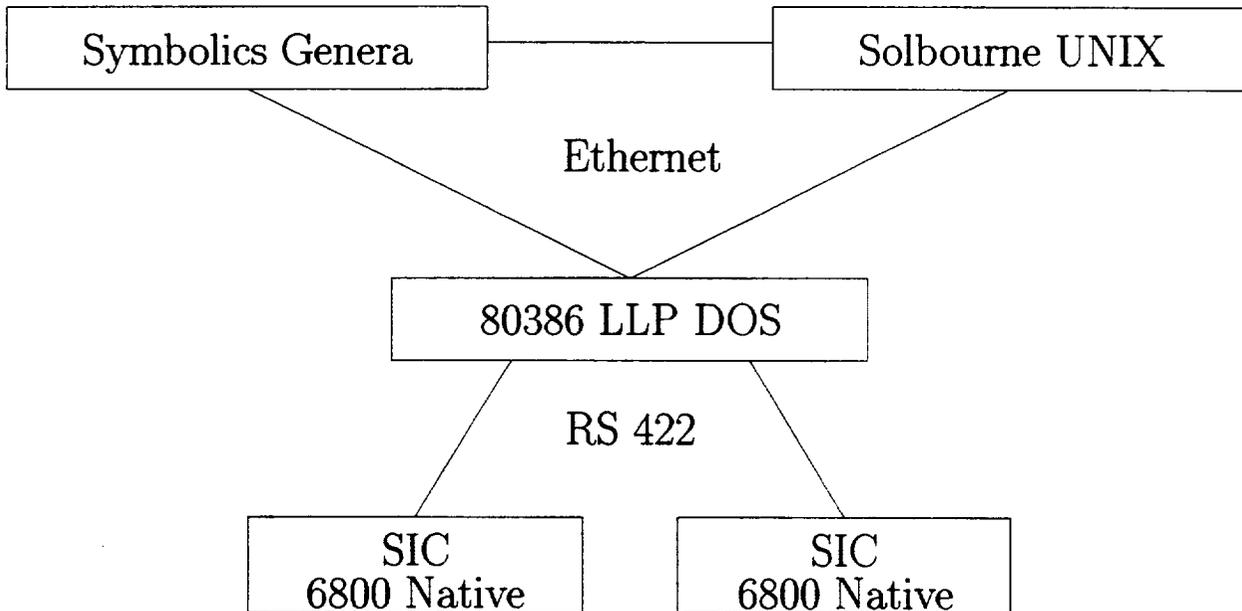


Figure 1.2: The Previous SSM/PMAD Automation Structure

The architecture of the SSM/PMAD test bed automation system has not changed significantly. The operating process flow is as documented in Volume II except that the MAESTRO scheduling system was rehosted from a Symbolics workstation to the Solbourne workstation. This was done in order to provide a more robust interface between the scheduler, the SSM/PMAD database, and the newly created activity editor. The previous machine architecture is shown in Figure 1.2, and Figure 1.3 depicts the new architecture.

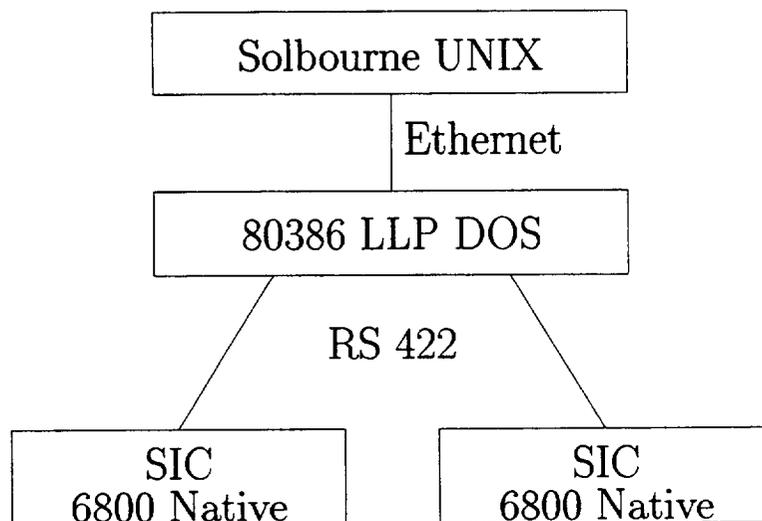


Figure 1.3: The Present SSM/PMAD Automation Structure

1.3 Summary

This volume will only cover those additions or enhancements to the SSM/PMAD test bed that have occurred since July 1990. Significant changes to be reported are:

- Rehosting the MAESTRO scheduler to the Solbourne Workstation.
- Reorganization of the automation software internals.
- A more robust communications package.
- The Activity Editor to the MAESTRO scheduler.
- Rehosting the LPLMS to execute under KNOMAD.
- Implementation of the Intermediate Levels of Autonomy.
- Completion of the KNOMAD knowledge management facility.
- Significant improvement of the User Interface.
- Soft and Incipient fault handling design.
- Intermediate Levels of Autonomy.
- Switch maintenance.

Chapter 1 of this report is this introduction. Chapter 2 is a system description, and details the requirements, top-level configuration, and hardware and software of the SSM/PMAD system. Chapter 3 provides an overview of the operational theory behind the system. Chapter 4 lists the technical achievements that were accomplished to make the system a reality. Chapter 5 is a description of the problems discovered while using the previously delivered versions of the system, with an indication for each of its eventual resolution. Chapter 6 suggests future directions for the SSM/PMAD system, both in the context of the current testbed and with respect to application of the various components of the system to other technology domains. Following Chapter 6 is a list of references where additional information about the technologies used may be found, and a glossary of terms used in the system. Appendix A is a listing of Conference and Journal articles about SSM/PMAD that have been published since July of 1990. Appendix B is the Knowledge Management and Decision (KNOMAD) system reference manual. Appendix C is the MAESTRO Internals Reference Manual. Appendix D is the updated Interface Control Document for the SSM/PMAD system. Appendix E is the SSM/PMAD Test Plan. Appendix F is the final version of the SSM/PMAD Users' Manual.

Chapter 2

System Description

2.1 Overview

This chapter describes the SSM/PMAD system at an intermediate level. Section 2.2 describes the requirements that drive the SSM/PMAD design. Section 2.3 describes the top-level configuration of the system, including the various functional entities and their relationships. Section 2.4 describes the current suite of power system hardware that makes up the SSM/PMAD testbed, and the computational hardware that controls the system. Section 2.5 contains cursory descriptions of each of the major software entities that make up SSM/PMAD. For more detail on these software systems, refer to the appendices, which include published papers about the systems and reference manuals for KNOMAD and MAE-STRO.

2.2 Requirements

2.2.1 Background

A goal of the Space Station Program was to accommodate increasing levels of automation that are time-phased and consistent with evolving systems and operational requirements, costs, and technology advancement. The requirements that drive the design of the SSM/PMAD system were originally specified in the Space Station C-4 specification, Space Station System Requirements, and have undergone considerable evolution since their initial definition. Those initial Space Station requirements are listed below.

- Accept power from independent, redundant utility buses.
- Protect utility and module power buses from user load faults.
- A single open circuit shall not cause the loss of the utility bus.
- All preliminary power circuits shall employ a single-point structure ground.
- Electrical power shall be transferred by internal, manual, modular connections.
- Functional redundancy shall be verifiable without ORU removal.
- Subsystem components shall be removable.
- Redundant components shall have diverse wiring.
- The subsystem shall be fail-safe/fail-operational and failures shall not propagate.
- The subsystem shall provide local power management and respond to utility commands.
- Provide automatic and on-demand self-test.
- Provide load status and monitoring.
- Provide autonomous redundancy management.
- Provide autonomous fault detection and fault isolation.
- Provide capability to accept primary power types, including (1) high-freq 3-phase ac; (2) low-freq (≤ 1 -kHz) single or multiphase ac at 150 Vac; and (3) ≥ 150 Vdc.
- Accommodate user, housekeeping and subsystem power needs.

- Accomodate power growth commensurate with Space Station growth from 75 to 300 kW.
- Provide IOC power capability of 50 kW.
- Provide capability for on-orbit maintainability through repair or replacement.
- Provide a design consistent with hardware technology readiness.
- Design transparency to Space Station energy conversion and storage technologies.
- Provide the capability to incorporate local energy storage for use as safe haven or uninterruptible power.

As the SSM/PMAD program was intended primarily as a technology demonstration, some of the above requirements did not apply to the breadboard. Others have since been superseded, e.g., on December 14, 1988, a NASA Change Request specifying 120 Vdc source power was put into effect. During successive SSM/PMAD design and implementation phases, deficiencies were noted and further modifications to the requirements were specified, sometimes via memos or just conversations with the NASA Contracting Office Technical Representative (COTR). Thus there is no complete set of requirements for the SSM/PMAD system. Rather, the system was designed to meet the goal stated at the top of this background section, with capability additions performed as feasible. In fact, the SSM/PMAD program has been considered research from its inception, therefore its primary requirement was to be a thorough exploration into the feasibility of meeting the Space Station requirements listed above. The individual phases of the contract have followed well-specified Statements Of Work, and these in combination with the above requirements and with discussions with the NASA COR have formed the basis for the system design.

With the above in mind, this section presents a set of high-level design goals for the SSM/PMAD system, some requirements for recent enhancements to the system, and a more detailed description for the user interface, whose functionality warrants that.

2.2.2 System Requirements

The following are high-level goals of the SSM/PMAD system design:

1. provide up to 3 kW of power to loads, powered redundantly
2. provide user with switch manipulation and breadboard status access capabilities
3. provide schedule generation capabilities, including
 - powered equipment creation and editing

- hierarchical activity definition and modification, i.e. activities and subtasks
 - schedule creation and modification
 - ability to save all objects created to a database
4. provide schedule execution capabilities, including
- download of saved schedule from database
 - transformation of schedule into power system events
 - commanding of switches according to list of power system events
 - user commanding of switches while system is executing a schedule
 - detection, isolation and work-around for power system faults
 - contingency rescheduling in response to power system faults
 - user schedule change while schedule is executing
 - prioritized load-shedding to handle power reductions
 - redundancy switching to continuously power high-priority loads
5. provide a graphical, menu-driven user interface that conforms to the more detailed specifications in the next subsection.

Communications and Algorithmic Software (CAS) Requirements

The various communications functions which exist are bundled to form the Communications and Algorithmic Software (CAS). The primary responsibilities of the CAS are to sort and deliver the Front End Load Enable Schedule into its appropriate sub-component representations for execution by the LLPs and to stage and deliver data between the Solbourne and the LLPs. It also contains manual mode operations interface. The CAS software will:

1. Provide a buffer between switchgear and the upper level processing.
2. Provide performance data calculations to the upper level process.
3. Receive schedules from upper level processes.
4. Receive priority lists from upper level processes.
5. Receive contingency lists from upper level processes.
6. Respond immediately (approximately 1 minute) to a source power level change.

Lowest Level Processor (LLP) Requirements

When a schedule is passed down through the various software components, pieces of it are made available to the appropriate LLPs for execution. The LLPs will command the switches to open and close according to their schedule pieces. The LLPs will also monitor the switchgear for adherence to schedule parameters and for hard fault events. When relevant data is available to the LLPs, they will make this data available to FRAMES on the Solbourne. This up and down transfer of data provides the components at each level with the information necessary for their operation. LLPs will monitor for evidence of soft faults. Positive evidence for these types of faults will be communicated to FRAMES. The LLPs will contain partial FRAMES knowledge at the lower level and will be capable of exercising this knowledge during fault diagnoses.

Fault Recovery and Management Expert System (FRAMES) Requirements

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

1. Diagnose faults (isolate and identify) within the scope of the available data
2. Recommend corrective recovery actions for a set of fault hypotheses to an operator at the user interface. This meets the following two requirements:
 - The capability to initiate such recommended corrective actions, subject to operator concurrence, through the breadboard power control unit.
 - Automatic corrective actions when possible.
3. Cooperate with the user interface, employing multiple windows as it diagnoses faults in the breadboard and determines appropriate corrective actions or suggestions.
4. These windows will use displays that are mouse sensitive and user friendly as defined by windowing techniques and the use of menus. The displays will indicate faults and corrective or suggested actions on graphical schematic displays.

Knowledge Base Management Requirements

The knowledge base management system will control the execution of FRAMES, and provide notification of events from external sources which require action by the Front End Load Enable Scheduler (FELES) or Load Priority List Management System (LPLMS). The knowledge management activity will be part of a larger system, KNOMAD, which will manage multiple knowledge agents and their associated data bases. The rules in FRAMES and LPLMS will be organized into modular groups.

Knowledge Agent Interface Requirements

The knowledge agent interface will exist on the Solbourne general purpose workstation. It will consist of elements of the Knowledge Management and Design Environment (KNOMAD), the FELES, FRAMES, and the LPLMS; and its user interface access will be managed as a knowledge element of the FELES user interface knowledge agent. The MAESTRO user interface will exist in an operating environment on the Solbourne workstation.

The FELES, managed by KNOMAD, will provide the user access to the scheduling environment, MAESTRO, through KANT, and will manage returning information from FRAMES. Whenever run-time rescheduling activities are required, MAESTRO causes KANT to initiate FELES and LPLMS activity with the appropriate update information.

The 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 and prevent shedding of critical or higher priority loads. LPLMS accomplishes this need function.

MAESTRO is an activity scheduling function. It contains a model of the overall power system and the heuristics required to ensure correct allocation of resources. The result of MAESTRO's work is the production of a schedule allocated by FELES to be carried out by the LLPs.

User Interface Requirements Update

Mission operations for the SSM/PMAD testbed are unique in that requirements for its operation have not been established from the standpoint of a using project. However, the testbed may be used in a peripheral role for missions such as SSF, STV, CTV, EOS, and others. No direct or derived requirements for peripheral operations have been specified.

Layout The SSM/PMAD User Interface shall have an application window with a corresponding application menu and scratchpad window for secondary information. The User Interface shall also have a status line window, a "NASA - MARSHALL SPACE FLIGHT

CENTER" window, a mode indicator window, a clock bar window, an applications selection window, a system flow window, and four fault diagnosis and detection windows. The layout of these windows is shown below in Figure 2.1.

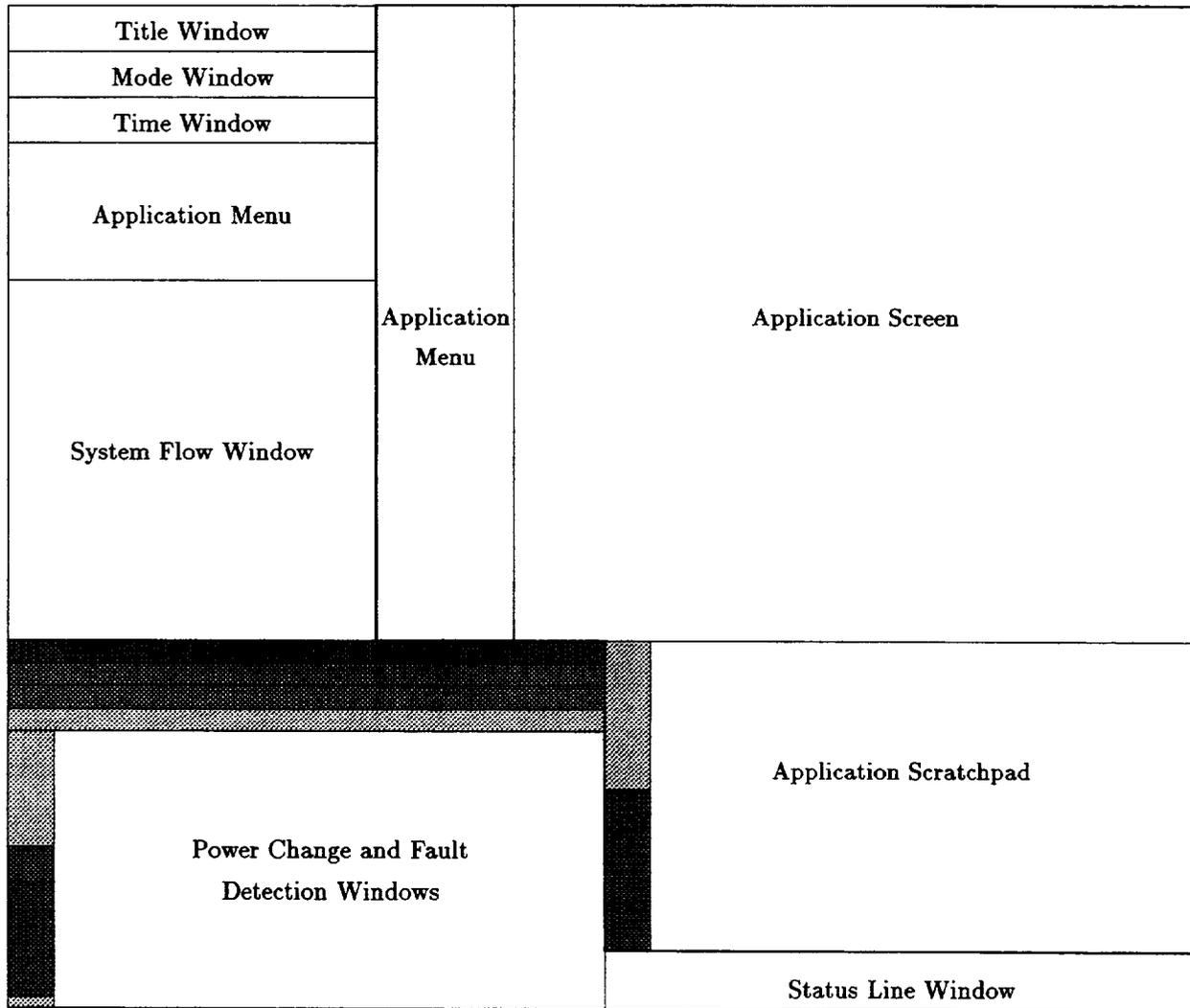


Figure 2.1: SSM/PMAD User Interface Screen Layout

Window Requirements

Application Window, Menu, and Scratchpad These windows as a group comprise an application screen. The SSM/PMAD User Interface shall have several application screens which may be accessed through the applications selection menu. Only one application screen may be visible on the SSM/PMAD User Interface at a given time. For this reason, application screens are dynamic and may be switched from one to another as a user sees fit. The application menu shall contain mouse button selected functionality which supports the application. These mouse sensitive areas shall be grayed out if the functionality is not available. The first item in the application menu shall always be the access to the system options menu. The application menu shall always contain a help button for more information on the application. Ancillary information resulting from application menu button functionality shall be displayed in the application scratchpad window. The application scratchpad window shall be a scrolling window with several windows worth of scrolling area.

Status Line Window This window shall supply the user information about which mouse button to press anytime the mouse cursor enters a mouse sensitive region, with the exception of areas internal to Pop Up Menus, Workboxes, and Scroll Bars. This window shall always be visible.

Title Window This window shall display the words "NASA - MARSHALL SPACE FLIGHT CENTER". This window shall always be visible.

Mode Window This window shall display the words "IDLE", "NORMAL", or "MAINTENANCE" based upon which mode the SSM/PMAD System is currently operating. This window shall always be visible.

Time Window This window shall display both Greenwich Mean Time abbreviated GMT and SSM/PMAD mission time. These times shall be displayed in a dd.hh:mm format where dd is day number, hh is hour, and mm is minute. These times shall be updated on a minute basis. This window shall always be visible.

Application Menu This window shall contain a list of applications which shall be mouse sensitive if available and grayed out if not available. Selection of an application (with a mouse click) will make visible the selected application screen and make invisible the presently active application screen. This window shall always be visible.

System Flow Window This window shall contain a picture of the software components operating in the SSM/PMAD system. (See Figure 2.2) The software components that must be displayed are MAESTRO, FELES, LPLMS, FRAMES, KANT, and User Interface. This window shall display interaction within the SSM/PMAD system's software components by highlighting the component and displaying a brief message about what is happening. The System Flow window shall display communication connections between software components not resident on the same machine as the user interface. The non-resident components with connectivity which must be displayed are APEX, and the Lowest Level Processors (LLPs). The APEX component box will be mouse sensitive and allow the user to attempt to connect or disconnect with/from the APEX system with a simple mouse click. This feature shall be available in all modes of operation of the SSM/PMAD system. The LLP component boxes shall also be mouse sensitive allowing the user to connect or disconnect with/from the LLP with a mouse click, but only in maintenance mode of operation. The system flow window shall contain four mouse sensitive boxes, hard faults, soft faults, incipient faults, and power changes. When a box is selected, it will make visible the appropriate fault diagnosis and detection window by placing it on top of the others. This window shall always be visible.

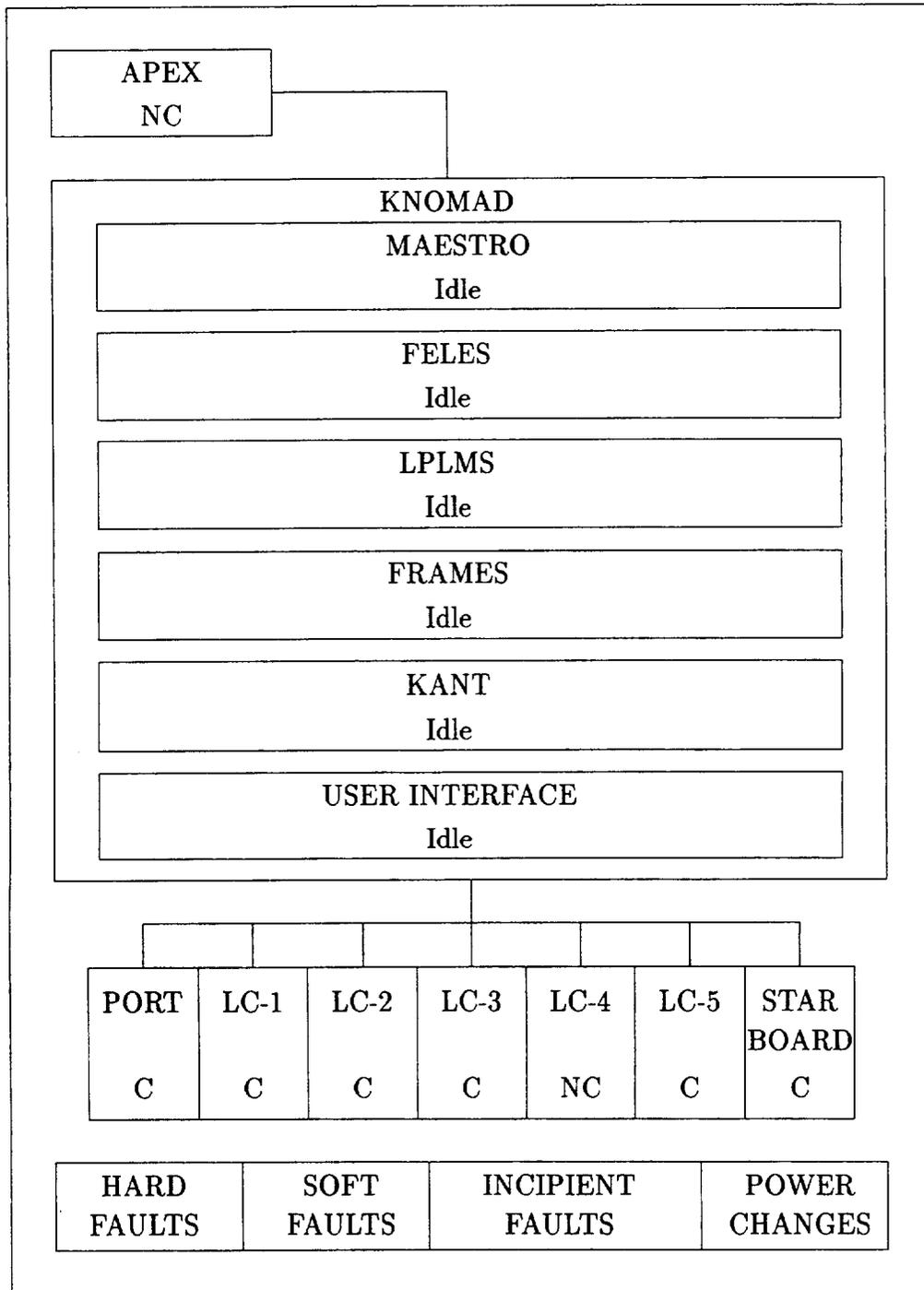


Figure 2.2: SSM/PMAD System Flow Window

Power Change and Fault Detection Windows The four fault diagnosis and detection windows are “Hard Fault Diagnosis”, “Soft Fault Diagnosis”, “Incipient Fault Detection”, and “Source Power Changes”. Each of these windows shall be a scrolling window with at least two window’s worth of scrollability. Each of these windows shall occupy the same space at the lower left hand corner of the User Interface Screen. Each of these windows shall have its title bar visible. Each title bar shall be mouse selectable with the selection moving the selected window to the top layer thereby making its scrolling window visible. Each title bar shall also list the number of hard fault diagnoses, soft fault diagnoses, incipient fault detections, and source power changes discovered since the beginning of the mission (mission time 0). As a feature, a fault diagnosis and detection window being written to will automatically rotate to the top. This feature will be capable of being disabled/enabled from the system options pop up menu.

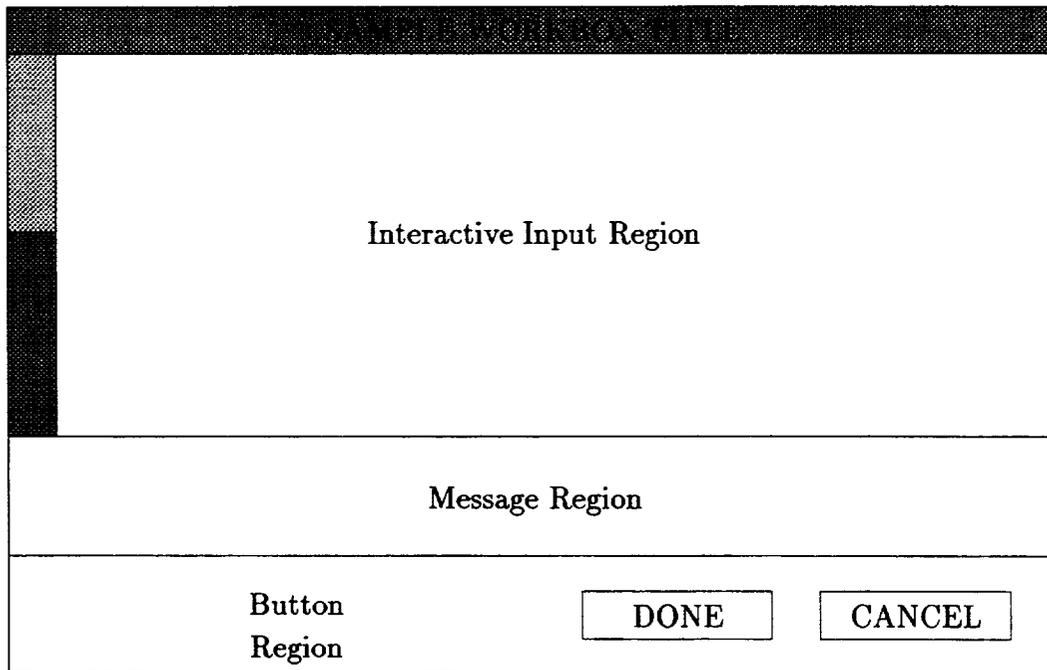


Figure 2.3: SSM/PMAD Workbox

Workboxes Workboxes shall be created for user input. A workbox shall be broken down into three regions. The first region shall be a scrolling graphics window in which the the user may interactively enter data. The second region will be below the scrolling window region

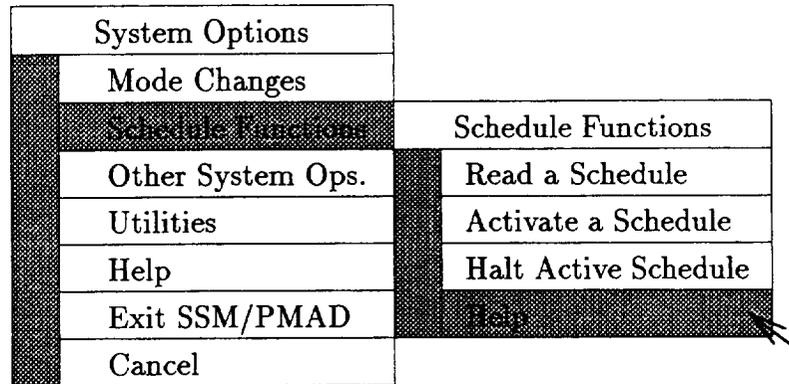


Figure 2.4: SSM/PMAD User Interface Single Selection Menu

and used for displaying messages to the user. The third region will contain all appropriate buttons for the workbox. All workboxes must have done and cancel buttons. Any button click outside the workbox shall be interpreted as a button waveoff which has the same effect as mouse clicking on the cancel button for the workbox. A sample workbox is displayed in Figure 2.3.

Pop Up Menus Pop up menus shall come in two types, single selection and multiple choice. Both menu types shall have the capability to scroll and contain pull-right sub-menus. A pull-right sub-menu shall be displayed upon entering its selection area in the parent menu. Likewise a pull-right sub-menu shall be removed upon exiting its selection region in the parent menu. Like the workbox, a button click outside the confines of the pop up menu is interpreted as a button waveoff of the pop up menu. A button waveoff of a pop up menu shall be treated as though the menu had never been popped up and the menu is removed. A single selection pop up menu shall return a list of return values corresponding to the menu hierarchy followed to reach the selection. A single selection shall return a cancel list in the event of a button waveoff. A multiple choice menu shall return a list of chosen values in the menu hierarchy and a list of return values corresponding to the menu hierarchy followed to reach the selection. Examples of both menu types are shown in Figure 2.4 and Figure 2.5.

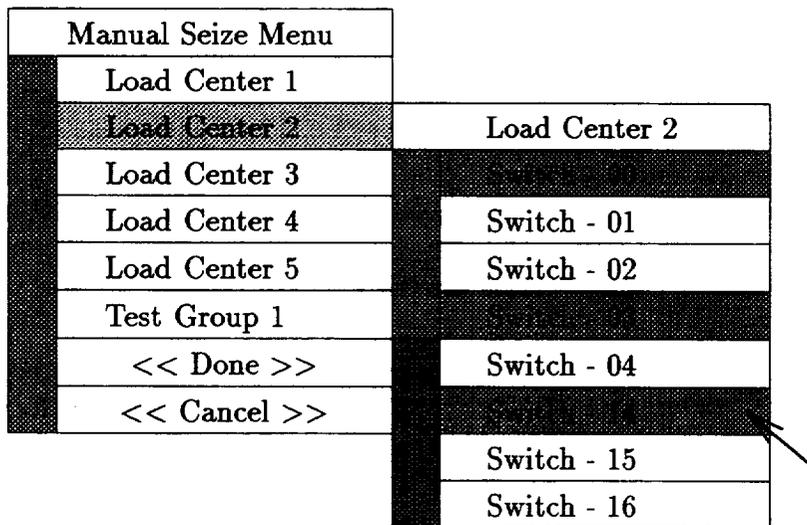


Figure 2.5: SSM/PMAD User Interface Multiple Choice Menu

2.3 Top Level Configuration

The power system consists of up to eight LLP's. Each LLP contains a computer controlling at least one Switchgear Interface Card (SIC). Each SIC card has the capability of interfacing the computer to an Analog to Digital (A/D) Card and Generic Controller (GC) cards, which control the switches. The LLP's are interfaced via ethernet to the Solbourne 5/501 workstation, performs all higher-level SSM/PMAD functions. The software suite running on the Solbourne includes the Communications and Algorithmic Control (CAC), the Fault Recovery and Management Expert System (FRAMES), the Knowledge Management and Decision (KNOMAD) system, the Knowledge Augmentation and Negotiation Tool (KANT), the MAESTRO scheduling system, the Hypothetical (HYPO) scheduler, the Load Priority List Maintenance System (LPLMS), the Front-End Load Enable Scheduler (FELES) and the User Interface (UI).

The previous machine architecture is shown in Figure 2.6, and Figure 2.7 depicts the new architecture.

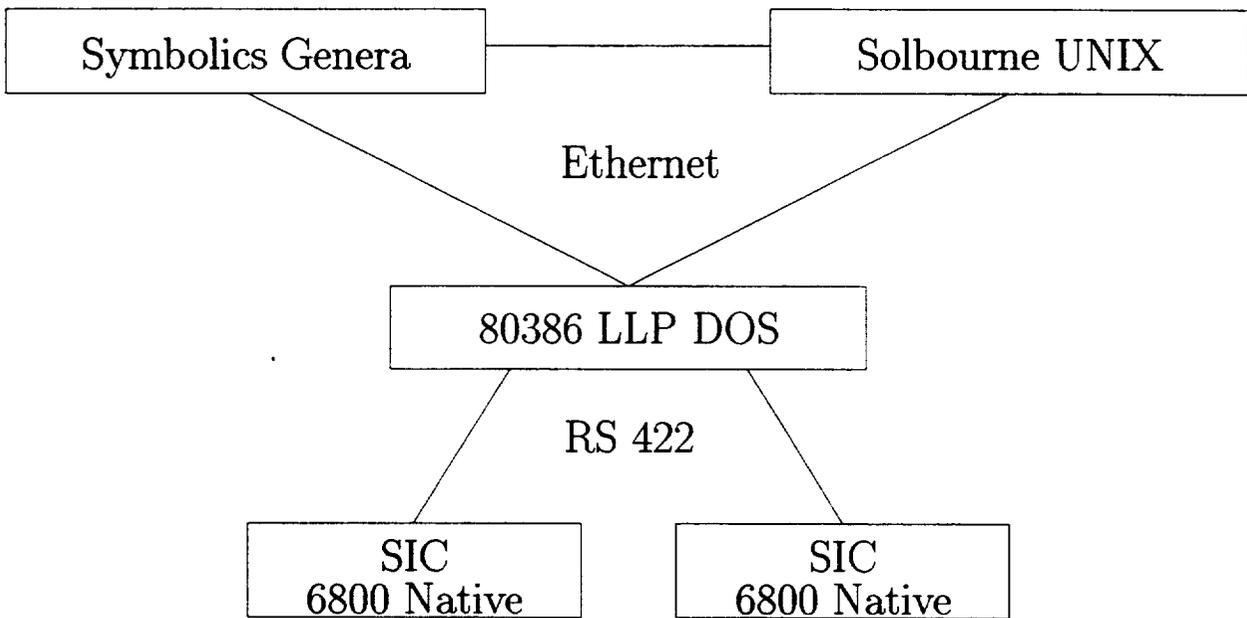


Figure 2.6: The Previous SSM/PMAD Automation Structure

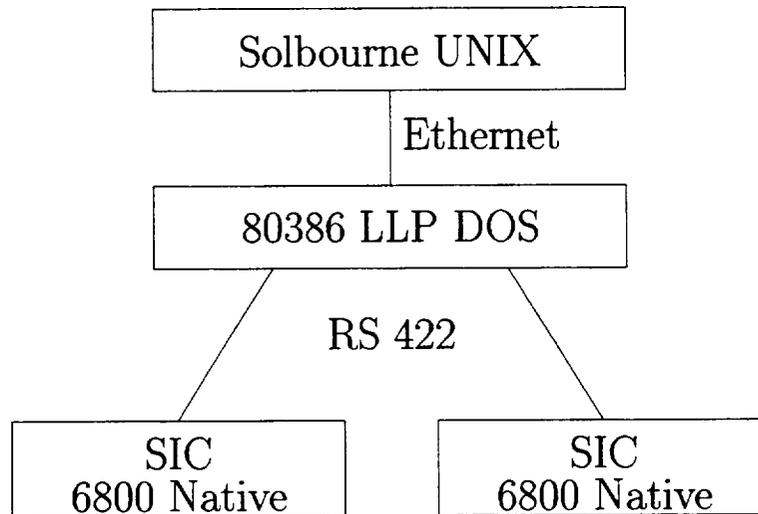


Figure 2.7: The Present SSM/PMAD Automation Structure

The system block diagram shown in Figure 2.8 is the same as the one reported in Volume II of the IFR except for the addition of software elements to the Solbourne workstation and the removal of software elements from the previously employed Symbolics workstation.

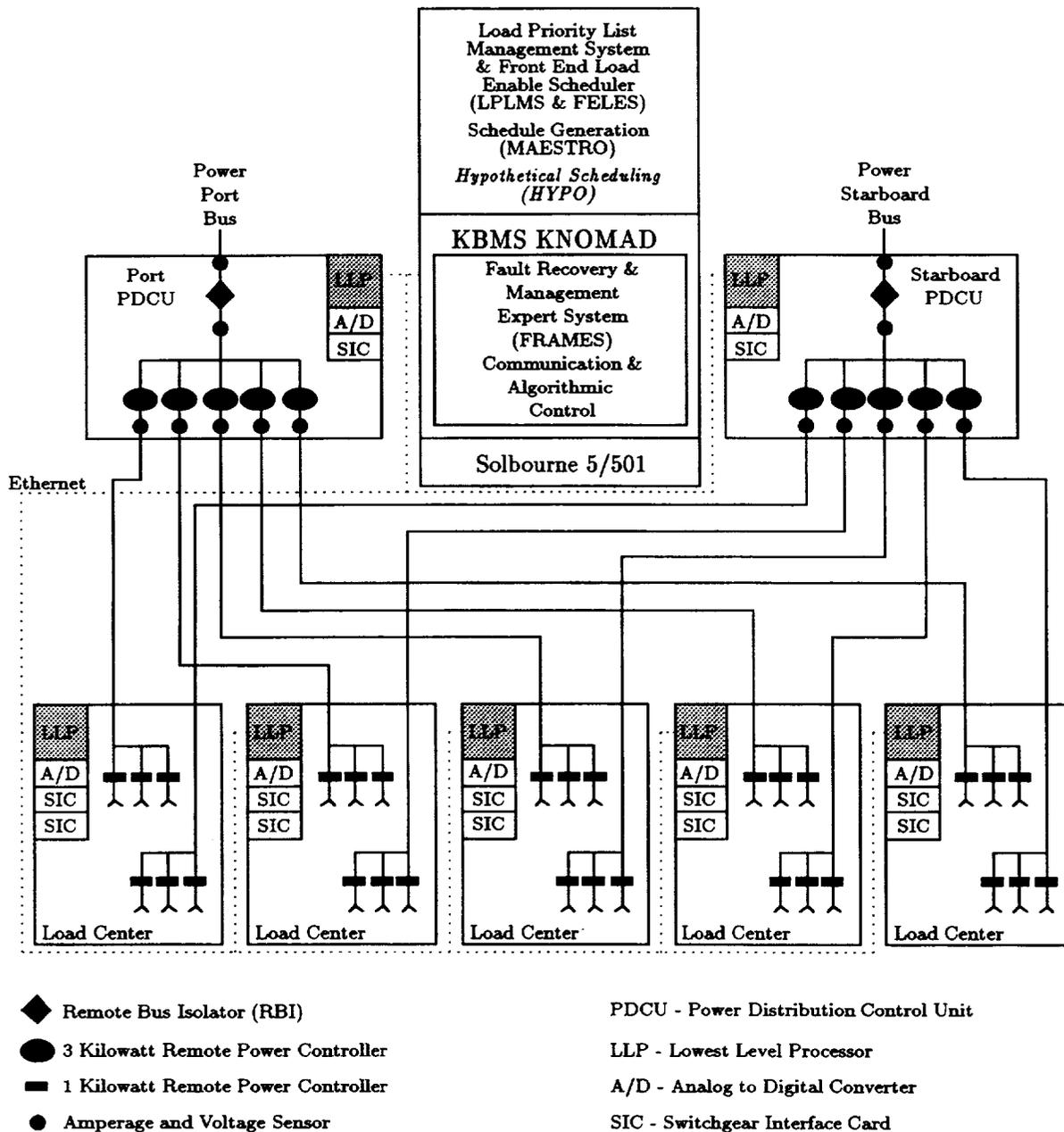


Figure 2.8: The SSM/PMAD System Block Diagram

2.4 Operational Theory

This section provides a high-level description of how the SSM/PMAD system works. For more detail on each component of the system, see Section 2.6 and the appendices.

2.4.1 Overview

The SSM/PMAD system takes user input to plan the activities that are to be performed autonomously. A schedule is built for the requested activities and passed to the lowest level processors for implementation. The system then monitors the power system for anomalous events. When an anomalous event occurs, the system safes itself, and checks the immediate local power system status. Once this has been accomplished, the SSM/PMAD system goes about isolating the fault or faults. Once the faults have been found, they are ordered and grouped for diagnosis. A determination is made for the cause(s) of each fault. The SSM/PMAD system then has MAESTRO reschedule around the contingency, and the updated schedule is passed to the LLPs. SSM/PMAD functioning is shown in figure 2.9.

2.4.2 Function Partitioning in Brief

The SSM/PMAD testbed was designed to make control decisions as fast as possible. The functionality in the system has been partitioned out to take advantage of processing capability. (See Figure 2.10.) Fault protection which requires a quick response is handled in the power system hardware. Fault detection and reporting is an LLP function. Fault diagnosis is a function of FRAMES on the workstation. Scheduling which requires the most time to accomplish is farthest from the actual hardware. By partitioning the testbed functionality in this way, the SSM/PMAD makes best use of its computational resources.

The SSM/PMAD system supports initialization, activity, resource and schedule creation and modification, normal schedule execution, fault handling, and two forms of Intermediate Levels of Autonomy (ILA) - user schedule change and manual control of portions of the testbed. Each of these will be described below.

2.4.3 Initialization

Initialization involves turning on power to the breadboard and the computing hardware, then loading the appropriate software systems. The mission clock is set to 0:00 at this point. Communications between the Solbourne and the LLPs are established. The system is placed in IDLE mode, and is ready for user control.

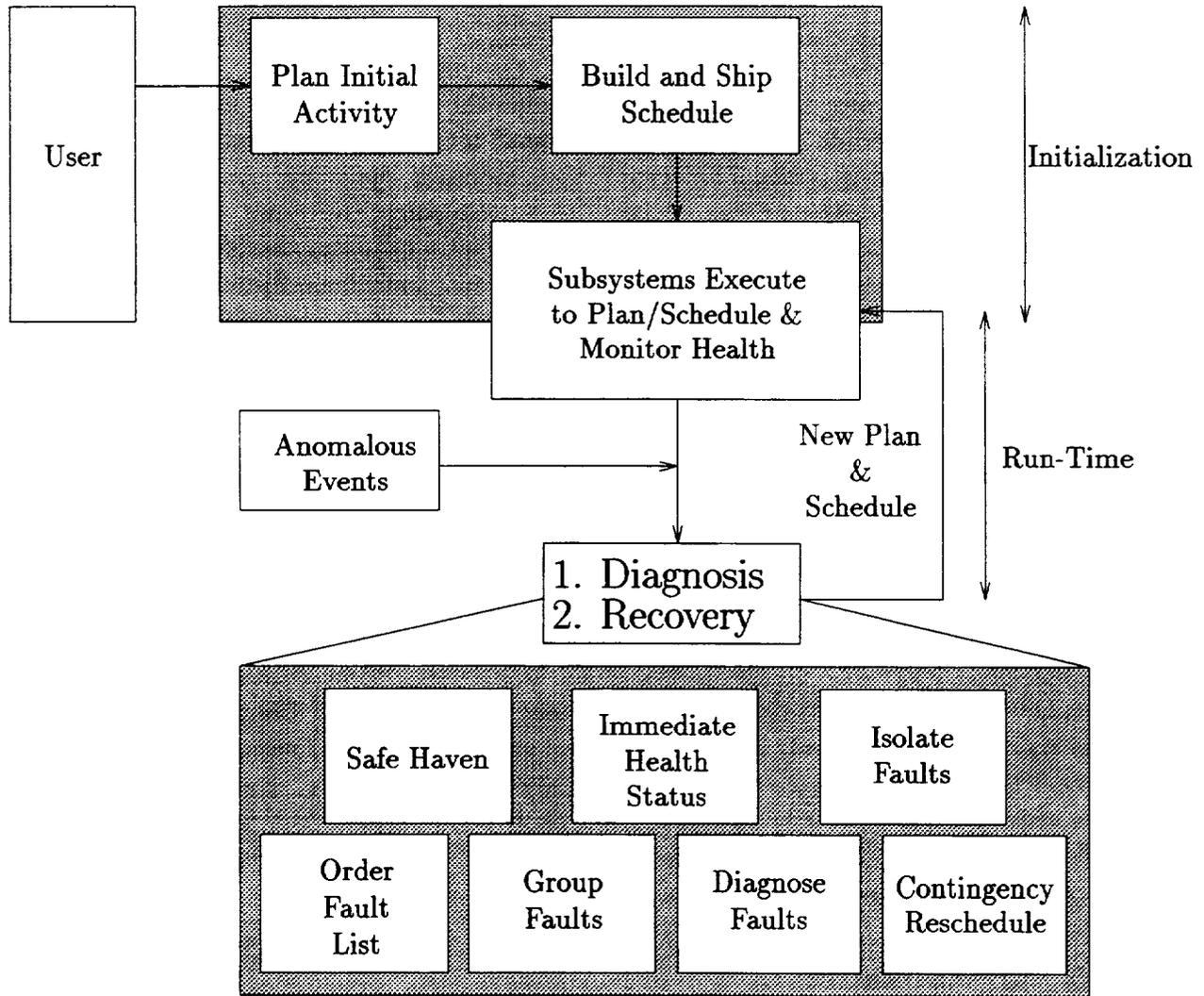


Figure 2.9: SSM/PMAD Software Flow

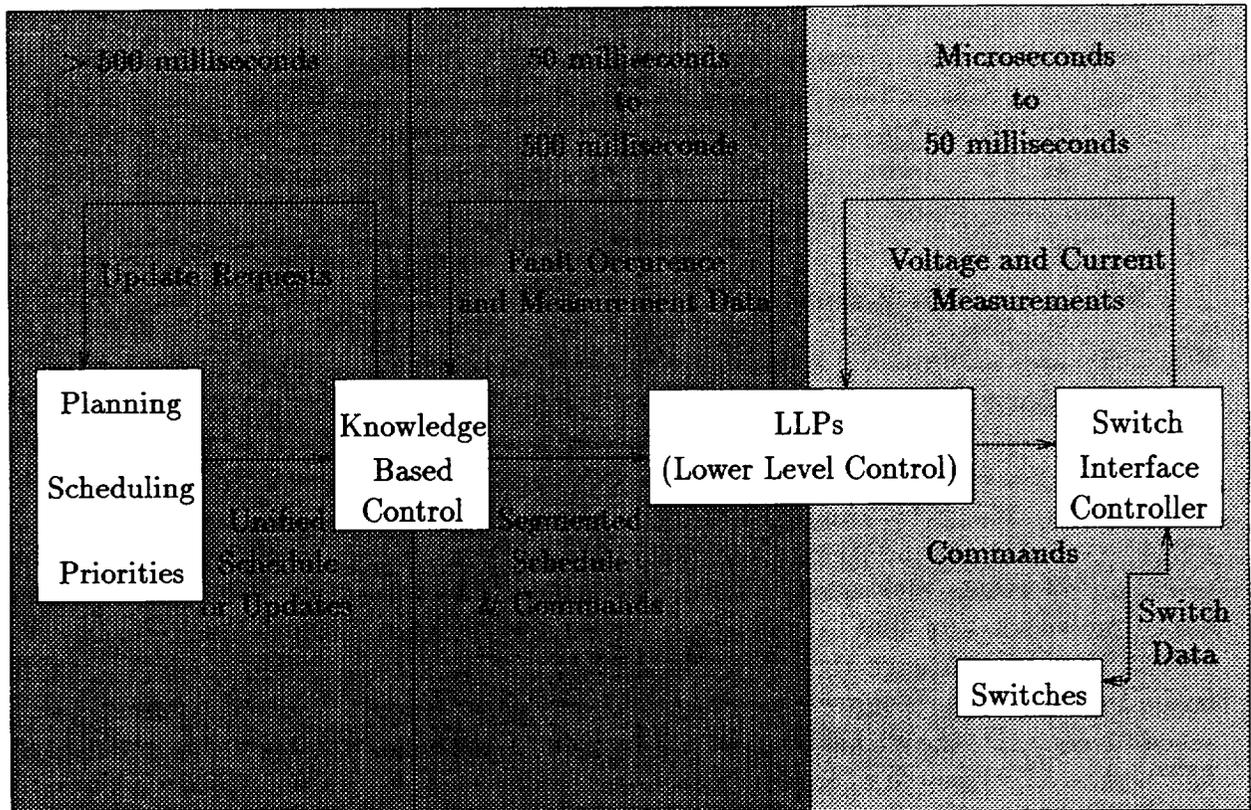


Figure 2.10: SSM/PMAD Control and Timing Flow

2.4.4 Schedule Creation

Prior to generating a schedule, the user can create new of activities to be scheduled, or can use any of the predefined activities. New resources and equipment may also be created. These objects are created by allowing the user to fill out data fields for the edited object and inserting that data entered into a frame managed by KNOMAD.

When the user selects a set of activities for scheduling and requests the creation of a schedule, the UI calls a method that causes MAESTRO to build a schedule, passing activity identification data to MAESTRO. MAESTRO first initializes its internal data structures, then performs a select-place-update cycle in which requested activities are evaluated and one is selected, that activity is placed on the schedule, and resource availabilities updated. When MAESTRO finishes schedule generation, control passes back to the UI, which displays the schedule in the appropriate window. The completed schedule is passed to KNOMAD through an assertion onto a frame.

2.4.5 Schedule Execution

When the user requests the activation of a saved schedule, the appropriate data structures are loaded, and a comparison is made of resources used by the schedule to resources available on the power system. If any power resources are faulted or otherwise unavailable, that availability change information is sent to MAESTRO along with a request to perform contingency rescheduling. MAESTRO updates the schedule and makes it available to KNOMAD. The FELES performs a translation of a segment of the schedule into a list of power system events (RPC on/off events with additional information). At the same time, the LPLMS generates a load priority list for that segment of the schedule. The event list and load priority list are sent to the LLPs through the CAS, and the LLPs begin executing the schedule by turning on and off switches, monitoring voltage and current levels, etc. Near the end of the first schedule segment, the FELES and LPLMS generate new lists, which are transmitted to the LLPs. The LLPs switch to execution of these lists when they reach the mission time specified for the new schedule segment. The user interface updates the display to indicate when scheduled events actually happen.

2.4.6 Fault Handling

Two types of faults are handled by the SSM/PMAD system. These are hard faults, such as short circuits on the breadboard, and soft or incipient faults, such as using a resource after its MTBF has passed. Hard fault handling is described first.

Handling Hard Faults

When a fault is detected by one or more of the LLPs, the affected LLPs immediately take whatever safing actions are possible given their level of authority and information. This involves possibly opening switches in for under-voltage or over-current, shedding loads for over-subscription on an RPC, and/or switching equipment to a redundant power source to continue a high-priority activity. The LLPs send their collected data along with an indication of the actions they took to FRAMES, which performs pattern matches against sets of rules to determine the most probable cause(s) of the faults. FRAMES may direct that other switches be opened or closed, either during fault diagnosis or after determining that a resource can be made available once more. When FRAMES has completed its diagnosis, it compiles a list of power system changes, and sends these to MAESTRO through KNOMAD. MAESTRO updates its representation of the running schedule by truncating activities whose subtasks were interrupted and changing resource representations for subtasks whose equipment was switched to a redundant source, then unchedules or interrupts other activities whose future resource requirements cannot be met, attempts to find ways to continue activities whose representations allow that, and finally attempts to schedule activities that can make use of resources released by other interrupted or unscheduled activities. MAESTRO sends the updated schedule back to KNOMAD, and the FELES and LPLMS generate a new event list and new load priority list, which are transmitted to the LLPs for execution. The user interface displays a list of faults, the diagnoses, indicates what components are performing what actions, and updates the power system and FELES displays to indicate the state of the power system and activity schedule.

Handling Soft Faults

The SSM/PMAD system monitors for power usage above assigned budgets but not above the capacities of the switches providing power to loads, and for resource readings that are out-of-balance. These soft faults are handled by notifying the user of their existence and allowing the user to fix the problems and/or, for current limit violations, clear the warnings from the screen. Both types of monitoring are performed by the LLPs, which send information through KNOMAD to the User Interface.

2.4.7 User Schedule Change

A new capability that has been implemented in SSM/PMAD is the ability for a user to modify a schedule during its execution. This is supported by the hypothetical scheduler, or HYPO. The hypothetical scheduler is so named because nothing the user does is more than a hypothesis until the user selects *IMPLEMENT* from the menu of possible actions. The HYPO system is described in more detail than the rest of the SSM/PMAD system because this is the first documentation of its operation.

When the user requests schedule modification, a hypothetical scheduling window is presented. The currently active schedule is written to a file and read into HYPO's internal data structures, and the user is allowed to specify when the schedule changes will be implemented. No schedule modifications will be allowed before that time, as they could not be implemented. Once the schedule is loaded into HYPO, the user has the option of using the activity and resource editors to create new scheduling objects, and/or to add activities to or delete them from the schedule. When the user chooses *ADD ACTIVITY* from the scheduling menu, the system presents the user with a list of activities from which to choose. The chosen activity's identifier is sent to HYPO, which creates scheduling structures for the activity and sends back a list of times when the first subtask could start. The user can specify a timepoint or time window in which to place the start of the activity, in which case the system will perform the same schedule manipulations that MAESTRO performs (e.g. using placement heuristics to add the activity, updating resource profiles and possibly bumping other, lower priority activities from the schedule). Alternatively, the user can request that the activity's priority be increased or decreased. An increase in priority (i.e., a lower number) will typically increase the number of placement options available to the activity, at the expense of other activities that may be bumped, while a decrease may reduce both the activity's placement options and the affects placing the activity will have on others. The user can also delete activities from the schedule, which will result in resources being released. All changes to the schedule are represented on an FELES-like display of the schedule.

When the user is satisfied and wishes to implement the schedule, a list of changes to the schedule (e.g., activity creations, additions, interruptions and deletions) are sent to KNOMAD, and the running schedule is replaced by the hypothetical schedule. The transition is handled the same way as a transition from one schedule segment to the next, resulting in a smooth hand-off. If the user decides not to implement the schedule modifications, he simply closes the HYPO window, and all data about the schedule change is lost. The user can start the schedule modification process over again, which resets the expected change completion time. This results in HYPO reading in the schedule again, and continuing as above. If the user does not implement the changes prior to the time chosen for implementation of the changes, the user must start the modification process again. If the power system experiences a hard fault while the HYPO system is running, the user must also start over.

2.4.8 Manual Control

The user can specify that one or more power system resources will be unavailable to the system for a period of time. This is known as manual seizure of breadboard resources. This functionality is supported by KANT, which allows the user to determine the effects of seizing a resource prior to his/her carrying out that action. When the user has determined which resources to preempt, these choices are communicated to the scheduler as RPC out-of-service and power availability change messages. MAESTRO treats them like

a contingency, rescheduling activities such that use of the chosen resources does not affect ongoing autonomous operations.

2.5 Hardware Description

2.5.1 Hardware Description

The SSM/PMAD testbed hardware may be broken down into two types of hardware. The first type is the actual power system hardware and the control electronics. The second type is the computation hardware platforms on which the software is based.

Power System Hardware

There are five types of power system hardware components which are used in the SSM/PMAD power system. These components are the card cage, the Switchgear Interface Controller (SIC), the Generic Controller (GC), the Remote Power Controller (RPC), and sensors. The functionality of each of these components will be described in the following paragraphs.

The Card Cage The Card Cage contains backplane slots which can hold up to nine GC cards and four SIC cards. Communication between a SIC card and a GC card occurs through a communications path in the backplane. The card cage provides the necessary capability to build a typical load center. The card cage provides the connectivity to control two separate power buses with each bus being controlled by a SIC card. The card cage further provides the capability to control each bus with a redundant SIC.

Switchgear Interface Controller The Switchgear Interface Controller (SIC) card provides the user with a series of commands which command the GC cards under its control. The SIC card is capable of addressing and controlling up to fourteen GC cards. This capability is limited to the first nine address locations by the card cage. The SIC card is interfaced to the Lowest Level Processor (LLP) via an optically isolated RS-422 communication link. Functionally, the SIC card is instructed to command the GC cards to turn on and off their RPC by the LLP. The LLP can also request RPC data from the SIC card.

Generic Controller The Generic Controller (GC) is used to control one Remote Power Controller (RPC). The GC will shut off the RPC when any of the following fault conditions are detected:

1. Under Voltage
2. Surge Current
3. Over Temperature
4. Over Current (I^2t)

The GC card also converts the analog current signal to a digital representation. This data is used to monitor the amperage passing through RPC being controlled.

Remote Power Controller The Remote Power Controller (RPC) is the actual power switching mechanism. The RPCs come in two types, 3 kilowatt RPCs and 1 kilowatt RPCs. The RPCs are solid state 120 volt DC switches. The RPCs designed to shut off with a Fast Trip Fault if the output of the switch is directly shorted. Functionally, the RPC contains a current sensor and the analog signal from that sensor is sent to the GC card. The RPC is connected to the GC card which commands it by a 37 pin cable which connects to the card cage and RPC directly.

Sensors The SSM/PMAD testbed also contains a number of independent sensors. The sensors come in two types, current and voltage. These sensors are placed at various points in the SSM/PMAD testbed to give a user and the FRAMES expert system additional information about the state of the power system. The sensors provide an analog signal which is converted to digital information at the LLP. These independent sensors are directly connected to a differential input analog to digital card in the LLP.

Computational Hardware

The computational hardware used in the SSM/PMAD consist of a SPARC based Solbourne 5/501 on which the user interface, KNOMAD database layer, and expert systems run and a series of 80386 microprocessors on which the Lowest Level Functions reside. Each 80386 microprocessor controls either a Load Center or a Power Distribution Control Unit. Each microprocessor is connected to the workstation via an Ethernet TCP/IP connection. In the case of Load Center, the controlling microprocessor communicates with two SIC cards in power system hardware through a RS-422 interface. Each SIC card controlled by the microprocessor in the Load Center controls switches on only one power bus. In the case of a Power Distribution Control Unit, the microprocessor communicates with only one SIC card and controls power for only one bus.

2.6 Software Description

2.6.1 Knomad

KNOMAD is a Knowledge Management and Design system that may be used for integrating and applying advanced computer programming technology to an application problem. KNOMAD currently provides support for data-driven and goal-driven reasoning (forward chaining and backward chaining rules, respectively), a constraint system with support for multiple contexts, and an advanced frame representation language for describing and reasoning about objects. KNOMAD is organized on top of a database layer (not persistent) that may be distributed if the application requires it. KNOMAD is written in Common LISP [Jr.89] and is easily interfaced with conventional LISP programs.

The reference manual included as Appendix B describes how to use KNOMAD in some detail. The description of KNOMAD in that appendix covers the following:

- Asserting and retrieving data to and from the database
- Aspects of the database such as views and database constraints
- The frame representation language
- The semantics of assertions and retrievals
- Database locks and the initialization of the database
- The distributed database
- The constraint system
- The rule management system
- Backward chaining
- Queues
- Events
- Semaphores
- The clock abstraction
- The unifier
- Object-oriented programming

A few comments from that manual are included here.

The database of KNOMAD provides an efficient mechanism for the storage and retrieval of tuples. KNOMAD uses two particular types of tuples, facts and frames. However, it is possible to use the database mechanism to store arbitrary tuples using primitives of KNOMAD. Database views as implemented in KNOMAD provide a logical storage area for tuples. Several database primitive functions are available. The database also allows for constraints to be placed upon tuples. A constraint specifies restrictions on what tuples are allowed in the database.

The distributed database of KNOMAD provides for the ability to share and distribute the same data among multiple computer processors. For the most part, this is completely invisible to the user. However, KNOMAD's default configuration is not to use the distributed database.

Three KNOMAD tools are available. These are the Rule Management System (for forward chaining), the Backward Chaining system, and the Constraint System. The mechanism for specifying a knowledge base, a description of a rule group, rule group methods, the execution strategy, the control strategy, the conflict resolution strategy, and the semantics of rules are all described in some detail in Appendix B. For more technical information about the implementation and perhaps clarification of the discussion in that appendix see [Rie90].

Performing inference using backward chaining in KNOMAD is very much like Prolog. Defining backward rules for facts and for slots of frames in effect defines virtual slots on frames and defines virtual facts in the database. Defining backward rules on facts (or frames) allows inferencing to produce implicit data in the database.

The constraint system of KNOMAD is quite advanced, however, it is not well integrated into KNOMAD, it does not use the KNOMAD database, it does not have a good set of user interface functions to manipulate it, etc. This system allows the user to specify a set of constraints, cells (being constrained), and their values. The constraint system does not return a solution to the network of constraints. It is up to the application to determine how underconstrained the network is and to further constrain it if necessary. The ability to ask the network to return all possible consistent solutions might be useful in the future. The constraint system also allows the user to specify values in contexts (which can subsume one another). This is done using Assumption-based Truth Maintenance System (ATMS) environments. Again, this integration of ATMS environments, needs further polishing and integration. The constraint system should be considered as an advanced prototype.

Miscellaneous Utilities

There are some miscellaneous utilities that are available to the application programmer utilizing KNOMAD. These utilities are queues, events, semaphores, a clock abstraction, and the unifier. An event is used by a process to wait for something. Events are generally used for synchronization. This implementation generalizes the basic event mechanism in two ways; 1)

multiple processes can wait on the same event, and 2) a waiting process can have a timeout.

Semaphores are another way to implement process synchronization. The semaphore provided here is a counting semaphore with three priority queues. There are four functions provided for it: `create-sem` for making a semaphore, `sem-p` and `sem-v` for grabbing and releasing a semaphore, and `init-sem` for reinitializing a semaphore.

The clock abstraction allows the application programmer to specify a clock that is independent of other application clocks, and even of the system clock. Using the clock it is possible to simulate a faster or slower time period using a specified units-per-second.

The KNOMAD unifier is a very simple symbolic expression unifier for LISP lists. It does not try to unify types with sub-types, etc.

Frames provide for the definition of objects and for inheritance. A method is generally considered to consist of some code that performs the indicated function and returns a result. The indicated function is usually specialized depending upon the object the function is being performed upon. Object-oriented languages naturally support the notion of function overloading. The frame representation language provides for this ability through the use of both if-needed aspects on slots as well as the backward chaining rule system of KNOMAD.

KNOMAD currently supports the FELES, the LPLMS and FRAMES, providing an environment for information management and tools supporting their functioning. For more detail see Appendix B.

2.6.2 MAESTRO

The MAESTRO scheduling system was developed by Martin Marietta as an internal research project. The Space Station scheduling problem was the original domain for which the system was designed, but it has since been applied to areas as diverse as launch vehicle countdown scheduling and tactical Air Force mission scheduling. This section provides a brief description of MAESTRO representations and functioning within the context of the SSM/PMAD system. For a more detailed description see Appendix C.

Objects the Scheduler Reasons About

MAESTRO has the capability to represent and reason about the following types of objects:

- Activities, defined by their structure (i.e., the subtasks making them up), their purpose or goal, and the constraints that may be imposed on them.
- Subtasks, well-defined sections of activities that use a consistent set of resources and are contiguous in time.
- Activity Groups, groupings of activities that comprise different ways of achieving a common goal.

- Resources, including three types of objects that are necessary to the performance of a subtask, e.g., voltmeters, crew, liquid nitrogen and lack of vibration.
- Temporal Constraints, specifications of how subtasks are temporally related to other subtasks, timepoints or events.
- Events, durations of time during which some condition holds true that is not under control of the scheduler but nevertheless affects scheduled activities, e.g., STS docking.
- Schedules, specifications of times when all subtasks in a set of activities will start and end.

Most of the above objects are implemented as classes in the Common Lisp Object System (CLOS).

Schedule Generation

MAESTRO schedules by first performing what is called opportunity calculation, which involves calculating all possible start and end times for all subtasks in a set of requested activities (considering both resource and temporal constraints), then executing a SELECT-PLACE-UPDATE cycle. During the SELECT phase of this cycle the scheduler heuristically chooses one activity from among those whose performance requests have not been fully satisfied. Information used by the scheduler to make this choice includes the priority of each activity, the relative constrainedness of each, the ratio of performances scheduled to requested to scheduled for each, whether each is involved in temporal constraints with one or more other activities, whether opportunity was found only such that one or more lower-priority activities must be unscheduled to place each, etc. The next phase, PLACE, involves specifying the starts and ends of all subtasks in the chosen activity such that no temporal or resource constraints are violated, and obeying placement preferences to the extent allowed by those constraints. In the UPDATE phase the system updates resource and conditions availabilities to reflect scheduling of the activity, then recalculates opportunity for all requested activities, which readies the scheduler for another Select phase. The system actually maintains four schedule versions, representing different priority levels of activities. The WHITE schedule has on it all activities, while the RED schedule only has those activities with medium (2) priority or higher, the blue only has those with high (1) priority or higher, and the GOLD schedule only has activities with life-critical (0) priority on it. If the scheduler cannot find any opportunity for a high-priority activity on the white schedule, it will look on the red schedule. If it finds opportunity there and that activity is chosen, placement of that activity will cause the unscheduling, or bumping, of one or more lower-priority activities. Thus bumping can comprise an optional portion of the PLACE phase of the SELECT-PLACE-UPDATE cycle.

Contingency Handling

One of the most important criteria for a scheduling system for Space Station is that it have the capability to modify schedules to account for changes in the assumptions upon which each schedule was based. For example, if it is assumed that a module will have a total of 25 kilowatts available for a certain time period, and creates a schedule based on that assumption, then later it is found that there will only be 20 kilowatts for that period, the scheduler must update the schedule to reflect that change. It may be that the schedule has been executing for a portion of that time period, so some activities may be already running when the need for that change is known, and in this situation it is not possible to recreate a schedule. Rather, it must be modified in such a way as to minimize the impact on activities being performed in the module.

MAESTRO performs contingency handling by first removing or truncating from the schedule activities that have been interrupted by power system events such as load shedding, and altering the representations for any activities whose power was switched to a redundant source. Next, all resource availabilities on the schedule are updated to conform to those on the breadboard. This may introduce situations in which there is a future portion of the schedule during which resource constraints are violated, so MAESTRO then performs contingency unscheduling, making use of a SELECT-UNSCHEDULE-UPDATE cycle and using twelve heuristics to choose activities and unschedule them until no constraint violations remain. This step is the same one that is executed during bumping, referred to previously. After all constraint violations are taken care of, the system tries to continue any interrupted activities, either by skipping, continuing or restarting the interrupted subtask, as allowed by the subtask descriptions and opportunities. During this step the scheduler actually creates new activities that are functional equivalents of the unexecuted portions of interrupted activities. Finally, the system tries to schedule any activities requested but not previously scheduled, in an attempt to make use of resources released by the unscheduling or interruption of other activities.

Schedule Change During Schedule Execution

One of the recent enhancements made to the SSM/PMAD system was the capability to modify a running schedule. Previously it was necessary to halt the schedule, perform schedule generation and then start the schedule anew in order to modify a running schedule. The current capability allows a user to add activities to or delete them from a running schedule. This capability is supported by a second scheduling system, functionally equivalent to the system that generates schedules, but loaded into a different LISP package, the :HYPO(thetical) package. During schedule modification, the running schedule with all contingency fixes is written out to a file and then read into the HYPO version of the scheduler. The projected time of schedule change completion is recorded, and no changes to the schedule are allowed before that time (so that the changes can be communicated to and implemented by the LLPs

when the modified schedule dictates). The user performs any desired schedule modifications, and if the resultant schedule is satisfactory, the user requests that the changed schedule be implemented. This causes a list of changes to the running schedule to be transmitted to KNOMAD and from there to MAESTRO, the FELES, the LPLMS, and eventually down to the LLPs. If the user does not complete schedule modifications before the predicted schedule change completion time, if the user decides not to implement the modified schedule, or if a breadboard contingency happens during schedule modification, then the modified schedule is thrown away and execution of the original schedule proceeds.

It must be noted that while the hypothetical scheduler implemented under this contract has all of the functionality of the original MAESTRO system, certain assumptions were made to facilitate specification and handling of user schedule modification requests. As delivered, the hypothetical scheduler expects activities which have no inter-activity temporal constraints. This is consistent with the delivered activity editor, which has no mechanism for specifying temporal constraints between activities. However, though there are some complex issues to be dealt with, a future enhancement of the activity editor to support the specification of temporal constraints between activities could be accompanied by an enhancement of the code used to make requests to the hypothetical scheduler such that it also would support the full array of temporal constraints that MAESTRO was designed to handle.

2.6.3 Front-End Load Enable Scheduler (FELES)

The FELES was so named because the scheduling system in SSM/PMAD was originally called the Load Enable Scheduler (LES), and the FELES was a front end for the LES. The FELES is a simple translation utility that converts a portion of the schedule of subtask start and end times generated by MAESTRO to a list of power system events to be passed along to the LLPs by the CAS. The FELES performs its function by obtaining from the subtask descriptions the set of power system requirements in each, then for each requirement adding an item to the event list that describes an identifier for the switch used, the amount of power required, the start and end of that power usage, and some additional information indicating whether the load is redundantly powered and has permission to use the redundant source. The event times listed by the FELES are offset from the times for the associated subtask starts and ends as specified in the schedule because the SSM/PMAD system clock begins running when the system is first loaded. The schedule could thus specify a start time of zero for the first subtask that may be several minutes after the system clock specified the SSM/PMAD mission time "zero".

This function is performed at schedule activation and intermittently as schedule execution progresses, and also both after contingency handling and after the user modifies a schedule using the hypothetical scheduler.

2.6.4 Load Priority List Management System Update

The Load Priority List Management System (LPLMS) periodically generates a list of priorities on loads to be used by the LLPs as a shed list in the event there is a reduction in the power available to support loads currently using power. The LPLMS has been redesigned and rehosted to run as a forward-chaining production system managed by KNOMAD. The LPLMS was previously hosted on the Symbolics workstation. It was rehosted to more strongly integrate the priority management algorithm with the SSM/PMAD database. As a result of rehosting the LPLMS, the knowledge and processes were formalized with definitions and categorical ordering.

The LPLMS Priority Definitions

The priorities originally established for the activities on a schedule are insufficient to perform intelligent load-shedding because, for example, two activities may have the same priority but one, if shut off, would perturb crew schedules while the other would not. Thus it was necessary to determine how to prioritize loads that support executing activities. The following definitions were established as a result of a knowledge engineering exercise between Martin Marietta and NASA/MSFC personnel, for the LPLMS update process. The primary goal of stating these definitions was to provide integrity and consistency across the multiple knowledge functions that utilize the common SSM/PMAD database.

Activity A set of tasks that accomplish a desired goal. Each task may have power system events associated with its start or end.

Crew Convenience A set of activities which enhance the capabilities and working conditions for humans. These activities may be performed at irregular time intervals.

Crew Safety A set of activities that must be accomplished in order to ensure human safety during a mission period. These activities may be performed at irregular time intervals.

Miscellaneous A set of activities that provide for maintaining the crew and scientific support environments. These activities should be performed at regular time intervals.

Priority A numerical representation of an activity's importance. Priorities in the SSM/PMAD system are integers ranging upward from 0, the highest priority.

Priority Rule A heuristic used in measuring and establishing priorities.

Resource An element necessary for the execution of activities.

Scientific A set of activities supporting experiments. These activities may be time varying.

Weight A heuristically adjusted numerical importance assigned to a component of an activity.

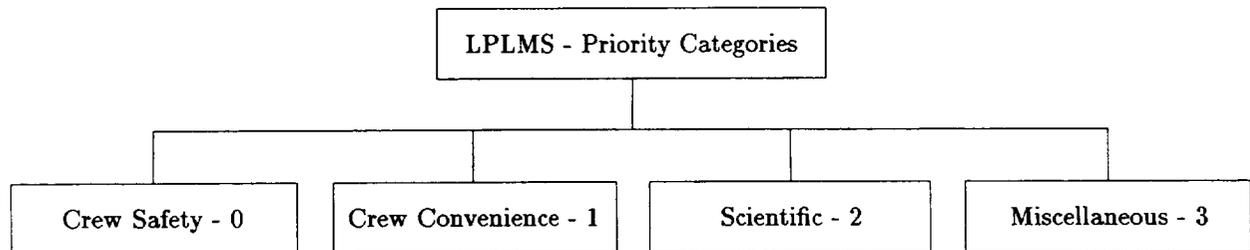


Figure 2.11: The Ordering of Priorities by Hierarchical Categories

Priority Ordering

Priorities are hierarchically ordered as shown in Figure 2.11. The grouping is by category, with category zero, Crew Safety, being the most important. Importance diminishes with increasing category number, with category three, Miscellaneous, being the least important. This categorization maintains agreement with the priority definitions.

Priority Calculation

An important constraint on the priority calculation is that categorical creeping is not allowed. This means that a priority ranging within category one, Crew Convenience, will not creep up to category zero, Crew Safety, or fall to category 2, Scientific. However, a priority will be allowed to range fully within its given category. The way this is accomplished is to calculate the priorities using a set of weights that are determined or adjusted by an appropriate set of rules. The weights are:

Crew Value The percentage of crew availability needed to perform an activity as compared to the total crew availability on the schedule.

Energy Usage Value The energy used by an activity as compared to the total energy available for the schedule.

General Equipment Value The (non-powered) equipment used by an activity as compared to the total (non-powered) equipment used during the complete schedule.

Requested to Scheduled Value The number of performances of an activity that were scheduled compared to the total number requested.

Events Value The number of events in an activity as compared to the total number of events in the schedule.

Conflicting Value This has a value of (0,1) and indicates whether an activity has any temporal constraints with other activities (it is more difficult to schedule or reschedule activities that are thus constrained). 0 → no conflicts; 1 → conflict(s) exist.

Opportunity Value This has a value of (0,1) and indicates whether an activity has a strong opportunity to be rescheduled. 0 → weak, 1 → strong.

Interruptable, Restartable, or Skippable Value This has a value of (0,1) and indicates whether an activity is interruptable, restartable, or skippable (any of which would make the activity easier to reschedule). 0 → it is, 1 → it is not one or more of these.

The LPLMS generates a list showing a single time-slice in a time-varying profile of activity priorities. It was necessary to determine the correct duration of each time-slice (or "hack"). Two values were examined for this hack. First a one minute hack was examined and a set of priorities were calculated. Next, a ten minute hack was examined and was found to produce the same set of priorities on the ten minute boundaries. Therefore, it is believed that priority creep occurs on a large time scale (> 10 minutes) rather than a small one. Also, the processing time to produce a one minute priority list is much greater than that to generate the ten minute priority list. Based on these results, ten minutes was chosen as the time hack. The resulting calculation of the activity's priority is:

$$P = (\sum \text{Weights}_{\text{Activity}})(\text{Hack-Start}/\text{Activity Duration}).$$

Each weight has associated rules used in its calculation. The weights, however, are static once the rules are executed in this implementation. This provided the advantage of moving the time varying calculation to the priority equation above, where each ten minute hack has a priority calculated separately. The way the calculation is performed, category creep does not occur. Also, if later requirements are imposed to calculate new weights dynamically, this can be done as periodic executions of the rules on the dynamic weights.

2.6.5 The KANT Run-Time Planning System

Under completely autonomous conditions, the flow of SSM/PMAD operational functionality is as depicted in Figure 2.12. This flow consists of three regions. First, the future parameters region which contains priority management, contingency scheduling, autonomous planning elements of KANT, and certain user interface functionality for requesting data and commanding full autonomous or full manual operation. Next, the near real-time control region, which contains knowledge management and the knowledge based fault management functions of FRAMES. Finally, the real-time control lower level process region which possesses

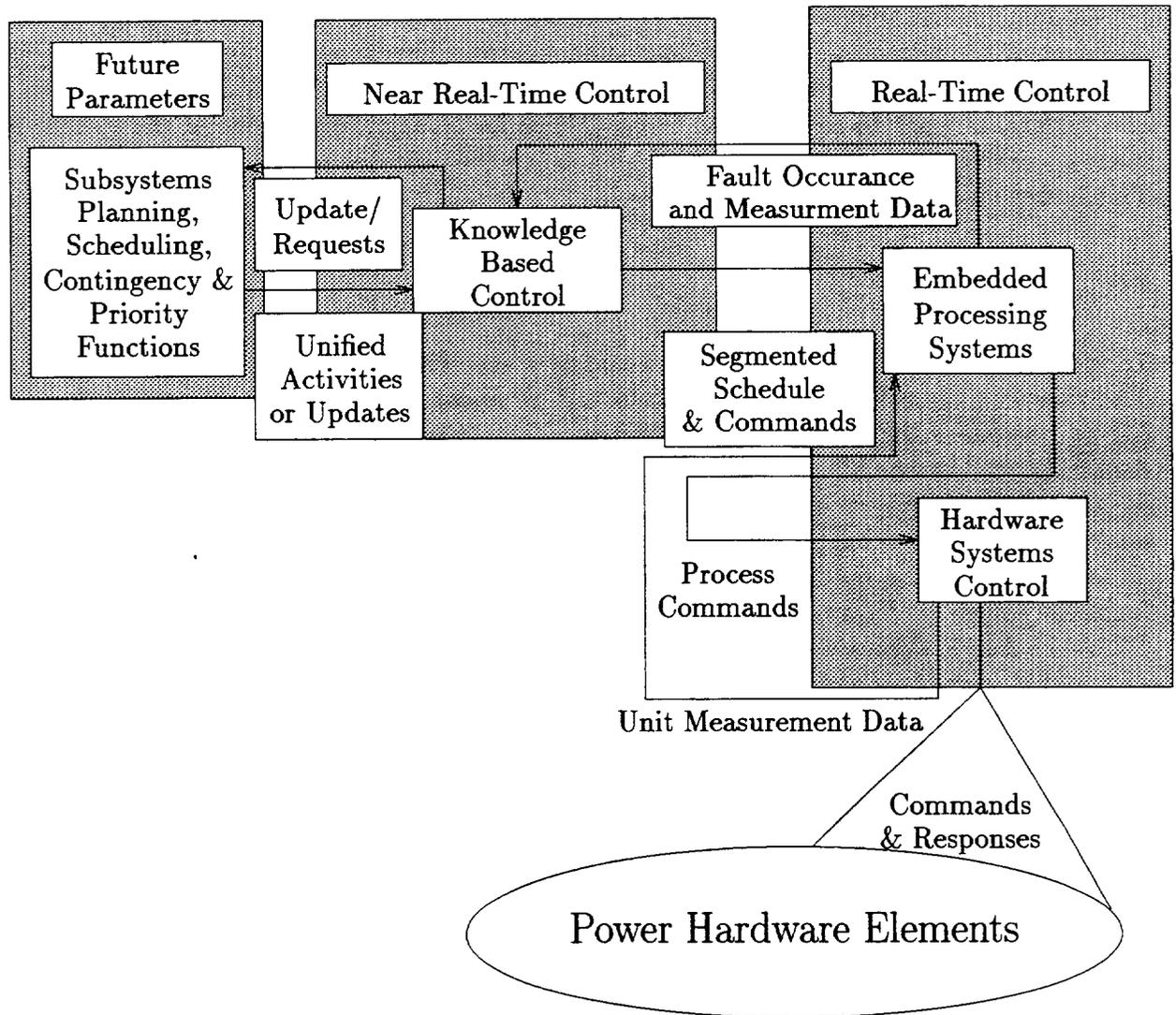


Figure 2.12: SSM/PMAD Test Bed Autonomous Operational Flow

the deterministic portion of FRAMES and the Lower Level Functions (LLFs) that perform actual hardware commanding.

When running in the fully autonomous mode these functions provide complete management, including fault handling. However, it is easy to see, as in Figure 2.13 that it would be strongly improbable for a user to know what to request or command in order to seize resources of the system to place under manual control. This is where the importance of the complete functionality of the KANT planning system becomes understood. KANT helps the user formulate what to do and how to do it in order to capture portions of the system for manual control. To accomplish this, KANT helps the user develop a plan to achieve ILA. This is depicted in Figure 2.14.

The User's Dilemma

To more fully understand the dilemma faced by a user attempting to achieve ILA, the block diagrams in Figure 2.15 and Figure 2.16 tell an encompassing story. Figure 2.15 shows the real resource the user desires to control; electrical power to loads, while Figure 2.16 demonstrates the complicated software functional interaction with each block representing an individual process environment within the complete distributed processing system. There are redundant power buses, the port side and the starboard side. Power is fed to loads on either a primary or a redundant basis. Primary loads have only one power feed. Redundant loads are either fed by or capable of being fed by both the port and starboard power buses. Each bus has a dedicated Power Distribution Control Unit (PDCU). These are in turn distributed lower to Load Centers (LCs) which complicate the ILA problem by combining scheduled control over the two power buses. Therefore, a user can be confused by interrupting loads in the following configurations where a load is:

1. Scheduled to begin execution and not redundant and not faulted
2. Scheduled to begin execution and redundant and not faulted
3. Executing and not redundant and not faulted
4. Executing and not redundant and faulted
5. Executing and redundant and not faulted
6. Executing and redundant and faulted
7. Completed with no system-wide need of the previous resource
8. Completed with a system need of the previous resource

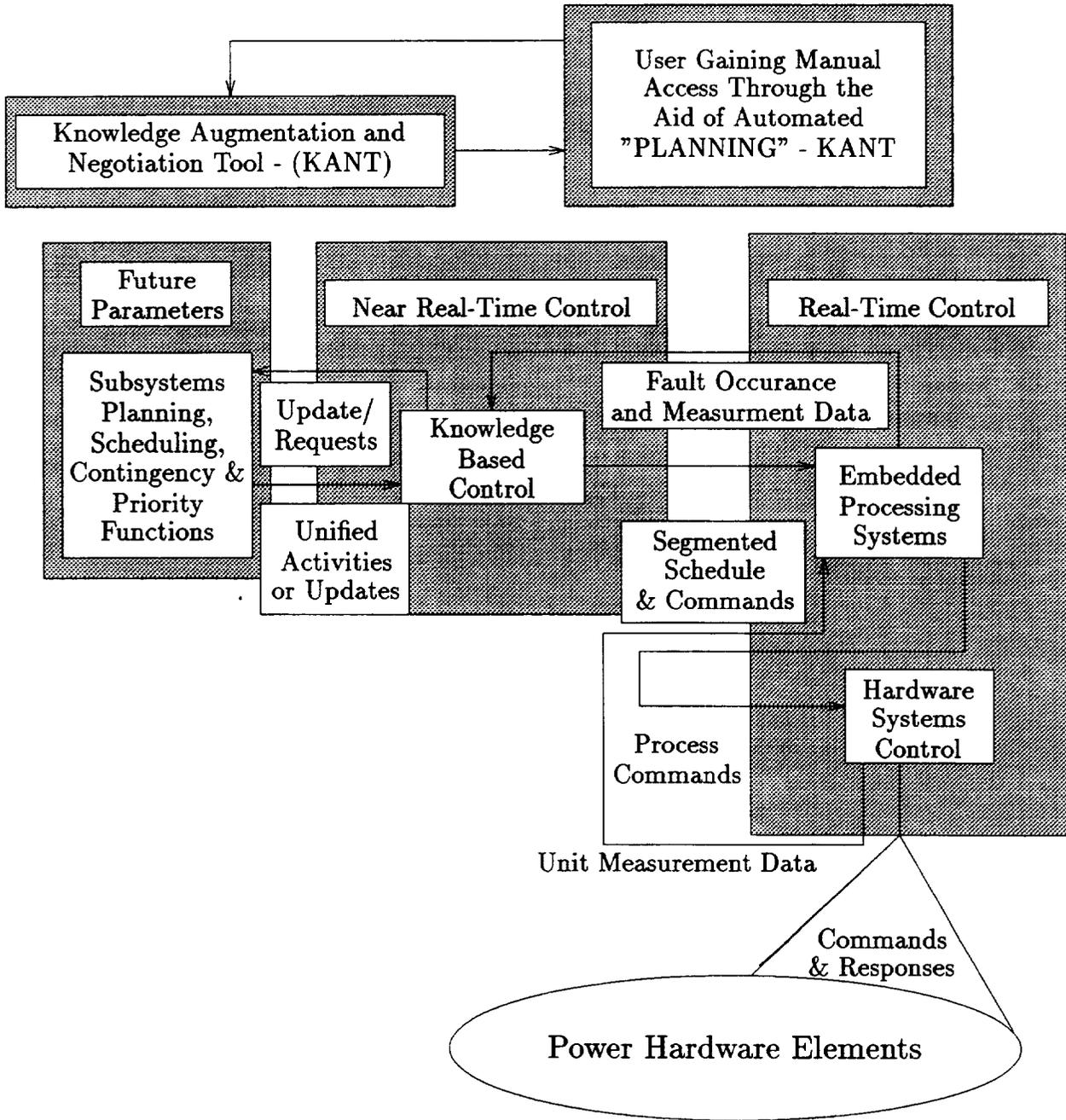


Figure 2.14: The KANT Planning Activity Interface

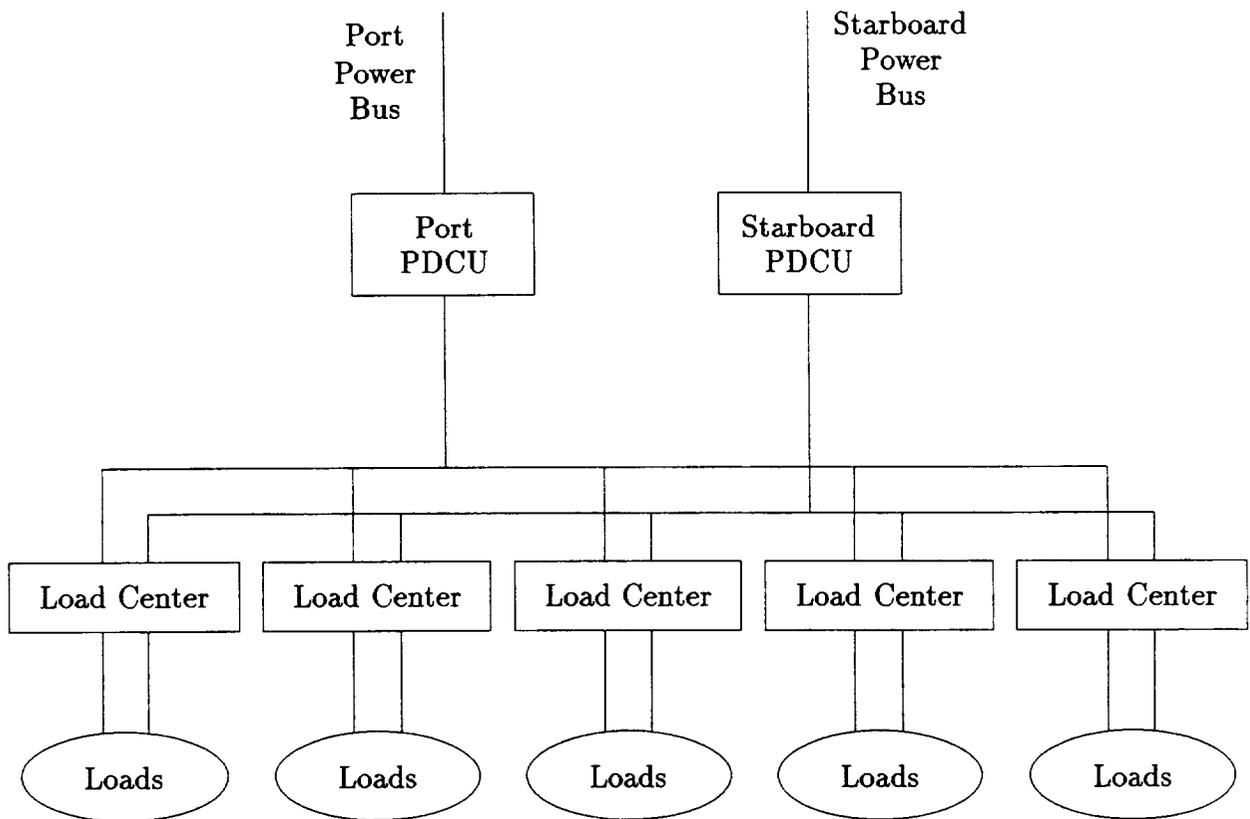


Figure 2.15: SSM/PMAD Power to Loads

These possible configurations give zero valued weights to users, contingencies, and priorities; something which is not reasonable for a real system, such as the SSM/PMAD test bed. The goal is to give the maximum valued weights to these attributes. Therefore, when considering the value of priorities, users and contingencies, one must evaluate their range within the system based on their effect on the possible configurations and on the allowable system elements.

The User

Figure 2.16 shows the software management and control elements which are present in the SSM/PMAD system. Notice that there are three management functions; the KNOMAD knowledge manager, the FRAMES fault management system, and the LPLMS priority manager; there is one planning function, KANT; one scheduler, MAESTRO; and one user interface consisting of the FELES and the U/I functions. All of these functional elements, whether knowledge based or procedural have a common interface to rule bases and a common interface to databases. This common rule base and database management function will be described in Section 2.6.1 on KNOMAD. The important thing here is to note that with common rule base and database interface capability, KANT, placed strategically between the user and the data, can help the user plan and decide which activities and resources to obtain for required periods of ILA activity.

The LLFs report data and state changes from the distributed 80386 processors up to FRAMES for processing at the upper level. The LLFs maintain a presently active schedule, a priority list, and a loads characteristics list with attributes such as upper and lower level amperage limits. It is important that the picture of the world remain consistent to these LLFs since all commanding of actual switchgear and readings of all sensors pass through them. Therefore, a user requesting ILA would need to know to communicate with the LLFs to provide necessary control parameters and activity lists. This is not feasible due to performance constraints and the fact that the LLFs have no direct user interface. That forces user interaction from the Solbourne workstation where the user interface does exist. This provides significant advantage in that much of the planning activity is knowledge based and also, the needed data spans the complete system, so central location at the Solbourne workstation with global knowledge management and central database access is a natural location for KANT based on present day processing architectures.

Contingencies

In the SSM/PMAD test bed, contingencies may be caused from different sources. The user requesting manual control fo switch groups may cause a contingency by interrupting an

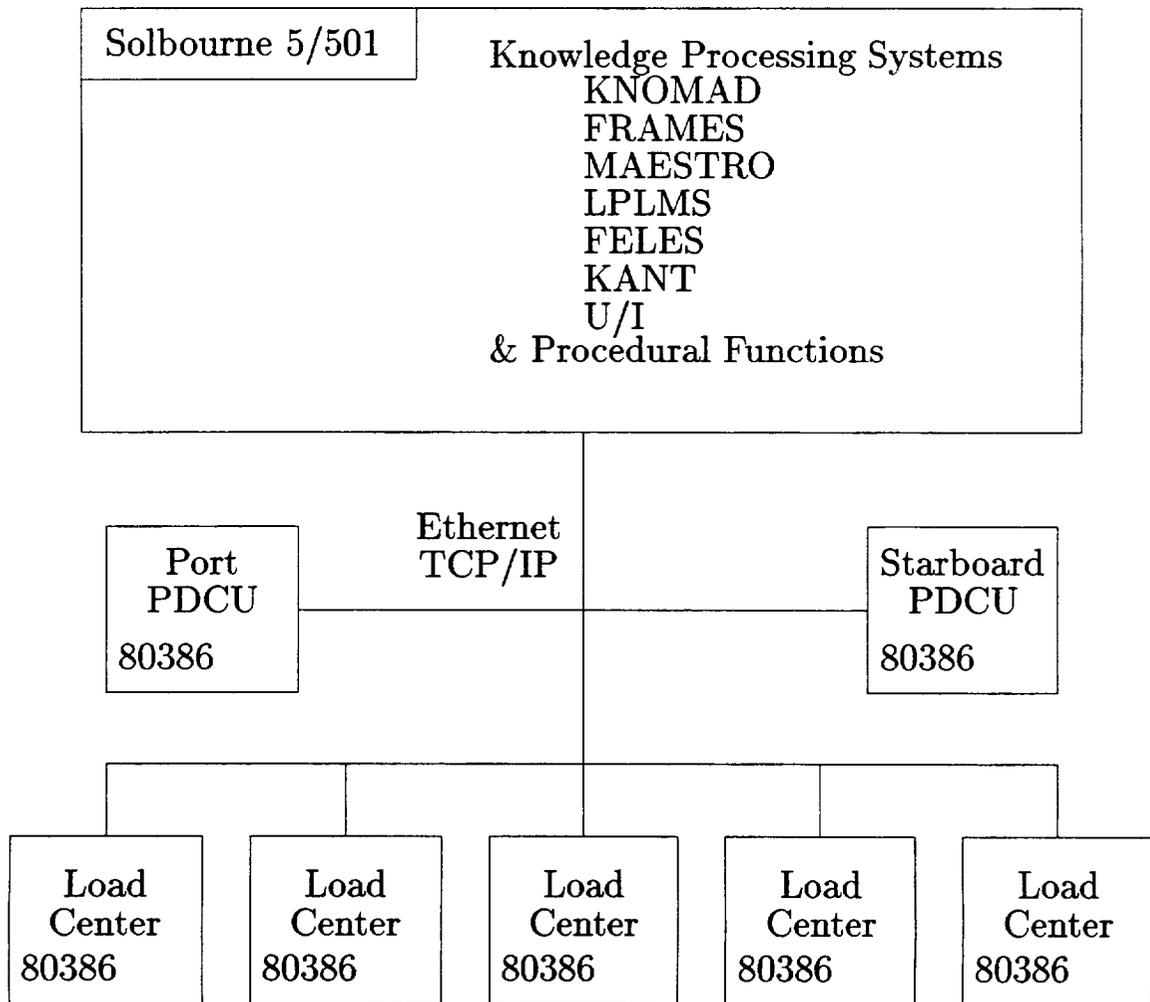


Figure 2.16: SSM/PMAD Distributed Control and Process Management

activity on the schedule. The approach of handling all contingencies the same within the common management system of the testbed provides a unique capability. That is, a schedule is generated and inability to execute the schedule causes a contingency management activity within the system. Therefore any contingencies generated by a user requesting manual control of switches are handled as perturbations to the schedule and cause rescheduling following the switch seizure. All contingencies are therefore handled in a consistent manner. This satisfied the goal for contingency handling.

Priorities

Priorities were handled through the introduction of Uniform Priority Blocks (UPBs). UPBs function the same as a fully autonomous system's priority list. The difference is that elements of the priority block are updated separately and are introduced at a later time than the original schedule. This is due to a user's perception and input when rating the priority of the seized switches. Also included in the determination is the possibility that the switch seizure duration may be indeterminate. This would yield a severely degraded priority update if priority adjustment were performed as usual. Therefore the user is allowed to introduce new priorities at manual seizure request time which may remain constant over the indeterminate period.

KANT Functionality

With users, contingencies, and priorities handled for all the possible execution configurations it is important to discuss what KANT does in its execution performance. The most important function within KANT is to be reactive and helpful to an ILA user's request. However, KANT's reaction must be able to maintain the highest level of system integrity at the same time. In order to achieve these goals KANT must help formulate the user's planned ILA activity while minimizing impacts to higher priority loads, schedule contingencies, and resource availability. While achieving its operational goals, KANT must fit within the framework of the fault management, scheduling, and priority management functions; otherwise, the operation and performance of the system-wide functionality may be compromised. Figure 2.17 presents the process flow in which KANT resides. The fit preserves the system-wide integrity whenever a user introduces the need for ILA activity. The functions identified from this fit of KANT into the complete environment are:

- The user's understanding of the interface
- Directing any needed rescheduling
- Adding newly calculated priorities to the system

- Forming process structures to be used for fault management in an ILA system
- Plan formulation for near-optimal introduction of the user to the autonomous system.

Adding New Priorities

New Priorities are introduced directly to the LPLMS from KANT. Needed priority interfacing to the LLFs is handled in an "as usual" manner, being passed from KNOMAD at the Solbourne workstation to the distributed 80386 processor.

Forming Fault Management Structures

All Process structures to allow fault management to occur during active periods of ILA are provided by KANT. FRAMES can therefore perform fault diagnosis, isolation, and recovery over the complete system; even though portions of the system are under manual control. This element of ILA also utilizes much related information from priority management and scheduling.

Plan Formulation for the User

A user may consume many resources within the SSM/PMAD system when requesting and initiating ILA. For instance, the user may effect the recovery process of a critical load if a redundant switch resource is consumed in establishing the ILA activity. This should not be acceptable whenever other lower priority non-redundant or unused resources are available. In essence, the plan should provide the user with a list of items and activities necessary to minimize impacts due to the execution of the overall ILA resource. For example, the actions a user must perform to place a faulted switch into service under ILA control. The generated plan list focuses on switch resources, load resources, and support resources, as well as the activities, priorities, and order of user directed execution to gain control of the necessary SSM/PMAD system elements in a manner that is near-optimal.

The previous paragraphs have presented a picture of the ILA activity and how the planning functionality of KANT helps make this possible. It was noted in several instances that KANT interfaced to different knowledge based functions, and that knowledge and data are managed globally. KNOMAD provides this knowledge and data management for the knowledge based elements of the SSM/PMAD. This relation is shown in Figure 2.18 with KANT to other knowledge base interfaces represented by the dumbbells, and the underlying management of knowledge and data, for all agents, being provided by KNOMAD. The next section describes this unique knowledge management function.

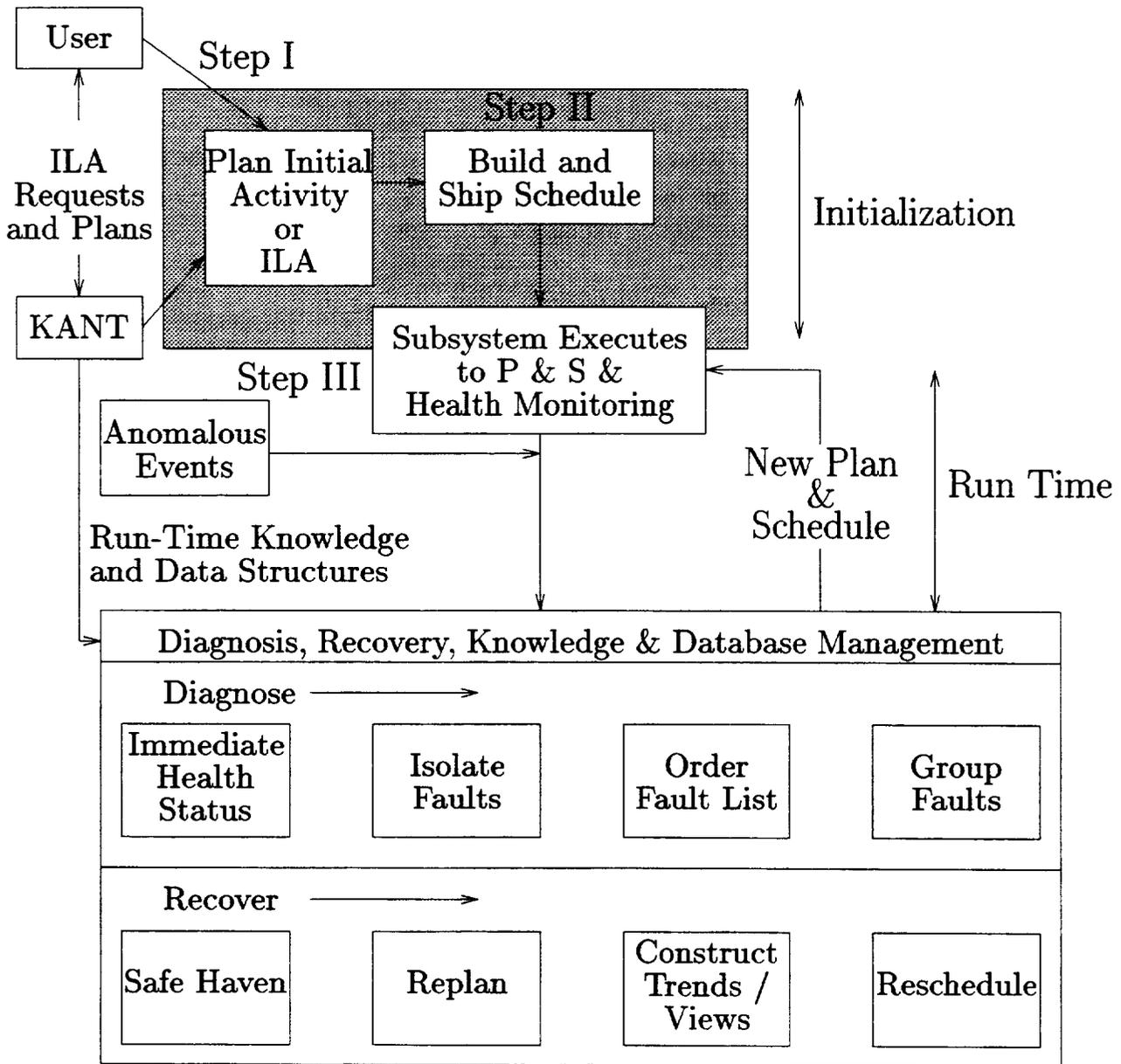


Figure 2.17: The SSM/PMAD Process Flow with KANT

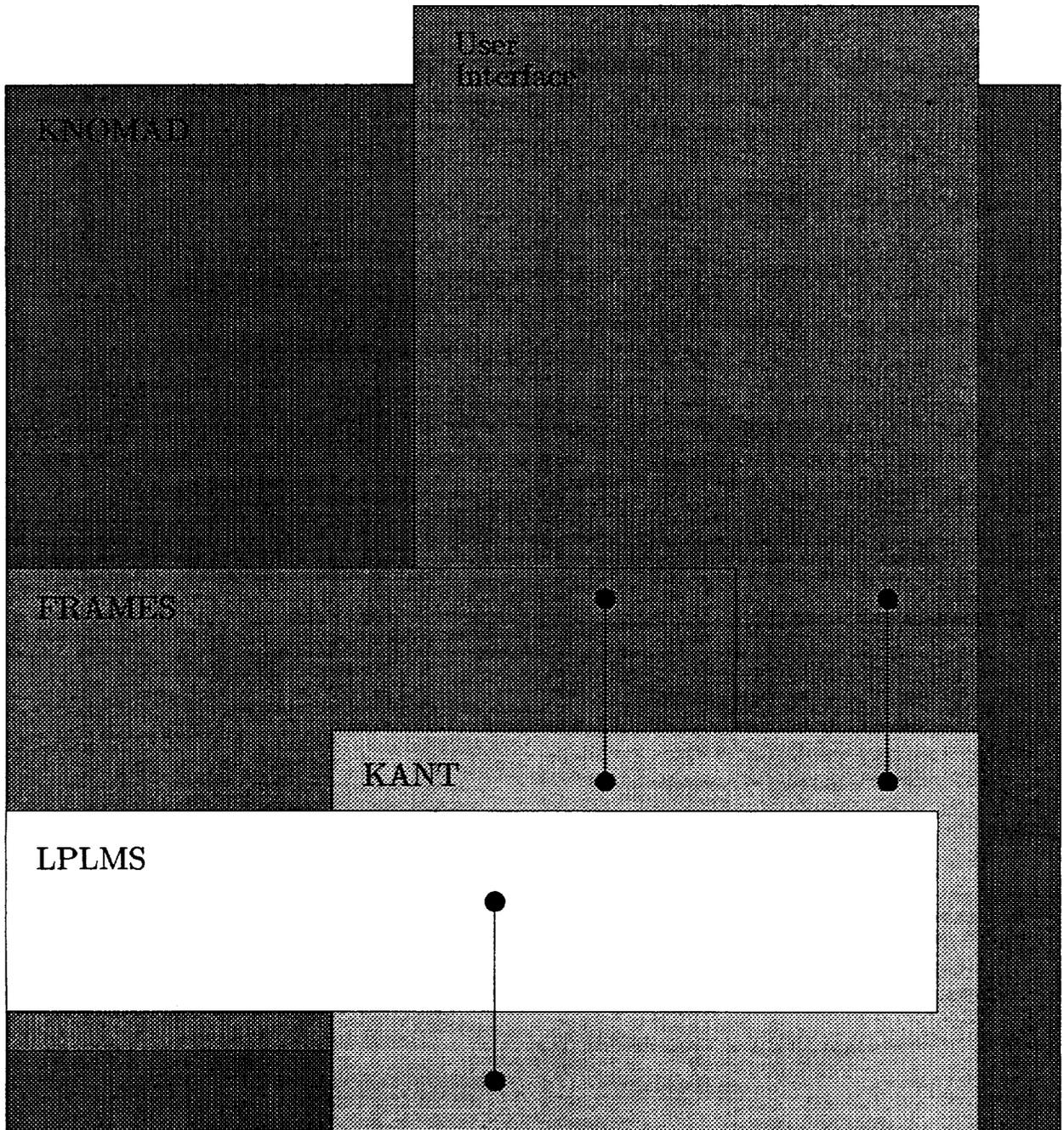


Figure 2.18: KANT Interface Relations

2.6.6 The Fault Recovery and Management Expert System

The hard fault expert system for FRAMES is described here. Soft and incipient fault handling is described in the section entitled "Soft and Incipient Faults". The hard fault expert system is organized as three rule groups, the multiple fault control rule group, the multiple fault rule group, and the diagnosis rule group. These are described in detail below:

Multiple Fault Control Rule Group

©©

©© The Control Rule Group

©©

©© This rule group controls the analysis of symptom sets by
 ©© calling the hard fault and diagnosis rule groups on symptom sets.

©©

©© There is a problem in this rule group.
 ©© The rules need to run for every symptom set of the symptom set queue.
 ©© However, the semantics of THERE EXISTS is to only fire once, as is
 ©© the semantics of simple rules. Until new and better semantics for
 ©© the needed kind of control is defined, we kludge this by resetting
 ©© the fired? slot on rules every time cluster-symptoms is called.

©©

RULE-GROUP : mf-control

```
CONTROL : ((start (mf-control-rule1)
  (mf-control-rule1 (mf-control-rule2))
  (mf-control-rule2 (mf-control-rule3 mf-control-rule4))
  (mf-control-rule3 (mf-control-rule1))
  (mf-control-rule4 (mf-control-rule1))))
```

©©

©© This rule watches for symptom sets in the symptom set queue.
 ©© This allows new symptom sets to arrive during diagnosis of
 ©© existing symptoms sets.

©©

```
MF-Control-Rule1
THERE EXISTS symptom-set in symptom-set-queue
< ::>
  [ the-symptom-set = symptom-set ] >
```

;

©©

©© When we have a symptom set we first cluster the symptom sets.
 ©© A symptom set may be the result of multiple independent faults,
 ©© hopefully from different parts of the power system topology.
 ©© Clustering the symptom set will produce one symptom
 ©© set from each identifiably distinct location in the power system

```

@@ that is bus-wise independent (trying to recognize separate faults).
@@
@@ This rule calls the hard fault rule group to determine a diagnosis
@@ for each cluster (a symptom set) in the-symptom-set.
@@ This consists of setting the
@@ symptom-set, setting multiple-hard-fault-analysis flag and
@@ initializing diagnosis-set. The multiple-hard-fault rule group
@@ is called followed by the mf-diagnosis rule group.
@@
MF-Control-Rule2
FOR ALL cluster in power-domain :: cluster-symptoms ( the-symptom-set )
  < ::>
    [ symptom-set = cluster ]
    [ multiple-hard-fault-analysis = started ]
    [ diagnosis-set = empty ]
    [ multiple-hard-fault :: execute ( multiple-hard-fault ) ]
    [ mf-diagnosis :: execute ( mf-diagnosis ) ] >
;
@@
@@ If there are more symptom sets in the symptom-set-queue
@@ we need to do the next symptom set.
@@
@@ This is a kludge. We want to fire this rule every time through.
@@ Normally, simple rules are only fired once (their semantics), to
@@ get them to fire more than once we will use a for all.
MF-Control-Rule3
FOR ALL symptom-set in symptom-set-queue
  < [ lisp :: length ( symptom-set-queue ) > 1 ]
  ::>
    [ power-domain :: send-out-of-services ]
    [ symptom-set-queue = symptom-set-queue MINUS the-symptom-set ] >
;
@@
@@ No more symptom sets, send an end-contingency too.
@@
MF-Control-Rule4
[ lisp :: length ( symptom-set-queue ) = 1 ]
::>
  [ power-domain :: send-out-of-services ]
  [ power-domain :: end-contingency ]
  [ symptom-set-queue = empty ]
:
@@
@@ There is no termination condition on this rule group.
@@ This rule group continually runs watching for symptom sets.
@@
DONE

```

Multiple Hard Fault Rule Group

00

00 The Hard Fault Rule Group for Multiple Faults

00

RULE-GROUP : Multiple-Hard-Fault

00

00

```
CONTROL : ((start (MF-rule1))
  (MF-rule1 (MF-rule2.1 MF-rule2.2 MF-rule2.3 MF-rule2.4
    MF-rule2.5 MF-rule3.1 MF-rule3.2 MF-rule4.1
    MF-rule4.2 MF-rule4.3 MF-rule4.4 MF-rule4.5))
  (MF-rule3.1 (MF-rule5 MF-rule5.1))
  (MF-rule3.2 (MF-rule3.2.1 MF-rule3.2.2 MF-rule3.2.3 MF-rule3.2.4
    MF-rule3.2.5 MF-rule3.2.6 MF-rule3.2.7))
  (MF-rule4.1 (MF-rule5 MF-rule5.1))
  (MF-rule4.2 (MF-rule5 MF-rule5.1))
  (MF-rule4.3 (MF-rule5 MF-rule5.1))
  (MF-rule4.4 (MF-rule5 MF-rule5.1))
  (MF-rule4.5 (MF-rule5 MF-rule5.1))
  (MF-rule5.1 (MF-rule6.1))
  (MF-rule6.1 (MF-rule6.2 MF-rule6.3))
  (MF-rule6.3 (MF-rule6.4 MF-rule6.5 MF-rule6.6 MF-rule6.7
    MF-rule6.8))
  (MF-rule6.8 (MF-rule6.8.1 MF-rule6.8.2))
  (MF-rule6.8.2 (MF-rule6.9))
  (MF-rule6.9 (MF-rule6.10 MF-rule6.11))
  (MF-rule6.10 (MF-rule6.12 MF-rule6.16))
  (MF-rule6.12 (MF-rule6.13 MF-rule6.14 MF-rule6.15))
  (MF-rule6.16 (MF-rule6.17 MF-rule6.22))
  (MF-rule6.17 (MF-rule6.18))
  (MF-rule6.18 (MF-rule6.19 MF-rule6.20 MF-rule6.21))
  (MF-rule6.21 (MF-rule6.9))
  (MF-rule6.22 (MF-rule6.23 MF-rule6.24 MF-rule6.25))
  (MF-rule6.2 (MF-rule5 MF-rule5.1))
  (MF-rule6.4 (MF-rule5 MF-rule5.1))
  (MF-rule6.5 (MF-rule5 MF-rule5.1))
  (MF-rule6.6 (MF-rule5 MF-rule5.1))
  (MF-rule6.7 (MF-rule5 MF-rule5.1))
  (MF-rule6.8.1 (MF-rule5 MF-rule5.1))
  (MF-rule6.11 (MF-rule5 MF-rule5.1))
  (MF-rule6.13 (MF-rule5 MF-rule5.1))
  (MF-rule6.14 (MF-rule5 MF-rule5.1))
  (MF-rule6.15 (MF-rule5 MF-rule5.1))
  (MF-rule6.19 (MF-rule5 MF-rule5.1))
```

```
(MF-rule6.20 (MF-rule5 MF-rule5.1))  
(MF-rule6.23 (MF-rule5 MF-rule5.1))  
(MF-rule6.24 (MF-rule5 MF-rule5.1))  
(MF-rule6.25 (MF-rule5 MF-rule5.1)))
```

00

00 First we need the top-symptoms of the current symptom-set.
00 This will give us an initial indication of a multiple fault
00 situation or no power to bus situation.

00

```
MF-rule1
```

```
::>
```

```
[ top-symptoms = power-domain :: top-symptoms ( symptom-set ) ]
```

```
;
```

00

00 Ok, things start getting quite a bit more complex here.
00 The series 2, 3, and 4 rules are all nondeterministic with respect to
00 each other. Therefore, we need to have negations of each of the
00 the series WRT to the other series in the rules themselves. This is
00 nothing new. It is a gotcha if you don't remember these things (like
00 I didn't - JDR). As of 4/26/90 we are embedding the negations of parts
00 of the series two rules into the series four rules. We don't need
00 to worry about the series three rules for a while as they depend upon
00 knowledge of activities at the lower switches. We don't yet have this
00 knowledge, therefore they will not be activated.
00 It turns out that only rule 4.5 seems to need modification.

00

00

00 The series 2.x rules deal with an under-voltage fault where there
00 is not current trip above the highest symptom; that is, the highest
00 symptom is either a single under voltage or a set of under voltages.

00

00 The 2.x rules determine broken input and output cables of switches
00 as well as possible no power to the bus situations.

00

00 All the 2.x rules first make sure that all the other switches on the
00 same bus that were on also tripped on under voltage and not on any
00 other fault.

00

00 May want to rethink these a bit. Switches do not have voltage.
00 Rule 2.5 will not fire.

00

00 Rule 2.1 checks if the top-sensor of the bus (either am0 or hm0) is
00 reading less than nominal voltage. If it is, then we probably don't
00 have power to the bus.

00

@@ The rest of the 2.x rules (2.2 2.3 2.4 2.5) are cases
 @@ where there does appear to nominal voltage at the top-sensor, and
 @@ therefore have to do with cases where there may be broken switches
 @@ and faulty sensors.

@@

```

MF-rule2.1
[ lisp :: length ( top-symptoms ) >= 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ voltage of top-sensor of switch of symptom <=
    under-voltage-value of switch of symptom ] > ]
[ THERE EXISTS symptom1 in top-symptoms
  < [ FOR ALL switch1 in siblings of switch of symptom1
    WHERE [ event of switch1 <> :unknown ]
          [ command of event of switch1 = n ]
        < [ THERE EXISTS symptom in top-symptoms
          < [ switch of symptom = switch1 ] > ] > ] > ]
::>
[ diagnosis = no-power-to-bus ]
[ multiple-hard-fault-analysis = done ]

```

;

@@

@@ Rule 2.2 checks if there is nominal voltage at the sensor above one
 @@ of the tripped switches. If there is, then either the cable is
 @@ broken between that sensor and the switches (for more than one switch)
 @@ or the switch-input of the tripped switches is broken.

@@

```

MF-rule2.2
[ lisp :: length ( top-symptoms ) >= 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom1 in top-symptoms
  < [ FOR ALL switch1 in siblings of switch of symptom1
    WHERE [ event of switch1 <> :unknown ]
          [ command of event of switch1 = n ]
        < [ THERE EXISTS symptom in top-symptoms
          < [ switch of symptom = switch1 ] > ] > ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ voltage of top-sensor of switch of symptom >
    under-voltage-value of switch of symptom ]
    [ voltage of sensor-above of switch of symptom >
    under-voltage-value of switch of symptom ] > ]
::>
[ diagnosis = broken-cable-between-sensor-above-and-u-v-switches ]
[ multiple-hard-fault-analysis = done ]

```

;

@@

SSM/PMAD Final Report

@@ Rule 2.3 checks the case where the sensor above the tripped switches
 @@ is reading less than nominal voltage and the switch above the
 @@ tripped switches can trip on under voltage but is reading a
 @@ a nominal voltage. In this case a broken output cable of the
 @@ switch above the tripped switches is hypothesized.
 @@

```

MF-rule2.3
[ lisp :: length ( top-symptoms ) >= 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom1 in top-symptoms
  < [ FOR ALL switch1 in siblings of switch of symptom1
    WHERE [ event of switch1 <> :unknown ]
          [ command of event of switch1 = n ]
        < [ THERE EXISTS symptom in top-symptoms
          < [ switch of symptom = switch1 ] > ] > ] > ] ]
[ THERE EXISTS symptom in top-symptoms
  < [ voltage of top-sensor of switch of symptom >
    under-voltage-value of switch of symptom ]
    [ voltage of sensor-above of switch of symptom <=
    under-voltage-value of switch of symptom ]
    [ under-voltage-trippable of switch-above of switch of symptom =
    true ] > ]
::>
[ diagnosis = broke-output-cable-of-switch-above ]
[ multiple-hard-fault-analysis = done ]

```

;

@@

@@ Rule 2.4 checks that the sensor above the tripped switches is reading
 @@ less than nominal voltage and that the switch above the tripped switches
 @@ is NOT trippable on under voltage and, at the same time, that the
 @@ sensor above the switch above the tripped switches IS reading a
 @@ nominal voltage. If this is the case we can hypothesize that the
 @@ switch above the tripped switches may have a broken input or output
 @@ cable or it may simply be busted.
 @@

@@

```

MF-rule2.4
[ lisp :: length ( top-symptoms ) >= 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom1 in top-symptoms
  < [ FOR ALL switch1 in siblings of switch of symptom1
    WHERE [ event of switch1 <> :unknown ]
          [ command of event of switch1 = n ]
        < [ THERE EXISTS symptom in top-symptoms
          < [ switch of symptom = switch1 ] > ] > ] > ] ]
[ THERE EXISTS symptom in top-symptoms
  < [ voltage of top-sensor of switch of symptom >

```

```

    under-voltage-value of switch of symptom ]
  [ voltage of sensor-above of switch of symptom <=
    under-voltage-value of switch of symptom ]
  [ under-voltage-trippable of switch-above of switch of symptom = false ]
  [ voltage of sensor-above of switch-above of switch of symptom >
    under-voltage-value of switch of symptom ] > ]
::>
[ diagnosis = broke-input-cable-of-switch-above ]
[ multiple-hard-fault-analysis = done ]
;
@@
@@ Rule 2.5 checks that there is a nominal voltage reading at the
@@ sensor above the tripped switches and that the switch above the
@@ tripped switches can trip on under voltage but is also reading
@@ a nominal voltage, but the sensor above the switch above is
@@ reading less than nominal voltage. In this case we can hypothesize
@@ that the input cable to the switch above may be broken as well as
@@ the under voltage sensor of the switch above.
@@
MF-rule2.5
[ lisp :: length ( top-symptoms ) >= 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom1 in top-symptoms
  < [ FOR ALL switch1 in siblings of switch of symptom1
    WHERE [ event of switch1 <> :unknown ]
      [ command of event of switch1 = n ]
    < [ THERE EXISTS symptom in top-symptoms
      < [ switch of symptom = switch1 ] > ] > ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ voltage of top-sensor of switch of symptom >
    under-voltage-value of switch of symptom ]
  [ voltage of sensor-above of switch of symptom <=
    under-voltage-value of switch of symptom ]
  [ under-voltage-trippable of switch-above of switch of symptom = true ]
  [ voltage of switch-above of switch of symptom >
    under-voltage-value of switch of symptom ]
  [ voltage of sensor-above of switch-above of switch of symptom <=
    under-voltage-value of switch of symptom ] > ]
::>
[ diagnosis =
  broke-in-cable-above-switch-above-and-bad-u-v-sensor-switch-above ]
[ multiple-hard-fault-analysis = done ]
;
@@
@@ The 3.x.x series of rules deal with two special cases.
@@ 3.1 checks that there is more than one top-symptom where all

```

@@ the top-symptoms are at the bottom level of switches (right above
@@ the loads, and that all the top-symptoms tripped on either
@@ over-current or fast-trip. Finally all the top-symptoms are related
@@ by an activity that is using the switches of the top-symptoms.
@@ 3.2 checks that the set of top-symptoms are not related by the same
@@ activity and that the switches are at the bottom level and
@@ that they all tripped on fast-trip.
@@

@@
@@ Rule 3.1, as stated above, checks that the switches of all the
@@ symptoms are at the bottom level and tripped on either over-current
@@ or fast-trip. The switches of the symptoms are also related by
@@ being used by the same activity.
@@ IF these conditions are satisfied then perhaps the reason this set
@@ of symptoms occurred is that the activity is behaving badly. To
@@ test this the 3.1.x rules will first flip and then close the switches
@@ in an effort to see if there is any repeatability in the symptoms.
@@

```
MF-rule3.1
[ lisp :: length ( top-symptoms ) > 1 ]
[ FOR ALL symptom in top-symptoms
  < [ switches-below of switch of symptom = empty ] > ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = over-current ]
    OR
    [ fault of symptom = fast-trip ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ FOR ALL symptom1 in top-symptoms
    < [ activity of switch of symptom =
      activity of switch of symptom1 ] > ] > ]
::>
[ the-individual-symptoms = top-symptoms ]
@ until we actually have activities to look at on the workstation
@ we will treat these as individual cases.
@[ switches-to-test = empty ]
```

```
;
```

@@
@@ This is where the 3.1.x rules will go
@@

@@
@@ Rule 3.2 checks that all the symptoms are fast-trip and all at the
@@ bottom level. Also all the switches of the symptoms are not related
@@ by being used by the same activity. In this case we consider the
@@ possibility that there was a fault below one of the switches and
@@ the other switches also tripped on fast trip due to energy storage
@@ in their loads.

SSM/PMAD Final Report

00 To test this we flip all the switches and try to get ONE of them
 00 to retrip on fast-trip.

00

```
MF-rule3.2
[ lisp :: length ( top-symptoms ) > 1 ]
[ FOR ALL symptom in top-symptoms
  < [ switches-below of switch of symptom = empty ]
    [ fault of symptom = fast-trip ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ THERE EXISTS symptom1 in top-symptoms
    < [ activity of switch of symptom <>
      activity of switch of symptom1 ] > ] > ]
::>
[ switches-to-test = empty ]
;
```

00

00 Rule 3.2.1 grabs all the switches that may be tested according
 00 to the permission-to-test field of the scheduled event for the switch.

00

```
MF-rule3.2.1
::>
[ FOR ALL symptom in top-symptoms
  WHERE [ permission-to-test of event of switch = y ]
  < [ switches-to-test = switches-to-test PLUS switch of symptom ] > ]
;
```

00

00 Rule 3.2.2. actually initiates the flipping process if there are
 00 switches to test.

00

```
MF-rule3.2.2
[ lisp :: length ( switches-to-test ) > 0 ]
::>
[ new-symptoms = power-domain :: flip-switches ( switches-to-test ) ]
;
```

00

00 Rule 3.2.3 checks to see if there aren't any switches to test. If there
 00 aren't then we diagnose that situation (basically we can't determine
 00 anything and can only notify the user).

00

```
MF-rule3.2.3
[ lisp :: length ( switches-to-test ) = 0 ]
::>
[ diagnosis = no-permission-to-test-possible-backrush ]
;
```

00

00 Rule 3.2.4 checks if we got more than one symptom as a result of
 00 flipping the switches. If so we have a problem. Flipping flips
 00 the switches one at a time individually, there should only be one

```

@@ symptom as a result of this operation.
@@
MF-rule3.2.4
[ lisp :: length ( new-symptoms ) > 1 ]
::>
[ diagnosis = unexpected-to-many-retrips-possible-backrush ]
;
@@
@@ Rule 3.2.5 checks to see if we didn't get any symptoms as a result
@@ of flipping these switches. If that is the case we also don't
@@ know what the problem is. If there is a short below a switch it ought
@@ to retrip the switch when the switch gets turned on. Could be a
@@ misbehaving load I suppose.
@@ This diagnosis should check if there were some switches that didn't
@@ have permission to test. That is, are switches-to-test and top-symptoms
@@ of the same length?
@@
MF-rule3.2.5
[ lisp :: length ( new-symptoms ) = 0 ]
::>
[ diagnosis = no-retrips-on-flips-possible-backrush ]
;
@@
@@ Rule 3.2.6 checks that if we did get a single new symptom and
@@ it is also a fast trip and one of the same switches of the original
@@ top-symptoms, then we can diagnose this situation as a possible
@@ backrush situation.
@@
MF-rule3.2.6
[ lisp :: length ( new-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-symptoms
  WHERE [ fault of symptom = fast-trip ]
  < [ THERE EXISTS symptom1 in top-symptoms
    < [ switch of symptom = switch of symptom1 ] > ] > ]
::>
[ diagnosis = found-possible-backrush ]
;
@@
@@ Rule 3.2.7 checks that if we did get a single new symptom but it
@@ isn't a fast trip or it isn't one of the switches of the original
@@ top-symptoms, then we have another unexpected situation.
@@
MF-rule3.2.7
[ lisp :: length ( new-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-symptoms
  < [ fault of symptom <> fast-trip ]
  OR
  [ FOR ALL symptom1 in top-symptoms
```

```

    < [ switch of symptom <> switch of symptom1 ] > ] > ]
::>
[ diagnosis = unexpected-retrip-possible-backrush ]
;

00
00 The 4.x rules set things up to treat multiple top symptoms as
00 individual top symptoms, as different faults. It grabs
00 all the cases that weren't applicable in the 2.x and 3.x series.
00
MF-rule4.1
[ lisp :: length ( top-symptoms ) > 1 ]
[ THERE EXISTS symptom in top-symptoms
  < [ switches-below of switch of symptom = empty ]
    [ fault of symptom = over-current ]
    [ THERE EXISTS symptom1 in top-symptoms
      WHERE [ symptom <> symptom1 ]
        < [ activity of switch of symptom <>
          activity of switch of symptom1 ] > ] > ]
::>
[ the-individual-symptoms = symptom-set ]
;
MF-rule4.2
[ lisp :: length ( top-symptoms ) > 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ THERE EXISTS switch1 in siblings of switch of symptom
    < [ command of event of switch1 = n ]
      [ FOR ALL symptom1 in top-symptoms
        < [ switch1 <> switch of symptom1 ] > ] > ] > ]
::>
[ the-individual-symptoms = top-symptoms ]
;
MF-rule4.3
[ lisp :: length ( top-symptoms ) > 1 ]
[ FOR ALL symptom in top-symptoms
  < [ fault of symptom <> under-voltage ] > ]
::>
[ the-individual-symptoms = top-symptoms ]
;
MF-rule4.4
[ lisp :: length ( top-symptoms ) > 1 ]
[ THERE EXISTS symptom in top-symptoms
  < [ fault of symptom = under-voltage ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ fault of symptom <> under-voltage ] > ]
::>

```

SSM/PMAD Final Report

```
[ the-individual-symptoms = top-symptoms ]
;
MF-rule4.5
[ lisp :: length ( top-symptoms ) = 1 ]
@@ This next selector is for the negation of the series two rules (Hah!)
@@ The under voltage should have been caught by the series two rules.
[ THERE EXISTS symptom in top-symptoms
  < [ fault of symptom <> under-voltage ] > ]
::>
[ the-individual-symptoms = top-symptoms ]
;

@@
@@ There are two ways that we can finish diagnosing hard faults.
@@ One way is by concluding a value for diagnosis based on one of the cases
@@ in the 2.x and 3.x series of rules. The other is by going through all
@@ the individual top symptoms and getting a diagnosis for all of them
@@ (adding them to the diagnosis set one by one).
@@ The 5.x series of rules are simply to manage the looping necessary
@@ to diagnose each top-symptom individually as individual faults.
@@
MF-rule5
[ the-individual-symptoms = empty ]
::>
[ multiple-hard-fault-analysis = done ]
;
MF-rule5.1
[ the-individual-symptoms <> empty ]
::>
[ top-symptom = lisp :: first ( the-individual-symptoms ) ]
[ the-individual-symptoms = lisp :: rest ( the-individual-symptoms ) ]
;

@@
@@ The 6.x series of rules are used to diagnose single faults with
@@ only a single top symptom (not multiple tops, they were handled
@@ above). This is very similar, but simpler, to the original set
@@ of rules for doing hard-fault diagnosis. Some simplicity is
@@ obtained by handling backrush and under-voltage faults in the
@@ earlier rules.
@@
@@ The types of faults that will be detected in the 6.x series of rules
@@ are over current and fast trip faults. They can also be with
@@ a number of masked faults where sensors didn't work properly.
@@ Additionally, these rules are smart enough that if a fault is
@@ found that is lower than the top, and there are other potential
@@ faults in a different portion of the tree below the top, then
@@ those other faults are added to the-individual-symptoms so that
```

@@ they may be diagnosed as well.
@@

@@
@@ Rule 6.1 simply opens all the switches from the top-symptom on
@@ down. This is an initialization step.
@@

```
MF-rule6.1
::>
[ switch of top-symptom = power-domain :: kludge-switch ( top-symptom ) ]
[ fault of top-symptom = power-domain :: kludge-fault ( top-symptom ) ]
[ new-symptoms = power-domain :: open-switches ( top-symptom ) ]
;
```

@@
@@ Rule 6.2 determines if we got some trips as a result of opening
@@ the switches. Getting new symptoms during the open operations is an
@@ unexpected result and notification to the user will be given of
@@ this problem.
@@

```
MF-rule6.2
[ lisp :: length ( new-symptoms ) > 0 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-symptoms-during-open
                                   :top-symp top-symptom
                                   :slot1 new-symptoms ) ]
;
```

@@
@@ Rule 6.3 flips the top switch. It collects any symptoms that may arise
@@ as a result of the flip.
@@

```
MF-rule6.3
[ lisp :: length ( new-symptoms ) = 0 ]
::>
[ new-symptoms = power-domain :: flip-switch ( switch of top-symptom ) ]
;
```

@@
@@ Rule 6.4 checks if there was more than one symptom as a result of the
@@ flip of the single top switch. If so this is an unexpected situation
@@ and the user will be notified.
@@

```
MF-rule6.4
[ lisp :: length ( new-symptoms ) > 1 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis
  ( :name unexpected-too-many-symptoms-flip-top
    :top-symp top-symptom :slot1 new-symptoms ) ]
```

```

;
@@
@@ Rule 6.5 finds the case where the top switch retrips on the
@@ same trip as a result of the flip operation. This is a strong
@@ indication of a short directly below the top switch.
@@ The diagnosis also will note if there were any fast trips at
@@ the bottom level (and that the top switch also tripped on fast trip)
@@ and suggest the possibility of energy storage in the loads
@@ driving those bottom fast trips.
@@
MF-rule6.5
[ lisp :: length ( new-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-symptoms
  < [ switch of symptom = switch of top-symptom ]
    [ fault of symptom = fault of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name retrip-on-flip
                                   :top-symp top-symptom ) ]
;
@@
@@ Rule 6.6 also detects a single retrip on the flip operation, but
@@ this retrip is either a different symptom or a different switch or
@@ both, and therefore is unexpected. The user will be notified.
@@
MF-rule6.6
[ lisp :: length ( new-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-symptoms
  < [ fault of symptom <> fault of top-symptom ]
    OR
    [ switch of symptom <> switch of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-retrip-during-flip
                                   :top-symp top-symptom
                                   :slot1 new-symptoms ) ]
;
@@
@@ Rule 6.7 checks the case where the top switch did not retrip during
@@ the flip operation and notes that there are no switches below
@@ the top switch. In this case, the fault is not found and there
@@ is nowhere else to check. Perhaps the fault burned itself clear?
@@
MF-rule6.7
[ lisp :: length ( new-symptoms ) = 0 ]
[ switches-below of switch of top-symptom = empty ]
::>
[ diagnosis-set = diagnosis-set PLUS

```

```

power-domain :: make-diagnosis ( :name not-found-no-levels
                                :top-symp top-symptom ) ]
;
@@
@@ Rule 6.8 checks that there were no new trips during the flip operation
@@ on the top switch, but in this case there are switches below that
@@ can be tested too. So this rule initializes some variables to
@@ start flipping and closing the lower switches.
@@ This rule also closes the top switch in preparation for flipping
@@ and closing the lower switches
@@
@@ It is at this point of the game that we are now interested in
@@ finding masked faults. It is a possibility that
@@ a lower switch's current trip sensor is broken and that therefore
@@ the top switch tripped.
@@
MF-rule6.8
[ lisp :: length ( new-symptoms ) = 0 ]
[ lisp :: length ( switches-below of switch of top-symptom ) > 0 ]
::>
[ switches-below-to-test = empty ]
[ switches-below-cant-test = empty ]
[ switches-below = switches-below of switch of top-symptom ]
[ new-symptoms = power-domain :: close-switch ( switch of top-symptom ) ]
;
@@
@@ Rule 6.8.1 checks to see if we got new symptoms during the close of
@@ of the top switch. If we did this is completely unexpected.
@@ A new symptom should have only come during the flip (since we
@@ are diagnosing current trips right now). We will notify
@@ the user of this problem.
@@
MF-rule6.8.1
[ lisp :: length ( new-symptoms ) > 0 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-trips-during-close-top
                                  :top-symp top-symptom
                                  :slot1 new-symptoms ) ]
;
@@
@@ Rule 6.8.2 checks that we didn't get any new symptoms as a result
@@ of the close of the top switch and therefore we can go on to rule
@@ 6.9
@@
MF-rule6.8.2
[ lisp :: length ( new-symptoms ) = 0 ]

```

```

::>
[ level = 1 ]
;
00
00 Rule 6.9 sets up the switches-below-to-test and switches-below-cant-test
00 variables based upon the status of the switch below and whether it
00 was supposed to be on.
00
MF-rule6.9
[ level > 0 ]
::>
[ FOR ALL switch in switches-below
  WHERE [ tripped of switch = under-voltage ]
        [ command of event of switch = n ]
        [ permission-to-test of event of switch = y ]
  < [ switches-below-to-test = switches-below-to-test PLUS switch ] > ]
[ FOR ALL switch in switches-below
  WHERE [ command of event of switch = n ]
        [ permission-to-test of event of switch = n ]
        OR
        [ command of event of switch = n ]
        [ tripped of switch <> under-voltage ]
  < [ switches-below-cant-test = switches-below-cant-test PLUS switch ] > ]
;
00
00 Rule 6.10 flips the switches that can be tested to see if
00 we get any retrips.
00
MF-rule6.10
[ lisp :: length ( switches-below-to-test ) > 0 ]
::>
[ new-symptoms = power-domain :: flip-switches ( switches-below-to-test ) ]
;
00
00 Rule 6.11 checks for the possibility that none of the lower switches
00 may be tested. In this case we have to diagnose with what we got.
00
MF-rule6.11
[ lisp :: length ( switches-below-to-test ) = 0 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name not-found-cant-test-further
                                   :top-symp top-symptom
                                   :slot1 switches-below-cant-test ) ]
;
00
00 Rule 6.12 checks to see if we got more than one new symptom during
00 the flips of the lower switches. This is possible if we have a

```

```
00 masked fault. At this point we find the top symptoms of the new
00 symptoms.
00
MF-rule6.12
[ lisp :: length ( new-symptoms ) > 1 ]
::>
[ new-top-symptoms = power-domain :: top-symptoms ( new-symptoms ) ]
;
00
00 Rule 6.13 checks if we have more than one new top symptom.
00 If we do, this is unexpected and the user will have to be notified.
00
MF-rule6.13
[ lisp :: length ( new-top-symptoms ) > 1 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-to-many-tops-after-flips
    :top-symp top-symptom
    :slot1 new-symptoms ) ]
;
00
00 Rule 6.14 checks that we did have only one new top symptom and that
00 it turns out to be the same as the original top symptom. This looks
00 like a case where we found a masked fault.
00 We also reclose all the switches above the culprit switch to enable
00 further testing of other possible faults below the top switch.
00 We determine if there are any other possible faults by calling
00 the function new-diagnosable-symptoms that looks for other fast
00 trips or current trips below the top switch and either siblings
00 of the switch that was the culprit or below the siblings of said
00 switch.
00
MF-rule6.14
[ lisp :: length ( new-top-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-top-symptoms
  < [ fault of symptom = fault of top-symptom ]
    [ switch of symptom = switch of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name found-below
    :top-symp top-symptom
    :slot1 which-switch ) ]
[ power-domain :: reclose-switches ( top-symptom which-switch ) ]
[ FOR ALL symptom in
  power-domain :: new-diagnosable-symptoms ( top-symptom which-switch )
  < [ the-individual-symptoms = the-individual-symptoms PLUS symptom ] > ]
;
00
```

@@ Rule 6.15 also checks that we got one new top symptom during the flips
 @@ of the lower switches. But in this case it is not the same as the
 @@ original top symptom. This is an unexpected situation and the user
 @@ will be notified.

```

@@
MF-rule6.15
[ lisp :: length ( new-top-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-top-symptoms
  < [ fault of symptom <> fault of top-symptom ]
    OR
    [ switch of symptom <> switch of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-different-top-after-flips
                                   :top-symp top-symptom
                                   :slot1 new-symptoms ) ]
;

```

@@ Rule 6.16 checks that we didn't get any new symptoms and therefore
 @@ we close the lower switches now.

```

@@
MF-rule6.16
[ lisp :: length ( new-symptoms ) = 0 ]
::>
[ new-symptoms = power-domain :: close-switches ( switches-below-to-test ) ]
;

```

@@ Rule 6.17 checks that we didn't get any new symptoms as a result of
 @@ closing the lower switches. If this is the case we determine that
 @@ we haven't yet found the fault at this level of testing. We set
 @@ switches-below to empty in preparation for the next level of testing.

```

@@
MF-rule6.17
[ lisp :: length ( new-symptoms ) = 0 ]
::>
[ switches-below = empty ]
;

```

@@ Rule 6.18 collects the next set of lower switches that might
 @@ be testable.

```

@@
MF-rule6.18
::>
[ FOR ALL switch in switches-below-to-test
  < [ switches-below = switches-below UNION switches-below of switch ] > ]
;
@@

```

@@ Rule 6.19 checks if there are no lower switches below to test.
 @@ If this is the case and if there were switches below the top switch
 @@ that we could not test we can diagnose the case as a not found case
 @@ in light of the fact that we also couldn't test some of the switches.
 @@

MF-rule6.19

```

[ lisp :: length ( switches-below ) = 0 ]
[ lisp :: length ( switches-below-cant-test ) > 0 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name not-found-cant-test-further
                                   :top-symp top-symptom
                                   :slot1 switches-below-cant-test ) ]
;

```

@@ Rule 6.20 checks that there are no lower switches below to test and
 @@ that we have tested everything that could be tested. This is a simply
 @@ not found case. Perhaps it was a transient?
 @@

MF-rule6.20

```

[ lisp :: length ( switches-below ) = 0 ]
[ lisp :: length ( switches-below-cant-test ) = 0 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name not-found-all-tested
                                   :top-symp top-symptom ) ]
;

```

@@ Rule 6.21 checks that there are lower switches that might be testable
 @@ and therefore sets up the variables so that we can go back to rule 6.9
 @@ and continue testing at the next level.
 @@

MF-rule6.21

```

[ lisp :: length ( switches-below ) > 0 ]
::>
[ switches-below-to-test = empty ]
[ level = level PLUS 1 ]
;

```

@@ Rule 6.22 determines that there were new symptoms as a result of
 @@ the close of the lower switches. We simply get the top symptoms
 @@ of the new symptoms in this rule.

MF-rule6.22

```

[ lisp :: length ( new-symptoms ) > 0 ]
::>
[ new-top-symptoms = power-domain :: top-symptoms ( new-symptoms ) ]
;
@@

```

@@ Rule 6.23 notes that we got a single top symptom and it is the
@@ same as the original top symptom. The difference between this
@@ rule and 6.14 is that this happened during the closes. The
@@ diagnosis needs to look at the topology to determine what might
@@ account for this (namely and over current trip and multiple
@@ lower switches drawing too much current in tandem while intermediate
@@ switches may have broken sensors or be the same rating of the
@@ top switch). If this is a fast trip case it doesn't make much
@@ sense.
@@ We don't do reclosing here like we do in 6.14 since this
@@ situation is not as well defined.
@@

```
MF-rule6.23
[ lisp :: length ( new-top-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-top-symptoms
  < [ fault of symptom = fault of top-symptom ]
    [ switch of symptom = switch of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name possible-found
                                   :top-symp top-symptom
                                   :slot1 which-switch
                                   :slot2 switches-below-to-test ) ]
;
```

@@
@@ Rule 6.24 checks that we got one new top symptom as a result of the
@@ closes of the lower switches but that this new symptom isn't the
@@ same as the original top symptom. This is an unexpected situation
@@ and the user will be notified.
@@

```
MF-rule6.24
[ lisp :: length ( new-top-symptoms ) = 1 ]
[ THERE EXISTS symptom in new-top-symptoms
  < [ fault of symptom <> fault of top-symptom ]
    OR
    [ switch of symptom <> switch of top-symptom ] > ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis
    ( :name unexpected-different-trip-during-closes
      :top-symp top-symptom :slot1 new-symptoms ) ]
;
```

@@
@@ Rule 6.25 detects the case where more than one new top symptom
@@ resulted from the close operation of the lower switches. This is
@@ also unexpected and the user will be notified.
@@

```
MF-rule6.25
```

```

[ lisp :: length ( new-top-symptoms ) > 1 ]
::>
[ diagnosis-set = diagnosis-set PLUS
  power-domain :: make-diagnosis ( :name unexpected-new-trips-during-closes
                                   :top-symp top-symptom
                                   :slot1 new-symptoms ) ]
@@
@@ The termination condition is setup by rule 5
@@
:
[ multiple-hard-fault-analysis = done ]

```

DONE

Diagnosis Rule Group

```

@@
@@ The Diagnosis Rule Group
@@
@@ This rule group takes a diagnosis and prints out relevant information about
@@ the diagnosis and sets up any out of service information on switches.
@@

```

RULE-GROUP : MF-Diagnosis

```

@@
@@ MF-diag-1 (no-power-to-bus)
@@ Diagnosed in MF-rule2.1
@@
MF-diag-1
[ diagnosis = no-power-to-bus ]
::>
[ power-domain :: write ( "The following switches tripped on under voltage:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( " ~a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ~a, the top sensor of the bus, registers less than the nominal"
      top-sensor of switch of symptom ) ]
    [ power-domain :: write ( "amount of voltage." ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write ( " Less than nominal voltage supplied to bus." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@

```

```

@@ MF-diag-2 (broken-cable-between-sensor-above-and-u-v-switches)
@@   Diagnosed in MF-rule2.2
@@
MF-diag-2
[ diagnosis = broken-cable-between-sensor-above-and-u-v-switches ]
::>
[ power-domain :: write ( "The following switches tripped on under voltage:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ^a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the top sensor of the bus, registers less nominal voltage."
      top-sensor of switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the sensor above the tripped switches, also registers"
      sensor-above of switch of symptom ) ]
    [ power-domain :: write ( "nominal voltage." ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( " Switch input cables to the switches disconnected in some fashion." ) ]
[ power-domain :: write ( " Less Likely:" ) ]
[ power-domain :: write
  ( " Break in cable between the sensor above the tripped switches and the bus" ) ]
[ power-domain :: write ( "of the tripped switches." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-3 (broke-output-cable-of-switch-above)
@@   Diagnosed in MF-rule2.3
@@
MF-diag-3
[ diagnosis = broke-output-cable-of-switch-above ]
::>
[ power-domain :: write ( "The following switches tripped on under voltage:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ^a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the top sensor of the bus, registers nominal voltage."
      top-sensor of switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the sensor above the tripped switches, registers less"
      sensor-above of switch of symptom ) ]

```

```

    [ power-domain :: write ( "than nominal voltage" ) ]
    [ power-domain :: write
      ( "while, ~a, the switch above the tripped switches, registers a nominal voltage."
        switch-above of switch of symptom ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( " The switch output cable of the switch above the tripped switches has" ) ]
[ power-domain :: write ( "been disconnected in some fashion." ) ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: out-of-service
    ( switch-above of switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-4 (broke-input-cable-of-switch-above)
@@ Diagnosed in MF-rule2.4
@@
MF-diag-4
[ diagnosis = broke-input-cable-of-switch-above ]
::>
[ power-domain :: write ( "The following switches tripped on under voltage:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( " ~a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ~a, the top sensor of the bus, registers nominal voltage."
      top-sensor of switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ~a, the sensor above the tripped switches, registers less than"
      sensor-above of switch of symptom ) ]
    [ power-domain :: write ( "nominal voltage," ) ]
    [ power-domain :: write
      ( "however, ~a, the sensor above the switch above the tripped switches also"
        sensor-above of switch-above of switch of symptom ) ]
    [ power-domain :: write ( "registers less than nominal voltage and" ) ]
    [ power-domain :: write
      ( "~a, the switch above the tripped switches, cannot trip on under voltage."
        switch-above of switch of symptom ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( " The switch input or output cable of the switch above the tripped switches" ) ]
[ power-domain :: write ( "has been disconnected in some fashion." ) ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: out-of-service
    ( switch-above of switch of symptom ) ] > ]

```

```

[ diagnosis = :unknown ]
;
@@
@@ MF-diag-5 (break-in-cable-above-switch-above-and-bad-u-v-sensor-switch-above)
@@   Diagnosed in MF-rule2.5
@@
MF-diag-5
[ diagnosis =
  break-in-cable-above-switch-above-and-bad-u-v-sensor-switch-above ]
::>
[ power-domain :: write ( "The following switches tripped on under voltage:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ^a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the top sensor of the bus, registers nominal voltage."
      top-sensor of switch of symptom ) ] > ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: write
    ( "Sensor ^a, the sensor above the tripped switches, registers less than"
      sensor-above of switch of symptom ) ]
    [ power-domain :: write ( "nominal voltage," ) ]
    [ power-domain :: write
      ( "however, ^a, the sensor above the switch above the tripped switches also"
        sensor-above of switch-above of switch of symptom ) ]
    [ power-domain :: write ( "registers less than nominal voltage and" ) ]
    [ power-domain :: write
      ( " ^a, the switch above the tripped switches, should have tripped but did not."
        switch-above of switch of symptom ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( " The switch input cable of the switch above the tripped switches has been" ) ]
[ power-domain :: write ( "disconnected in some fashion and" ) ]
[ power-domain :: write ( " also has a bad under voltage sensor." ) ]
[ THERE EXISTS symptom in top-symptoms
  < [ power-domain :: out-of-service
    ( switch-above of switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-6 (no-permission-to-test-possible-backrush)
@@   Diagnosed in MF-rule3.2.3
@@
MF-diag-6
[ diagnosis = no-permission-to-test-possible-backrush ]
::>

```

```

[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ~a" switch of symptom ) ] > ]
[ power-domain :: write ( "None of the switches has permission to test." ) ]
[ power-domain :: write
  ( "It is possible that these switches tripped on fast trip due to a low impedance" ) ]
[ power-domain :: write
  ( "short below one of them and the others due to energy storage in the loads." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-7 (unexpected-to-many-retrips-possible-backrush)
@@   Diagnosed in MF-rule3.2.4
@@
MF-diag-7
[ diagnosis = unexpected-to-many-retrips-possible-backrush ]
::>
[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ~a" switch of symptom ) ] > ]
[ power-domain :: write
  ( "During testing by flipping the switches the following symptoms occurred:" ) ]
[ FOR ALL symptom in new-symptoms
  < [ power-domain :: write ( "~a on ~a" switch of symptom fault of symptom ) ] > ]
[ power-domain :: write
  ( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-8 (no-retrips-on-flips-possible-backrush)
@@   Diagnosed in MF-rule3.2.5
@@
MF-diag-8
[ diagnosis = no-retrips-on-flips-possible-backrush ]
[ lisp :: length ( switches-to-test ) = lisp :: length ( top-symptoms ) ]
::>
[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ~a" switch of symptom ) ] > ]
[ power-domain :: write ( "None of the switches retripped during testing." ) ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write ( "  A low impedance short that was burned clear." ) ]
[ power-domain :: write ( "  A transient in a load below one of the switches." ) ]

```

```

[ power-domain :: write
  ( "Cause of other switches tripping could be due to energy storage in the loads" ) ]
[ power-domain :: write ( "below them." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-9 (no-retrips-on-flips-possible-backrush)
@@   Diagnosed in MF-rule3.2.5
@@
MF-diag-9
[ diagnosis = no-retrips-on-flips-possible-backrush ]
[ lisp :: length ( switches-to-test ) < lisp :: length ( top-symptoms ) ]
::>
[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ^a" switch of symptom ) ] > ]
[ power-domain :: write ( "None of the switches retripped during testing." ) ]
[ power-domain :: write
  ( "However, only the following switches had permission to test:" ) ]
[ FOR ALL switch in switches-to-test
  < [ power-domain :: write ( "  ^a" switch ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( "  A low impedance short below one of the switches that was not tested." ) ]
[ power-domain :: write
  ( "Cause of other switches tripping could be due to energy storage in the loads" ) ]
[ power-domain :: write ( "below them." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;
@@
@@ MF-diag-10 (found-possible-backrush)
@@   Diagnosed in MF-rule3.2.6
@@
MF-diag-10
[ diagnosis = found-possible-backrush ]
[ lisp :: length ( switches-to-test ) < lisp :: length ( top-symptoms ) ]
::>
[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( "  ^a" switch of symptom ) ] > ]
[ THERE EXISTS symptom in new-symptoms
  < [ power-domain :: write
    ( "During testing of the switches, ^a, retripped on fast trip."

```

```

        switch of symptom ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ THERE EXISTS symptom in new-symptoms
  < [ power-domain :: write
    ( " Low impedance short below ~a." switch of symptom ) ] > ]
[ power-domain :: write
( "Cause of other switches tripping due to energy storage in the loads below them." ) ]
[ THERE EXISTS symptom in new-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ]
    [ FOR ALL symptom1 in top-symptoms
      WHERE [ switch of symptom1 <> switch of symptom ]
        < [ power-domain :: out-of-service ( switch of symptom1 ) ] > ] > ]
[ diagnosis = :unknown ]
;

@@
@@ MF-diag-11 (unexpected-retrip-possible-backrush)
@@ Diagnosed in MF-rule3.2.7
@@
MF-diag-11
[ diagnosis = unexpected-retrip-possible-backrush ]
::>
[ power-domain :: write ( "The following switches tripped on fast trip:" ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: write ( " ~a" switch of symptom ) ] > ]
[ power-domain :: write
( "During testing by flipping the switches the following symptoms occurred:" ) ]
[ FOR ALL symptom in new-symptoms
  < [ power-domain :: write ( " ~a on ~a" switch of symptom fault of symptom ) ] > ]
[ power-domain :: write
( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ FOR ALL symptom in top-symptoms
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ]
[ diagnosis = :unknown ]
;

@@
@@ MF-diag-12 (unexpected-symptoms-during-open)
@@ Diagnosed in MF-rule6.2
@@
MF-diag-12
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = unexpected-symptoms-during-open ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "~a tripped on ~a."
  switch of top-symp of diagnosis
  fault of top-symp of diagnosis ) ]

```

```
[ power-domain :: write
  ( "During opening of the switches for testing the following syptoms occurred:" ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: write
    ( " ~a on ~a" switch of symptom fault of symptom ) ] > ]
[ power-domain :: write
  ( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
@@
@@ MF-diag-13 (unexpected-too-many-symptoms-flip-top)
@@   Diagnosed in MF-rule6.4
@@
MF-diag-13
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = unexpected-too-many-symptoms-flip-top ]
< ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "~a tripped on ~a."
    switch of top-symp of diagnosis
    fault of top-symp of diagnosis ) ]
  [ power-domain :: write
    ( "During flipping of the top switch for testing the following syptoms occurred:" ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: write
      ( " ~a on ~a" switch of symptom fault of symptom ) ] > ]
  [ power-domain :: write
    ( "This is not a situation that is diagnosable in the existing rule set." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
@@
@@ MF-diag-14 (retrip-on-flip)
@@   Diagnosed in MF-rule6.5
@@
@@ Need to look at the possibility of some bottom level switches below
@@ the top switch that may have tripped on fast trip due to energy storage.
@@
MF-diag-14
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = retrip-on-flip ]
< [ fault of top-symp of diagnosis = over-current ]
::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "~a tripped on ~a." ]
```

```

        switch of top-symp of diagnosis
        fault of top-symp of diagnosis ) ]
[ power-domain :: write ( "During testing ~a tripped on ~a."
        switch of top-symp of diagnosis
        fault of top-symp of diagnosis ) ]
@@ high-impedance or low-impedance (over-current or fast-trip)
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
( " High impedance short in cable below switch, switch output of switch, or the" ) ]
[ power-domain :: write ( "switch input of one of the lower switches." ) ]
[ power-domain :: write ( " Less Likely:" ) ]
[ power-domain :: write ( " Current sensor in switch reading high." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
@@
@@ MF-diag-15 (retrip-on-flip)
@@ Diagnosed in MF-rule6.5
@@
@@ Need to look at the possibility of some bottom level switches below
@@ the top switch that may have tripped on fast trip due to energy storage.
@@
MF-diag-15
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = retrip-on-flip ]
< [ fault of top-symp of diagnosis = fast-trip ]
::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "~a tripped on ~a."
        switch of top-symp of diagnosis
        fault of top-symp of diagnosis ) ]
[ power-domain :: write ( "During testing ~a tripped on ~a."
        switch of top-symp of diagnosis
        fault of top-symp of diagnosis ) ]
@@ high-impedance or low-impedance (over-current or fast-trip)
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
( " Low impedance short in cable below switch, switch output of switch, or the" ) ]
[ power-domain :: write ( "switch input of one of the lower switches." ) ]
[ power-domain :: write ( " Less Likely:" ) ]
[ power-domain :: write ( " Current sensor in switch reading high." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
@@
@@ MF-diag-16 (unexpected-retrip-during-flip)
@@ Diagnosed in MF-rule6.6

```

00

```
MF-diag-16
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = unexpected-retrip-during-flip ]
< ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "'a tripped on 'a."
                           switch of top-symp of diagnosis
                           fault of top-symp of diagnosis ) ]
  [ power-domain :: write
    ( "During flipping of the top switch for testing the following symptom occurred:" ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: write
      ( " 'a on 'a" switch of symptom fault of symptom ) ] > ]
  [ power-domain :: write
    ( "This is not a situation that is diagnosable in the existing rule set." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
```

00

00 MF-diag-17 (not-found-no-levels)

00 Diagnosed in MF-rule6.7

00

```
MF-diag-17
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = not-found-no-levels ]
< ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "'a tripped on 'a."
                           switch of top-symp of diagnosis
                           fault of top-symp of diagnosis ) ]
  [ power-domain :: write
    ( "The switch did not retrip during testing and there are no switches below" ) ]
  [ power-domain :: write ( "this switch." ) ]
  [ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
  [ power-domain :: write ( " Most Likely:" ) ]
  [ power-domain :: write ( " A temporary short that was burned clear." ) ]
  [ power-domain :: write ( " A transient in the load below the switch." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
```

00

00 MF-diag-18 (unexpected-trips-during-close-top)

00 Diagnosed in MF-rule6.8.1

00

```
MF-diag-18
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = unexpected-trips-during-close-top ]
```

```

< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "~a tripped on ~a."
      switch of top-symp of diagnosis
      fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "During the close of the top switch for subsequent testing the following" ) ]
[ power-domain :: write ( "syntom occurred:" ) ]
[ FOR ALL syntom in slot1 of diagnosis
  < [ power-domain :: write
    ( " ~a on ~a" switch of syntom fault of syntom ) ] > ]
[ power-domain :: write
  ( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
[ FOR ALL syntom in slot1 of diagnosis
  < [ power-domain :: out-of-service ( switch of syntom ) ] > ] >
;
@@
@@ MF-diag-19 (not-found-cant-test-further)
@@ Diagnosed in MF-rule6.11 and MF-rule6.19
@@
MF-diag-19
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = not-found-cant-test-further ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "~a tripped on ~a."
      switch of top-symp of diagnosis
      fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "The fault has not been repeated (and therefore not found)." ) ]
[ power-domain :: write ( "The following switches cannot be tested:" ) ]
[ FOR ALL switch in slot1 of diagnosis
  < [ power-domain :: write ( " ~a" switch ) ] > ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write ( " A transient short somewhere below ~a."
      switch of top-symp of diagnosis ) ]
[ power-domain :: write
  ( " A short below one of the switches that were not testable." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
@@
@@ MF-diag-20 (unexpected-to-many-tops-after-flips)
@@ Diagnosed in MF-rule6.13
@@
MF-diag-20
FOR ALL diagnosis in diagnosis-set

```

```

WHERE [ name of diagnosis = unexpected-to-many-tops-after-flips ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "'a tripped on 'a."
                        switch of top-symp of diagnosis
                        fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "During testing of the switches below 'a, the following symptoms occurred:"
    switch of top-symp of diagnosis ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: write
    ( "'a on 'a" switch of symptom fault of symptom ) ] > ]
[ power-domain :: write
  ( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;

@@
@@ MF-diag-21 (found-below)
@@ Diagnosed in MF-rule6.14
@@
MF-diag-21
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = found-below ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "'a tripped on 'a."
                        switch of top-symp of diagnosis
                        fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "During testing of the lower switches the fault was repeated when 'a was flipped."
    slot1 of diagnosis ) ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write
  ( " A short below 'a and (if the switches between 'a and 'a are of the same"
    slot1 of diagnosis slot1 of diagnosis switch of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "rating) a race to determine which switch actually trips." ) ]
[ power-domain :: write
  ( " Otherwise, the switches between 'a and 'a have failed current sensors"
    slot1 of diagnosis switch of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "in addition to the short below 'a." slot1 of diagnosis ) ]
[ power-domain :: out-of-service ( slot1 of diagnosis ) ] >
;

```

```

**
** MF-diag-22 (unexpected-different-top-after-flips)
** Diagnosed in MF-rule6.15
**
MF-diag-22
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = unexpected-different-top-after-flips ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "a tripped on a."
                          switch of top-symp of diagnosis
                          fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "During testing of the lower switches the following symptoms occurred:" ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: write
    ( " a on a" switch of symptom fault of symptom ) ] > ]
[ power-domain :: write
  ( "This is not a situation that is diagnosable in the existing rule set." ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
[ FOR ALL symptom in slot1 of diagnosis
  < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
**
** MF-diag-23 (not-found-all-tested)
** Diagnosed in MF-rule6.20
**
MF-diag-23
FOR ALL diagnosis in diagnosis-set
WHERE [ name of diagnosis = not-found-all-tested ]
< ::>
[ diagnosis-set = diagnosis-set MINUS diagnosis ]
[ power-domain :: write ( "a tripped on a."
                          switch of top-symp of diagnosis
                          fault of top-symp of diagnosis ) ]
[ power-domain :: write
  ( "The fault has not been repeated (and therefore not found)." ) ]
[ power-domain :: write ( "All the testing that is possible has been done." ) ]
[ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
[ power-domain :: write ( " Most Likely:" ) ]
[ power-domain :: write ( " A transient short somewhere below a."
                          switch of top-symp of diagnosis ) ]
[ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
**
** MF-diag-24 (possible-found)
** Diagnosed in MF-rule6.23
**

```

SSM/PMAD Final Report

```

MF-diag-24
FOR ALL diagnosis in diagnosis-set
  WHERE [ name of diagnosis = possible-found ]
  < [ fault of top-symp of diagnosis = over-current ]
  ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "'a tripped on 'a."
    switch of top-symp of diagnosis
    fault of top-symp of diagnosis ) ]
  [ power-domain :: write
  ( "During testing of the lower switches the fault was repeated when 'a was closed."
    slot1 of diagnosis ) ]
  [ power-domain :: write ( "POSSIBLE CAUSES:" ) ]
  [ power-domain :: write ( " Most Likely:" ) ]
  [ power-domain :: write
  ( " This may be indicative of a an over-current fault being generated by too" ) ]
  [ power-domain :: write ( "many loads all in over-current in tandem." ) ]
  [ power-domain :: write
  ( "Since this situation is not very likely, 'a is still being declared"
    switch of top-symp of diagnosis ) ]
  [ power-domain :: write ( "out of service." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
cc
cc MF-diag-25 (possible-found)
cc Diagnosed in MF-rule6.23
cc
MF-diag-25
FOR ALL diagnosis in diagnosis-set
  WHERE [ name of diagnosis = possible-found ]
  < [ fault of top-symp of diagnosis = fast-trip ]
  ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "'a tripped on 'a."
    switch of top-symp of diagnosis
    fault of top-symp of diagnosis ) ]
  [ power-domain :: write
  ( "During testing of the lower switches the fault was repeated when 'a was closed."
    slot1 of diagnosis ) ]
  [ power-domain :: write ( "This should not be possible." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ] >
;
cc
cc MF-diag-26 (unexpected-different-trip-during-closes)
cc Diagnosed in MF-rule6.24
cc
MF-diag-26
FOR ALL diagnosis in diagnosis-set

```

```

WHERE [ name of diagnosis = unexpected-different-trip-during-closes ]
< ::>
  [ diagnosis-set = diagnosis-set MINUS diagnosis ]
  [ power-domain :: write ( "a tripped on a."
                           switch of top-symp of diagnosis
                           fault of top-symp of diagnosis ) ]
  [ power-domain :: write
    ( "During testing of the lower switches the following symptoms occurred:" ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: write
      ( " a on a" switch of symptom fault of symptom ) ] > ]
  [ power-domain :: write
    ( "This is not a situation that is diagnosable in the existing rule set." ) ]
  [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
  [ FOR ALL symptom in slot1 of diagnosis
    < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
00
00 MF-diag-27 (unexpected-new-trips-during-closes)
00   Diagnosed in MF-rule6.25
00
MF-diag-27
FOR ALL diagnosis in diagnosis-set
  WHERE [ name of diagnosis = unexpected-new-trips-during-closes ]
  < ::>
    [ diagnosis-set = diagnosis-set MINUS diagnosis ]
    [ power-domain :: write ( "a tripped on a."
                             switch of top-symp of diagnosis
                             fault of top-symp of diagnosis ) ]
    [ power-domain :: write
      ( "During testing of the lower switches the following symptoms occurred:" ) ]
    [ FOR ALL symptom in slot1 of diagnosis
      < [ power-domain :: write
        ( " a on a" switch of symptom fault of symptom ) ] > ]
    [ power-domain :: write
      ( "This is not a situation that is diagnosable in the existing rule set." ) ]
    [ power-domain :: out-of-service ( switch of top-symp of diagnosis ) ]
    [ FOR ALL symptom in slot1 of diagnosis
      < [ power-domain :: out-of-service ( switch of symptom ) ] > ] >
;
  [ diagnosis = :unknown ]
  [ diagnosis-set = empty ]
DONE

```

2.6.7 The SSM/PMAD Seamless User Environment Interface

The SSM/PMAD system user interface allows a power system user the ability to perform power management tasks with a minimum of computer knowledge. The present user interface is the culmination of several years of development and interface iteration. The first interface was developed on a Xerox 1186. This interface was very primitive and was not easy to use. Significant changes were made in the second version of the interface which somewhat clarified the picture of the power system, but the system was still difficult to use and was too bright and colorful. It would grate on the user's eyes, if used for any length of time. At this point, the user interface underwent review within the engineering and cognitive science domains. This yielded several important suggestions which greatly improved the readability and usability of the user interface. These changes were implemented in the latest version of the SSM/PMAD user interface. This latest version of the user interface has reached the level of maturity and sophistication needed to effectively show the power system in operation and to provide a user the ILA functionality with understandable confidence.

In designing the latest version of SSM/PMAD user interface, several types of interface objects needed for system interaction were defined. These objects being windows, workboxes, buttons, and menus. A window is defined as a rectangular area on the user interface screen where graphic or textual information may be displayed. A workbox is defined as a window within which the system prompts the user for information. Workboxes are dynamic with their context being driven transparently to the user. A button is defined as a rectangular region on the user interface which, when selected, initiates an interface response. Lastly, a menu is defined as a list of possible selections which are accessed through mouse cursor movement and button clicks. With this set of object definitions and added graphics, it is possible to define the structure of the present SSM/PMAD user interface.

The SSM/PMAD user interface contains a system screens selection menu, a system options button, a time bar window, a system mode window, a system flow window, a messages window, a focused messages window, an application window, and the graphics objects which represent the SSM/PMAD components and structure. The application window is further divided into an application main window, an application menu, and a secondary window. The interface layout is shown in Figure 2.19.

The screens selection menu enables the user to switch between application windows, with the application window presently being displayed not available for selection. If the user selects an application on the screen selections menu, the user interface displays the selected application to the appropriate application screen. The context for the user view is updated automatically with no prompting to the user.

A system options button brings up a menu of various system level operations. These operations include mode changes, scheduler functions, other system operations, utilities, and help. A user may change the system mode of operation to idle, normal, or maintenance. The user may also choose to read a schedule, activate a schedule, or halt an active schedule

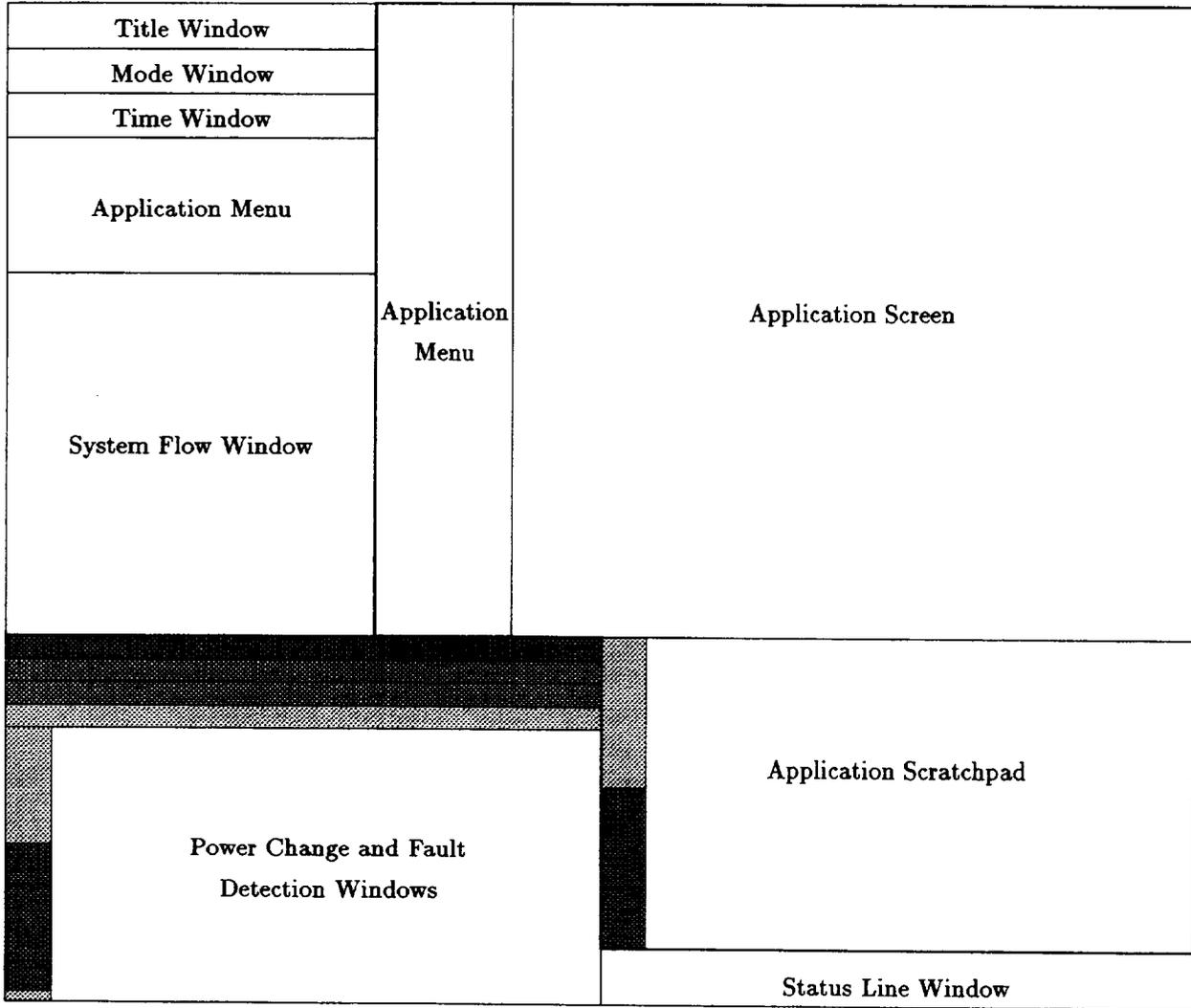


Figure 2.19: SSM/PMAD User Interface Layout

under system options. There are other system operations providing connections to software and hardware, as well as help and utility functions. The help function is context driven by where the user is in the overall system execution. The utility functions allow the user to focus for more detailed information display, to enable and disable system functions, and to output information to selected devices. All these options provide the user with significant capability within the real-time framework of the SSM/PMAD test bed.

The time bar window contains both Greenwich Mean Time (GMT) and SSM/PMAD mission time. Both times are updated on a one minute basis. The mission time is used in the scheduling system. Mission time is adjustable to occur at different times with respect to GMT.

The mode indicator window tells the user in which mode the system is presently operating. The modes of operation are normal, idle, and maintenance. In normal mode, a user may bring up an autonomous schedule of power system operations and simultaneously take control of a set of switches for manually controlled operations. In idle mode the system is not connected to the lowest level processors and therefore the power system. This mode is a startup and shutdown mode. In maintenance mode, the user is in complete control of the system with all of the commensurate responsibilities.

The system flow window shows the user system software component interaction. For example, if a user selects the system option, GO TO NORMAL, the KANT graphic representation in the System Flow Menu highlights and the user interface display context is updated automatically. All necessary connections and operations are initiated for the user automatically, relieving the user of these tedious activities.

The messages window and focused messages window present the user system warnings, fault diagnoses, general error messages, and other unsolicited information. This provides the user with the power to select information for display and, in some cases, to what detail the information should be provided.

The application window is the only window in the system which is dynamic. Only one application window may be visible at a given time. If a different application screen is selected from the screen selections menu, it will take the place of the previous application screen. All the other windows are static, they are always available no matter which application screen is visible. This feature allows a user to access and monitor system level information, while concurrently working in a particular application.

The application window contains an application main window, an application main menu, and a secondary window. Buttons within the application menu apply to the application in which they exist. These application menu button functions may return solicited data (e.g. help) which is displayed in the secondary window or they might somehow change the viewpoint from which the application views data (e.g. Change Window of Time). Following are descriptions of the various applications windows.

Power System

The Power System screen contains a topological map of the power system. The application screen presents power busses, switches, and sensors as representative graphics and icons. The Power System screen was designed to be the primary application on display during operation of the SSM/PMAD. Because of this, its application menu contains options with limited capabilities of some of the other applications. When a user observes something interesting on the Power System screen, he may obtain ancillary information by using the options in the application menu and displaying the information to the secondary window. If the user still has not obtained enough information on the event observed, switching to a more appropriate application is still an option for more information.

Power System Topology

The SSM/PMAD autonomous power system contains two power busses, port and starboard, and each feeds a Power Distribution Control Unit (PDCU). Both PDCUs distribute power to each load center and the interface is capable of displaying up to five load centers although there are only three presently. Up to five switches may be displayed per bus per load center on the interface. Redundant sourcing of an individual load is only permitted within a load center and is displayed that way at the user interface.

Power System Representation

Graphical pipes represent the power busses on the interface. A filled graphical pipe represents an energized bus, and an empty pipe represents a non-energized bus. Every switch icon contains a multiplicity of information, including its type, state, status, amount of current flowing through the switch, scheduled current for the switch, fault type, whether it is under manual control, whether it is supplying current to a redundantly sourced load, and whether it has been selected for interface operations. Each switch looks like a relay made with graphical pipes which overlays a geometric shape. The geometric shapes indicate the type of switch, a diamond indicates a Remote Bus Isolator (RBI), an oval indicates a three kilowatt Remote Power Controller (RPC), and a rectangle indicates a one kilowatt RPC. The color of the geometric shape expresses the state of the switch, with green being normal, red being faulted, and brown being out of service. Each closed switch icon contains an analog bar which represents the current flowing through the RPC versus maximum current capacity. This analog bar also has a scheduled current indicator for the switch. The switch label and current flowing through the switch are displayed on the geometric background. If the switch is tripped or out of service, an abbreviation for the fault type which caused the problem is displayed in place of the current. If the switch has been placed under manual control, a hand icon points to the switch and partially overlays the switch. When a switch supplies current to a redundantly sourced load, a redundant icon will appear below the switch and

its redundant partner within a load center. The primary switch in a redundant pair will have a redundant icon with black foreground and tan background. The secondary switch in a redundant pair will have the same icon pattern, but a black background and a tan foreground. If the switch has been selected for interface operations, a checkmark icon will partially overlay the switch. Hollow circles represent sensors on the interface. Sensors may be selected with a button click in the same manner as switches. When a sensor is selected, the sensor is overlaid with a selection checkmark.

FELES

The Front End Loads Enable Scheduler (FELES) application screen displays the schedule presently being run by the SSM/PMAD system. The schedule is displayed in the form of a gantt chart which shows when an activity is running in a temporal context. Information about an activity may be obtained by selecting an activity and asking for load information. The schedule being executed at a selected load center may also be displayed on this application screen.

Power Utilization

The Power Utilization application screen displays plots of system power usage and system power availability versus time. The power usage plots contain a plot of scheduled power usage versus time which is overlaid by the actual system power usage.

Activity Editor

The Activity Editor application allows the user to modify existing activities, subtasks, requirements, and powered equipment. The user may also create new activities, subtasks, requirements, and powered equipment. The user may create new schedules for operation in normal mode or delete an existing schedule from the set of available schedules. In modifying a schedule component (activity, subtask, requirement, powered equipment) presently existing schedules may require rescheduling if their components have been changed.

Background and Purpose The ability to create new activities became increasingly more important as the SSM/PMAD testbed evolved. Modifying existing schedules to incorporate changes that model the Space Station more accurately was equally important. The creation and modification of activities allow the user to generate and demonstrate Space Station mission scenarios using the SSM/PMAD testbed.

To support the activity editor and its purpose, MAESTRO was ported to the Solbourne. Common data structures were created for MAESTRO and the rest of the system to share. These data structures allow changes made by the activity editor to be seen immediately by the scheduler without having to read the scheduling files into memory.

Definition of Activity The activity editor creates and modifies activities. Each activity accomplishes a goal using a set of resources and a set of actions which execute in some sequence. MAESTRO defines an object called a subtask for contiguous actions on a uniform set of resources. This subtask may require that a particular environment state be maintained and may be dependent on the execution of other subtasks. An activity is defined as a linear, non-overlapping sequence of one or more subtasks.

Scheduling is the process of specifying the start and end times for the subtasks in activities. A valid schedule is a specification of start and end times for subtasks such that all activities may be successfully executed by the power system breadboard.

The user may want an activity performed more than once. This is typically the case with experiments in space, where the cost of getting equipment in orbit is so expensive that a single execution of an activity is not cost-effective. MAESTRO has the ability to schedule multiple performances of activities.

An activity group in MAESTRO contains multiple activities which accomplish the same goal. An activity group in MAESTRO has a primary model and zero or more alternative models for accomplishing the same goal. To reduce complexity, a design decision was made to choose only the primary model in the initial design of the activity editor.

One of the goals of the activity editor interface is to minimize input without curtailing the ability to create complex activities. To minimize input, defaults are used for some of the subtask fields.

MAESTRO MAESTRO schedules with respect to both resource and temporal constraints. During each placement cycle, for each activity MAESTRO first performs resource opportunity calculation, then temporal constraint propagation. Resource opportunity calculation narrows subtask placement choices to include only those whose resource availabilities can support each subtask's resource requirements. Temporal constraint propagation further narrows subtask placement choices such that choosing any subtask start or end time from those specified will allow the whole activity to be scheduled. For a more detailed description of MAESTRO see subsection 2.6.2 and Appendix C.

For resource opportunity calculation, the user can enter resource and environmental constraint information into an activity. This is done by adding requirements for powered equipment and non-powered resources to accomplish the activity.

Since powered equipment is the most important resource that we schedule on the SSM/PMAD testbed, the majority of all subtasks that are created have at least one piece of powered equipment. Because the power system topology can change, it was decided that the user should only specify powered equipment and not the power system resources to support that equipment. That way a change in how power gets to the equipment does not necessitate editing the activities using that equipment. The system automatically determines the correct power system resources that will support each subtask.

Resources can be categorized as rate controlled, equipment, consumables, conditions,

and alterable conditions. Currently, rate controlled resources and equipment are the only resources that can be created using the activity editor. Conditions and alterable conditions are ways of modeling the user's environment such as daylight availability and vibration. Conditions and alterable conditions have not been incorporated yet because they have not been used, and their complexity of creation did not warrant inclusion into the current activity editor. Consumable resources will be incorporated in the activity editor when time permits.

As mentioned above, after resource opportunity calculation MAESTRO performs temporal constraint propagation. The only timing constraints currently implemented are delays between subtasks and durations of subtasks. An attribute field called *temporal constraint* has been added to activities which allows the user to specify temporal constraints between activities. For example, a time dependency between activity A and activity B can be created by declaring a time dependency of ten minutes. This means that activity A must run within ten minutes of activity B. Initially, the time dependency field is set to nil in all activities, because time dependency is a relatively new feature to MAESTRO and its usefulness has not been fully determined. The user is able to do the same thing with a subtask within an activity, so the concept is being abstracted to another area of the system.

SSM/PMAD Requirements From the previous section, it is apparent that an activity can be quite complex. The main goal of the activity editor was to try to give the user an easy way to create and update schedules that are run on the SSM/PMAD testbed. In order to do this, the activity editor must meet three basic criteria:

- Minimize the input
- Layer the data
- Reuse the basic building blocks

Minimizing input is achieved by allowing the user to have access only to those characteristics of an activity that are associated with scheduling *electrical power*. This meant that a lot of activity information that MAESTRO could use was either set to a default values or omitted altogether.

Layering the data is a natural extension of the current activity editor data structure. A method called expand/unexpand has been devised to allow the user to see the different layers that are requested. This allows the user to view the data in a very broad sense, for example, the subtasks within an activity. It also allows the user to view the data in a very narrow sense, for example, what are the mode parameters for a piece of powered equipment.

Reusing the basic building blocks is harder to implement because the idea of reusing subtasks is not a part of the MAESTRO scheduling system. There are four basic types of data that can be re-used to build a schedule.

- Powered equipment whose use is essential in building subtasks.
- Resources whose use is essential in building subtasks.
- Subtasks which are the essential part of an activity.
- Activities which are used in building a schedule.

To reuse any of the above data items, new fields had to be added to the current MAESTRO data structures. The new fields allow the user to keep track of who is using what and where and guarantee the integrity of the schedules. An example would be a subtask that is used in two different activities and those activities are used in five different schedules. If the user were to edit this subtask and change some of the information, this would change the meaning of the two activities which might consequently invalidate all five schedules. This information must be maintained along with the activity in order to inform the user of any conflicts and resolve them before they crash the system.

Implementing the above three criteria simplifies the creation of a schedule by the user. The new method of creating schedules is much simpler than the old method of creating and maintaining schedules that existed on the MAESTRO system. Considerable effort was made to meet the needs of the power system and at the same time utilize the functionality that exists in MAESTRO.

The activity editor is still in a development stage. Therefore we encourage user feedback in order to determine how the functionality of the activity editor can be improved. A future consideration might be to add the power functionality that MAESTRO has for scheduling. At the current time, the user chooses the activities they would like to have scheduled and the time interval. The latest MAESTRO tools allow placement of a chosen activity onto the schedule and then scheduling around the placement along with many other features.

Window Layout To get to the activity editor, the user selects the "Activity Editor" menu item from the available applications. The Activity Editor is only available during idle and maintenance mode. Because the same data structures are used by MAESTRO and SSM/PMAD, allowing the user to edit during normal mode could denigrate the integrity of the system. When the activity editor is first invoked, it will look like Figure 2.20.

Like all other applications, the activity editor has an applications menu and a scratchpad window. The scratchpad window provides the user with textual data that is either user requested information or activity editor error information. The application menu contains the main functions for the activity editor. These function are:

- Edit
- Create

System Options	Activity Editor	
Edit	Description: <input type="text" value="The operation of the pumps to supply cabin air."/>	
Create	Name: <input type="text" value="Cabin Air Supply"/>	
Delete	Base Priority: <input type="text" value="0"/>	
Save	Performances Requested: <input type="text" value="100"/>	
Create Schedule	Minimum Performances Needed: <input type="text" value="1"/>	
Delete Schedule	Subtask: Main Cabin Air Supply 1	
Help	Max Duration: <input type="text" value="129600"/>	
	Min Duration: <input type="text" value="1"/>	
	Maximum Delay: <input type="text" value="0"/>	
	Minimum Delay: <input type="text" value="0"/>	
	Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/>	
	Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input checked="" type="radio" value="On"/> Location: <input type="text" value="LC-3-4"/>	
	Powered Equipment: <input type="text" value="*****Undefined*****"/> Mode: <input checked="" type="radio" value="On"/> Location: <input type="text" value="*****"/>	

Figure 2.20: Activity Editor Window

Left, Middle, or Right Mouse Button to Edit This Field

Figure 2.21: Activity Editor Status Line Category I

Left, Middle, or Right Mouse Button NOT VALID

Figure 2.22: Activity Editor Status Line Category II

- Delete
- Save
- Create Schedule
- Delete Schedule
- Help

The implementation of each function is discussed in subsequent sections of this manual. The main window of the activity editor screen is broken down into four parts.

- The first part contains the title "Activity Editor" and is only information.
- The second part is the main window used by the activity editor. It is a scrolling window that is used to display the data that the user can edit.
- The third part is a window that displays the status information about the mouse functions that are available for the particular fields within the main window. An example of the status line can be found in Figure 2.21. When the mouse is located within a data field, the status line shows the user what each mouse button will do for that field. This status line appears when the description field is entered. Figure 2.22 appears when the name field is entered. Not valid indicates this is not an editable field. Figure 2.23 appears when the subtask field is entered. All data fields will have one of the three different status line shown in Figure 2.21 through Figure 2.23.

Left: Edit, Middle: Expand/Unexpand, Right: Menu Options

Figure 2.23: Activity Editor Status Line Category III

System Options	Activity Editor	
Edit	<div style="background-color: #cccccc; padding: 5px; border: 1px solid black;"> Edit Activity Subtask Requirement Powered Equipment </div>	
Create		
Delete		

Figure 2.24: Activity Editor Edit Menu

All data displayed in the main window of the activity editor screen comes in the same general format. First comes the title for the data followed by the data itself. The bounds of the data is discriminated using a box that has a blue background color. A feature known as highlighting is used to let the user know when the mouse has entered the editable region of a data field. The highlight causes the background color of the box to turn red. When the mouse leaves the editable region, the background color will return to its red color. At the same time as the background color of the box is turning red, the status line is being updated to reflect the editing information for that editable region.

Viewing and Editing the Data at Different Levels To understand what editing capabilities the activity editor has, the user chooses data to edit. To do this the user selects the Edit option on the applications menu. This will cause a menu to appear in the upper left hand corner of the application main window. Refer to Figure 2.24 to see an example of this menu.

Notice that there are four entries in the menu:

- Activity
- Subtask
- Requirements
- Powered Equipment

From an earlier discussion, the user will remember that requirements and powered equipment are resource building blocks used in the creation of subtasks, which are used to build activities, which are used to build schedules. The activity editor gives the user the ability to edit each item directly or as a part of an encompassing item. For example, if there is a subtask called FOO which is used in activity BAR, the user can edit the subtask FOO as an independent object or as part of the activity BAR. The user cannot edit a schedule directly, only the components that make up a schedule. This is why schedule is not a menu item in the edit window.

Each item in the edit menu has a pull-right menu associated with it. So the user first chooses the category to edit, like activity, and then moves the mouse onto it. This will cause the pull-right to appear and the user uses any of the three mouse buttons to select that item. The item in this case is an activity that the user would like to edit. Some pull-right menus are scrollable in order to accommodate the large number of items that must fit into a fixed size menu. Once the selection has been made, the item will appear in the main window of the activity editor. An example of this is in Figure 2.25 where the activity "Cabin Air Supply" has been selected for editing. The activity is shown first in order to show the layering in the system from activity to resource.

Now that the item of interest is displayed in the activity editor, it is time to begin updating it. All the data fields fall into basically three categories.

- The first category is a data field that can only be changed directly from keyboard inputs. See Figure 2.21. All three mouse buttons edit the field. The user selects the data field with any of the mouse buttons and a cursor will appear at the beginning of the data field. The user then uses standard emacs commands for cursor movements to move the cursor around in the data field.
- The second category is for the data fields that are not editable. See Figure 2.22. All mouse buttons are invalid. Since the user cannot update this field, any attempt to select the field will be ignored.
- The third category is the most complex. Each of the mouse buttons performs a different function. See Figure 2.23. The left mouse button allows the user to edit the field

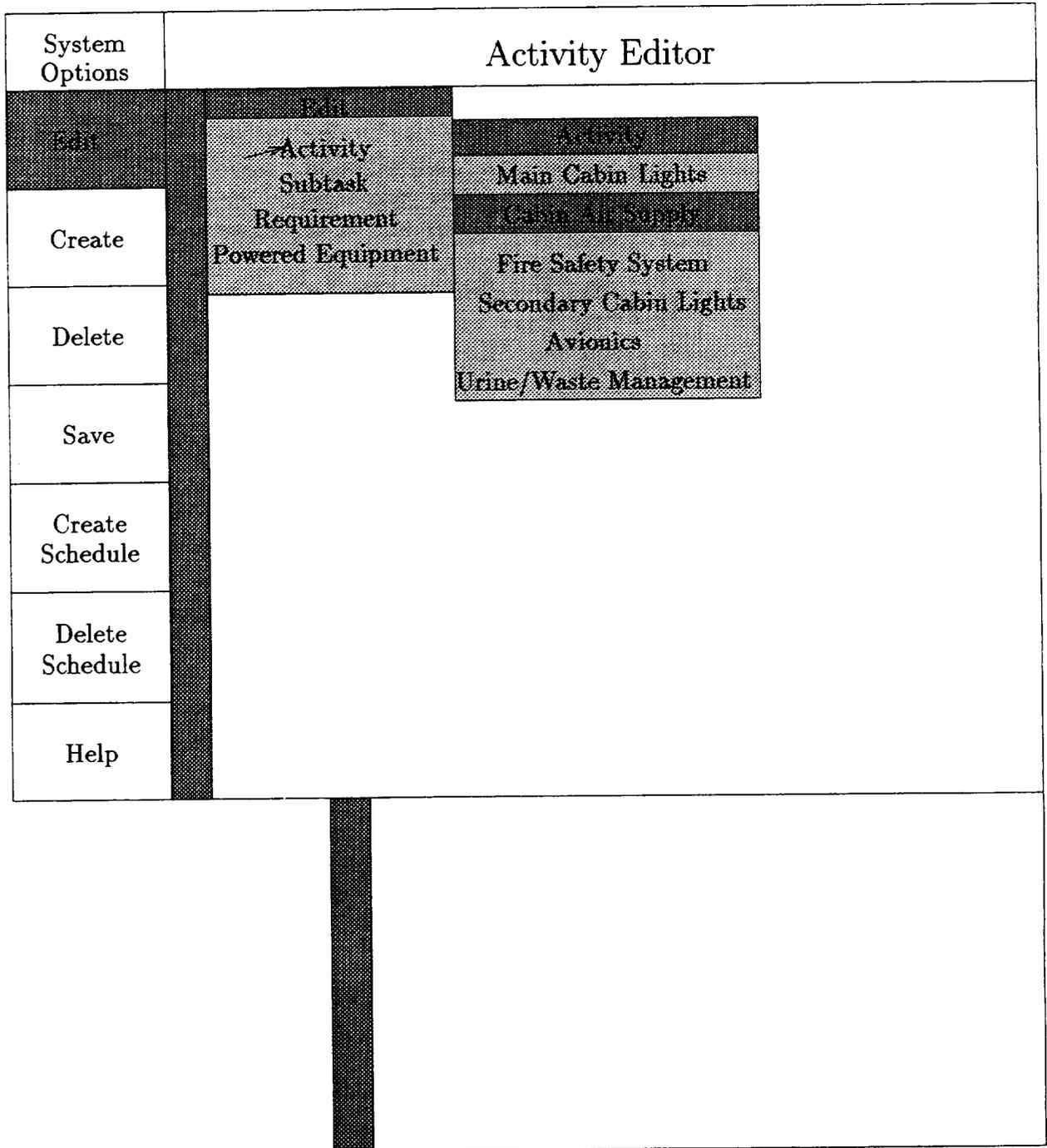


Figure 2.25: Activity Editor Activity Pull Right

directly. The right mouse button brings up a menu of options that the user can choose from to update this field. Later in this manual, we will cover the menu options in detail when we discuss each data field. The middle mouse button performs the expand/unexpand of the data field.

The expansion of a data field shows the rest of the information that makes up that data field. The middle mouse button provides both the function of expanding and unexpanding the data object based on the current state of the data object. For example, if the data object is already expanded, then the middle mouse button will cause an unexpand of that data object.

Once the data object is expanded, the user will see the data indented just below the data field that was expanded. Figure 2.20 shows a subtask that has been expanded within an activity. Indentations are used to show levels of expansion. Figure 2.26 shows a good example of nested expansion. First the subtask is expanded, then a piece of powered equipment within that subtask, and finally a mode is expanded within that piece of powered equipment. By following the indentation, the user can easily tell where an expansion starts and ends and to whom that expansion belongs. Also note how overwhelming the data would become if all the data associated with the activity were automatically displayed to the user. It would make it very difficult if not impossible to figure out what the activity was doing!

Layout of an Activity, Subtask, Requirement and Powered Equipment Up to now, editing data fields has been discussed in the broad sense. This section examines in detail the individual fields that make up the activities, subtasks, requirements and powered equipment. We will do this by stating the field name and the category that it belongs to and in case of category II, we will discuss what the menu options do.

Activity Fields

- **Description (Category I)** An alpha-numeric string that describes the task of the activity.
- **Name (Category I)** An alpha-numeric unique string that represents the name of the activity.
- **Static Priority (Category I)** A number between zero and three that represents the static priority of the activity.
- **Performances Requested (Category I)** A number representing how many times the user would like to have this activity scheduled.
- **Minimum Performances Needed (Category I)** The minimum number of performances that must be scheduled for this activity to be placed in the schedule. This number must be less than the number of performances requested.

System Options	Activity Editor	
Edit	Description: <input type="text" value="The operation of the pumps to supply cabin air."/>	
Create	Name: <input type="text" value="Cabin Air Supply"/>	
Delete	Base Priority: <input type="text" value="0"/>	
Save	Performances Requested: <input type="text" value="100"/>	
Create Schedule	Minimum Performances Needed: <input type="text" value="1"/>	
Delete Schedule	Subtask: Main Cabin Air Supply 1	
Help	Max Duration: <input type="text" value="129600"/>	
	Min Duration: <input type="text" value="1"/>	
	Maximum Delay: <input type="text" value="0"/>	
	Minimum Delay: <input type="text" value="0"/>	
	Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/>	
	Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input type="text" value="On"/> Location: <input type="text" value="LC-3-4"/>	
	Default Location <input type="text" value="LC-3-4"/>	
	Start: <input type="text" value="0"/> End: <input type="text" value="525600"/> Quantity: <input type="text" value="1"/>	
	Mode: On	
	Maximum Power: <input type="text" value="800"/>	
	Minimum Power: <input type="text" value="0"/>	
	Switch to Redundant: <input checked="" type="checkbox"/>	

Figure 2.26: Activity Editor Expansion of a Subtask

System Options	Activity Editor	
Edit	<div style="background-color: #cccccc; padding: 5px;"> Insert Existing Subtask Above Insert Existing Subtask Here Copy an Existing Subtask Above Copy an Existing Subtask Here Remove This Subtask Entry </div>	
Create		
Delete		

Figure 2.27: Activity Editor Subtask Options Menu

- **Subtask (Category III)** The individual sequential tasks that must be accomplished by the activity. The number of subtasks is unlimited and the ordering is important. The order that the subtasks appear in the activity is the order of the subtask execution in that activity.

This is a category II field so using the right mouse button on this data field produces a menu. See Figure 2.27. There are five items in this menu. The first four items have pull-right menus associated with them that contain all the subtasks that currently exist. See Figure 2.28.

- The first menu item "Inserting Existing Subtask Above" will insert the subtask selected above the current highlighted subtask (the initial subtask that was selected to bring up the menu).
- The second menu item "Insert Existing Subtask Here" will replace the current subtask that is highlighted with the one that is selected.
- The third menu item is very similar to the first menu item except that it will first make a copy of the subtask selected and then insert it above the highlighted subtask.
- Likewise the fourth menu item is similar to the second menu except that it will first make a copy of the subtask selected and then insert it into the highlighted subtask.

System Options	Activity Editor	
Edit	Subtask Options Insert Existing Subtask Above Insert Existing Subtask Here Copy an Existing Subtask Above Copy an Existing Subtask Here Remove This Subtask Entry	
Create		
Delete		Copy an Existing Subtask Here Operate Main Cabin Lights Main Cabin Air Supply 1 Main Cabin Air Supply 2 Operate Secondary Cabin Lights Operate Avionics Air Process Waste Heat Waste

Figure 2.28: Activity Editor Subtask Options Menu

- The fifth menu item “Remove This Subtask Entry” will remove the highlighted subtask from the activity.

The copy options in the third and fourth menu items are added for flexibility. This allows the user to select a subtask that needs slight modification, but the user does not want the modification to affect other activities that might use this subtask.

Subtask Fields

- **Maximum Duration (Category I)** The maximum number of minutes that the subtask is allowed to run.
- **Minimum Duration (Category I)** The minimum number of minutes that the subtask must run. This number must be less than or equal to the Maximum Duration number.
- **Maximum Delay (Category I)** The maximum delay in minutes between the start of this subtask and the ending of the preceding subtask in this activity. The field is not applicable if this is the first subtask of an activity.
- **Minimum Delay (Category I)** The minimum delay in minutes needed between this subtask and the preceding subtask in this activity. This number must be less than or equal to the Maximum Delay and is not applicable if this is the first subtask in the activity.
- **Requirement (Category III)** The individual non-powered equipment resources that the subtasks uses. The number of requirements is unlimited and the ordering is not important. This is a category III field so the right mouse button on this data field will cause a menu of options to be displayed. This menu has five items in the menu. They are exactly like the subtask where the user can insert and remove requirements from the subtask. The big difference is that there is another field associated with the requirement field that is called the amount field. The amount field is located just left of the requirement field on the same line. See the requirement field in Figure 2.26. This was done to let the user know that the two fields work as a group and should be treated as one. When the user inserts a new requirement into the subtask, the amount field will default to 1. It is the user’s responsibility to change the amount field to reflect how much of that resource will be used. The amount field is a category I field. Currently there are two types of requirements that are used: rate controlled (Crew) and equipment (Wrench).
- **Amount (Category I)** The number representing how much of the requirement will be used for this subtask. This number cannot be larger than the maximum amount that the requirement can have. This value is located in the availability list for the requirement.

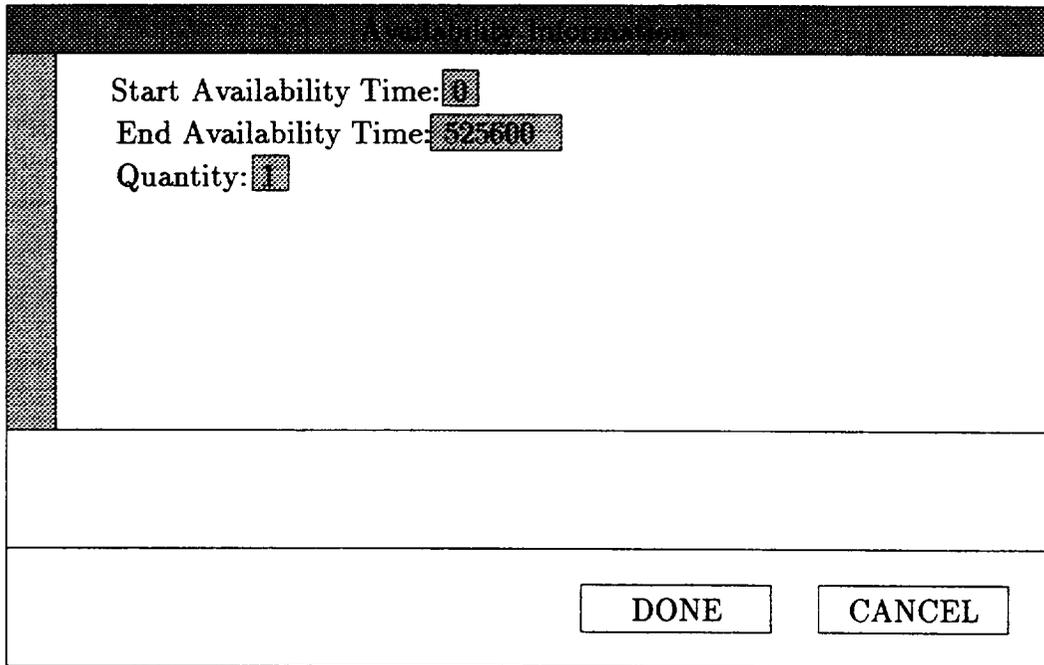
- **Powered Equipment (Category III)** The individual powered equipment that will be used by this subtask. The number of pieces of powered equipment for a subtask is unlimited. The order of the powered equipment is not important because by definition it represents what pieces of powered equipment the subtask uses over a given time interval. This is a category II data field that contains a menu option that acts just like the menu options in the subtask menus. See Figure 2.27 and Figure 2.28. The user uses the menu options to insert or remove pieces of powered equipment from the subtask.

There are two other pieces of data that are associated with every piece of powered equipment that are inserted in the mode and location fields. These two data fields are on the same line as the piece of powered equipment to show they belong together. Each is discussed below.

- **Mode (Category I*)** The mode in which the piece of powered equipment operates. The mode represents in which operational configuration the piece of powered equipment operates. When the user inserts a new piece of powered equipment into a subtask, the mode default is the first mode in the modes list for that piece of powered equipment (very non-deterministic). It is the user's responsibility to change this field to support the goal of the subtask. The I* means that the user does not edit it directly. When the user selects the data field with any mouse button, it will provide a menu of valid modes for that particular piece of powered equipment, from which the user selects one of the options.
- **Location (Category I*)** The location data field represents which switch the piece of powered equipment will be plugged into. When the powered equipment is inserted into the subtask, this data field defaults to the equipment's default location. This is the location that the piece of powered equipment will reside when the subtask is running. This is an I* category because the update is done through a menu that contains all the 1K switches in the system. The location of 3K switches is not currently implemented. When the user inserts a piece of powered equipment into the system, this data field is set to the default location specified in the powered equipment definition structure.

Requirement Fields

- **Name (Category II)** An alpha-numeric unique string that represents the name of this requirement. The name will appear at different indentation levels. In the case where the user has expanded the requirement from within a subtask, the name will be the requirement name seen in the subtask. It will not be prefaced by name. If the user edits the requirement directly, the name will be preceded by the word "NAME:" and the indentation level for all the data will be the same.



Start Availability Time: 0
End Availability Time: 525600
Quantity: 1

DONE CANCEL

Figure 2.29: Activity Editor Availability Edit Workbox

- **Resource Type (Category I*)** The type for this requirement. There are currently only two valid responses: rate controlled and equipment. MAESTRO will handle five and future versions of the activity editor will be upgraded to handle all five. The category type is a I* because the only way to change the data is through a menu that contains only valid resource types.
- **Start End Quantity (Category I)** The three data fields on one line represent an entry in the availability list for the requirement. The availability list is made up of a list of lists. Each element of the list contains a start time, end time and quantity. This is the way the user describes how much of the resource there is over time.

Figure 2.26 shows an example of a very basic availability list that just covers one interval with the quantity 1. The time span must cover from 0 minutes to 525,600 minutes. To change the availability list the user selects any of the data fields. A work box will appear in the scratch pad window. See Figure 2.29. This workbox will contain three data fields: start, end and quantity. There will be default values in each one of the data fields: start is 0, end is 525,600, and quantity is 1.

When the user edits the availability data fields, a specific interval is created to insert in the availability list. Suppose the user enters a start time of 10 minutes, an end time of 100 minutes, and a quantity of 2. The new availability list would look like Figure 2.30. Notice in Figure 2.30 that the new availability list now has three entries. The first goes from 0 to 10 minutes at a quantity of 1, the next entry goes from 10 minutes to 100 minutes at a quantity of 2 (the one just entered), and the third entry goes from 100 minutes to 525,600 minutes at a quantity of 1. When the user adds a new interval, the system knows to fill out the availability list to cover the time span from 0 to 525,600 minutes. Note that if the user had entered the quantity value of 1 instead of 2 then there would have been no change in the availability list because all the quantities would have the same value. By changing the interval, the user can create any required complexity for the availability list.

To remove an interval, extend the interval preceding it, and extend it to cover the interval you want to remove. This may seem confusing, but just try it and it will become second nature in just a few clicks of the mouse.

Powered Equipment Fields

- **Default Location (Category I*)** This is the default switch that the piece of powered equipment will default to when selected for use in a subtask. The selection is made through a menu that contains a list of available 1K switches from which the user can choose. When the user creates a new piece of powered equipment, they specify this location.
- **Start End Quantity (Category I)** This is exactly the same as the availability discussed in Section 2.6.7.
- **Mode (Category III*)** These represent all the currently defined modes for this piece of powered equipment. This is a category III* because the left and right mouse buttons do different actions. The left mouse button has no action at all because there is no list of other modes that can be placed into this list of modes. All valid modes are in the list.

The right mouse button is also different. It brings up a menu of options, but this menu only has two options while the others had five. The two options are creating a new mode for this piece of powered equipment and deleting a mode from this piece of powered equipment. The others allowed the user to insert existing or copied data into the particular data type, but because the mode is unique to powered equipment, this paradigm had to be changed to handle this. In creating a new mode, the name must be unique within one piece of powered equipment. That is, two pieces of powered equipment could each have a mode named "ON" without causing a problem.

System Options	Activity Editor	
Edit		Description: <input type="text" value="The operation of the pumps to supply cabin air"/> Name: <input type="text" value="Cabin Air Supply"/> Base Priority: <input type="text" value="0"/>
Create		Performances Requested: <input type="text" value="180"/> Minimum Performances Needed: <input type="text" value="1"/>
Delete		Subtask: Main Cabin Air Supply 1 Max Duration: <input type="text" value="129600"/> Min Duration: <input type="text" value=""/>
Save		Maximum Delay: <input type="text" value="0"/> Minimum Delay: <input type="text" value="0"/>
Create Schedule		Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/> Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input type="text" value="On"/> Location: <input type="text" value="LC 3 4"/>
Delete Schedule		Default Location: <input type="text" value="LC 3"/> Start: <input type="text" value="0"/> End: <input type="text" value="10"/> Quantity: <input type="text" value=""/>
Help		Start: <input type="text" value="10"/> End: <input type="text" value="100"/> Quantity: <input type="text" value="2"/> Start: <input type="text" value="100"/> End: <input type="text" value="32560"/> Quantity: <input type="text" value="1"/> Mode: On

Figure 2.30: Activity Editor Updated Availability List

Mode Fields Modes are only editable through a piece of powered equipment. This allows the user to re-use the same mode name (i.e. ON) in more than one piece of powered equipment. It is not possible to edit a mode from the application menu, because the user needs the context within which the mode is being edited.

- **Maximum Power (Category I)** The maximum power in watts that the piece of powered equipment may use in this mode. This number may not exceed 1000 or be less than 0. This number must also be greater than the minimum power. Equal is not allowed.
- **Minimum Power (Category I)** The minimum power in watts that a piece of powered equipment uses. This number must be between 0 and 1000 and must be less than the maximum power.
- **Switch to Redundant (Category I)** A single alpha-character, "Y" or "N", that specifies whether the mode for this piece of powered equipment has a switch to redundant capability. "Y" means it has redundant capability for this mode, and "N" means there is no redundant capability for this mode.
- **Permission to Test (Category I)** A single alpha-numeric character, "Y" or "N", that specifies whether the mode for this piece of powered equipment is testable during a fault. A "Y" means the piece of equipment is testable, and "N" means that the piece of equipment is not testable in this mode.

Scheduling The activity editor's job is to create and update activities and other building blocks used to create activities. These activities are then used to run schedules on the SSM/PMAD system.

The user can create schedules from the activity editor by using the "Create Schedule" option on the activity editor application menu. Upon selection of this item the system will pop up a scrollable menu window. See Figure 2.31. This menu is different from any we have seen so far in the activity editor. By selecting any item that is not enclosed by << >>, the item will become selected. The item is highlighted after it is selected. The user will then go through and select the activities to be scheduled. If the user erroneously selects an activity, selecting it again will cause it to be de-selected and the menu item will be de-highlighted. This menu is also scrollable because the number of activities may be very large. Once the user is done selecting activities, choosing the menu item titled <<Schedule Selected Activities>> will start the scheduling processes.

The first step of the scheduling process is to determine if the database has been modified or not. If any of the data in the activity database has been changed, then the database must be saved prior to scheduling. Also if any of the data has been changed, the scheduling process determines if any of these changes will affect already existing schedules and informs the user of the conflicts. Any change to the data that affects a schedule must be dealt with

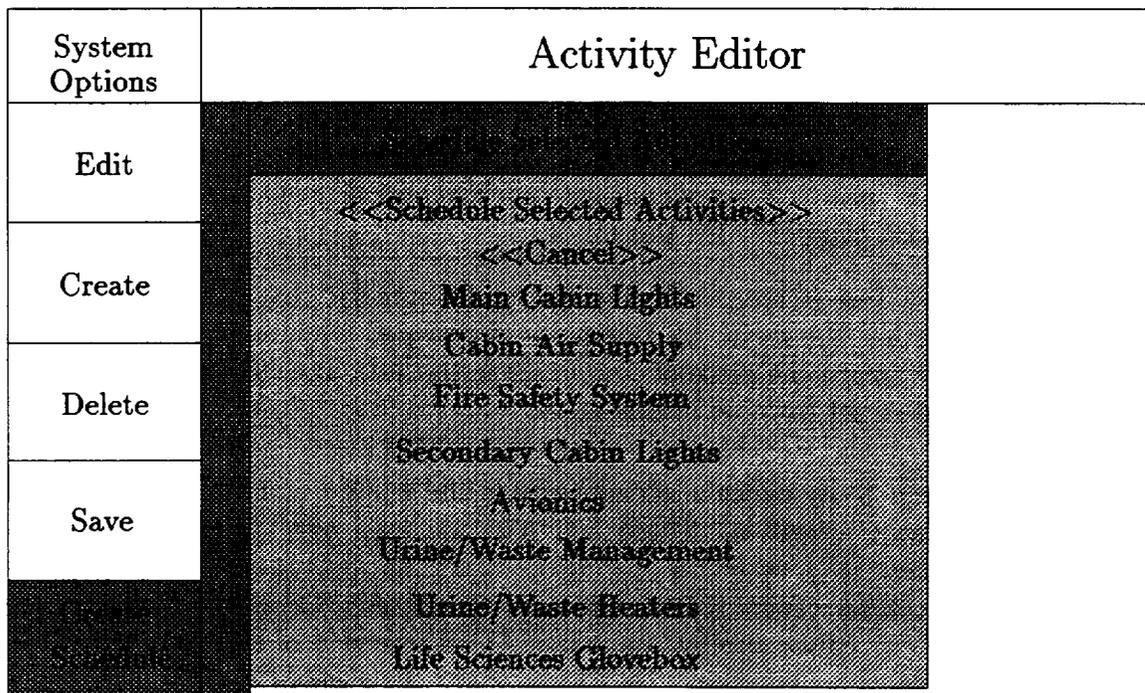


Figure 2.31: Activity Editor Scheduling Menu

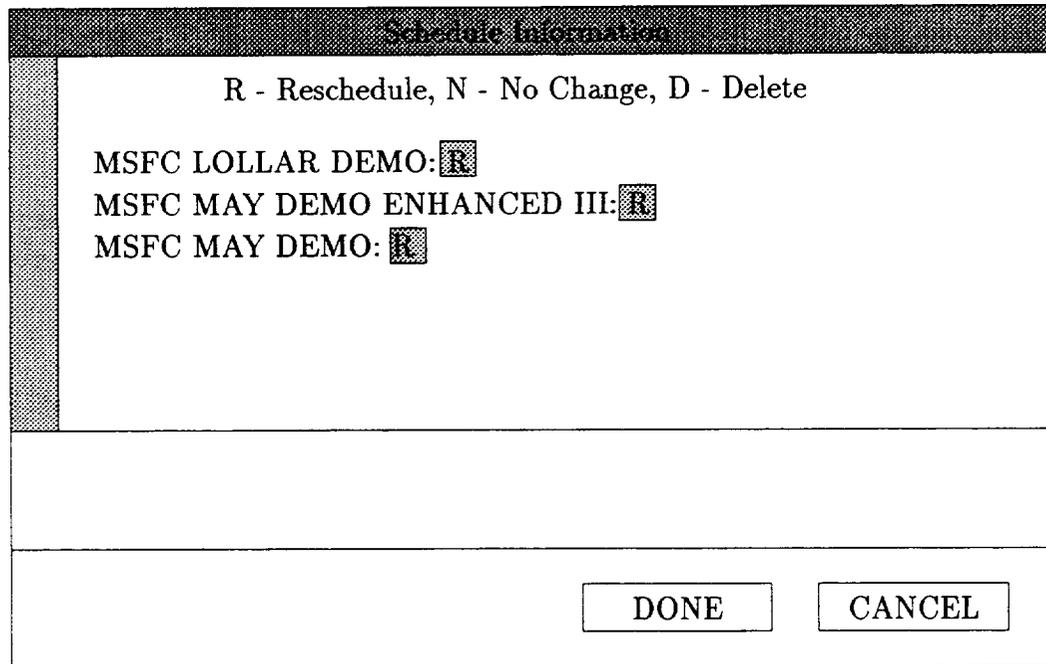


Figure 2.32: Activity Editor Scheduling Changes

immediately, so the system will put up a workbox of all the effected schedules and let the user decide what to do with them. Figure 2.32 shows an example of the workbox. In this case the changes to the activity database have invalidated three schedules.

If a schedule is invalidated due to activity changes to the activity database, the user can do one of three things to the schedule:

- Reschedule the activities that make up the schedule
- Delete the schedule
- Ignore the problem

The third option is very dangerous because if the user invalidated the schedule and decides not to rectify the problem, there is no guarantee that the schedule will run correctly. This option is included for flexibility, but the user should exercise caution when using it.

After the user has saved the database, the system will pop up another workbox to obtain non-activity pertinent information. At the present time there are only three pieces of non-activity information needed for scheduling:

- Schedule name
- Start time for the schedule in minutes
- End time for the schedule in minutes

For now, leave the start time at zero. The effect of changing it to a non-zero value is still being investigated. The end time must be greater than the start time and less than 525,600 minutes. Once the user has finished entering this data, the data will be sent to MAESTRO for scheduling.

When scheduling is completed, look at the scratch pad window to see if any problems may have occurred. Scheduling information concerning activities which were not scheduled for some reason will appear in the window, and can be used to correct the activity.

Deleting the Data To use the deleting capabilities of the activity editor, the user selects the Delete option on the applications menu. This causes a menu to appear in the middle portion of the left hand side of the application main window. Notice that there are four entries in the menu:

- Activity
- Subtask
- Requirements
- Powered Equipment

From an earlier discussion, the user will remember that requirements and powered equipment are resource building blocks used in the creation of subtasks, which are used to build activities, which are used to build schedules.

The four choices for deletion (activity, subtask, requirements, and powered equipment) are discussed in the following four sections. Each of the four choices in the delete menu has a pull-right menu associated with it. These pull-rights contain all the items that have been created for that category. Some pull-right menus are scrollable in order to accommodate the large number of items that must fit into a fixed size menu.

Delete Activity One of the four choices for deletion is an activity. The user can delete an activity by selecting the Delete Option of the Application menu. Selecting Activity will produce a pull-right with all the available activities for deletion. The user uses any of the three mouse buttons to select an activity for deletion. After selecting an activity to delete, one of two workboxes will appear in the scratchpad area of the screen. If the activity is not a part of any schedule, a workbox entitled "Confirmation!" will appear. If any schedules are

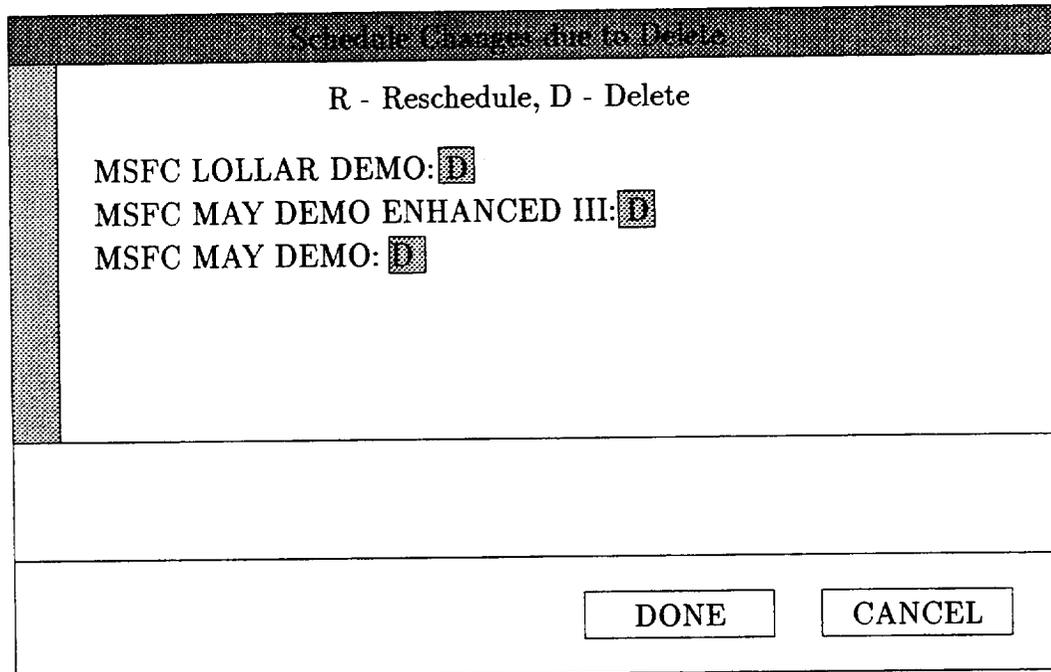


Figure 2.33: Schedule Changes due to Delete

affected by the deletion of the selected activity, a workbox entitled "Schedule Changes due to Delete" will appear.

If the workbox entitled "Schedule Changes due to Delete" appears, the changes to the schedule must be dealt with immediately, and the user decides if the schedule should be rescheduled or deleted. The user types R for reschedule or D for delete for each of the schedules affected. Clicking "Done" will initiate the changes or clicking "Cancel" will cancel the request. Figure 2.33 shows an example of the workbox where schedules are affected. In this case the deletion of activity Main Cabin Lights has invalidated the listed schedules.

After clicking "Done", the request is processed, and a report will be printed to the scratch pad area displaying the success rate of rescheduling the activities in the schedule. The activity will not appear in the revised schedule. If the schedule is requested for deletion, the schedule will be deleted from the list of all schedules in the database.

If the activity to be deleted is not a part of any schedule, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox cancels the request and erases the workbox from the scratchpad area. The confirmation

Activity Editor	
Create	
Delete	Activity
Save	Requirement Powered Equipment
Create Schedule	Operate Main Cabin Lights Main Cabin Air Supply 1 Main Cabin Air Supply 2 Operate Avionics Air Process Waste Heat Waste
Delete Schedule	

Figure 2.34: Delete Subtask Pull Right Menu

workbox is used to confirm the user's intention before really deleting an activity.

Several slots of the Frames database are affected by deleting an activity. As the result of the deletion of an activity, the activity is deleted from each of the subtasks that comprise the activity. This relationship between activities and subtasks is recorded in the subtask frame in a slot called activities. The activity and its execution sequence number are also deleted from the subtask frame slot called sequence. The activity group is deleted from the list of all activity groups in the system. The activity is deleted from the list of activities on the edit and delete pull-right menus. The activity is deleted from the list of activities on the create schedule menu.

Delete Subtask Another choice for deletion is a subtask. The user can delete a subtask by selecting the Delete Option of the Application menu. Selecting Subtask will produce a pull-right with all the available subtasks for deletion. See Figure 2.34.

The user uses any of the three mouse buttons to select a subtask for deletion. After selecting a subtask to delete, a workbox or a message will appear in the scratchpad area of the screen. If the subtask is a part of an activity, a scratchpad message entitled "Subtask

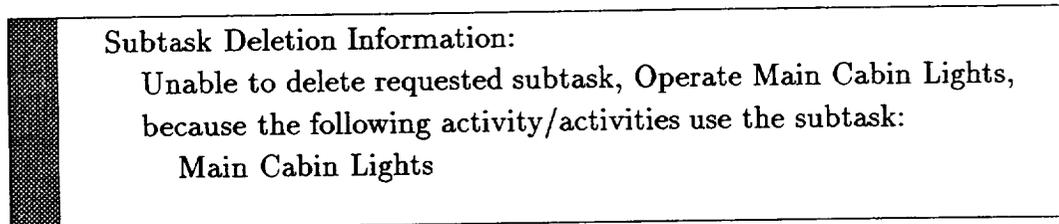


Figure 2.35: Subtask Deletion Information

Deletion Information” will appear. See Figure 2.35. If the subtask is not a part of any activity, the confirmation workbox appears.

The message entitled “Subtask Deletion Information” lists all the activities that contain the subtask to be deleted. The message explains that the subtask can not be deleted because there are activities that use the subtask. In general, the user must first delete the activities that use the subtask, and then delete the subtask. At this point in time, the system will not do that automatically for the user, because the connection between subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the subtask to be deleted is not a part of any activity, the confirmation workbox will be displayed. The only action required from the user is to click “Done” to validate the deletion, or click “Cancel” to cancel the request. Clicking anywhere outside the workbox cancels the request and erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user’s intention before really deleting an subtask.

As the result of the deletion of a subtask, the subtask is deleted from a list of all subtasks in the system, from each of the pieces of powered equipment that use the subtask, from the list of subtasks on the edit and delete pull-right menus, and from the list of subtasks displayed when “Insert an Existing Subtask Above” is selected while editing subtasks.

Delete Requirement Another choice for deletion is a requirement. The user can delete a requirement by selecting the Delete Option of the Application menu. Selecting Requirement will produce a pull-right with all the available requirements for deletion. The user uses any of the three mouse buttons to select a requirement for deletion. After selecting a requirement to delete, a workbox or a message will appear in the scratchpad area of the screen. If the requirement is a part of a subtask, a scratchpad message entitled “Requirement Deletion Information” will appear. If the requirement is not a part of any subtask, the confirmation

workbox appears.

The message entitled "Requirement Deletion Information" lists all the subtasks that contain the requirement to be deleted. The message explains that the requirement can not be deleted because there are subtasks that use the requirement. In general, the user must first delete the subtasks that use the requirement, and then delete the requirement. At this point in time, the system will not do that automatically for the user, because the connection between requirements, subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the requirement to be deleted is not a part of any subtask, the confirmation workbook will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbook cancels the request and erases the workbook from the scratchpad area. The confirmation workbook is used to confirm the user's intention before really deleting a requirement.

As the result of the deletion of a requirement, the requirement is deleted from a list of all requirements in the system. The list of all requirements is a part of the the list of all resources in the system. The requirement is deleted from the list of requirements on the edit and delete pull-right menus. The requirement is deleted from the list of requirements displayed when "Insert an Existing Requirement Here" is selected while editing requirements.

Delete Powered Equipment Another choice for deletion is a piece of powered equipment. The user can delete a piece of powered equipment by selecting the Delete Option of the Application menu. Selecting Powered Equipment will produce a pull-right with all the available pieces of powered equipment for deletion. The user uses any of the three mouse buttons to select a piece of powered equipment for deletion. After selecting a piece of powered equipment to delete, a workbook or a message will appear in the scratchpad area of the screen. If the piece of powered equipment is a part of a subtask, a scratchpad message entitled "Powered Equipment Deletion Information" will appear. If the piece of powered equipment is not a part of any subtask, the confirmation workbook appears.

The message entitled "Powered Equipment Deletion Information" lists all the subtasks that contain the piece of powered equipment to be deleted. The message explains that the piece of powered equipment can not be deleted because there are subtasks that use the piece of powered equipment. In general, the user must first delete the subtasks that use the piece of powered equipment, and then delete the piece of powered equipment. At this point in time, the system will not do that automatically for the user, because the connection between pieces of powered equipment, subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the piece of powered equipment to be deleted is not a part of any subtask, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox also erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user's intention before really deleting a piece of powered equipment.

As the result of the deletion of a piece of powered equipment, the piece of powered equipment is deleted from a list of all resources in the system. The piece of powered equipment is deleted from the list of pieces of powered equipment on the edit and delete pull-right menus. The piece of powered equipment is deleted from the list of requirements displayed when "Insert an Existing Powered Equipment Here" is selected while editing pieces of powered equipment.

Intermediate Levels Of Autonomy - User Interface

As previously described, under ILA the SSM/PMAD system supports both scheduled operations and manual intervention into the normal (the NORMAL operational mode means AUTONOMOUS) mode of operation. The user interface supports this capability by giving the user easy point and click access to the switches. Should the user choose to seize control of a switch, a workbox is provided to the user with default information for the new manual activity which may be changed as desired. This workbox will return a menu of possible activity performance times which may be selected for initiation of the new ILA activity compatible within the schedule. The user may be informed that manual operations initiation is not allowed if the system is processing a contingency. Also, if the activity presently on the switch has higher priority, the user will be informed by KANT so that another switch may be selected or the user's priority may be elevated. Manual operations are also discouraged on a switch which is being used as a redundant resource to a redundantly powered load. This method of operation provides the user with flexibility of adding a manually controlled activity to the system on the fly, with autonomously scheduled operations on the same switch being planned for and rescheduled around the manual intervention. The resultant presentation of ILA manual intervention for controlling a switch resource is shown in Figure 2.36. The human hand symbol pointing to the switch is added, displaying the fact that the switch is now manually controlled. This is a simple and very understandable addition.

2.6.8 The Transaction Software

Basic Overview

The old transaction software worked, but because it was not well defined, it was cumbersome to add new transaction types. The old software required that the user update five files with some files requiring multiple additions. The five files requiring updates are transaction.cl, ssmpmad-domain.cl, processes.cl, globals.cl and one of the semantic processing files (e.g.

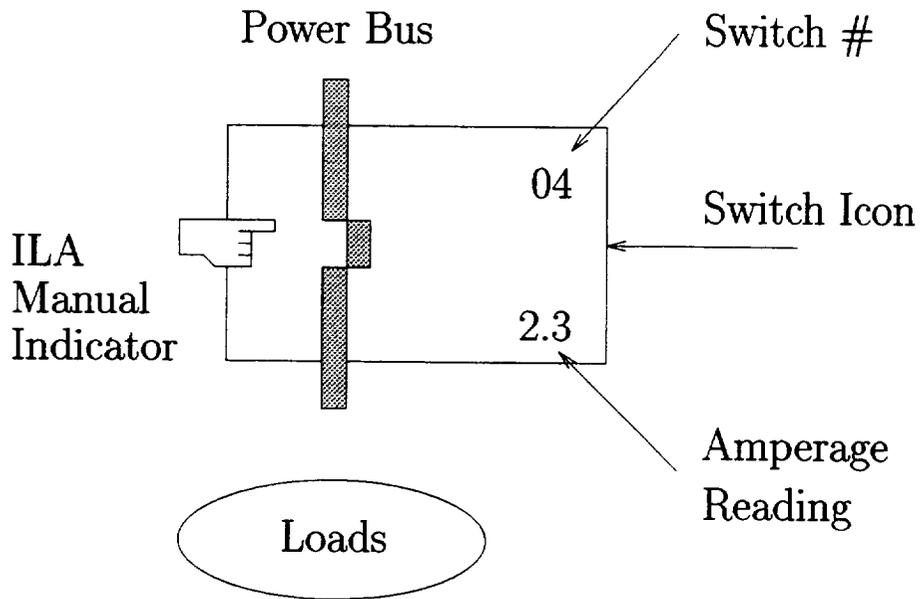


Figure 2.36: SSM/PMAD User Environment ILA Representation

ssmpmad-if-tr.cl, feles-tr.cl, or lplms-tr.cl). In the transaction file the user performs multiple updates:

1. Define blocking and deblocking information.
2. Create TRANSFORM IN and TRANSFORM OUT functions even if they perform no useful service.
3. Make a structure definition to create accessor functions and/or the ability to create instances of the transaction type.
4. Add the transaction id into the transaction table.

In the ssmpmad-domain.cl file the user had to create the frame for the transaction type. The processes.cl file was then updated to assert the semantic process function name into the frame for the transaction type. Next the user updated the globals file to assure name consistency across all packages. The last file modified was one of the semantic process files to add a new process function ¹

The new design reduces the number of files that may need to be updated to three and also simplifies the procedure for adding a new transaction type. It confines the basic transaction information to the individual transaction frame while providing the same overall functionality the user had before. The new design also adds a new feature to add transactions on the fly. The basic file the user modifies is comm.domain file. This file contains all the transaction specifications and templates for creating transactions.

Specifying a Transaction

A transaction is a specification, or an agreement, that describes how data will be transmitted and received between two parties. A transaction in the SSM/PMAD system contains a number of items, possibly assembled in list format (e.g. (1 "1234567890" 3 000.2 five1 five2)), that gets assembled (blocked) into a character string for transmission over a communication channel. On the receiving end the character string is disassembled (deblocked) into the corresponding data fields that make up the transaction.

The specification of a transaction is a template which is simply a list of descriptors, one for each item to be blocked. Transactions may be of arbitrary complexity with any item being described with a nested template. The template descriptor is a list of either three or four fields, optionally followed by an accessor field.

¹The process function is the main routine to process the input transaction information into the system. The transform in and out functions perform the low level data transformation (e.g. Big Indian, Little Indian).

Three Field Descriptors The definition of the three field descriptors is very straightforward. The first field is a string describing the descriptor. This field can be as long as the user wants and its purpose is to provide a clear description of the field type. The second field is an integer representing the size of the element the descriptor will block/deblock. The third field defines the type of element to be blocked/deblocked. An example of a template using the three field descriptor is as follows:

```
("Example of PACKED" 1 packed)
("Example of STRING" 10 string)
("Example of NUMERIC" 5 numeric)
("Example of FLOAT" 5 float)
("Example of SYMBOL" 5 symbol)
("Example of SYMBOL+" 5 (symbol "comm"))
```

This example show that the first element in the transaction list would be an integer of base 79 and a length of one. The second element in the list would be a string that can contain up to 10 characters for blocking and exactly 10 characters for deblocking². The third element would be a numeric whose range would be from 0 to 99999. The fourth element would be a floating point number that contains four numeric numbers and one decimal point. The decimal point can be anywhere. The fifth element is a Lisp symbol that can be up to five characters in length. The sixth and final element is the same as the fifth except for the string "comm". This sting would be used by the deblocking code to intern the symbol into the :comm package. This allows the user to specify in which package the symbols are to be defined.

The third field defines the type of the descriptor element. In the previous example the types were all "simple" types. The simple types are internally defined and are one of the following:

1. PACKED
2. STRING
3. NUMERIC
4. FLOAT
5. SYMBOL

Besides the simple types the user can create complex data types. A complex data type is one that is itself another template. This allows the user to organize data into logical groups

²In the case of blocking the blocking code will pad the string with blanks to fill out the entire string length.

and then use that group repeatedly within many transactions. Suppose the user wants to send out the data ((A05 5) (A06 10) (B06 15) (c05 12)) which represents switches and times. Without complex data types the user would have to block/deblock each pair separately. The complex data type allows the user to block all the data in a single step. This example would now simplify to:

```
((("Switches and Time" 10 (("Switch Id" 3 symbol)
                               ("Time Stamp" 6 numerical))))
```

When processing a descriptor with a complex data type, the second field represents the number of elements of the nested type to be block/deblocked. The depth of the nesting of complex data type is theoretically infinite (not practical) with all the base types being simple types.

Four Field Descriptors The four field descriptor allows the user to have variable length data. In the three field case, if the second field was five then the number of bytes could not exceed five for a simple type or in the case of complex types could not exceed five elements of the complex type. So in the example with the Switch and Time values, if the user does not have 4 switch-time pairs the blocker/deblocker code will give unpredictable results. The four field descriptor dynamically determines the number of the data elements being sent out and prepends that number to the outgoing transaction. In the case of incoming data that number is used to determine how many elements of that data type to deblock.

The first field is exactly the same as the three field descriptor case. The third field is of type numeric or packed and will be used to represent the number of elements of the type to be blocked/deblocked. The second field is the number of bytes that will be used to represent the number described by the third field. This means that the user must have some idea how big it can be in order to set the right order of magnitude for field two. For example if the user knew there could be 1000, then field two would be 4 if field three were numeric or 2 if field three were packed. The fourth field is the data type and can be either of type simple or complex as before. An example descriptor using the four field case could look like:

```
("Variable Length Numeric" 1 packed numeric)
```

This four field specification would block/deblock any length numeric up to 79 characters in length³. The first field in a transaction represented by (123456789 . . .) would be blocked into ":1234567890". Notice the ":", it is the one byte representation of the number 10 in packed format which is the length of the numeric element. The blocker determined the length of the numeric and attached it to the front end of the numeric string. This allows the same deblocking code to read that number and deblock the next ten characters into a numeric.

³Packed is base 79 with each digit offset from "48" in the ASCII table

In general, the packed format will take up less space than its numeric counterpart but will cost more in processing time. The user is advised to use numeric if the number is between 0-9 and 80-99 and packed if any other number.

Accessor Field The optional accessor field allows the user to define an accessor function to access particular elements in the transaction list. This is a useful feature because the user does not have to remember which element in the list represents specific data. This provides data abstraction by allowing the user to change the details of the data without changing the code. To fully appreciate this let's look at an example:

```
(("Date Time Stamp" 1 (("Month" 2 numeric)
                        ("Day" 2 numeric)
                        ("Year" 2 numeric)
                        ("Hour" 2 numeric)
                        ("Minute" 2 numeric)
                        ("Second" 2 numeric)))
 ("Dog Description" 1 packed (("Dogs Name" 1 packed string)
                              ("Color" 1 packed symbol)
                              ("Age" 1 packed numeric)
                              ("Sex" 1 symbol))))
```

To access the age of the dog the user would have to do something like:

```
(third (second <transaction-list>))
```

This does not lend itself to being very readable - one does not know what the data represents. The user can define accessors as follows:

```
(("Date Time Stamp" 1 (("Month" 2 numeric (:accessor dts-month))
                        ("Day" 2 numeric (:accessor dts-day))
                        ("Year" 2 numeric (:accessor dts-year))
                        ("Hour" 2 numeric (:accessor dts-hour))
                        ("Minute" 2 numeric (:accessor dts-minute))
                        ("Second" 2 numeric (:accessor dts-second)))
 (:accessor date-time-stamp))
 ("Dog Description" 1 packed
  (("Dogs Name" 1 packed string (:accessor dog-name))
   ("Color" 1 packed symbol (:accessor dog-color))
   ("Age" 1 packed numeric (:accessor dog-age))
   ("Sex" 1 symbol (:accessor dog-gender)))
 (:accessor dog-descript)))
```

To access the dog's age would now look like this:

```
(dog-age (dog-descript <transaction-list>))
```

This is much more informative in two respects. First the user no longer has to remember where in the transaction list a particular piece of data is and seconding the person reading the code will be able to see what data is being accessed. To modify the value, use the "SETF" macro like you would for changing a slot on a structure.

The MAKE-TEMPLATE Function The preceding sections explained how the templates are built and how they are used. It was shown that templates could be nested to create more complex transactions. The ability to nest transactions may also cause some templates to be too complex for the average user to understand the transaction. To show this, consider the following example taken from the current SSM/PMAD transaction:

```
(("Bus A Begin Time" 6 numeric (:accessor bus-a-start))
 ("Bus A End Time" 6 numeric (:accessor bus-a-end))
 ("Number A Entries" 1 packed (:accessor number-a-entries))
 ("Bus A Util" 1 packed (("Component" 3 symbol (:accessor spec-id))
                        ("Power" 5 numeric (:accessor spec-power))
                        ("Time Stamp"
                         1
                         (("Day" 2 numeric (:accessor dts-day))
                          ("Month" 2 numeric (:accessor dts-month))
                          ("Year" 2 numeric (:accessor dts-year))
                          ("Hour" 2 numeric (:accessor dts-hour))
                          ("Minute" 2 numeric (:accessor dts-minute))
                          ("Second" 2 numeric (:accessor dts-second))))
                         (:accessor spec-time-stamp)))
                        (:accessor bus-a-utilization))
 ("Bus B Begin Time" 6 numeric (:accessor bus-b-start))
 ("Bus B End Time" 6 numeric (:accessor bus-b-end))
 ("Number B entries" 1 packed (:accessor number-b-entries))
 ("Bus B Util" 1 packed (("Component" 3 symbol (:accessor spec-id))
                        ("Power" 5 numeric (:accessor spec-power))
                        ("Time Stamp"
                         1
                         (("Day" 2 numeric (:accessor dts-day))
                          ("Month" 2 numeric (:accessor dts-month))
                          ("Year" 2 numeric (:accessor dts-year))
```

```

        ("Hour" 2 numeric (:accessor dts-hour))
        ("Minute" 2 numeric (:accessor dts-minute))
        ("Second" 2 numeric (:accessor dts-second)))
        (:accessor spec-time-stamp)))
    (:accessor bus-b-utilization)))

```

To help alleviate the complexity problem a function has been developed called "MAKE-TEMPLATE". This function allows the user to put whole templates as well as subsections of templates into a common database that is accessible through a unique symbol name to the other templates or in a transaction definition. Rewriting the above examples using the "MAKE-TEMPLATE" function would transform into:

```

(make-template 'time-stamp
  (('("Day" 2 numeric (:accessor dts-day))
    ("Month" 2 numeric (:accessor dts-month))
    ("Year" 2 numeric (:accessor dts-year))
    ("Hour" 2 numeric (:accessor dts-hour))
    ("Minute" 2 numeric (:accessor dts-minute))
    ("Second" 2 numeric (:accessor dts-second))))

(make-template 'utilization
  (('("Component" 3 symbol (:accessor spec-id))
    ("Power" 5 numeric (:accessor spec-power))
    ("Time Stamp" 1 packed (:template time-stamp)
      (:accessor spec-util))))

(("Bus A Begin Time" 6 numeric (:accessor bus-a-start))
 ("Bus A End Time" 6 numeric (:accessor bus-a-end))
 ("Number A Entries" 1 packed (:accessor number-a-entries))
 ("Bus A Util" 1 packed (:template utilization)
  (:accessor bus-a-utilization))
 ("Bus B Begin Time" 6 numeric (:accessor bus-b-start))
 ("Bus B End Time" 6 numeric (:accessor bus-b-end))
 ("Number B Entries" 1 packed (:accessor number-b-entries))
 ("Bus B Util" 1 packed (:template utilization)
  (:accessor bus-b-utilization)))

```

The above modularization makes it much easier for the average user to understand what is in the transaction. This also allows the user to use the time-stamp and utilization as separate parts in other transactions as well. One other feature of "MAKE-TEMPLATE" that should be noted is that it creates a function for the user to create instances of the template just

like with structures. This feature along with the accessor function "MAKE-TEMPLATE" provides functionality very similar to "DEFSTRUCT".

Defining a Transaction

The basic element of a transaction is the template but in order to define a transaction for the system to use, more information is required. The first element is the template and it has already been discussed. The second element is the transaction id. The id is a unique number for each transaction between 0 and 6241 and it is required. This ID must be unique so having two transactions with the same ID will cause one of them to be lost in the computer ozone. The next element is the priority and it can be either high, normal, or low. This element will default to normal. The fourth element is the TRANSFORM_IN function name. This is the name of a user function that performs low level data manipulation or transformation on the transaction data after it has been read in. This is useful if the data coming in is in a format of feet and the system is using meters. This function could also be used to transform data from EBCDIC to ASCII but this is not recommended. The latest POSIX work has developed the TLI layer which stands for the Transport Layer Interface which handles transferring data from different platforms and makes sure that a floating point number on one machine is the same floating point number on another. This field is optional and the default is nil. The fifth element is the name of the "TRANSFORM.OUT" function, and it serves the same purpose as the "TRANSFORM-IN" but for outgoing transactions. the sixth element holds a list of "SEMANTIC-PROCESS-FUNCTIONS". These functions are the main routine to process incoming transactions data into the system. This field is optional with the default set to nil. If this is set to nil then the data will be read in and then dumped into the bit bucket so it is **VERY STRONGLY RECOMMENDED** to always have "SEMANTIC-PROCESS-FUNCTIONS" for incoming transactions. Now the functions in the list are called in order in parallel. An example of making a transaction would look like:

```
(make-template 'time-stamp
  (('("Day" 2 numeric (:accessor dts-day))
    ("Month" 2 numeric (:accessor dts-month))
    ("Year" 2 numeric (:accessor dts-year))
    ("Hour" 2 numeric (:accessor dts-hour))
    ("Minute" 2 numeric (:accessor dts-minute))
    ("Second" 2 numeric (:accessor dts-second))))
```

```
(make-template 'utilization
  (("Component" 3 symbol (:accessor spec-id))
   ("Power" 5 numeric (:accessor spec-power))
   ("Time Stamp" 1 packed (:template time-stamp))
```

```

(:accessor spec-util))))

(make-transaction 'utilization-data
  :template
  '(("Bus A Begin Time" 6 numeric
    (:accessor bus-a-start))
    ("Bus A End Time" 6 numeric
    (:accessor bus-a-end))
    ("Number A Entries" 1 packed
    (:accessor number-a-entries))
    ("Bus A Util" 1 packed
    (:template utilization)
    (:accessor bus-a-utilization))
    ("Bus B Begin Time" 6 numeric
    (:accessor bus-b-start))
    ("Bus B End Time" 6 numeric
    (:accessor bus-b-end))
    ("Number B Entries" 1 packed
    (:accessor number-b-entries))
    ("Bus B Util" 1 packed (:template utilization)
    (:accessor bus-b-utilization)))
  :id 10
  :priority 'normal
  :transform-in-function 'util-transform-in
  :transform-out-function nil
  :semantic-process-function '(process-utilization-data))

```

Opening and Closing Connections, Sending and Receiving Data

The transaction defines what to do to the data prior to sending the data out or after receiving the data in. The next step is to discuss the mechanism for sending the data out and receiving the data. Unfortunately this code is specific to the task of the SSM/PMAD system. All the basic building blocks are present to create these very generic constructs that would be very simple for the user to use, but this was not a critical issue. The current SSM/PMAD system would more likely just change the type of transactions being used then to open and close connections to other hardware. For this reason the general purpose functions for opening and closing connections as well as sending and receiving data were not developed. It is hoped that in the future that this capability is added. In order to make the communications a plug in application to KNOMAD this would need to be finished. For now the user must be content with a very generic tool for creating transactions.

2.6.9 Lowest Level Functions

The Lowest Level Functions are a series of algorithms designed to run continuously on the 80386 based Lowest Level Processors. These algorithms maintain contact with the testbed power system hardware, command the power system hardware, and report faults detected by the power system hardware. The Lowest Level Functions also keep track of certain power system performance characteristics. In addition, the LLF's accept event and priority lists from FELES and LPLMS respectively. They also carry out switching operations commanded by FRAMES for fault isolation.

2.6.10 Intermediate Levels of Autonomy

Description

The system developed on this contract represents state-of-the-art in intelligently controlled autonomous power management and distribution systems. Autonomous systems, however, present unique problems whenever manual human intervention becomes desirable; and manual human intervention is not only desirable but is mandatory in environments with direct human contact, such as is the case with Space Station Freedom. The SSM/PMAD testbed is especially sensitive to human intervention into its activities because of its high degree of autonomy.

The primary purpose of the SSM/PMAD project is to automate a breadboard level power management and distribution system test bed. This functionality has been successfully implemented, resulting in a completely autonomous test bed. It has proven to be of significant benefit in terms of real-time management, fault containment, fault isolation, and fault recovery. In several cases the automated fault management system, FRAMES, performed fault diagnosis and recovery in only seconds which could otherwise have taken one or more man-days with a completely manual system. This was achieved in several demonstrations of managing both single and multiple faults within the test bed. However, the test bed possessed a significant weakness interfacing human intervention to the autonomous management software. The system behaved as do many other automated systems - if one desires manual intervention into the automated domain, then one must assume full manual control of all system elements, yielding any presently executing plan and schedule totally ineffective. To overcome this weakness and to ensure maximum plan and schedule continuity, new functionality for planning, knowledge management, and the user interface was added to the system's automated software suite, allowing incremental manual intervention instead of total. This is the need that led to ILA development.

The SSM/PMAD system now possesses the capability to add a human user to its autonomously executing plan and schedule. The system (at a user's request) autonomously introduces the user to the run-time environment, minimizing system-wide impacts and localizing the user's region of control. The key new technology developments which were

paramount in producing this operational system were:

1. Development of a run-time planning system (KANT, described elsewhere in this report) which could be reactive to the user's needs and could also function independently with the priority management and scheduling environments to provide alternatives to a user.
2. Provision for simultaneous multi-agent knowledge system processing, allowing a user to remain free from having to possess specific knowledge about system state in a rapidly changing environment.
3. Addition of "what-if" capabilities to the scheduling system and management of the complexities involving allowing the user to change a schedule while it is running.
4. Development of a seamless user environment so that a user didn't become overwhelmed by presentations of vast amounts of information from a highly distributed process and control system.

The following is a high-level definition for ILA.

Intermediate Levels of Autonomy Definition That level of directed autonomous systems activity that ranges between fully manual and fully autonomous but not including the end points.

It is important to recognize that with this definition a component control function can be formulated from both manual and autonomous elements simultaneously.

ILA Manual Seizure Interface Design

The first implementation of ILA did little in the area of defining a procedure for having the user be more involved in seizing a switch. The first cut worked out the internal mechanics for seizing a manual switch. The purpose of this section will be to discuss the original implementation of ILA and then discuss the transition to phase II of ILA. Phase II will start by giving a system overview and show basic control flow. The rest of the phase II discussion will cover specific interface design issues that are broken into three sections. The first section will discuss grouping of switches. The second will cover Manual Seizure and the third will be the process for Manual Release.

Discussion of Phase I In this phase, ILA activities allow the user to manually control specified switches during normal operations. These switches are specified by the user's selection and subsequent request for manual seizure.

In the normal mode of operation, the SSM/PMAD system must be able to reason about switches that the user has under manual control. The primary reason for this is for the

purpose of making adjustments to the schedule and being able to deal appropriately with them during fault situations. To support these needs, a manual seizure is represented as a schedulable activity and merged into the existing schedule being operated.

When the user chooses to manually seize a switch from autonomous control in the normal mode of operation, the user must specify some information about how it is used. This information includes when the switch is to be under manual control and for how long, what priority the manual seizure is compared to other activities on the schedule, the power requirements of the switch (in watts), and whether FRAMES has permission to test the switch if any faults occur in the system while the switch is under manual control.

When the user specifies all the necessary information, the ILA functions will determine if it is possible to place the switch on the schedule and prompt the user for one of the possible placements (usually there will only be one). These possible placements are indicated by a region of time over which the switch will be under manual control. If one of the possible placements is accepted by the user, the ILA functions will notify MAESTRO of the changes of the availability of the switch with respect to autonomous operations, causing MAESTRO to generate any further repercussions that may occur as a result. Once all the necessary changes to the schedule have been made and the switch has been represented in the schedule as its own activity, the switch will be automatically turned on (under manual control) at the start of the accepted choice.

The user has two ways of detecting when the switch is finally available for manual use. As the user seizes a switch for manual control, the ILA functions return a menu of choices of which the user selects one. The selected choice indicates when the switch will be turned on under manual control. The second indication to the user is that when the switch is finally turned on a *hand* icon will be placed on the user interface (on the particular switch) to indicate that it is now under manual control.

The preceding discussion should indicate that the manually seized switch is automatically turned on for the user at the start of the manual control period. During the time that the user has the switch under manual control, the user may turn the switch on or off (via the manual on and manual off choices on the user interface) at his convenience. Manually controlling the switch in this fashion does not indicate to the rest of the SSM/PMAD system that you are done with the switch in any way. These manual ons and offs are completely controlled by the user while the switch is under manual control. However, while the switch is under manual control, the allocated power originally requested by the user is what is available. Furthermore, even if the switch is off while under manual control, the FELES screen will indicate that the activity representing the manually controlled switch is still active.

When the period of time requested for the manual switch is over, the switch will automatically be turned off and given back to the autonomous operations (and made available to MAESTRO). The *hand* icon will no longer be visible. It is also possible for the user to release the switch from manual control early by utilizing the manual release function on the user interface.

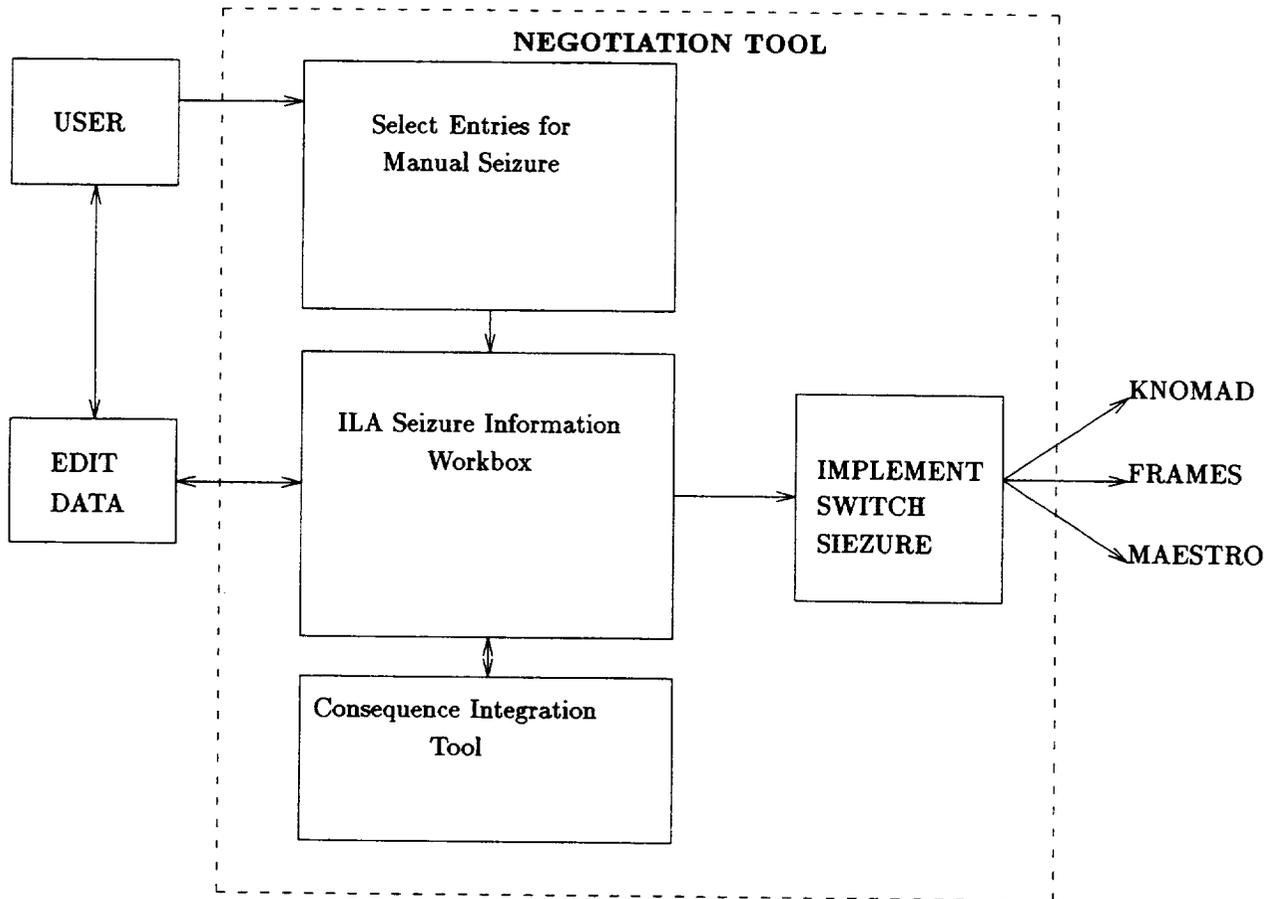


Figure 2.37: System Overview

System Overview and Flow Figure 2.37 is a flow diagram showing the main functions for transferring a resource from the system to the user via a Negotiation Tool. As can be seen from figure 2.37 the action starts with the user who requests control of some switch(es). These request are then displayed in a special workbox titled *ILA Switch Seizure Information*. The user controls what happens by changing the information about the switch being seized. When the user has updated the switch seizure information, the user asks the system to go out and calculate the probable effects the manual seizure⁴ will have on the current system. Note that this effects calculation is only a conservative guess as to the effects of the seizure; a complete effects analysis could be performed by MAESTRO (see the discussion on the Hypothetical scheduler). After the effects have been calculated they are displayed to the user again for information purposes. The user can go back and change some switch information, add or drop switches, or add or drop switch groups. This cycle of negotiation continues until the user is satisfied that the effects on the system are acceptable. At this time the requests are incorporated into the currently running system through the function *Implement Resource Seizure Tool*.

Groups The ability to group switches gives the user a way to tell ILA that a set of switches must be obtained in order to accomplish this task. A failure to obtain any switch in the set negates the entire group. This means that all the other switches in that group will not be seized. This type of grouping is very similar to the way a subtask is arranged within an activity. The use of groups is intended to make it easier for the user to define new activities on the fly for incorporation into the running system. For now ILA is only interested in power requirements but the next step is to handle all resources that make up an activity and many of these resources may not be power related at all.

The ability to group switches is performed through the user interface, specifically an option in the applications menu of the power system screen titled *Group Switches*. This selection will cause the *Group Switches* selection menu to appear (see figure 2.38 for an example). The workbox contains two data fields. The first field is the Group Name which will initially be blank. As with all workboxes it is the users responsibility to fill in the field with a unique name.⁵ The Second field, called Switches in the Group, is different from the standard workbox field in that it is not directly editable from within the workbox. The user goes through the menus selecting the switches that they wish to have in the group. Every time a selection is made, that menu item will become highlighted. To unselect any switch just select the highlighted item and it will become unhighlighted and thus drop from the group. When the user has finished selecting the switches from the group the <<GROUP SELECTED ITEMS>> menu item will put the selected switches into the workbox.

⁴The terms MANUAL SEIZURE and MANUAL CONTROL are interchangeable

⁵If the user enters a name that already exists for another group the workbox will notify the user and insert either a blank or the last valid group name that the user had entered. The same is true if the user selects the done button and no name has been entered.

SWITCH GROUPING	
Group Name:	<input type="text" value="Experiment 14"/>
Switches In The Group:	
LC-01 SW: 03	
LC-02 SW: 14	
LC-02 SW: 12	
LC-03 SW:21	
<input type="button" value="ADD/DROP"/> <input type="button" value="DONE"/> <input type="button" value="CANCEL"/>	

Figure 2.38: Switch Grouping

Manual Seizure *MANUAL SEIZURE* is the Application Menu Option of the Power System Screen that initiates the dialogue between a user and the system to negotiate the transfer of control of a resource under system control. The Manual Seizure option displays a menu that contains the individual switches as well as the groups that are available to negotiate for user control.

The menu items in the **Seizure Select Menu** are of three types (Please refer to figure 2.39). The first are the three Load Center Items. These menu items have pull-rights menus that contain the individual switches for that load center. So if the user would like to select switch 4 of load center 3 they would move the mouse to the load center three menu item and then select switch 4 of that pull-right menu. It is this combination of load center and switch that allows the user to select individual switches for user control. The second type of menu item in the Seizure Select Menu is the group item. This item is also pull-right menu but this menu contains the name of the individual switches that make up the group. At the present these items are not selectable, and show the user what switches make up the group. The user selects the group as a whole for the first phase. Upon selection of a switch item or a group item that item will become highlighted. This lets the user know what selections they have made to this point. If the user would like to de-select a switch or group, select that switch or group and it will become un-highlighted. The third menu item are the two action items in the menu. The first one of these is the cancel item. Upon selection of this item the process for taking user control will be abandoned. The second action item is the one marked **DONE** and the selection of this item will cause the next phase of ILA (negotiation) to be activated.

Negotiation The data is sent to the Negotiation tool in a Switch Seizure List with each element in the Switch Seizure List being a Switch Seizure Packet. The Switch Seizure Packet is made up of 11 elements whose definitions are detailed in table 1. When the data is first passed to the Negotiation tool, all the fields except the the name and the group are set to default values. Table 2 shows the 9 data fields and their default values.⁶ Negotiation displays the initial switch information to the user in a *ILA Seizure Information Workbox* with their default values (see figure 4). Notice the initial data in the result field of all the entries⁷ in the workbox of figure 4. They all contain the same result *REASON: RUN CALCULATE TO SEE EFFECTS* which means the data has not been run through the Consequence Interrogation Tool of the Negotiation Tool. The reason for this is that the Consequence Interrogation Tool must be invoked by the user via the Negotiation Tool and

⁶The internal representation of binary values is represented using the lisp T and NIL construct. To make it clearer to the user, the ILA user interface will display characters Y and N for yes and no.

⁷An entry is either a group of switches or an individual switch. When switches are grouped together there is only one result field for that group not a separate result field for each switch in that group.

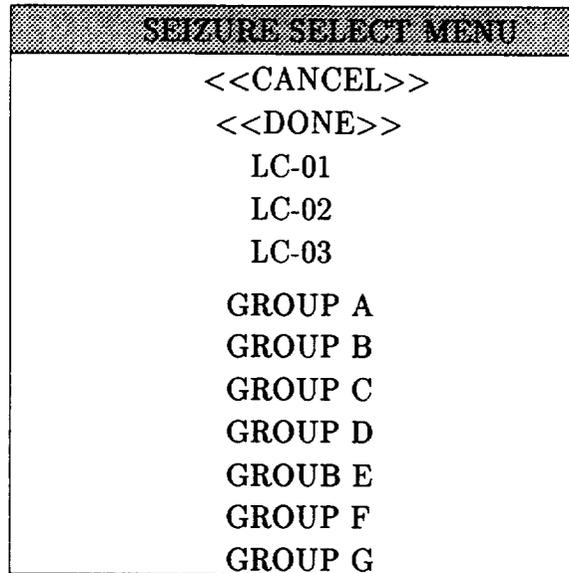


Figure 2.39: Seizure Select Menu

at this point the user has just input the data to the Negotiation Tool.⁸ The workbox has five buttons the user can select, they are CALCULATE, HELP, ADD/DROP, DONE, and CANCEL. The CANCEL button is like the cancel button on any workbox; it will abandon the Negotiation and no seizures will be implemented. The DONE button will implement any seizures that are allowed at this time.⁹ This means that if done were to be selected from figure 4 no switches would be seized because they have not been run through the Consequence Interrogation Tool. The ADD/DROP buttons will be discussed later.

The CALCULATE button will invoke the Consequence Interrogation tool on the current switch data. This tool will first order the data based on the static and relative priority. It uses the static priority to put the switch request into four categories and uses the relative priority to order the switches within that category.¹⁰ After the switches have been ordered,

⁸We could set it up to run the Consequence Interrogation Tool on the initial invocation of the Negotiation Tool with the default values that have been sent in but when doing a complex seizure of switches' the user is probably going to want to edit the switches default data.

⁹This means switches that have been accepted by the Consequence Interagation Tool. This can be determined by looking at the results field of an entry; if the result field starts with the word EFFECTS then it has been accepted.

¹⁰Grouped switches have the same relative priority so their ordering is non-deterministic. This is not a problem because ordering within a group would not make sense. If a switch in a group needed to have priority over another switch in that group for bumping purposes then the group would fail anyway.

the tool will take the highest priority switch and try to determine if that seizure is legal. If the seizure is not allowed the tool will collect the reasons why the seizure was disallowed and store that information into a Seizure Status Packet. This packet contains three elements. The first is the switch id, the second is the keyword :EFFECTS or :REASON, and the third is a string that contains all the effects or reasons. When the tool has finished with all the switches it will send the data back to the ILA Seizure Information Workbox to be displayed to the user (see figure 5 for an example of what might come back from the Consequence Interrogation Tool). The user decides if this is acceptable or not. If it is not the user has the option to edit existing switch information, add a new switch or drop a switch and then send the changes back to the Consequence Interrogation Tool. This sequence of steps (negotiation) continues until the user is satisfied with the results.

When the user is satisfied with the results, the DONE button is hit in the workbox and the Negotiation Tool implements the user's request. This is the final step of the negotiation tool. It will first package up the switch information from the *ILA Seizure Information Workbox* of the switches that are allowed for seizure into a Switch Seizure List. This list will then be sent to the *Implement Resource Seizure Tool* which is charged with letting MAESTRO know about the seizures as well as creating the pseudo activities for the switches being seized. This tool also will then merge the pseudo activities into the activity schedule so that the seizure for the switch will occur at the right time. The process to update MAESTRO may take several minutes to complete but that does not mean the switch(es) are not available to the user right away. If the user specified that they wanted the switch immediately then that switch will be under user control as soon as the DONE button was pressed. The process of getting MAESTRO up to speed does not hinder the users control of their resources. After the user has gained control of the switch(s) they can open and close them at will because that resource is under their control. It still must conform to the agreement made for seizure. This includes power constraints and priority for shedding.

Up to now, a broad explanation has been given on how the user can gain control of a system resource (only switches for now). Now its time to talk about some specifics: how the user edits the Switch Seizure Data, and how the user interprets the data in the ILA Switch Seizure Workbox. The next two sections will answer these questions.

Switch Seizure Information Menu Layout How does the user interpret the information on the ILA Seizure Information Workbox? Refer to figure 2.41. First notice that the data is broken into discrete groups which will be referred to as entries. They are called entries to prevent confusion with the concept of grouping switches. Each entry contains either a switch or a group of switches (A grouping of switches allows the user to show that these switches are working together to accomplish the same goal). Each entry is broken down into two parts. The first part is the switch description line(s). Figure 2.43 shows an example of an entry with a single switch and grouped entries. The Switch Description Line displays the Switch Seizure Packet in a shorthand notation. Refer to figure 2.44 for a break down the

Switch Name

This is a symbol that has a length of characters that follow the old format for switch identification in the SSM/PMAD database.

Starting Time

This is the time in mission time that the seizure of the switch is to begin. This number must be between 0 and 525600.

Duration

This is the length in minutes that the user wants to seize the switch. This number must be between 1 and 525600.

Static Priority

This is the priority category that this switch should be seized in. The integer must be between 0 and 3 with 0 being the highest priority and 3 being the lowest.

Relative Priority

This is the users ordering of the switches for a manual seizure event within a particular priority category. The user is required to make sure the ordering is correct or the calculation routine will error on bad input.

Power Consumption

This is an integer value between 0 and 1000 that represents the maximum amount of power that will be consumed by this switch.

Redundant?

This value is either T or NIL and it represents if the switch has redundant capability.

Testable?

This value is either T or NIL and it represents whether the switch is testable by the FRAMES system if a fault occurs.

Diagnosable?

This value is either T or NIL and it represents if the FRAMES system should bother trying to diagnose a fault on this switch if one were to occur.

Bumpable?

This value is either T or NIL and represents whether this switch can bump other switches within the same priority category when trying to seize this switch.

Switch Position

This value is either ON or OFF and represents the position the switch will be in upon seizure of the switch.

Table 2.1: Switch Seizure Packet

ILA Seizure Information Workbox						
LC-01 SW: 02 NONE [00.00:00 - 99.99:99] [3 - 1] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS						
LC-02 SW: 03 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF LC-02 SW: 04 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF LC-03 SW: 21 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS						
LC-01 SW: 23 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D OFF LC-03 SW: 02 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS						
LC-03 SW: 04 NONE [00.00:00 - 99.99:99] [3 - 4] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS						
LC-01 SW: 07 NONE [00.00:00 - 99.99:99] [3 - 5] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS						
<table border="1"><tr><td>CALCULATE</td><td>HELP</td><td>DROP/ADD</td><td>DONE</td><td>CANCEL</td></tr></table>		CALCULATE	HELP	DROP/ADD	DONE	CANCEL
CALCULATE	HELP	DROP/ADD	DONE	CANCEL		

Figure 2.40: ILA Seizure Information Menu

ILA Seizure Information Workbox

LC-01 SW: 02 NONE [00.00:00 - 99.99:99] [3 - 1] 1000W —D ON
EFFECTS: ACTIVITY GLOVEBOX HOUSKEEPING DISPLACED BY SWITCH SIEZURE
EFFECTED SWITCH(S): LC-01 SW: 02
LC-02 SW: 06
ACTIVITY STAR GAZER DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCHE(S): LC-01 SW: 04

LC-02 SW: 03 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON
LC-02 SW: 04 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF
LC-03 SW: 21 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON
EFFECTS: ACTIVITY EXO BIOLOGY EXP DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCH(S): LC-01 SW: 05
LC-03 SW: 03
ACTIVITY MICRO COUNTER DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCH(S): LC-02 SW: 06

LC-01 SW: 23 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON
LC-03 SW: 02 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON
REASON: LC-01 SW: 23 STATIC PRIORITY TO LOW
LC-03 SW: 03 RELATIVE PRIORITY TO LOW

LC-03 SW: 18 NONE [00.00:00 - 99.99:99] [3 - 4] 1000W —D ON
EFFECTS: ACTIVITY STAR COUNTER DISPLACED DUE TO LOSS OF REDUNDANT
SWITCH LC-03 SW: 18
EFFECTED SWITCH(S): LC-03 SW: 04

LC-01 SW: 07 NONE [00.00:00 - 99.99:99] [3 - 5] 1000W —D ON
REASON: LC-01 SW: 07 POWER REQUIREMENT EXCEEDS MAX ALLOWD ON 3K

CALCULATE HELP ADD/DROP DONE CANCEL

Figure 2.41: Updated ILA Seizure Information Menu

LC-01 SWITCH: 02	
STARTING TIME (dd.hh:mm):	00.18:30
DURATION (dd.hh:mm):	0.05:20
STATIC PRIORITY(0 - 3):	1
RELATIVE PRIORITY(1 - 99):	1
POWER CONSUMPTION (Watts):	500
IS SWITCH REDUNDANT (Y/N):	N
IS SWITCH TESTABLE (Y/N):	Y
IS SWITCH DIAGNAOSABLE BY FRAMES (Y/N):	Y
PERMISSION TO BUMP OTHER SWITCHES (Y/N):	Y
INITIAL SWITCH POSITION (O/F):	O
<input type="button" value="DONE"/> <input type="button" value="CANCEL"/>	

Figure 2.42: Switch Seizure Workbox

1. **Starting Time** 0 (Now)
2. **Duration** 525600 (Forever)
3. **Static Priority** 3 (Lowest)
4. **Relative Priority** Next value in the static leve based on input position.
5. **Power Consumption** 1000 (Highest)
6. **Redundant?** NIL (No)
7. **Testable?** NIL (No)
8. **Bumpable?** NIL (No)
9. **Diagnosable?** T (Yes)
10. **Switch Position** OFF

Table 2.2: Switch Seizure Packet Default Input

Switch Description Line. It should be pointed out that an “-” represents an uninitialized flag value. If those flags had been set, they would have been R for REDUNDANT, T for TESTABLE, B for BUMPABLE, and D for DIAGNOSABLE. Part two of the entry is the result field. The result field is the textual information that describes either the EFFECTS manual seizure will have on the system or a REASON the seizure was not allowed for this entry. A list of all possible EFFECTS and REASONS can be found in tables 2.3 and 2.4 respectively. Below each EFFECT will be listed the switches that are associated with that activity. These are the switches that will probably get shed. I said probably because we don't know what MAESTRO will do with these displaced activities. It might try to reschedule them or even move them to other switches depending on how the activity has been described. ILA is only giving us the worst case scenario of what might happen. One thing will be guaranteed and that is that the switches that the user is able to grab during ILA will not get shed due to the MAESTRO rescheduling effort associated with the manual seizure. This does not mean that the switch(s) could not be shed later due to a source power change, for example.

Editing Switch Seizure Data How does the user change the data displayed in the switch description line? This is done by using the mouse. Move the mouse to the switch name that needs to be updated (the switch name has the format LC-xx SW: xx) in the

1. Activity xxxxxx Interrupted Due to Seizure of Switch LC-xx SW: xx
2. Activity xxxxxx Interrupted Due to Displacement of Activity xxxxxx
3. Activity xxxxxx Interrupted Due to Redundant Needs on Switch LC-xx SW: xx
4. Activity xxxxxx Interrupted Due to Loss of Redundant Switch LC-xx SW: xx

Table 2.3: Switch Seizure Effects

1. Relative Priority too Low for Switch LC-xx SW: xx
2. Static Priority too Low for Switch LC-xx SW: xx
3. Power Requirement Exceeds Max for Switch LC-xx SW: xx
4. Unable to Grab Redundant Switch for Switch LC-xx SW: xx
5. Run Calculate to See Effects

Table 2.4: Switch Seizure Failure Reasons

ILA Switch Seizure Information Workbox. That name is mouse sensitive so when the mouse enters the switch name region the name will be highlighted. When the name is highlighted, a mouse click will cause a workbox containing the data for that switch's seizure packet to be displayed.¹¹ The user can then update any field in the workbox using the standard workbox editing commands. An Example of the Switch Seizure Workbox can be found in figure 2.42.

The Add/Drop Option What does the user do to either add or drop a switch from the switch seizure list? This is done by using the Add/Drop button in the workbox. Upon selection a menu will appear in the main window of the power system screen (see figure 2.45). This menu contains at least three items. The first item will be labeled <<<SEIZE SELECTED ITEMS>>> and its use will be discussed later in this section. Each load center has a pull-right that contains all the switches in that load center. When the mouse enters the region of a menu item that contains a pull-right menu the pull-right menu will be activated. The user uses the pull-rights to select the individual switches they would like to add or drop from the switch seizure list. An item is highlighted if it is selected and selecting a selected item will cause it to be unselected (dropped). This is how the user can add and delete

¹¹The switch description of an entry is the shorthand display of the switch seizure packet.

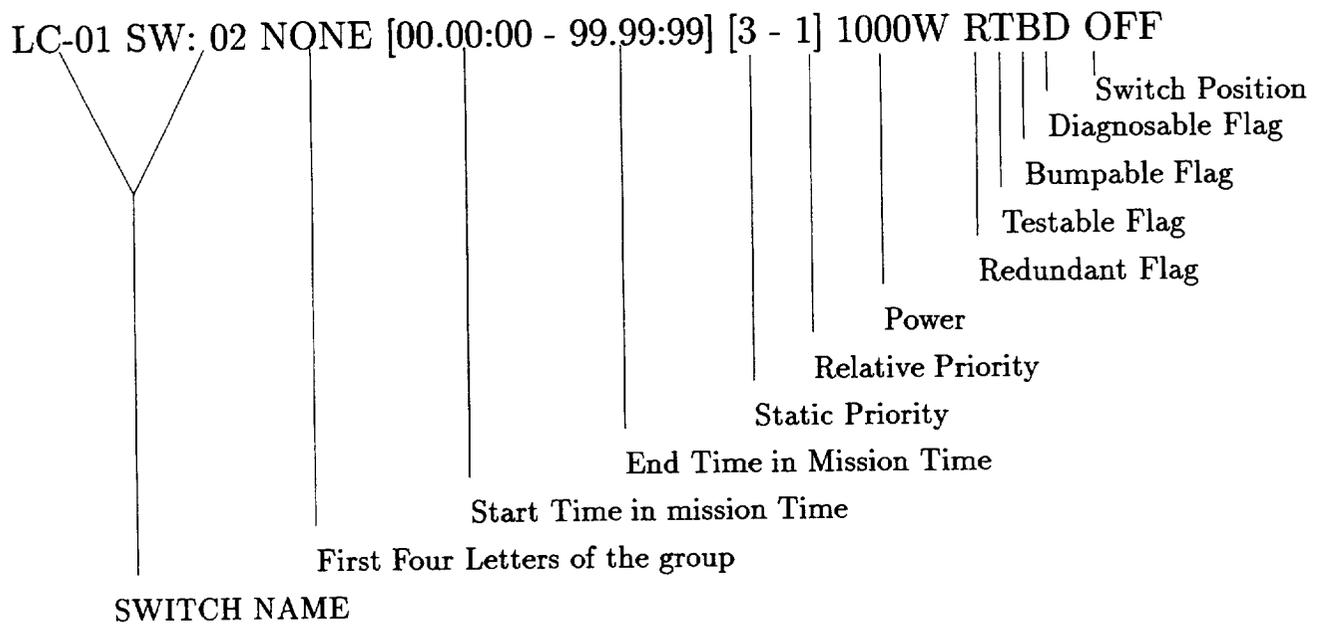


Figure 2.44: Switch Description Line Reference

individual switches from the switch seizure list.

What about Groups? All the items listed after the *Load Center & PDCU* pull-right components are the groups in the system. Please refer to figure 2.45 again. By using the mouse and selecting a group the user can add or drop a group from the switch seizure list.

When the user has finished adding and dropping switches or switch groups, the user will select the <<SEIZE SELECTED ITEMS>> option. This will cause the ILA Seizure Information Workbox to be updated to reflect the new switch seizure list. Any switch that has been added to the list will have the default REASON for failure *RUN CALCULATE TO SEE EFFECTS* so the user will have to run calculate again.

Manual Release Once a user has seized some switches to perform a task, that switch will stay under manual control for the duration specified in that switch's Switch Seizure Packet. In order to release that switch back to the system prior to that time the user must select the *Manual Release* option which is located in the Applications Menu of the power system screen.

When the *Manual Release* option is chosen a menu is popped up that contains all the current manually controlled switches and switch groups. Some of these entries will occur as individual switches while other entries will be groups. This menu is just like the one used in the ADD/DROP button of the ILA Seizure Information Menu. It is, except for the title of the menu and the action that is taken when an item is selected. Figure 2.46 is an example of how the Manual Release Menu might look. Selecting an item from the menu that is just an individual switch will cause that switch to be returned to the system. Selecting a group will cause the switches in that group to be released back to the system.

What happens when a switch is released back to the system? First MAESTRO must be notified that this switch is now available for general use. MAESTRO will then try to reschedule any activities that might have been displaced due to the manual seizure of that switch. This implies that the switch could be used again by the currently running system.

User Schedule Change

This section discusses recent enhancements made to the SSM/PMAD system that allow a user to modify a running schedule. This capability represents a higher level of user intervention than that discussed in the preceding section, in that the addition of an activity to a schedule represents user control of multiple switches over a period of time, without the user having to specify the switches to be controlled or the duration of control for each. To facilitate allowing the user to change a running schedule, the hypothetical scheduler (HYPO) was developed. It differs from the existing MAESTRO scheduling system only superficially, primarily in how it is exercised and in its role within the larger system. This section describes those aspects of HYPO that are unique to its operation and not covered in the section on MAESTRO.

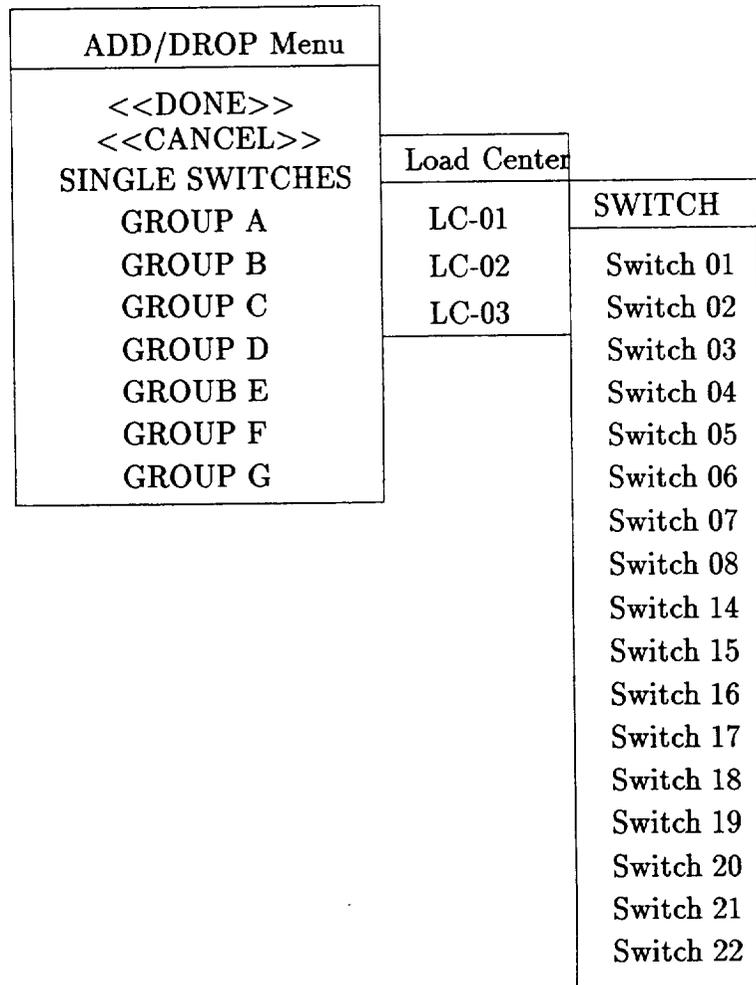


Figure 2.45: Add Menu with Load Center Pull Right

Manual Release Menu	
LC-01 SW: 04	
LC-01 SW: 16	
LC-02 SW: 03	
LC-02 SW: 06	
LC-03 SW: 19	
GROUP1	
GROUP2	
	SWITCHES
	LC-01 SW: 04
	LC-02 SW: 20
	LC-02 SW: 18

Figure 2.46: Manual Release Menu

Hypothetical Scheduling The HYPO system is so named because it supports the user's intent to test hypotheses about schedule changes before making them a part of the running schedule. The user will typically not be able to foresee all the ramifications of adding an activity, so s/he can use the HYPO scheduler to see exactly what those ramifications are and accept or decline them with full knowledge of the consequences of his actions. This is analogous to the function of the KANT system with respect to manual seizure of switches, but presents a higher-level view. The user changes the schedule, looks at the result, then either 1) changes it more, 2) discards the changes or, 3) implements the changes.

The specifics of how the user interacts with HYPO are described in the section on the User Interface above, but a brief summary is included here. The user selects CHANGE SCHEDULE from the main SSM/PMAD menu. A HYPO screen appears which has options for activity, equipment and resource editing as well as schedule modification. The user acknowledges (or changes) the time the altered schedule will replace the running schedule (if the user chooses). Then the user selects an activity to add to the running schedule. A workbox presents the user with opportunities for starting the activity and allows the user to change the activity's priority to increase or decrease those options. The user selects a start time or range of start times, and the system places the activity, possibly bumping other activities (according to its priority and opportunities). The user can then add another activity, select an activity for deletion, begin the schedule modification process anew, cancel and continue with the old schedule, or implement the changes made.

Implementation Details The HYPO system is, as alluded to earlier, a copy of the MAESTRO scheduling system which has been ported to a different LISP package and modified appropriately. When the user first brings up the HYPO system, a copy of the running schedule, with any modifications made during contingency handling, is written to a file and read in by the HYPO system. All data are either loaded from disk or read in from the KNOMAD database such that there is no duplication of data structures with the MAESTRO system. When the user selects an activity to schedule, the UI sends a message to HYPO indicating which activity was chosen. After initializing an internal representation of the activity, HYPO calculates all possible placement opportunities for the activity and returns these to the UI via a slot on the SSM/PMAD frame. When the user selects a start time window for the activity and clicks on OK, the endpoints of that window are passed to HYPO, which uses its heuristics to place the activity and returns a list of changes made to the schedule, including that placement as well as unschedulings and other changes made to accommodate that placement. If the user chooses to change the activity's priority, the new priority is sent to HYPO, which responds with a new list of opportunities (calculated against a different schedule, e.g., red, blue or gold).

The user can select an activity performance to delete, in which case the identifier and start time are passed to HYPO, which deletes the activity performance, updates resource profiles, etc., and returns confirmation to the UI. If the user selects START OVER, HYPO is re-initialized as if it had never yet been brought up.

If the user selects IMPLEMENT, the changed schedule resulting from modifications that have already been communicated to the KNOMAD database and UI are instantiated as the active schedule, the HYPO UI screen is replaced by the normal UI, and the LLPs quite executing the old schedule and begin executing the new one at the time specified at the start of the HYPO process.

Timing Issues It is important to note that there are restrictions on what the user is allowed to change on a running schedule. These restrictions are imposed by the fact that changes the user makes must be communicated to the LLPs before they are to be implemented. Thus, a user is prohibited from changing a schedule such that a new activity would start at a time prior to the time the user would direct the adoption of the schedule changes. The UI enforces this restriction.

2.6.11 Soft and Incipient Faults in the SSM/PMAD Test Bed

The implementation of soft and incipient fault handling has been considered in detail from the concept of what constitutes such fault conditions and how an observer of the SSM/PMAD Testbed should be alerted to their occurrence or presence. The following is the result of knowledge engineering Martin Marietta performed in these areas. Augmented power management has been implemented and is used in soft fault management. It is important to note

that augmented power management remains consistent during periods of fault conditions, user invoked ILA or both. The fault conditions and their descriptions are discussed in the following paragraphs.

Soft Faults Soft faults are presently classified as one of two types.

1. The first type of soft fault concerns the use of power above a budgeted limit but not exceeding an absolute power limit. This suggests a new concept for power management as follows:
 - First, an overall power assignment is made to the testbed.
 - Then, the testbed generates an overall plan and schedule utilizing all the power available within the assignment. This is known as the budget. Therefore, for a certain period the assigned power level (the budget) will be used as well as possible. If the scheduled power exactly matches the assigned power, then no excess power can be used within the overall assignment. However, if there is an amount of power remaining after a plan and schedule have been generated, then that excess can be allocated to loads based on their priority and their required power level. But in no case can a load be allowed to consume beyond its assigned budget. Also the sum of the power for all loads may not exceed the initial assignment. The budget as managed by the testbed will be readjusted dynamically based upon the power assignment. Figure 2.47 shows the *SSM/PMAD* Test Bed power management concept which is used when not including soft faults. Figure 2.48 depicts how the augmented power changes the power management concept creating a better approach to total power allocation.
2. The second type of soft fault takes place at sensors, yielding resource readings that are out-of-balance. These are discovered by performing balancing calculations. Sensors are sensitive to calibration and the periodicity of the associated calibration operations. The balancing calculations are performed as applications of Kirchoff's current law summing the currents into and out of nodes. This is a tractable implementation given the hierarchical control and star bus configuration of the *SSM/PMAD* Testbed.

Incipient Faults One type of incipient fault has been included in the FRAMES fault-recognition knowledge bases. Refer to Figure 2.49 for the hierarchy of the present incipient fault conditions. The incipient fault is:

1. **MTBF Expiration.** - This fault is one which will be flagged whenever the mean time between failures (MTBF) is within Δ time of occurring. Δ is defined as the time a given piece of powered equipment is scheduled to be in an on state.

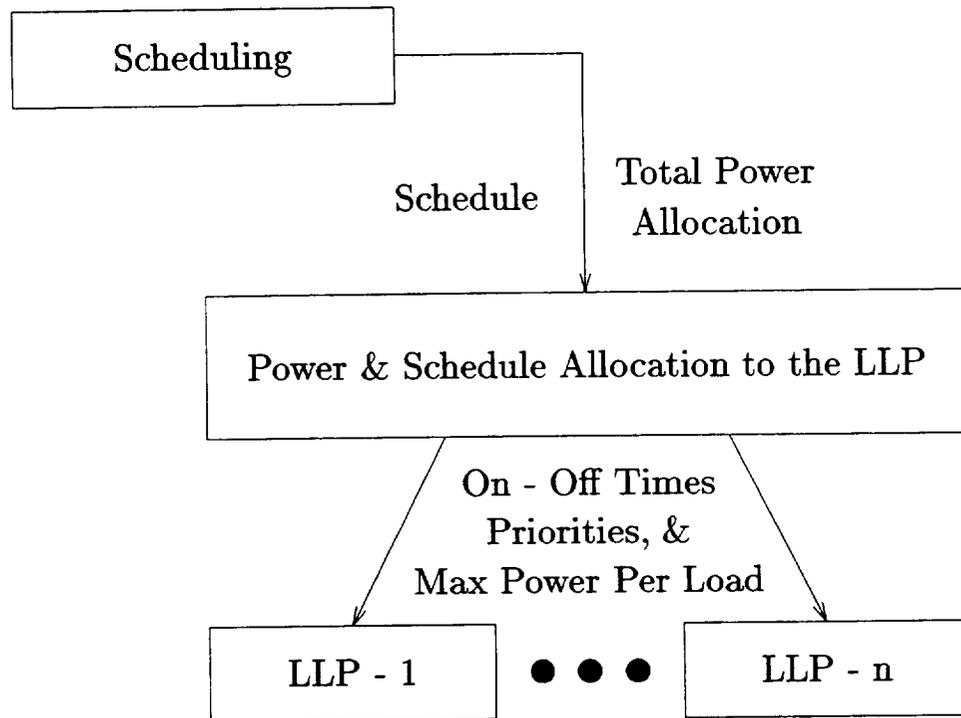
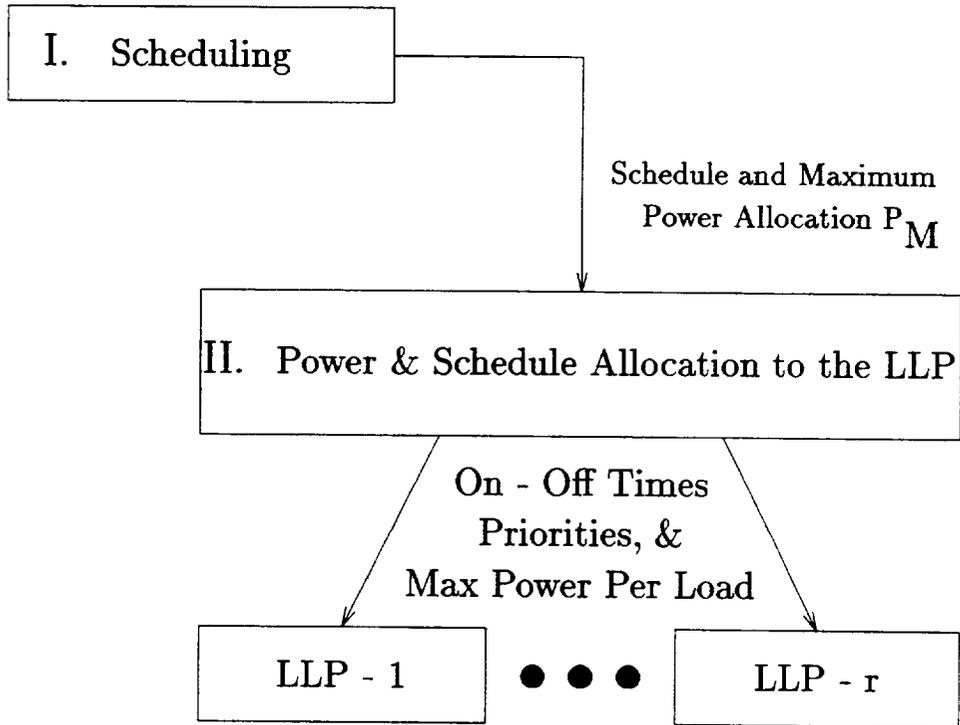


Figure 2.47: Present *SSM/PMAD* Test Bed Power Management Process



$$\text{Total Maximum Power Used} = P = \sum_{i=1}^n p_i$$

Allocation Rule: If $P < P_M$
 Then, Allocate $(P - P_M)$
 to the Loads According to their
 Requested Augmented Power and Priority

Figure 2.48: Power Management Process Allocating Unused Power

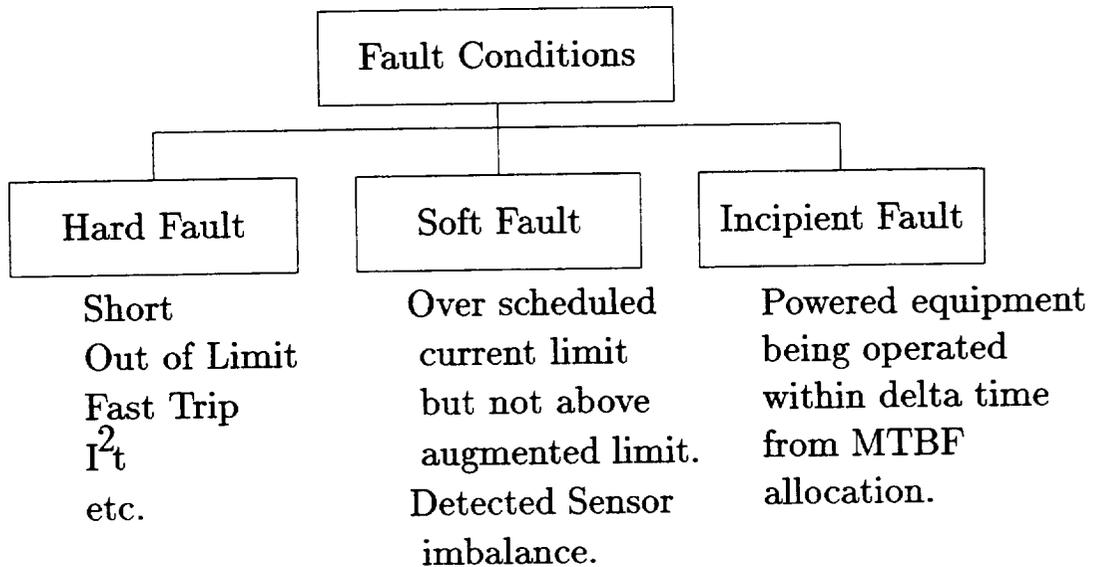


Figure 2.49: The SSM/PMAD Testbed Fault Type Hierarchy

Incipient faults will remain assigned to the powered equipment on which they occur until they are *specifically cleared by an operator*. This provides user insight whenever the powered equipment items are accessed for operation. The form of the information will be an incipient fault detection message written to the incipient fault detection window in the lower left corner of the user interface screen.

Clearing Soft and Incipient Faults

The above scheduled current limit soft fault which is reported by the LLP is clearable by using the clear soft fault menu option in the utilities submenu of the system options menu on the user interface screen. This will change the color of the switch icon from gold (i.e. soft faulted), back to green (i.e. good) on the power system screen. This is used as an acknowledgement by the user to the testbed.

The sensor balancing soft fault which tests each node of the power system for sensors out of balance, presently has no method for clearing a given soft fault. However, the user may disable the sensor balancing feature of the testbed by selecting the disable sensor balancing option in the utilities submenu of the system options menu of the user interface. The default state for the sensor balancing option in the system is disabled. It may be enabled by selecting enable sensor balancing from the utilities submenu of the system options menu on the user

interface.

The MTBF incipient fault may be cleared in two ways. The user may select the clear incipient fault menu option in the utilities submenu of the system options menu. Doing this acknowledges to the testbed that the user knows the given piece of powered equipment has passed its MTBF and does not wish to be continuously alerted to this fact each time the equipment is turned on. This acknowledgement is not stored to the permanent database, and therefore will need to be repeated for a new run. The only other way an incipient fault may be cleared is to adjust either the minutes of operation field or the MTBF field in the powered equipment description on the activity editor screen. This might be done if piece of equipment had been replaced.

Reporting at the User Interface

During the occurrence of soft and/or incipient faults two types of reporting can occur. These are:

1. Messages alerting a new soft or incipient fault condition in the appropriate fault window in the lower left corner on the user interface.
2. An above scheduled current limit at the LLP is reported with the changing of the switch icon color on the user interface.

Figure 2.50 shows the general layout of the Workstation User Interface for the testbed. Soft fault diagnosis messages and incipient fault detection messages will be displayed in the appropriate power change and fault detection windows. Soft faulted switch icons will be displayed in the application screen for the power system application.

Power Allocation Augmentation

Power Allocation Augmentation In reference to Figure 2.48, an initial power allocation should specify how any unused power may be allocated to the loads operating within a given schedule duration. The power allocation may not exceed the bus constraints of the power system. (See figure 2.51) The allocation should be directed by load priority and requested augmented power. The algorithm for allocating the augmented power is straightforward and is described as follows:

1. Take all activities which have requested augmented power and order them by priority.

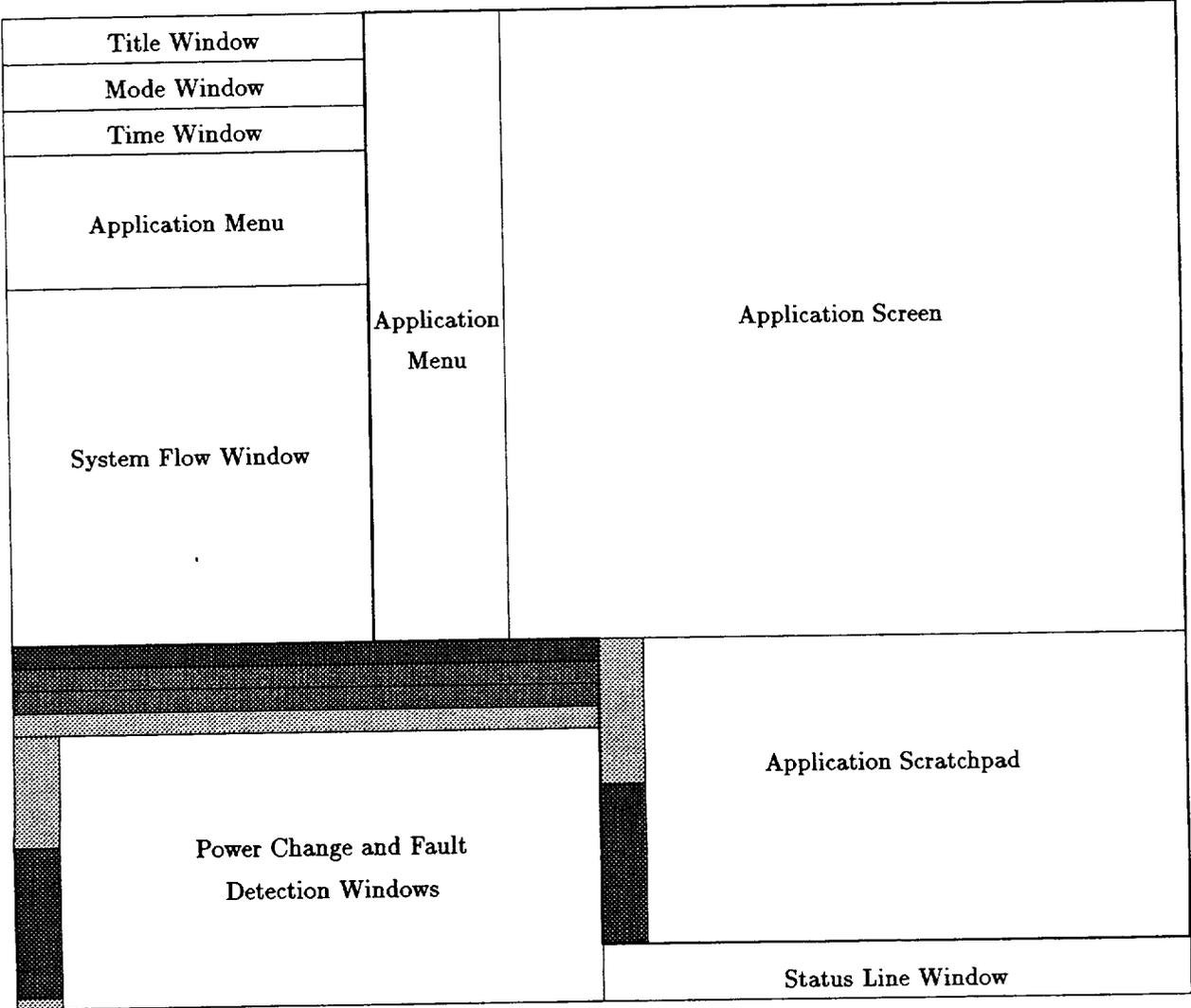
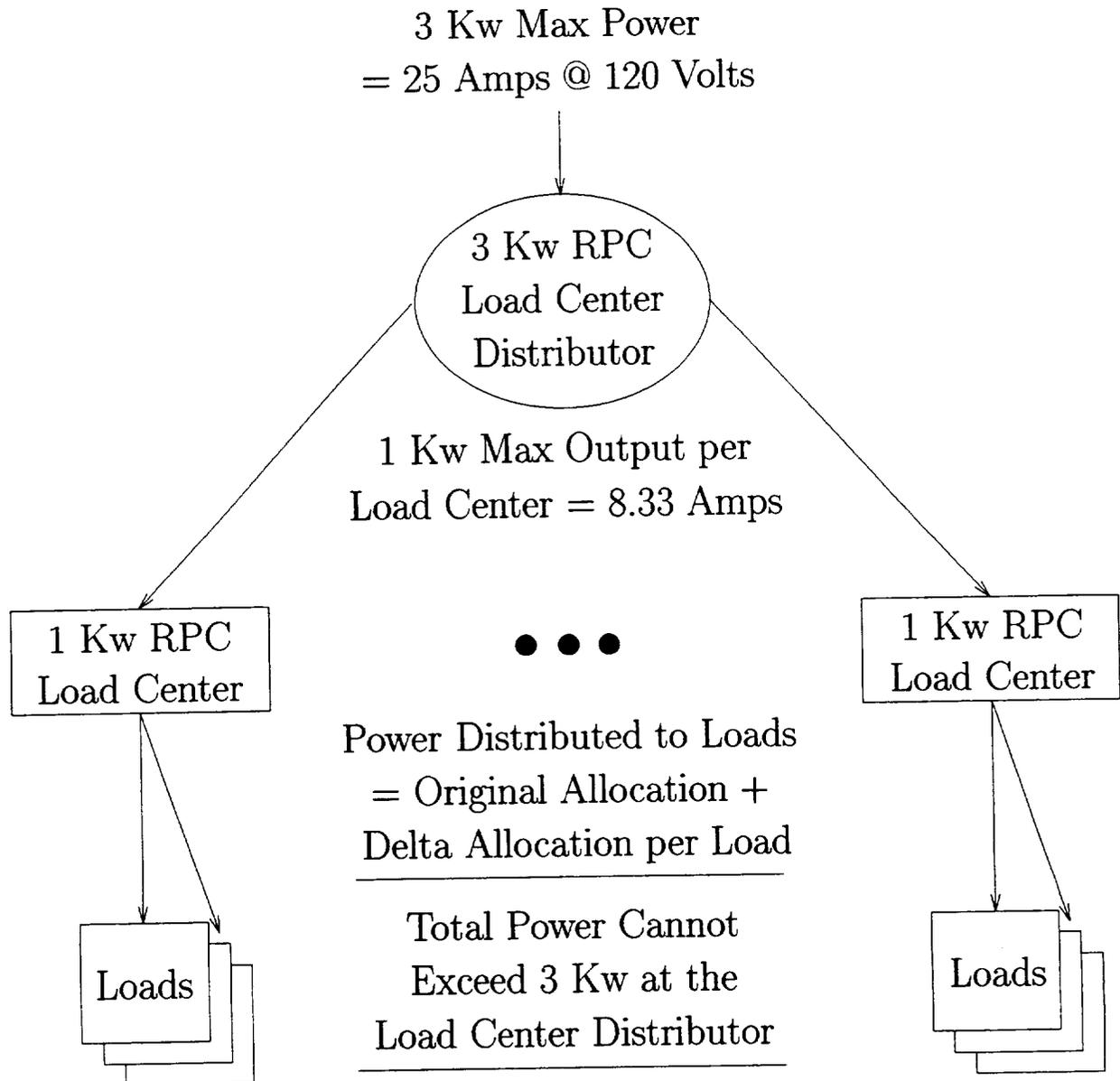


Figure 2.50: The SSM/PMAD User Interface Screen Layout

2. Proceed through the ordered list of prioritized activities and allocate unused power to each activity in turn, if enough power is available to meet the requested augmented power for that activity and the power capacity for the bus supplying power will not be exceeded. Power allocated in this fashion is no longer considered unused.

This provides a maximum power to loads which requested the unused power in order of priority. Since the requested augmented power is specified in the powered equipment description on the activity editor screen, it may be changed in that description, allowing for differing augmented requests.



Therefore, the original allocation plus all delta allocations must be less than or equal to 3 Kw, and the original allocation plus all delta allocations at a Load Center must be less than or equal to 1 Kw.

Figure 2.51: The Management and Control Allocation Constraints

(This Page Intentionally Left Blank)

Chapter 3

Future Directions

There are two different topics to be discussed under the heading of future directions. One is further enhancements that could be performed on the SSM/PMAD testbed. The other is technology transfer opportunities, other domains where the technology developed for SSM/PMAD could be applied. This chapter suggests potential enhancements, then discusses possible future applications.

3.1 Testbed Enhancements

Though much has been accomplished over the life of the SSM/PMAD project, there will always remain the potential for improving SSM/PMAD. The main areas requiring improvement to support realistic Space Station operations are those of redundancy switching management and load shedding.

When a redundantly powered load is scheduled, its redundant power source is not reserved, as this would waste power in the vast majority of cases. However, in those cases where the redundant source is needed due to a fault on the primary power path, the system will try to shed other loads only within that load center, forcing either the redundantly powered load to be denied power or in some cases allowing the redundant power use and causing an over-current at a higher-level RPC. These are caused by the system's need to restrict the scope of the load-shedding to each load center.

It is also possible for a fault to occur such that a redundantly powered load is switched to a higher-level RPC that is scheduled to begin supporting one or more other loads during the time MAESTRO is performing rescheduling, and MAESTRO may make different assumptions about which activities to shed than the LPLMS, which would cause the updated schedule to include loads that were shed and not include others that were on. This could be solved by giving the scheduler a module that duplicates LPLMS reasoning, allowing it to predict which loads the LPLMS would shed on those situations.

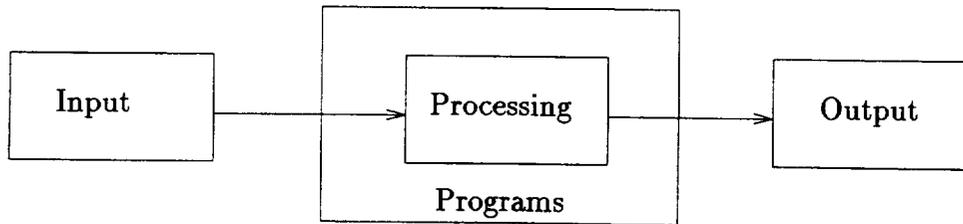


Figure 3.1: Data Management Environment

The user schedule change capability recently added could be augmented to show the user a single picture of opportunities with respect to all four schedules allowing a choice of placement for an added activity that would only affect others the user intended. Also, an UNDO capability was found require more effort than was available, and could be implemented to allow changes to the HYPO schedule to be rescinded, giving the user a more flexible what-if capability.

3.2 Technology Transfer Possibilities

The SSM/PMAD system was developed to provide a power management and distribution system. Although the system was developed in response to a desire to enhance power management on board a spacecraft, the hardware and software developed and the lessons learned could be applied to other environments. The technology employed could be used in systems as diverse as data management, robotics, control processing, and marine environments. Examples of each are discussed below.

3.2.1 Data Management Environment

A data management environment could benefit from a fault diagnosis and recovery system. A simplified data management view of a system containing input to a processing environment, processing to be done, and output from the processing is depicted in figure 3.1.

The input is from a device - hard disk, optical disk, solid state primary memory, tape, CDROM, or a telecommunication network. The processing, the focal point of the system, is done on a "very large scale integrated" (VLSI) system with 486 or 586 processors, cache memory, etc. The output is to any of the aforementioned devices. However, the device could even be another process. If the output is another process, the system view is instantiated with the new process. The input to the new process is the output from the previous process.

The process is really just a set of programs which when executed accomplish a goal. The process monitors the state of the system and graphically displays the results. The SSM/PMAD system monitors and displays the power system containing the RBIs and RCPs. The process has the intelligence to know the origin and destination of all data being used. With this type of prescient knowledge, the process averts disaster and increases productivity by expediently recovering from faults. For example, if the process monitors the temperature information for the device providing the data and the temperature exceeds an acceptable limit, alternative actions are executed. If the data is not critical, the device shuts down, diagnoses and corrects the problem, and then the interrupted process is initiated again. If the data is critical, redundant capabilities built into the system provide uninterrupted processing.

3.2.2 Robotics Environment

Robotics could also benefit from a fault diagnosis and recovery system. A telerobot is generally equipped with an arm, a manipulator, camera eyes, and an artificially intelligent brain to do complex assembly and maintenance tasks with minimal directives. The telerobot can be directed manually using a hand controller. Equipped with sensors, the telerobot sends back readings which generate the next sequence of movements. The telerobot can also grab objects through the use of a manipulator and manipulator controller.

The entities defined in the SSM/PMAD environment - activities, subtasks, powered equipment, and modes - could be redefined for a robotic environment. ILA-like capabilities developed for the SSM/PMAD environment would give an operator the option of assuming partial control of the telerobot to test components while the telerobot continues to execute the designated schedule.

A fully automated robot is needed for situations where a human's life is endangered. In a nuclear spill, for example, a robot could be programmed to enter a contaminated area and provide accurate camera shots of the damage. The robot knows the layout of the room as depicted in 3.2.

Using the floor plan of the contaminated area, an intelligent robot could negotiate around objects scattered by an explosion and position itself close to the spill as depicted in figure 3.3.

If the robot loses power and a fault occurs preventing its movement in one direction, the robot might recover by choosing another set of moves in another direction to accomplish the same task. These "what-if" scenarios are the level of sophisticated processing that ILA is currently capable of exercising.

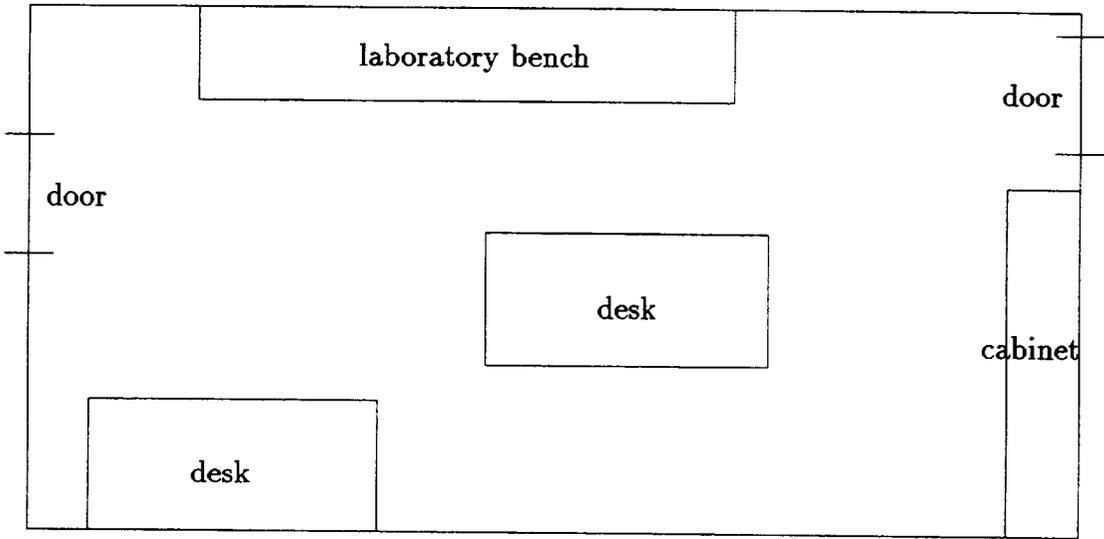


Figure 3.2: Robot A Environment

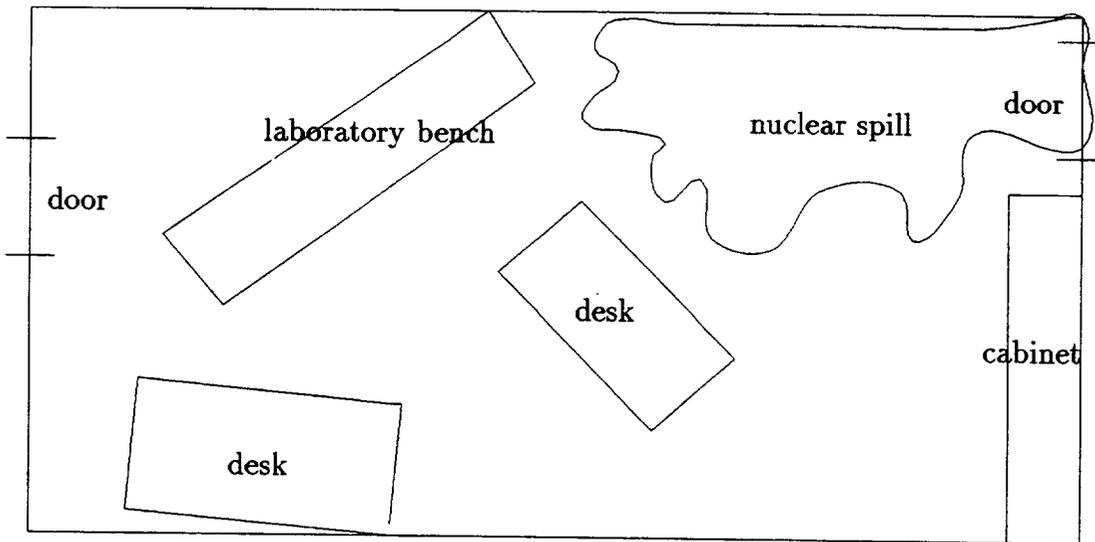


Figure 3.3: Robot B Environment

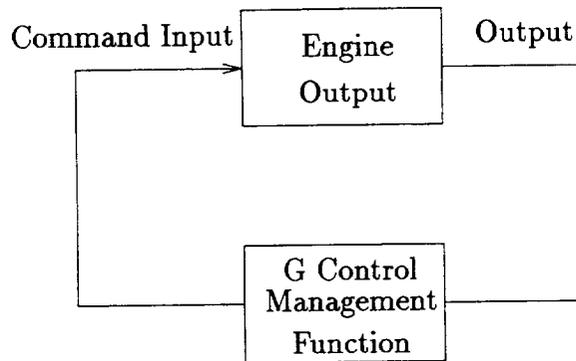


Figure 3.4: Marine Environment

3.2.3 Control Processing Environment

Chemical processing is an example of a control process where SSM/PMAD technology could be employed. Chemical processing starts with raw elements or compounds which are combined to produce a product. The other variables in the system are the supply of the raw elements or compounds, power systems utilized, equipment required, available facilities, personnel needed, mixing procedures followed, and budget and timing constraints.

Take, for example, mixing procedures which call for vat A with 200 pounds of compound X to be mixed with vat B with 400 pounds of compound Y. Two conveyors are needed to transport the materials within Z amount of time in order for the conveyor belt to be cleaned before the next stage of the process. If this can not be accomplished, SSM/PMAD technology could help determine the recovery possibilities after the fault occurs. Recovery procedure might include using another facility's conveyor. The time and cost impact of switching to another facility's conveyor could be projected using ILA technology.

3.2.4 Marine Environment

Marine environments could also be the beneficiaries of SSM/PMAD technology. Besides monitoring and charting steering and course control, the most obvious use of SSM/PMAD technology would be the management of power output for the engine. A simplified view of a closed loop design is shown in figure 3.4.

The control management function G, would monitor the output and control the input. Fault tolerance would be embedded in the G-function.

(This Page Intentionally Left Blank)

Bibliography

- [CG89] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, 32(4), 1989.
- [DeK86] Johan DeKleer. An assumption based tms. *Artificial Intelligence*, 28(2):127-162, 1986.
- [GBG91a] A. Geoffroy, D. Britt, and J. Gohring. Maestro technical reference. Contract No. NAS8-39433 MAESTRO Technical Reference MCR-91-1360, Martin Marietta, June 1991.
- [GBG91b] A. Geoffroy, D. Britt, and J. Gohring. Maestro users manual. Contract No. NAS8-39433 MAESTRO Users Manual MCR-91-1358, Martin Marietta, June 1991.
- [Jr.89] Guy L. Steele Jr. *Common LISP The Language*. Digital Press, second edition, 1989.
- [Rie90] Joel D. Riedesel. Knowledge management: An abstraction of knowledge base and database management systems. Technical Report Contractor Report 4273, NASA, January 1990.
- [WT78] et al. Warren Teitelman. *InterLISP Reference Manual*. Xerox Palo Alto Research Center, third revision edition, 1978.

(This Page Intentionally Left Blank)

Glossary

Fault Isolation A method for determining the cause of a fault in normal mode of operation.

Fault Reporting Displaying the fault when it happens and any diagnosis and recovery which may occur following the fault.

KANT The Collection of Planning and Negotiation Software Activities within the SSM/PMAD System

KNOMAD-SSM/PMAD Knowledge Management and Design Environment applied to the SSM/PMAD Domain.

Limit Checking A function which tests the amount of current which is flowing through a switch and makes certain that the amount is within specified limits.

Load Shedding The process of turning off loads within a load center in response to a contingency.

Performance Monitoring The capability for a user to observe how power is being utilized compared to how it was scheduled.

Redundant Switching The function switching an important load to a redundant power source when the primary power has been interrupted due to a fault.

RS-423 Electronic Industries Associates RS Standard for Communications. The RS-423 Communications Between the LLPs and the SICs

Schedule Execution The sequential execution of events which cause switches to given loads to be turned on and off.

SSM/PMAD Interface Used to Operate the System in Both Normal and Maintenance Modes. Initializes and Stops the System

System Operation Running the SSM/PMAD system through the user interface to control the hardware elements of the testbed.

~~REVISIONS PAGE BLANK NOT FILLED~~

(This Page Intentionally Left Blank)

Appendix A

Relevant Papers

PRECEDING PAGE BLANK NOT FILMED

**USER INTERFACE DESIGN PRINCIPLES FOR THE SSM/PMAD
AUTOMATED POWER SYSTEM**

Laura M. Jakstas

Chris J. Myers

Martin Marietta Astronautics Group

MS-S0550

P.O. Box 179

Denver, Colorado 80201

jakstas@den.mmc.com

chris@daneel.den.mmc.com

ABSTRACT

Computer - human interfaces are an integral part of developing software for spacecraft power systems. A well designed and efficient user interface enables an engineer to effectively operate the system, while it concurrently prevents the user from entering data which is beyond boundary conditions or performing operations which are out of context. A user interface should also be designed to ensure that the engineer easily obtains all useful and critical data for operating the system and is aware of all faults and states in the system.

Martin Marietta, under contract to NASA George C. Marshall Space Flight Center, has developed a user interface for the Space Station Module Power Management and Distribution (SSM/PMAD) automated power system testbed which provides human access to the functionality of the power system, as well as exemplifying current techniques in user interface design. The testbed user interface was designed to enable an engineer to operate the system easily without having significant knowledge of computer systems, as well as provide an environment in which the engineer can monitor and interact with the SSM/PMAD system hardware. The design of the interface supports a global view of the most important data from the various hardware and software components, as well as enabling the user to obtain additional or more detailed data when needed. The components and representations of the SSM/PMAD testbed user interface are examined in this paper. An engineer's interactions with the system are also described.

1 NOMENCLATURE

FELES Front End Load Enable Scheduler

FRAMES Fault Recovery and Management Expert System

LPLMS Load Priority List Management System

MSFC (George C.) Marshall Space Flight Center

NASA National Aeronautics and Space Administration

RPC Remote Power Controller

SSM/PMAD Space Station Module (Automated) Power Management and Distribution
(System)

2 INTRODUCTION

The SSM/PMAD is an intelligently controlled autonomous power management and distribution system. The SSM/PMAD is the result of a project to automate a testbed level

power management and distribution system which possesses many characteristics of a specified Space Station Power System. The automated portions of the system, leading to its autonomous capability, were developed under a NASA/MSFC contract called "Space Station Automation of Common Module Power Management and Distribution System". The objective is to develop autonomous intelligent power control software elements with smart power hardware developed under a NASA/MSFC contract called "Common Module Power System Network Topology and Hardware Development".

Inherent in the system is the capability to perform diagnosis whenever a distribution fault is encountered. The system autonomously reconfigures its operation during run-time and reschedules activities around the fault.

Designing an effective user interface for SSM/PMAD is important because a user should be able to operate the breadboard without having special computer skills. The interface should also ensure that the user can easily obtain all useful and critical data for operating the system and the user is aware of all faults and states in the system.

3 GENERAL PRINCIPLES

The SSM/PMAD user interface was designed to conform to certain guidelines. It was intended to be a module independent of other software modules. This was accomplished by designing the user interface independently of the other modules, as well as enforcing the concept of 'event - driven' modules receiving data and performing computations while displaying their results on the workstation monitor. The result of this design is that modifications can be made to the user interface without having impacts on the rest of the system.

Certain concepts were followed in the strategy of the user interface design. The first of these, consistency, allows a user to learn to use the system in a small amount of time and recall operations easily. Another concept is ease of operation. A system should not require the user to remember special command line languages, nor have any special computer skills. An effectively designed system takes into account these concepts in its design. The third concept, perhaps of most importance, is that the user needs to be alerted of all critical information. The user also needs to have the capability of accessing secondary data. If the user is not made aware of all of the faults and states of the system, he may make an incorrect assessment of the power subsystem. The ability to access secondary data provides a user with the capability of focusing on specific parts of the power system and thus having a more accurate view of the activities within.

Certain requirements were addressed in the development of the user interface. The testbed needs to be represented in a graphical manner. The user should be able to manually turn the switches on and off. The

user should always be able to identify whether the system is in autonomous, manual or semi-autonomous mode. System messages and information about diagnoses should be

readily visible. In general, the user should be able to access all data which is critical to operating the power system breadboard, as well as data which may not be critical for the user, but may be of interest.

4 SSM/PMAD USER INTERFACE ORGANIZATION

There are four screens that comprise the SSM/PMAD user interface. These are the Power System screen, the Front End Load Enable Scheduler (FELES) screen, the Load Priority List Management System (LPLMS) screen, the Fault Recovery and Management Expert System (FRAMES) screen, and the Power Utilization screen. The user interface was designed such that the Power System screen would be the one that the user would spend the most time interacting with. Therefore, it was decided that some information regarding the other screens would be available from the Power System screen. In the event that the user wants to obtain more details about this additional information, they can switch to the appropriate screen.

The issue of user interface consistency was primarily addressed by keeping certain windows the same for all the screens, and only changing those that directly pertain to the changing screens. The windows that remain the same for all the screens are: the title window, which serves to display the MSFC logo; the clock bar, which displays the clock time and mission time; the screen selections menu, which allows the user to select the screen they wish to view; the mode indicator, which informs the user of whether the system is in autonomous, manual, or semi-autonomous modes; the main menu bar, which allows the user to perform operations which aren't critical in operating the system, or are only used occasionally; the system flow window, which illustrates the activities of all of the software modules of SSM/PMAD, as well as the communications of the modules; and the messages and diagnosis window, which serves to display the system messages and diagnosis information. The only windows which change between screens are: the main window, which displays the primary data of the screen; the scratchpad window, which displays the secondary data related to a screen; and the main window menu, which allows the user to choose options, some of which will allow the user to view data in the primary or secondary windows. The SSM/PMAD user interface screen layout is depicted in Figure 1.

The issue of ease of user interface operation is addressed by making the interface mouse driven and making operations accessible through menus. This was a method of designing the interface so that the user could learn to operate the system quickly, and increase the likelihood of the operator in retaining this information.

The issue of keeping the user informed as to the states and faults of the system is addressed by giving the user as much feedback as possible. The system messages are displayed in the messages and diagnosis window. The messages that appear with respect to faults are coordinated with the representations of switches on the schematic, as shown in Figure 2.

Screen Selections Menu	Clock Bar		Marshall Space Flight Center	
	Utilities Menu	Functions Menu	Help Menu	Mode
System Flow Window	Main Window Menu	Main Window		
Messages and Diagnosis Window				

Figure 1: SSM/PMAD Screen Layout

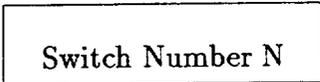
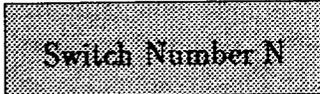
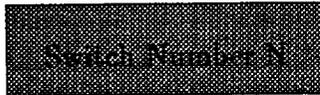
Message	Switch Representation	Status
No Related Message		OK
Fault reported on Switch Number N		Fault in progress
Switch Number N taken out of service		Out of Service

Figure 2: Correlations Between Switches and Messages

4.1 Power System Screen

The function of the Power System screen is to allow the user to view a functional representation of the breadboard and interact with it. The Power System main window is where a schematic drawing representing the breadboard is displayed. The user can select switches in the main window by clicking them with the mouse, and then choosing the "Command Switch On / Off" option in the main window menu. The user can also select the options "Schedule for Selected Item" and "Schedule for a Switch and Switches Below It" (in the switch hierarchy) so that a bar chart of all selected switches appears in the secondary window, indicating when they are scheduled to be on and off. The user can also choose the option "Deselect All" at any time to deselect all of the switches in the main window that have been selected.

The remaining options in the main window menu allow the user to view secondary data in the scratchpad window. If the user selects switches in the main window and chooses the "Detailed Data" or the "Detailed Data for a Switch and Switches Below It" option, textual data for each switch, and a bar chart will be displayed in the secondary window. Refer to Figure 3 for an illustration of this concept. Choosing the "Load Information" or "Load Information for a Switch and Switches Below It" options will display textual data in the secondary window that indicates which activity is tied to a switch at the time load information is selected. The "Power Utilization for Select Items" option will display a plot of actual and projected power for a two hour time frame.

The Power System main window displays a schematic level representation of the breadboard. Colors are used to indicate the states of the switches within the schematic – green is used to indicate a normal state, red is used to indicate a faulted state, and brown indicates a switch that is out of service. If a switch is in a normal state, a numerical value representing the amperage flowing through it is displayed. If it is faulted, then the reason for the fault is displayed instead of the amperage. The actual switches are represented by relays, which are closed if they have power flowing through them, and open if not. The bus cables that connect the switches are colored if they have power flowing through them, and are hollow if not.

4.2 FELES Screen

The purpose of the FELES screen is to display different aspects of the system's schedule. It makes the user aware of the activities and updates to activities that have been scheduled. The FELES main window, by default, displays a chart of the entire system's schedule as per activities. The user can choose the option "Change Window of Time" in the main window menu at any time to change the time frame for the information in the main window that is displayed. Choosing the "View Activity Schedule" option shows the default main window. The user can choose an activity(s) and select the "Selected Activity Switch Schedule" option to display the switches that are scheduled for the selected activity(s). The "View Load

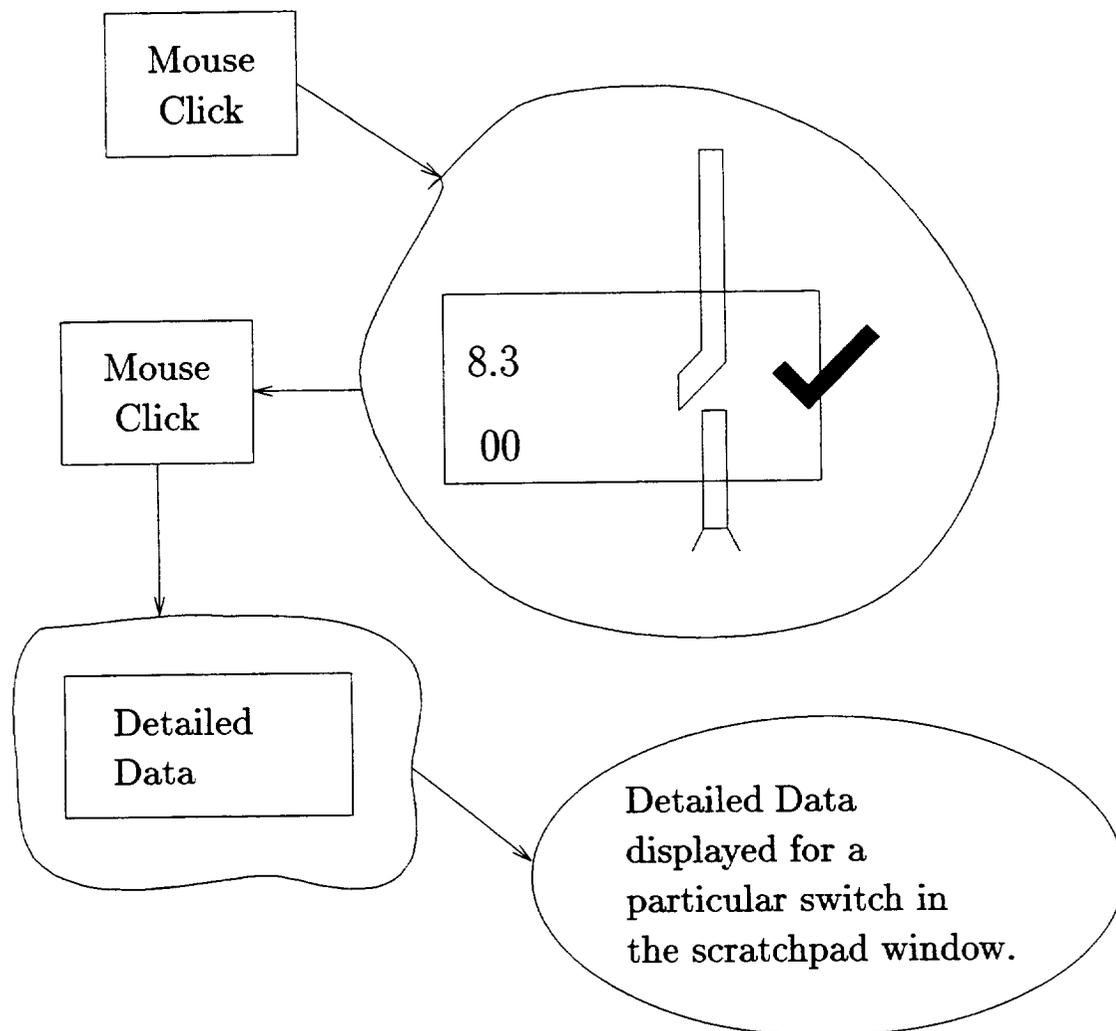


Figure 3: Process to Obtain Detailed Data

Center Schedule” prompts the user to indicate the load center for which they wish to view scheduling information, and then displays the schedule for that load center in the main window. ”Selected Activity Description” requires a user to select an activity(s) so that it can display textual information in the secondary window about the activity or activities selected. Figure 4 illustrates this concept. ”Selected Load Information” displays information in the secondary window depending upon whether activities or RPCs are being displayed in the main window. If an activity is selected, a list of all RPCs in use and their attached loads are displayed. If an RPC is selected, a list of all activities using the RPC and the time frame for which each activity uses the RPC, as well as a list of load equipment attached to the RPC is displayed.

4.3 LPLMS Screen

The purpose of the LPLMS screen is to allow the user to view information relating to the load shedding priorities. The main window displays the rules that are used in computing the priorities for load shedding. The user’s understanding of priorities is important because often times, one would like to know how priorities are computed. One of two options for viewing priorities can be chosen in the main window menu. One of the options allows the user to view the current switch priorities, with respect to load shedding, in the scratchpad window. The other option allows the user to view the weightings used to compute the priorities.

4.4 Power Utilization Screen

The purpose of the Power Utilization screen is to provide the user with information about the power usage of the system. It allows the user to view power consumption graphs for the system.

The Power Utilization main window defaults to system level power usage graphs. This window is composed of a graph for each bus of a load center. This allows for the simultaneous viewing of 12 different graphs which include five sets of load center graphs and one overall system graph for each bus. Each graph contains the actual power used displayed on top of the power scheduled.

Choosing the option in the Power Utilization main window menu, ”Change Window of Time”, allows the user to change the viewport for all plots being displayed to a new time frame. This function will change the start time and end time plotted on the y- axis and scale the data to fit the new axis constraints. ”System Power Usage” draws the system power usage graphs described above. This is the default screen.

”Load Center Power Usage” prompts the user to indicate which load center should be graphed. It then graphs the power used along with the power scheduled, but the data plotted is for individual switches in the load center. The graphs are arranged for viewing in the same

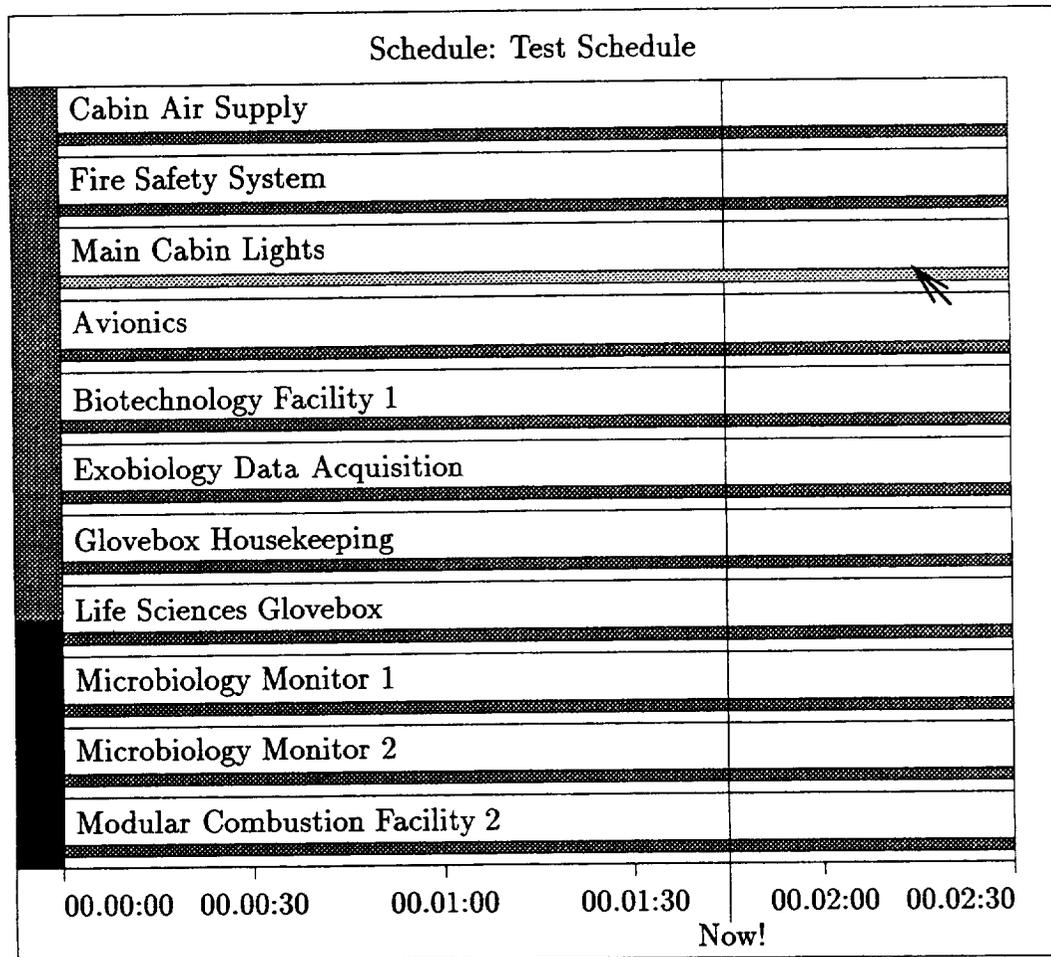


Figure 4: Selected Activity Description

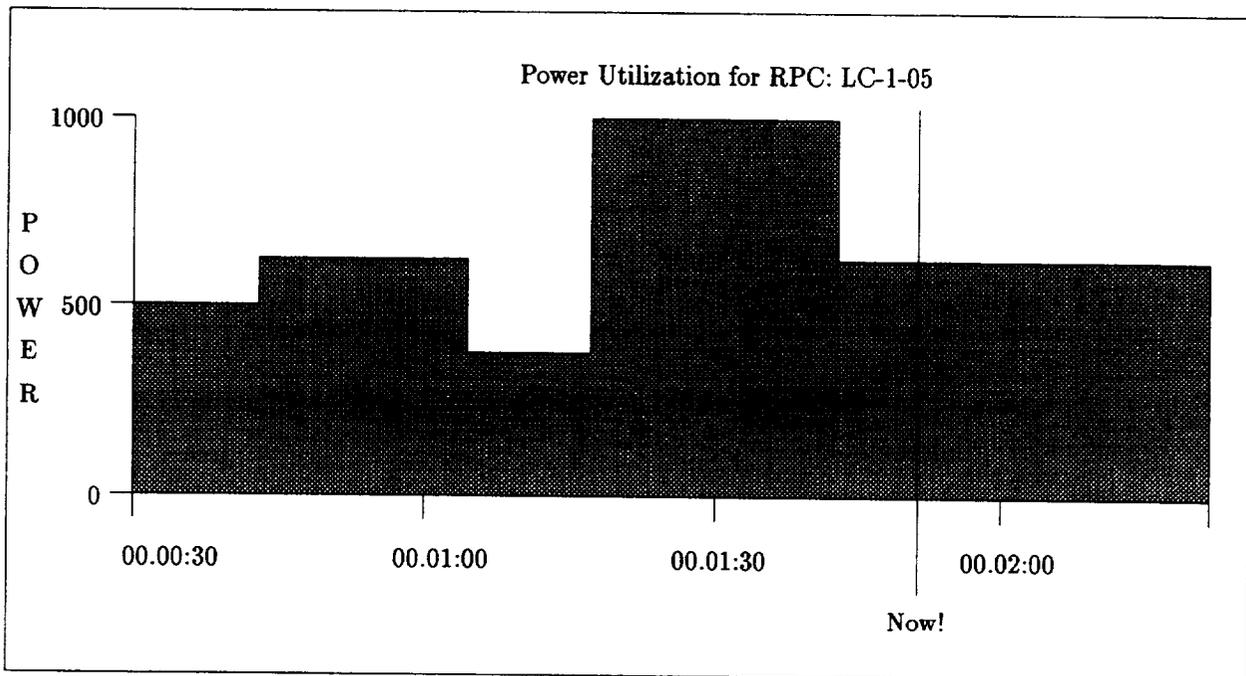


Figure 5: Power Usage Plot

manner as "System Power Usage" with the switches separated by bus. The bottom graphs in the window contain the bus totals for the load center, as illustrated in Figure 5.

"Component Power Usage" prompts the user for an individual switch and plots the switch's power usage to the secondary window.

Selecting "System Availability" allows the user to view the amount of power available for scheduling on load centers and in the overall system.

"Load Center Availability" shows the user the power availability picture on a load center basis. This picture is similar to "Load Center Power Usage" in format except that available power is plotted against time.

5 USER INTERFACE EXPANSION

There are software modules to be incorporated into the SSM/PMAD system in the near future. These include intermediate levels of autonomy, planning, an activity editor, and a rule viewing capability. These modules can be easily added into the SSM/PMAD architecture

because of its modular design. The user interface will be required to allow the user to access this new functionality as well as view additional data, and can support these changes through its own modular, object-oriented design.

6 SUMMARY

A well designed user interface insures that a user can easily obtain all useful and critical data for operating the system and is aware of all faults and states in the system. The SSM/PMAD user interface provides access to the functionality of the power system as well as exemplifying current techniques in user interface design. It was designed to enable an engineer to operate the system easily without having significant knowledge of computer systems, as well as provide an environment in which the engineer can monitor and interact with the SSM/PMAD system hardware. The design of the interface supports a global view of the most important data from the various hardware and software components, as well as enabling a user to obtain additional or more detailed data when needed.

Three main concepts were followed in the design of the user interface: consistency, ease of operation, and alerting the user of all critical information.

The SSM/PMAD user interface supports four screens: the Power System screen, the FELES screen, the LPLMS screen, and the Power Utilization screen. The Power System screen serves to allow the user to view a functional representation of the breadboard and interact with it. The FELES screen displays different aspects of the system's schedule. It depicts activities and updates to activities that have been scheduled. The LPLMS screen allows the user to view information relating to the load shedding priorities. The Power Utilization screen provides the user with information about the power usage of the system. It allows the user to view graphs that illustrate the power that has been consumed by the system.

The SSM/PMAD user interface will be expanded in the future to encompass intermediate levels of autonomy, planning, an activity editor and a rule viewing capability. The SSM/PMAD's modular architecture allows these additions to be easily incorporated into the existing design.

7 ACKNOWLEDGEMENT

This work was performed by Martin Marietta Astronautics Group under contract number NAS8-36433 to the NASA George C. Marshall Space Flight Center, Huntsville, Alabama.

8 REFERENCES

- {1} Judith R. Brown and Steve Cunningham, *Programming the User Interface: Principles and Examples*, John Wiley & Sons, New York, NY, 1989.
- {2} Richard M. Cohen, Timothy P. McCandless and Elaine Rich, "A Problem Solving Approach to Human- Computer Interface Management". MCC Technical Report ACT-HI-306-89. August 89.
- {3} A. Lemke. "Design Environments for High- Functionality Computer Systems", Ph.D. diss., Department of Computer Science, The University of Colorado at Boulder, 1989.
- {4} W. Miller, et al, "Space Station Automation of Common Module Power Management and Distribution", NASA Contractor Report 4260, Martin Marietta Aerospace, Denver Astronautics Group, 1989.
- {5} NASA Space Station Freedom Program: Human- Computer Interface Guide, Version 2.1, Johnson Space Center, Houston, TX, December 1988.
- {6} J. Riedesel, C. Myers, B. Ashworth, "Intelligent Space Power Automation", In Proceedings of the IEEE International Symposium on Intelligent Control, 1989.
- {7} Sid L. Smith and Jane N. Mosier, "Guidelines for Designing User Interface Software", Report ESD-TR- 86-278, The MITRE Corporation, Bedford, MA 01730, Electronic Systems Division, August 1986.

**PRIORITY SCHEME PLANNING FOR THE ROBUST SSM/PMAD
TESTBED**

Michael R. Elges

Barry R. Ashworth

Martin Marietta Astronautics Group

MS-S0550

P.O. Box 179

Denver, CO 80201

elges@daneel.den.mmc.com

bashworth@den.mmc.com or bashworth on NASAMail

ABSTRACT Whenever mixing priorities of manually controlled resources with those of autonomously controlled resources, the Space Station Module Power Management and Distribution (SSM/PMAD) environment requires cooperating expert system interaction between the planning function and the priority manager. These cooperating expert systems should minimize any adverse side effects from introducing new uniform priority blocks into the overall operation of the autonomous system. Martin Marietta, under contract to NASA George C. Marshall Space Flight Center since 1985, is continuing the development of technologies and methodologies for use in the SSM/PMAD automation.

Enabling manual seizure of system resources that are currently under automated control or introducing new power demands while requiring autonomous stability are accomplished under the auspices of an expert system planning function. The Planner's task is to implement, with all possible stability, the engineer's new resource requirement. In doing so, the load priority management system must cooperate with the planner. The interaction between these two knowledge base systems encompasses many items which must be addressed by the developer. These items are 1) General protocol between the planner and the load priority manager. 2) Protocol for conflict resolution between the planner and the load priority manager. 3) Weighting factor paradigm based on the planners temporal understanding of power consuming resources. 4) Interactive solution capabilities so the planner can propose several options and the load priority manager adds its weighting factors and system impacts (this is a useful option for conflict resolution) and 5) Incorporation of the planner's new power requirement (with subsequent notification to the scheduler) into the a new uniform priority block requirement. The elements and interactions of the SSM/PMAD planning and priority management functions are presented in this paper. Their adherence to cooperating for common achievement will be described.

1 Nomenclature

AI Artificial Intelligence

FELES Front End Load Enable Scheduler

FRAMES Fault Recovery and Management Expert System

GC Generic Controller

K, k Kilo (1000)

KANT Knowledge Augmentation and Negotiation Tool

KW, kw Kilowatt (1000 watts)

LC Load Center

- LLF** Lowest Level Function
- LLP** Lowest Level Processor
- LPLMS** Load Priority List Management System
- MAESTRO** Master in the Art of Expert Scheduling Through Resource Orchestration
- MSFC** (George C.) Marshall Space Flight Center
- NASA** National Aeronautics and Space Administration
- PDCU** Power Distribution Control Unit
- RBI** Remote Bus Isolator
- RPC** Remote Power Controller
- SIC** Switch Interface Controller
- SSM/PMAD** Space Station Module (Automated) Power Management and Distribution (System)
- UPB** Uniform Priority Block

2 Introduction of Planning into the SSM/PMAD Testbed

In the course of SSM/PMAD Testbed autonomous control, new user, power, or resource requirements may be introduced into the autonomous portion of the operational system for unspecified reasons. To satisfy any new power or control demands, careful planning must be undertaken to ensure minimal system disturbance and provide process integrity. We must ensure that the constraints which defined the present schedule of activities are not violated. At the same time, we must successfully manage the adding and merging of new constraints from any new user, power, or resource requirements in order to define a new schedule. The conditions that may be encountered that could cause new user, power, and resource requirements are many and varied; but, fundamentally they occur as one of two types. The first type is a failure at some level in the complete system, be it at a 1kW switch, a 3kW switch, a load, a power source, or any unplanned event. This failure could cause some activity to be removed from the presently executing schedule. The activity may not be rescheduled in a contingency because the recovery and scheduling system may not anticipate all of the types of failures and their consequences. The second type is a new requirement that was unknown prior to the production of the presently executing schedule. An example of this occurs when a user needs to access partial level switch control of the autonomously executing testbed.

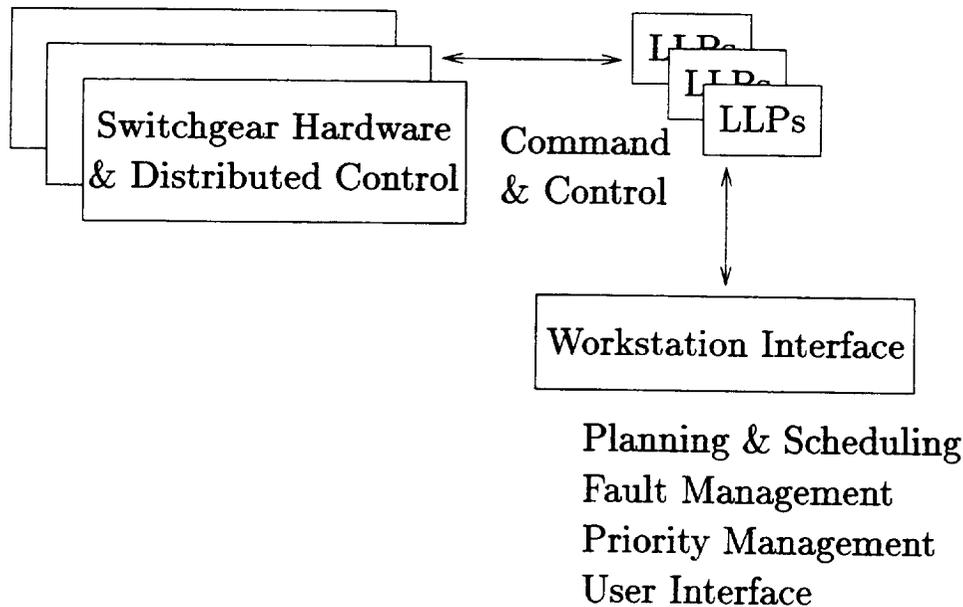


Figure 1: SSM/PMAD Structure

This type is more probable to encounter because we can never fully anticipate all future requirements in very complex environments which are composed of incompletely defined systems.

The architectural structure of the SSM/PMAD Testbed yields distributed hardware control, and yet a single user interface. Each elemental entity of the distributed portion of the architecture is executing an independent segment of an overall schedule. Each also possesses its own set of priorities. Both the priorities and the overall schedule of activities are produced and maintained at single locations so that they may take advantage of global knowledge. The view of the system as shown in Figure 1 provides visualization of how this is accomplished. The planning and scheduling activities exist at a central communications point with respect to the distributed lower level control functions, LLFs which reside on the LLPs. Therefore the workstation provides a global point for knowledge and communications, and in turn functions as the residence for the planning, scheduling, fault management, and priority management functions. Since the ultimate goal of the system is to automatically manage power distribution, even during the occurrence of faults, these functions are all capable of executing autonomously. However, complete and safe systems must allow a user to interrupt an autonomous flow of events. The SSM/PMAD Testbed, therefore, provides levels of autonomous activity {Ash, '90}.

When a user demands that elements of the system be turned over for manual control, several actions must take place to ensure operational integrity for those elements remaining under autonomous control. In the SSM/PMAD Testbed these actions are guided by having a system planning function, KANT, which has insight to the executing system and its automated database. First, the user must be given access to all information which may have an effect on the desired outcome. KANT transparently gives the user access to the global and local databases. KANT also contains a knowledge based element which can relate user goals to the constraints and resources available. Second, the fault manager element, FRAMES, must be informed as to the change so that correct diagnoses and operations take place if and when faults occur. Third, some element must engage as mediator for selection of resources and actions to be added or removed at the user's request. This is performed by the priority manager, LPLMS. Lastly, the scheduling mechanism, MAESTRO, must provide future schedules adhering to the user modified resource base. All of this is glued together by a common knowledge base and database management system, KNOMAD, which understands the knowledge primitives of the complete system. Only when all of these functions interact on behalf of the user may the system safely be partitioned to partly autonomous and partly manual, or simply, semi-autonomous.

The KANT planning function allows the user to experiment with possible accesses to the testbed activities and resources without committing the human to absolute unaided control of testbed processes and hardware. This is highly desirable since the testbed is capable of concurrently running 10 to 20 critical loads, some of them in a redundant power mode. When deciding on ways to guide the human user in accessing and controlling the testbed and its resources the planner must use the most convenient and accurate discriminators in minimizing system impacts. For this, KANT uses the priority values produced by the LPLMS. Figure 2 depicts how the priorities are considered and if need be, changed. The execution of LPLMS should be considered as a two step procedure. First, a set of weighted values are established for each activity over the relevant schedule period. The weighted values examine several activity features, such as needed crew and crew availability, available and required load power levels, conflicts between loads, available opportunities for load activities, and the activity intrinsic: interruptability, restartability, and skipability. Once the needed weights have been calculated, the second step of combining the weights mathematically for specified time periods produces the required priority values. The need to recalculate priority values may be triggered by events within the nominal execution profile or by anomalous occurrences, such as faults. In any case, KANT uses information of the needed and available resources; the activities and their priorities; the known and proposed schedules; any user-directed and autonomously-driven requests; and, historical and presently active testbed situations in formulating its list of proposed events to accomplish user and system goals.

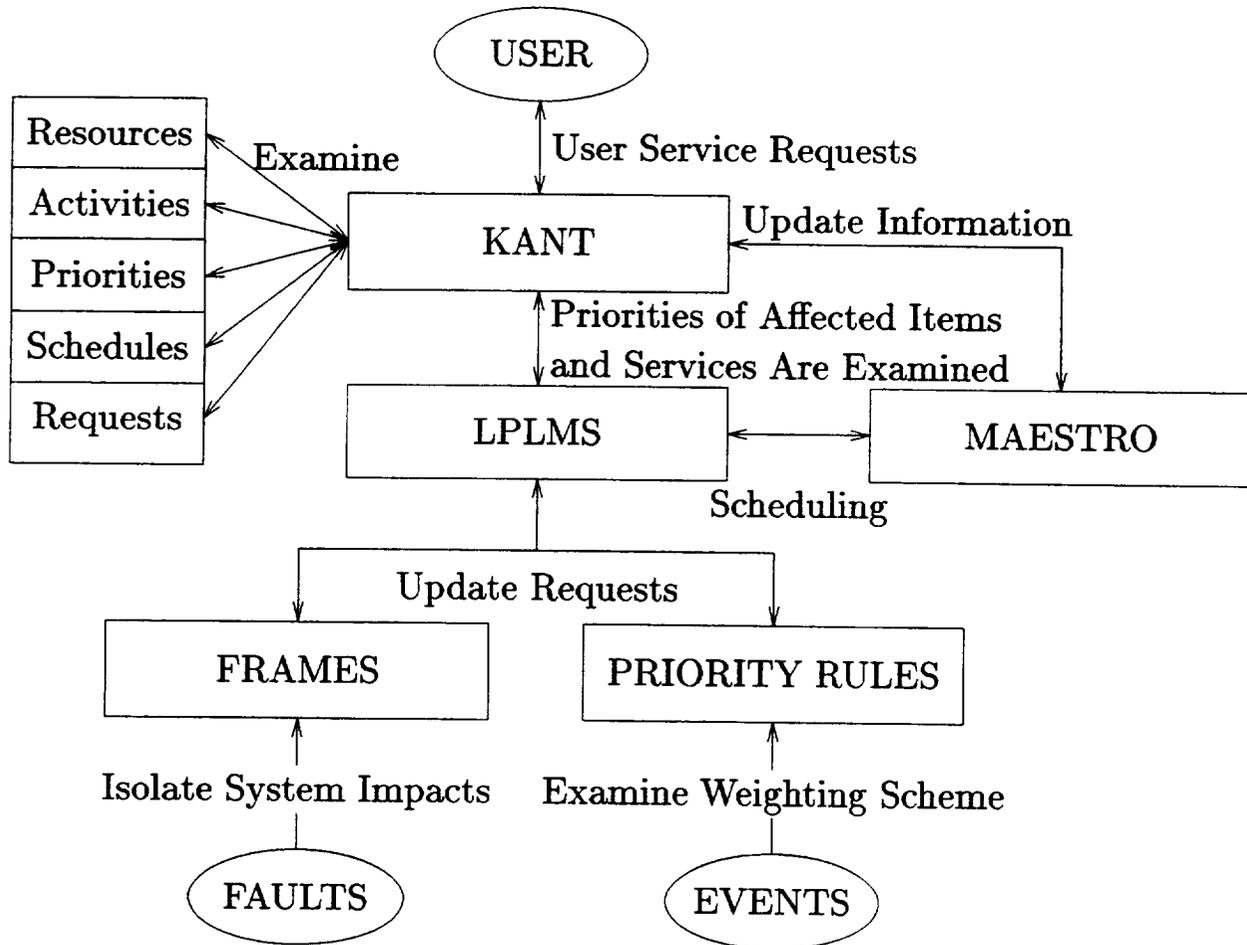


Figure 2: Priority Examination and Update Process

3 The Functionality of the KANT Planner within the Testbed

The most important function within the KANT suite of jobs is to aid the user in obtaining portions of the autonomously controlled testbed for manual control use. However, KANT performs other critical functions as well. KANT directs schedule flow and production before requests are forwarded to the MAESTRO scheduling system. Resource usage within the nominal system is also tracked, and system availability knowledge is maintained for later use under any less-than-nominal situations.

A description of how the planner works within the testbed is imperative for understanding what the planner's functions accomplish. The SSM/PMAD Testbed uses the idea of a unified schedule, consisting of many activities over a specified time period. Each activity details a simple low-level plan on what switches to turn on when and where; and, what power and non-power resources are needed to accomplish the activity's task. Yet, this is not a complete plan because the activity has no provision for being certain that each state is correct before proceeding with the power actions for that state. The system, without the planner, only responds to switch event activities at certain times; and, any related knowledge to automated processes or users and operators would be limited. Therefore, in performing its tasks, KANT maintains knowledge of activities, resources, and events for application in achieving user and autonomous system goals. This is consistent with the concept of planning and organization {Geo, '87} and a system providing such.

As a result of a dominating conflict potential and the resulting user frustration when trying to access only minimal resources within an autonomously executing environment, KANT also possesses a unique negotiation feature. The primary purpose of the negotiator is to aid the user. This is because the user always possesses the capability to control the SSM/PMAD testbed under the complete manual control option {Jak, '91}. However, partial manual control is desirable. This way, a user can concentrate only on the part of the system in which there is an interest, rather than managing the control of the entire testbed.

4 User Driven Effects with KANT Functioning as Negotiator

Utilizing the functionality within KANT, the testbed operator can troubleshoot system components, plan for optimal transitions between autonomous and manual modes, place the testbed into semi-autonomous mode, or achieve emergency control over anomalous system elements. Figure 3 shows how the user's influence of control to autonomous system management flows into control of the switchgear hardware.

KANT provides the user with a governing capability on an almost limitless (with respect

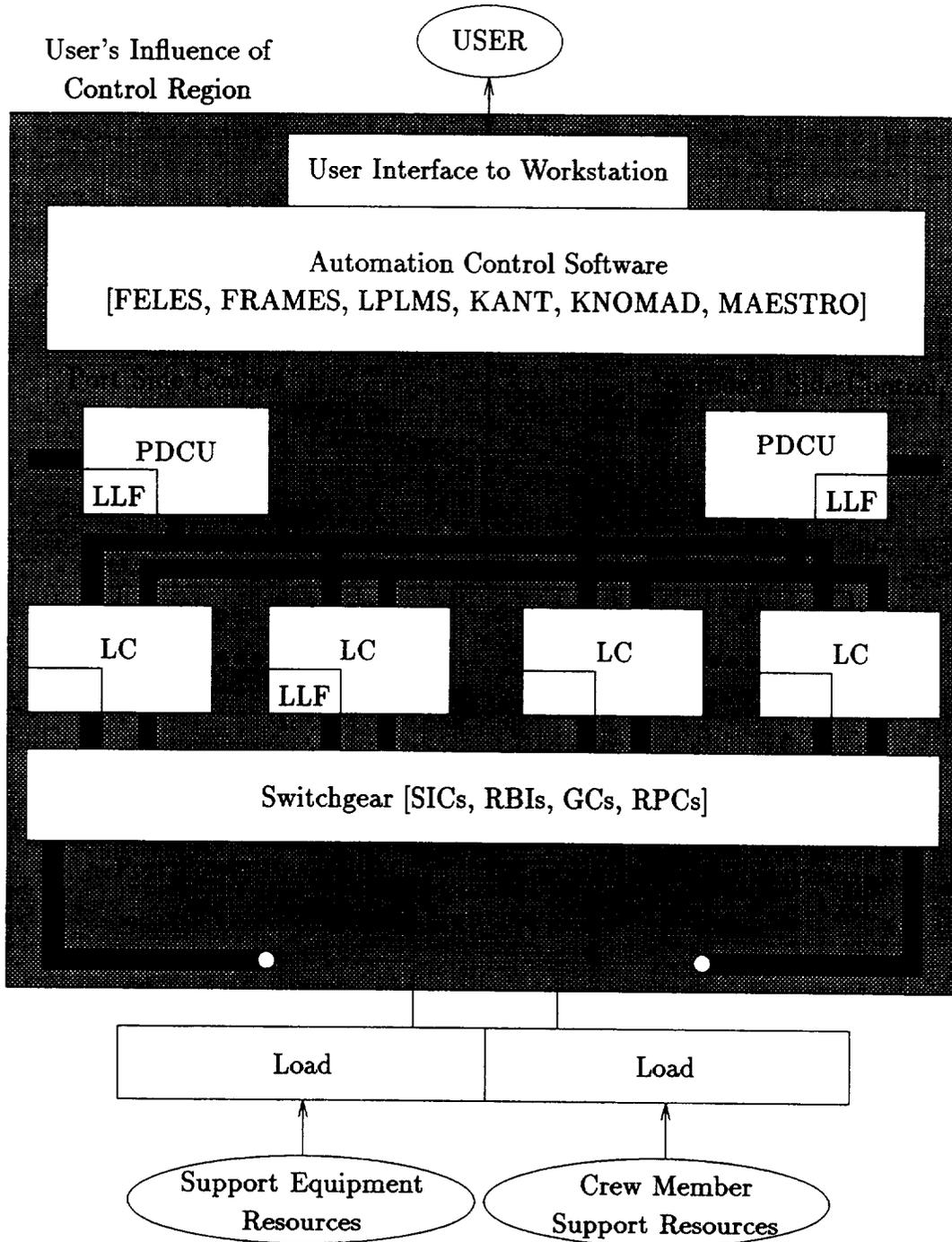


Figure 3: SSM/PMAD Resources and Control

to the testbed) Influence of Control region. This is critical to the user achieving partial control within an otherwise completely autonomous process. It can be seen from the figure that control must range over the lower level process environments, such as the PDCUs and the LCs, as well as the smart switching elements of the switchgear. This would require the user to possess perfect state knowledge at the time when manual control was instated, a task with a very low probability of success. Therefore, without the services of an automated planning function, a shutdown of testbed autonomous activity followed by acquisition of full manual control would seem the most likely approach for any level of manual control. This approach would be very intractable and disruptive, however, and is not acceptable.

The successful management of the partially autonomous testbed is achieved through dialogue between the user and the knowledge planning agent. The purpose of the dialogue is to isolate the user's request(s) to the elements of the testbed which cause the least disruption while still fulfilling the user's requirements. In order to accomplish this the user must identify or provide aid to the planning function in identifying the goals and resources needed. KANT will in turn satisfy (with the user's assistance) system constraints as needed. Figure 4 illustrates the flow of activities needed to achieve this process.

The most important of the plan generation constraints are the priorities of the activities. The user may enter a priority for any manual activity to be introduced. KANT will isolate the entered priority within the overall priority list. Once the priority is isolated, KANT forms a new UPB which serves as the overall priority list for power reduction and load shedding by the LLFs. This is very important as it makes the fault handling function at FRAMES uniform, whether the system is completely autonomous or semi-autonomous.

During plan task generation KANT must ensure that all necessary resources to achieve all goals are in place both for manual activities and for scheduled autonomous activities. Constraints on resources may generate conflicts with scheduled operations beyond the scope of immediate impacts. Therefore, KANT must identify these impacts to a user and may in some cases cause autonomous activities to be rescheduled. For those cases, KANT communicates out-of-service messages on needed resources to the MAESTRO Scheduler and marks the descheduled activity for rescheduling. However, rather than deschedule an activity, KANT will attempt to negotiate alternative goals with the user. If any alternative(s) are acceptable to the user the autonomous activity will continue with little impact and the MAESTRO Scheduler will be notified only of resources which have been removed from service.

Once again, the primary consideration is the priority of the activity either being entered into or removed from operation. Other constraints are, however, important. Some of these are needed power, available power, number of power connections required, and number of autonomous activities affected, among others. KANT primarily performs "Priority Scheme Planning" since the load shed decisions at the LLPs are formed fundamentally on the priorities. Also of considerable importance is the user's view of system-wide testbed activity. Viewing the system from the priority concept provides the user with a homogeneous approach to activities and changes within the testbed.

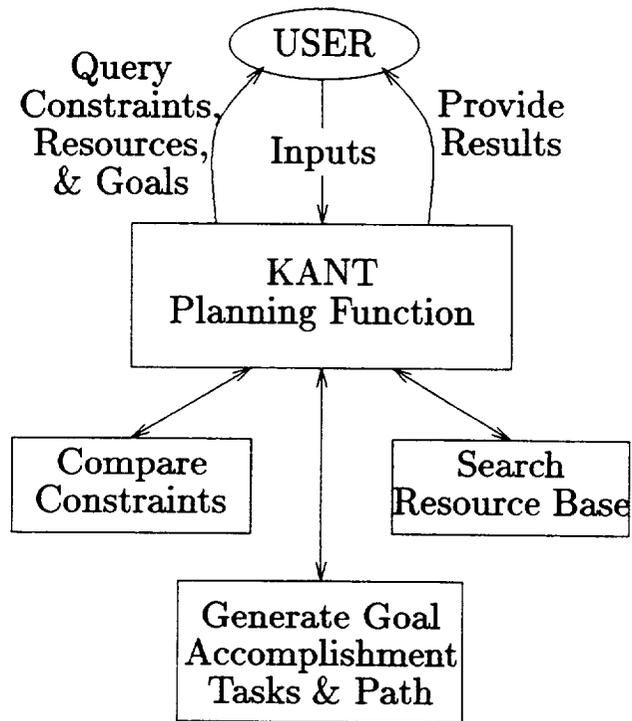


Figure 4: Planning Task Generation

5 KANT Directions to the User

KANT provides the user an interactive dialogue during the constraint isolation and plan construction phase. Based upon the user's direction and the negotiated results of plan formation, KANT forms a user activity list to achieve the plan goals. One of KANT's internal operating constraints is the minimization of human activity to achieve a plan's goal(s). Therefore, the user activity list may be as little as:

1. Connect Load to Switch B,
2. Control Switch B from Manual Interface.

Or a complex list of user activities could be generated depending on the user's request and direction in achieving the overall goal. In any case, required user actions are listed in a step-by-step procedure designed to achieve the overall goal.

A very important concept within the KANT/User interaction scheme is that of constraint identification. The user may directly input known constraints, or the user may request aid in constraint formation. Figure 5 illustrates the constraint identification flow. Note that the user activity list described above is generated once the constraints have been sufficiently identified and validated as to resource utilization and priority scheme continuity. If the user and KANT cannot negotiate a settlement on the constraints and the final plan, the user may have final authority on changing the priority scheme or changing resource usage associated with scheduled tasks. The authority is granted by KANT upon the user entering a known password of authority. When the user has taken such action, priority activities at least as high as the user's are removed to allow the user's new activity on the testbed, and the plan activity list is then formed for the user.

6 Further Applications of KANT within the Entire SSM/PMAD Testbed

The services of a planning function are very useful within many domains. The most important candidate for furthering the activities of KANT is the FRAMES function. The dynamic quality of fault production within environments that may massively reconfigure is not well understood as an empirical process. A priori knowledge provided by the planner to the fault manager on possible cause and effects may be of important value in isolating and recovering from faults.

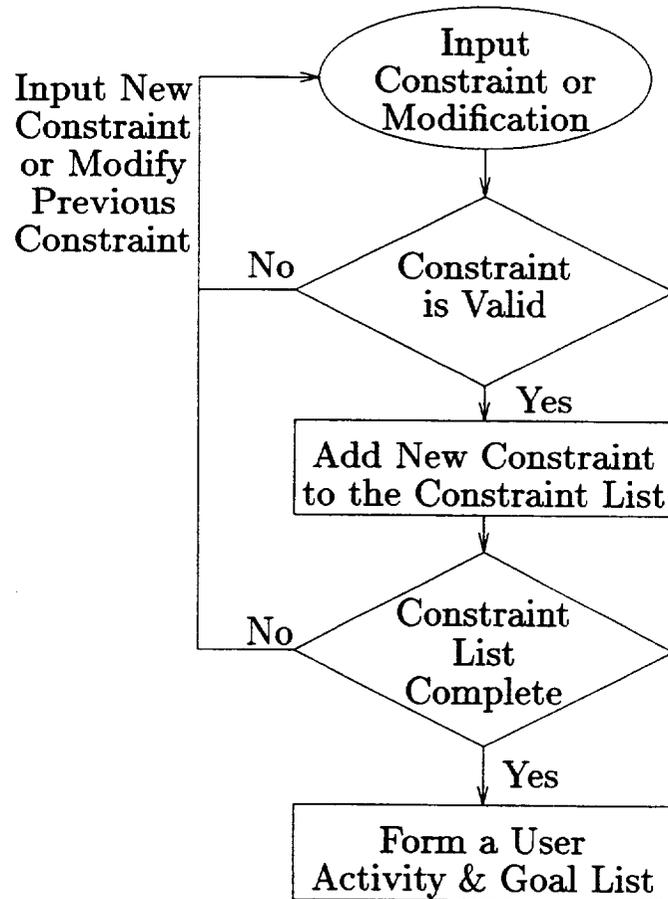


Figure 5: Constraint ID Process

7 Acknowledgement

This work was performed by Martin Marietta Astronautics Group under contract number NAS8- 36433 to the NASA George C. Marshall Space Flight Center, Huntsville, Alabama.

8 References

- {All, '90} J. Allen, J. Hendler, and A. Tate, "Readings in PLANNING", Morgan Kaufmann Publishers, Inc., 1990.
- {Ash, '90} B. Ashworth, "Managing Autonomy Levels in the SSM/PMAD Testbed", In Proceedings of the 25th IECEC, 1990.
- {Geo, '87} M. Georgeff, "Planning", In Annual Reviews of Computing Science, 1987.
- {Jak, '91} L. Jakstas and C. Myers, "User Interface Design Principles for the SSM/PMAD Automated Power System", In Proceedings of the 26th IECEC, 1991.
- {Mil, '89} W. Miller, et al, "Space Station Automation of Common Module Power Management and Distribution", NASA Contractor Report 4260, Martin Marietta Aerospace, Denver Astronautics Group, 1989.
- {NAS, '88} NASA, "Space Station Advanced Automation Study Final Report", Strategic Plans and Programs Division, Office of Space Station, NASA Headquarters, May 1988.

**INTEGRATING THE AUTONOMOUS SUBSYSTEMS MANAGEMENT
PROCESS**

Barry R. Ashworth
Martin Marietta Astronautics Group
MS-S0550
P.O. Box 179
Denver, CO 80201
bashworth@den.mmc.com or bashworth on NASAMail

ABSTRACT

The Space Station Module Power Management and Distribution (SSM/PMAD) testbed system uses cooperating expert systems to autonomously manage the power distribution process. There are several cooperating expert systems continually interacting to provide this management. Whenever other subsystems activities are considered, they also must provide either portions of or complete autonomous management elements which can interact with other subsystems. This poses two problems. First, how do the subsystems agree on an acceptable level of performance; and, second, how do the autonomous management elements determine relative levels of importance amongst each other. Martin Marietta, through its independent research, and under contract to NASA George C. Marshall Space Flight Center since 1985, is continuing the development of these technologies and methodologies for use in the SSM/PMAD and other platforms for automation.

Organizing the subsystems and their associated elements into a cohesive and consistent picture requires new approaches. These new approaches include visualizing the ranking of each subsystem's activities and further ranking the complete system level activity and goals. Management of the goals at each subsystem may be done using the priority tensor based approach used in the SSM/PMAD testbed. Ranking the complete system activity requires an integration activity that is capable of dynamically ascertaining and managing a subsystem's relative fundamental relation to the other subsystems; thus, forming a complete rank ordered listing of all the subsystems within the system. How the complete system ranking may be achieved and how an individual subsystem's internal priorities may be managed within the complete system are examined in this paper. Further, how these results may be used in the integration and performance leveling of the autonomously managed system are also presented.

1 NOMENCLATURE

AI Artificial Intelligence

FELES Front End Load Enable Scheduler

FRAMES Fault Recovery and Management Expert System

GC Generic Controller

KANT Knowledge Augmentation and Negotiation Tool

KNOMAD Knowledge Management and Design System

LC Load Center

LLF Lowest Level Function

LLP Lowest Level Processor

LPLMS Load Priority List Management System

MAESTRO Master in the Art of Expert Scheduling Through Resource Orchestration

MSFC (George C.) Marshall Space Flight Center

NASA National Aeronautics and Space Administration

PDCU Power Distribution Control Unit

RBI Remote Bus Isolator

RPC Remote Power Controller

SIC Switch Interface Controller

SSM/PMAD Space Station Module (Automated) Power Management and Distribution (System)

2 INTRODUCTION TO INTEGRATED SUBSYSTEMS MANAGEMENT

In almost all applications, systems are composed of a host of subsystems; each providing a task sequence that appears to be uniquely within its domain. However, this is illusory. And, when an overall system architecture appears to be understood, requirements within subsystems may produce subtle effects that encroach on the already established overall system interfaces. Figure 1 depicts this concept of a system and its subsystem interfaces. The figure also introduces the Common System Control element.

The common system control functions are generally an array of activities carried on by independent management functions causing mutual agreement between the various subsystems. However, when the subsystems become increasingly complex, leading to many and difficult decisions, agreement may not be an easily achievable goal. An readily ostensible example of this is the execution of natural systems (by this I refer to non-machine systems; e.g., human political systems). Engineered intelligent machine systems can also display this tendency towards decision diversity. The SSM/PMAD system at the NASA Marshall Space Flight Center is an engineered intelligent machine system.

In these intelligent machine systems the key component upon which to focus when trying to achieve autonomous operation is the common control function. Between subsystems directly there are usually requests for data, keeping each subsystem's control strategy hidden and private. This makes good sense. Otherwise, every subsystem would need to contain all

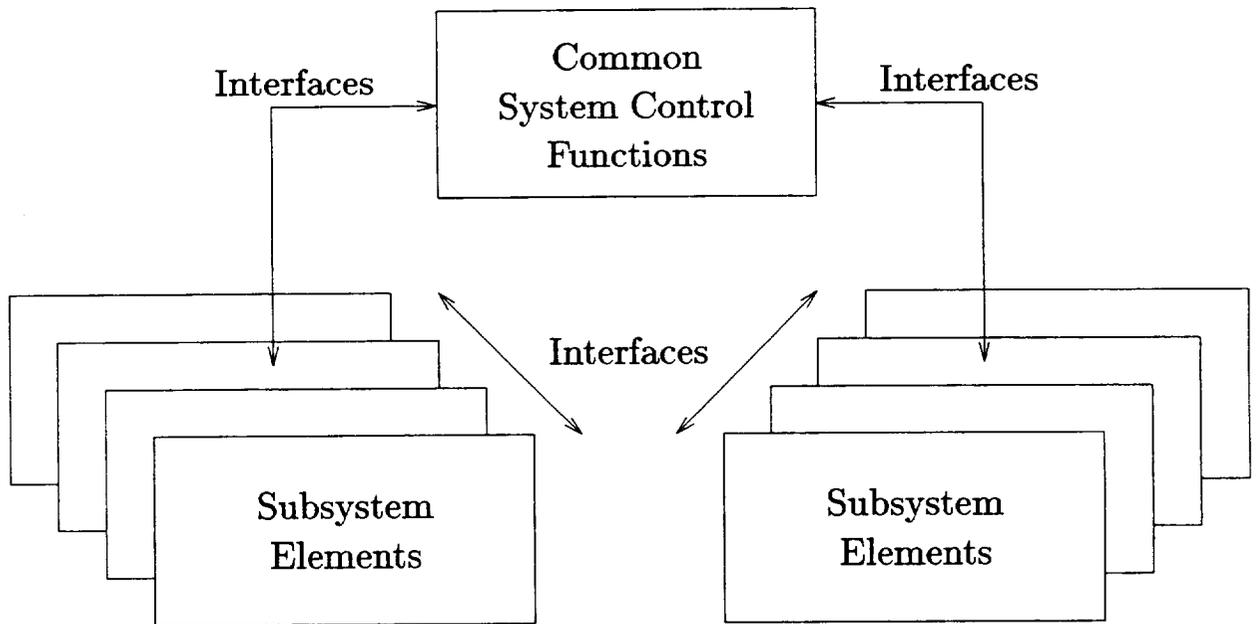


Figure 1: System Components and General Architecture

the knowledge and information for every other subsystem, causing wasteful duplication! Further complicating a distributed subsystem control approach would be the process to supply a strategy at system execution time that would provide appropriate decision making within a correct subsystem - an intractable task. Therefore, a common control function is mandatory.

The primary task of the common control function should be to preserve system-wide integrity and allocate scarce resources in a manner which preserves the highest priorities of the overall system. It should be noted that the privacy of subsystem internal activities suggests only that other subsystems are ignorant of a particular subsystem's internal activities and needs. Due to this restriction, the common control function should be considered a management and administration function, not another subsystem or an extension of any particular subsystem.

Integrated Subsystems Management is then achieved when the common control function provides this cohesive management. The cohesiveness comes from the subsystems all working to achieve a common goal, even though they may not be aware of a common goal. An example should help clarify this concept. Suppose there is a system that consists of:

1. A Power Subsystem,
2. A Data Processing Subsystem,
3. A Thermal Management Subsystem,
4. An external unpredictable environment.

Let us further suppose that this system is asynchronous (synchronous systems present unique situations which we will touch on later) with no data interdependencies, which is reasonable. The situation is the power subsystem needs to supply available power for thermal management and data processing. The data processing subsystem needs to update subsystem state for itself, the power subsystem, and the thermal management subsystem. For some unknown reason, the power source is no longer delivering an adequate amount of power. The immediate response is to save the data processing subsystem so that it does not lose its present state. Then, the data processor should reduce its activity on an intermittent basis so that the thermal management subsystem can perform its diminished tasks. Now, each subsystem's workload has been diminished (but note, each is still working) but is still trying to achieve the system goal. Any individual subsystem would probably not be able to achieve this on its own. Therefore, the integration of the subsystems using the common control function would be an appropriate approach.

3 THE AUTONOMOUS FUNCTIONALITY OF THE INTEGRATED SSM/PMAD TESTBED

The SSM/PMAD Testbed provides a common control function approach to autonomous management of a Space Station Freedom type power management and distribution system. Its elements and architecture can be seen in Figure 2. The system executes to a schedule of events completely autonomous from human control. A human user can, however, interrogate the system or assume control over either selected portions of the testbed or the complete testbed.

Considering only the autonomous (no human intervention) activity, the common control function for the SSM/PMAD is composed of the Automation Control Software Functions - FELES, FRAMES, LPLMS, KANT, KNOMAD, and MAESTRO. These functions are mostly knowledge based systems relying on different AI techniques. It should also be noted that each PDCU and LC contains a resident LLF. The LLFs reside in a distributed set of LLPs and are individually capable of handling the low level control functions that consist of data handling and switch commanding. The important concept is that the SSM/PMAD Testbed developed a common control function composed of independent entities whose values for management of a complete system may be assessed. However, the SSM/PMAD represents only one subsystem's view, the power subsystem. So when assessing the overall applicability of each functional element for integrated subsystem management, only the abstracted element's contribution to that integrated function is considered.

FRAMES is a fault management function. It is imperative that subsystems be allowed to recover from faults internally. It is also imperative that a system's overall goal be achieved if possible. So whenever a faulted subsystem's level of performance has diminished, a system-wide fault manager should execute to autonomously press forward, guiding each subsystem on an as needed basis. From a planning viewpoint, integrated subsystem fault management is applicable system-wide.

KANT is a planning function within the SSM/PMAD Testbed that aids a user in gaining manual control over selected power subsystem components. A user would also need access to an integrated system for both control and redirection purposes. Planning (reactive and otherwise) is applicable to integrated autonomous subsystems management.

FELES and MAESTRO are scheduling oriented functions. System-wide scheduling is an important activity within integrated subsystems management.

The KNOMAD knowledge management function manages rule oriented and frame oriented knowledge bases. More importantly, underlying its AI paradigm is the fact that its primary function is to make known appropriate information to the various management entities (e.g., a fault may cause rescheduling) so that each entity is aware of the information which may affect its performance. Selected knowledge and information management is very important between subsystems so that an individual subsystem can more effectively use its

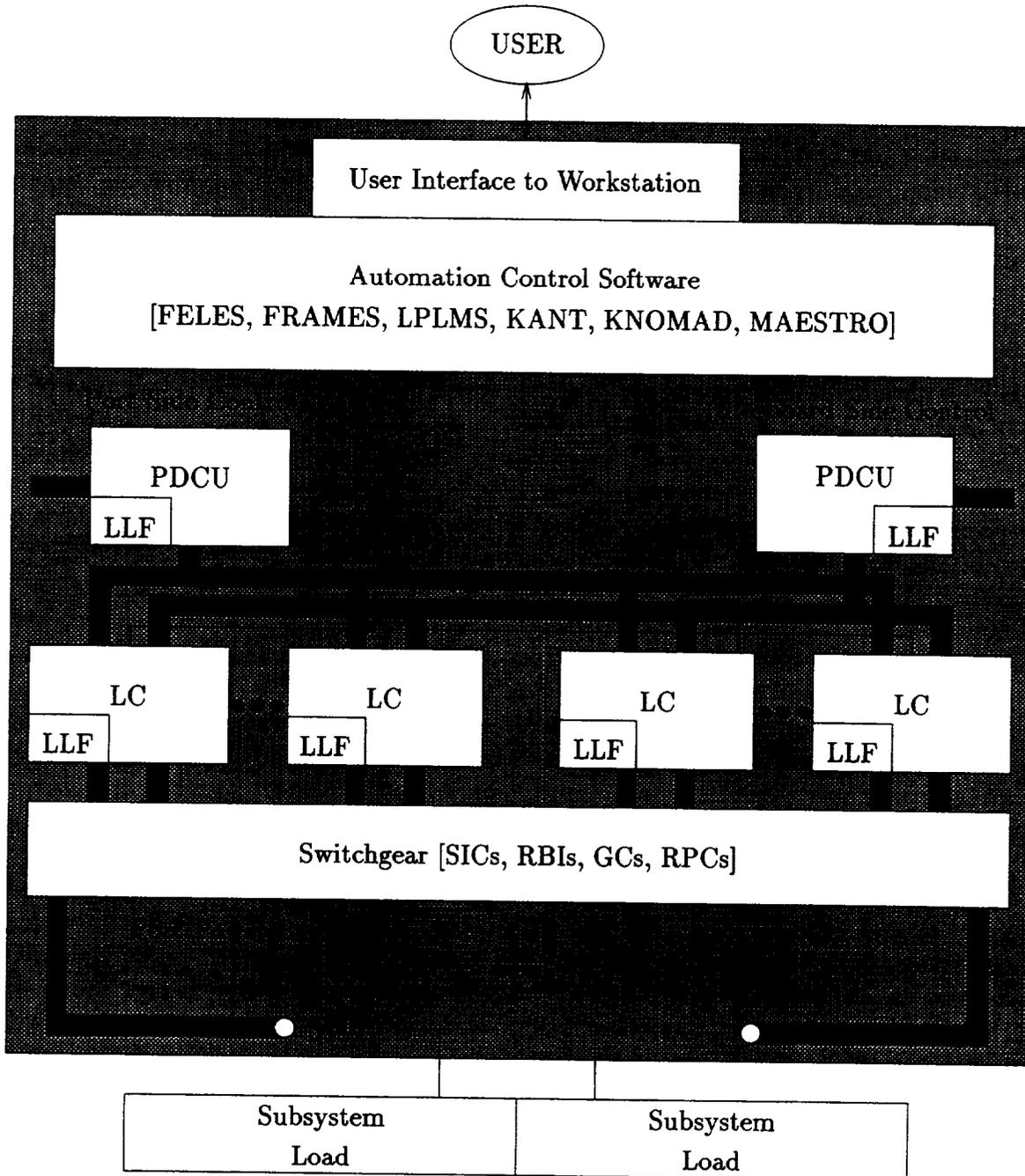


Figure 2: SSM/PMAD Testbed Elements and Architecture

$$\begin{bmatrix} S_{11} & S_{12} & \dots & S_{1n} \\ S_{21} & S_{22} & \dots & S_{2n} \\ \dots & \dots & \dots & \dots \\ S_{m1} & S_{m2} & \dots & S_{mn} \end{bmatrix} \quad \begin{bmatrix} T_{11} & T_{12} \\ T_{21} & T_{22} \\ \dots & \dots \\ T_{n1} & T_{n2} \end{bmatrix}$$

Figure 3: The Priority Calculation Array Concept

resources.

Priorities are managed within the SSM/PMAD Testbed by the LPLMS. When viewing a complete system, this is probably the single most important function. This is because a goal has a greater chance of being achieved when the priorities of each subsystem and the relation of subsystems interactions are known with increased exactness. To achieve this capability the SSM/PMAD Testbed employs a multi-dimensional approach which is conveniently termed a tensor approach. The reader is referred to {Arf, '70} for a complete introductory treatment of tensors and their mathematical properties.

The SSM/PMAD priority tensor is not treated with the same mathematical rigor as the tensors commonly used in physics (e.g., the metric tensor). What the priority tensor allows is definition of a priority treatment such that the individual priorities remain consistent when viewed from any subsystem within the system employing the priority tensor.

For the SSM/PMAD priority tensor there are two distinct elements: first, the load category which at present is not allowed to change; and second, the priority rating which is calculated by establishing a set of weights and then calculating the priority number within the range of the weights. The weights can be thought of as algebraic coefficients whose domain can be adjusted by a given set of productions.

For a multiple subsystem system this approach can be expanded allowing each subsystem to represent a single dimension within the tensor representation. Relations and rules can then preserve the representation of the priority tensor, no matter which subsystem is the observer. A representation would look like that in Figure 3. Each subsystem is represented by one row of elements within the S matrix, and operations on the priorities are represented by its associated operations T array, with individual priorities summing across array operators.

A most stringent restriction (one that was not needed in the SSM/PMAD Testbed LPLMS) would be placed on calculating the weights for the S elements; that is, any individual subsystem priority must remain consistent, irrespective of what subsystem is viewing it. However it may be allowable for the overall system priority to change. For example, for

a given system state, the power subsystem may view its individual priority ranking as 0.8, the thermal management subsystem's priority ranking as 0.5, and the overall system priority as 0.95. At the same time the data handling subsystem may view the power subsystem's priority ranking as 0.8, the thermal management subsystem's priority ranking as 0.5, and the overall system priority as 0.9 because its view of the overall system is different.

4 DIFFERENT SUBSYSTEMS - DIFFERENT SUBSYSTEM MANAGEMENT NEEDS

Each different subsystem to be managed under an integrated management function poses a unique set of needs to that management function. Generally speaking, this may not always be the case as some complex subsystems possess many different operational characteristics, such as synchronous and asynchronous behavior (e.g., attitude control subsystems - {Wer '84}). These more complex subsystems require a broader range of management functionality.

4.1 Asynchronous Subsystems

The SSM/PMAD Testbed operates in the asynchronous mode. Most subsystem activity that can be viewed as being free of processing data on consistent time intervals can be considered as asynchronous. Many other subsystem activities fall into this category. For example, radiating heat into space from a thermal radiator is asynchronous with respect to any associated data processing. This is not to say that asynchronous subsystems are not subject to scheduling for activation and deactivation, for they are; and, an integrated subsystems manager must take their particular needs into consideration.

These asynchronous subsystems present their own unique problems for integrated management activities. First, even though the subsystem may not be counting time directly, it may be servicing a subsystem that is. Therefore, connected subsystems must be considered whenever a servicing subsystem is engaged in a support activity. Next, asynchronous subsystems may depend upon other subsystems for portions of their internal management creating a need for a more robust integrated fault manager.

4.2 Synchronous Subsystems

Subsystems that operate with dependencies on processing data at particular and repeatable time intervals are considered synchronous. These type subsystems are particularly vulnerable to interruption, for they may be processing on data being supplied from other operational subsystems. These other subsystems in turn may also be synchronous in their operational characteristics. Therefore, when a state update is incrementally occurring in a synchronous

system it is important that no interruptions occur to that system during that time. This requires the full attention of an integrated subsystems manager.

4.3 Data Processing Subsystems

Data processing elements may appear to be synchronous or asynchronous depending on the particular subsystem application. Also, the integrated subsystems management process would typically be resident in the data processing environment. Due to this and many other historic reasons, fault tolerant computing is a highly developed (usually more so than any other subsystem) engineering discipline. In a critical environment this limited approach would not be enough. All subsystems would need to be equally reliable, and their interactions managed to ensure the fundamental system goal. This relation between the data processing subsystem and the integrated subsystems manager further unifies data handling and processing elements into a unique subsystem.

4.4 Sensory Subsystems

Sensory subsystems are unique. Most often they provide data to an array of subsystems in a synchronous manner. However, they can also provide direct data on a completely asynchronous basis, such as a microprocessor based voltage sensor that only provides a voltage reading upon request.

Sensory subsystems can be as simple as one sensor, or they can consist of a large complex array of multiple sensor types. No matter their topology, sensors provide the elemental data for the individual subsystems and the integrated subsystems manager.

5 INTEGRATED SUBSYSTEMS MANAGEMENT, END-TO-END SYSTEM RELIABILITY, AND OPERATIONS COST

The primary goal of an integrated subsystems manager should be to increase overall system reliability while also driving down system operational cost. The system reliability concept is formed considering all applicable subsystems, all possible users, and for the lifetime of the system. Usually, the reliability of individual subsystems is viewed as a parallel process as shown in Figure 4 (a). The parallelism is provided through redundant components. Individual component failure is then compensated by a fresh component assuming the function. The equation for the reliability of parallel component systems is {Dou, '88}:

$$R(t) = 1 - \prod_{i=1}^n (1 - R_i(t)),$$

with the reliability function being allowed to vary with time.

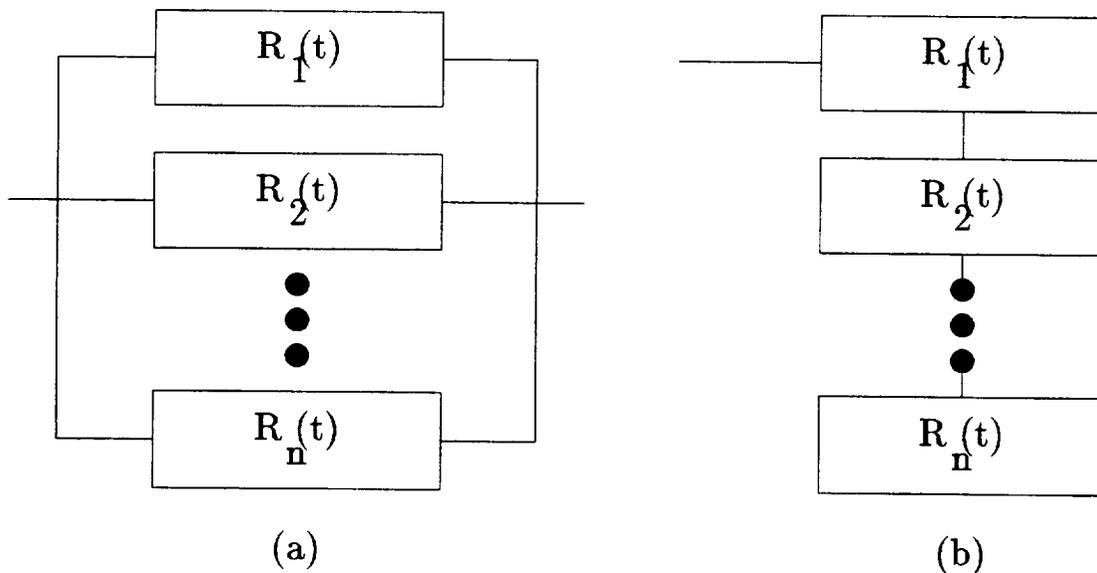


Figure 4: Parallel and Series System Reliability Models

A complete system is composed of individual subsystems operating in series as shown in Figure 4 (b). The equation for this configuration is:

$$R(t) = \prod_{i=1}^n (1 - R_i(t)),$$

with once again the reliability function being allowed to vary with time.

It can be readily seen from the above equations and Figure 4 that (n-1) failures can be handled in a parallel approach while a single failure in the sequential model spells doom. This makes it ever more important to include an integrated subsystems manager to increase the operating reliability of the overall system by better directly managing the complete capabilities of the series system and providing more complete state information to subsystems for better utilization of parallel resources.

By providing autonomous, rapid response fault management for the system, the integrated subsystems manager can relieve some of the heavy personnel burden that is experienced with complex systems. This in turn helps to lower operational costs and decreases system down-time. Coupled with the increase in end-to-end system reliability, the integrated subsystem management provides a higher degree of usability and efficiency.

6 INCREMENTAL REDUNDANCY

Whenever systems of the nature of those discussed in this paper are implemented one problem always arises, a large increase in the system parts count. This in turn adds complexity to the system and can further feed upon itself by again increasing the system parts count. Therefore, care must be taken to manage this situation when designing and implementing any integrated subsystems management function.

One technique that has been used in successful subsystems design, that of incremental component redundancy, works well with integrated subsystems management. An example from the data processing subsystem is incremental shared memory. A block of bulk memory is added onto an existing block of memory with a separate access controller providing shared access if needed. So if memory is more likely to fail than a processor, then enough memory blocks could be added to match the total number of processors to match some target reliability. This is then managed by the integrated subsystems manager, especially in a distributed architecture.

Another example would be adding cells from a quiescent battery to an active battery with cell failure where the two batteries would not be active at the same time. This would eliminate the need for a complete battery replication while providing equivalent duplication capability of the power source.

7 ACKNOWLEDGMENT

This work was performed by Martin Marietta Astronautics Group as Independent Research and Development in conjunction with contract number NAS8-36433 to the NASA George C. Marshall Space Flight Center, Huntsville, Alabama.

8 REFERENCES

- {Arf, '70} George Arfken, "Mathematical Methods For Physicists", Academic Press, 1970.
- {Dou, '88} Edward R. Dougherty and Charles R. Giardina, "Mathematical Methods for Artificial Intelligence and Autonomous Systems", Prentice Hall, 1988.
- {Wer, '84} James R. Wertz, et al, "Spacecraft Attitude Determination and Control", D. Reidel Publishing Company, 1984.

Managing Autonomy Levels in the SSM/PMAD Test Bed

Barry R. Ashworth
Martin Marietta Astronautics Group
MS: S-0550
P.O. Box 179
Denver, Co. 80201
bashworth@den.mmc.com

ABSTRACT

Whenever mixing autonomous operations with those of a manual nature, concepts concerning the boundary of operations and responsibility become clouded. The Space Station Module Power Management and Distribution (SSM/PMAD) automation testbed at the NASA, George C. Marshall Space Flight Center in Huntsville, Alabama possesses the need for such mixed-mode capabilities. Martin Marietta, under contract to NASA/MSFC since 1985, is continuing the development of technologies and methodologies for use in the SSM/PMAD automation.

Enabling manual seizure of system elements which are currently under automated or autonomous control is a complex problem. Element relations may exist in some systems which do not lend themselves to isolated autonomous management, forcing an unaided manual control user into managing elements beyond the user's capabilities. Also, termination of manual activities within an environment originally intended for autonomous management does not necessarily reinstate the system into its intended autonomous mode. Issues which are surfaced by this manual seizure of autonomously managed elements are: 1) system resource allocation, 2) managing the system component priorities, 3) aiding the user in lessening impacts on any remaining autonomous elements, 4) planning the proper course of events, 5) maintaining an operational system configuration, 6) maintaining any special autonomous capabilities, such as fault recovery, and 7) striving to minimize impacts to any scheduling activities. Here the concept of managing the SSM/PMAD testbed in the presence of changing levels of autonomy is examined. A knowledge-based approach to implementing autonomy management in the distributed SSM/PMAD utilizing a centralized planning system is presented. Its knowledge relations and system-wide interactions are discussed along with the operational nature of the currently functioning SSM/PMAD knowledge-based systems.

1 NOMENCLATURE

AI Artificial Intelligence

AC, ac Alternating Current

CAC Communications and Algorithmic Controller

CAS Communications and Algorithmic Software

CC(E) Computation and Control (Engine)

DC, dc Direct current

FELES Front End Load Enable Scheduler

FRAMES Fault Recovery and Management Expert System

GC Generic Controller

Hz, hz Hertz (cycles per second)

K, k Kilo (1000)

KW, kw Kilowatt (1000 watts)

LC Load Center

LLF Lowest Level Function

LLP Lowest Level Processor

LPLMS Load Priority List Management System

MSFC (George C.) Marshall Space Flight Center

NASA National Aeronautics and Space Administration

PDCU Power Distribution Control Unit

RBI Remote Bus Isolator

RCCB Remotely Controlled Circuit Breaker

RPC Remote Power Controller

SIC Switch Interface Controller

SSM/PMAD Space Station Module (Automated) Power Management and Distribution (System)

2 INTRODUCTION

Future space systems are showing increased requirements for applications of advanced automation. Many of these systems will be manned and present unique situations whenever people interact with them. The SSM/PMAD {Mil, et al, '89, NASA Contractor's report on the SSM/PMAD,} is one such system. It has been developed to perform the management of a representative Space Station Freedom power management and distribution system in a completely automated manner. However, if such a system were in actual use, it would be required to allow human intervention at any level within its operational environment.

A user's capability to interrupt the automated management of a system must exist, and does on the SSM/PMAD. However, in its present configuration, the user assumes control for

the entire SSM/PMAD management function when interrupting its automated operation. This exposes areas of operational weakness for both the automated elements and the users. First, a human user could probably not control the complete SSM/PMAD testbed with success; and second, the testbed should allow partial levels of intervention by someone who requires a manually controlled subset of testbed components. This paper focuses on the second point: that which must be considered when managing changing levels of automation in the SSM/PMAD testbed.

The controlling operational elements of the SSM/PMAD are shown in Figure 1. Users interface to this system only through the workstations. This does not limit the utility or accessibility of the system. Instead, it makes the system more understandable for the user. Presently, user-interfaces exist on two distinct workstations: a Solbourne 5/501 and a Symbolics 3620D. The initial schedule of automated operations is generated at the Symbolics workstation, using its interface. After that, it is utilized only to access information on the schedule of operations that was initially produced. All other run-time accesses of data and control are made at the Solbourne interface. It is at this interface that a user assumes manual control of the testbed; presently, only complete control is issued.

If the SSM/PMAD allowed a user to assume chosen partial control of the testbed, many questions arise. Some of these are:

- where and how should resources be allocated?
- how are priorities combined and managed?
- what guidance should the user be given?
- should planning guide the course of events?
- what is an acceptable system configuration?
- will a new plan interfere with automation?
- can effects on peripherally scheduled operations be avoided?

Operational aspects of system automation for the SSM/PMAD must be understood in order to answer the above questions. These operational aspects and their relation to automated levels of testbed management are paramount to providing incremental levels of user control.

3 AUTONOMY MANAGEMENT

There are three prime functions within the SSM/PMAD testbed. First, provide lower level control for the power hardware elements; second, perform health monitoring and system

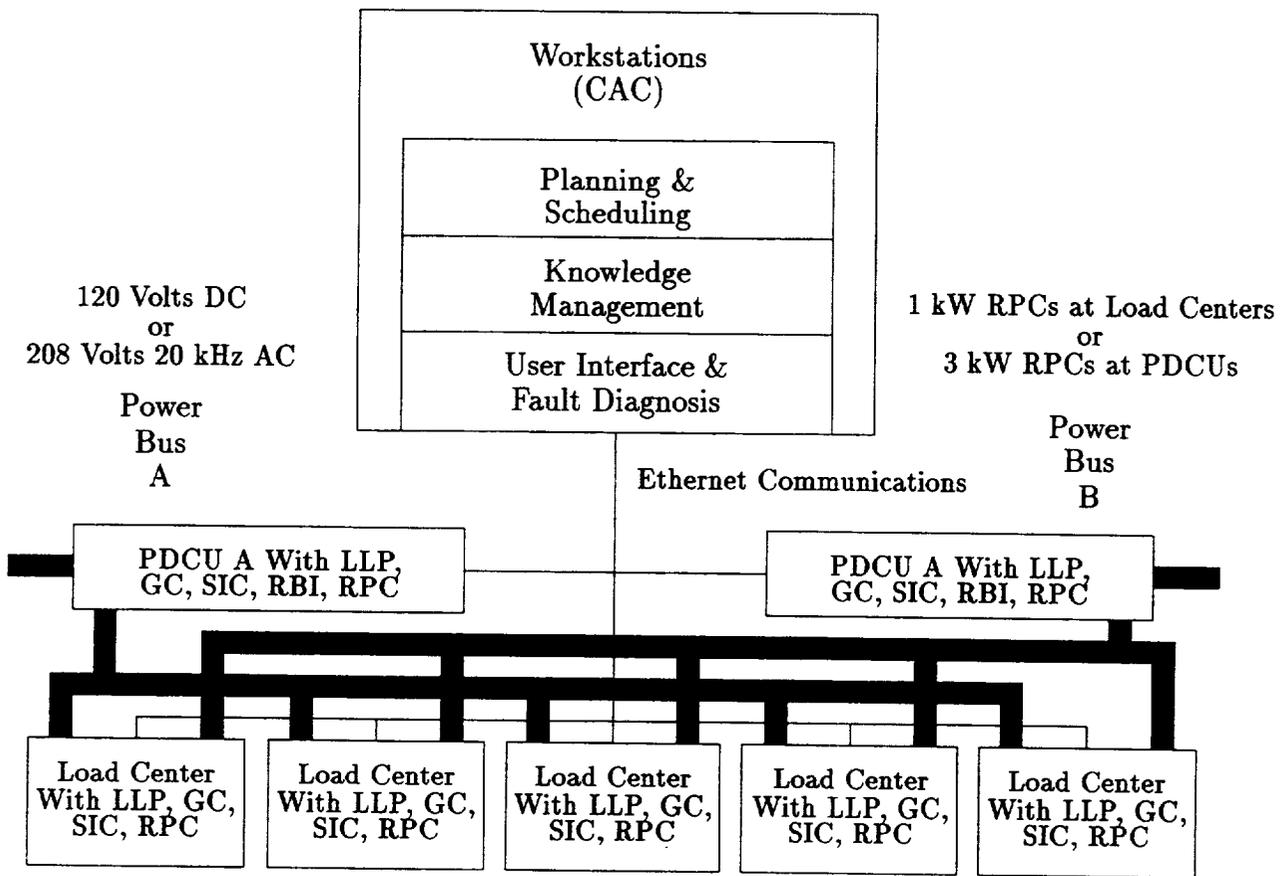


Figure 1: The SSM/PMAD Block Diagram

level management; and third, perform diagnosis and recovery during the occurrence of faults. These fundamental activities of the testbed give rise to many subtle interactions, producing further functional elements. Careful and concise consideration must be given to all functional interaction when changing the roles of the automated elements.

Planning activities can integrate control over the functional interactions whenever a user interdicts with manual control. A reactive automated planning process {Ash, '89} is necessary to achieve decreased levels in automation. This should be provided such that any critical functional considerations are given to all of the elements which are targeted for removal from automated management.

Consideration should be given to minimizing response times whenever users interact with automated planning activities. This is especially important in large distributed systems, such as the SSM/PMAD. For further explanation see {She, '89}.

3.1 Lower Level Control

Control at the PDCUs and LCs (see Figure 1) is provided by the LLPs. Each LLP manages its local switch elements as defined by an event list schedule. This schedule is produced by the automated program element, MAESTRO. After production, the schedule is divided into the appropriate enabling lists by the FELES. It is then sent to each active LLP by FRAMES, which then notes the LLP activities at a system management level. Each active LLP then becomes responsible for managing its local environment. This provides the distributed autonomous lower level control for the testbed.

It is apparent that control of switches should not be given to a user of this automated system without first informing each affected LLP of that user's intervention. This prevents an LLP from changing the state of switches which a user believes he is controlling. Therefore, starting from the bottom, the first element which must be informed of a change in the automation structure is the LLF, as to its view of the local resource domain.

The LLFs also perform load shedding according to a priority list built by the LPLMS. Since an LLF could only load shed those resources within its control domain, its priority list must be modified whenever its resource domain changes.

A question of how the LLF becomes informed of resource domain and priority changes is important because the answer dictates where the automation control over the change exists. If the LLP is informed at its level by a user that some resources are to be removed, it must suspend its operation while communicating to the management level for new domain schedules and priorities. This would leave automated control elements at risk during the interim process period. This strongly urges that the LLPs are informed of autonomous reduction in operations only after the user has gained access and the control and management decisions have been made.

3.2 System Level Management

Management and health monitoring are critical in the SSM/PMAD. Data are collected at the LLPs and sent to FRAMES for an integrated view of the testbed's health. During periods of nominal activities FRAMES does not intervene in the management of the loads at the lower level. However, if an anomalous event occurs, FRAMES performs diagnosis while viewing the offending element from a system-wide perch.

Being rule driven and utilizing frame based reasoning, FRAMES is an AI agent. There are other AI agents in the system. They are the LPLMS which manages the priority list for the testbed loads, the MAESTRO scheduler, and the FELES which performs some knowledge based activities when managing the user interface. Therefore, the user must interact with a multi-agent system during its execution, and this revolves around the individual agents sharing knowledge about the power domain through common plans, models, and schedules.

With so many managing agents, a user would probably not know where to start in order to gain control of power resources. To provide this capability, and to ensure uniform system behavior, automated reactive planning is not enough; a complete knowledge management environment is also needed {Ria, '90}. Managing the various knowledge agents in the SSM/PMAD testbed is a complex automated parallel activity. Figure 2 shows a multiple event update which describes the equation:

$$\omega_i = \sum_j \{ \prod_k \frac{\Delta f(E)}{A_j} \}; j = i, N; k = j, N;$$

A = Set of Knowledge Agents;

E = Vector of events.

The result is that the activities taken at each managed knowledge agent are functionally dependent upon the shared events among all knowledge agents. This provides for managing the multiple knowledge agents in parallel within the system. Thus, the user need not be concerned to which knowledge agent he makes access; the knowledge manager provides correct access to the appropriate knowledge function.

Managing knowledge base execution and control must be complemented by an integrated approach to database management. This alleviates a manual-mode user from accessing data items whenever system resources are placed under manual control. For example, a user who gains control of a particular suite of switches should be free of identifying any internal software representations or data values for those switches.

Just as the knowledge and data are managed in an autonomous manner, so are the priorities of the loads being powered. If several switches are available the reactive planning function should cooperate with the user to minimize any priority impacts. A user could otherwise remove higher priority loads being managed autonomously.

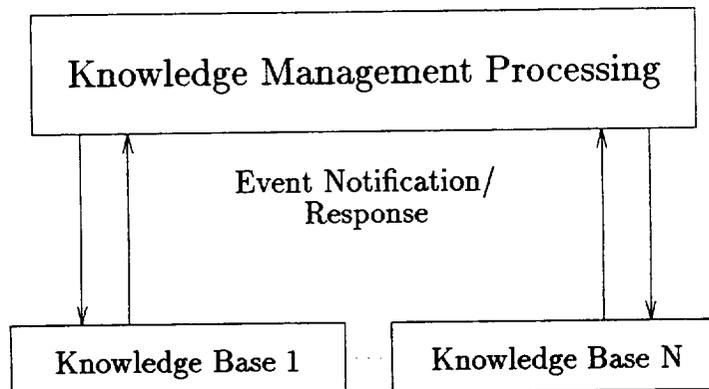


Figure 2: Event Updating

3.3 Diagnosis and Recovery

Diagnosis, isolation, and recovery of faults {Rib '90} is handled by the specialized FRAMES knowledge base. A strong model dependency exists for FRAMES to understand the power network. Therefore, it is important that any changes in activity or configuration be reported to FRAMES. This could be handled by the reactive planning agent responding not only to the user's requests, but also to FRAMES specialized knowledge needs.

Another aspect of fault management within a large distributed system {Dur, '89} is resource deadlock {Sin, '89}. It is important that any planning function should examine the relation between resources. Whenever a resource conflict arises, deadlock scenarios and potentially non-recoverable faults become possible. It then becomes imperative that a mediator or planning function discovers these scenarios before they become real.

Recovery from faults is more than just the recognition of a fault. For example, the SSM/PMAD activities depicted in Figure 3 show that the system management activity is of ultimate importance. Interference with the activities at the lower levels by FRAMES does not occur. However, FRAMES does redirect the production of schedules according to a plan of activities and a model of the power network. In doing so, FRAMES is most concerned with system health and fault isolation during fault occurrence. This demonstrates the importance of allowing FRAMES the understanding of any system reconfiguration or redirection of control. Otherwise, erroneous fault diagnosis would possess a strong probability for occurrence. Following this, any replanning or rescheduling activities would be equally faulty.

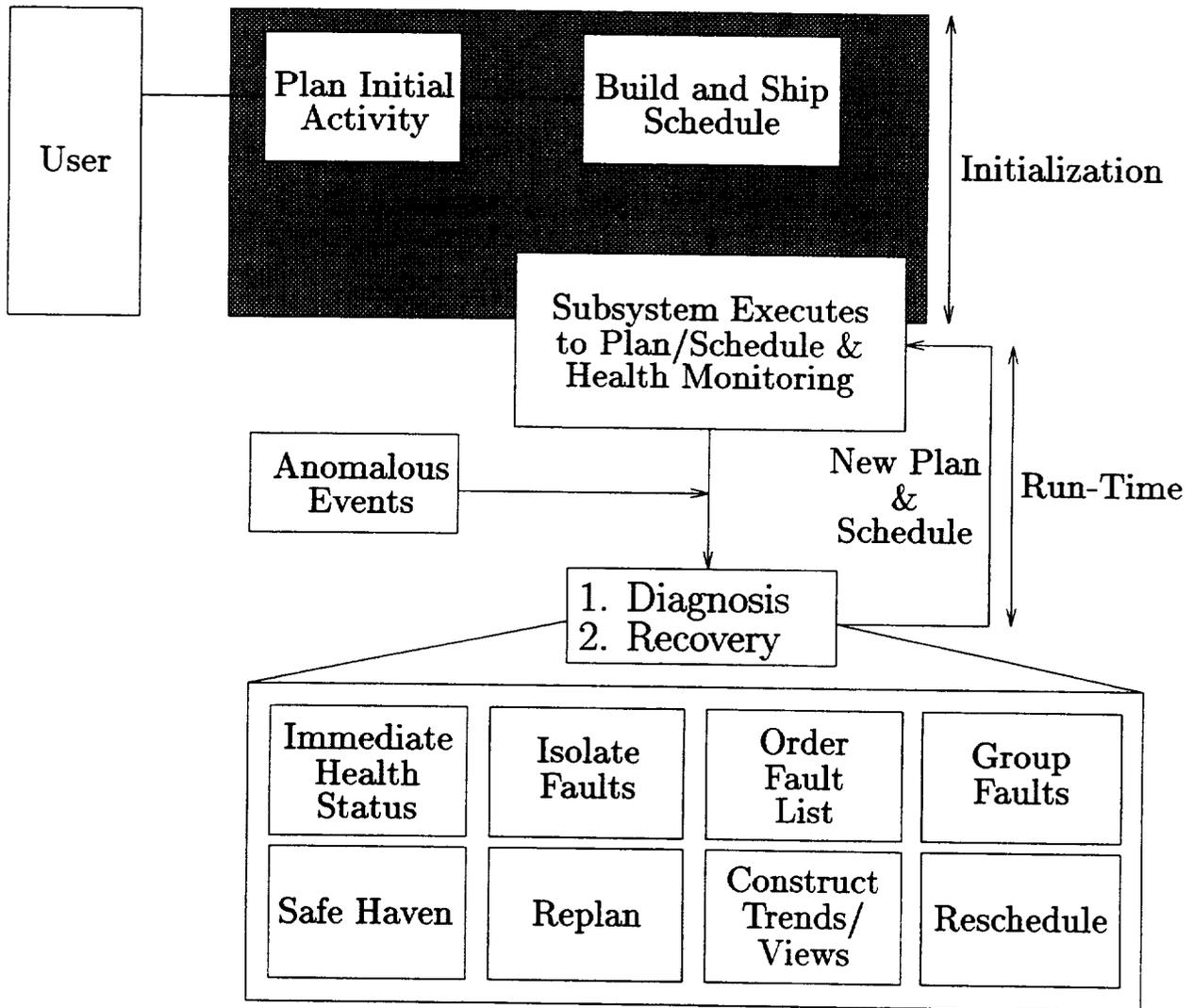


Figure 3: The SSM/PMAD Diagnosis and Recovery Operation

3.4 Planning and the Manual User

In order for a user to manually intervene into the automated management of the SSM/PMAD testbed in a near optimal manner, it is necessary for an automated planning function to be available. The primary purpose behind the reactive planning element would be to cause as little disturbance as possible to the loads being managed under the autonomous portion of the system. The knowledge flow within the SSM/PMAD providing this capability is shown in Figure 4.

A successful planning approach within this real-time critical environment would be reactive to the dynamic nature of loads and users. The planner would not solely initiate activities or decisions as to the status of resources. It would instead function as a working entity to relate the knowledge and information from the automated managing elements to a user. Where appropriate, the planner would provide suggestions to a user as to what resources may be available and at what times. This could be done by forming a list of Lower Level Process Activities and their termination times. Combining this with the information in the priority list and a system health status update from FRAMES, the planner could aid the user in avoiding impacts to the automated system. The user could also query for aid in determining his priorities as compared to those in the automated system.

Once the planner had aided the user in inserting the manual activities, the planner could further assist the user in combining the manual mode priorities with those on the automation list to form uniform priority blocks. There may be a large advantage to this in some cases. For example, if a power reduction occurred the user would not be responsible at that time to participate in priority reordering. The system would perform the activity and the manually controlled switches would be subject to their user/planner developed priorities.

3.5 Consistent User Interface

The user's view into the SSM/PMAD system should be consistent, whether the elements are being managed autonomously or manually. This view applies strong requirements on the system from bottom to top.

If a user is allowed to query the interface for data on manually controlled resources while the automated management resources are simultaneously executing, then the automated management resources must respond to the manually controlled resources in a modified manner. Commanding a switch resource on or off, or issuing a data query should appear consistent. The user should not be concerned with the system percentage of autonomously controlled switches.

In order for the user interface to remain consistent, the FRAMES activities should be informed of manually controlled activities in the system. The LLFs should be given a switch recognition message which identifies maximum power limits (these can be defaulted to some averaged value if needed). Whenever a switch is manually activated, a message would be

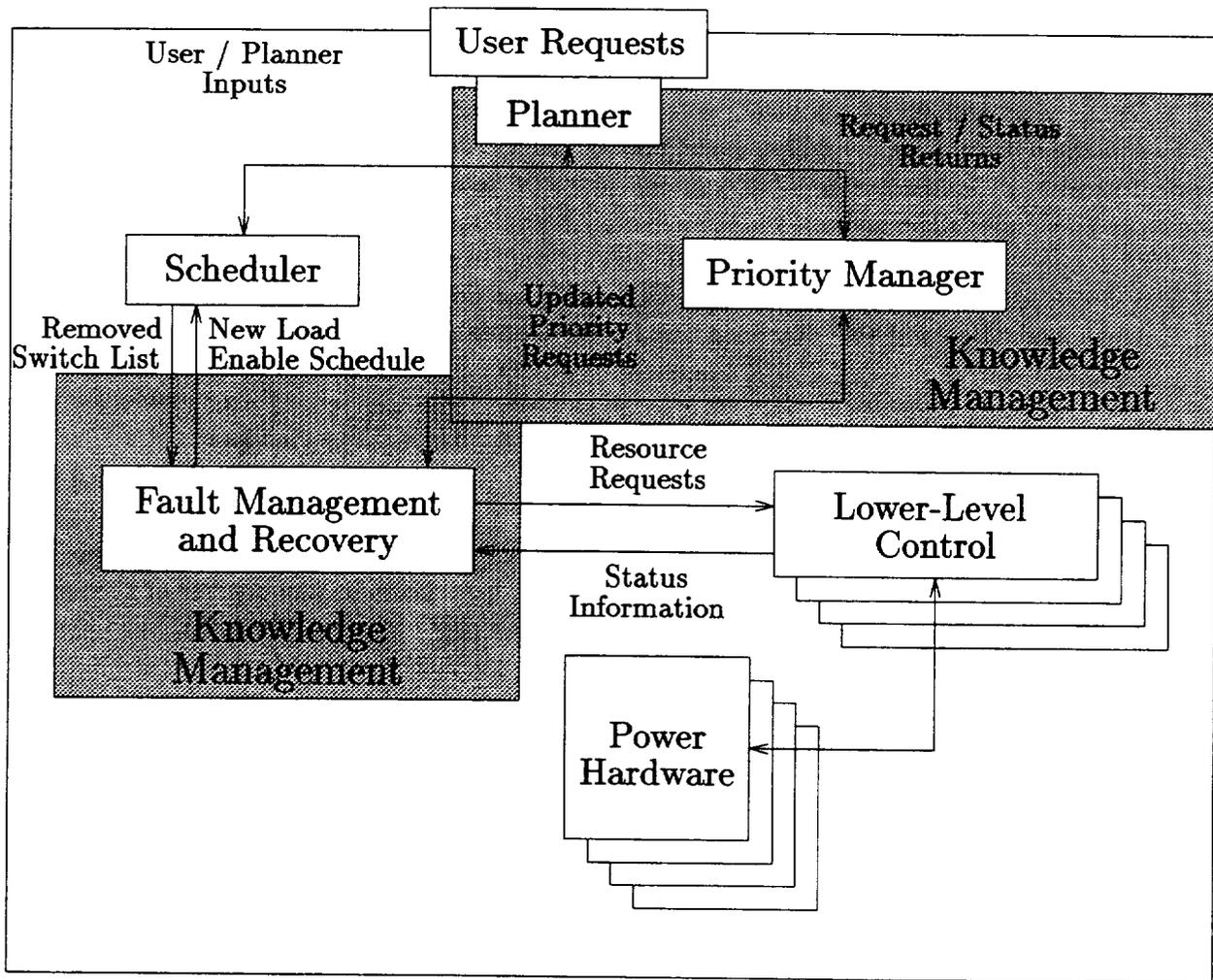


Figure 4: The SSM/PMAD System Knowledge Flow

sent from the user interface to FRAMES. The activation would be noted and the command forwarded to the appropriate LLP. Progress and power usage for the manually controlled switches could then be monitored. This would allow automated messages to the user in the event of problems at manually controlled switches.

4 SUMMARY

Managing human and machine automated interactions presents unique problems, not the least of which is measuring the effects for introducing new uniform priority blocks into a currently operating system. Short of being able to rule by a sovereign and complete automated interaction, a concept for allowing this human/automation mix in real-world systems is to provide graceful return of auto-managed functions to human control. This requires that specialized planning activities be introduced into the automation system which react to the environment modifications and to the system health status.

The planning function interacting with the present automated management functions must establish the operational environment. This environment represents that portion of the operational world which is removed from the automation system's responsibility. Removal of elements from automated management does not imply that the automation system need not be informed of their behavior. In cases of faults or power reduction, manually controlled resources are subject to system examination.

Whenever a user takes manual control of a subset of SSM/PMAD system resources, there are many functional considerations. The model which is used by FRAMES must reflect any user directed changes. Also, the LLPs must be informed of each individual control command as it takes place. FRAMES must examine and log each user command so that the system health status may be maintained. In order for these things to be accomplished with little or no user impact, the user interface must remain consistent whether being driven manually or autonomously.

By automatically planning and managing those changes introduced by a manual user the SSM/PMAD can gracefully return automated system resources to manual control as needed. This aids the user in managing resources and priorities, while providing guidance in planning a course of events. Therefore, significant impacts are avoided and automated management is maintained where needed.

The SSM/PMAD has always provided a full manual mode capability. This capability would not be disturbed by the introduction of partial autonomy modes, but would be enhanced.

5 ACKNOWLEDGEMENT

This work was performed by Martin Marietta Astronautics Group under contract number NAS8-36433 to the NASA George C. Marshall Space Flight Center, Huntsville, Alabama.

6 REFERENCES

- {Ash, '89} B. Ashworth, "Reactive Autonomous Planning in Spacecraft", In Proceedings of the ACM Conference on Aerospace Applications of Artificial Intelligence, 1989.
- {Dur, '89} E.H. Durfee, V.R. Lesser, and D.D. Corkill, "Trends in Cooperative Distributed Problem Solving", In IEEE Transactions on Knowledge and Data Engineering, Vol. 1, No. 1, 1989.
- {Kam, '89} S. Kambhampati and J. A. Hendler, "Control of Refitting During Plan Reuse", In Proceedings of the Eleventh IJCAI, 1989
- {Kli, '85} P.J. Kline and S.B. Dolins, "Choosing Architectures for Expert Systems", Rome Air Development Center, Air Force Systems Command, Griffis AFB, RADC-TR-85-192, October 1985.
- {Lan, '88} A.L. Lansky, "Localized Event-Based Reasoning for Multiagent Domains", In Computer Intelligence, Volume 4, 1988.
- {Mil, '89} W. Miller, et al, "Space Station Automation of Common Module Power Management and Distribution", NASA Contractor Report 4260, Martin Marietta Aerospace, Denver Astronautics Group, 1989.
- {NAS, '88} NASA, "Space Station Advanced Automation Study Final Report", Strategic Plans and Programs Division, Office of Space Station, NASA Headquarters, May, 1988.
- {Ria, '90} J. Riedesel, "Knowledge Management: An Abstraction of Knowledge Base and Database Management Systems", In Proceedings of the Fifth AI Systems in Government Conference, 1990.
- {Rib, '90} J. Riedesel, "Diagnosing Multiple Faults in SSM/PMAD ", In Proceedings of the 25th IECEC, 1990.
- {Rie, '89} J. Riedesel, C. Myers, B. Ashworth, "Intelligent Space Power Automation", In Proceedings of the IEEE International Symposium on Intelligent Control, 1989.
- {She, '89} S. Shekhar and S. Dutta, "Minimizing Response Times in Real Time Planning and Search", In Proceedings of the Eleventh IJCAI, 1989.

- {Sin, '89} M. Singhal, "Deadlock Detection in Distributed Systems", In IEEE Computer, November, 1989.
- {Tu, '89} S.W. Tu, et al, "Episodic Skeletal-Plan Refinement Based on Temporal Data", In Communications of the ACM, December, 1989.
- {Zha, '89} Hui Zhang, A.C. Sanderson, L.H. de Mello, "Generation of Precedence Relations For Mechanical Assembly Sequences", In Proceedings of the IEEE International Symposium on Intelligent Control, 1989.

An Efficient, Distributed Architecture for Cooperating Expert Systems¹

Joel D. Riedesel
Martin Marietta Astronautics Group
P.O. Box 179, MS: S-0550
Denver, Co. 80201
jriedesel@den.mmc.com

¹This work was performed under contract NAS8-36433 to NASA, George C. Marshall Space Flight Center.

Abstract

An architecture to support cooperating expert systems is presented here. This architecture has two primary requirements: one, it must support a mechanism for sharing relevant parts of working memory between the expert systems; and two, it must support a mechanism that provides efficiency in the expert systems themselves.

These mechanisms have been implemented in KNOMAD, a **Knowledge Management Design System. The sharing of relevant portions of working memory is provided by a database; similar in concept to a blackboard mechanism, but more general. Efficiency is provided by compiling rules and rule patterns into a data network; again, similar in concept to the RETE net, but more general in some aspects.**

While it may not be difficult to define a paradigm for cooperating expert systems for a particular domain, providing a general paradigm that is domain independent and remains efficient is much more difficult. The expert system paradigm of KNOMAD will be presented here as one example of providing domain independence without sacrificing efficiency.

Conference Topic: Automated Reasoning

SubTopic: Distributed Artificial Intelligence

1 Introduction

Current expert system applications have become much more complex than in the past. It is no longer sufficient to write a single expert system as a solution to a problem. Today's problems require interaction between subsystems and controlling systems, between networks of systems cooperating together, and between artificial intelligence systems and conventional systems. To automate space vehicles and platforms, for example, it is only possible to define an expert system for each subsystem (telemetry, attitude control, power system, life support, etc.) and then have them cooperate with each other. It is quixotic to expect one expert system to control all the subsystems at once. Many aspects of cooperating expert systems also expect communication and cooperation with conventional, algorithmic processes that are more natural for the particular problem.

These needs for cooperating expert systems derive a number of requirements that an architecture for supporting cooperating expert systems will need. Perhaps the foremost requirement is that of communication. There are at least two aspects to communication: signalling other processes and sharing data. Both aspects may be supported by some sort of a database mechanism, whether a shared memory or a message passing approach is used.

A second requirement that an architecture needs to fulfill is that of generality. In order for an architecture to be useful it must be possible to use it in many different domains. The generality of an architecture implies that one expert system may be defined for one subsystem (domain) and another expert system for a second subsystem (domain). These expert systems must be able to communicate with each other. And, perhaps most importantly, they must be able to reside on different physical machines while the architecture continues to provide the support for communication between them (which is what most blackboard systems do not provide). A second, practical implication of providing generality is that the architecture must also provide efficiency. For the expert systems to be useful, they must be fast.

Although there are probably many more requirements for cooperating expert systems, these two were used to design the KNOMAD (**Knowledge Management Design System) architecture. The following section gives an overview of KNOMAD. The three sections after that describe how KNOMAD supports communication and efficiency as well as providing generality. Finally a discussion and conclusions are given.**

2 KNOMAD

The KNOMAD architecture provides general support for knowledge representation and reasoning. KNOMAD is defined in three layers (see Figure 1). The bottom layer defines the database that is used to store working memory data, model data, and any other data that might be used by an expert system or algorithmic process. The database layer is designed to be completely modular such that any database could be used in its place. The second

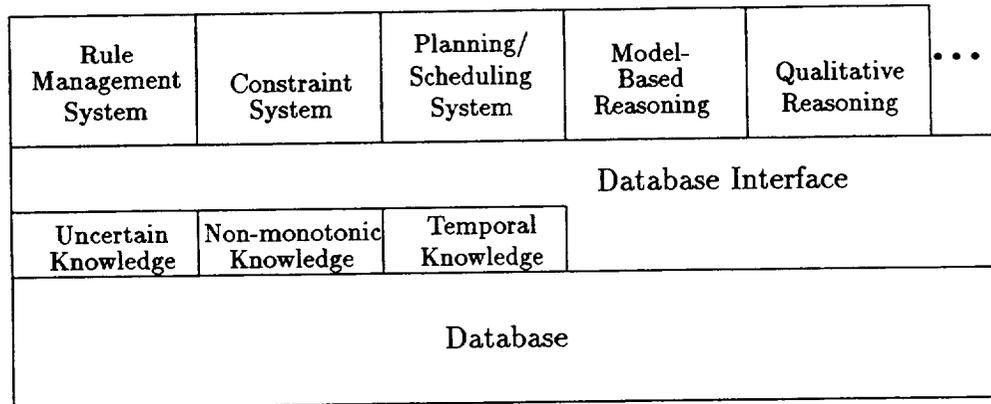


Figure 1: KNOMAD Layered Architecture

layer is the database interface. This layer manages and organizes data for various knowledge representations. It also manages implicit links between data and the inference mechanisms of the top layer. Conceptually, the database interface provides the means for supporting a knowledge representation in terms of simple data and linking it to an inference mechanism that uses that knowledge representation. Finally, the top layer is where the various inference mechanisms are defined. This paper discusses the rule management system in terms of its paradigm for expert systems.

The following sections discuss how the KNOMAD architecture supports sharing data between expert systems, how efficiency is provided for expert systems and how the expert system paradigm supports generality. These three aspects of KNOMAD provide the necessary ingredients for cooperating expert systems. For a more detailed discussion of the KNOMAD architecture see [?].

3 Sharing Working Memory

The primary means for communication between expert systems in the KNOMAD architecture is through a database. The database provides the support for storing and sharing tuples. Similar in concept to the LINDA model of communication ([?]), expert systems may attempt to retrieve a tuple out of the database as well as assert or remove one. This mechanism supports cooperation among expert systems without having to explicitly model one expert system in another in order to communicate with it. The LINDA model also provides an abstraction that hides the actual mechanisms of data sharing, whether they be message passing or shared memory (see Leler [?] for an example of this).

The database provides for multiple expert systems residing in one instantiation of KNOMAD or in multiple instantiations of KNOMAD on different machines to communicate. When there are multiple instantiations of KNOMAD on different machines, all that needs to

be done is to make the database distributed. What this entails is that when an expert system is added to the environment, the interface to the distributed database must be notified of what data the new expert system is interested in so that that data may be made available to the expert system when it exists. The KNOMAD system isn't concerned with the issues of distributed databases per se; currently, it is planned that an off-the-shelf distributed database may be plugged in and used. However, there are still issues of data integrity when more than one user of the data operates on the data at the same time, as in the KNOMAD architecture. The current system provides a minimal level of support by providing locks in the database so that one expert system may lock the data it is using in order to update it in an atomic fashion.

The database provides the mechanism for sharing and storing data. It doesn't provide much in the way of data abstraction and isn't intended to. It is the role of the database interface to maintain and abstract the connection between the data of the database and the inference mechanisms of KNOMAD. The interface layer provides two major functions. One function it provides is to present the data in all its richness. That is, to provide data abstraction and inheritance as in a frame system. Other views of data may also be present at this layer, for example, temporal aspects of data, uncertainty in data, and non-monotonicity. The second function the interface layer provides is that of a notification function (similar to a blackboard). As data is asserted to the database, any clients that reference that data are notified of the change. The database interface provides the necessary link between data and the ability to notify the clients in a data-driven manner.

As an example of the abstraction on data the database interface provides a frames abstraction. Frames are an abstract organization of data into conceptual units ([?]). A frame may have any number of slots. Frames may be defined as children of multiple parents (making inheritance potentially more complex) and may also have code attached to them that is executed whenever a new instance of the frame or one of its children is created. In this implementation of a frame system, slots have six optional aspects that help determine how frame accesses will be made. The most used aspect is the `:value` aspect. This aspect is where a value for the slot is located. The `:if-needed` aspect is used to store code that is executed if a slot value is asked for. The `:if-added` aspect is used to store code that is executed whenever the slot gets a new value. There are two aspects that are used to constrain the value of the slot. The `:constraint` aspect is used to store code that checks if an added value passes the constraint. This is user-defined code that must return a true or false status indicating the result of passing the constraint, but may otherwise have unpredictable side-effects. The `:mustbe` aspect constrains the value to be one of a list of formal values or frames. Finally, the `:distribution` aspect is used to determine if the value of the slot is for global distribution (accessible to all knowledge agents), or only for local usage. The only purpose of the distribution aspect is to provide data that enables extra efficiency in the database. If a slot is labeled with a local distribution, there is no reason to maintain the slot's value in a distributed fashion.

Supposing that Clyde is an elephant and that we know that elephants have four feet and are grey as well as weighing two tons, we would see the following tuples in the database:

```
(frame elephant num-feet :value four)
(frame elephant weight :value 2-tons)
(frame elephant color :value grey)
(frame Clyde :parents :value (elephant))
```

Thus, if one expert system asserts the information about Clyde and elephants and subsequently notifies another expert system that it ought to find out about the weight of Clyde, then the second expert system can ask for the tuple: (frame Clyde weight :value \$value). The database interface is smart enough to notice that a frame access is being made and since this tuple does not exist directly to try to inherit it by finding out who the parents of ClydeXS are.

The second function of the database interface is to provide a linkage between the inference mechanisms of the top layer of KNOMAD and the database of the bottom layer. The linkage provides the mechanism whereby the inference mechanisms may operate in a data-driven fashion and thus operate efficiently. This function of the database interface is discussed in the next section.

4 Providing Efficiency

Efficiency of expert systems in the KNOMAD architecture is provided by parsing rules and rule groups of an expert system into a data network of database accesses. The database interface manages the data network and notifies the expert system when a rule becomes activated.

To understand this better, the syntax of a rule needs to be explained. A rule in KNOMAD obeys the following syntax (a BNF-like syntax where braces define options and brackets are part of the language syntax):

```
rule ::= { condition } ::> condition |
       { condition }
       quantifier
       { WHERE condition }
       < rule >

quantifier ::= FOR ALL ID IN expr { ID IN expr }* |
             THERE EXISTS { NUMBER } ID IN expr
             { { NUMBER } ID IN expr }*

condition ::= selector+ { choice selector+ }
```

```

choice ::= EXCPT | OR

selector ::= [ quantifier { WHERE condition }
              < condition > } ] |
             [ referee { relation reference } ]

referee ::= expr

relation ::= NOT MEMBERIN | NOT UNIQUEIN | MEMBERIN | UNIQUEIN |
           = | <> | >= | <= | > | <

reference ::= expr

expr ::= message | constant | path | expr op expr

constant ::= STRING | NUMBER

op ::= PLUS | MINUS | UNION | TIMES | IDIV | RDIV | MOD

path ::= KEY | KEY OF path

message ::= path :: KEY { ( expr+ ) }

```

It is very straightforward to parse a non-quantified rule into a data network. All the necessary referee and reference patterns on the LHS of the rule, together with the reference patterns on the RHS of the rule are linked with a rule-node. As a pattern gets asserted to the database, the database interface notes if the pattern activates any of the patterns linked to a rule node. If so, the rule node is then notified. The rule node then checks all its links to see if they are all activated, if they all have values. If they are all activated, the rule associated with this rule node is then activated; the rule is queued on an activation queue of the expert system for subsequent firing.

This method of parsing rules into a data network based on database patterns allows links for rules to be set up so that rules may be used in either a forward or backward chaining method. The RETE network, on the other hand, is only implemented as a forward chaining network ([?]). However, the method used here does have a potential drawback. Rules may be activated when all the necessary patterns have values, but the values do not need to be correct. Therefore a rule still needs to be evaluated before it may be interpreted.

Quantified rules are more difficult. The patterns a rule is being quantified over may be used as links to a rule node in the data network. However, reference and referee patterns that depend upon values for the variable being quantified over cannot be used as links. Values for these patterns must be found dynamically by evaluating the rule after it has been activated. Patterns in the sub-rule of a quantified rule that do not depend on a quantified variable may be linked into the data network similar to the non-quantified rule.

The RETE network is much faster but not as general. The method in KNOMAD allows

rules to be linked in the data network in a completely non-directional fashion. In the current system, links are made for both the LHS and RHS of a rule. If a rule is activated by LHS links, the rule is put on a LHS activation queue for forward chaining paradigms. If the RHS, it is put on a RHS activation queue for backward chaining paradigms. This method also restricts quantification by effectively only allowing typed quantification on a rule. Therefore, when a quantified rule is finally activated, its scope of activation is very well defined and allows for rich semantics in the quantified rules (as shown in the syntax above, universal quantification as well as a variety of mechanisms for existential quantification).

5 The Expert System Paradigm

Generality in the architecture and the rule system of KNOMAD is provided by a number of mechanisms. One of these is the syntax provided for expressing rules in the rule system. Another is the abstraction provided in the database interface. The third mechanism that the rule system provides is the paradigm for performing inference over an expert system.

In the expert system paradigm of KNOMAD's rule system, an expert system consists of a number of rule groups. These rule groups may execute in parallel or execute from one another. There must always be at least one rule group of an expert system that begins execution. Each rule group may be thought of as executing a match-select-fire cycle. The KNOMAD architecture eliminates most of the match phase by queueing rules that are activated. These activated rules must still be evaluated, but the activation set is much smaller and a direct result of changes in assertions in the database.

The expert system paradigm in KNOMAD first modifies the cycle slightly by introducing a phase before the match phase. This phase is the control phase. Georgeff ([?]) defines a mechanism whereby a regular expression can be used to define a level of determinism in the rules of the rule group. This rule control is necessary in any complex problem. Expert systems are nice for their maintainability and declarative nature. Complex, real world problems, however, have aspects that are very procedural and must be controllable. What Georgeff has asserted is that the necessary control can be stated in a regular expression that defines the next set of rules that are *matchable* based on what rules have fired before. This makes the control explicit that is otherwise embedded in the rules themselves; making maintenance of the rule group, and thus the expert system, much simpler. The control in the rule system of KNOMAD uses a transition table (directly derivable from a regular expression) instead for clarity.

Once the control phase determines those rules that really may exist in the activation set on a particular cycle, the control strategy of the rule group is executed. The control strategy is a user definable algorithm that examines the rules in the activation queue and computes a conflict set. This may be a simple forward chaining algorithm that uses the rules on the LHS activation queue, or it may be a backward chaining algorithm, or even some unique

combination. When the conflict set is computed, the conflict resolution algorithm of the rule group is invoked (also user definable) to compute the fire set of the rule group. When the fire set is computed, all the rules in it are interpreted. The cycle then repeats.

This paradigm allows user or programmatic control to be inserted into a rule group in a very flexible manner. As most complex problems require specialized inferencing as well as the declarative knowledge represented in the rules, this mechanism can become very useful. Traditionally, control would be added directly in the rules themselves to control the amount of determinacy in the rules as well as the inference mechanisms represented by the control strategy and the conflict resolution strategies of the rule group. The paradigm presented here allows the user unrestricted access to these mechanisms. Furthermore, the control and conflict resolution strategies may be declared to be other rule groups which perform the necessary algorithm. This is a fairly inefficient approach when speed is important, but allows experimentation on different inference strategies for the expert system.

6 Discussion

The three issues of cooperating expert systems mentioned in the introduction and described further in the later sections are very relevant to a useful architecture for solving complex problems. The issue of communication between expert systems has been dealt with by defining a distributed database layer upon which expert systems share data. There currently exist no systems which can support distributed expert systems, residing on different machines, as a homogeneously managed unit. Lander ([?]) describes a system similar to KNOMAD but defined upon GBB ([?]) instead of a generic database or distributed database layer.

The database interface layer is also unique for cooperating expert systems. It behaves somewhat like a blackboard by notifying the various systems of the top layer of KNOMAD when something of interest has happened. Conversely, it is also very different from a blackboard system. It allows various types of data abstraction to be defined on data. Currently only a frames abstraction exists. Part of the process of abstracting data is that aspects of the abstraction must be recorded in the database as well as the data being abstracted. In this manner, other expert systems may be able to use the abstracted data without having to know beforehand how the abstraction was built (e.g., inheritance information). The database interface layer is the glue between data and the systems of the top layer of KNOMAD. It provides a non-directional data network mechanism to support whatever inferences might be necessary by a top layer system. In the rule system, both forward and backward chaining may be defined without having to set up special mechanisms. The database interface layer provides the necessary support.

Finally, the top layer of KNOMAD provides a conceptual glue where multiple inference mechanisms reside that may operate on the same data. In this fashion, multiple cooperating knowledge agents may easily be defined that work together (a generality of multiple cooper-

ating expert systems). An example is a complex power system where various fault diagnosis strategies need to be tested on various system elements. A particular fault diagnosis strategy might be implemented as an expert system. A power system would be modelled by using the frames abstraction of the database interface layer. The power system would have to be simulated by using both a constraint system and qualitative reasoning to determine when a switch might trip. In summary, a variety of inference mechanisms are necessary to solve complex problems in an efficient manner.

7 Conclusions and Future Work

The KNOMAD architecture has been implemented and is being used to autonomously manage a complex space station-like power system in the presence of faults. The Space Station Module/Power Management And Distribution (SSM/PMAD) project consists of a set of distributed processors for managing load centers and power distribution control units, a system for managing and recovering from faults, and a scheduling system, all operating autonomously. The overall goal is to define a system that schedules activities for execution, turn switches on and off to enable the activities, detect and recover from faults by safing parts of the system, performing fault isolation and diagnosis, and rescheduling. See Riedesel, et. al. ([?]) for a description of the SSM/PMAD system.

KNOMAD has been very successful to date. The rule system of KNOMAD has been tested on three domains. The first is the system described above where the fault diagnosis expert system consists of three rule groups and over 160 rules. The other two domains are the monkeys and bananas problem and one other toy problem. The toy problem was augmented by defining an additional rule group to perform the control strategy algorithm.

We have been very pleased with the performance of KNOMAD. It would be of interest to compare it with a shell such as OPS5, however, the goals of the two systems are not similar enough to allow a fair comparison—the languages and the architecture are completely different. If KNOMAD were to be compared with systems such as ART and KEE, the differences would be in two places: the database mechanism for sharing data between cooperating expert systems and the flexibility of the execution of a rule group.

Future work includes incorporating temporal knowledge into the database interface in some form, perhaps using a temporal database mechanism as Dean has accomplished ([?, ?]). A constraint system will also be added. These two items, temporal reasoning and the constraint system, will then be used to support the building of a planning system.

Work will also be done to look into the possibility of incorporating RETE-like network structures into the data network during syntactic analysis of rules. This is certainly possible where complete matches can be determined beforehand by syntactic analysis, but some aspects of quantification in KNOMAD seem to allow only partial matches of the sort already existing in the database interface layer.

Finally, work is also planned for defining and implementing a distributed database. This work will have to look at the issues of locking more deeply as well as deadlocks, crash recovery, rollbacks, etc. Hopefully, we will be able to use results from existing research in distributed databases without reinventing the wheel.

8 Acknowledgements

This work was performed by Martin Marietta Astronautics Group under contract number NAS8-36433 to NASA, George C. Marshall Space Flight Center, Huntsville, Alabama.

References

- [1] Nicholas Carriero and David Gelernter. Linda in context. *Communications of the ACM*, 32(4), 1989.
- [2] Daniel D. Corkill, Kevin Q. Gallagher, and Kelly E. Murray. A generic blackboard development system. In *Proceedings of the National Conference on Artificial Intelligence*, 1986.
- [3] Thomas Dean. Using temporal hierarchies to efficiently maintain large temporal databases. *Journal of the Association for Computing Machinery*, 36(4), 1989.
- [4] Thomas L. Dean and Drew V. McDermott. Temporal data base management. *Artificial Intelligence*, 32(1), 1987.
- [5] Charles L. Forgy. Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, 19(1), 1982.
- [6] M.P. Georgeff. Procedural control in production systems. *Artificial Intelligence*, 18:175-201, 1982.
- [7] Susan Lander and Victor R. Lesser. A framework for the integration of cooperative knowledge-based systems. In *Proceedings of the IEEE International Symposium on Intelligent Control 1989*, 1989.
- [8] Wm Leler. Linda meets unix. *IEEE Computer*, 23(2):43-54, February 1990.
- [9] M. Minsky. A framework for representing knowledge. In P. Winston, editor, *The Psychology of Computer Vision*, pages 211-277. McGraw-Hill, 1975.

- [10] Joel D. Riedesel. Knowledge management: An abstraction of knowledge base and database management systems. In *Proceedings of the Fifth Annual AI Systems in Government Conference*, 1990.
- [11] Joel D. Riedesel, Chris Myers, and Barry Ashworth. Intelligent space power automation. In *Proceedings of the Fourth IEEE International Symposium on Intelligent Control*, 1989.

(This Page Intentionally Left Blank)

Appendix B

Knomad Reference Manual

B.1 Introduction

KNOMAD is a Knowledge Management and Design system that may be used for integrating and applying advanced computer programming technology to an application problem. KNOMAD currently provides support for data-driven and goal-driven reasoning (forward chaining and backward chaining rules, respectively), a constraint system with support for multiple contexts, and an advanced frame representation language for describing and reasoning about objects of interest. KNOMAD is organized on top of a database layer (not persistent) that may be distributed if the application requires it. KNOMAD is written in Common LISP [Jr.89] and is easily interfaced with conventional LISP programs.

This reference manual will describe how to use KNOMAD in some detail. Asserting and retrieving data to and from the database is fundamental to any application and will be described in the section on the database. This will include aspects of the database such as views and database constraints. The frame representation language will also be described. The semantics of assertions and retrievals will be defined. Database locks and the initialization of the database will wrap up the database section.

The distributed database will be described in the section three. This will cover its initialization, how to use it, how multiple database nodes should utilize it, and how it is integrated into the previously defined database.

The tools provided by KNOMAD will be described in section four. The constraint system, backward, and forward chaining will be described in detail, including their syntax and semantics. Section five will provide additional tools available to developers utilizing KNOMAD. These include queues, events, semaphores, and a clock abstraction. Finally, section six will provide a few comments on object-oriented programming in KNOMAD.

B.2 The Database Management System

The database of KNOMAD provides an efficient mechanism for the storage and retrieval of tuples. KNOMAD uses two particular types of tuples, facts and frames, to be discussed shortly. However, it is possible to use the database mechanism to store arbitrary tuples using primitives of KNOMAD. Tuples will be discussed first including database views (a logical storage area for tuples). Following this will be a description of primitive functions that are available.

The database also allows for constraints to be placed upon tuples. A constraint specifies restrictions on what tuples are allowed in the database. Constraints will be discussed after the primitive functions.

The next two subsections will describe facts and frames (the substance of KNOMAD), and assertions and retrievals. Following that, this section will conclude with a description of database locks and how to initialize the database.

B.2.1 Tuples and Views

The database supports the storage and retrieval of tuples. A tuple is an ordered sequence of (possibly) typed fields. For example, the following are all tuples:

```
(color car-324 white)
(fact fact-23 john)
(car-23)
(eat restaurant-34 joel mud-pie-587 june-23-1990)
(parent (son jane) tarzan)
(if (and cloudy wet) raining)
```

As can be seen, tuples may be arbitrarily complex. Tuples can represent records, as in a relational database, as well as logical statements (as in the last example).

Tuples are the basic item that is stored and retrieved in the KNOMAD database. The reason being that most knowledge based applications are interested in small statements and not large sets of relationships as in typical commercial applications. This does not mean that complicated relational structures cannot be easily represented. Frames, described later, are an example of complex relational and object-oriented structures using the tuple representation.

Each field of the tuple may be typed. For example, if the first field is `parent`, it may be that the final two fields must be people. These types of constraints may be represented using the constraint system of the database.

A further abstraction provided by the database system are views. In the traditional database world a view provides a restricted access into the database. For example, if a record of employees with fields of salary, labor grade, age, project, and address is in the

database a view of the record for normal database users may only allow access to all the fields except for the salary field. Thus the traditional view provides restricted access to a relation. In KNOMAD a view is quite different (and perhaps ought to be called something else). A view provides a distinct database for storing data. In effect, each view in KNOMAD is a separate database. This can be quite useful for storing related information about tuples in different databases.

`database:*view*` [Variable]

This is the default view that is used by the database for storing and retrieving tuples. It is not a good idea to rebind it.

`database:view-init view` [Function]

`dnet:dnet-init view` [Function]

`view-init` is used to make a new view. *view* should be a symbol that will be bound to the new view. `dnet-init` is used to initialize a particular view.¹ When the user initializes the database, the user should also use `dnet-init` to initialize the views created.

Primitive Database Functions

In the following functions, *who* should be a unique identifier. It is possible for more than one process to be performing database operations at the same time. If one process wants to lock an item it is necessary to identify the processes with unique identifiers. At the higher levels this is hidden from the user by defaulting *who* to the value of (`mp:process-name *current-process*`).

`dnet:index tuple value who view` [Function]

This function will store the given *value* under the index *tuple* in view *view*.

`dnet:fetch tuple view` [Function]

This function will retrieve the value stored under the index *tuple* in view *view*.

`dnet:dnet-remove tuple who view` [Function]

This function will remove the value stored under index *tuple* in view *view*. Subsequent fetches will not find any value under that index.

¹You guessed it. The database is implemented as a discrimination net. Each view is a distinct net.

`dnet:dnet-lock tuple who view` [Function]

This function will lock the value at the index *tuple* in view *view*. The lock will be owned by *who*. Any attempts at storing a new value there by another process will fail. Other processes may still read the value.

`dnet:dnet-unlock tuple who view` [Function]

This function will unlock the value stored at index *tuple* in view *view* in two cases: The current lock is owned by *who* or there is no lock—in which case nothing happens.

B.2.2 Database Constraints

Another aspect of database usage is the ability to add constraints to tuples in the database. Constraints in KNOMAD are used to specify and type fields of tuples.

`database:constraint constraint-pattern &optional code` [Function]

`constraint` is used to specify a constraint on certain tuples in the database. The *constraint-pattern* has the form (in BNF notation):

```
constraint-pattern ::= ( item+ ) | ID
item                ::= constraint-pattern | (:type-decl ID) | (:type-spec ID)
```

The constraint pattern matches a tuple in three ways. One, if the current field of the pattern is ID then the corresponding field of the tuple must be the same as ID. Two, if the current field of the pattern is (:type-spec ID) then the corresponding field must have the type as specified by ID. And three, if the current field of the pattern is (:type-decl ID) then all tuples with the corresponding field must have type ID.

Now that's confusing. To clarify this, the pattern will match all tuples based on the ID and (:type-spec ID) fields. For all those tuples matched, the fields with (:type-decl ID) are *constrained* to have the correct type or else are thrown out of the database (or not allowed to be stored).

So, for example, the constraint pattern (color (:type-spec t) (:type-decl color)) will match the earlier tuple (color car-324 white) and constrain it to have a color in the third field.

Constraints, as specified with this function, will be applied to all tuples as they are added to the database (regardless of view). If the constraint does not pass the tuple, the tuple will not be entered into the database and the storage operation will return :fail. When a new constraint is added, all the existing facts that match the constraint (only in the default view

`*view*`) and do not pass will be returned by the constraint function.² Finally, an optional argument *code* may be provided. If this is provided, the code will be funcalled on tuples as they are added to the database. The code should return a boolean value indicating pass or fail.

Constraint Caveats

The database constraints and the rest of the KNOMAD database functions have not evolved at the same rate. The constraints will likely not work well on fact and frame tuples and especially other types of tuples. It is currently recommended that one does not use constraints. However, if sufficient demand is made, support for them can be made available.

B.2.3 Facts and Frames

The higher levels of the KNOMAD database are configured to represent facts and frames (as opposed to arbitrary tuples). Facts are simple tuples of the form (fact <fact> :value <value>). This simply says that <fact> has value <value>.

Frames are very similar to objects in object oriented programming but are more dynamic. Frames allow one to store knowledge in the form of a fixed set of slots (representing the frame) with fillers that can vary from frame to frame. This is like a basic record with fields. However, each slot has structure as well.

Every slot has one or more aspects. There are five aspects defined here: *value*, *mustbe*, *if-needed*, *if-added*, and *constraint*. The *value* aspect represents the value of the slot if it has one. The *if-needed* aspect, if present, provides a LISP procedure for returning a value for the slot if the slot does not have a *value* aspect. The *if-added* aspect, if present, is a list of functions that are executed whenever a slot is written to, that is, whenever a value is added to the slot (or changed). The *constraint* aspect, if present, is a list of lisp predicates that must all be satisfied for a value to constitute a valid filler. The *mustbe* aspect, if present, gives frames or atoms that fillers must either be an instance of, or equal to. The *mustbe* aspect can either be a single such item, or a list of [atom|frame]*, where only one item in the list must be matched (representing disjunction).

The *if-needed*, *if-added*, and *constraint* code are all funcalled with the current tuple as the only argument (e.g. (frame <frame> <slot> :value <value>)). Other aspects than those described above are legal, but have no special meaning to the frame system. All code is run in an environment where the special variable *self* is bound to the frame itself.³

Multiple parents for a frame are defined. The slot aspects for a slot are inherited in a depth-first left to right search through the list of parents of the frame.

²It is therefore suggested that the user of database constraints define all the constraints before asserting data into the database.

³This is not really useful since the code being called is passed the entire tuple anyway.

An `isnew` method can be defined for frames that executes some LISP code when a frame is instantiated. The `isnew` code is not passed any variables and is simply evaluated (not funcalled). The set of `isnews` collected from all parent frames is run when creating a new frame.

Frames can be defined as follows:

```
(frame :name name
      :parents parent | ( parent* )
      :isnew code
      :slots ({(slot aspect value {aspectvalue})*})
      )
```

```
aspect ::= :if-needed | :constraint | :if-added | :value | :mustbe | :ddb
```

```
mustbe ::= atom | frame-name | (atom* frame-name*)
```

```
frames:frame &key :name :parents :isnew :slots [Macro]
```

```
frames:fcreate-instance parents name [Function]
```

The `frame` macro allows the user to define a new frame with the parameters as discussed earlier. Creating a frame using the `frame` macro will also define a predicate and a `make` function for making new instances of the frame. The predicate will be the name of the frame appended with “-p.” The predicate takes one argument and returns a non-nil value if the argument is of the defined frame type. The `make` function will be the name of the frame prepended with “make-.” It is described here:

```
make-<frame> &optional name &key <slot1> <slotn> [Function]
```

This function will make a frame of the define type (<frame>). If the optional *name* argument is not provided a new name will be created and returned. The keyword arguments allow for initializing the values of any of the slots of this frame instance.

`fcreate-instance` is used to instantiate a new frame. This takes two parameters *parents*, and *name*. *parents* must be either then name of a list of the names of previously defined frames.

When an instance of a frame is created, that instance is considered to be just as much a frame for purposes of inheritance as those frames created using the `frame` macro. This means that instances may be referenced as parents in other frame definitions.

The following LISP session shows an example of defining a frame, creating an instance and asserting a value (more on assertions later).

```
lisp> (defun sick-dog-test (tuple)
      (if (eq (fifth tuple) 'dry)
          (format t "%~a is sick!" (second tuple))))
```

SICK-DOG-TEST

```
lisp> (frame :name dog :parents ()
          :slots ((skin :value furry)
                 (nose :value wet))
          :isnew (format t "%Creating a dog frame!"))
```

Creating a dog frame!

T

```
lisp> (assert! '(frame dog nose :if-added ,(list #'sick-dog-test)))
```

```
(FRAME DOG NOSE :IF-ADDED (#<Interpreted Function SICK-DOG-TEST @ #xbce006>))
lisp> (make-dog 'gandalf)
```

Creating a dog frame!

GANDALF

```
lisp> (assert! '(frame gandalf nose :value dry))
```

GANDALF is sick!

```
(FRAME GANDALF NOSE :VALUE DRY)
```

```
lisp>
```

Miscellaneous Frame Functions

`frames:grind-frame frame` [Function]

This function can be used to describe a frame, its children, and the slots and their values.

`frames:frame? frame` [Function]

If *frame* is a frame, then this function will return a non-nil value.

`frames:frame-isa? child frame` [Function]

Returns `t` if *child* is a child frame of *frame*. Otherwise it returns nil. If *child* and *frame* are the same, this will also return `t`.

`frames:collect-aspect frame slot aspect &optional view` [Function]

This function will return a list of tuples with values over the inheritance graph of *frame* under slot *slot* and aspect *aspect*. The list of inherited tuples will be in depth-first, left to

right inheritance order. This function may be useful if one would like to define a new aspect on frame for use by some tool (the backward chaining tool uses this feature).

`frames:frames-who-inherit? frame slot aspect &optional view` [Function]

This function returns a list of all those children of *frame* who don't have a value on slot *slot* and aspect *aspect*. This is useful so that one can determine who would inherit a value just asserted to some aspect on a slot of a frame.

`frames:remove-frame frame` [Function]

This function removes the given frame and all its children from the database.

B.2.4 Assertions and Retrievals

Now that the preliminaries (tuples, facts, and frames) have been discussed asserting and retrieving tuples from the database can be described. There are five assertion and retrieval operations available: `assert!`, `remove!`, `match!`, `retrieve!`, and `wait!`. These functions are defined for facts and frames but not for the general tuple.

`dbif:assert! tuple &optional who view` [Function]

tuple may be a fact or a frame and must be quoted (this is a function). The `assert!` operation uses a primitive database function to actually store the pattern represented by *fact*. If for some reason the pattern cannot be stored because of a failed constraint `:fail` will be returned, otherwise the asserted fact will be returned.

The process performing the assertion is represented by *who* and is used for purposes of locking mechanisms in the database. If the tuple being updated is locked, only the process owning the lock may change its value. *view* is used to determine what view to use for the database storage. And *view* defaults to `*view*`.

Another primary function of `assert!` is to perform data-driven processing on tuples. Any if-added on frames will be run at this time. The forward chaining rule system is also triggered if relevant tuples are asserted (currently this is all hard-coded, it would certainly be a lot nicer if if-added were used).

`dbif:assert-eq! tuple` [Function]

By default, the `assert!` operation uses `equal` to determine if the new value is different from what is already in the database. By using `assert-eq!` the `eq` test will be used instead.

`dbif:assert-eql! tuple` [Function]

The `assert-eql!` operation is exactly like `assert-eq!` except that it uses the `eql` test.

`dbif:remove! tuple &optional who view` [Function]

The `remove!` function is used to remove a fact or frame from the database. If the pattern represented by `tuple` is locked then `who` must be the locker of the tuple, otherwise the tuple will not be removed.

`dbif:match! pattern &optional view` [Function]

The `match!` function takes a pattern with variables in it and returns a list of tuples in the database that match the pattern. A variable in the pattern is simply a symbol that starts with "\$." If there are two variables in the pattern and they are the same, then the value they match must be identical in both places (that is, they must be unifiable).

`dbif:retrieve! pattern &optional view` [Function]

The `retrieve!` function takes a pattern that may not contain variables in it and returns the tuple in the database that is identical to it, if there is one.

`dbif:wait! pattern &optional who view` [Function]

The `wait!` function is exactly like `match!` except that if there are no matching tuples in the database, the calling process will be blocked until a matching tuple is asserted (by some other process).

B.2.5 Locks

Locks are used for performing updates to tuples in the database. Locks are very simple to use, one simply locks a tuple, reads and updates the value, and unlocks it.

`dbif:lock pattern &optional who view` [Function]

`dbif:unlock pattern &optional who view` [Function]

The `pattern` argument may be any tuple pattern with variables as in the `match!` function described above. This way the user may lock and unlock multiple related tuples. `who` represents the process doing the locking and unlocking.

B.2.6 Initialization

Initialization of the KNOMAD database is done by using the initialization function. Initializing the database will completely initialize it. If the user has any other views they will also have to be reinitialized as described earlier.

`dbif:initialize!`

[Function]

Simply initializes the database.

B.3 The Distributed Database

The distributed database of KNOMAD provides for the ability to share and distribute the same data among multiple computer processors. For the most part, this is completely invisible to the user. However, KNOMAD's default configuration is not to use the distributed database.

To enable the use of the distributed database, one should set `user:*use-ddb*` to `t` in LISP. Then you will need to reinitialize `knomad` by using the `initialize-knomad` function. This will initialize the KNOMAD system to use the distributed database. When using the distributed database it is a good idea to enter the form `(ddb:signoff)` to LISP when you wish to halt a KNOMAD system running with the distributed database. This tells all the other nodes in the distributed database that a node is leaving, they will then no longer use that node for further updates to the database.

There are many restrictions to the distributed database in KNOMAD. These arise from the design of KNOMAD. KNOMAD was initially designed without the idea of adding a distributed database later. It has turned out to be very difficult to support it effectively. It is also likely that performance may be relatively slow when using the distributed database. This has not been fully evaluated yet, so if anyone does start using it for a real application, be sure to let me know how it goes.

B.3.1 Using the Distributed Database

Using the distributed database is almost no different than not using it. The user will still perform assertions and retrievals as in the non-distributed KNOMAD database. However, the user must tell KNOMAD what is to be distributed (by default, nothing except frame definitions is distributed).

It is only possible to distribute frame slot values. Other aspects of slots of the frame system are not distributed. The frame definitions are always distributed (one cannot therefore define two different frames with the same name on different parts of the distributed database). The question arises as to how if-added's are run and constraints checked when using the distributed database. The distributed database is smart enough to check constraints on other machines when performing an update. If a some constraint is not met, the update will not get performed. Similarly, if-added's are run when the update actually goes through. The reasoning is that if-added's are methods that are local to some process as opposed to global throughout the entire database. It only makes sense to distribute data, not methods.

To tell KNOMAD that a slot value should be distributed the user must define a frame with a new aspect in the slot to be distributed. This aspect is the `:ddb` aspect. By default, the

value of the `:ddb` aspect is `:local`. If the user defines the frame slot with the value `:global` for the `:ddb` aspect, that slot value will now be distributed throughout the database. For example:

```
(frame :name dog
      :slots ((color :value black :ddb :global)
             (age :value 2 :ddb :global)
             (int-slot :value foo)))
```

In this example, both the `color` and `age` slots will be distributed, while the `int-slot` will not.

Frame slot values are the *only* items in the database that may be distributed. Facts may not be, and neither can arbitrary tuples (if using the primitive database functions). This is one restriction that is a result of the initial design of KNOMAD.

When defining a slot to be distributed it is very important that only simple data be asserted to the slot. Data that the LISP reader can read. Therefore, arbitrary functions, for example, should not be asserted. If non-readable data is asserted to a slot, other nodes of the distributed database will get an error, and the distributed database itself will no longer operate.

One other thing to keep in mind is that it is not possible to use the `lock!` and `unlock!` functions on distributed frame slot values. Don't do it. It won't work and may break the distributed database.

These restrictions sound very limiting, but it is still possible to gain a lot of benefits. By careful design it would be possible to distribute many complex aspects of a distributed processing system, communicating simply with readable data. This is very much like the LINDA model of parallel processing.[CG89]

B.3.2 Locking and Unlocking Concepts

Since it is not possible to use the normal locking and unlocking mechanisms of KNOMAD to gain exclusive control in a distributed system using the distributed database of KNOMAD, one may wonder how to get this control using other mechanisms.

This is very simple really. The idea being to have a frame slot in the distributed database that is used to control the interaction among multiple processes in the distributed environment. One way to approach this is to define a frame with many locks, each used for some different purpose:

```
(frame :name locks
      :slots ((lock1 :value :unlocked :ddb :global)
             (lock2 :value :unlocked :ddb :global))
```

To use this a process will attempt to lock out other processes by asserting a different value to the particular lock he is interested in. But, before doing so, the process needs to make sure that the lock is not currently locked. Here is some example LISP code that will grab the lock as soon as possible:

```
(block got-lock
  (let (result)
    (loop (wait! '(frame locks lock1 :value :unlocked))
      (when (not (eq :fail
                     (assert! '(frame locks lock1 :value :foo))))
        (return-from got-lock t))))))
```

If all processes that need exclusive control over some region of code use this mechanism for grabbing a lock before entering their critical region, then they are guaranteed exclusive access.⁴ It does not matter what value a process tries to grab the lock with (unless they are both on the same node of the distributed database). If two processes (on different nodes) try to assert the same value, only one of them will succeed. Updates are ordered in a system dependent manner, and only one process can succeed in updating a value at a time. If two processes need exclusive access via a lock and reside on the same distributed database node, they should use a different means to perform their locking.

To release the lock a process simply needs to assert `:unlocked` back into the value of the lock: `(assert! '(frame locks lock1 :value :unlocked))`.

B.3.3 Error Checking

The KNOMAD distributed database is a prototype system that was carefully and sometimes with many kludges, shoehorned into the KNOMAD database design. It is not a robust system. It does utilize RPC on top of TCP streams, and therefore will not encounter loss of data on the network. However, if one node goes down without executing the `ddb:signoff` function, the distributed database will not figure this out. This type of error checking has not been implemented in the database (timeouts, etc.). It was felt that since the distributed database does not match well with the existing KNOMAD implementation, that it was not worthwhile to expend too much effort to make a robust system. This should properly be done with a new design from the ground up (including KNOMAD).

Another point that is probably worth mentioning here is that there is some strange interaction between LISP and RPC. When you start a new LISP running, you can initialize KNOMAD using the distributed database. If you later exit the distributed database (`signoff`) without exiting LISP and reinitialize KNOMAD again (to join the distributed database), other

⁴The proof is obvious by inspection :-).

machines will not be able to perceive that you really exist in the network. The RPC functions do not seem to accept requests over the network. The temporary solution to this is to start a fresh LISP every time you want to initialize KNOMAD using the distributed database. The caveat is: Be careful.

B.4 KNOMAD Tools

Three KNOMAD tools will be described here. These are the Rule Management System (for forward chaining), the Backward Chaining system, and the Constraint System. It is important to utilize these tools independently with one another. Using backward chaining to find a value does not automatically cause any forward chaining rules to trigger off of that value.⁵ Similarly, using the constraint system to derive legal values for variables does not assert anything to the database automatically that could be used by either the forward or backward chaining systems.

B.4.1 The Rule Management System

This subsection describes how to use the rule management system tool of the KNOMAD system. It assumes that the reader has some knowledge of expert systems and forward chaining. The mechanism for specifying a knowledge base is first presented. This is followed by a description of the rule group. The rule group methods are then discussed including the execution strategy, the control strategy, and the conflict resolution strategy. Finally the semantics of rules are described. For more technical information about the implementation and perhaps clarification of the discussion here see [Rie90].

The Knowledge Base

Adding a knowledge base to the KNOMAD rule management system must obey fairly rigid syntax. A knowledge base consists of declaring the name of the knowledge base, optionally specifying a domain file that defines the domain, the rule groups of the knowledge base, optional further domain knowledge specific to the knowledge base, which rule groups should begin execution, and finally an end symbol. the exact syntax is defined in the KNOMAD BNF Syntax appendix. The knowledge base generally looks like:

```
@@ Comments are given by using a '@' symbol. Everything to the end of
@@ line is regarded as comment.
```

```
@@ <kb-name> is simply a symbol representing the name of the knowledge base.
```

⁵This is an incompleteness in many expert system tools with both forward and backward chaining.

KB : <kb-name>

@@ The DOMAIN is optional, but if given, <domain file name> is a filename
@@ that will be loaded to load the domain. The file is loaded as a
@@ LISP file would be loaded using the load function.

DOMAIN : <domain file name>

@@ Rule groups may be directly specified by inserting them directly in the
@@ knowledge base definition. They may also be modularized by placing them
@@ in a separate file in which case the FILE keyword is used.

<rule group>

FILE : <rule group file name>

<rule group>

@@ Domain knowledge is optional and is used to specify further domain
@@ knowledge that is not appropriate in the domain file, but particular
@@ to the knowledge base. Domain knowledge consists of constants, facts,
@@ and frames. Constants are specified by specifying symbols separated
@@ by semicolons. An example constant is 'true'. Defining a constant
@@ asserts a fact into the database that specifies that the constant's
@@ value is the same as the constant. The constant 'true' would be
@@ asserted as: (fact true :value true).

@@ Facts are used to specify values for particular facts. Each fact
@@ is separated by a semicolon. An example fact could be:

@@ empty = ()

@@ This fact would be asserted into the database as: (fact empty :value ()).

@@ Finally, frames may also be specified. Frames are specified a little
@@ differently than constants and facts. To specify a frame simply put
@@ the LISP definitions for frames and their instances here. They are
@@ not separated by semicolons but they must be terminated by a period.

@@ The frames will be read and evaluated by LISP directly.

Domain-Knowledge :

constants : <constants> .

facts : <facts> .

frames : <frames> .

@@ Before the knowledge base is finished the rule groups that should
@@ be executed are specified. These are the names of the rule groups
@@ as in their specifications. None to any number may be executed
@@ in parallel for each knowledge base. If more than one are specified,

@@ they must all be on the same line as the begin statement.

Begin : <rule group names>

@@ Finally the end symbol is given.

End-KB

An operation specific caveat is that the rule management system loads knowledge bases into the rule-system package of LISP. This means that any functions to be used by the rule groups, symbols in the database, etc., all must be defined to exist (or be visible) in the rule-system package.

rule-system:parse-kb *file* [Function]

This function is used to parse a knowledge base. The *file* must specify the filename where the knowledge base is defined. This function will return the knowledge agent.

rule-system:load-ka *ka* [Function]

rule-system:kill-ka *ka* [Function]

load-ka is exactly analogous to parse-kb. It expects a symbol whose value is the name (pathname) of a knowledge base. It will simply call parse-kb.

kill-ka is used to kill a knowledge agent (which is simply a knowledge base). It is used to stop a knowledge agent from executing and remove any hooks it has in the database.

The Rule Group

The rule group is used to specify a set of rules that are related to one another in some fashion. The rule group consists of six basic parts as specified here:

@@ The first part of a rule group is its name.

RULE-GROUP : <rule group name>

@@ The rule group rules may be controlled using a transition graph

@@ of what rules follow what other rules. This control specification is

@@ optional.

CONTROL : <transition table>

@@ The control strategy for a rule group may also be specified here.

@@ If it is it should be the name of another rule group or of a function

@@ that will be used as the control strategy for this rule group.

CONTROL-STRATEGY :

@@ The conflict resolution strategy is exactly analogous to the control
@@ strategy and is optional as well.

CONFLICT-RESOLUTION-STRATEGY :

@@ Finally the rules are given. Each rule is separated by a semicolon.
@@ The rules will be described shortly.

<rules>

@@ After the last rule a colon, optionally followed by a termination
@@ condition is specified. The termination condition is a regular
@@ rule condition and when it evaluates to true, the rule group is
@@ considered to be finished.

: <termination condition>

@@ If the rule group is specified in a separate file from the knowledge base
@@ the following must also be in the rule group as the last symbol.

DONE

The semantics of executing a rule group are given in the next section on rule group methods. However, as an introduction, the basic execution cycle will be described here for context.

The idea behind a group of rules is to match those rules with data in the database, and from the rules that are matched, select some of them and fire them. This is termed the match-select-fire cycle. The match phase of expert systems is the most expensive. This involves matching the left hand sides (LHS) of rules with data in the database and finding those rules that are *satisfied*. From the resulting set, usually only one, but possibly more, rule(s) are selected. This is the select phase, also called conflict resolution. Finally, those rules that are selected are fired. Firing a rule means that the statements on its right hand side (RHS) are asserted into the database. This cycle continues until the termination condition (basically a LHS of a rule) is satisfied.

The rule group definition in KNOMAD provides a lot of flexibility to this model. Specifically, the user may specify how rules can be matched. In a generic expert system, rules match non-deterministically. To get around this most developers put statements into the rules themselves to sequence them properly. The control transition table allows a user to specify up front how rules should be sequenced. The user may also specify different control strategies and conflict resolution strategies for matching and selecting rules.

The basic building blocks of rules are selectors. A selector has the form [referee relation reference]. The referee and reference are used for specifying database values. For example a referee could be simply true or color of car-34. relation is used

to specify how the referee and reference are related to one another. For example, a relation could be = or <=. When a selector is on the LHS of a rule it is generally used to ask if the referee is related to the reference according to the relation. When a selector is on the RHS of a rule the referee is being given the value of the reference according to the relation (in this case the relation must be something that makes semantic sense, for example =).

Selectors may also be quantified. In a quantification, a symbol is bound to the result of an expression and is used in the condition being quantified over. For quantified selectors, if the selector is on the LHS of a rule, the selector is satisfied if the all the bindings are satisfied for the quantified condition in the case of universal quantification, and if there exists one binding which satisfies the quantified condition in the case of existential quantification. If the selector is on the RHS, then either one assertion is done non-deterministically for existential quantification or, for universal quantification, all assertions are done.

At the next layer are the conditions. A condition is most simply made up of a set of selectors. If on the LHS of a rule, all the selectors must be satisfied. If on the RHS, all the selectors are asserted. Conditions may also be represented by two sets of selectors. The second set either represents a disjunction with the first set or exceptions to the first set. If a disjunction is represented then it is only semantically valid on the LHS of a rule. When a set is used as exceptions then the first set must be satisfied and the *exception* set must not be satisfied for the condition to be satisfied.

The final layer is the rule. Rules may be either simple or complex. A simple rule has a LHS and a RHS, each of which is a condition. A complex rule may also have a LHS, but primarily involves the quantification of a variable over a nested rule. The quantification is stated similarly to that of selectors but occurs over a nested rule. The quantification and nested rule make up the RHS of a complex rule.

Rule Group Methods

The rule group is an object in the database and is specified as follows:

```
(frame :name rule-group
      :slots ((rules :value nil)
             (name)
             (window)
             (quantified-vars)
             (rg-var)
             (plan)
             (plan-state)
             (plan-table :value nil)
             (viable-set :value nil)
             (not-yet-viable-set :value nil)
             (fire-set :value nil))
```

```
(local-variables)
(rules-fired :value nil)
(conflict-set :value nil)
(tickle-set :value nil)
(satisfied-set :value nil)
(unsatisfied-set :value nil)
(cant-fire-set :value nil)
(fired-set :value nil)
(untickled-set :value nil)
(lhs-event)
(rhs-event)
(*lhs-tickled-queue* :value nil)
(*rhs-tickled-queue* :value nil)
(termination-condition :value nil)
(rules-with-dynamic-lhs-patterns :value nil)
(rules-with-dynamic-rhs-patterns :value nil)
(backtrack-stack :value nil)
(control-strategy :value #'default-control-strategy)
(conflict-resolution-strategy :value
#'default-conflict-resolution-strategy)
(execute :value #'execute1)
))
```

There are many slots to the rule-group definition. Not all of them are currently being used. However, different parts of the execution of the rule group use different slots. The control strategy and conflict resolution strategy are represented as slots on the rule group as well.

When a rule group is executed the following steps are carried out:

1. Initialize the rules of the rule group for execution.
2. Check the termination condition of the rule group. If the termination condition is satisfied the rule group is done.
3. Use the control transition table of the rule group to determine what rules can be matched next. If there is no transition table then all of the rules of the rule group are eligible.
4. Find out which rules of the eligible set are satisfied using the control strategy.
5. Use the conflict resolution strategy to pick a subset of the rules to fire.
6. Fire the selected rules.

7. Loop to number 2.

It is possible to specify a new control strategy or conflict resolution strategy using separate rule groups or user defined functions. However, it is not recommended that the user define a strategy using a rule group since it will most likely be an order of magnitude slower.

One other caveat of the rule group execution is how the rule management system and the database of KNOMAD are connected. When data is asserted in the database, if the data completes the satisfaction of the LHS of some rule, that rule will be put on the `*lhs-tickled-queue*` slot of the rule-group frame. This slot is then used by the rest of the execution methods to perform forward chaining.

Rule Semantics

The rule semantics will follow the construction in the BNF description given in the appendix. We will start from the selector and work our way up to the full-blown quantified rule.

Simple Selectors The general form of the simple selector consists of an expression in relationship to another expression. This is denoted: [referee relation reference]. If the selector is on the left hand side (LHS) of a rule, then it asks if referee is in relation to reference. Both the referee and reference must have a value for the selector to be true. On the right hand side (RHS) of a rule the selector will assert that referee will be in relation to reference. In most cases, the only legal relation for a selector on the RHS is “=.” The possible relations are given in the appendix.

It is also possible for a simple selector to consist of simply an expression, denoted: [expression] (or [referee]). On the LHS of a rule the selector evaluates to true if the expression evaluates to a non-nil value. On the RHS of a rule this selector is simply evaluated and the result is ignored.

What can we really do with this, what is an expression? An expression can be one of four things: `message`, `constant`, `path`, or `expr op expr`. The last is a compound expression. A `constant` can be either a string or a number, nothing else. A `path` is the mechanism for hooking the rule management system up with the database of KNOMAD. A path identifies a particular fact or frame whose value will be used when evaluating or interpreting the rule. The path can be a simple fact in which case one simply references it (e.g. `fact1`), or it can be a frame reference of arbitrary nesting (e.g. `color of car1`, or `type of refrigerator of kitchen of home of person45`).

A `message` is the means by which one can perform a LISP escape from a rule. A message consists of a path followed by “:” followed by an `id`. The path of a message should result in a frame of which the `id` references a slot. The value of that slot should then be a function which will be executed. Any arguments to the message (function) should be indicated with an expression for each argument, all wrapped in a single set of parenthesis.

The compound expression is made up of two expressions in a relationship to each other (op). The possible operations are indicated in the appendix.

To sum up, when a simple selector is on the LHS of a rule, it is true if all the references to variables exist and are in the indicated relationship to each other. If the simple selector is simply an expression not in relationship to anything, then that expression must evaluate to a non-nil value. What this means for expressions such as messages, is that the selector will not even attempt to be evaluated if the path to the function the message references does not have a value.

A simple selector on the RHS of a rule may currently only consist of two expressions with a relationship of = (indicating assignment, or an assertion into the database), or a single expression which will simply be evaluated. This is called *interpreting* the selector. A simple selector will not be interpreted if any variables it references in the reference do not have values.

Quantified Selectors A quantified selector is quite another beast. It's syntax is: [quantifier {ID IN expr}⁺ WHERE condition <condition>]. This says that one must first specify the type of quantification which is either FOR ALL or THERE EXISTS. Then the variables being quantified over must be specified, of which at least one must be given (otherwise this makes no sense). When specifying what variables are being quantified over, each subsequent variable expression may refer to previous variables. Following the variables is an optional condition that may be specified to prune the set of variable candidates being quantified over. Finally, the quantification being specified occurs over the given condition.

So, what does this *mean*? First, let us assume that evaluating a condition will return either nil or a non-nil value and that interpreting a condition will also be well-defined (conditions will be discussed next). If a quantified selector is on the LHS of a rule, then it must evaluate to a non-nil value for it to be true. If a quantified selector is on the RHS of a rule, then the given condition will be interpreted according to the quantification (that is, FOR ALL will interpret the condition as many times as there are bindings for the variables, while THERE EXISTS will only interpret the condition once).

To be more explicit, let's describe the four cases in the quantified selector: evaluated FOR ALL, evaluated THERE EXISTS, interpreted FOR ALL, and interpreted THERE EXISTS.

Evaluated FOR ALL In this case, the quantified selector will evaluate to a non-nil value if, for *all* the variables specified, passing the optional WHERE condition, the given condition evaluates to a non-nil value. If the resulting set of variables being quantified over and pruned through the WHERE condition is empty, the evaluation of the selector is vacuously true. The selector will only be evaluated if all the necessary references in the selector have values in the KNOMAD database.

Evaluated THERE EXISTS In this case, the quantified selector will evaluate to a non-nil value if, for *one* (or if more than one binding, one set) of the variables specified, passing

the optional **WHERE** condition, the given condition evaluates to a non-nil value. If the resulting set of variables being quantified over and pruned through the **WHERE** condition is empty, the evaluation of the selector is nil. The selector will only be evaluated if all the necessary references in the selector have values in the KNOMAD database.

Interpreted FOR ALL Interpreting the **FOR ALL** quantified selector will first prune the quantified variables through the optional **WHERE** condition, of which the given condition will be interpreted for each set of possible variable bindings. The selector will only be interpreted if all the necessary references in the selector have values in the KNOMAD database.

Interpreted THERE EXISTS Interpreting the **THERE EXISTS** quantified selector will first prune the quantified variables through the optional **WHERE** condition, of which the given condition will be interpreted for each set of variable bindings until one of them cause a non-nil value to be returned (or none remain). (This is unlike the existentially quantified rule, as we will see later.) The selector will only be interpreted if all the necessary references in the selector have values in the KNOMAD database.

One must be careful when specifying multiple variables to be quantified over. If each variable evaluates to more than a few items, the resulting combinatorics for computing all the possible variable binding sets could be quite large. If this is necessary it would be more efficient to restructure the problem and nest the quantification. For example, suppose you want to quantify over three variables, each of which evaluate to a list of ten items. The combinatorics specify that you will have 1000 combinations of variable bindings. It might be more efficient to represent this as a three quantified selectors, all nested within one another like so: `[FOR ALL foo in foo-var <[FOR ALL bar in bar-var <[FOR ALL mumble in mumble-var <[...] [...]>>>]]]` (which will still perform 1000 evaluations of the enclosed condition, but with perhaps less LISP consing).

Conditions Ah, conditions are much easier to describe than selectors. This will be a nice change of pace.

In its simplest form a condition is simply a sequence of selectors, one following another. If a condition is on the LHS of a rule it may also have a little more structure, this being a choice. The choice may be either **OR** or **EXCPT** (but not both). What this means is that a condition may have a number of selectors and either **OR** or **EXCPT** followed by more selectors. If the **OR** choice is used, either the selectors before the choice must evaluate to a non-nil value, or the selectors after the choice must evaluate to a non-nil value for the condition to be true. If the choice is **EXCPT**, then the selectors before the

choice must all evaluate to a non-nil value *and* the selectors after the choice must all evaluate to a nil value.

A condition on the RHS of a rule may not use the choice construct. Representing this possibility of alternate models or worlds in the current design of KNOMAD is simply to formidable a problem! Therefore, interpreting a condition simply amounts to interpreting each of its selectors.

Simple Rules A simple rule consists of an optional condition on the LHS and a condition on the RHS. If no condition is specified on the LHS, a default true condition is implied and the rule's LHS will always evaluate to true.

A simple rule is said to be satisfied if its LHS evaluates to a non-nil value and all the references on the RHS of the rule have values (and thus may fire). This is a little different than traditional expert systems, where satisfaction does not depend on variables on the RHS having values.

Firing a rule amounts to interpreting the rule's RHS. A rule may fire more than once (if selected via the control-strategy of the rule-group the rule is in). To clarify this further, a simple rule will be placed upon an activation queue if it is satisfied. When it is fired it will be taken off the activation queue. It will not be placed back on the activation queue unless a variable it references changes value. If this occurs, then the rule has the possibility of firing again.

As an example suppose we have the following rule:

```
[type of mammal = duck]
::>
[lisp::format (t "%Quack Quack!!")]
```

Then, if we (assert! '(frame mammal type :value duck)) this rule will be satisfied and placed upon the activation queue. When fired, the rule prints "Quack Quack!!" to LISP's standard output. If we subsequently enter (assert! '(frame mammal type :value dog)) and (assert! '(frame mammal type :value duck)) the rule will again be placed upon the activation queue and possibly fired again, depending upon the control-strategy of its rule group.

Quantified Rules Quantified rules are very much like quantified selectors except that they are not satisfied or fired for their truth-value as selectors are (evaluated and interpreted that is). A quantified rule will be placed on the activation queue just if all the variables it references have a value. This does not mean that the LHS of the rule is satisfiable. Some of the rule's selectors will refer to the bindings of the quantified variables. Therefore, the rule's LHS must be evaluated prior to determining if it is satisfied (this is true for simple rules too, just much simpler).

A quantified rule consists of an optional condition, the type of quantification (FOR ALL or THERE EXISTS), at least one specification of a variable to be quantified over, an optional WHERE condition, and finally a further rule over which the quantification takes place (the syntax is given in the appendix).

Evaluating a quantified rule is the same for either type of quantification. If the optional LHS condition exists, it must evaluate to a non-nil value. Then the variable binding sets are determined exactly the same way as in the quantified selectors. These are pruned through the optional WHERE condition and the resulting set of variable bindings saved. If there are any resulting variable bindings, the evaluation of the rule is satisfied, otherwise it is not.

Interpreting the RHS of a FOR ALL quantified rule proceeds by evaluating the sub-rule for each variable binding set. For each sub-rule whose evaluation indicates it is satisfied, it is then interpreted. Interpreting the FOR ALL rule always indicates success no matter whether any of the sub-rules were fireable or not.

Interpreting the existentially quantified rule only attempts to evaluate the sub-rule for one of its variable binding sets. It will evaluate the sub-rule and if the evaluation indicates success, interpret it. No matter what happens with the sub-rule, interpreting the THERE EXISTS rule always indicates success.

Universally quantified rules may fire many times, existential rules, only once (for any single invocation of the rule group).

B.4.2 The Backward Chaining System

Performing inference using backward chaining in KNOMAD is very much like Prolog. Defining backward rules for facts and for slots of frames in effect defines virtual slots on frames and defines virtual facts in the database. Furthermore, the virtual slots on frames will also benefit from the automatic inheritance of the frame representation language. Thus, if a backward rule is defined on a person frame to find a person's ancestors, then that rule will also be accessible from any subclass/instance of the person frame (unless a more specific rule overrides it).

Backward chaining allows the user to define data in the database that is only implicitly present. Backward rules are the means by which this is done. One type of implicit data is where a slot on a frame does not explicitly exist, but a backward rule for determining it by using other slots will allow inferencing to find it. Another type of data is where a slot for a frame does exist, but a particular query cannot find the desired data explicitly so backward rules are tried that may produce additional data only implicitly present. In summary, Defining backward rules on facts (or frames) allows inferencing to produce implicit data in the database.

This section will not try to provide a tutorial on backward chaining or Prolog. It is suggested that the reader peruse one of the many excellent books on Prolog, such as *The*

Art of Prolog."**bwd:bwd-rule** *head tail*

[Function]

The head is a clause while the tail is a disjunction of a conjunction of clauses. The head must be either a "frame" or "fact" tuple and the particular frame or fact must be grounded (not a variable). If the head clause is a frame, the slot may be a variable in which case the rule will be applicable to a query for any slot on the specified frame.

The clauses in the tail have less restrictions. Here you may have "frame" or "fact" tuples as in the head. Furthermore, you may have functional escapes for computing a variable, negation, and explicit failure. An explicit failure is indicated with a clause consisting of the single symbol `:fail`. Only a single level of negation is allowed and is indicated with: `(:not <clause>)`. A functional escape may be used to give a variable a value. Its syntax is `(:function <variable> <function> <argument>*)`. This will apply `<function>` to the given arguments and unify the result with `<variable>`. Thus, if one wants to keep the result for other parts of the backward chaining, `<variable>` should be a true variable as opposed to a constant of some sort.

Some examples of legal function clauses are:

```
(:function t <= $number 1) ;; test if $number is less than or equal to one
(:function $var my-function $arg1 $arg2)
```

The first clause will be true if `$number` is less than or equal to one. This is because, if it is, the function application will return `t` which unifies with `t`. If the test is not true, then the function application will return `nil` which does not unify with `t`.

The second clause illustrates the application of `my-function` to the current bindings of the arguments `$arg1` and `$arg2` with the result being unified with `$var`. If `$var` already has a value, then it better be the same as the result of the function call, otherwise `$var` will be given a new binding.

To illustrate what can be expressed using backward rules, a number of example databases are given.

The Ancestor Relationship database

```
(frame :name person
      :slots (mother father))

(make-person 'joel :mother sue :father wil)

(bwd-rule '(frame joel ancestor :value $who)
```

SSM/PMAD Final Report

```

      '(((frame joel mother :value $who))))
(bwd-rule '(frame joel ancestor :value $who)
          '(((frame joel father :value $who))))

(bwd-rule '(frame joel ancestor :value $who)
          '(((frame person mother :value $mom)
              (frame $mom ancestor :value $who))
            ((frame person father :value $dad)
              (frame $dad ancestor :value $who))))

```

This database defines four rules in three statements. All four rules could be done in one statement if desired. Backward chaining will infer over applicable rules in the order they are defined and will try to prove each of the clauses of the tail in a left to right order.

The Foo database

```

(assert! '(fact bar :value 4))
(assert! '(fact mumble :value 12))

(bwd-rule '(fact foo :value $var)
          '(((fact bar :value $var1)
              (fact mumble :value $var2)
              (:function $var * $var1 $var2))))

```

The Fibonacci Function database

```

(frame :name fib :slots ())

(bwd-rule '(frame fib $number :value $result)
          '(((function t <= $number 1)
              (function $result identity 1))))
(bwd-rule '(frame fib $number :value $result)
          '(((function t > $number 1)
              (function $number1 - $number 1)

```

```
(frame fib $number1 :value $result1)
(:function $number2 - $number 2)
(frame fib $number2 :value $result2)
(:function $result + $result1 $result2))))
```

So far we have seen how to define a database of both explicit and implicit data. The next trick is to determine what is in the database. This is done using the `show!`, `show-many!`, and `more!` functions.

`bwd:show! clause` [Function]

Any clause defined as legitimate for the tail of a backward rule may be used here. It is generally not safe to specify the particular frame of a frame clause as a variable (e.g. `(frame $frame slot1 :value fred)`) unless your rules are specifically set up to handle it appropriately (the frame representation language bases its inheritance upon the actual frames in existence in the database). An example rule that would work correctly is:

```
(bwd-rule '(frame person respect :value $someone)
  '((frame $someone can-do-rubiks-cube :value true))))
```

If a query binds `$someone` so that the first clause of the tail has it bound, then you are safe. If, however, the query leaves `$someone` a variable, chances are you will not find what you expected (you will get failures even though the data may be implicit in the database, the check for `(frame $someone can-do-rubiks-cube :value true)` will not use inheritance to find matching tuples or inherited rules).

`show!` will return a result and a continuation structure as multiple values. It is a good idea to use `multiple-value-setq` or a variant in order to capture the continuation structure. The continuation structure is used to retrieve further results that may exist in the database.

`bwd:more! continuation` [Function]

`more!` takes a continuation and returns a result and another continuation structure for further results.

When either `show!`, `show-many!`, or `more!` return `:fail` then the continuation structure will be `nil` and no more results will be forthcoming.

`bwd:show-many! clause*` [Function]

`show-many!` takes any number of clauses and will try to prove all of them, maintaining bindings across the clauses. This function will return a list of the clauses (with variables bound) if they can be proven from the database as well as a continuation structure for `more!`.

We now illustrate some examples of using these functions:

The Ancestor Relationship database

```
[cl]: (multiple-value-setq (result continuation)
      (show! '(frame joel ancestor :value $who)))
```

```
(FRAME JOEL ANCESTOR :VALUE SUE)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
(FRAME JOEL ANCESTOR :VALUE WIL)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
(FRAME JOEL ANCESTOR :VALUE MOM1)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
(FRAME JOEL ANCESTOR :VALUE DAD1)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
(FRAME JOEL ANCESTOR :VALUE MOM2)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
(FRAME JOEL ANCESTOR :VALUE DAD2)
```

```
[cl]: (multiple-value-setq (result continuation)
      (more! continuation))
```

```
:FAIL
```

Here is an example of a query that will bind \$who in order that the second clause will work properly.

```
[cl]: (multiple-value-setq (res con)
      (show-many! '(frame joel ancestor :value $who)
                  '(frame $who father :value $someone-else)))

((FRAME JOEL ANCESTOR :VALUE SUE) (FRAME SUE FATHER :VALUE DAD1))
[cl]: (multiple-value-setq (res con)
      (more! con))

((FRAME JOEL ANCESTOR :VALUE WIL) (FRAME WIL FATHER :VALUE DAD2))
[cl]: (multiple-value-setq (res con)
      (more! con))

:FAIL
```

The Fibonacci Function database

```
[cl]: (show! '(frame fib 6 :value $value))

(FRAME FIB 6 :VALUE 13)
#<Continuation>
[cl]: (multiple-value-setq (res con)
      (show! '(frame fib 8 :value $value)))

(FRAME FIB 8 :VALUE 34)
[cl]: (multiple-value-setq (res con)
      (more! con))

:FAIL
```

In summary, this tool for backward chaining is very much like Prolog. It provides a large part of the functionality of Prolog with the main exception of the restriction on variables for particular frames (this is primarily a result of the frame representation language inheritance system). For that reason, this system is currently incomplete. However, it is sound.

Future work should address the problem of the frame inheritance system. Other options could consider compiling rules to the Warren Abstract Machine and running that for efficiency. It may also be useful to provide Prolog cuts.

B.4.3 The Constraint System

The constraint system of KNOMAD is quite advanced. The constraint system allows the user to specify a system of constraints in a very general manner. However, it is not well integrated into KNOMAD. It does not use the KNOMAD database. It does not have a good set of user interface functions to manipulate it. It hasn't been used much yet and so has not had a chance to develop those little things that are very useful, but not always obvious at first cut.

This constraint system allows the user to specify a system of constraints, cells (being constrained), and their values. All cells in the constraint system must be given a domain (a type). Cells may also be given hooks (callbacks) that are run when the cell value is derived down to a single value or no values (inconsistent). The constraint system does not return a solution to the network of constraints. It is up to the application to determine how underconstrained the network is and to further constrain it if necessary. The ability to ask the network to return all possible consistent solutions might be useful in the future. The constraint system also allows the user to specify values in contexts (which can subsume one another). This is done using Assumption-based Truth Maintenance System (ATMS) environments. Again, This integration of ATMS environments, needs further polishing and integration. The constraint system should be considered as an advanced prototype at this writing.

The creation of a constraint system, its constraints and cells, will be described here. First we will discuss the normal usage without contexts. Then we will describe how to use the contextual ability of the constraint system. But, before we go on to cells and constraints we need to know how to make a constraint system.

`constraints:make-constraint-system` *[Function]*

This function will return a new constraint system with which one can make cells and constraints. By specifying cells and constraints in the context of a constraint system it is possible to have many constraint systems active at once.

Cell and their Domains

Each constraint in the constraint system specifies how a number of cells are constrained with respect to each other. The possible constraints will be discussed next. Here we discuss what legal domains a cell may have.

The possible domains of a cell are listed in Table B.4.3. The listed types are all specified in the description of the cell domain using their symbol in the keyword package. Thus, the integer type would be specified as “:integer.” The domain for all the simple types except nominal are simply specified as is. Thus if I have a cell of type string I would specify it as “:string.” The nominal type must include the elements of the enumeration. If I have a

- Simple Types
 - **Integer** Any integer, positive or negative.
 - **Positive** Any integer greater than zero.
 - **Natural** Any non-negative integer.
 - **Real** The set of real numbers (there are some constraints on floating point precision and range).
 - **String** Any string.
 - **Nominal** A single valued type, like an enumeration. Boolean falls into this category.
- Complex Types
 - **Set** A set of some other type.
 - **Record** A structured type of other types, each field of a record holds the value for some other type.

Table B.1: Constraint system cell domain types

cell for representing primary colors I would specify its domains as: (:nominal red green blue). All complex types and the nominal type are specified using this list notation in which the first element of the list is the type and the following elements, as necessary, are the member types or elements (as in the case of nominal).

A set type must specify what it is a set of. A set of integer looks like: (:set :integer). A set of set of integers is (:set (:set :integer)). A record type is more complex. A record type must include field names, with each field containing a description of its type. A record of two integers may look like: (:record (field1 :integer) (field2 :integer)). A more complex example of a record is:

```
(:record (field1 (:record (s1 :integer)
                        (s2 (:set :integer))))
         (field2 (:nominal honda chevy ford nissan toyota gm isuzu)))
```

`constraints:create-cell` *c-system domain* &optional *s-hooks v-hooks* [Function]

This function is used to create a cell in a given constraint system. The domain of the cell is given as described earlier. *s-hooks* and *v-hooks* are lists of functions which will be called by

the constraint system when propagation of constraints has derived a single value for the cell or no value for the cell (respectively). When no value is derived for a cell, this means that the network of constraints is over constrained (and therefore inconsistent).

After creating cells it is sometimes desirable to give a cell a value. This is done by using the `set-cell-value` function. It is only possible to provide a single value for a simple cell type. It is not possible to set the value of an entire record cell at once. It must be done by setting the value for each leaf of the record separately. As an example, to give the value 34 to `cell1`, an integer type, we would specify the value as simply 34. If `cell1` had the domain of the complex record type defined earlier, to set the value of `s1` we would specify (`field1 (s1 34)`) as the cell value.

Setting a value for any type is thus very straightforward, a set type receiving a set as a value, a nominal type receiving a single element of the defined enumeration, etc. Numeric values (integer, positive, natural, and real) may be specified as an interval as well as a single value. If one wants to say that `cell1`, an integer type, has a value in the range 34 to 45, inclusive, one specifies (`34 45`) as the value.

`constraints:set-cell-value cell value &optional env start-propagation` [Function]

This function gives *value* to *cell*. If the cell is defined as a complex type, the given value must refer to a leaf of the complex type (i.e. record types). We will discuss the *env* parameter in the section on contexts later. The *start-propagation* parameter defaults to `t`. It is used to start propagation upon setting the value of a cell. Sometimes it is useful to specify this as `nil` when setting a lot of values and wanting to have the propagation occur only when you are done. However, this flag is not consistently implemented at the moment.

`constraints:get-cell-value cell &optional env` [Function]

This function returns the current value set for the given cell in the optionally provided context *env* (to be discussed later). The value-set can be a list, each element being a possible value, `nil`, indicating that the cell has no value and the network is overconstrained, and `:unknown`, indicating that this cell simply does not yet have any values.

Constraint Types

Most constraints operate in a multi-directional fashion. For example, if a `:plus` constraint is applied to three cells such that it is intended that `cell13` equal the value of `cell12` plus `cell11`, then, as long as a value is provided on two of the three cells, the third cell will also get a propagated value. This can happen in any direction for most constraints. Some constraints are limited, such as the `:function` constraint.

- Plus
- Minus
- Times
- Integer Division
- Real Division
- Mod
- Equal
- Not Equal
- Less Than
- Greater Than
- Less Than or Equal
- Greater Than or Equal
- Union
- Intersection
- In
- Not In
- Unique In
- Not Unique In
- Function

Table B.2: Constraints available in the Constraint System.

Plus		Constraint name: :plus	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>addend1</i>	cell3 <i>addend2</i>
set	↔	set	T
integer	↔	integer	integer
natural	↔	natural	natural
positive	↔	positive	positive
real	↔	real	real

Table B.3: The Plus Constraint. The plus constraint allows the application user to add an element to a set as well as the normal concept of mathematical addition. The reverse direction of adding an element to a set is only partial. If the set is larger than a constant (currently 10) the reverse direction will not be propagated.

Minus		Constraint name: :minus	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>minuend</i>	cell3 <i>subtrahend</i>
set	←	set	set
set	↔	set	T
integer	↔	integer	integer
natural	↔	natural	natural
positive	↔	positive	positive
real	↔	real	real

Table B.4: The Minus Constraint. The minus constraint allows the application user to subtract an element from a set as well as the normal concept of mathematical subtraction. The reverse direction of subtracting an element of a direction is only partial. If the set is large than a constant (currently ten), then the reverse direction will not be propagated. Subtracting one set from another is the set-difference operation.

Times		Constraint name: :times	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>multiplicand1</i>	cell3 <i>multiplicand2</i>
integer	↔	integer	integer
natural	↔	natural	natural
positive	↔	positive	positive
real	↔	real	real

Table B.5: The Times Constraint

IDiv		Constraint name: :idiv	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>numerator</i>	cell3 <i>denominator</i>
integer	↔	integer	integer
natural	↔	natural	natural
positive	↔	positive	positive

Table B.6: The IDiv Constraint

RDiv		Constraint name: :rdiv	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>numerator</i>	cell3 <i>denominator</i>
real	↔	real	real

Table B.7: The RDiv Constraint

Mod		Constraint name: :mod	
cell1 <i>result</i>	<i>direction</i>	cell2 <i>numerator</i>	cell3 <i>denominator</i>
integer	↔	integer	integer
natural	↔	natural	natural
positive	↔	positive	positive

Table B.8: The Mod Constraint

Equal Constraint name: :equal		
cell1	direction	cell2
set	↔	set
record	↔	record
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string
nominal	↔	nominal

Table B.9: The Equal Constraint

Not Equal Constraint name: :not-equal		
cell1	direction	cell2
set	↔	set
record	↔	record
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string
nominal	↔	nominal

Table B.10: The Not Equal Constraint

The currently provided constraints in the KNOMAD constraint system are listed in Table B.2. Tables B.3 through B.21 describe the constraints, their arity and types, and directionality. Any unusual semantics will be described in the captions. Directionality is indicated by an arrow. If the arrow is one direction then the arguments can only influence the result, if the arrow is bidirectional, then both the results and the arguments can influence each other. Type "T" matches any type.

`constraints:make-constraint c-system constraint &rest cells` [Generic Function]

This generic function is used to make a constraint. The actual constraint is one of the defined constraints in the given tables. All the rest of the arguments must be cells that will be constrained by the given constraint. The cell arguments must be specified in the proper order as described in the tables.

Less Than Constraint name: :less-than		
cell1	direction	cell2
set		set
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string

Table B.11: The Less Than Constraint. One set being less than another set is defined as the subset relationship. Propagation is not done in either direction for reasons of combinatorial explosions. If either set gets a different value from other propagation, then the relationship will be checked. Less than on strings is defined in a lexicographic ordering.

Greater Than Constraint name: :greater-than		
cell1	direction	cell2
set		set
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string

Table B.12: The Greater Than Constraint. One set being greater than another is the superset relationship. Strings are ordered lexicographically.

Less Than Equal Constraint name: :less-than-equal		
cell1	direction	cell2
set		set
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string

Table B.13: The Less Than Equal Constraint. One set being less than or equal to another is defined as to be the proper subset relation between sets. Strings are treated similarly with a lexicographic ordering.

Greater Than Equal Constraint name: :greater-than-equal		
cell1	direction	cell2
set		set
integer	↔	integer
natural	↔	natural
positive	↔	positive
real	↔	real
string	↔	string

Table B.14: The Greater Than Equal Constraint. One set being greater than or equal to another is defined to be the proper superset relation between sets. Strings are treated similarly with a lexicographic ordering.

Perusal of the tables will show that it is not possible to specify a constraint between a field of a record and anything else. This is a limitation that would be nice to retract in a future upgrades to this system. There are certainly a large number of other operators and domain types that would also be useful. One can think of more set operations such as the power set of a set. It could be nice to represent matrices and operations upon them. If users have suggestions for what would be useful, please be sure to contact the author.

Constraint System Contexts

The constraint system of KNOMAD also supports the notion of contexts. A value for a cell may be represented with respect to a particular context. If propagation is being done, it is always done in some context, usually the default context (which subsumes all other contexts). When propagation is being performed, or when a value is being examined, the context determines what value is seen. This value will be determined from the current context and any subsuming contexts. If a subsuming context is inconsistent, all subsumed contexts will also be inconsistent.

This use of contexts is exactly that of environments in the ATMS. In fact, a context is an ATMS environment. Each constraint system has an associated ATMS in the current implementation. Thus to generate new contexts, assumptions must be made from which new environments can be made using the ATMS. Environments made in this fashion are suitable to those functions described earlier that take an optional *env* parameter. The functions to do this are described here.

This discussion of contexts is not intended to be understandable by those who do not understand the operation of the ATMS. This is intentional for two reasons. One, the integration of an ATMS and a constraint system into KNOMAD is really quite weak and must be used with a lot of care. Two, the use of contexts is a very powerful and also potentially combinatoric. I do not wish to take the space in this document to describe the ATMS or the

Union Constraint name: :union			
cell1	direction	cell2	cell3
result		arg1	arg2
set	←	set	set

Table B.15: The Union Constraint

Intersection Constraint name: :intersection			
cell1	direction	cell2	cell3
result		arg1	arg2
set	←	set	set

Table B.16: The Intersection Constraint

In Constraint name: :in		
cell1	direction	cell2
T	↔	set

Table B.17: The In Constraint

Not In Constraint name: :not-in		
cell1	direction	cell2
T	↔	set

Table B.18: The Not In Constraint

Unique In Constraint name: :unique-in		
cell1	direction	cell2
T	↔	set

Table B.19: The Unique In Constraint

Not Unique In Constraint name: :not-unique-in		
cell1	direction	cell2
T	↔	set

Table B.20: The Not Unique In Constraint

Function Constraint name: :function		
cell1	direction	argument cells
T	←	T

Table B.21: The Function Constraint. This constraint allows for an arbitrary function to be used to constrain a result cell. If the value of any cell (result or arguments) changes as a result of propagation, the function will be applied to further constrain or enforce the result cell.

use of contexts as that is not central to KNOMAD. Instead the reader is referred to DeKleer's original paper [DeK86].

`constraints::constraint-system-atms` *c-system* [Function]

This function returns the ATMS of the given constraint system.

`atms:build-assumption` *datum atms* [Function]

This function will return a valid assumption for use in making new environments. The *datum* can be anything at all. Usually it is meaningful to the application user.

`atms:make-env-from-assumptions` *assumptions* [Function]

This function takes a list of assumptions, all of which must be from the same ATMS, and returns an environment.

B.5 Miscellaneous Utilities

This section describes a few of the miscellaneous utilities that are available to the application programmer utilizing KNOMAD. These utilities are queues, events, semaphores, a clock abstraction, and the unifier.

B.5.1 Queues

The queue data structure is ubiquitous in LISP programming. It is therefore important to have a good queue structure that is very efficient. The one provided here has constant time and space performance whether adding or removing items from the queue.⁶

⁶Not linear time and space, *constant*. Obvious methods, such as `nconc`, are linear in the size of the queue.

`create-queue &optional name` [Function]

This function will return a newly created and initialized queue. If *name* is not given, a gensymed name will be provided.

`init-queue q` [Function]

This function will initialize the given queue. It's return value is undefined.

`add-entry q item` [Function]

This function will add the entry *item* to the end of the queue *q*. The *q* is returned (i.e., nothing useful).

`remove-entry q` [Function]

This function will remove the top entry of the queue *q* if it is not empty, otherwise it will return `nil`. Use `get-queue-entries` if you need to determine if there are any entries in the queue or not.

`get-entry q` [Function]

This function returns the top entry of the queue *q* without removing it from the queue. If the queue is empty, `get-entry` will return `nil`.

`get-queue-entries q` [Function]

This function will return a list of the entries in the queue *q*. The list is ordered with the car being the top element of the queue. If you wish to modify the list, make a copy of it first. For efficiency reasons, modifications to the list will result in undefined problems occurring later. If there are not entries in the queue *q*, `get-queue-entries` will return `nil`.

B.5.2 Events

The following description of events in KNOMAD is taken from the source code.

An event is used by a process to wait for something. Generally, an event can be thought of as a situation. To wait for an event a process calls the `await-event` function. To wake up from an event some other process must call `notify-event` on the event that the sleeping process is waiting for. Of course the event must be first created with the `create-event` function.

The `await-event` function may also take an optional timeout argument. This argument is the number of *seconds* that the process is willing to wait for a notification. If the notification

does not come in that time, it will be awoken regardless. If the timeout is not given, the process may potentially wait forever.

An event may be notified before a process is awaiting it. If this happens the signal is stored in the event. The first process that awaits the event will then be immediately woken up. Multiple notifications to an event do *not* stack. They will be treated as one notification.

If there is more than one process waiting for an event, they will all be woken up if the event is signaled. This is probably the way one would do events in OS programs as well. If it turns out that all the processes are waiting on the same resource and only one can have it then you have a case where you would want to use Dijkstra's locking algorithm. There is a difference here between events and locks that I am trying to distinguish. Events are used to signal things in general. Locks are used to obtain exclusive access rights over something. Events may cause locking operations, while locking mechanisms may use events internally in some manner.

This manner of event handling is *very* similar to the Interlisp notion of events [WT78].

Some more thoughts as a result of scanning "An Introduction to Operating Systems" by Harvey M. Deitel (Addison-Wesley, 1984):

Events are generally used for synchronization. This implementation generalizes the basic event mechanism in two ways; one, so multiple processes can wait on the same event, and two, a waiting process can have a timeout (is this similar to some of Ada's mechanisms for process synchronization?). Another generalization of the event mechanism is to allow a process to wait on a boolean combination of events. To do this one would want to define events as bits in a word. As events are signalled, blocked processes (waiting on events) are anded (logically) with the signalled event to see if they are woken up.

`create-event` *&optional name* [Function]

This function will return a newly created event. If *name* is not given, a gensymed name will be provided for the event.

`await-event` *event &optional timeout* [Function]

This function will cause the calling function to be suspended until *event* is notified. If *event* has been notified previously with no intervening `await-event`s being called, `await-event` will return immediately. If *timeout* is given, the calling process will return from `await-event` either when *event* is notified or when *timeout* seconds have elapsed, whichever is sooner.

`notify-event` *event* [Function]

This function will notify *event*, waking any processes that are have called `await-event` on *event*.

`user::describe-event event` [Function]

This function is useful for determining what the status of *event* is. It will print the name of the event, the signal of the event (`t` or `nil`), the sleepers (these are processes that are managing any timeouts for processes that have called `await-event` with a timeout), and the waiting processes.

B.5.3 Semaphores

This is the basic semaphore that one reads about in any operating system book. The semaphore provided here is a counting semaphore with three priority queues. There are four functions provided here: `create-sem` for making a semaphore, `sem-p` and `sem-v` for grabbing and releasing a semaphore, and `init-sem` for reinitializing a semaphore. `init-sem` will not be defined here since it is not recommended that it be used. It is not clear what its semantics ought to be if any processes are waiting on the semaphore. Please avoid it.

The semaphore defined here uses three priority queues: `:hi`, `:med`, and `:lo` in order of priority. As processes try to grab the semaphore, they can specify which queue to be placed upon if the semaphore is already taken (the default queue is `:med`). As other processes release the semaphore, processes that are waiting on the semaphore are given access to the semaphore in priority queue order. Furthermore, each priority queue is accessed as a typical queue, first in, first out.

`create-sem &optional count name` [Function]

This function returns a newly created semaphore. *count*, if provided, should be a positive integer. If it is not provided, it will be initialized to one. The count specifies how many processes may grab the semaphore at once. A count of one indicates exclusive access. The *name* will be used to provide a name for the semaphore, if it is not provided, one will be generated.

`sem-p semaphore &optional priority` [Function]

If the current count of *semaphore* is greater than zero, this function will decrement the count and let the process continue. Otherwise, the process will be placed upon the indicated *priority* queue (defaulting to `:med`) and suspended until some other process releases the semaphore.

`sem-v semaphore` [Function]

This function is used to release *semaphore*. If there is a process waiting for the semaphore, the first process accessed will be resumed (priority queue ordering).

B.5.4 Clocks

The clock abstraction allows the application programmer to specify a clock that is independent of other application clocks, and even of the system clock. Using the clock it is possible to simulate a faster or slower time period using a specified units-per-second.

Starting the clock starts a process running that control the time of the clock. That process will update the clock time once every real second (approximately). Thus, it is not possible to implement an application which requires a granularity of less than one real second. For example, if one defines the units-per-second of their clock to two, when cannot sleep for only one of those clock seconds. But, everytime the clock does update, you should see that the time increments by two seconds.

To use a clock one should first make a clock, set the clock time and then start it. When one is done with it they should kill the clock.

All clock time parameters are in `universal-time` format.

`make-clock` [Function]

This function returns a new clock. The clock is completely uninitialized.

`set-time clock time` [Function]

This function will set the current time of the clock. It is well-defined to set this to any valid universal time in the past or future.

`clock-speed clock &optional units-per-second` [Function]

This function will set the speed of the clock to *units-per-second* if provided. *units-per-second* must be a positive integer. `clock-speed` returns the current units-per-second of *clock*. If -1 is returned, this means that an illegal value of *units-per-second* was provided.

It is okay to change the speed of the clock at any time during the clock execution. If this is done, the time of the clock will reflect a value as if the current units-per-second has always been the same during the execution of the clock.

`start-clock clock` [Function]

This function starts the clock *clock*. If *clock* is already running, it will first be killed.

`get-time clock` [Function]

This function returns the time of *clock* in `universal-time` format.

`clock-sleep` *clock seconds* [Function]

This function will cause the calling process to sleep for the indicated number of *seconds*. *seconds* will be evaluated in terms of the current speed of the clock. If someone changes the speed of the clock while another processes is sleeping on it, the sleeping process will not see this change reflected in how long it sleeps.

`kill-clock` *clock* [Function]

This function stops the clock if it is running.

B.5.5 The KNOMAD Unifier

The KNOMAD unifier is a very simple symbolic expression unifier for LISP lists. It does not try to unify types with sub-types, etc.

`unify` *pat1 pat2* [Function]

`unify` takes two patterns and returns a list of bindings or `:fail`. An empty list of bindings indicate the two patterns unified without creating any new bindings. The binding list is a disembodied property list with the key being the variable and the value being the binding.

`var-p` *symbol* [function]

This function returns `nil` if the given symbol is not a variable. A variable is indicated by the first character of its print name being equal to “\$.”

B.6 Object-Oriented Programming in KNOMAD

This section briefly discusses the role of object-oriented programming in KNOMAD. Object-oriented programming is a very popular programming trend. There are many variants of what it means to have or perform object-oriented programming. These variants include Ada as an object-oriented language, C++, CLOS, and many others. Frame representation languages, since their inception, have also had elements of object-oriented programming. These elements are discussed here.

B.6.1 Objects and Inheritance

Frames provide for the definition of objects and for inheritance. We have already seen how to create a frame object and how to make instances or children of frame objects. The frame representation language is different from many other object-oriented languages in that all

frame objects/instances are first class frames. It is possible to subclass a frame class as well as a frame instance.

When looking up a value on a slot of a frame, inheritance is automatically used to find the nearest value in the inheritance hierarchy. This is what would be expected in most any object-oriented language.

B.6.2 Methods

A method is generally considered to consist of some code that performs the indicated function and returns a result. The indicated function is usually specialized depending upon the object the function is being performed upon. Object-oriented languages naturally support the notion of function overloading.

The frame representation language provides for this ability through the use of both if-needed aspects on slots as well as the backward chaining rule system of KNOMAD. In order for this to make sense one has to understand that defining a function based upon an object type to return a value is analogous to defining a function on an if-needed aspect of a slot and asking for the value of that slot. Furthermore, by using backward rules, one can define virtual slots that depend upon other slots with their own if-needs or backward rules. These if-needed functions and backward rules are also inherited in the frame representation language, exactly as methods are in traditional object-oriented languages.

Thus, to see that the frame representation language qualifies as one of the more traditional sorts of object-oriented languages, one only has to slightly modify their concept of how to call a method on an object. Once that is understood, it is very natural to use object-oriented programming styles in the frame representation language of KNOMAD.

B.7 KNOMAD BNF Syntax

B.7.1 Definitions

This appendix describes the syntax of KNOMAD using extended BNF notation. The meanings of the meta-characters are given in the table.

Symbol	Meaning
{	begin optional grouping
}	end optional grouping
	alternative
+	one or more
*	zero or more
ID	analogous to any LISP atom
STRING	a string
NUMBER	a number

B.7.2 Rule Management System

```
knowledge-base ::= KB : ID {DOMAIN : ID} rule-group+ {domain-knowledge}
                BEGIN : ID+ END-KB
```

```
rule-group ::= RULE-GROUP : ID
            {CONTROL : transition-table}
            {CONTROL-STRATEGY : control-strategy}
            {CONFLICT-RESOLUTION-STRATEGY : conflict-resolution-strategy}
            rules
            {: termination-condition}
```

```
domain-knowledge ::= {CONSTANTS : constants .}
                  {FACTS : facts .}
                  {FRAMES : frames .}
```

```
control-strategy ::= FUNCTION | ID
```

```
conflict-resolution-strategy ::= FUNCTION | ID
```

```
termination-condition ::= condition
```

```
transition-table ::= ((ID (ID+))+)

```

```
constants ::= ID | ID ; constants
```

SSM/PMAD Final Report

```

facts ::= fact | fact ; facts

fact ::= ID = ID | ID = (ID*)

frames ::= frame+

frame ::= (FRAME :NAME ID
           {:PARENTS parents}
           {:ISNEW FUNCTION}
           {:SLOTS ((ID aspect VALUE { aspect VALUE }*)*)})

parents ::= ID | (ID+)

aspect ::= :IF-NEEDED | :IF-ADDED | :CONSTRAINT | :MUSTBE | :VALUE | :DDB

rules ::= rule | rule ; rules

rule ::= {condition} ::> condition |
        {condition} quantifier {ID IN expr}+ {WHERE condition} <rule>

quantifier ::= FOR ALL | THERE EXISTS

condition ::= selector+ {choice selector+}

choice ::= EXCPT | OR

selector ::= [quantifier {ID IN expr}+ {WHERE condition} <condition>] |
            [expr {relation reference}]

relation ::= NOT MEMBERIN | NOT UNIQUEIN | MEMBERIN | UNIQUEIN |
           = | <> | >= | <= | > | <

reference ::= expr

expr ::= message | constant | path | expr op expr

constant ::= STRING | NUMBER

op ::= PLUS | MINUS | UNION | TIMES | IDIV | RDIV | MOD

path ::= ID | ID OF path

```

```
message ::= path :: ID {(expr+)}
```

B.7.3 Frames

```
frame ::= (FRAME :NAME ID
           {:PARENTS parents}
           {:ISNEW FUNCTION}
           {:SLOTS ((ID aspect VALUE {aspect VALUE}*)*))}
```

```
parents ::= ID | (ID+)
```

```
aspect ::= :IF-NEEDED | :IF-ADDED | :CONSTRAINT | :MUSTBE | :VALUE | :DDB
```

B.7.4 Database Assertions

In the following, VALUE represents any object.

```
fact ::= (FACT ID :value VALUE)
```

```
frame ::= (FRAME ID slot aspect VALUE)
```

```
slot ::= ID
```

```
aspect ::= :IF-NEEDED | :IF-ADDED | :CONSTRAINT | :MUSTBE | :VALUE | :DDB
```

B.7.5 Integrity Constraints

```
i-constraint ::= (item+) | ID
```

```
item ::= i-constraint | type-decl | type-spec
```

```
type-decl ::= (:type-decl ID)
```

```
type-spec ::= (:type-spec ID)
```

B.8 Glossary

arity

Refers to the number of arguments a function may be passed. In the case of tuples, the number of elements in the tuple. Thus, a 5-tuple has an arity of five.

ATMS

Assumption-Based Truth Maintenance System. See [DeK86].

KNOMAD

KNnowledge MAnagement and Design system.

lexicographic

An ordering. The order of elements in the ordering is based upon a position by position lexical comparison of each element. Generally, the lexicon used is the the ASCII character set.

(This Page Intentionally Left Blank)

Appendix C

MAESTRO Internals Reference Manual

C.1 Introduction

This section is a high-level description of the various components of the scheduling system MAESTRO. The topics covered in this section are introduced and definitions of some scheduling terms are given. Section C.2 describes some of the data objects used in the system. Section C.3 describes how MAESTRO builds a schedule. Section C.4 describes non-real-time contingency handling, while section C.5 discusses real-time contingency handling. Refer to the MAESTRO Users Manual [GBG91b] for details on how to operate the system, and to the MAESTRO Technical Reference [GBG91a] for a more in depth discussion of how MAESTRO works.

MAESTRO scheduling revolves around activities, resources, and subtasks. An **activity** can be thought of as a sequence of actions which, when executed, accomplishes a goal. These sequences of actions typically require the use of **resources**. Each action in the sequence will require a particular set of resources, and we define a **subtask** as an action or contiguous sequence of actions which requires a uniform set of resources. These subtasks may also require that a particular environmental state be maintained, and may be dependent on the execution of other subtasks. An activity is thus defined as a linear, non-overlapping sequence of one or more subtasks. Within this context, **scheduling** may be defined as the specification of start and end times for the subtasks making up activities. We define a **valid schedule** as a specification of start and end times for subtasks such that all activities on the schedule may be successfully executed at the times specified by the schedule.

In terms of a real-life experience, activity, resources, and subtasks can be described as follows. Making a cake can be described as an activity composed of a sequence of actions such as measuring and combining ingredients, preheating an oven, and baking the cake. In our example, the resources would include the ingredients, the bowls and utensils used,

the cake pan, the mixer, the oven and the electricity to power them. A series of subtasks representing these actions might include initial preparation (finding the recipe, verifying that all necessary ingredients are on hand, etc.), preheating the oven, mixing liquid ingredients, mixing in dry ingredients, preparing a cake pan and filling it, placing the cake in the oven and setting a timer, baking the cake, testing for doneness, and removing the cake from the oven.

C.2 MAESTRO Data Objects

In the MAESTRO system, **activities** are represented as sequences of **subtasks**. Activities which represent different ways of achieving a single goal are aggregated into **activity-groups**. There is typically a preferred method of achieving a goal, and we refer to the activity associated with this method as the **primary model** for the activity-group. Other methods are referred to as **alternate models**. The placement of activities on a schedule is controlled by the scheduler. Intervals of time during which things are happening that are not under the control of the scheduler are called **events**. In MAESTRO activity-groups, activities, subtasks and events are implemented as flavors. Activity-groups contain a field which is a list of activity-ids indicating which activity models belong to each group. Activities have a field which contains a pointer to the instance of the group to which they belong. Activities also have a field in which are stored the list of subtasks making up the activity, and a field containing a list of ids for these subtasks. Subtasks, in turn have a field in which is stored the id of the parent activity for each.

Activities have a number of characteristics editable by users. These include activity priority, number of performances requested, temporal constraints, and placement preferences, among others.

Activities in MAESTRO sometimes have effects on one another aside from competing for resource use. An activity may facilitate another in some way, or may interfere with it. These effects are represented by **temporal constraints**. It is sometimes necessary for MAESTRO to reason about more than one activity at a time when making scheduling decisions, so activities are considered to be elements of one or more **related-sets**. A related-set is a list of activities (and related information) which is dealt with as a single schedulable entity in MAESTRO with respect to the SELECT-PLACE-UPDATE cycle that MAESTRO executes. An activity may be the only component of a related-set or it may be one of several. A single activity may be represented in one or several different related-sets. The data regarding an activity that is specific to a related-set is stored in a structure called a **schedulable-activity**, which includes a list of **schedulable-subtask** structures. These structures store intermediate results of various calculations for use by the scheduler or user.

Related-sets are structures which contain a list of activity instances and another list of schedulable-activities, each of which is a structure containing a list of schedulable-subtasks,

which are themselves structures. These structures also contain other information relevant to the scheduling of related-sets.

There are several types of constraints MAESTRO represents that come under the general heading "resources". These include rate-controlled resources, quantity-controlled resources or consumables, and conditions. **Rate-controlled resources** are those whose availability returns at the time a subtask using them ends. In our cake-baking example, the person doing the mixing, measuring, etc. is available to do something else the moment they finish those tasks. **Consumables**, in contrast, remain depleted after the consuming subtask has ended, only becoming available when they are replenished by some other activity or event. The ingredients used to make the cake are examples of consumables. **Conditions** are used to model states of the environment or a system which have an impact on the execution of subtasks. In order to bake a cake, for example, the kitchen must be reasonably clean, warm and well lighted.

A **schedule** in MAESTRO is implemented as a flavor, and has instance variables in which to store various attributes of the schedule. The primary attributes are the **ACTIVITIES** instance variable, which stores activity-performance structures, and the **PERFORMANCE-INDEX** instance variable which stores pointers to activity performances sorted by start time.

A **schedule version** is a structure that stores a schedule and its associated resource database. MAESTRO maintains four different schedule versions. The **white** schedule version has on it all activities. The **red** version includes all activities with priority higher than 3, i.e. those with priority 0, 1, or 2. The **blue** version includes those activities with priority 0 or 1, and the **gold** version has only activities with priority 0.

C.3 Schedule Generation

The generation of a schedule begins with selection of the activities to be scheduled and specification of the duration of the schedule, or schedule period. After this initialization, the system builds a schedule by repeatedly executing what is called a **select-place-update** cycle. In this cycle a related-set is selected for scheduling, that set is placed somewhere on the schedule if possible, resource and conditions availabilities are updated, and depending on selection criteria, opportunity is again calculated for each related-set. The system quits executing this cycle when 1) there is no related-set that has opportunity to be scheduled, 2) the user halts the system via the user interface, or 3) the system finishes executing a user-selected number of cycles.

C.4 Non-Real-Time Contingency Handling

Contingency handling is an important part of MAESTRO's capabilities. The three main types of contingencies handled by MAESTRO are rate-controlled or consumable resources,

conditions, and temporal contingencies. Each is discussed in greater detail in subsequent paragraphs. In the event that resource availabilities are projected to be reduced below what is required for the activities on the white schedule (which can happen in several ways) while a schedule is being generated, the scheduler will execute a **select-remove-update** cycle, removing activities until there are no constraint violations. This involves selecting one performance of an activity involved in a constraint violation, uncheduling that performance, and updating resource and conditions profiles to reflect that uncheduling.

For resource availability changes, selection involves choosing among users of each over used resource. The function **handle-resource-contingencies** loops through resources that have constraint violations. For each it calls **make-relevant-performances** to make a list of performances that could be uncheduled to alleviate the problem. It then calls **set-performance-ratings** to evaluate each performance with respect to ten criteria. These parameters include 1) the base priority of the activity; 2) another priority factor which heavily weights life-critical activities, making them absolutely the last to be uncheduled; 3) a parameter called resource fit, which measures how well the use of an overbooked resource by the activity fits the amount of the overbooking, such that a 5-watt deficit is not fixed by uncheduling a 50,000-watt user, for example; 4) success, the ratio of performances scheduled to performances requested for the activity; 5) whether or not crew members are involved with the activity under consideration; 6) whether the activity is running or yet to be run; 7) if running, the percentage of the activity that has been completed; 8) whether the interrupted subtask is restartable; 9) whether the interrupted subtask is continuable; and 10) whether the activity constrains others such that they too would have to be uncheduled if it were. These criteria are normalized, multiplied with user-specified weightings, and combined to provide a rating for each activity performance. **handle-resource-contingencies** sorts performances according to these ratings, then calls **unschedule-performance** to interrupt or remove the lowest-rated performance. All performances are re-evaluated after each uncheduling so that the selection factors are appropriate to the schedule that results from each. The function exits when all constraint violations for all resources have been dealt with.

The function **unschedule-performance** will call **unschedule-one-activity** to unchedule a performance, if it has not begun as of effective-time (an input parameter that specifies when the new schedule should be in place at the resource subsystem), or will interrupt the performance, if it has begun.

When conditions availabilities are changed, all activities whose conditions requirements are not met must be uncheduled. In order to find and remove conditions constraint violations, the system has to inspect every performance of every activity on the schedule. The method **violation-intervals** finds intervals of time during which a subtask condition requirement is at odds with that condition availability. It calls **constraint-met**, which compares the condition requirement to that available by a method that is determined by the constraint type, of which there are 7. These include:

- EQ - the subtask need must equal the condition availability, either by being the same symbol or having the same numerical value;
- ONE-OF - the subtask specifies a list of possible values, one of which must equal the condition availability;
- NONE-OF - the subtask specifies a list of values, none of which can be the condition available in order for the constraint to be met;
- GREATER-THAN - the subtask need must be greater than the existing condition value;
- BETWEEN - the condition availability must be between the two values specified by the subtask need;
- OUTSIDE-OF - the condition availability must not be between the two values specified by the subtask need;
- LESS-THAN - the subtask need must be greater than the existing condition value. All activities in which any subtask has a constraint not met will be unscheduled or interrupted, if executing, by `unschedule-performance`.

Temporal contingencies are caused by the unscheduling of activities upon which others depend, and are resolved by unscheduling those unsatisfied activities. Since any number of activities may be linked by temporal constraints, `handle-temporal-contingencies` does not make a list of performances to unschedule. Rather, it finds one performance whose temporal constraints aren't met, calls `unschedule-performance`, then looks for another, continuing until it can find no performance whose temporal constraints aren't met. Only precedes, follows, starts and ends constraints are dealt with in this way, as they are the only constraints that can be violated by unscheduling a performance of an activity.

C.5 Real-Time Contingency Handling

Real-time contingency handling is more complex than that previously described. Typically real-time contingencies are caused by an immediate reduction in the availability of some resource that is being used by currently executing activities. In this case the scheduler must first update its representation of the state of the system such that interrupted activities are represented as such. It then must unschedule activities to be started in the near future which violate resource constraints. Finally, it will try to minimize the impact of the resource reductions, typically by finding ways to continue the interrupted activities.

All changes to a schedule that is being executed must be made in such a way that the changes can be communicated to the personnel and systems responsible for executing them.

This means that the scheduler must set a deadline for its own schedule manipulations and not make changes that come before this deadline. The deadline must be far enough in the future not only to allow the scheduler to finish, but also to allow time for communication of schedule changes to these various personnel and systems.

Resource providers can change the resources supporting an activity in ways that are incompatible with the rest of the schedule. This requires changing the description of resource use for the activity, then dealing with any resource overbookings that result. When all resource over use has been dealt with, the scheduler can try to find performances of activities which can be executed given the new resource availability and use profiles.

The main control structure used for real-time contingency handling is a loop that executes continuously, implemented in the method **contingency-driver**. This loop waits until the instance variable STATE is set to 'CONTINGENCY', at which time it calls the method **distribute-resource-effects** to take items off the contingency-handler data structure and put them on the appropriate contingency-driver queues. These items were sent to the contingency-handler data structure by the deblocking routines in the systems interface to KNOMAD. The system then evaluates a cond statement within the loop which is the implementation of prioritized command queues, evaluating each condition in turn and executing the code following the first one that evaluates to t. One queue or queue item is dealt with on each iteration of the loop. The conditions test for the presence of various contingency situations, including redundancy switches, resource allocation changes, load sheds, resource violations, continuation requests, and scheduling requests. After all these are dealt with considering timing considerations, STATE is set to 'IDLE', which stops the loop until the next contingency situation arises.

Appendix D

LLP / FRAMES Interface Control Document

Initialization List -

Direction - FRAMES to LLP

Description - An initialization list is sent to the LLP during initial startup or when the automation system wishes to reinitialize.

FIELD	LENGTH	FORMAT	DESCRIPTION
LLP Designator	2	Alphanumeric	Which LLP - 'A' to 'H'

Time List -

Direction - FRAMES to LLP

Description - Time synchronization message for distributed software system.

FIELD	LENGTH	FORMAT	DESCRIPTION
Month Now	2	Numeric	Calendar/clock month
Day Now	2	Numeric	Calendar/clock day
Year Now	2	Numeric	Calendar/clock year
Hour Now	2	Numeric	Calendar/clock hour
Minute Now	2	Numeric	Calendar/clock minute
Second Now	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

SSM/PMAD Final Report

Event List -
 Direction - FRAMES to LLP
 Description - A list of events from the Load Enable Schedule for operation of the breadboard.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	8	Numeric	Effective time of the event list
Number of Events	2	Packed79	Number of Events
EVENT	22	GROUP	AN EVENT DESCRIPTOR
Time of Event	8	Numeric	Time event is to be initiated
Component	3	Alphanumeric	Identity of component
Event	1	Alphanumeric	F-off, N-on, C-change
Type of Event	1	Alphanumeric	Always N-Normal
Redundancy	1	Alphanumeric	Y-Redundant, N-Not Redundant
Switch to Redundant	1	Alphanumeric	Y-Permission, N-No Permission
Maximum Current	3	Numeric	(0-999) deciAmps
Minimum Current	3	Numeric	(0-999) deciAmps
Maximum Budgetted Current	3	Numeric	(0-999) deciAmps

Priority List -

Direction - FRAMES to LLP

Description - Relative Switch Priority list for switches in an LLP.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	8	Numeric	Effective time of the priority list
Number of Components	2	Packed79	Number of components
Component	3	Alphanumeric	Identity of component

SSM/PMAD Final Report

Contingency Event List -

Direction - FRAMES to LLP

Description - A new Event List sent in response to a contingency situation.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	8	Numeric	Effective time of the event list
Number of States/Events	2	Packed79	Number of (States + Events)
STATE	22	GROUP	A STATE DESCRIPTOR
Time of Event	8	Numeric	Always 00000000
Component	3	Alphanumeric	Identity of component
Event	1	Alphanumeric	F-off, N-on, C-change
Type of Event	1	Alphanumeric	N - Normal, M-Manual
Redundancy	1	Alphanumeric	Y-Redundant, N-Not Redundant
Switch to Redundant	1	Alphanumeric	Y-Permission, N-No Permission
Maximum Current	3	Numeric	(0-999) deciAmps
Minimum Current	3	Numeric	(0-999) deciAmps
Maximum Budgetted Current	3	Numeric	(0-999) deciAmps
EVENT	22	GROUP	AN EVENT DESCRIPTOR
Time of Event	8	Numeric	Time event is to be initiated
Component	3	Alphanumeric	Identity of component
Event	1	Alphanumeric	F-off, N-on, C-change
Type of Event	1	Alphanumeric	Always N-Normal
Redundancy	1	Alphanumeric	Y-Redundant, N-Not Redundant
Switch to Redundant	1	Alphanumeric	Y-Permission, N-No Permission
Maximum Current	3	Numeric	(0-999) deciAmps
Minimum Current	3	Numeric	(0-999) deciAmps
Maximum Budgetted Current	3	Numeric	(0-999) deciAmps

Switch Control List -

Direction - FRAMES to LLP

Description - A switch command list which is executed immediately. This list is used for immediate source reduction load shedding, fault isolation switch manipulation, and manual intervention.

FIELD	LENGTH	FORMAT	DESCRIPTION
Effective Time	8	Numeric	Effective time of the event list
Number of Events	2	Packed79	Number of Events
EVENT	22	GROUP	AN EVENT DESCRIPTOR
Time of Event	8	Numeric	Not Used
Component	3	Alphanumeric	Identity of component
Event	1	Alphanumeric	F-off, N-on, C-change
Type of Event	1	Alphanumeric	F-Fault Isolation H-Hold until Contingency List M-Manual Control R-Release Manual Control
Redundancy	1	Alphanumeric	Y-Redundant, N-Not Redundant
Switch to Redundant	1	Alphanumeric	Y-Permission, N-No Permission
Maximum Current	3	Numeric	(0-999) deciAmps
Minimum Current	3	Numeric	(0-999) deciAmps
Maximum Budgetted Current	3	Numeric	(0-999) deciAmps

Switch Conversion Constants List -

Direction - FRAMES to LLP

Description - A switch conversion constant list allows the user to tweak the conversion constants used for determining amperage through an RPC.
(RBIs presently do not have amperage sensors.)

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Constants	4	Integer	No. of new switch conversion constants
CONSTANT	11	GROUP	CONSTANT DESCRIPTOR
Component	3	Alphanumeric	Identity of switch getting new conversion constants
Slope	4	Integer	Value in microAmps
Intercept	4	Integer	Value in microAmps

A/D Sensor Conversion Constants List -

Direction - FRAMES to LLP

Description - An A/D sensor conversion constant list allows the user to tweak the conversion constants used for determining amperage, voltage, and temperature through a given sensor set.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Constants	4	Integer	No. of new sensor set conversion constants
CONSTANT	35	GROUP	CONSTANT DESCRIPTOR
Component	3	Alphanumeric	Identity of sensor set getting new conversion constants
I-Slope	4	Integer	Value in mAmps
I-Intercept	4	Integer	Value in mAmps
V-Slope	4	Integer	Value in mVolts
V-Intercept	4	Integer	Value in mVolts
P-Slope	4	Integer	Value in mWatts
P-Intercept	4	Integer	Value in mWatts
T-Slope	4	Integer	Value in mDegrees
T-Intercept	4	Integer	Value in mDegrees

SSM/PMAD Final Report

Query List -

Direction - FRAMES to LLP

Description - A query list may be sent to the LLP to ask for switch status, sensor status, temperature sensor status, configuration data, switch conversion constant data, or sensor conversion constant data.

FIELD	LENGTH	FORMAT	DESCRIPTION
Query type 1		Alphanumeric	A-A/D Conversion Constants C-Configuration F-Offgoing Switch Status I-Incipient / Soft Fault Query N-Ongoing Switch Status Q-Quiescent State Request R-Switch Status S-Sensor Status T-Temperature Sensor Status W-Switch Conversion Constants X-Expunge (Clear) Soft Fault
Component 3		Alphanumeric	Only used for Query type 'F' , 'N' , 'X'

Switch Status List -

Direction - LLP to FRAMES

Description - A switch status list is sent to FRAMES when a fault occurs and also in response to a query or switch command list. In addition, this list is sent to FRAMES whenever a switch changes position.

FIELD	LENGTH	FORMAT	DESCRIPTION
Switch number	4	Integer	Used in fault isolation only
Anomalous	1	Alphanumeric	Y-Any fault or warning set N-All OK (nothing set)
LLP Flags	4	Integer	Bit Defined
bit 0		bit	Quiescent
bit 1		bit	Acknowledge
bit 2 - bit 31		bit	Not Used
Number of switches	4	Integer	Number of switches
SWITCH STATUS	18	GROUP	SWITCH STATUS DESCRIPTOR
Switch component	4	Integer	Identity of switch
Switch position	1	Alphanumeric	F-off, N-on, T-trip'd, U-Unavailable
Switch hold type	1	Alphanumeric	N-Normal, M-Manual, H-Contingncy
Status word	4	Integer	Bit Defined (True if set)
bit 0		bit	Anomalous flag
bit 1		bit	Surge Current Trip
bit 2		bit	Over Current Trip
bit 3		bit	Under Voltage Trip
bit 4		bit	Ground Fault Trip
bit 5		bit	Over Temperature Trip
bit 6		bit	Fast Trip
bit 7		bit	Already tripped flag
bit 8		bit	Already on flag
bit 9		bit	Already off flag
bit 10		bit	Soft Fault Flag
bit 11		bit	Not Used
bit 12		bit	SIC not present flag
bit 13		bit	Generic Card not present flag
bit 14		bit	Not enough power available flag
bit 15		bit	Could not schedule flag
bit 16		bit	Switched to Redundant flag
bit 17		bit	Switch has been shed flag
bit 18		bit	Unable to Command flag

SSM/PMAD Final Report

bit 19		bit	Current Overrange flag
bit 20		bit	Out of Current limits flag
bit 21		bit	Over Temperature warning flag
bit 22 - bit 31		bit	Not Used
Amperage	4	Integer	Current through switch (dAmps)
Trip Tag	4	Integer	0 - Ignore Number-Number of Trip for LLP

Sensor Status List -

Direction - LLP to FRAMES

Description - A sensor status list is sent to FRAMES when a fault occurs and also in response to a query list. In addition, this list is sent to FRAMES on a temporal basis.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of sensor sets	4	Integer	Number of sensor sets
SENSOR SET	20	GROUP	SENSOR SET DESCRIPTOR
Sensor set component	4	Integer	Identity of sensor set
Amperage	4	Integer	Amperage reading (dAmps)
Voltage	4	Integer	Voltage reading (Volts)
Power	4	Integer	Power reading (Watts)
State	4	Integer	Bit defined
bit 0		bit	Amperage out of Range
bit 1		bit	Voltage out of Range
bit 2		bit	Power out of Range
bit 3 - bit 31		bit	Not Used

SSM/PMAD Final Report

Temperature Sensor Status List -

Direction - LLP to FRAMES

Description - A temperature sensor status list is sent in response to a query list.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of temperature sets	4	Integer	Number of temperature sets
TEMPERATURE SET	8	GROUP	TEMPERATURE SET DESCRIPTOR
Temperature set component	4	Integer	Identity of temperature set
Temperature	4	Integer	Temperature reading (degrees)

Switch Performance List -

Direction - LLP to FRAMES

Description - A switch performance list is sent to FRAMES whenever a switch changes position and on a temporal basis.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of switches	4	Integer	Number of switches
AMPERAGE DATA	28	GROUP	A SWITCH PERFORMANCE DESCRIPTOR
Component	4	Integer	Identity of Switch
Start time	4	Integer	Start of Performance Interval (sec)
End time	4	Integer	End of Performance Interval (sec)
Average current	4	Integer	Time based averaged current (dAmps)
Maximum current	4	Integer	Maximum interval current (dAmps)
Minimum current	4	Integer	Minimum interval current (dAmps)
Maximum time	4	Integer	Time of Max. current reading (sec)
Minimum time	4	Integer	Time of Min. current reading (sec)

Sensor Performance List -

Direction - LLP to FRAMES

Description - A sensor performance list is sent to FRAMES on a temporal basis.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of sensors	4	Integer	Number of sensors
SENSOR PERFORMANCE DESCRIPTOR			
SENSOR	52	GROUP	SENSOR PERFORMANCE DESCRIPTOR
Component	4	Integer	Component Identifier
Start time	4	Integer	Start of performance interval (sec)
End time	4	Integer	End of performance interval (sec)
Average Voltage	4	Integer	Time based average voltage (Volts)
Maximum Voltage	4	Integer	Maximum interval voltage (Volts)
Minimum Voltage	4	Integer	Minimum interval voltage (Volts)
Average Current	4	Integer	Time based average current (dAmps)
Maximum Current	4	Integer	Maximum interval current (dAmps)
Minimum Current	4	Integer	Minimum interval current (dAmps)
Average Power	4	Integer	Time based average power (Watts)
Maximum Power	4	Integer	Maximum interval power (Watts)
Minimum Power	4	Integer	Minimum interval power (Watts)
Energy Consumed	4	Integer	Energy Consumed (Watt - Hours)

Switch Conversion Values List -

Direction - LLP to FRAMES

Description - A switch conversion constant list allows the user to see the present values of the conversion constants used for determining amperage through an RPC.
(RBIs presently do not have amperage sensors.)

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Constants	4	Integer	No. of new switch conversion constants
CONSTANT	12	GROUP	CONSTANT DESCRIPTOR
Component	4	Integer	Identity of switch getting new conversion constants
Slope	4	Integer	Value in microAmps
Intercept	4	Integer	Value in microAmps

SSM/PMAD Final Report

A/D Sensor Conversion Values List -

Direction - LLP to FRAMES

Description - An A/D sensor conversion constant list allows the user to see values of the conversion constants used for determining amperage, voltage, and temperature through a given sensor set.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of Constants	4	Integer	No. of new sensor set conversion constants
CONSTANT	36	GROUP	CONSTANT DESCRIPTOR
Component	4	Integer	Identity of sensor set getting new conversion constant.
I-Slope	4	Integer	Value in mAmps
I-Intercept	4	Integer	Value in mAmps
V-Slope	4	Integer	Value in mVolts
V-Intercept	4	Integer	Value in mVolts
P-Slope	4	Integer	Value in mWatts
P-Intercept	4	Integer	Value in mWatts
T-Slope	4	Integer	Value in mDegrees
T-Intercept	4	Integer	Value in mDegrees

Switch / Sensor Configuration List -

Direction - LLP to FRAMES

Description - This list tells the requestor the configuration of the LLP. This list is sent during initialization and in response to a query list.

FIELD	LENGTH	FORMAT	DESCRIPTION
Sensors Available	1	Alphanumeric	Y-Available, N-Not available
Number of switches	4	Integer	Number of switches
SWITCH	6	GROUP	SWITCH DESCRIPTOR
Switch Number	4	Integer	Identity of switch
Switch type	1	Alphanumeric	1-1 kW RPC, 3-3 kW RPC, R-RBI U-Unavailable S-Unavailable (SIC) *
Switch Position	1	Alphanumeric	F-off, N-on, T-Tripped, U-Unavailable

*Switch type will be set to 'S' only for switch number 0 or 14 if the SIC is unavailable on Bus A or Bus B respectively.

SSM/PMAD Final Report

Quiescent Status Message -

Direction - LLP to FRAMES

Description - This message is used to inform FRAMES of a fault in progress at the LLP software level. This message is also sent in response to a quiescent query when the LLP software has reached a quiescent state.

FIELD	LENGTH	FORMAT	DESCRIPTION
Quiescent Status 1		alphanumeric	T - LLP has reached quiescent state. F - LLP has a fault in progress.

Sensor Balancing Status List -

Direction - LLP to FRAMES

Description - A sensor balancing status list is sent to FRAMES when requested.

The sensor balancing is used to find components which do not meet Kirchoff's current law.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of records	4	Integer	Number of records (switches + sensors)
Number of switches	4	Integer	Number of switches
Number of sensors	4	Integer	Number of sensors
SWITCH SET	12	GROUP	SWITCH SET DESCRIPTOR
Component	4	Integer	Switch Component ID
Short term avg. current	4	Integer	Component avg. current
Short term avg. voltage	4	Integer	Component avg. voltage
SENSOR SET	12	GROUP	SENSOR SET DESCRIPTOR
Component	4	Integer	Sensor Component ID
Short term avg. current	4	Integer	Component avg. current
Short term avg. voltage	4	Integer	Component avg. voltage

Load Shed Status List -

Direction - LLP to FRAMES

Description - A load shed status is sent up when a load is "cheating" against the schedule. The load is shed at the LLP and this message is sent.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of sheds	4	Integer	Number of shed switches
LOAD SHED SET	8	GROUP	LOAD SHED SET DESCRIPTOR
Component	4	Integer	Component ID
Status Word	4	Integer	Component Status Word

Soft Fault Status List -

Direction - LLP to FRAMES

Description - A Soft Fault Status list is sent to the FRAMES system when a load exceeds its scheduled current limit, but not its maximum operational current.

FIELD	LENGTH	FORMAT	DESCRIPTION
Number of soft faults	4	Integer	Number of soft fault records
SOFT FAULT SET	16	GROUP	SOFT FAULT SET DESCRIPTOR
Component	4	Integer	Component ID
Scheduled Current	4	Integer	Scheduled Current (dAmps)
Maximum Operational Current	4	Integer	Maximum Operational Current (dAmps)
Current	4	Integer	Present Current (dAmps)

Appendix E

Bugs and Their Disposition

- 1. Problem:** [CLOSED] 1 JUN 92 [CHRIS]
When a fault occurs that is not diagnosable then the faulted state for the out of service switch is set to "OT" and this is wrong that value should be set to "??"

Disposition:
Fixed the case statement that used the default as OT to now have the default be ??.
Files Modified: UI-LABELS.CL
- 2. Problem:** [CLOSED] 01 JUN 92 [MIKE]
Had a temporary fault on a 3K switch: The load center switches supposedly weren't testable but this is not the case for the 1K switches. This still needs to be run to prove that it still happens.

Disposition:
This turned out to be a problem in the rule group that was looking for "Y" and "N" when the data was really "T" and "NIL". This has been changed and tested in the system. Files Modified: MF-DIAGNOSIS.RG.
- 3. Problem:** [CLOSED] 01 JUN 92 [CHRIS]
When taking an upper level switch out of service it may be nice to somehow indicate that the lower level switches are no longer usable as well.

Disposition:
When a 3K becomes out of service all the 1K's below it get made unavailable and the switch will turn an army green color. This change has been tested and implemented into the system. Files Modified: PROPOGATE.CL UI-DOMAIN.CL, UI-FUNCTIONS.CL, UI-ICONS.CL, UI-LLPS.CL, and UI-REGIONS.CL

4. **Problem:** [CLOSED] 16 MAR 92 [CLOSED]

It would be nice if data from the LLP's came up faster than every 30 seconds say every 10 or 15 seconds. To do this we have to look at how much free time the CPU at this time to determine that the increase in data will not bog down the system.

Disposition:

Data is still sent up every 30 seconds but now if a switch changes positions the data is sent up immediately to reflect the change. Files Modified: some files in the LLP code I need to ask chris for this.

5. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

Data base write function has the name of the index file hardcoded and this should not be done.

Disposition:

This change has been implemented and incorporated into the system. Files Modified: DB-FUNCTIONS.CL and SCHEDULE-FUNCTIONS.CL

6. **Problem:** [CLOSED] 01 JUN 92 [JANET]

In the activity editor the user needs the ability to delete Activities, Subtasks, Powered Equipment and Requirements.

Disposition:

This task has been given to Janet Weinz as of 03 MAR 92. It was finished on 4 APR 92 and has been tested. This delete capability is new to the system. Files Modified: ACTIVITY-EDITOR.CL and WORKBOXES.CL.

7. **Problem:** [OPEN Priority 1] 15 OCT 91 [JANET]

In the Activity Editor the user needs the capability of deleting a mode from a piece of powered equipment.

Disposition:

This task has been given to Janet Weinz as of 03 MAR 92.

8. **Problem:** [CLOSED] 01 JUN 92 [JANET]

The activity editor still needs to have it's help written.

Disposition:

This task has been finished and implemented and tested in the system. Files Modified: ACTIVITY-EDITOR.CL

9. **Problem:** [CLOSED] 01 JUN 92 [JANET]

What should the system do if the user reads in a schedule that uses a switch that has been taken out of service or for that matter the switch does not exist in the system. This causes the FELES and LPLMS to send down information on that particular switch to the LLP's. The FELES command for that switch in question will be ignored but it is not known what the LLP's will do with the bogus data from LPLMS. The LLP's may be fixed to handle this but we need to answer the bigger picture of what to do at the system level.

Disposition:

A change was made to call a function which produces a workbox showing the switches that are not available for the schedule being activated. This will then send MAESTRO the list of switches that are unavailable for contingency rescheduleing. Files Modified: PLANNER-PROCESSES.CL and PLANNER-SUPPORT.CL.

10. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Priority list is wrong after seizing manual control of a switch. This caused the LLP's to shed the wrong switch due to a power constraint.

Disposition:

Mike Elges re-wrote the LPLMS code to correctly calculate the priority lists. The old list used the activity's time hack to determine system time hacks that required the other activities to have start times on that hack in order to be included in the priority list for that particular time. Manual Seizure events happened at random times so those time hacks only had the manual events in that time even though other activities span that time hack. Code was changed to ensure all switches that span the time hacks were included in that time hack. Also changed how the time hacks were calculated by making sure that start times for subtask performances were included in calculating what time hacks to use to calculate priorities.

11. **Problem:** [CLOSED] 29 FEB 92 [MIKE AND JOEL]

Turn on and off manually controlled switches.

Disposition:

Code written by Joel Riedesel and tested by Mike Elges appears to function correctly. No anomalies were seen in doing this in normal mode of operations.

12. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

Each LLP has its own drawing semaphore. It has been noticed that in the past that the power system screen would have what appeared to be unexplained colors on the screen when updates were made to the power system screen. Making a semaphore for

each LLP does nothing to synchronize X events to the console.

Disposition:

Chris Myers made one semaphore for all the LLP's. This semaphore is now a class allocation on the LLP class description.

13. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Redundancy icons show up approximately 1 minute after the switch has been turned on.

Disposition:

Mike Elges modified the monitor-processes.cl to fix two problems. The first was to change the monitor processes from looking at the switch status and looking for closed to assert the the frame to put the icon on the screen. This is because the switch is closed at the same moment that the monitor updates the event that asserts the information to have the icons drawn. The data from the llp's had not yet arrived to tell the user that the switch was closed so the icons would not show up. The current fixes only check that the switch is out there and available and assumes that it will be closed because it is scheduled to be closed. This assumption is valid because the system uses this model to follow the state of the hardware. The second change was made to the monitor-processes.cl to prevent the monitor from updating the each event twice. A logic error was detected that should that each event was updated twice once at it's scheduled time and one more time one minute later. This is why the icons should up one minute late because by then the data had been asserted to show that the switch was in a closed position. One problem found two bugs!!!!

14. **Problem:** [CLOSED] 29 FEB 92 [JOEL]

Contingency re-scheduling does not does not work correctly with the MAESTRO we are running on the Solbourne.

Disposition:

Joel Riedesel found two errors with this code. The first was determining which activities had been effected by the contingency was in error due to porting from the symbolics. The second error had to do with making alternate models on the fly generated numbers as the new id for activities etc, but the ssmpmad system was looking for symbols. This was changed to generate symbols and has been tested and appears to work correctly.

15. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Manual Seizure of events does not work correctly.

Disposition:

There were many errors in this code that were worked out during testing. All were not documented but they ranged from calling a non existing function to passing the wrong data from routine to routine. These changes were made by both Mike Elges and Chris Myers.

16. **Problem:** [CLOSED] 29 FEB 92 [MIKE and CHRIS]

Manual Release did not work correctly.

Disposition:

This problem was very similar to the one on Manual Seizure where the code was written by Joel Riedesel but never tested. So while testing the code many minor bugs were found and corrected by Mike Elges and Chris Myers. Currently this code is working.

17. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

The detailed data from the power system screen causes the system to crash.

Disposition:

This was due to the changing the size of the scratchpad window to accommodate the new screen layout to deal with message lists. The width of the scratchpad was now too small for some of the bitmaps that were being displayed in detailed data and this caused the system to crash. This was fixed by Chris Myers who changed the size and position of the bitmaps to fit within the scratchpad area.

18. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

Detailed data from a point down on sensors crashes the system.

Disposition:

This was caused by the algorithm to calculate the sensors from a sensor down. This was fixed by Chris Myers and tested for the current system.

19. **Problem:** [CLOSED] 29 FEB 92 [MIKE and CHRIS]

On bringing up a dumped lisp the icons colors changed.

Disposition:

The fix to this is only partial at this point. The run-startup-functions will create the icons after the dump lisp has been brought up. But the object was to make start-up very fast and the creation of the icon doubles the time in doing this so it needs to be fixed outside the start-up functions. Franz has a hack that may be usable but Chris Myers needs to look at it and try to find a place to implement it.

20. Problem: [CLOSED] 29 FEB 92 [CHRIS]

On bringing up a dump lisp the scroll menus are all blank except for the activity editor which are created in the run-startup-functions. The workboxes are also all blank.

Disposition:

It was determined that because we relied on the server doing the backing store that when you bring up our dumped lisp which is a client that the server does not have the information of the data in the scrolling menus or the workboxes to restore them on an expose event. Chris Myers changed both the workbox code and the scrolling menu code to do their own backing store. These changes have been implemented but have not been fully tested at this time. When the repaint methods were added to the scrolling menu code this took out the need for having the second menu for the system backing store. The reason we had this menu in the first place is still a mystery to me.....

21. Problem: [CLOSED] 11 AUG 92 [CHRIS]

The highlight and dehighlight are incorrect on some of the static menus. This seems to be that multiple events where being generated on entering or exiting the ctive region. Because the same routine handled both the in and out events and it just inverted the region two in's without an out screwed up the menu items.

Disposition:

Chris changed the one function into two functions that change the color of the active region versus just inverting the current color of the active region. This change has been designed, implemented and test. The affected files are: CLASSES.CL, NEWSYS-FLOW.CL and WORKBOX.CL

22. Problem: [CLOSED] 11 MAR 92 [MIKE]

When going to the FELES the active schedule does not get updated properly for unknown reasons at this time.

Disposition:

Whenever the user left normal mode to go into maintenance or idle mode the system reset the active schedule with a new one. This caused problems for the FELES code because it assumes that the active schedule will always be the same so when it checked the active schedule with the current schedule being displayed and they did not match it assumed that the schedule was an alternate schedule being reviewed by the user so there was no reason to update it. This is wrong! The fix was to change the the internal not to create a new active schedule every time the user drops out of normal mode.

23. Problem: [CLOSED] 29 FEB 92 [JANET]

Can only show one day of time in the FELES screen.

Disposition:

This was a pre-existing constraint because the graphtool code could not label the axis correctly if the time was greater than that. A fix was done by Janet Weinz to allow the time to extend from minutes to 99 days. This change was tested and incorporated into the system. This caused a problem in the power util screen with the labels overwriting themselves. This meant that the algorithm needed to be re-thought and Janet Weinz has since finished the design and implemented the changes. The changes have been incorporated into the system and partially tested.

24. **Problem:** [CLOSED] 29 FEB 92 [JANET]

When selecting: read or activate a schedule the selection caused a menu to be popped up but we are in a scrolling menu that can handle pull-rights.

Disposition:

Janet Weinz made these menus pull-rights to the activate and read schedule menu items of the pull-right menu schedule option of the system options menu. This change has been tested and incorporated into the system.

25. **Problem:** [CLOSED] 29 FEB 92 [JANET]

In the activate schedule code it requires that the user read the schedule in before the schedule can be activated. This code should read the schedule in for the user and not just exist.

Disposition:

Janet Weinz modified the activate schedule code to read the schedule in if it had not been read in before the request to activate it. This has been implemented into the system.

26. **Problem:** [CLOSED] 29 FEB 92 [JANET]

The active regions that make up the FELES's bars do not update the status the system status line on what mouse button options are available to them.

Disposition:

Janet Weinz Created an after method on the active regions that updated the system status line. This code has been implemented and tested.

27. **Problem:** [CLOSED] 29 FEB 92 [JANET]

The scroll bar is incorrect the first time a schedule is displayed in FELES until the users scrolls for the first time. The scroll bar will also be wrong every time the gantt chart is recomputed in FELES.

Disposition:

A fix was made by Janet Weinz and incorporated but this is still a problem. The next solution was to make the graph code insure that the graph always comes out at the top. This insures that the scroll bar is correctly sized for the data. This change was done by Chris Myers and has been incorporated into the system and partially tested.

28. Problem: [CLOSED] 29 FEB 92 [MIKE]

When removing the last item from many lists in the activity editor caused the asserted data to be wrong and this would crash the system.

Disposition:

This problem was caused by not taking into account removing the last item from a list and then consing nil onto nil which creates a list with nil init versus what we wanted, just a nil. The was fixed by Chris Myers and incorporated into the system. This change was partially tested to date.

29. Problem: [CLOSED] 01 JUN 92 [MIKE, CHRIS, and LANCE]

When grabbing more than one switch for manual seizure the next workbox comes up and is blocked while still processing the data from the previous workbox. This workbox should not appear until all the data has been processed by the previous workbox.

Disposition:

ILA has been re-written to be more functional and allow the grabbing of mutiple switches at one time so this problem has been resolved. Files Modified: PLANNER-PROCESSES.CL, PLANNER-SUPPORT.CL, PROCESSES.CL, SSMPMAD-IF.CL, CLASSES.CL, ILA-DATA-DATA-FIELD.CL, UI-DOMAIN.CL, UI-FUNCTIONS.CL, UI-LABELS.CL, UI-LLPS.CL, UI-REGIONS.CL, SCROLL-MENU.CL WORKBOX.CL and WORKBOXES.CL

30. Problem: [CLOSED] 01 JUN 92 [MIKE]

When grabbing an already manually seized switch in normal mode the workbox gives an erroneous error "Handling Contingency" this is not the reason. The real reason is that the switch is under manual control already. Also the error messages appear in two different places the first is the scratchpad of the power system screen and the other is in the workbox error message area. Well first it should only be in one place second the workbox is onto of the scratch pad so the user cannot see the message ant way!!!

Disposition:

ILA has been re-written to be more functional and allow the grabbing of mutiple switches at one time so this problem has been resolved. Files Modified: PLANNER-PROCESSES.CL, PLANNER-SUPPORT.CL, PROCESSES.CL, SSMPMAD-IF.CL,

CLASSES.CL, ILA-DATA-DATA-FIELD.CL, UI-DOMAIN.CL, UI-FUNCTIONS.CL,
UI-LABELS.CL, UI-LLPS.CL, UI-REGIONS.CL, SCROLL-MENU.CL WORKBOX.CL
and WORKBOXES.CL

31. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]
Manual Icons and for that matter any switch state change is lost when the user is on another application screens at the same time when the switch state change should be updated on the power system screen.
Disposition:
Chris Myers says this has been fixed and implemented as of now I have not tested it to confirm that this has been done.
32. **Problem:** [CLOSED] 16 MAR 92 [CHRIS]
When loading the system from scratch upon loading of the ui system sometimes the system will crash because the system is trying to repaint a workbox that has not yet been created?? This is an intermittent problem that needs to be studied.
Disposition:
This was fixed by Chris Myers. The thought was that in the method initialize-instance, the repaint :after method for the workbox was setup before the data that the repaint function needed was created. This data was created in the initialize-instance :after method. The story goes the a repaint event was sometimes received prior to the creation of all the objects that the repaint method needed. Why this is done no one knows and also this is an intermittent problem. So some type of race condition must be being setup. The fix was not to assert the repaint method until after creating all the data that is needed by the repaint method. This change has been made and tested. The problem seems to have gone away. These changes have not yet been incorporated into the system.
33. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]
Power Util Component Menu crashes when items are deleted from it.
Disposition:
This was caused by making the extent negative when deleting menu items from a menu. Changed the code to guarantee a positive extent. This code has been implemented but not tested thoroughly.
34. **Problem:** [CLOSED] 29 FEB 92 [CHRIS and MIKE]
Power Util screen crashes when it tries to draw the data to the screen

Disposition:

Lots of bugs found in GRAPHTOOL due to no one tested code for initial delivery. To many to mention here the fixes have been made and implemented and to date the code seems to function normally.

35. Problem: [CLOSED] 29 FEB 92 [CHRIS and MIKE]

Change window of time workbox crashes for the power util screen whenever a cancel or a waveoff is done.

Disposition:

The code that called the workbox was looking for a nil value to be returned in the case of a cancel or wave-off. The workbox was returning a :cancel in that case and the routine went on and tryed to use that as switch causing the crash. Chris Myers has designed and implemented a change to fix this problem. The change has been incorporated into the system and has been fully tested.

36. Problem: [CLOSED] 29 FEB 92 [JANET]

Repaint method for the workbox code puts the text for the ASAP button in two places and they overlap.

Disposition:

The repaint method for the workbox had incorrect location for the ASAP button causing the word "ASAP" to be written in the wrong location. This was fixed by Janet Weinz and incorporated into the system. This change has been fully tested.

37. Problem: [CLOSED] 29 FEB 92 [CHRIS]

Power util should not be available in Maintenance and Idle mode. I need to know why this is a problem.

Disposition:

There is no useful information for the user unless they are in normal mode so it makes sense not to let the user go into power utilization when the user is not in normal mode. This change was designed and implemented by Chris Myers and has been incorporated into the system. this change has been partially tested.

38. Problem: [CLOSED] 03 APR 92 [JANET]

Changing the redundant field of a mode for a piece of powered equipment should make all schedules that use that piece of powered equipment in that mode need to be re-scheduled.

Disposition:

Changes were made to ensure that all changes made in the activity editor would be propagated to affected schedules.

39. **Problem:** [CLOSED] 29 FEB 92 [JOEL]
Source Power change code stopped working
Disposition:
New code has been implemented by Joel Riedesel but not fully tested.
40. **Problem:** [CLOSED] 29 FEB 92 [MIKE and JOEL]
Apex code does not work with MAESTRO being ported to the Solbourne.
Disposition:
This code has been fixed by Joel Riedesel but has not been tested. Testing will be very difficult to do without writing some lisp on another machine that would mimic what Lewis Research Center would send us. CLOSED because we don't project to be working with APEX.
41. **Problem:** [CLOSED] 29 FEB 92 [CHRIS and MIKE]
Workbox sometimes goes away as soon as it is activated. This is probably caused by the event queue having more than one entry in it. How this is caused is not clear at this time but may be due to the fact that the user double clicks to deactivate at some time and this causes the extra event to get generated that sticks around. I don't know but this needs to be looked into.
Disposition:
If the user double clicks fast enough X will generate another event that will be put into the event queue for workboxes. The next time a workbox is activated it will deactivate immediately because of the extra event. The solution is to clear any events in the workbox event queue to ensure that no old events cause problems. Chris Myers designed and implemented the change. This included adding a new function to the events.cl file to clear all events and to add this call to the get-workbox info code in workbox.cl. This change has been incorporated into the system and partially tested.
42. **Problem:** [CLOSED] 29 FEB 92 [MIKE]
The System Hangs on making a diagnosis on a bus failure. The bus failure happens after another 3k switch on that bus has been previously tripped.
Disposition:
The reason for this lies in the fact that on the bus failure the multiple fault rule group does not diagnose the bus going down because it notices that one of the switches (the

one previously faulted). Says it is on based on the event data the diagnosis code looks at. There were two reasons for this the first had to do with the fact that when MAE-STRO returned with the the reply the data was not getting propagated because the slot that we were looking at to propagate was misspelled in the planner-processes code so it always returned nil. Fixing this did not solve the whole problem because even though the event list was now right that did not mean that the system model would be updated. In the past when we updated twice for every event for the minute that it should occur and the next minute after for the most part this would ensure by luck that the system model updated. Now it was missed because the monitor process does not process events that have already occurred in the past and only does them the minute that they are suppose to. So updating the system model during a contingency was moved into the contingency processing realm. This made sure that the model was updated immediately when a switch was shed, switch to redundant or taken out of service. This work was done by Mike Elges and has been implemented but not fully tested.

43. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

The system will hangs in the mf-diagnosis code. The last rule it will fire will be mf-diag-15 and then it hangs. Which rule it is trying to fire I am not sure of. This was seen due to another error that has since been fixed. (That error was the one just before this one)

Disposition:

The Rule Group MF-DIAGNOSIS has been thrown out and replaced with a lisp function. To date the behavior that generated this error has not been seen.

44. **Problem** [CLOSED] 29 FEB 92 [CHRIS]

The status word for an out of service 3k switch is being changed. This change is reflected the next time a fault occurs on that same LLP. This causes the interface to display incorrect data on the icon. This always is OT because of the way the data is calculated.

Disposition:

This was caused by the status word being reset when an open was sent down by the event list when all activities were terminated by the contingency re-scheduling that occurred when the 3k in question faulted. Now the user did not see the change until the next fault on that bus occurred which was the power going down on that bus this caused the need to redraw all the icons on that llp because now there was no power to them. When the icons are re-drawn they use the information in the data base about the switches state and now the status word is 0 because of the open sent down by the event list. Cause the out of service reason to be OT because that is the fall through reason of the case statement. (by the way this needs to be changed to something else

because OT is only for Over Temp so unknown reasons should come UK) Chris Myers change the Process-switch-data routine to check the state of the switch to see if it was out-of-service. If the switch is out of service then the status word would not be updated. This change has been implemented and partially tested.

45. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

Update the system to reflect the changes to support Message Lists.

Disposition:

Let Chris fill this in. Includes changes to: load-time.cl classes.cl planner-processes.cl and many more..... Other bugs have been reported on this update.

46. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Source power change no longer works

Disposition:

Due to evolution of the system the source power change became broken due to many parameters. The main one was there was no tested user interface to the system for it. Chris Myers has fixed this by first making it available via Other System Operation pull right of the system options menu. The other change done by Chris was to fix the workbox code to return the correct data as well as the calling sequence to call the source power change to send it the data in the format it was expecting. Other changes included fixing the source power change code itself and this work was done by Mike Elges and Joel Riedesel. These changes have been implemented and partially tested. There is one small problem that needs to be looked into. What should the code do if the source power change increases power not just decreasing it. The current system seems to ignore an increase in power and this is not right this information should be passed onto MAESTRO to see what MAAESTRO can do with the extra power.

47. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Need the ability to put a switch back into service.

Disposition:

This code was Written by Mike Elges to take a list of out of service switches test each switch and put those switches back into service that displayed no anomalies data. The user interface side of this function was written by Chris Myers and it exists in the Other System Options menu of the System Options main menu. This code has been completed but is still in the testing stage. There currently exists a problem of not calling the diagnosis code when the code turns on the 3k above a 1k to test the 1k and the 3k trips. When this happens the diagnosis code need to be informed that the 3k trip but it will not have an updated system model to reflect this because the put-switch

back in service code has done this for testing purposes. The short term solution is to see what happens if I try kicking off the fault diagnosis code and let it handle it.

48. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

Redundant icons are not removed when the switches get turned off. This can be seen if the user loads a schedule that needs redundant icons. If a switch to redundant happens because of a fault then when the schedule is halted the redundant icons stay up and do not go away.

Disposition:

This was fixed by having redundant icons and manual icons controlled at one central location and that is in the MONITOR processes. For the redundant icons we also added some code in the PROCESSES to handle the case of a contingency operation much more quickly. Files Modified: MONITOR-PROCESSES.CL and PROCESSES.CL

49. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

Why is there the same if-added on the LLP frame as well as all instances of the LLP frames????

Disposition:

It was discovered that when the if-added update-comm was added to the if-added list for the particular instances of the llp's that the routine that added the if-added had an oversight that when it did the match to get the current if-added's on that frame the match also returns the inherited if-added's. Well this is not a problem if if-added's had already resided on the individual instances of the llp's but none existed. This caused the only list of if-added to be returned were the inherited ones. Since the third field was not checked on the return data the user assumed the first element of the list of return elements was the if-added's for the particular llp. So when the new if-added was conseed on and reasserted the individual llp's now had all the functions of its parent if-added which meant for the same assertion all the functions got called twice.... This was fixed by Chris Myers by putting the update-comm function on the parent class to begin with. NOTE: other code that adds if added makes this same assumption and the rest of the system needs to be checked to fix this problem. This change was done in the system-flow.cl file and has been implemented and partially tested.

50. **Problem:** [CLOSED] 29 FEB 92 [MIKE]

Activity Editor shows incorrect results for powered equipment location after fault has occurred on a schedule that uses that activity.

Disposition:

The reason for this has to do with the fact that the activities in question have re-

dundant switches and when the 3k P04 was faulted the activities on those 1k's below changed to reflect going to redundant. The activities in the schedule were updated by the MAESTRO contingency handling program and the changes were incorporated into the system. This meant that the activities in question got alternate models generated for them. These alternate models for activates where than added to the activity groups using a cons which meant they were first now in the list. When the user then went into the activity editor and wanted to edit one of the activities the activity editor just grabs the first activity in the activity group. This is because the current activity editor does not allow more that one activity for a group so the concept of an activity group is hidden from the user. The activity chosen by the user is then displayed and it will be in error because it is the wrong one. This could lead to other problems so the solution is to make sure that the user always edits the last activity in the activities slot of the activity group being edited. The functions that write out the database already take this into account... This has not yet been implemented.

51. **Problem:** [CLOSED] 29 FEB 92 [MIKE and CHRIS]

Switch state change does not update correctly when on another application window and then transfer to the power system screen.

Disposition:

It was found that the switch current change was overwriting the state change data in the to-do-list for the power system screen. This was corrected by Chris Myers and Mike Elges and the changes have been implemented and partially tested.

52. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

LLP's Boxes frame is mouseable and it should not be.

Disposition:

This was done By Chris Myers but what he did to fix it is I don't know???

53. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

Delete menu item causes the the process to crash with -1 not valid for extent.

Disposition:

I do not know what caused this problem or how it was fixed. But the work was done by Chris Myers and the changes have been implemented but not fully tested.

54. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

When a bus would go down the system would diagnose an Under Voltage and then take the 3k's out of service. When the 3K's are taken out of service the there is now

no testing done on the switches. This means that the 1K's below are never cleared of their faults and they are left faulted in red on the interface. They need to be cleared after a diagnosis is made and they are not being put out of service.

Disposition:

It was determined that they were not being cleared because no testing needed to be done on the system because the diagnosis could be made with the current information. This meant no commands would be sent down to the lp's to open all the faulted switches which would clear there faulted status. To fix this problem when a switch is asserted out-of-service the system needs to clear all the faulted flags below the switch that has been put out of service. These changes have been implemented by Chris Myers and partially tested.

55. **Problem:** [CLOSED] 29 FEB 92 [MIKE and JOEL]

When the port bus is turned off the system makes an incorrect diagnosis because it thinks that the sensor data from AM0 is less than nominal, which it is, but that AM1 is reading nominal data but its value is 0 as is AM0.

Disposition:

A race problem was found between when data got asserted to the system on a collect snapshot and when the rule system was looking at it. This caused the rule system to make an incorrect diagnosis because the data had not yet been asserted. Thus the race condition. This was one of those situations where the problem was not always repeatable..

56. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

When opening switches during fault diagnosis the system will list all the possible switches in the cluster even though the system only has a subset of all the possible switches.

Disposition:

This has been fixed by Chris Myers but what caused this and how it was fixed has not been given to me as of yet.

57. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]

The soft fault window list Anomalous bit info for hard fault data.

Disposition:

The system did not check to see if the anomalous bits set were related to hard fault data. If they are do not display them to the soft fault window. This change was designed and implemented by Chris Myers. The changes have been incorporated the

system but not fully tested.

58. **Problem:** [CLOSED] 01 JUN 92 [CHRIS]
When getting sensor data from a point down on the power system screen the data displayed will show information on sensors that do not exist.
Disposition:
A change was made to prevent printing information on non-existing sensors in the system. This change has been test and implemented into the system. Files Modified: UI-FUNCTIONS.CL
59. **Problem:** [CLOSED] 01 JUN 92 [CHRIS]
Sensor names need to be more mnemonic like the switch names. They currently have names like AM0 when it should be something like Port Sensor 0 or something like that.
Disposition:
This has been done for the Power System screen and implemented and tested but it still needs to be done for the OUTPUT-DUAGNOSIS code. Problem 101 covers this need. Files Modified: UI-FUNCTIONS.CL
60. **Problem:** [CLOSED] 29 FEB 92 [CHRIS and MIKE]
The way the data is written to the screen for a diagnosis looks bad. This data needs to be formatted correctly.
Disposition:
The re-write of the mf-diagnosis from a rule-group to a lisp function fixed this problem. The file that contains these routines is now called OUTPUT-DIAGNOSIS and the main procedure is now called from the MF-CONTROL rule group.
61. **Problem:** [CLOSED] 01 JUN 92 [CHRIS]
The Number for the number of faults in the hard fault title bar is always zero even though many faults have occurred. This needs to reflect the number of faults that have occurred. The same goes for the source power change window title bar for the number of changes.
Disposition:
This change has been implemented and tested in the system. Files Modified: CLASSES.CL.
62. **Problem:** [CLOSED] 29 FEB 92 [CHRIS]
When selecting component power usage menu from the FELES application menus nothing appears in the scratchpad and it should.

Disposition:

The call to create the bitmap and send it to the scratchpad window was not being made. Chris Myers has changed the code to make the call and the changes have been implemented and tested.

63. Problem: [CLOSED] 11 AUG 92 [MIKE]

There seems to be an anomaly these days that after the user brings up the system there is no backing store on the system flow window or the title bars of the fault windows. This seems to be a problem with the servers backing store because it only happens intermittently.

Disposition:

This was determined to be a bad diode on the graphics board. This problem was only seen on the Solbourne. On upgrading the X server on the Solbourne this problem along with others became more of a problem and Solbourne sent us a new Graphics Board and the problems have not been encountered since. This is being closed because we are 99% sure this was the only cause and because the problem seemed to be intermittent.

64. Problem: [CLOSED] 29 FEB 92 [MIKE and CHRIS]

The system is crashing in update time in both the FELES and POWER UTILIZATION modules for some unknown reason.

Disposition:

The problem is that two processes are acting on the data at the same time. One is the update time module and the other is one that wants to change the type of data we are looking at. This causes the second process to flush the active regions that the first process is using. This first process (update-time) then crashes because active regions it is using have now been flushed.

65. Problem: [CLOSED] 01 JUN 92 [MIKE]

Message graph stuff has been left in the system even though it is no longer used.

Disposition:

This has been removed and tested. Files Modified: SSMPMAD-DOMAIN.CL

66. Problem: [CLOSED] 01 JUN 92 [MIKE]

Tripped all the 3K's and then tried to do a manual seize prior to putting the switch back into service. First of all, grabbing a 3K is illegal and second it is also illegal to manual seize a switch that is out of service. These two issues need to be addressed by the team and the customer to see if they should be changed.

Disposition:

With the new ILA code it is now permissible to grab any switch in the system as well as out of service switches. Files Modified: PLANNER-PROCESSES.CL, PLANNER-SUPPORT.CL, PROCESSES.CL, SSMPMAD-IF.CL, CLASSES.CL, ILA-DATA-DATA-FIELD.CL, UI-DOMAIN.CL, UI-FUNCTIONS.CL, UI-LABELS.CL, UI-LLPS.CL, UI-REGIONS.CL, SCROLL-MENU.CL WORKBOX.CL and WORKBOXES.CL

67. **Problem:** [CLOSED] 16 MAR 92 [CHRIS]

Output Diagnosis will Output a nil-nil for a switch in the diagnosis broken output cable of switch above. Also diagnosis retrip on flip did not take care of the case for the re-trip being an undervoltage.

Disposition:

Chris Myers has fixed these two problems in the the file OUTPUT-DIAGNOSIS.CL and these changes have been tested.

68. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

If a switch above is put back into service the routine must be smart enough to make sure that none of the switches below have mf-indication flags set. If this is the case, then they need to be taken out of service or at least tested to see what kind of effect that they have on the 3K. If it is not done then when the 3K is put back into service then the scheduler might turn on the suspect 1K and cause a trip.

Disposition:

The change to fix this problem resides in that any switch below a faulted switch is made unavailable so the user is forced to test them after the switch above has been put back into service. Files Modified: PLANNER-PROCESSES.CL, PLANNER-SUPPORT.CL, PROCESSES.CL, SSMPMAD-IF.CL, CLASSES.CL, ILA-DATA-DATA-FIELD.CL, UI-DOMAIN.CL, UI-FUNCTIONS.CL, UI-LABELS.CL, UI-LLPS.CL, UI-REGIONS.CL, SCROLL-MENU.CL WORKBOX.CL and WORKBOXES.CL

69. **Problem:** [CLOSED] 16 MAR 92 [MIKE and CHRIS]

If a user tried to place a switch back into service and it did not work then any attempt after that would result in an under voltage being reported back by the LLP's when the switch was closed during testing.

Disposition:

The problem was assigned to Chris Myers and it was determined that the testing code needed to send first an unconditional open to the switch to clear the old fault. If this was not done than the close command would not be executed by the LLP's. A change was made in the function test-switch to first send down an unconditional open to the

switch being tested. This change has been tested but not yet implemented into the system. Files modified: PLANNER-PROCESSES.CL

70. **Problem:** [CLOSED] 16 MAR 92 [CHRIS]

The ui system adds a lot of if-added's to the system and this is done by hand every where and none of them take into account the inheritance that might occur.

Disposition:

The problem was given to Chris Myers who wrote a function call add-if-added. This functions makes sure that the if added is placed on to a list of other if-addes for this particular frame and not the inherited frame. This change has been incorporated into the classes file and the subsequent modification to the files: UI-DOMAIN.CL ACTIVITY-EDITOR.CL UI-LLPS.CL CLASSES.CL UI-REDUNDANT.CL NEW-SYS-FLOW.CL FELES-UI.CL POWER-UTIL.CL.

71. **Problem:** [CLOSED] 16 MAR 92 [CHRIS]

The power util screens redraws all the plots every ten minutes to show the new utilization data. This is just dumb because all we need to do is added the new data onto the existing plot data.

Disposition:

This problem was given to Chris Myers. The fix was to change the way the is done. First the update is not done on a ten minute hack but whenever the data for switch or sensor performance data is received. Second the new way requires the data to be added to the plot data and this is done in the function process-switch-pereformance-data and process-sensor-performance-data. The graphtool.cl was modified to add a new function for plots called redraw-plot-lines. This function is called after the plot data has been updated. This change has been completed but not yet tested or incorporated into the system. Files Modified: POWER-UTIL.CL, SSMPMAD-IF-TR.CL, and GRAPHTOOL.CL.

72. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

The system needs to keep track of the total power available to the system for soft and incipient faults, ILA and source power-change code.

Disposition:

Changes where made to ensure that system integrity for power is maintained during operation. There were several holes in the system that allowed the user to go through a source power change and the switches would be shed and MAESTRO would be notified of the power restrictions but the resource profiles on the SSMPMAD side would not be updated allowing ILA to go off and grab switches when there was no power left in the

system. this has been changed with the only restriction left is that when the system transitions to another state or the active schedule is halted the system starts over with all the power levels reset to maximum. In the future this needs to be changed and this will require checking the current system power and if it does not meet the new scheduled requirements then a contingency rescheduling will need to be done around the time of activation of the new schedule. (This would be a lot easier if MAESTRO would just handle more than one schedule at a time.

73. **Problem:** [CLOSED] 01 JUN 92 [JANET]

Source Power change works when the amount of power in the system is reduced but it does not work when the power in the system is increased. This needs to be fixed.

Disposition:

Source power change has been expanded to allow changing power on each bus and now works in adding power to the system. This change has been implemented and tested in the system. Files Modified: SSMPMAD-IF-TR.CL.

74. **Problem:** [OPEN Priority 1] 11 MAR 92 [MIKE and CHRIS]

If the system has a broken 1K switch in such a way that the 1K is always closed. The problem is that the system needs to know that the switch is pulling power even though it is said to be open. The next problem to arise deals with shorting that broken 1K. When this is done the 3K above goes on a fast trip because the 1K is broken. This causes a bad diagnosis on our part because we think there is something wrong with the 3K switch not the real culprit the broken 1K that now has a shorted load on it.

Disposition:

The LLP's need to make sure to send up all the data on a switch even when it is open and let the soft or incipient process look at this data and make a decision about what may be going on here. Also the hard fault diagnosis code needs to look at these observations of the soft and incipient fault process in order to correctly diagnose the problem in the system.

75. **Problem:** [CLOSED] 16 MAR 92 [MIKE]

Common Lisp 4.1.beta the backdoor capability no longer works.

Disposition:

Due to the Lisp-Emacs Protocol changing from 4.0 to 4.1 the old backdoor hack set up by Joel Riedesel no longer worked correctly. This required understanding of the new lisp protocol and the creation of a function on the emacs side as well as the lisp side. The old lisp side file BACKDOOR.CL is no longer valid and has been replaced with a function located in SSMPMAD-IF.CL called SET-BACKDOOR-HOOKS. This

functions was added to the run-startup-functions list. On the emacs side the file BACKDOOR.el was complete replaced with new functions. Files Modified: PACKAGES.CL, BACKDOOR.EL SSMPMAD.SYS, AND SSMPMAD-IF.CL.

76. **Problem:** [CLOSED] 10 MAR 92 [MIKE]

Update time is crashing if the user goes to the LPLMS screen after having transition from NORMAL to MANUAL back to NORMAL.

Disposition:

The was caused by the ACTIVE-SCHEDULE getting reset every time the system transition out of NORMAL mode into any other mode. This meant that the next time the user went to LPLMS screen that the ACTIVE-SCHEDULE that was last used no longer existed. This cause the system to compute a GANTT chart with no data. This then caused the UPDATE-TIME-NOW function to crash (this should not have happened). The fix was to make sure that the ACTIVE-SCHEDULE did not change as when the user changed modes. Also the graphtool needed to be fixed to handle the NIL data case under any circumstances. Files Modified: SSMPMAD-IF.CL and GRAPH.CL.

77. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

The system failed to finish making a diagnosis. The scenario went like this.

- (a) Brought up the system.
- (b) Placed it in Normal Mode.
- (c) Activated "Walls Schedule 1".
- (d) Tripped LC-2-19.
- (e) Placed this switch back into service.
- (f) Seized manual control of LC-2-16.
- (g) tripped LC-2-17 FT.
- (h) tripped LC-2-19 FT.
- (i) tripped LC-2-18 FT.
- (j) Put switches 17, 18 and 19 back into service.
- (k) LC-2-16 was turned on
- (l) tripped LC-2-16 FT
- (m) Tripped LC-2-17 FT
- (n) Tripped LC-2-18 UV

The system never came back with a diagnosis for the fault.

Disposition:

The problem lied in the MF-DIAGNOSIS.RG which was not resetting some needed flags in multiple hard fault situations. This fix has been implemented and tested. Files Modified MF-DIAGNOSIS.RG.

78. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

MAESTRO is crashing when ever the 3K on port bus is tripped on load center 3. MAESTRO crashes talking about trying to struct-ref of a nil.

Disposition:

This problem came up after LC-3 Switch 15 blew up. This switch was taken out of the system to be repaired but the schedule that the users were running remained the same. So MAESTRO thought that switch 15 was still being used. Now when the fault occurred the LLP's switched to redundant the two 1K's that undervoltaged. But now switch 19 was not shed because the total power on the bus did not exceed 3K because switch 15 did not exist. In MAESTRO's eyes though this switch did exist so there must have been a shed to be within the power allotment for the 3K. This confusion some how caused MAESTRO to crash. At this time there is not enough knowledge to fix MAESTRO and that is probably not the right thing to do anyway because we are giving MAESTRO inconsistent data anyway. The better solution is to make sure that when a schedule is activated that all the switches that the schedule uses really exist. This has been solved with the inclusion of fix from PROBLEM 9.

79. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

After the user takes Manual control the priority list does not appear to be correct. The example was the user took control of a switch that was under a priority 2 control. The user seized this switch under a priority 0 control. The new priority list had the switch in the same position as it was when it was at a lower priority!!! This is wrong what is the system doing about the priority of a manual seized switch.

Disposition:

In the re-write of the LPLMS rules into a procedural format (this was done to accomodate Louis Lollar's design of a better LPLMS system) the problem was not encountered. This is because there was a error in the rule group that was not detected.

80. **Problem:** [CLOSED] 20 MAR 92 [MIKE and CHRIS]

When an I²T fault is put into the system sometimes the LLP's will catch over power utilization before the GC card will. The problem is that when the LLP's catch the problem the switch is not taken out of service but MAESTRO is told that it has been

shed and depending on the activity MAESTRO will just re-schedule the activity and this will start all over again. If the GC catches the problem the switch is taken out of service. This seems very wrong because in the first case it is diagnosed as a soft fault while in the second case it is diagnosed as a hard fault. Also in manual mode I don't think that the LLP's should ever care about being over any software limit. That's the not it's job in manual mode.

Disposition:

With the definition of soft and incipient faults, the LLP code was changed to handle this problem. The LLP code was fixed and tested by Chris Myers and Janet Weinz.

81. **Problem:** [CLOSED] 20 MAR 92 [MIKE]

When the user turned on a 1K switch in maintenance mode and that switch faulted the redundant to that switch is turned on. This is not suppose to happen.

Disposition:

The problem lay in how the event list were being created to turn on the switches in MAINTENANCE mode. The code in manual.cl created the event using the character 'N for both the Switch to redundant field and the permission to test field. The Transform out function that sent this message out was looking for T or NIL. This caused the transform out function to change the character 'N to 'Y meaning the switches had redundant and were able to test. The fix was to change the manual on commands to nil instead of 'N for those to fields and that fixed the problem. This change has been implemented into the system. Files Modified: MANUAL.CL.

82. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

There needs to be a option to load patch files from RUN-STARTUP-FUNCTIONS. This is needed to supply the customer with fixes that do not require us to deliver a new dumped lisp every time a change is made to fix a bug.

Disposition:

This change has been implemented and tested. Files Modified: SSMPMAD-IF.CL

83. **Problem:** [CLOSED] 01 JUN 92 [MIKE]

When a manually controlled switch is shed the system does not remove the switch from the user control correctly.

Disposition:

This problem was solved by letting the MONITOR processes take control of displaying and removing MANUAL icons from the power system screen. Files Modified: MONITOR-PROCESSES.CL

84. Problem: [OPEN Priority 1] 25 MAR 92 [MIKE and CHRIS]

When an I²T trip happens that sometimes is shed by the LLP's but once in a while the hardware will catch the fault after the LLP's have shed it a few times before. This happens because the Over Current trip has been left on and MAESTRO took the shed switch and rescheduled the activity that was using that switch and turned that switch back on. When the switch finally went on a hardware over current the switch did not go to redundant like it should for all hard faults. This also brought up another problem, in the case of the LLP catching the over current it did not go to redundant when it shed the load and it should have. This is an inconsistency that needs to be fixed. In this case the shed was preventing an over current not catching someone using too much power and in the case of priority zero loads those that are critical for life it makes no sense to shed the air supply switch and not go to redundant!!!! The loads need to be protected at all costs even if that means bringing down the bus.

Disposition:

Chris thinks he has found a reason why the llp do not switch the switch to redundant when it finally tripped on an over current. It has to do with chris setting the shed flag after determining the over current but before he sends commands to the switches. So when he gets to that part of the code to send out the command to the switch he sees the shed flag and therefore does not send the command to turn on the redundant switch. It looks like a change to the LLP code will be needed to fix this and at the same time a change will be needed to fix not turning on the redundant of a shed switch. This will change to always try to do that. We might also think about not letting the LLP's ever software shed a priority 0 load no matter how over budget the system is.

85. Problem: [CLOSED] 11 AUG 92 [MIKE]

No longer need the other set of database files. The shadow files should be renamed to the standard file name to prevent confusion in the future. The shadow files were used because MAESTRO needed on format and PMAD needed another format that had more information. Since MAESTRO has been ported into the PMAD system they now use the same data structures as PMAD. This means we only need one database and that is the old shadow databases. This requires renaming them and should not be done until the next delivery that will be done in april or the end of april.

Disposition:

This has been completed by removing the reference to these file from the SSMP-MAD.SYS and DB-FUNCTIONS.CL files.

86. Problem: [CLOSED] 01 JUN 92 [MIKE]

When a 3K faults but upon retest nothing is found the system does not think it can retest any of the 1K's below even though they are testable in the event list.

Disposition:

This problem seems to be related to the fact the all events in the event list are now represented by T or NIL for all data that was looked at with a 'y or 'n. The change was not propagated completely through the multiple hard fault rule group. In some places the rule group still checks against the 'y 'n values. This change has been implemented and tested in the system. Files Modified: MF-DIAGNOSIS.RG

87. Problem: [CLOSED] 01 JUN 92 [MIKE]

When an I²T trip is applied to a 3K the 3K trips on over current and the 1K's below undervoltage. When the system tries to retest the switch the switch does not overcurrent because of the reduced load on the system but it is pulling 27 AMPS. The system then decides since it cannot test the 1K switch (See problem above) there is no way to diagnosis the problem.

Disposition:

We need to use the data coming back from the LLP's telling us the amount of power being consumed by the 3K's and check that against some threshold value because the 3K may not always re-trip but if it is pulling 27 AMPS by itself there must be something wrong and this 3K needs to be taken out of service with the diagnosis being possible high impedance short. We should not try to test the 1K in this case because in this case the first 1K turned on will cause the 3K to over current and we will diagnosis a bad 1K when it is really the 3K that is having problems. Need to use all the data that is presented to us. This has been done by adding new rules to the MULTIPLE-HARDFULT rule group as well as adding some new functions that will be called by these rules to check this condition. Files Modified: MF-DIAGNOSIS.RG and DOMAIN-FUNCTIONS.CL

88. Problem: [CLOSED] 01 JUN 92 [MIKE]

The FRAMES system hangs on the fact NEW-SYMP TOMS thinking it has not been set even though the value has been set to NIL. This only seems to occur after the system has been unable to diagnosis a particular fault. The case is a 3K is made to over current so the GC for the 3K shuts it off and this causes all the 1K's below to fault on under voltage. On retesting the 3K it does not retrip (See Problem Above) and then the 1K's are not tested (See Problem above) so the system diagnosis a "NO FAULT FOUND" and takes the 3K out of service.

Disposition:

In the failed diagnosis the system did not clear out (reset) some data slots so when the next fault occurred the system did not attach the new symptoms to the fact NEW-SYMP TOMS. This change has been implemented and tested in the system. Files

Modified: MF-DIAGNOSIS.RG and DOMAIN-FUNCTIONS.CL.

89. **Problem:** [CLOSED] 01 JUN 92 [CHRIS]
The diagnosis for "NOT FOUND CAN'T TEST FURTHER" is printing out NIL-NIL for the lower level switches that the system can not test. Also a NIL-NIL is being printed in the phase "A transient short somewhere below NIL-NIL and a short below one of the switches that are not testable." This should read or a short below one of the switches that are not testable.
Disposition:
Problem had to do with the value in slot1 of frame diagnosis. Everywhere else in the system that slot held symptoms but in this particular case the slot help a list of switches. Files Modified: OUTPUT-DIAGNOSIS.CL.
90. **Problem:** [CLOSED] 01 JUN 92 [JANET]
System crashes in graphtool some where as it tries to plot values over the original max value.
Disposition:
Fixed the out-of-range y-coordinate problem by clipping data of the max value. This change has been implemented and tested. File Modified: GRAPH.CL
91. **Problem:** [CLOSED] 01 JUN 92 [CHRIS]
The actual profiles get zeroed when going from NORMAL mode into any other mode causing the power util plots to look strange because they always loop through the zero point on a mode change.
Disposition:
Change was made to take the data when it comes up from the llp and through out the zero time point because it is no meaningful data. This change has been implemented and tested. File Modified: SSMPMAD-IF-TR.CL.
92. **Problem:** [CLOSED] 01 JUN 92 [MIKE]
When a switch goes to redundant and the redundant switch fails, MAESTRO crashes because the system sent it a succesful switch to redundant command and at the same time telling MAESTRO that the redundant switch has failed.
Disposition:
The fix was to make sure that if the redundant switch fails that the switch to redundant is not sent to MAESTRO. This was done in the planner processes code that handled

a contingency. File Modified: PLANNER-PROCESSES.CL

93. **Problem:** [CLOSED] 11 AUG 92

Writing to the scratchpad and the diagnosis window the data in those windows will get jumbled and misprinted.

Disposition:

This problem only seems to occur when we have another X window overlapping either one of those windows. No one is sure why that is and that may be a bug within Common Windows. We need to come up with an example that we can send to FRANZ. On further examination it was determined the the scroll menu code had been so kludged up with brain damaged repaint code with server repaint also turned on the code was confusing the server. On handle damage event was generating up to 11 repaints. The entire scroll window code was thrown out and redone. This has been tested and shown to work much faster now. Files Modified: TEXT-SCROLLER.CL and STATIC-SCROLL-BAR.CL.

94. **Problem:** [CLOSED] 01 JUN 92 [JANET]

Need to make sure that the switches above a redundant switch are on in case the system needs to turn on that switch in the case of the primary switch faulting in some way.

Disposition:

Change has been made that will make sure that when making an event list that the redundant switches are taken into account when generating the event lists. These changes have been implemented and tested in the system. Files Modified: FELES-PROCESSES.CL

95. **Problem:** [CLOSED] 01 JUN 92 [JANET]

Need to change the database to add three new fields for the powered equipment. PERMISSION-TO-AUGMENT (NIL), MAX-PERCENT-TO-AUGMENT (0), and MAX-VALUE-TO-AUGMENT (0). This will also require a change the database read and write functions as well as the activity editor. The EVENT-LIST needs to augmented to add a new field called the POWER-BUDGET-LIMIT. This will also require changes in the communications code.

Disposition:

Janet added these fields to the database and the code has been tested.

96. **Problem:** [CLOSED] 10 MAY 92 [MIKE]

When putting a switch back into service and a upper switch needs to be turned on to test the switch. The upper switches need to be turned off after testing because the system does not know that they are on so leaving them on can cause them to under-voltage.

Disposition:

This has been fixed by keeping track of the upper level switches that needed to be turned on for testing of lower level switches. These switches are then turned off when leaving the method PROCESSES-PLANNER-DATA for putting switches back into service. Files Modified: PLANNER-PROCESSES.CL

97. **Problem:** [OPEN Priority 2] 24 MAY 92 [MIKE]

We need to generate unique names for all ILA Activities. Currently if the user grabs switch LC-1 02 by itself the system will generate an activity with that name and add it to the system. If later on the user grabs the same switch again in ILA another activity with the same name will be generated. So the FELES screen will show two activities with the exact same name.

Disposition:

None at this time.

98. **Problem:** [CLOSED] 24 MAY 92 [JANET]

When activating a schedule the user must conform to the power limitations put on the system this must also be reflected in the resource profiles of all the upper level switches. The default values per RBI is 12,500 WATTS.

Disposition:

Janet implemented this change and the code has been tested.

99. **Problem:** [OPEN Priority 1] 24 MAY 92 [MIKE]

Off events are not generated for activities that were on less than one minute prior to them being removed from the system. This is because the code to truncate the activity sees that it starts and ends in the same minute so it removes the entire activity. Once the activity has been removed from the schedule the event list has lost all information about that activity so no off event gets generated.

Disposition:

This problem is an effect of the one minute MAESTRO granularity problem. Fixing the MAESTRO problem is the answer here.

100. **Problem:** [CLOSED] 11 AUG 92 [CHRIS]

Need to have mnemonic names for sensors used in the OUTPUT-DIAGNOSIS code.

Disposition:

The output diagnosis code was updated to put out MNEMONIC names for the sensors. This change has been designed, implemented and tested. Files Modified: OUTPUT-DIAGNOSIS.CL.

101. **Problem:** [OPEN Priority 1] 27 MAY 92 [MIKE]

When testing a switch to put back into service we have to look at how much power that switch might pull and determine if that much power is available for testing. This might require that in order to put a switch back into service, the user must have that switch under manual control so we know because we assigned the power how much power that switch might use during testing.

Disposition:

In order to accomplish the testing of switches to put them back in service the system needs to determine if enough power is available. If not, find a way to make the power available or cancel at user's discretion.

102. **Problem:** [CLOSED] 11 AUG 92 [CHRIS]

On a dumped lisp all the scroll bars are missing from the diagnosis windows and the scratchpad windows until those window scroll for the first time.

Disposition:

Changed the startup function to restore the scrollbars at startup. We use our own repaint method. When the image was brought up and an expose event was generated for the window the scroll bar repaint method did not repaint the scroll bar because it thought nothing had changed. The fix to this was to force the scroll bar to repaint once but only on startup of the system.

103. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

PROCESS-SUBTASK-PERFORMANCE no longer needs the ACT-ID (third argument) because the subtask performance is correct in the performance in the schedule. This is in either the file SCHEDULE-FUNCTIONS.CL or DB-FUNCTIONS.CL

Disposition:

This was removed from the file SCHEDULE-FUNCTIONS.CL. Since moving MAESTRO to the SOLBOURNE this was no longer needed. Files Modified: DB-FUNCTIONS.CL

104. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

When an activity has a redundant switch and that switch becomes unavailable because

of an upstream power failure of some type the system needs to remove that redundant capability from the system until such time that the upstream problem has been solved. The current system will that switch redundant and on a failure switch to that switch and of coarse it will not work.

Disposition:

The trick to make this work is to remove redundant capability when the lower level switch that is the redundant becomes unavailable. This capability was added to the make unavailable code. This fix turns out to be a hack because it changes the mode of the activity that is losing its redundancy and this should not be done this way. The real way to correct this problem would be to have MAESTRO handle redundancies better and let it send over new modes when redundancies are affected in some way. But this would also require MAESTRO to better integrate our understanding of a redundant switch. Files Modified: PLANNER-PROCESSES.CL.

105. **Problem:** [OPEN Priority 2] 27 MAY 92 [MIKE]

The current handling of redundants has the redundant code changing the mode on the fly to reflect whether that mode can be redundant or not. This is a kludge to make the redundant ICONS work correctly. This means when a primary switch switches to redundant it no longer has a redundant until the failure reason that affected the primary switch is resolved. So the mode must be changed to show that it is no longer redundant. When the primary has been restored to the system then redundancy needs to be restored to the mode. Currently this is done by modifying the mode on the fly when we should be making a new mode depending on the situation.

Disposition:

See the above disposition of PROBLEM 104 for a better explanation on what needs to be done. This needs to be addressed by our customer to determine if these changes should be done. It looks like a 3 man month effort to fix this correctly on the MAESTRO side.

106. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

The compute priority weightings need to be changed to only look at performances that have not already passed. The current system slows way down in computing the priority weightings because every time the schedule is halted and then restarted the number of performances has doubled on schedule but only half of them are relevant the other half have passed in time and should not be looked at.

Disposition:

This enhancement was added when the LPLMS code was re-written from the rule based implementation it originally was to its current procedural form. This code has been

design implemented and tested. Files Modified: LPLMS-PROCESSES.CL

107. **Problem:** [OPEN Priority 1] 27 MAY 92 [CHRIS]

LLP's need to do shedding of 3K's. This will be needed for soft and incipient fault handling. This will require a change in the EVENT LIST code so that the redundant power is add to the 3K's where the redundant switches reside. This value should reflect how much power will be drawn if the redundants are turned on. This value should be up to but not greater than 3000 Watts (The LLP's will shed to make sure the 3000 Watt limit is not broken).

Disposition:

To date, 11 AUG 92, no real effort has been put in to design a way to do this because it is difficult to do without major changes down at the LLP level. This stems from the fact that when a 1 K switch goes to redundant it does so without notifying the upper level 3K switch which has a power profile based on the redundants not being on. So this means that when a redundant switching event happens the Load Center LLP's need to communicate with the PDCU LLP's to notify of the power profile change so that the 3K would not be shed. This is the driving force for inter-LLP communication.

108. **Problem:** [OPEN Priority 1] 13 AUG 92 [CHRIS and MIKE]

Need to have the ability to adjust manual parameters on the fly with switches that are under manual control.

Disposition:

Both Chris and Mike need to work on this. Chris needs to add the user interface hooks to allow this to happen. Mike need to write the code that would implemented the changed switch parameters.

109. **Problem:** [CLOSED] 11 AUG 92 [MIKE]

If a switch is under manual control when it is put back into service and FRAMES deems that the switch can be put back into service than the system needs to release the switch from manual control. The thought here is that if the user wanted to keep the switch under manual control he would not have put it back into service and for now this is okay. In the future you might give the user the choice on what to do with the switch when it is being put back into service.

Disposition:

The current working of this code is to always remove the switch from manual control whenever FRAMES has succesfully tested and accepted the switch for inclusion into the set of non-faulted switches. This has been designed, implemented and tested.

Modified Files: PLANNER-PROCESSES.CL.

110. **Problem:** [CLOSED]

29 FEB 92 [CHRIS]

12 JUL 92 [CHRIS]

11 AUG 92 [CHRIS]

Upgrade how information to the diagnosis window is displayed.

Disposition:

The first change was to make 4 different windows that overlay each other. they would be broken into four categories. Hard Faults, Soft Faults, Incipient Faults and Source Power Change. In the Hard Fault window the way the data was displayed was changed. It now uses a 3 column format. The first column is the error message or system information. The second column is the FRAMES action taken and the third column is the result/diagnosis. After this initial delivery in FEB the code was expanded to have putting switches back into service show its information in the hard fault window because it deals with putting hard faulted switches back into service and it is run by the FRAMES system. At the same time the diagnosis was set off so the user would know where one diagnosis ended and the next one started. The last upgrade to this was to add highlight bars to the differnt diagnosis windows turning them red when ever data was written to them and that window was not exposed. This lets the user know that there is new information.

111. **Problem:** [OPEN priority 1] 11 AUG 92 [CHRIS]

The customer has requested that ILA seizing and releasing use the check mark interface of the power system screen like the rest of the applications on the power system screens.

Disposition:

To date no work has been performed on this task. The problem is that we are not sure how to implement the add/drop functionality of the ILA code because the workbox code will interpret any button click as a waveoff and remove the ILA Workbox. Once this problem is solved there will still be the problem of sending the switch from the button click to the workbox. Both of these have been given to Chris to look into.

112. **Problem:** [OPEN Priority 1] 12 AUG 92 [MIKE]

If switches are switched to their redundants, the power on that bus must be sufficient to power the activities for the redundant activities. If the power is less than the required amount, and a contingency occurs that causes those redundant switches to be turned on MAESTRO crashes in a call to "get-performances" because the activity to run the

redundant switch was not scheduled. A specific example that causes this crash is:

Mikes Test 4 was activated

B02, B03 and D19 were grouped as test group 4

C03 was grouped as test group 5

Seized manual control of test group 4 and test group 5

Did a source power change on port to 2600 W and on starboard to 0W one minute after the current mission time.

All switches were shed except D04, D06 and D07.

Tripped P04. Result: D04 and D06 went to redundants and D18 and D20

Maestro crashed with a message that came up just prior to the crash. Attempted to get performance 27 of activity A2855. Performance not scheduled.

The 0 power on starboard when MAESTRO tried to schedule the redundant activities caused the crash. Part of MAESTRO assumes that redundants always get scheduled while the main scheduling algorithm may disagree.

Disposition:

A simple kludge is to make sure that there is enough power above what is currently being used to turn the redundant on so MAESTRO will schedule the redundant activities. A long term solution is to upgrade MAESTRO to correctly handle the redundant capabilities. This is the second example where mis handling of the redundant switch has caused problems. The first is **Problem: 104.**

113. **Problem:** [OPEN Priority 1] 12 AUG 92 [MIKE]

If the power available on a bus is not enough to support the redundant switches that could be turned on and the position in the LPLMS would not allow them, then the redundant capability needs to be removed.

Disposition:

This is directly the result of the problem discussed above. Part of this has been solved on the SSMPMAD side but to implement all of it would create some problems because the LLP's would need to talk to each other to decide if the redundants could be turned on based on how much bus power is available. And not the hard coded 3K limit that the individual LLP has for the redundant bus. Look at graying out the redundants.

114. **Problem:** [CLOSED] 11 AUG 92 [CHRIS]

Because the system has to page or is getting ready to do a global gc sometimes it is

very difficult to determine if the button click in an active region was accepted. This causes the user to often times double click the same selection giving unwanted results.

Disposition:

Upon receiving any selection that selection is highlighted in blue for 1/3 of a second to notify the user that his button click has been accepted. This code has been designed, implemented and tested. Files Modified: CLASSES.CL, LOAD-TIME.CL and WORKBOX.CL.

115. **Problem:** [OPEN Priority 1] 11 AUG 92 [MIKE]

When the user turns on a manually controlled switch but the switch above is not closed the manual operation will send the upstream switches close commands. If one of the upstream switches is faulted then this will cause the manual switch to fault. When this happens the manual switch goes into a faulted state.

Disposition:

There needs to be smarts in the FRAMES system to recognize that the fault that has occurred is due to user error I.E. the user tried to turn on a switch whose upstream switches are not available. This would require writing a few more rules and that is not that hard to do. The expected time to do this is about 1 man month.

116. **Problem:** [CLOSED] 11 AUG 92 [LANCE]

The rule system is very difficult to debug when there is a problem. The major problem is the FRAMES system hanging. When this happens there is almost no way to determine what caused the system to hang.

Disposition:

The multiple hardfault rule group was re-written in lisp as rules. that means that there is a left hand side and a corresponding right hand side for each rule. But now since the rule is already in lisp and not in a language that had to be translated into lisp and that lisp translation was then run uncompiled this has increased the speed of rule execution by a factor of 50. The new method also is much easier to debug because the debugging tools built into it allow us to trace the rule execution if there is a problem. This change also lets us change rules on the fly without reloading the system which was required before. This new definition now allows the people at Marshall who will look at writing the FRAMES system in ADA an easier way of doing it. Files Modified: MF-FRAMES.KB, MF-CONTROL.RG and added a new file MULTIPLE-HARDFULT.CL to replace MULTIPLE-FAULT.RG.

117. **Problem:** [CLOSED] 11 AUG 92 [MIKE and JANET]

A Source Power Change is not done correctly if there is a group of ILA switches and

one of them is shed due to the Source Power Change.

Disposition:

The problem here is that when one switch in the ILA group was shed the entire group should have been removed from manual control. First a modification to the function "process-source-power-change" was done to shed all switches in a ILA group when the sum of the power for the switches in the group for a particular port exceeds the source power change limit for that port. The next step was to remove the activity for the ILA group and replace it with an activity that was truncated at the time of the load shed. This was done in a new function called "ila-switch-shed" which also made sure to delete the old group. This change has been designed, coded and tested. Files Modified: FELES-PROCESSES.CL, SSM/PMAD-IF-TR.CL, PLANNER-PROCESSES.CL and PLANNER-SUPPORT.CL

118. **Problem:** [CLOSED] 14 JAN 93 [CHRIS]

On the Power Utilization screen the utilization line will be on prior to when the start of when power is being drawn on a sensor. The reason for this is that power is averaged over a five minute time period and then sent up to the Workstation. If a switch below the sensor turned on some time during the five minute period than the power value sent back for that entire interval will be the average of no power being drawn and some power being drawn. This causes two problems. The first is that it appears on the power utilization screen that the 3K switches were using power when they were not scheduled to and next there is a power change in the next five minute interval. The power change is not really there because in the first interval a bunch of zeros are being added in while in the next power interval it is continuously drawing power.

Disposition:

This problem has not been dispositioned at this time.

119. **Problem:** [OPEN Priority 1] 14 DEC 92 [MIKE]

There are several incorrect diagnosis being performed on a 3K intermittent failure associated with a child 1K fault. A child switch is a switch of an upper level switch that feeds that switch current. In the current system setup a parent 3K switch may have up to 6 child 1K switches that it is feeding current to. There are three distinct problems that have been documented with the same fault scheme. The fault scheme is a 3K Over Current trip applied in conjunction with a 1K child being faulted on Fast Trip. In the first case the Over Current and Fast Tripped are received prior to the start of diagnosis. Put the diagnosis is that there is an intermittent 3K fault but does not test the 1K that faulted. In the second test case has the 3K fault occurring just prior to the 1K fault being applied. Here the FRAMES system starts diagnosing the 1K fault because it has not seen the 3K fault yet. When the 3K fails the conclusion is

that the 3K can not fail during an open operation of a 1K so this type of fault is not diagnosable. The last fault has the 3K failure occurring before the 1K fault is registered so that switches fails on under voltage . In trying to diagnosis this the system tests the 1K's below and the 1K with the Fast Trip attached fails on Fast Trip but the failure is not related to the 3K fault and after the correct diagnosis for the 3K the system tries to test the 1K with the parent 3K out of service.

Disposition:

This information has been sent to the customer in an expanded format for is input on the course of action to take.

120. Problem: [CLOSED] 18 DEC 92 [MIKE]

When a 3K switch has failed, all the 1K's below that switch are made unavailable. If the user then decides to grab some of these switches that where made unavailable from the 3k failure and the 3K is then returned to service the 1K's that where under manual control loose there manual icons.

Disposition:

The problem arose because the contingency code dolists over all the switches under manual control. If the switch above is in the out of service list even though that out of service call is being done to put the switch back into service the code removes the switch from manual control. An easy software change to the PROCESSES-PLANNER-DATA :CONTINGENCY code of PLANNER-PROCESS has fixed this problem. This change has been designed, implemented and tested. Files Modified: PLANNER-PROCESSES.CL.

121. Problem: [OPEN] 10 DEC 92 [MIKE]

P04 Fast Tripped and then was diagnosed as such an was put out of service and under manual control. At the same time switches LC-3-03, LC-3-04, LC-3-06, and LC-3-07 were mad unavailable. They are unavailable because P04 is out of service. Next LC-3-03 and LC-3-04 was put under manual control by using the ILA capability. These two switches were then turned on. They do turn on but frames collects a snapshot to start diagnosis but then later decides that no contingency has occurred.

Disposition:

The collecting the snapshot is due to the 3K having a maximum power allocation of 1W. When the 1K's where turned on the llp controlling the 3K saw that the power being drawn through it was greater than the 1W limit it was given so sent up fault information. The code GATHER-DATA-AND-START-DIAGNOSIS code than saw that there really was no fault so it prevented the diagnosis system from running. The solution to this problem is to make sure the limit for the faulted 3K is such thatit can handle the

power of the manual controlled 1K's below. This change has not yet been implemented.

122. **Problem:** [CLOSED] 11 DEC 92 [MIKE]

With the fault scheme of a 3K Over Current and 1K Fast Trip that leads to an incorrect fault diagnosis which is documented in an earlier bug there is another problem. After the completion of the diagnosis the system collects a snapshot and then decides that no contingency has occurred.

Disposition:

This problem is related to putting the 1K switches out of service. In this scheme the 3K is taken out of service so all the 1K's below it must be made unavailable. Because of the type of diagnosis that was performed all the switches including the 3K above are still on. The system send down the command to turn off the 1Ks first then send down the command to turn off the 3K. The off commands are sent down in a switch control list with the fault isolation attribute. The fault isolation attribute take 5 seconds to turn off each switch. For the 1K's this means 15 seconds. in the mean time there is only a 2 second delay between sending down the 1K open commands and the 3K open command. This leads to the 3K opening prior to the rest of the 1K being opened so the other two 1K's fail on under voltage. The system then gathers data to see what has happened but by now the open commands have occurred and cleared the under voltage trip so the system thinks no contingency has occurred. The fix was to change the off commands to be unconditional so that the llp turning off the 1K's could now turn off all the 1K's before the 3K was turned off. Files Modified: PROCESSES.CL

123. **Problem:** [OPEN] 10 DEC 92 [MIKE]

P04 Fast Tripped and then was diagnosed as such an was put out of service and under manual control. At the same time switches LC-3-03, LC-3-04, LC-3-06, and LC-3-07 were mad unavailable. They are unavailable because P04 is out of service. Next LC-3-03 and LC-3-04 was put under manual control by using the ILA capability. These two switches were then turned on. The 3K switch is than put back into service with the two 1K's still closed. During the testing phase to return the 3K switch back to service the two 1K's under voltage when the 3K is opened during testing. The 3K is subsequently returned to service and FRAMES goes off to try to diagnosis the two 1K failures. The diagnois is wrong for the failure and the system never reschedules any activities for those two switches even when they are returned to service. After the diagnosis for the 1K's are done the switches are left unavailable (green) when they should have been maked Out Of Sevice.

Disposition:

The first part of the problem is that the testing routine needs to make sure that if the switch being tested is not already on. If the switch is already on then the switch must

be okay so it can be returned to service without testing. The diagnosis is wrong because the symbol that represented the fault type should be :broken-output-cable-of-switch-above. Why the switches are not taken out of service has still not been determined. The final disposition of this problem has not been determined but should be by the end of February.

124. **Problem:** [OPEN] 10 DEC 92 [MIKE]

FELES Time Now does not show up on the FELES screen any more. Also the screen appears to be redrawn several times when there is an update done to GNATT Chart.

Disposition:

Inspection of the graph tool code has shown that there are many problems to be fixed in the "CHART" code of the Graphtool. This is going to be re-written. At the same time the FELES-UI.CL contains all the code to create and update the FELES screen is going to be re-written to make better use of the new Graphtool GANTT chart capability.

125. **Problem:** [OPEN] 10 DEC 92 [CHRIS]

When a selection is made on an active region of the activity editor there is no acknowledgment highlighting that is done with all other button selections in the system.

Disposition:

This was an oversight when the initial work was done to add highlighting to the system it needs to be incorporated here. This bug has not been dispositioned at this time.

126. **Problem:** [CLOSED] 10 DEC 92 [CHRIS]

When the scratchpad has been cleared the scroll bar is not reset.

Disposition:

This problem was fixed and tested 29 March 93 by Chris Myers.

127. **Problem:** [CLOSED] 1 APRIL 93 [CHRIS]

The way information is displayed in the FELES screen for Activity Info and Load Info is at best difficult to read and at worst misleading.

Disposition:

This problem has been alleviated with the Selected not available menu items in the static menu for the FELES-UI application.

128. **Problem:** [CLOSED] 1 APRIL 93 [CHRIS]

The selected static menu item does not stay selected when the mouse is in the item region when it is deactivated.

Disposition:

This problem has been fixed by not allowing the mouse-out method to execute if the region is not active.

129. **Problem:** [CLOSED] 1 APRIL 93 [CHRIS]

The fault diagnosis code responds with a no contingency seems to have occurred in a switch which has been taken out of service and placed under manual control.

Disposition:

This problem resulted due to an improperly updated switch event when place the switch under manual control. This problem was fixed and tested by Chris Myers.

130. **Problem:** [CLOSED] 1 APRIL 93 [CHRIS]

During fault diagnosis, specifically a no power to bus fault, some of the 3K rpc icons on screen display ?? instead of UV when they are taken out of service and placed under manual control.

Disposition:

This problem resulted due to a race condition between taking individual switches out of service, and the ensuing switch status data which was returned from the LLP's in response to turning off a switch. The algorithm was restructured to eliminate this problem. The code was fixed and tested by Chris Myers.

131. **Problem:** [OPEN] 21 JUN 93 [CHRIS]

Customer desires a new switch icon color to designate that a switch is being tested.

Disposition:

Under consideration, need to examine possible impacts with the diagnosis code with respect to adding a new switch state to the system.

132. **Problem:** [OPEN] 21 JUN 93 [CHRIS]

Customer wants to move the MAESTRO block in the system flow window on the User Interface into the KNOMAD architecture block.

Disposition:

This requires a minor change in the User Interface software and can be done with minimal difficulty. This change will have no side-effects on the rest of the system.

133. **Problem:** [OPEN] 21 JUN 93 [CHRIS]
Manually turning on a tripped 1K RPC causes the SSM/PMAD system to hang?
- Disposition:**
This bug has not been examined yet.
134. **Problem:** [OPEN] 23 JUN 93 [CHRIS]
Manual Seizure is having difficulty with effects. Effects aren't appearing for bumped switches on the port bus?
- Disposition:**
This bug has not been examined yet.
135. **Problem:** [OPEN] 23 JUN 93 [CHRIS]
Scheduled current tick marks are missing from the soft faulted switch icon bar chart.
- Disposition:**
This is caused due to improper redraw of bar chart carat tick mark when state changes from :good to :soft-fault.
136. **Problem:** [OPEN] 23 JUN 93 [CHRIS]
Customer desires the ability to halt the current schedule as an option when activating a schedule, if needed.
- Disposition:**
This is a user interface issue, should not be difficult to change.
137. **Problem:** [OPEN] 23 JUN 93 [CHRIS]
Customer desires a Group Name on the FELES application screen for a manually seized group.
- Disposition:**
This is a user interface issue, at this time it is not known how difficult it will be to make this change.
138. **Problem:** [OPEN] 23 JUN 93 [CHRIS]
Customer desires the scheduled current bar tick marks to be displayed on open switches
-

in addition to the closed ones. The reasoning for this is the need to see the scheduled current for a manually controlled switch when it is off.

Disposition:

This will require changes in the generation of the user interface switch icons to draw the bar chart. Also the if-added code for scheduled current and bar charts will need to be updated to handle data on an open switch.

Appendix F

SSM/PMAD Test Plan

(This Page Intentionally Left Blank)

SSM/PMAD Test Plan

Barry R. Ashworth

Michael R. Elges

Chris J. Myers

Janet K. Weinz

Regina K. Palmer

William D. Miller

Regina K. Palmer
SSM/PMAD Product Assurance

William D. Miller
SSM/PMAD Program Manager

Regina K. Palmer
SSM/PMAD System and
Occupational Safety

William D. Miller
SSM/PMAD Test

(This Page Intentionally Left Blank)

Contents

1	Acronyms	369
2	Hardware	371
3	Software	373
4	Test Data	375
4.1	Input Data	375
4.2	Output Data	375
5	Startup Test Procedures	377
5.1	Test Setup	377
5.2	Test Initialization	378
6	User Interface: How to Execute the Test Procedures	379
7	User Interface: System Options Menu	387
7.1	Mode Changes Submenu	387
7.1.1	Go To Maintenance	387
7.1.2	Go To Normal	387
7.1.3	Go To Idle	388
7.1.4	Help	388
7.2	Schedule Functions Submenu	388
7.2.1	Read a Schedule	389
7.2.2	Activate a Schedule	389
7.2.3	Halt the Active Schedule	390
7.2.4	Help	390
7.3	Other System Operations Submenu	390
7.3.1	Connect to LLP	391
7.3.2	Disconnect from LLP	391
7.3.3	Connect to APEX	391
7.3.4	Disconnect from APEX	392
7.3.5	Source Power Change	392
7.3.6	Return RPCs to Service	392
7.3.7	Clear Soft Faults	393
7.3.8	Clear Incipient Faults	393
7.3.9	Current Fault State	394
7.3.10	Help	394
7.4	Utilities Submenu	394

7.4.1	Clear Scratchpad	394
7.4.2	Enable Sensor Balancing	395
7.4.3	Disable Sensor Balancing	395
7.4.4	Disable SysFlow Highlighting	395
7.4.5	Enable SysFlow Highlighting	396
7.4.6	Disable GC Cursor	396
7.4.7	Enable GC Cursor	396
7.4.8	Screendump	397
7.4.9	Windowdump	397
7.4.10	Help	397
7.5	Help	398
7.6	Exit From <i>SSM/PMAD</i>	398
7.6.1	Yes	398
7.7	Cancel	399
8	User Interface: Power System Application	401
8.1	System Options	401
8.2	Detailed Data	401
8.3	Detailed Data Point Down	402
8.4	Deselect All	402
8.5	Manual On	403
8.5.1	Manual On in MAINTENANCE mode	403
8.5.2	Manual On in NORMAL mode	404
8.6	Manual Off	404
8.6.1	Manual Off in MAINTENANCE mode	405
8.6.2	Manual Off in NORMAL mode	405
8.7	Seize Manual Control	406
8.8	Update Manual Control	406
8.9	Release Manual Control	407
8.10	Component Information	407
8.10.1	Load Information	408
8.10.2	Load Information Point Down	408
8.10.3	Power Utilization	409
8.10.4	Switch Schedule	409
8.10.5	Switch Schedule Point Down	409
8.11	Group Switches	410
8.12	Ungroup Switches	410
8.13	Help	411

9	User Interface: FELES Application	413
9.1	System Options	413
9.2	View Active Schedule	413
9.3	View Alternative Schedule	413
9.3.1	View Alternative Schedule When No Schedules have been Read . . .	414
9.3.2	View Alternative Schedule When Schedules have been Read	414
9.4	Change Window of Time	414
9.5	View Schedule by Activity	415
9.6	View Schedule by Load Center	415
9.7	Activity Information	416
9.8	Load Information	416
9.9	Help	417
10	User Interface: Activity Editor Application	419
10.1	System Options	419
10.2	Edit	419
10.2.1	Activity	419
10.2.2	Subtask	420
10.2.3	Requirement	420
10.2.4	Powered Equipment	420
10.3	Create	421
10.4	Delete	421
10.5	Save	422
10.6	Create Schedule	422
10.7	Delete Schedule	422
10.8	Help	423
11	User Interface: Power Utilization Application	425
11.1	System Options	425
11.2	Change Window of Time	425
11.3	System Power Usage	425
11.4	Load Center Power Usage	426
11.5	Component Power Usage	426
11.6	System Power Availability	427
11.7	Load Center Power Availability	427
11.8	Help	428
12	User Interface: Workboxes	429
12.1	General Workboxes	429
12.1.1	Activate Time Workbox	429

12.1.2	Schedule Activation Confirmation Workbox	431
12.1.3	Halt Time Workbox	432
12.1.4	Source Power Change Workbox	432
12.1.5	Error Information Workbox	433
12.2	FELES Workboxes	433
12.2.1	Change Window of Time	433
12.3	Power Utilization Workboxes	433
12.3.1	Change Window of Time	433
12.4	Activity Editor Workboxes	434
12.4.1	New Activity Workbox	434
12.4.2	New Subtask Workbox	435
12.4.3	New Requirement Workbox	435
12.4.4	New Powered Equipment Workbox	436
12.4.5	New Schedule Workbox	437
12.4.6	New Mode Workbox	437
12.4.7	Availability Workbox	438
12.4.8	Default Workbox	439
12.4.9	Percent Workbox	440
12.4.10	Power Workbox	441
13	User Interface: Menus	443
13.1	General Menus	443
13.1.1	System Options	443
13.1.2	Schedule Menu	443
13.2	Power System Menus	443
13.2.1	Component Information Menu	443
13.3	FELES Menus	444
13.3.1	Load Center Menu	444
13.4	Power Utilization Menus	444
13.4.1	Load Center Menu	444
13.4.2	Component Usage Menu	444
13.5	Activity Editor Menus	444
13.5.1	Create Menu	444
13.5.2	Edit/Delete Menu	444
13.5.3	Select Activities Menu	444
13.5.4	Augmented Power Menu	445
14	Intermediate Levels of Autonomy	447
14.1	Maintenance Mode	447
14.2	ILA Test for Group Switches	448

14.3	ILA Test for Redundant Switches	458
14.4	ILA Test to Flip/Flop Redundants	469
14.5	ILA Test for Group Manual Seizure	473
15	Source Power Change Test	479
16	Soft Fault Test	487
17	Incipient Fault Test	489
18	Hypothetical Scheduler Test	491
19	Return to Idle Mode	495
20	Shutdown	497
20.0.1	Power System	497
20.0.2	Solbourne	497

(This Page Intentionally Left Blank)

List of Figures

1	System Options Button	381
2	System Options Menu	381
3	System Options Menu with Schedule Functions Submenu	382
4	System Options Menu with Schedule Functions and Activate a Schedule Sub- menus	383
5	Selecting a Schedule	384
6	Activate Time Workbox	385
7	Schedule Activation Confirmation Workbox	386
8	Typical <i>SSM/PMAD</i> Workbox	430

(This Page Intentionally Left Blank)

Purpose

The Test Plan for SSM/PMAD, contract NAS8-36433, program is written to provide guidance for the management and technical effort necessary throughout the test period and to 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. This test plan will demonstrate the capabilities added in Change Order 31 including an updated Knowledge Augmentation and Negotiation Tool (KANT), Intermediate Level of Autonomy (ILA), User Interface, Source Power Change, high impedande faults (Soft Faults) and incipient faults. This plan will also demonstrate the capabilities added in change order 37 including an increased capability in the ILA mode of testbed operation and expanded graphical user interface (GUI).

Project References

The documents utilized by this contract are:

1. Contract Agreement, NAS8-36433, dated June 25, 1985 (including Change Orders: 9, 17, 26, 29, 31, 37, 44 and 49)
2. Software Development Plan for SSM/PMAD, Revision E, dated April 1994
3. SSM/PMAD Requirements Document, Revision B, dated March 1994
4. MCR-85-706, Interim Final Report, Appendix I Volume II, SSM/PMAD Interface User Manual, Version 1.0

(This Page Intentionally Left Blank)

1 Acronyms

A/D Analog to Digital

AI Artificial Intelligence

COTS Commercial Off the Shelf

FELES Front End Load Enable Scheduler

FRAMES Fault Recovery and Management Expert System

FT Fast Trip

GC Generic Controller

GUI Graphical User Interface

ILA Intermediate Levels of Autonomy

KANT Knowledge Augmentation and Negotiation Tool

KBMS Knowledge-Based Management System

KNOMAD Knowledge Management Design System

LC Load Center

LES Load Enable Schedule

LLP Lowest Level Processor

LPL Load Priority List

LPLMS Load Priority List Maintenance System

MAESTRO Master of Expert Scheduling through Resource Orchestration

MSFC Marshall Space Flight Center

OC Over Current

RPC Remote Power Controller

SIC Switchgear Interface Card

SSL Software Support Library

SSM/PMAD Space Station Module Power Management and Distribution

TWM Tom's Window Manager

UI User Interface

UV Under Voltage

2 Hardware

The SSM/PMAD software will exist on the Solbourne 5/501 general purpose workstation processors, and the 80386 Lowest Level Processor (LLP) processors. The SSM/PMAD communications architecture (Figure 2.1-1) is identified below:

The power system consists of up to eight LLP's. Each LLP contains a computer controlling at least one Switchgear Interface Cards (SIC). Each SIC card has the capability of interfacing the computer to an Analog to Digital (A/D) Card and Generic Controller (GC) cards, which control the switches.

(This Page Intentionally Left Blank)

3 Software

The configuration of the Commercial Off-the-Shelf (COTS) software is identified below:

1. MS-Dos Operating System, version 3.30, 849CMP91100-330
2. Multiline Port Interrupt Driver, version 2.22, 849CMP91200-222
3. PC Products Installation Program, version 1.0, 849CMP91400-100
4. TCP/IP for Dos CMC-640, version 4.1, 849CMP91400-410
5. Solbourne Operating System, version 4.0C, 849CMP92100-40C
6. Allegro X11R4, release R4, 849CMP92400-R30
7. Allegro Common Lisp, release 4.1, 849CMP92400-314
8. Allegro Common Windows on X, release 2.0 Pilot, 849CMP92400-130

The configuration of the applications software is identified below:

1. Solbourne Applications Software, 849CMP11000-008
2. LLP Software, 849CMP14000-007

This test is designed to demonstrate the capabilities of the SSM/PMAD system. It is intended to satisfy the requirements of Task III. All test requirements are satisfied by demonstration.

(This Page Intentionally Left Blank)

4 Test Data

The as-run test procedure, test input, and test output will be maintained to ensure repeatability of tests.

4.1 Input Data

Input data are controlled by this test in order to demonstrate system capabilities.

4.2 Output Data

Output data will consist of observable events in adherence to the listed responses. Any output data received from the test will be placed in the Software Support Library (SSL).

(This Page Intentionally Left Blank)

5 Startup Test Procedures

5.1 Test Setup

Power System The power system of the SSM/PMAD must be turned on for the demonstration to be run. This includes Housekeeping power, main power, and power to the LLP's.

Action	Response
1. Turn on housekeeping power to the rack.	Power is turned on.
2. Bring up the 120v DC main power for data buses.	Power is turned on.
3. Power up the LLP's. Make sure each LLP has a disk in its drive.	
a. Make sure that LLP-A has the LLP-A disk in its drive and that LLP-C has the LLP-C disk in its drive.	
b. Turn on the power to the LLP's.	LLP's without attached keyboards will beep.
Q_____	

Communications, User Interface and FRAMES Software - Solbourne

Action	Response
1. To load the software on the Solbourne, the steps will be performed at the "%" prompt.	
a. Type "cd /usr/local/lib"	
b. Type "mt -f /dev/rst8 rew"	
c. Type "tar -xvf /dev/nrst8"	
d. Type "tar -xvf /dev/rst8"	Software is loaded.
2. Change any special initialization scripts so that "usr/local/lib/ssmpmad.cl" is the last item executed (e.g., .xinitrc of pmad1)	

Q_____

5.2 Test Initialization

To operate the SSM/PMAD breadboard, the operator will bring up the system in the order of the power system, then MAESTRO and a schedule, and finally the Solbourne interface.

Initializing the SSM/PMAD Interface and FRAMES

Action	Response
1. Login to the Solbourne as PMAD1	UNIX prompt appears.
Q_____	
2. Type "ssmpmad" at the UNIX prompt	X windows and the TWM window manager will start. The X initialization start LISP. The LISP initialization file will automatically load and start the SSM/PMAD interface.
Q_____	

6 User Interface: How to Execute the Test Procedures

This section of the test procedure is designed to test the main features of the user interface. It is broken up into 8 sections, System Options Menu, Power System Application, FELES Application, Activity Editor Application, Power Utilization Application, Workboxes, Menus, and Intermediate Levels of Autonomy. This test procedure assumes a working knowledge of the *SSM/PMAD* testbed and software.

The user interface test procedure is meant to be an exhaustive test of all parts of the user interface. As a tester who is familiar with the entire *SSM/PMAD* you may decide that some of the sections are not affected by the code to be tested. With careful discretion, the test may choose only sections of the following user interface test plan.

The test procedure is hierarchical in nature. This means that in order to complete a given section, the user must complete all of the sections above the given section. In simple terms, a procedure like Go To Normal in section 7.1.2 can only be executed if System Options and Mode Changes Submenu in sections 7 and 7.1 respectively have been executed.

As an example, let's go through the procedure for testing the Schedule Activation Confirmation Workbox in section 12.1.2.

Schedule Activation Confirmation Workbox (Section 12.1.2)

Action	Response
1. Follow the procedure documented in section 12.1.1 for activating a schedule through step 2.	The Schedule Activation Confirmation Workbox is on the screen.

First, the procedure in section 12.1.1 must be followed through its step 2.

Activate Time Workbox (Section 12.1.1)

Action	Response
1. Follow the procedure documented in section 7.2.2 for activating a schedule through step 2.	The Activate Time Workbox is on the screen.
2. Mouse button click on the ASAP Button.	The ASAP Button will highlight, the Activate Time Workbox will disappear, and the selected schedule will be read, if needed. The Schedule Activation

Confirmation workbox
will be displayed.

Before this procedure may be implemented, the procedure for activating a schedule must be exercised in section 7.2.2. This procedure must be followed through step 2.

Activate a Schedule (Section 7.2.2)

Action

1. Move mouse cursor to *Activate a Schedule* item in the Schedule Functions Submenu.
2. Mouse button click on any schedule within Activate a Schedule Submenu.

Response

Activate a Schedule region will highlight and the Activate a Schedule Submenu appears. The selected schedule will highlight, the System Options Menu will disappear, and the selected schedule will be read and pop up the *Schedule Activation Time* workbox.

It should be noted that to exercise the procedure for activating a schedule, the system must be in **NORMAL** mode of operation. For this example, this is assumed to be the case. It has been stated, that in order to execute a procedure all the procedures hierarchically above it must also be executed. This means that the System Options section and the Schedule Functions section, sections 7 and 7.2 respectively, must be executed first.

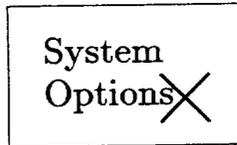


Figure 1: System Options Button

System Options (Section 7)

Action

1. Mouse button click on System Options region of the User Interface.

Response

System Options region will highlight and the System Options menu appears.

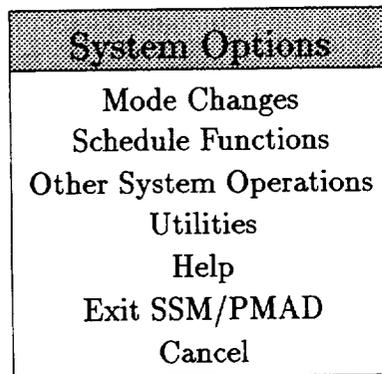


Figure 2: System Options Menu

Schedule Functions (Section 7.2)**Action**

1. Move mouse cursor to *Schedule Functions* item in the System Options Menu.

Response

Schedule Functions region will highlight and the Schedule Functions Submenu appears.

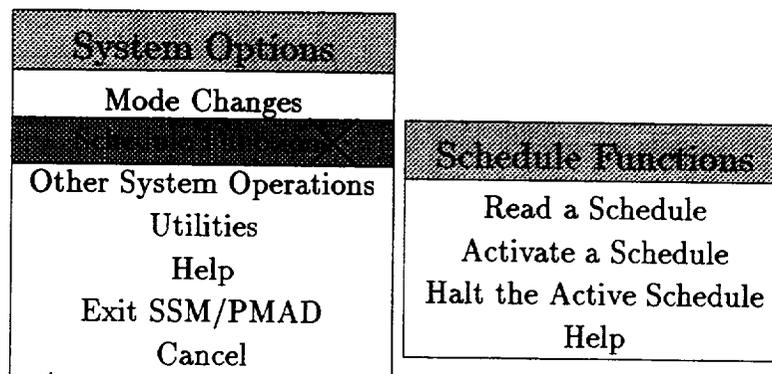


Figure 3: System Options Menu with Schedule Functions Submenu

Activate a Schedule (Section 7.2.2)

Action

1. Move mouse cursor to *Activate a Schedule* item in the Schedule Functions Submenu.

Response

Activate a Schedule region will highlight and the Activate a Schedule Submenu appears.

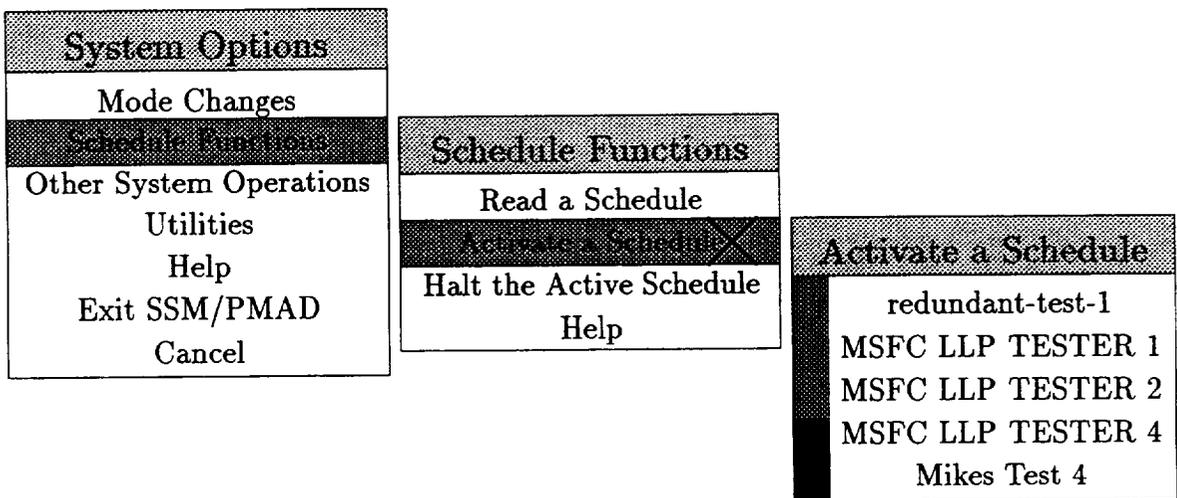


Figure 4: System Options Menu with Schedule Functions and Activate a Schedule Submenus

Activate a Schedule (Section 7.2.2)

Action

2. Mouse button click on any schedule within Activate a Schedule Submenu.

Response

The selected schedule will highlight, the System Options Menu will disappear, and the selected schedule will be read and pop up the *Schedule Activation Time* workbox.

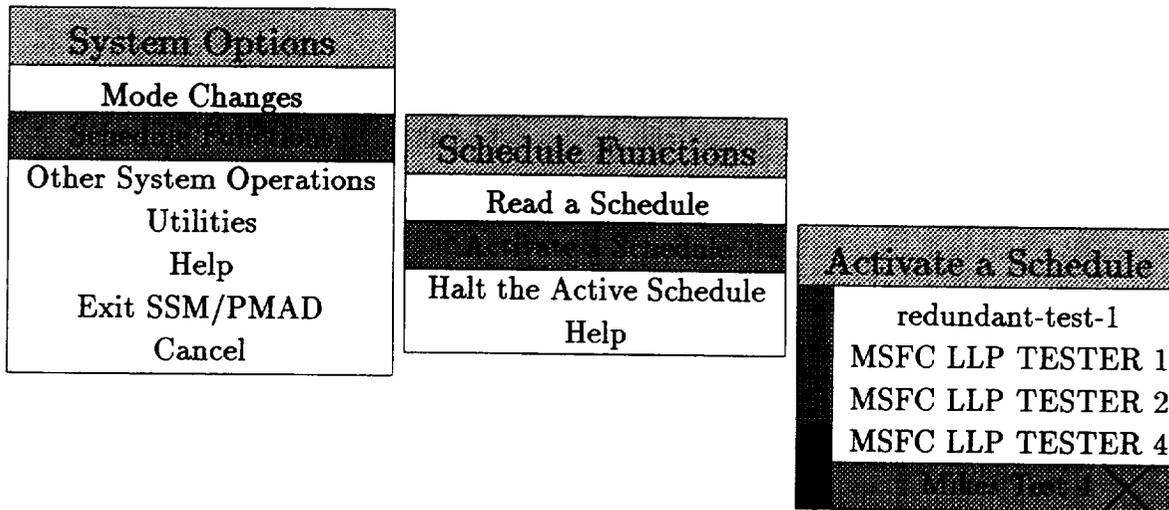


Figure 5: Selecting a Schedule

Activate Time Workbox (Section 12.1.1)**Action**

1. Follow the procedure documented in section 7.2.2 for activating a schedule through step 2.
2. Mouse button click on the **ASAP** Button.

Response

The Activate Time Workbox is on the screen.

The **ASAP** Button will highlight, the Activate Time Workbox will disappear, and the selected schedule will be read, if needed. The Schedule Activation workbox will be displayed.

Screenshot of the Schedule Activation Time dialog box. The title bar reads "Schedule Activation Time". Inside, there is a section titled "Activation Time" with three input fields: "Day: 0", "Hour: 0", and "Minute: 43". At the bottom, there are three buttons: "ASAP" (highlighted with a dark background), "Done", and "Cancel".

Figure 6: Activate Time Workbox

Schedule Activation Confirmation Workbox (Section 12.1.2)

- | Action | Response |
|--|--|
| 1. Follow the procedure documented in section 12.1.1 for activating a schedule through step 2. | The Schedule Activation Confirmation Workbox is on the screen. |
| 2. Mouse button click on the Done Button. | The Done Button will highlight, the Schedule Activationx Workbox will disappear, and the selected schedule will be activated. |

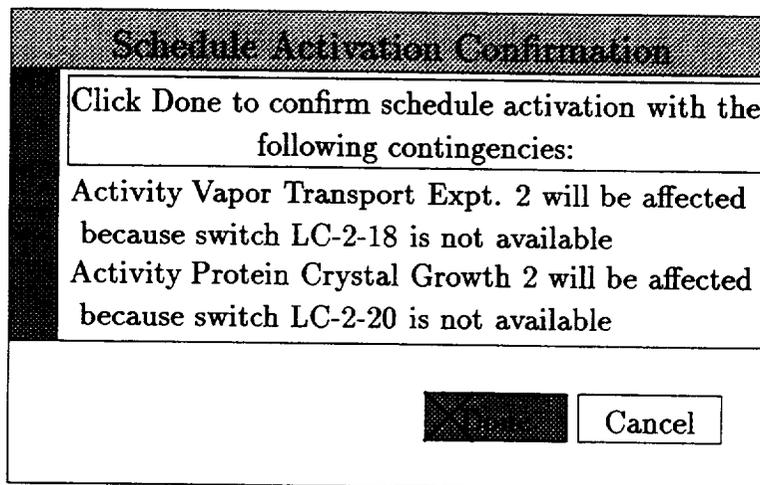


Figure 7: Schedule Activation Confirmation Workbox

7 User Interface: System Options Menu

Action

1. Mouse button click on System Options region of the User Interface.

Q_____

Response

System Options region will highlight and the System Options menu appears.

7.1 Mode Changes Submenu

Action

1. Move mouse cursor to *Mode Changes* item in the System Options Menu.

Q_____

Response

Mode Changes region will highlight and the Mode Changes Submenu appears.

7.1.1 Go To Maintenance

Action

1. Mouse button click on *Go To Maintenance*.

Q_____

Response

Go To Maintenance will highlight and the *SSM/PMAD* testbed will go to **Maintenance** mode of operation. The System Options Menu will disappear.

7.1.2 Go To Normal

Action

1. Mouse button click on *Go To Normal*.

Response

Go To Normal will highlight and the *SSM/PMAD* testbed will go to **Normal**

mode of operation.
The System Options Menu
will disappear.

Q_____

7.1.3 Go To Idle

Action

1. Mouse button click on *Go To Idle*.

Response

Go To Idle
will highlight and
the *SSM/PMAD* testbed
will go to **Idle**
mode of operation.
The System Options Menu
will disappear.

Q_____

7.1.4 Help

Action

1. Mouse button click on *Help*.

Response

Help will highlight
and the help for the
Mode Changes Submenu
will be displayed in the
application scratchpad.
The System Options Menu
will disappear.

Q_____

7.2 Schedule Functions Submenu

Action

1. Move mouse cursor to *Schedule Functions*
item in the System Options Menu.

Response

Schedule Functions
region will highlight
and the Schedule Functions
Submenu appears.

Q_____

7.2.1 Read a Schedule

Action	Response
1. Move mouse cursor to <i>Read a Schedule</i> item in the Schedule Functions Submenu.	<i>Read a Schedule</i> region will highlight and the Read a Schedule Submenu appears.
2. Mouse button click on any schedule within Read a Schedule Submenu.	The selected schedule will highlight, the System Options Menu will disappear, and the selected schedule will have been read.

Q_____

7.2.2 Activate a Schedule

This operation will only work in **NORMAL** mode of operation.

Action	Response
1. Move mouse cursor to <i>Activate a Schedule</i> item in the Schedule Functions Submenu.	<i>Activate a Schedule</i> region will highlight and the Activate a Schedule Submenu appears.
2. Mouse button click on any schedule within Activate a Schedule Submenu.	The selected schedule will highlight, the System Options Menu will disappear, and the selected schedule will be read and pop up the <i>Schedule Activation Time</i> workbox.
3. Mouse button click the Cancel button within the <i>Schedule Activation Time</i> workbox.	The Cancel button will highlight and the workbox will be waved off.

Q_____

7.2.3 Halt the Active Schedule

This operation will only work in **NORMAL** mode of operation.

- | Action | Response |
|---|---|
| 1. Mouse button click <i>Halt the Active Schedule</i> item within Schedule Functions Submenu. | <i>Halt the Active Schedule</i> will highlight, the System Options Menu will disappear, and the Halt the Active Schedule workbox will appear. |
| 2. Mouse button click the Cancel button within the Halt the Active Schedule workbox. | The Cancel button will highlight and the workbox will be disappear. |

Q_____

7.2.4 Help

- | Action | Response |
|--|---|
| 1. Mouse button click on <i>Help</i> . | <i>Help</i> will highlight and the help for the Schedule Functions Submenu will be displayed in the application scratchpad. The System Options Menu will disappear. |

Q_____

7.3 Other System Operations Submenu

- | Action | Response |
|---|---|
| 1. Move mouse cursor to <i>Other System Operations</i> item in the System Options Menu. | <i>Other System Operations</i> region will highlight and the Other System Operations Submenu appears. |

Q_____

7.3.1 Connect to LLP

This option is only available in *Maintenance Mode* of operation.

Action	Response
1. Move mouse cursor to <i>Connect to LLP</i> item in the Other System Operations Submenu.	<i>Connect to LLP</i> region will highlight and the Connect to LLP Submenu appears.
2. Mouse button click on any available LLP within the Connect to LLP Submenu.	The selected LLP menu item will highlight, the System Options Menu will disappear, and the <i>SSM/PMAD</i> testbed will attempt to connect to the selected LLP.

Q_____

7.3.2 Disconnect from LLP

This option is only available in *Maintenance Mode* of operation.

Action	Response
1. Move mouse cursor to <i>Disconnect from LLP</i> item in the Other System Operations Submenu.	<i>Disconnect from LLP</i> region will highlight and the Disconnect from LLP Submenu appears.
2. Mouse button click on any available LLP within the Disconnect to LLP Submenu.	The selected LLP menu item will highlight, the System Options Menu will disappear, and the <i>SSM/PMAD</i> testbed will disconnect from the selected LLP.

Q_____

7.3.3 Connect to APEX

Action	Response
1. Mouse button click on <i>Connect to APEX</i>	The <i>Connect to Apex</i>

within the Other System Operations Submenu.

region will highlight, the System Options Menu will disappear, and the *SSM/PMAD* testbed will attempt to connect to the selected APEX.

Q_____

7.3.4 Disconnect from APEX

Action

1. Mouse button click on *Disconnect from APEX* within the Other System Operations Submenu.

Response

The *Disconnect from Apex* region will highlight, the System Options Menu will disappear, and the *SSM/PMAD* testbed will disconnect from the selected APEX.

Q_____

7.3.5 Source Power Change

Action

1. Mouse button click on *Source Power Change* within the Other System Operations Submenu.
2. Waveoff the Source Power Change Workbox by mouse button clicking outside the workbox.

Response

The *Source Power Change* region will highlight, the System Options Menu will disappear, and the Source Power Change workbox will appear. The Source Power Change workbox disappears.

Q_____

7.3.6 Return RPCs to Service

Action

1. Mouse button click on *Return RPCs to Service*

Response

The *Return RPCs to Service*

within the Other System Operations Submenu.

2. Acknowledge the error workbox by mouse button clicking the okay button in the workbox.

Q_____

region will highlight, the System Options Menu will disappear. An error workbox will appear, telling the user no switches are presently out of service. The error workbox disappears.

7.3.7 Clear Soft Faults

Action

1. Mouse button click on *Clear Soft Faults* within the Other System Operations Submenu.
2. Acknowledge the error workbox by mouse button clicking the okay button in the workbox.

Q_____

Response

The *Clear Soft Faults* region will highlight, the System Options Menu will disappear. An error workbox will appear, telling the user no switches are presently soft faulted. The error workbox disappears.

7.3.8 Clear Incipient Faults

Action

1. Mouse button click on *Clear Incipient Faults* within the Other System Operations Submenu.

Response

The *Clear Incipient Faults* region will highlight, the System Options Menu will disappear. An error workbox will appear, telling the user no powered equipment items

2. Acknowledge the error workbox by mouse button clicking the okay button in the workbox.
- are presently incipient faulted. The error workbox disappears.

Q_____

7.3.9 Current Fault State

Action

Not presently implemented.

Response

7.3.10 Help

Action

1. Mouse button click on *Help*.

Response

Help will highlight and the help for the Other System Operations Submenu will be displayed in the application scratchpad. The System Options Menu will disappear.

Q_____

7.4 Utilities Submenu

Action

1. Move mouse cursor to *Utilities* item in the System Options Menu.

Response

Utilities region will highlight and the Utilities Submenu appears.

Q_____

7.4.1 Clear Scratchpad

Action

Response

1. Mouse button click on *Clear Scratchpad*.

Clear Scratchpad will highlight and the application scratchpad will be cleared of data. The System Options Menu will disappear.

Q_____

7.4.2 Enable Sensor Balancing

Action

1. Mouse button click on *Enable Sensor Balancing*.

Response

Enable Sensor Balancing will highlight and the fault detection system will attempt to find sensor balancing faults. The System Options Menu will disappear.

Q_____

7.4.3 Disable Sensor Balancing

Action

1. Mouse button click on *Disable Sensor Balancing*.

Response

Disable Sensor Balancing will highlight and the fault detection system will no longer attempt to find sensor balancing soft faults. The System Options Menu will disappear.

Q_____

7.4.4 Disable SysFlow Highlighting

Action

1. Mouse button click on *Disable SysFlow Highlighting*.

Response

Disable SysFlow Highlighting will highlight and the

System Flow Window will no longer Highlight data. The System Options Menu will disappear.

Q_____

7.4.5 Enable SysFlow Highlighting

Action

1. Mouse button click on *Enable SysFlow Highlighting*.

Response

Enable SysFlow Highlighting will highlight and the System Flow Window will again Highlight data. The System Options Menu will disappear.

Q_____

7.4.6 Disable GC Cursor

Action

1. Mouse button click on *Disable GC Cursor*.

Response

Disable GC Cursor will highlight and the LISP Garbage Collection will no longer appear during a Global Garbage collect. The System Options Menu will disappear.

Q_____

7.4.7 Enable GC Cursor

Action

1. Mouse button click on *Enable GC Cursor*.

Response

Enable GC Cursor will highlight and the LISP Garbage Collection will again appear during a Global Garbage collect. The System

Options Menu will disappear.

Q_____

7.4.8 Screendump

Action

1. Mouse button click on *Screendump*.

Response

Screendump will highlight and a raster image bitmap of the whole user interface screen will be placed in the logged user's ssmpmad-if-archive directory. The System Options Menu will disappear.

Q_____

7.4.9 Windowdump

Action

1. Mouse button click on *Windowdump*.
2. Move the mouse cursor to the window to be dumped and mouse button click.

Response

Windowdump will highlight and the System Options Menu will disappear. A crosshairs mouse cursor will appear.

The selected window will copied into a raster image bitmap and placed in the logged user's ssmpmad-if-archive directory.

Q_____

7.4.10 Help

Action

Response

1. Mouse button click on *Help*.

Help will highlight and the help for the Utilities Submenu will be displayed in the application scratchpad. The System Options Menu will disappear.

Q_____

7.5 Help

Action

1. Mouse button click on *Help*.

Response

Help will highlight and the help for the System Options Menu will be displayed in the application scratchpad. The System Options Menu will disappear.

Q_____

7.6 Exit From *SSM/PMAD*

This System Options menu option is only available in **IDLE** mode of operation. To put the user interface in this mode of operation, follow the procedure in section 7.1.3.

Action

1. Move mouse cursor to *Exit From SSM/PMAD* item in the System Options Menu.

Response

Exit From SSM/PMAD region will highlight and a confirmation pull-right menu will appear.

Q_____

7.6.1 Yes

Action

Response

1. Mouse button click on *Yes*.

Yes will highlight and the System Options Menu will disappear and the Activity Editor databases will be saved and the *SSM/PMAD* software will exit to the Unix System.

Q_____

7.7 Cancel

Action

1. Mouse button click on *Cancel*.

Response

Cancel will highlight and the System Options Menu will disappear.

Q_____

(This Page Intentionally Left Blank)

8 User Interface: Power System Application

Action

1. Mouse button click on the *Power System* item of the Applications Selection Menu.

Response

Power System region will highlight and the Power System Application appears. The Power System item will become unavailable for mouse button clicks and the previously displayed application will become available in the Applications Selection Menu.

Q_____

8.1 System Options

Handled in Section 7.

8.2 Detailed Data

This application menu option is only accessible in either **MAINTENANCE** or **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **MAINTENANCE** or **NORMAL** mode of operation follow procedure in section 7.1.1 or section 7.1.2 respectively.

Action

1. Mouse button click on as many switch icons and sensor icons on the application screen as desired.
2. Mouse button click on the *Detailed Data* region in the Power System Application Menu.

Response

Each selected switch or sensor will display a check mark signifying that it has been selected.

For each selected switch or sensor, the Power System Application will display Detailed Data on the switch or sensor to the

application scratchpad.

Q_____

8.3 Detailed Data Point Down

This application menu option is only accessible in either **MAINTENANCE** or **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **MAINTENANCE** or **NORMAL** mode of operation follow procedure in section 7.1.1 or section 7.1.2 respectively.

Action	Response
1. Mouse button click on a switch or sensor icon in each PDCU on the application screen.	Each selected switch or sensor will display a check mark signifying that it has been selected.
2. Mouse button click on the <i>Detailed Data Point Down</i> region in the Power System Application Menu.	For each selected switch or sensor, the Application will display Detailed Data on the switch or sensor to the application scratchpad. In addition, each switch or sensor topologically below the selected switches or sensors will also have its Detailed Data displayed to the application scratchpad.

Q_____

8.4 Deselect All

This application menu option is only accessible in either **MAINTENANCE** or **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **MAINTENANCE** or **NORMAL** mode of operation follow procedure in section 7.1.1 or section 7.1.2 respectively.

Action

Response

- | | |
|---|---|
| 1. Mouse button click as many switch and sensor icons on the application screen as desired. | Each selected switch or sensor will display a check mark signifying that it has been selected. |
| 2. Mouse button click on the <i>Deselect All</i> region in the Power System Application Menu. | Each selected item on the Power System Application Screen will be deselected and the signifying check marks will disappear. |

Q_____

8.5 Manual On

This application menu option is only accessible in either **MAINTENANCE** or **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **MAINTENANCE** or **NORMAL** mode of operation follow procedure in section 7.1.1 or section 7.1.2 respectively. *Manual On* operates differently in **MAINTENANCE** mode and **NORMAL** mode. As such, each case will be covered separately.

8.5.1 Manual On in **MAINTENANCE** mode

- | Action | Response |
|--|---|
| 1. Mouse button click on as many switch icons on the application screen as desired. | Each selected switch or sensor will display a check mark signifying that it has been selected. |
| 2. Mouse button click on the <i>Manual On</i> region in the Power System Application Menu. | Each selected switch on the Power System Application Screen will be manually commanded on. Selected switches which are already on will remain that way. |

Q_____

8.5.2 Manual On in NORMAL mode

When operating in **NORMAL** mode of operation, switches must first be manually seized from the automated scheduling system before they may be manually commanded on. Manual seizure is described in section 8.7.

Action	Response
1. Mouse button click on as many switch icons on the application screen as desired.	Each selected switch will display a check mark signifying that it has been selected.
2. Mouse button click on the <i>Manual On</i> region in the Power System Application Menu.	Each selected switch on the Power System Application Screen will be manually commanded on, if it was previously manually seized. Selected switches not under manual control will be displayed in a pop up error workbox. This workbox must be acknowledged with a mouse button click to its OKAY button. Previously manually seized selected switches which were already on will stay on.

Q_____

8.6 Manual Off

This application menu option is only accessible in either **MAINTENANCE** or **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **MAINTENANCE** or **NORMAL** mode of operation follow procedure in section 7.1.1 or section 7.1.2 respectively. *Manual Off* operates differently in **MAINTENANCE** mode and **NORMAL** mode. As such, each case will be covered separately.

8.6.1 Manual Off in MAINTENANCE mode

Action	Response
1. Mouse button click on as many switch icons on the application screen as desired.	Each selected switch or sensor will display a check mark signifying that it has been selected.
2. Mouse button click on the <i>Manual Off</i> region in the Power System Application Menu.	Each selected switch on the Power System Application Screen will be manually commanded off. Selected switches which are already off will remain that way.

Q_____

8.6.2 Manual Off in NORMAL mode

When operating in **NORMAL** mode of operation, switches must first be manually seized from the automated scheduling system before they may be manually commanded off. Manual seizure is described in section 8.7.

Action	Response
1. Mouse button click on as many switch icons on the application screen as desired.	Each selected switch will display a check mark signifying that it has been selected.
2. Mouse button click on the <i>Manual Off</i> region in the Power System Application Menu.	Each selected switch on the Power System Application Screen will be manually commanded off, if it was previously manually seized. Selected switches not under manual control will be displayed in a pop up error workbox. This workbox must be acknowledged with a mouse button click to its OKAY

button.

Previously manually seized selected switches which were already off will stay off.

Q_____

8.7 Seize Manual Control

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action

1. Mouse button click on the *Seize Manual Control* region in the Power System Application Menu.
2. Waveoff this menu by mouse button clicking outside the menu.

Response

A menu containing LLP submenus with selectable switches and previously defined switch groups will appear.

The seizure menu will disappear.

Q_____

8.8 Update Manual Control

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action

1. Mouse button click on the *Update Manual Control* region in the Power System Application Menu.
2. Waveoff this menu by mouse button clicking

Response

A menu containing LLP submenus with previously switches and switch groups will appear.

The update menu will

outside the menu.

disappear.

Q_____

8.9 Release Manual Control

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action	Response
1. Mouse button click on the <i>Release Manual Control</i> region in the Power System Application Menu.	A menu containing LLP submenus with selectable seized switches and seized switch groups will appear.
2. Waveoff this menu by mouse button clicking outside the menu.	The release menu will disappear.

Q_____

8.10 Component Information

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action	Response
1. Mouse button click on a switch or sensor icon in each PDCU on the application screen.	Each selected switch or sensor will display a check mark signifying that it has been selected.
2. Mouse button click on the <i>Component Information</i> region in the Power System Application Menu.	A menu containing <i>Load Information</i> , <i>Load Information Point Down</i> , <i>Switch Schedule</i> ,

*Power Utilization, and
Power Utilization Point Down*
will appear.

Q_____

8.10.1 Load Information

Action

1. Mouse button click on the *Load Information* region in the Component Information Menu.

Response

For each selected switch, Load Information pertaining to that switch is displayed in the application scratchpad. The Component Information Menu disappears.

Q_____

8.10.2 Load Information Point Down

Action

1. Mouse button click on the *Load Information Point Down* region in the Component Information Menu.

Response

For each selected switch, Load Information pertaining to that switch is displayed in the application scratchpad. In addition, all switches which fall below the selected switches in the power system topology will have their Load Information displayed to application scratchpad. The Component Information Menu disappears.

Q_____

8.10.3 Power Utilization

Action

1. Mouse button click on the *Power Utilization* region in the Component Information Menu.

Response

For each selected switch, Power Utilization data for that switch is displayed in graphical format to the application scratchpad. The Component Information Menu disappears.

Q_____

8.10.4 Switch Schedule

Action

1. Mouse button click on the *Switch Schedule* region in the Component Information Menu.

Response

For each selected switch, a Switch Schedule chart for that switch is displayed in the application scratchpad. The Component Information Menu disappears.

Q_____

8.10.5 Switch Schedule Point Down

Action

1. Mouse button click on the *Switch Schedule Point Down* region in the Component Information Menu.

Response

For each selected switch, a Switch Schedule pertaining to that switch is displayed in the application scratchpad. In addition, all switches which fall below the selected switches in the power system topology will have their Switch Schedule displayed to application scratchpad. The Component Information

Menu disappears.

Q_____

8.11 Group Switches

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action	Response
1. Mouse button click on the <i>Group Switches</i> region in the Power System Application Menu.	A menu containing LLP submenus with selectable switches will appear.
2. Waveoff this menu by mouse button clicking outside the menu.	The Group Switches menu will disappear.

Q_____

8.12 Ungroup Switches

This application menu option is only accessible in **NORMAL** mode of operation. Then only when Lowest Level Processors are connected and switches visible on the interface. To put the user interface in **NORMAL** mode of operation follow procedure in section 7.1.2.

Action	Response
1. Mouse button click on the <i>Ungroup Switches</i> region in the Power System Application Menu.	A menu containing previously defined switch groups will appear.
2. Waveoff this menu by mouse button clicking outside the menu.	The Ungroup Switches menu will disappear.

Q_____

8.13 Help

- | Action | Response |
|--|---|
| 1. Mouse button click on <i>Help</i> . | Help for the Power System Application Screen will be displayed in the application scratchpad. |

Q_____

(This Page Intentionally Left Blank)

9 User Interface: FELES Application

Action	Response
1. Mouse button click on the <i>FELES</i> item of the Applications Selection Menu.	<i>FELES</i> region will highlight and the FELES Application appears. The FELES item will become unavailable for mouse button clicks and the previously displayed application will become available in the Applications Selection Menu.

Q_____

9.1 System Options

Handled in Section 7.

9.2 View Active Schedule

Action	Response
1. Mouse button click on the <i>View Active Schedule</i> region of the Application Menu.	<i>View Active Schedule</i> region will highlight and the active schedule will be computed. A new gantt chart will be generated.

Q_____

9.3 View Alternative Schedule

This application menu option can have two different responses, depending upon whether or not any previously constructed schedules have been read in to the *SSM/PMAD* System. Each case will be handled separately.

9.3.1 View Alternative Schedule When No Schedules have been Read

Action

1. Mouse button click on the *View Alternative Schedule* region of the Application Menu.
2. Mouse button click on the **OKAY** button of the error workbox to acknowledge the error.

Q_____

Response

View Alternative Schedule region will highlight and an error workbox will be displayed informing the user that No Schedules have been read into the system.

The error workbox will disappear.

9.3.2 View Alternative Schedule When Schedules have been Read

Action

1. Mouse button click on the *View Alternative Schedule* region of the Application Menu.
2. Mouse button click on the desired schedule for viewing.

Q_____

Response

View Alternative Schedule region will highlight and a menu of read in schedules will be displayed.

A new gantt chart will be generated for the selected schedule.

9.4 Change Window of Time

Action

1. Mouse button click on the *Change Window Of Time* region of the Application Menu.

Response

Change Window Of Time region will highlight and the Change Window of Time workbox will be displayed.

2. Mouse button click on the **Cancel** button to waveoff the workbox.

The Change Window of Time workbox will disappear.

Q_____

9.5 View Schedule by Activity

Action

1. Mouse button click on the *View Schedule by Activity* region of the Application Menu.

Response

View Schedule by Activity region will highlight and a gantt chart for the currently displayed schedule will be generated in terms of Activities and displayed to the application main window.

Q_____

9.6 View Schedule by Load Center

Action

1. Mouse button click on the *View Schedule by Load Center* region of the Application Menu.
2. Mouse button click on the desired load center in the Load Center Menu.

Response

View Schedule by Load Center region will highlight and a menu of available load centers will appear.

The selected load center item will highlight, the Load Center Menu will disappear and a gantt chart for the currently displayed schedule will be generated in terms of the switches within the selected load center and displayed to the application main window.

Q_____

9.7 Activity Information

Action	Response
1. Mouse button click on a desired gantt bar.	The gantt bar will change color indicating that it has been selected.
2. Mouse button click on the <i>Activity Information</i> region of the Application Menu.	<i>Activity Information</i> region will highlight and the FELES application will display all of the powered equipment modes of operation for the selected gantt bar. This information is displayed in the FELES application scratchpad.

Q_____

9.8 Load Information

Action	Response
1. Mouse button click on a desired gantt bar.	The gantt bar will change color indicating that it has been selected.
2. Mouse button click on the <i>Activity Information</i> region of the Application Menu.	<i>Activity Information</i> region will highlight and the FELES application will display only the powered equipment mode of operation for the selected gantt bar. This information is displayed in the FELES application scratchpad.

Q_____

9.9 Help

- Action**
1. Mouse button click on *Help*.

Response
Help for the FELES
Application Screen will
be displayed in the
application scratchpad.

Q_____

(This Page Intentionally Left Blank)

10 User Interface: Activity Editor Application

Action

1. Mouse button click on the *Activity Editor* item of the Applications Selection Menu.

Response

Activity Editor region will highlight and the Activity Editor Application appears. The Activity Editor item will become unavailable for mouse button clicks and the previously displayed application will become available in the Applications Selection Menu.

Q_____

10.1 System Options

Handled in Section 7.

10.2 Edit

Action

1. Mouse button click on the *Edit* region in the application menu.

Response

Edit region will highlight and the Edit menu appears.

Q_____

10.2.1 Activity

Action

1. Move mouse cursor on the *Activity* item in the Edit menu.

Response

Activity item will highlight and a submenu of activities appears.

2. Waveoff the *Edit* menu structure by

The Edit menu disappears.

mouse button clicking outside the *Edit* menu.

Q_____

10.2.2 Subtask

Action

1. Move mouse cursor on the *Subtask* item in the Edit menu.
2. Waveoff the *Edit* menu structure by mouse button clicking outside the *Edit* menu.

Response

Subtask item will highlight and a submenu of subtasks appears.

The Edit menu disappears.

Q_____

10.2.3 Requirement

Action

1. Move mouse cursor on the *Requirement* item in the Edit menu.
2. Waveoff the *Edit* menu structure by mouse button clicking outside the *Edit* menu.

Response

Requirement item will highlight and a submenu of requirements appears.

The Edit menu disappears.

Q_____

10.2.4 Powered Equipment

Action

1. Move mouse cursor on the *Powered Equipment* item in the Edit menu.

Response

Powered Equipment item will highlight and

- a submenu of powered equipment appears.
2. Waveoff the *Edit* menu structure by mouse button clicking outside the *Edit* menu.
- Q_____

10.3 Create

- | Action | Response |
|--|---|
| 1. Mouse button click on the <i>Create</i> item in the application menu. | <i>Create</i> item will highlight and a <i>Create</i> menu appears. |
| 2. Waveoff the <i>Create</i> menu structure by mouse button clicking outside the <i>Create</i> menu. | The <i>Create</i> menu disappears. |
- Q_____

10.4 Delete

- | Action | Response |
|--|---|
| 1. Mouse button click on the <i>Delete</i> item in the application menu. | <i>Delete</i> item will highlight and a <i>Delete</i> menu appears. |
| 2. Waveoff the <i>Delete</i> menu structure by mouse button clicking outside the <i>Delete</i> menu. | The <i>Delete</i> menu disappears. |
- Q_____

10.5 Save

Action

1. Mouse button click on the *Save* item in the application menu.

Q_____

Response

The *Save* item will highlight and the Activity Editor saves off the activity and powered equipment databases.

10.6 Create Schedule

Action

1. Mouse button click on the *Create Schedule* item in the application menu.
2. Waveoff the Schedule Selected Activities menu by mouse button clicking outside the menu

Q_____

Response

Create Schedule item will highlight and the Schedule Selected Activities Menu appears.

The Create Schedule menu disappears.

10.7 Delete Schedule

Action

1. Mouse button click on the *Delete Schedule* item in the application menu.
2. Waveoff the *Delete Schedule* menu structure mouse button clicking outside the *Delete Schedule* menu.

Q_____

Response

Delete Schedule item will highlight and a Delete Schedule menu appears.

The Delete Schedule menu disappears.

10.8 Help

Action	Response
1. Mouse button click on <i>Help</i> .	Help for the Activity Editor Application Screen will be displayed in the application scratchpad.

Q_____

(This Page Intentionally Left Blank)

11 User Interface: Power Utilization Application

Action	Response
1. Mouse button click on the <i>Power Utilization</i> item of the Applications Selection Menu.	<i>Power Utilization</i> region will highlight and the Power Utilization Application appears. The Power Utilization item will become unavailable for mouse button clicks and the previously displayed application will become available in the Applications Selection Menu.

Q_____

11.1 System Options

Handled in Section 7.

11.2 Change Window of Time

Action	Response
1. Mouse button click on the <i>Change Window Of Time</i> region of the Application Menu.	<i>Change Window Of Time</i> region will highlight and the Change Window of Time workbox will be displayed.
2. Mouse button click on the Cancel button to waveoff the workbox.	The Change Window of Time workbox will disappear.

Q_____

11.3 System Power Usage

Action	Response
---------------	-----------------

1. Mouse button click on the *System Power Usage* region of the Application Menu.

System Power Usage region will highlight and the Power Utilization application will display *SSM/PMAD* system level power utilization plots. These plots are displayed to the application main window.

Q_____

11.4 Load Center Power Usage

Action

1. Mouse button click on the *Load Center Power Usage* item in the Application Menu.
2. Mouse button click on the load center desired from the Load Center menu.

Response

Load Center Power Usage item will highlight and a Load Center menu appears.

The selected load center highlights and the Load Center menu disappears. The Power Utilization application will display the selected load center level power utilization plots. These plots are displayed to the application main window.

Q_____

11.5 Component Power Usage

Action

1. Mouse button click on the *Component Power Usage* item in the Application Menu.

Response

Component Power Usage item will highlight and the Component Power Usage menu appears.

-
- | | |
|--|--|
| 2. Move mouse cursor into the desired load center item in the Component Power Usage Menu. | The selected load center item will highlight and the switch submenu appears. |
| 3. Mouse button click on the switch component desired from the switch submenu. is displayed in graphical | For the selected switch, Power Utilization data format to the application scratchpad. The Component Power Usage Menu disappears. |

Q_____

11.6 System Power Availability

Action

1. Mouse button click on the *System Power Availability* region of the Application Menu.

Response

System Power Availability region will highlight and the Power Utilization application will display *SSM/PMAD* system level power availability plots. These plots are displayed to the application main window.

Q_____

11.7 Load Center Power Availability

Action

1. Mouse button click on the *Load Center Power Availability* item in the Application Menu.
2. Mouse button click on the load center desired from the Load Center menu.

Response

Load Center Power Availability item will highlight and a Load Center menu appears.

The selected load center highlights and the Load

Q_____

Center menu disappears.
The Power Utilization application will display the selected load center level power availability plots.
These plots are displayed to the application main window.

11.8 Help

Action

1. Mouse button click on *Help*.

Q_____

Response

Help for the Power Utilization Application Screen will be displayed in the application scratchpad.

12 User Interface: Workboxes

Workboxes contain three regions, the scrollable data field region, the error message region, and the button region (see Figure 8). Inside the scrollable data field region is where data interaction takes place. Editable data fields are mousable and may be edited by mouse button clicking on the data field.

In editing a data field, the following commands may be used:

- Ctrl-A - Move to start of line
- Ctrl-B - Move left one character
- Ctrl-D - Delete character to the right
- Ctrl-E - Move to end of line
- Ctrl-F - Move right one character
- Tab - End edit and advance to next edit field, Start edit on new field
- Backspace - Delete character to the left
- Delete - Delete character to the left
- Carriage Return - End edit
- Mouse Button Click on another data field - End edit, Start edit on new field
- Mouse Button Click on a Workbox Button - End edit, Execute Button

Data fields will insert data at the cursor location as it is typed. Some data fields will only accept numeric input. Any errors which are detected in input data are displayed to the error region of the workbook.

12.1 General Workboxes

Workboxes in this section may appear no matter which application is running.

12.1.1 Activate Time Workbook

This workbook is only accessible in **NORMAL** mode of operation. In order to put the *SSM/PMAD* system in **NORMAL** mode of operation follow the procedure in section 7.1.2.

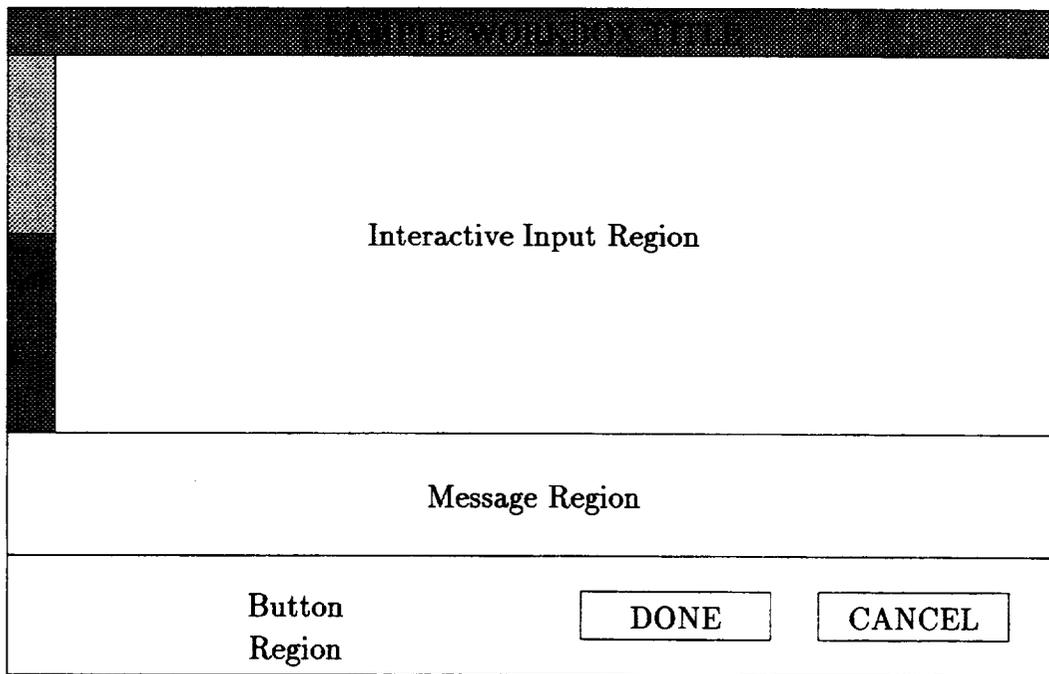


Figure 8: Typical *SSM/PMAD* Workbox

Action	Response
1. Follow the procedure documented in section 7.2.2 for activating a schedule through step 2.	The Activate Time Workbox is on the screen.
2. Mouse button click on the ASAP Button.	The ASAP Button will highlight, the Activate Time Workbox will disappear, and the selected schedule will be read, if needed. The Schedule Activation workbox will be displayed.
3. Proceed to section 12.1.2.	

Q_____

12.1.2 Schedule Activation Confirmation Workbox

This workbox is used to show any schedule anomalies in the schedule to be activated and allows the user to cancel the operation if so desired. A typical schedule anomaly is a scheduled switch which is no longer present in the hardware configuration. The schedule is activated as described in section 12.1.1.

Action	Response
1. Follow the procedure documented in section 12.1.1 for activating a schedule through step 2.	The Schedule Activation Confirmation Workbox is on the screen.
2. Mouse button click on the Done Button.	The Done Button will highlight, the Schedule Activationx Workbox will disappear, and the selected schedule will be activated.

Q_____

12.1.3 Halt Time Workbox

This workbox is only accessible in **NORMAL** mode of operation and then only when there is an active schedule. In order to put the *SSM/PMAD* system in **NORMAL** mode of operation follow the procedure in section 7.1.2.

Action	Response
1. Follow the procedure documented in section 7.2.3 for halting a schedule through step 1.	Halt the Active Schedule Workbox is on the screen.
2. Mouse button click on the ASAP Button.	The ASAP Button will highlight, the Halt the Active Schedule Workbox will disappear. The Active Schedule will be halted.

Q_____

12.1.4 Source Power Change Workbox

This workbox is only accessible in **NORMAL** mode of operation. In order to put the *SSM/PMAD* system in **NORMAL** mode of operation follow the procedure in section 7.1.2.

Action	Response
1. Follow the procedure documented in section 7.3.5 for changing source power through step 1.	Source Power Change Workbox is on the screen.
2. Edit the data fields as needed for the source power change.	The changes will be shown in the data fields as they are edited
3. Mouse button click on <i>Done</i> in the workbox.	The <i>Done</i> button will highlight, the workbox will disappear, loads will be shed, if needed, and the scheduler will schedule around the new power constraint.

Q_____

12.1.5 Error Information Workbox

This workbox will display error information which must be acknowledged. The workbox is used in many places. The easiest test of this workbox is to follow the procedure in section 9.3.1.

12.2 FELES Workboxes

The FELES application uses only one workbox, the Change Window of Time Workbox. By changing the window of time on the FELES application screen, the gantt chart segments of the graphtool code may be exercised.

12.2.1 Change Window of Time

Action	Response
1. Follow the procedure documented in section 9.4 for changing window of time through step 1.	Change Window of Time Workbox is on the screen.
2. Edit the data fields as needed for the new window of time.	The changes will be shown in the data fields as they are edited
3. Mouse button click on <i>Done</i> in the workbox.	The <i>Done</i> button will highlight, the workbox will disappear, the gantt chart will be redrawn for the new time window.

Q_____

12.3 Power Utilization Workboxes

The Power Utilization application uses only one workbox, the Change Window of Time Workbox. By changing the window of time on the Power Utilization application screen, the plot segments of the graphtool code may be exercised.

12.3.1 Change Window of Time

Action	Response
1. Follow the procedure documented in section 11.2	Change Window of Time Workbox

- | | |
|---|--|
| <p>for changing window of time through step 1.</p> <p>2. Edit the data fields as needed for the new window of time.</p> <p>3. Mouse button click on <i>Done</i> in the workbox.</p> | <p>is on the screen.</p> <p>The changes will be shown in the data fields as they are edited</p> <p>The <i>Done</i> button will highlight, the workbox will disappear, the plots will be redrawn for the new time window.</p> |
|---|--|

Q_____

12.4 Activity Editor Workboxes

12.4.1 New Activity Workbox

- | Action | Response |
|--|--|
| <p>1. Follow the procedure documented in section 10.3 for creating activity editor items through step 1.</p> | <p>The Create Menu is on the screen.</p> |
| <p>2. Mouse button click on the <i>Activity</i> item in the Create Menu.</p> | <p>The <i>Activity</i> item will highlight, the Create Menu will disappear, and the New Activity Workbox will be displayed.</p> |
| <p>3. Edit the data fields as needed for the new activity.</p> | <p>The changes will be shown in the data fields as they are edited</p> |
| <p>4. Mouse button click on <i>Done</i> in the workbox.</p> | <p>The <i>Done</i> button will highlight, the workbox will disappear, and the new activity will be displayed to the Activity Editor main window.</p> |

Q_____

12.4.2 New Subtask Workbox

Action	Response
1. Follow the procedure documented in section 10.3 for creating activity editor items through step 1.	The Create Menu is on the screen.
2. Mouse button click on the <i>Subtask</i> item in the Create Menu.	The <i>Subtask</i> item will highlight, the Create Menu will disappear, and the New Subtask Workbox will be displayed.
3. Edit the data fields as needed for the new subtask.	The changes will be shown in the data fields as they are edited
4. Mouse button click on <i>Done</i> in the workbox.	The <i>Done</i> button will highlight, the workbox will disappear, and the new subtask will be displayed to the Activity Editor main window.

Q_____

12.4.3 New Requirement Workbox

Action	Response
1. Follow the procedure documented in section 10.3 for creating activity editor items through step 1.	The Create Menu is on the screen.
2. Mouse button click on the <i>Requirement</i> item in the Create Menu.	The <i>Requirement</i> item will highlight, the Create Menu will disappear, and the New Requirement Workbox will be displayed.

- | | |
|--|--|
| 3. Edit the data fields as needed for the new requirement. | The changes will be shown in the data fields as they are edited |
| 4. Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new requirement will be displayed to the Activity Editor main window. |

Q _____

12.4.4 New Powered Equipment Workbox

Action

1. Follow the procedure documented in section 10.3 for creating activity editor items through step 1.
2. Mouse button click on the *Powered Equipment* item in the Create Menu.
3. Edit the data fields as needed for the new powered equipment.
4. Mouse button click on *Done* in the workbox.

Response

The Create Menu is on the screen.

The *Powered Equipment* item will highlight, the Create Menu will disappear, and the New Powered Equipment Workbox will be displayed.

The changes will be shown in the data fields as they are edited

The *Done* button will highlight, the workbox will disappear, and the new powered equipment will be displayed to the Activity Editor main window.

Q _____

12.4.5 New Schedule Workbox

Action	Response
1. Follow the procedure documented in section 10.6 for creating activity editor items through step 1.	The Schedule Selected Activities menu is on the screen.
2. Mouse button click on the desired activities for the new schedule to be created.	Each selected item will stay highlighted when selected.
3. Mouse button click on the end item << <i>Schedule Selected Activities</i> >>	The end item will highlight, the Schedule Selected Activities menu will disappear, the databases will be saved and the New Schedule Workbox will appear.
4. Edit the data fields as needed for the new schedule.	The changes will be shown in the data fields as they are edited
5. Mouse button click on <i>Done</i> in the workbox.	The <i>Done</i> button will highlight, the workbox will disappear, and the new schedule will be created with the results of scheduling displayed to the Activity Editor scratchpad window.

Q_____

12.4.6 New Mode Workbox

In order to test this workbox, a piece of powered equipment must be edited.

Action	Response
1. Follow the procedure documented in section 10.2.4 for creating activity editor items through step 1.	The submenu containing the Powered Equipment is on the screen.

- | | | |
|----|---|---|
| 2. | Mouse button click on the desired piece of powered equipment. | The selected piece of powered equipment will highlight, the edit menu structure will disappear, and the selected piece of Powered Equipment will be displayed to the Activity Editor main window. |
| 3. | Mouse button click on the ****Undefined**** data field next to the label Mode: with the right mouse button. | The Equipment Mode Options menu will appear. |
| 4. | Mouse button click on <i>Create a New Equipment Mode</i> .
right mouse button. | The New Mode Workbox will appear. |
| 5. | Edit the data fields as needed for the new equipment mode. | The changes will be shown in the data fields as they are edited |
| 6. | Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new mode will be created and inserted into the Powered Equipment item being edited. |

Q_____

12.4.7 Availability Workbox

In order to test this workbox, a piece of powered equipment must be edited.

- | Action | Response |
|---|--|
| 1. Follow the procedure documented in section 10.2.4 for creating activity editor items through step 1. | The submenu containing the Powered Equipment is on the screen. |
| 2. Mouse button click on the desired piece of powered equipment. | The selected piece of powered equipment will highlight, |

- | | |
|--|---|
| | the edit menu structure will disappear, and the selected piece of Powered Equipment will be displayed to the Activity Editor main window. |
| 3. Mouse button click on the Start:, End: or Quantity: data field. | The Availability Information Workbox will appear. |
| 4. Edit the data fields as needed for the new availability information mode. | The changes will be shown in the data fields as they are edited |
| 5. Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new availability information data will be created and inserted into the Powered Equipment item being edited. |

Q_____

12.4.8 Default Workbox

In order to test this workbox, a piece of powered equipment must be edited.

- | Action | Response |
|---|---|
| 1. Follow the procedure documented in section 10.2.4 for creating activity editor items through step 1. | The submenu containing the Powered Equipment is on the screen. |
| 2. Mouse button click on the desired piece of powered equipment. | The selected piece of powered equipment will highlight, the edit menu structure will disappear, and the selected piece of Powered Equipment will be displayed to the Activity Editor main window. |

- | | | |
|----|---|--|
| 3. | Mouse button click on a Mode: data field for an existing mode with the middle mouse button. | The Mode will expand on the screen. |
| 4. | Mouse button click on the Augmented Power: data field with the right mouse button. | The Augmented Power menu appears. |
| 5. | Mouse button click on the <i>Default</i> item in the Augmented Power menu. | The <i>Default</i> item highlights, the Augmented Power menu disappears and an Augmented Power Selection Workbox appears. |
| 6. | Edit the data field as needed for the new Default Augmented Power in Watts. | The changes will be shown in the data field as it is edited. |
| 7. | Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new augmented power data will be inserted into the Powered Equipment item being edited. |

Q_____

12.4.9 Percent Workbox

In order to test this workbox, a piece of powered equipment must be edited.

- | Action | Response |
|---|---|
| 1. Follow the procedure documented in section 10.2.4 for creating activity editor items through step 1. | The submenu containing the Powered Equipment is on the screen. |
| 2. Mouse button click on the desired piece of powered equipment. | The selected piece of powered equipment will highlight, the edit menu structure will disappear, and the |

- | | |
|--|--|
| | selected piece of Powered Equipment will be displayed to the Activity Editor main window. |
| 3. Mouse button click on a Mode: data field for an existing mode with the middle mouse button. | The Mode will expand on the screen. |
| 4. Mouse button click on the Augmented Power: data field with the right mouse button. | The Augmented Power menu appears. |
| 5. Mouse button click on the <i>Percent</i> item in the Augmented Power menu. | The <i>Percent</i> item highlights, the Augmented Power menu disappears and an Augmented Power Selection Workbox appears. |
| 6. Edit the data field as needed for the new Augmented Power as a Percent. | The changes will be shown in the data field as it is edited. |
| 7. Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new augmented power data will be inserted into the Powered Equipment item being edited. |

Q_____

12.4.10 Power Workbox

In order to test this workbox, a piece of powered equipment must be edited.

- | Action | Response |
|---|--|
| 1. Follow the procedure documented in section 10.2.4 for creating activity editor items through step 1. | The submenu containing the Powered Equipment is on the screen. |

-
- | | |
|--|---|
| 2. Mouse button click on the desired piece of powered equipment. | The selected piece of powered equipment will highlight, the edit menu structure will disappear, and the selected piece of Powered Equipment will be displayed to the Activity Editor main window. |
| 3. Mouse button click on a Mode: data field for an existing mode with the middle mouse button. | The Mode will expand on the screen. |
| 4. Mouse button click on the Augmented Power: data field with the right mouse button. | The Augmented Power menu appears. |
| 5. Mouse button click on the <i>Watts</i> item in the Augmented Power menu. | The <i>Watts</i> item highlights, the Augmented Power menu disappears and an Augmented Power Selection Workbox appears. |
| 6. Edit the data field as needed for the new Augmented Power in Watts. | The changes will be shown in the data field as it is edited. |
| 7. Mouse button click on <i>Done</i> in the workbox. | The <i>Done</i> button will highlight, the workbox will disappear, and the new augmented power data will be inserted into the Powered Equipment item being edited. |

Q_____

13 User Interface: Menus

The *SSM/PMAD* system user interface contains two distinct types of pop-up menus, single selection menus and multiple choice menus. The single selection menus provide a convenient method for selecting a single item or action from a list of possibilities. The multiple choice menu allows the user to select more than one item for a given operation and the operations are shown as end items. The end items are bracketed by << and >> symbols. Both menu structures may contain an infinite number of sub-menus. When an item in a single selection menu or an end item in a multiple choice menu is selected, the menu item will highlight and dehighlight and the menu will then disappear.

13.1 General Menus

The menus described in this section are independent of the *SSM/PMAD* applications and may be accessed from any application.

13.1.1 System Options

Handled in Section 7.

13.1.2 Schedule Menu

This menu is both a separate menu and a submenu to the system options menu. To test this menu as a submenu proceed to section 7.2.1. To test this menu as a separate menu proceed to section 10.7 in the activity editor application.

13.2 Power System Menus

The menus in this section may only be tested from the Power System Application Screen.

13.2.1 Component Information Menu

This menu is only accessible in **NORMAL** mode of operation. To put the system in **NORMAL** mode of operation, follow the procedure documented in section 7.1.2.

Action	Response
1. Follow the procedure documented in section 8.10 in its entirety.	The Component Information will be visible.
2. Mouse button click outside the menu in order to wave it off.	The menu disappears

Q_____

13.3 FELES Menu

The menus in this section may only be tested from the FELES application screen.

13.3.1 Load Center Menu

This menu may be tested by following the procedure in section 9.6.

13.4 Power Utilization Menus

The menus in this section may only be tested from the Power Utilization application screen.

13.4.1 Load Center Menu

This menu may be tested by following the procedure in section 11.4.

13.4.2 Component Usage Menu

This menu may be tested by following the procedure in section 11.5.

13.5 Activity Editor Menus

The menus in this section may only be tested from the Activity Editor application screen.

13.5.1 Create Menu

This menu may be tested by following the procedure in section 10.3.

13.5.2 Edit/Delete Menu

This menu may be tested by following the procedure in section 10.2.1.

13.5.3 Select Activities Menu

This menu is a multiple choice menu which is used when creating a new schedule. to test this menu follow the procedure in section 10.6.

13.5.4 Augmented Power Menu

This menu is used when augmenting power with available excess power for an equipment mode.

Action	Response
1. Follow the procedure documented in section 12.4.8 through step 4.	The Power Augmentation Menu will be visible.
2. Mouse button click outside the menu in order to wave it off.	The menu disappears

Q_____

(This Page Intentionally Left Blank)

14 Intermediate Levels of Autonomy

14.1 Maintenance Mode

Purpose: To prove that maintenance mode activities are still working as expected with the new ILA software installed.

Action	Response
<p>1. Initialize the system by going to maintenance mode: Select "System Options" Select "Mode Changes" Select "Go to Maintenance"</p> <p>Q_____</p>	<p>The Mode Window will display "Maintenance".</p>
<p>2. Select 1K switch, Load Center 2 switch 17 Use "Manual On" to turn on load center 2 switch 17 from the Solbourne an event list to the LLP</p> <p>Q_____</p>	<p>Check mark appears. The Solbourne sends to command the switch. The LLP sends a switch and sensor configuration transaction back to the Solbourne to reflect the new state of the switches as a result of the command. The LLP updates its performance data on the switches and sends performance data to the Solbourne.</p> <p>1K switch is turned on (closed) and shows power.</p>
<p>3. Trip 1K switch, load center 2 switch 17.</p>	<p>The LLP detecting the fault sends a fault transaction to the Solbourne.</p>

The Solbourne sends a quiescent transaction to all the LLP's. This transaction is a request for the LLP's to send a quiescent is true transaction back to the Solbourne when their data is quiescent.

Each LLP sends a quiescent is true transaction to the Solbourne.

The Solbourne then sends a request for switch status information to each LLP.

Each LLP responds with the switch transactions.

LC-2-17 turns red with hard fault.

Hard fault window will display "LC-2-17 FT tripped".

Q_____

- | Action | Response |
|---|-----------------|
| 4. Remove 1K trip, load center 2 switch 17. | None. |

Q_____

- | | |
|---|------------------------------|
| 5. Use "Manual Off" to turn off 1K switch, LC-2-17. | Switch is available (green). |
|---|------------------------------|

14.2 ILA Test for Group Switches

Purpose: To exercise menu options related to grouping switches, testing switches under manual control, releasing manual control of switches, and returning them to service.

NOTE: THE FOLLOWING TEST IS DONE FROM THE POWER SYSTEM APPLICATION SCREEN.

- | Action | Response |
|---|--|
| 1. The system is connected to LLPs - PORT, LC-1, LC-2, LC-3, and STARBOARD. | State of PORT, LC-1, LC-2, LC-3, and STARBOARD is "C". |

Q_____

Go to Normal Mode:

Select "System Options"

Select "Mode Changes"

Select "Go to Normal Mode"

The Mode Window will display "NORMAL".

Q_____

2. Activate Schedule "Mike Test 4"
Select "System Options"
Select "Schedule Functions".
Select "Activate Schedule"
Select "Mike Test 4"

The Power Application screen will display power to the buses (beige) and the switches will display current being drawn when the schedule is activated. A workbox may appear for items which may not be scheduled due to missing switches. If this happens, click done on the workbox.

Q_____

3. Use the "Group Switches" application menu option to group:
Load Center 1 Switch-03
Load Center 2 Switch-03

The switches selected will be a reddish brown color.

Button click <Group Selected Items> at the top of the "Group Switches" menu.

"New Switch Group Information" Workbox appears in the scratchpad area.
Switches in the group will be:
LC-1-03
LC-2-03

Q_____

4. In the "New Switch Group Information" workbox, type the name of the new group, "Test Group 1", hit <RETURN>, and button click "Done".

Workbox will be cleared from the scratchpad area.

Q_____

To prove that the system knows about the group just created, button click "Ungroup Switches" menu option.

Pop Up Menu "Ungroup Switches"

will display "Test Group 1" as a group that could be ungrouped. Wave off the pop up window or select "Cancel".

- Q_____
- Use "Seize Manual Control" application menu option to seize control of:
Test Group 1
Load Center 2 Switch 04

Items selected will be reddish brown in color.

Q_____

After selecting the above switches, button click at the top of the menu <<<Seize Selected Items>>> "ILA Seizure Workbox" will appear in the scratchpad area, with the switches that have been seized.

Q_____

The contents of the "ILA Seizure Workbox" will be:

LC-2-03	Test Group 1	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
LC-1-03	Test Group 1	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
	Run Calculate for Effects			
LC-2-04	None	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
	Run Calculate for Effects			

Q_____

- In the "ILA Seizure Workbox", button click "Calculate".
Note: The switch information in parentheses is not displayed but is included in the test plan for informational purposes.

The new display will be: (Effects may vary with respect to time of execution)

LC-2-03 Test Group 1 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
 LC-1-03 Test Group 1 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

Effects

Activity 'Glovebox Housekeeping' (LC-1-3) Interrupted Due to Bumpable
 Seizure of LLP LC-1 Switch: 3.
 Activity 'Modular Combustion Facility 2' (LC-2-3) Interrupted Due to Bumpable
 Seizure of LLP LC-2 Switch: 3.
 Activity 'Space Station Furnace Facility 2' (LC-2-4) Interrupted Due to Power
 Capacity Shortfall on LLP PORT Switch: P03 when Seizing LLP LC-2 Switch: 3.

LC-2-04 None [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

Effects

Activity 'Space Station Furnace Facility 1' (LC-2-4) Interrupted Due to
 Bumpable Seizure of LLP LC-2 Switch: 4.

Q_____

Action	Response
7. Button Click "Done"	KANT says "Manual Seize" Manual icon (hand) appears on: LC-1-03 LC-2-03 LC-2-04

For all the above switches, the switch will open. No power is shown for any of the manually seized switches.

Q_____

Action	Response
8. Button click switches that have been manually seized: LC-1-03 LC-2-03 LC-2-04	Check marks will appear on

the seized switches:

LC-1-03

LC-2-03

LC-2-04

Q _____

Action

Response

- 9. Execute "Manual On" for each of the seized switches:
 LC-1-03
 LC-2-03
 LC-2-04

All 3 switches close. Power is shown for each of the 3 switches. Manual icon (hand) is still present for each of the 3 switches.

Q _____

Action

Response

- 10. Trip switch which is in a group that is manually seized, and trip switch which is not in a group but is manually seized:
 LC-1-03
 LC-2-04

Switch has hard fault (red).
Switch opens.
Switch is available (green).
Switch has hard fault (red).

After testing, LC-1-03 and LC-2-04 switches trip and are out of service (brown). The manual icon (hand) which previously indicated under ILA manual control now indicates the switch is under user control for testing purposes.

Q _____



The Hard Fault Diagnoses display window will say:
LC-1-3 FT tripped.
LC-2-4 FT tripped.

Opening
switches
LC-1-3
Reclosing:
LC-1-3

LC-1-3 FT tripped.

LC-1-3 FAST-TRIP tripped
During testing of the LLP,
LC-1-3 retripped on FAST-TRIP
Possible Causes:
Most Likely:
Low impedance short in the
cable below LC-1-3, switch
output LC-1-3, or switch input
of a lower switch.
Less Likely:
Current sensor in switch
reading high.

Same message is repeated for LC-2-4 as it is tested.

Q_____

Other responses:

Since LC-1-03 is a member of Test Group 1, the other switch in that group, LC-2-03, will be released from manual control and returned to the system. The switches will be open. The switches will show no power. The manual icons (hands) will disappear.

Q_____

ACTION

RESPONSE

11. Trip PORT P02

PORT P02 hard faults (red) with OC. Then it is out of service (brown). It is put under manual control (hand). "???" appears on the switch, indicating it did not retrip on OVER CURRENT but did draw more power than allowed.

Q_____

The Hard Fault Diagnoses will say:

LC-1-5 UV tripped.
PORT-P02 OC tripped.

Opening
switches:
PORT-P02
LC-1-3
LC-1-5
LC-1-6
Reclosing:
PORT-P02
Closing:
PORT-P02
Opening:
PORT-P02

PORT-P02 OVER-CURRENT tripped
During testing of the LLP,
PORT-P02 did not retrip on
OVER-CURRENT but did draw more
current than was allowed.

Possible Causes:

Most Likely:

Low impedance short in the
cable below PORT-P02 switch
output PORT-P02, or switch input

of a lower switch.
Less Likely:
Current sensor in switch
reading high.
Diagnoses Finished.

Q_____

Other responses:

All switches below PORT-P02 are made unavailable (off green).

Q_____

LC-1-03 was under manual control because of a trip (10 above). Since LC-1-03 is a lower level switch to PORT P02, it will no longer be under manual control. Manual icon (hand) will disappear. PORT-P02 will show ???. It was already open with no current being drawn. PORT-P02 tripped on an over-current and did not retrip, however it was pulling more current than was allowed.

Q_____

Switches LC-1-3, LC-1-5, and LC-1-6 below PORT P02 are "unavailable" (off green).

Q_____

12. Remove the trips on all switches which have been tripped: LC-1-03 LC-2-04 PORT P02 None.

Q_____

ACTION	RESPONSE
13. Deselect switches: LC-1-03 LC-2-03	

-
- LC-2-04
- Q_____
13. Select switch:
PORT P02
- Q_____
14. Use "Manual On" menu option to turn on:
PORT P02
- Q_____
15. (Note: Always allow at least 1 minute after
seizing manual control before releasing
manual control)
- Use "Release Manual Control" menu option to
select switches:
Load Center 2 Switch 04
PORT Switch 2
Button click <<<Release Selected ILA Items>>>
- Q_____
16. Use "Other System Operations" to "Return RPCs to Service"
Pop Up Menu appears showing all switches unavailable (off green)
or out-of-service (brown).
- Q_____
- Select the following:
- PORT P02

Check marks will disappear.

Check mark appears.

Switch PORT P02 closes.
Remains out of service (brown).
Shows 0.0 power. Switch
is determined to be OK.

Manual icon (hand)
disappears. PORT P02
will open. "???" will
return as the diagnosis.

LC-1-06
LC-1-05
LC-1-03
LC-2-04

Selected items turn reddish brown in color.

Q_____

Button click <<<Return Selected Switches to Service>>>

KANT says "Put switch back into service"

Hard Fault Diagnoses says:

-----Returning RPC's to Service-----

Attempting to return the following switches to service:

Port-P02
LC-2-4
LC-1-3
LC-1-5

LC-1-6

Testing
Switch:
PORT-P02
Reclosing
Port-P02

Put Switch: PORT-P02 back into service.

Same for LC-1-06
Same for LC-1-05
Same for LC-1-03
Same for LC-2-04

PORT P02 switch will change from out of service (brown) to available (green) and be on with power.

LC-2-04 will change from out of service (brown) to available (green) and be on with power.

LC-1-03, LC-1-05, and LC-1-06 will change from unavailable (off-green) to available (green).

LC-1-03 and LC-1-5 will be on with power.

LC-1-4 will be open.

Q_____

14.3 ILA Test for Redundant Switches

Purpose: Two critical loads will flip from their primary switches on the Port bus to the redundant switches on the Starboard bus.

ACTION

RESPONSE

1. Look at redundant icons on Load Center 3.

The redundant icons on the LC-3-4 and LC-3-6 indicate they are primary switches (background color of icon is tan), and the redundant icons on LC-3-18 and LC-3-20 indicate they are redundant switches (background color of icon is black).

Q_____

Apply a Fast Trip to the 3K PORT P04 switch

PORT P04 will hard fault (red) with a Fast Trip (FT).

All the 1K switches below PORT P04 (LC-3-03, LC-3-04, LC-3-06, LC-3-07) will hard fault (red) on Under Voltage (UV).

Frames will diagnose the 3K PORT P04 fault and make it out of service (brown).

Q_____

FRAMES system will make all the 1K switches below PORT P04 unavailable (off green). The 1K switches will open.

Q_____

Manual icon (hand) will appear on PORT P04 with FT.

Q_____

Diagnosis will say:

PORT-P04 FT tripped
LC-3-3 UV tripped
LC-3-4 switched to redundant
LC-3-4 UV tripped
LC-3-6 switched to redundant
LC-3-6 UV tripped
LC-3-7 UV tripped

Opening
switches:
PORT-P04
LC-3-3
LC-3-4
LC-3-6
LC-3-7
Reclosing:
PORT-P04

PORT-P04 FT tripped.

PORT-P04 FAST-TRIPPED
During testing of the LLP,
PORT-P04 retripped on FAST-TRIP.
Possible Causes:
Most Likely:
Low impedance short in the
cable below PORT-P04 switch,
output PORT-P04, or switch input
of a lower switch.
Less Likely:
Current sensor in switch
reading high.
Diagnoses Finished.

Q_____

Other responses:

LC-3-04 and LC-3-06 will switch to their redundant switches:
LC-3-18 and LC-3-20, respectively. LC-3-18 and LC-3-20 will
close, and show power is being drawn.

Q_____

Redundant icons will be removed since the redundant switches are not
available (off green).

Q_____

ACTION	RESPONSE
2. With the trip on PORT P04, select PORT P04. Use "Manual On" to turn on the switch.	Check mark will appear on PORT P04.

The 3K PORT P04 will trip again and the switch will go from out of service
(brown) to hard fault (red) for 4 seconds. FRAMES does some testing.
However, since PORT P04 is under manual control with a "no diagnosable" flag
set, it will not perform any diagnosis on PORT P04. The switch will return
to out of service (brown) and the diagnosis will be FT.

Q_____

ACTION	RESPONSE
3. Remove the trip on PORT P04.	None.

Q_____

ACTION	RESPONSE
4. Use "Manual On" to turn on the switch.	The switch is closed. The color is still out of service (brown). However, the diagnosis

will change from FT to 0.0 amps.
This indicates that the switch is fixed.

Q_____

ACTION

5. Use "Manual Off" to turn the switch off.

Deselect PORT P04.

Q_____

ACTION

6. Release the seized switch to the system by using the "Release Manual Control" menu option of the Power System Screen.

RESPONSE

The switch will open and the diagnosis is changed back to FT.

Check mark will go away.

RESPONSE

The "ILA Release" Menu pops up.

Q_____

ACTION

7. Select PORT Switch-04.

RESPONSE

PORT P04 turn reddish brown.

Q_____

ACTION

Button Click "Release Selected ILA Items"

RESPONSE

The hand icon will be removed from PORT P04, but the switch will remain out of service (brown), since it is still

considered out of service by the FRAMES system.

Q _____

ACTION

RESPONSE

- 8. Select "Return RPCs to Service" menu option under "Other System Operations" under the "System Options" Menu.

The "Put a Switch Back in Service" menu will contain all out of service (brown) switches as well as all the unavailable (off green) switches.

The switches include:

- LC-3-03
- LC-3-04
- LC-3-06
- LC-3-07
- PORT P04

Q _____

ACTION

RESPONSE

Select PORT P04

Menu item PORT P04 will turn reddish brown.

Button Click

<<Return Selected Switches to Service>>

KANT says:

"put switch back into service".

Frames tests the switch. If it does not fail, the switch will change from out of service (brown) to available (green). PORT P04 will

open with no power.
MAESTRO is notified that the switch
is available.

Q _____

ACTION

9. Use the "Group Switches" menu option to group all the 1K switches which are unavailable because of the failure of PORT P04.

Select the following switches from the pop up menu:

Load Center 3 Switch-03

Load Center 3 Switch-04

Load Center 3 Switch-06

Load Center 3 Switch-07

After selecting the above switches, button click
<Group Selected Items> at the top of the menu.

RESPONSE

The switches selected will be
a reddish brown color.

"New Switch Group Information"
Workbox will appear in the
scratchpad area. Switches in
the group will be:
LC-3-03
LC-3-04
LC-3-06
LC-3-07

Q _____

ACTION

10. In the "New Switch Group Information" workbox,
type the name of the new group, "Test Group 2",

RESPONSE

hit <RETURN>, and button click "Done".

Workbox will be cleared from the scratchpad area.

Q_____

To prove that the system knows about the group just created, button click "Ungroup Switches" menu option. Pop Up Menu "Ungroup Switches" will display "Test Group 2" as a group that could be ungrouped. Wave off the pop up window or select "Cancel".

Q_____

ACTION

RESPONSE

11. Use "Seize Manual Control" application menu option to seize control of: Test Group 2

Test Group 2 Item selected will be reddish brown in color.

Q_____

After selecting the above group, button click <<<Seize Selected Items>>> "ILA Seizure Workbox" will appear in scratchpad area, with the switches that have been seized.

Q_____

The contents of the "ILA Seizure Workbox: will be:

LC-3-07	Test Group 2	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
LC-3-06	Test Group 2	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
LC-3-04	Test Group 2	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
LC-3-03	Test Group 2	[00.00:xx - 00.02:xx]	[3-50]	1000W -TBD OFF
Run Calculate for Effects				

Q_____

ACTION

RESPONSE

12. Edit the time parameter for 1 hour and button click "Done"

The time parameter will be changed to 1 hour for all 4 switches.

Q_____

Edit the power for each to the following.
Button click "Done" after each change.

LC-3-03 600W 1000W will become 600W
LC-3-04 850W 1000W will become 850W
LC-3-06 450W 1000W will become 450W

Q_____

ACTION

RESPONSE

13. In the "ILA Seizure" workbox, button click "Calculate".

The new display will be:

LC-3-07 Test Group 1 [00.00:xx - 00.01:xx] [3-50] 1000W -TBD OFF
LC-3-06 Test Group 1 [00.00:xx - 00.01:xx] [3-50] 450W -TBD OFF
LC-3-04 Test Group 1 [00.00:xx - 00.01:xx] [3-50] 850W -TBD OFF
LC-3-03 Test Group 1 [00.00:xx - 00.01:xx] [3-50] 600W -TBD OFF
 Effects: None

Q_____

ACTION	RESPONSE
14. Button Click "Done"	KANT says "Manual Seize"
	Manual icon (hand) appears on: LC-3-03 LC-3-04 LC-3-06 LC-3-07 at the start time.

For all the above switches, the switch will open. No power is shown on any of the manually seized switches.

Q_____

ACTION	RESPONSE
15. Button click switches that have been manually seized: LC-3-03 LC-3-04 LC-3-06 LC-3-07	Check marks will appear on seized switches: LC-3-03 LC-3-04 LC-3-06 LC-3-07

Q_____

ACTION	RESPONSE
16. Execute "Manual On" for each of the seized switches: LC-3-03 LC-3-04 LC-3-06 LC-3-07	All 4 switches close. Power

is shown for each of the 4 switches.
Manual icon (hand) is still present
for each of the 4 switches.

Q _____

ACTION

RESPONSE

17. Execute "Manual Off" for each of the seized switches:
LC-3-03
LC-3-04
LC-3-06
LC-3-07

All 4 switches open. Power is
not shown for any of the 4 switches.

Q _____

ACTION

RESPONSE

18. Use "Release Manual Control" menu option
to select switches: Test Group 2

Selected item turns reddish brown.

Button click <<<Release Selected ILA Items>>>

Manual icons (hand) disappear.

Q _____

ACTION

RESPONSE

19. Use "Other System Operations" to "Return RPCs to Service"

Pop Up Menu appears showing
all switches unavailable
(off green) or out-of-service
(brown).

Q _____

Select the following:

LC-3-07
LC-3-06
LC-3-04
LC-3-03

All selected items turn reddish brown.

Button click
<<<Return Selected Switches to Service>>>

KANT says:
"Put switch back into service".

-----Returning RPCs to Service-----

Attempting to return the
following switches to
service:

LC-3-3
LC-3-4
LC-3-6
LC-3-7

Testing
Switch:
LC-3-3
Reclosing
LC-3-3

Put Switch: LC-3-3 back into service.

Same for LC-3-4
Same for LC-3-6
Same for LC-3-7

All the switches change from unavailable (off-green) to available (green).
The redundant icons are displayed. LC-3-6 becomes the redundant switch for
LC-3-20. LC-3-4 becomes the redundant switch for LC-3-18.

Q_____

14.4 ILA Test to Flip/Flop Redundants

Purpose: Two critical loads will flip from their primary switches on Starboard to the redundant switches on the Port bus.

ACTION

RESPONSE

1. Look at redundant icons on Load Center 3.

The redundant icons on LC-3-18 and LC-3-20 indicate they are primary switches (background color of icon is tan), and the redundant icons on LC-3-4 and LC-3-6 indicate they are redundant switches (background color of icon is black).

Q_____

Apply a Fast Trip to the 3K STARBOARD S04 switch

STARBOARD S04 will hard fault (red) with a Fast Trip (FT).

All the 1K switches below STARBOARD S04 (LC-3-16, LC-3-18, LC-3-19, LC-3-20) will hard fault (red) with Under Voltage (UV).

FRAMES will diagnose the 3K STARBOARD S04 fault and make it out of service (brown).

Q_____

FRAMES system will make all the 1K switches below STARBOARD S04 unavailable (off green). The 1K switches will open.

Q_____

Manual icon (hand) will appear on STARBOARD S04 with FT.

Q_____

Diagnosis will say:

- STARBOARD-S04 FT tripped
- LC-3-16 UV tripped
- LC-3-17 UV tripped
- LC-3-19 UV tripped
- LC-3-20 UV tripped

Opening
switches:
STARBOARD-S04
LC-3-16
LC-3-17
LC-3-19
LC-3-20
Reclosing:
STARBOARD-S04

STARBOARD-S04 FT tripped.

STARBOARD-S04 FAST-TRIPPED
During testing of the LLP,
STARBOARD-S04 retripped on FAST-TRIP.

Possible Causes:

Most Likely:

Low impedance short in the
cable below STARBOARD-S04 switch,
output STARBOARD-S04, or switch input
of a lower switch.

Less Likely:

Current sensor in switch
reading high.

Q_____

Other responses:

LC-3-18 and LC-3-20 will switch to their redundant switches: LC-3-04 and LC-3-06, respectively. LC-3-04 and LC-3-06 will close, and show power is being drawn.

Q_____

Redundant icons will be removed since the redundant switches are not available (off green).

Q_____

- | | | |
|----|-----------------------------------|-------|
| 3. | Remove the trip on STARBOARD S04. | None. |
|----|-----------------------------------|-------|

Q_____

Action	Response
---------------	-----------------

- | | | |
|----|---|--|
| 4. | Release the seized switch to the system by using the "Release Manual Control" menu option of the Power System Screen. | |
|----|---|--|

The "ILA Release" Menu pops up.

Q_____

Action	Response
---------------	-----------------

- | | | |
|----|--------------------------------|--|
| 5. | Select:
STARBOARD Switch-04 | |
|----|--------------------------------|--|

Selected switch turns reddish brown.

Q_____

Action	Response
---------------	-----------------

Button Click "Release Selected ILA Items"

The hand icons will be removed

from the selected items, but the switches will remain out of service (brown) or unavailable (off-green), since they are still considered out of service or unavailable by the FRAMES system.

Q_____

Action

Response

- 6. Select "Return RPCs to Service" menu option under "Other System Operations" under the "System Options" Menu.

The "Return RPCs to Service" menu will contain all out of service (brown) switches as well as all the unavailable (off green) switches. The switches include:
LC-3-16
LC-3-18
LC-3-19
LC-3-20
STARBOARD S04

Q_____

Action

Response

Select:
STARBOARD S04
LC-3-16
LC-3-18
LC-3-19
LC-3-20

Selected menu items will turn reddish brown.

Button Click

<<Return Selected Switches to Service>>

KANT says:
"put switch back into service".

FRAMES tests the switches. If they do not fail, the STARBOARD S04 switch will change from out of service (brown) to available (green). The switches below will change from unavailable (off-green) to available (green). Switches LC-3-16, LC-3-18, LC-3-19, LC-3-20 will be open with no power. Starboard S04 will be closed with 0.0 power MAESTRO is notified that the switches are available.

Q _____

14.5 ILA Test for Group Manual Seizure

Purpose: Try to seize a group of switches, but it is unsuccessful because a power constraint is detected. Exercise the Add/Drop option of grouping switches.

Action	Response
1. In order to get the following "effects" in the ILA Seizure Workbox, the schedule Mike's 4 must be activated and the following test run immediately. If other ILA activities have been run, to get the following "effects", halt the schedule, go to idle, go to normal, and activate the schedule again.	
Use the "Group Switches" application menu option to group: Load Center 1 Switch-03 Load Center 2 Switch-03	The switches selected will be a reddish brown color.
Button click <Group Selected Items> at the top of the "Group Switches" menu.	"New Switch Group Information" Workbox appears in the scratch-pad area. Switches in the group will be:

LC-1-03
LC-2-03

Q_____

Action

Response

- 2. In the "New Switch Group Information" workbox, type the name of the new group, "Test Group 3", hit <RETURN>, and button click "Done".

Workbox will be cleared from the scratchpad area.

Q_____

Action

Response

- 3. Use "Seize Manual Control" application menu option to seize control of:
Test Group 3
Load Center 1 Switch 05

Items selected will be reddish brown in color.

Q_____

After selecting the above switches, button click at the top of the menu <<<Seize Selected Items>>>

"ILA Seizure Workbox" will appear in the scratchpad area, with the switches that have been seized.

Q_____

The contents of the "ILA Seizure Workbox" will be:

LC-1-05 None [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
Run Calculate for Effects

Button click on LC-2-05 to select (add) it.

is displayed.

Button click on "Seize Selected Items"

Items selected are reddish brown in color.

"ILA Seizure Workbox" is re-displayed without LC-1-04. It now includes LC-2-05.

Q_____

Action

Response

- 6. To calculate the effects of adding LC-2-05, in the "ILA Seizure Workbox" button click "Calculate".

"ILA Seizure Workbox" is re-displayed with the effects of adding LC-2-05.

The new display will be:

LC-1-05 None [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

Effects:

Activity 'Exobiology Data Acquisition' (LC-1-5) Interrupted Due to Bumpable Seizure of LLP LC-1 Switch: 5.

LC-2-03 Test Group 3 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

LC-1-03 Test Group 3 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

Effects:

Activity 'Glovebox Housekeeping' (LC-1-3) Interrupted Due to Bumpable Seizure of LLP LC-1 Switch: 3.

Activity 'Modular Combustion Facility 2' (LC-2-3) Interrupted Due to Bumpable Seizure of LLP LC-2 Switch: 3.

Activity 'Space Station Furnace Facility 1' Interrupted Due to Power Capacity Shortfall on LLP PORT Switch: P03 when Seizing LLP LC-2 Switch: 3.

LC-2-05 None [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF

Effects:

Activity 'Space Station Furnace Facility 2' (LC-2-5) Interrupted Due to Bumpable Seizure of LLP LC-2 Switch: 5.

Q _____

Action	Response
7. Button Click "Done"	KANT says "Manual Seize" Manual icon (hand) appears on: LC-1-03 LC-2-03 LC-1-05 LC-2-05

All the above switches will open. No power is shown for any of the manually seized switches.

Q _____

Action	Response
8. Use "Release Manual Control" menu option to return all manually controlled switches to service:	"ILA Release Menu" is displayed.
Select: Test Group 3 LC-1-05 LC-2-05	Selected items are reddish brown in color.
Button click <<<Release Selected ILA Items>>>	Kant says "Manual Release". Manual icons (hand) disappear.

Q _____

(This Page Intentionally Left Blank)

15 Source Power Change Test

Purpose: To shed switches in accordance with priorities when power is reduced, and to reschedule the activities which are cancelled.

NOTE: THE FOLLOWING TEST IS DONE FROM THE POWER SYSTEM APPLICATION SCREEN.

Action	Response
<p>1. Bring up the system. The system is connected to LLPs - PORT, LC-1, LC-2, LC-3, and STARBOARD.</p> <p>Q_____</p> <p>Go to Normal Mode: Select "System Options" Select "Mode Changes" Select "Go to Normal Mode"</p>	<p>State of PORT, LC-1, LC-2, LC-3, and STARBOARD is "C".</p> <p>The Mode Window will display "NORMAL".</p>
<p>Q_____</p> <p>Action</p> <p>2. Activate Schedule "Mike Test 4" Select "System Options" Select "Schedule Functions". Select "Activate Schedule" Select "Mike Test 4"</p>	<p>Response</p> <p>The Power Application screen will display power to the buses (beige) and the switches will display current being drawn when the schedule is activated.</p>
<p>Q_____</p> <p>Action</p>	<p>Response</p>

3. Use the "Group Switches" application menu option to group:
Load Center 1 Switch-03
Load Center 3 Switch-19

Button click ;Group Selected Items; at the top of the "Group Switches" menu.

The switches selected will be a reddish brown color.

"New Switch Group Information" Workbox appears in the scratchpad area.

Switches in the group will be:

LC-1-02

LC-1-03

LC-3-19

Q_____

Action

Response

4. In the "New Switch Group Information" workbox, type the name of the new group, "Test Group 4", hit <RETURN>, and button click "Done".

Workbox will be cleared from the scratchpad area.

Q_____

Action

Response

5. Likewise, use the "Group Switches" application menu option to group:
Load Center 2 Switch-03

Button click <Group Selected Items> at the top of the "Group Switches" menu.

The switch selected will be a reddish brown color.

"New Switch Group Information" Workbox appears in the scratchpad area. Switch in the group will be:

LC-2-03

Q_____

Action

Response

- 6. In the "New Switch Group Information" workbox, type the name of the new group, "Test Group 5", hit <RETURN>, and button click "Done".

Workbox will be cleared from the scratchpad area.

Q_____

Action

Response

- 7. After the schedule is up and running and power is shown on the switches, use "Seize Manual Control" application menu option to seize control of:
Test Group 4
Test Group 5

Items selected will be reddish brown in color.

Q_____

After selecting the above switches, button click at the top of the menu <<<Seize Selected Items>>>

"ILA Seizure Workbox" will appear in the scratchpad area, with the switches that have been seized.

Q_____

The contents of the "ILA Seizure Workbox" will be:

LC-2-03 Test Group 5 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
Run Calculate for Effects

LC-3-19 Test Group 4 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
 LC-1-03 Test Group 4 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
 Run Calculate for Effects

Q_____

Action

Response

8. In the "ILA Seizure Workbox",
 button click "Calculate".

Note: The switch information in parentheses is not displayed but is included
 in the test plan for informational purposes.

The new display will be:
 (Effects may vary with respect to time of execution)

LC-2-03 Test Group 5 [00.00:xx - 00.02:xx] [0-50] 1000W -TBD OFF
 Effects
 Activity 'Modular Combustion Facility 2' (LC-2-3) Interrupted Due
 to Bumpable Seizure of LLP LC-2 Switch: 3.

Activity 'Space Station Furnace Facility 1' (LC-2-4) Interrupted Due
 to Power Capacity Shortfall on LLP PORT Switch: P04 when Seizing
 LLP LC-2 Switch: 3.

LC-3-19 Test Group 4 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
 LC-1-03 Test Group 4 [00.00:xx - 00.02:xx] [3-50] 1000W -TBD OFF
 Effects
 Activity 'Glovebox Housekeeping' (LC-1-3) Interrupted Due to Bumpable
 Seizure of LLP LC-1 Switch: 3.
 Activity 'Urine/Waste Heater' (LC-3-19) Interrupted Due to Bumpable
 Seizure of LLP LC-3 Switch: 19.

Q_____	Action	Response
	9. Button Click "Done"	KANT says "Manual Seize" Manual icon (hand) appears on: LC-1-03 LC-3-19 LC-2-03
	For all the above switches, the switch will open. No power is shown for any of the manually seized switches.	
	10. Button Click on the FELES screen selection	FELES screen appears with a window of time of 2 hours. A Manual activity for D19, B03 exists. A Manual activity for C03 exists.
	Q_____	
	Return to the Power System screen.	Power screen appears.
	Q_____	
	11. Now execute the source power change. Select "System Options" Select "Other System Operations" Select "Source Power Change"	"Source Power Change" workbox is displayed.
	Q_____	

12. Edit the following parameters:
 Change the start time to Mission Time + 2 minutes.
 Power Allotment Port Bus 2600 W
 Power Allotment Starboard Bus 0 W

Button click "Done"

Workbox will disappear.
 Activities will be shed,
 and switches will be opened.

Group with LC-1-2, LC-1-3
 and LC-3-19 will be shed.
 LC-2-3 will also be shed.

Before Source Power Change: (M= Manual, R= Redundant)
 After Source Power Change: (S = Shed)

B		03	05	06		02	03	04	05	
		M	ON	--		S	S	S	S	
	16	17	18	19		16	17	18	19	
	ON	ON	ON	ON		S	S	S	S	
C		03	04	05	06		03	04	05	06
		M	ON	ON	ON		S	S	S	S
	17	18	19	20		17	18	19	20	
	ON	ON	ON	ON		S	S	S	S	
D		03	04	06	07		03	04	06	07
		ON	ON	ON	ON		S	ON	ON	ON
	16	18	19	20		16	18	19	20	
	ON	R	ON	R		S	R	S	R	

Note: S04 stays "on" even though no power is being drawn because of the redundant possibility on that bus.

13. Button click "Power Changes"

The following will be displayed in the Power Changes window:

```
Source Power Change Requested:
--      Starting:          00.00:xx
--      Ending:           00.02:xx
--      Port Bus New Maximum Power:      2600 Watts
--      Starboard bus New Maximum Power:  0 Watts
source power change to occur in 0 minutes
Processing source power change.
Maximum Power on Port Bus Changing from POWER-SOURCE-BUS-A to 2600
Maximum Power on Starboard Bus Changing from POWER-SOURCE-BUS-B to 0
Shedding Switches:
--      List of all switches shed
```

Q_____

Action	Response
14. Button Click on the FELES screen selection	FELES screen appears with a window of time of 2 hours.
Button Click on "Change Window of Time"	"Change Window of Time" workbox appears.
Change the window of time from 2 hours to 9 hours. Button click "Done".	The screen will display 9 hours instead of 2 hours.

The schedule will show activities shed for the source power change.
The activities will be rescheduled after 2 hours.

Q_____

Button Click on Power System

The Power Screen appears.

Q_____

15. Using the results from the Source Power Change Test, test redundant capabilities:

Trip PORT P04 switch

Hand appears on PORT P04.
LC-3-03, LC-3-04, LC-3-06,
LC-3-07 turn khaki green
in color.

Loads with redundant capability LC-3-04,
and LC-3-06 switch to their redundants,
LC-3-18 and LC-3-20 respectively.

Q_____

Remove the trip on PORT P04.

Q_____

Put RPCc back into service

16 Soft Fault Test

Purpose: To show switches which have exceeded their initial power allocation, but not their augment power allocation.

NOTE: THE FOLLOWING TEST IS DONE FROM THE POWER SYSTEM APPLICATION SCREEN.

Action	Response
<p>1. Bring up the system. The system is connected to LLPs - PORT, LC-1, LC-2, LC-3, and STARBOARD.</p> <p>Q_____</p> <p>Go to Normal Mode: Select "System Options" Select "Mode Changes" Select "Go to Normal Mode"</p>	<p>State of PORT, LC-1, LC-2, LC-3, and STARBOARD is "C".</p> <p>The Mode Window will display "NORMAL".</p>
<p>Q_____</p> <p>Action</p> <p>2. Activate Schedule "Soft Fault Current Limiting" Select "System Options" Select "Schedule Functions". Select "Activate Schedule" Select "Soft Fault Test"</p>	<p>Response</p> <p>The Power Application screen will display power to the buses (beige) and the switches will display current being drawn when the schedule is activated.</p>
<p>Q_____</p> <p>Action</p>	<p>Response</p>

3. Wait for schedule to activate.

When the load center 1 switches close, the power system representation of the switch should turn gold colored indicating an out of current limit soft fault. Current limiting soft fault is also reported to the soft fault diagnosis window.

Q_____

Action

Response

4. Go to Idle Mode.
Select "System Options"
Select "Mode Changes"
Select "Go to Idle"

System goes to idle mode of operation

Q_____

17 Incipient Fault Test

Purpose: To show a piece of powered equipment has been scheduled to for a for a period of time which would cause it to exceed its mean time between failure.

NOTE: THE FOLLOWING TEST IS DONE FROM THE POWER SYSTEM APPLICATION SCREEN.

Action	Response
<p>1. Bring up the system. The system is connected to LLPs - PORT, LC-1, LC-2, LC-3, and STARBOARD.</p> <p>Q_____</p> <p>Go to Normal Mode: Select "System Options" Select "Mode Changes" Select "Go to Normal Mode"</p> <p>Q_____</p>	<p>State of PORT, LC-1, LC-2, LC-3, and STARBOARD is "C".</p> <p>The Mode Window will display "NORMAL".</p>
<p>2. Activate Schedule "Mikes Test 4" Select "System Options" Select "Schedule Functions". Select "Activate Schedule" Select "Soft Fault Test"</p> <p>Q_____</p>	<p>The Power Application screen will display power to the buses (beige) and the switches will display current being drawn when the schedule is activated.</p>
<p>Action</p>	<p>Response</p>

3. Wait for schedule to activate.

When the load center 1 switches close, the powered equipment for the glovebox housekeeping experiment, the space physiology experiment 1, and the modular combustion facility 2 will exceed their mean time between failures during this run and this information will be reported to the incipient fault detection window.

Q _____

Action

Response

4. Go to Idle Mode.
Select "System Options"
Select "Mode Changes"
Select "Go to Idle"

System goes to idle mode of operation.

Q _____

18 Hypothetical Scheduler Test

Purpose: To add an activity to the currently active schedule and delete an activity-performance from the currently active schedule. This will be accomplished by performing these operations on the hypothetical schedule and then Implementing the hypothetical schedule.

NOTE: THE FOLLOWING TEST IS DONE FROM THE POWER SYSTEM APPLICATION SCREEN.

Action	Response
<p>1. Bring up the system. The system is connected to LLPs - PORT, LC-1, LC-2, LC-3, and STARBOARD.</p> <p>Q_____</p> <p>Go to Normal Mode: Select "System Options" Select "Mode Changes" Select "Go to Normal Mode"</p> <p>Q_____</p>	<p>State of PORT, LC-1, LC-2, LC-3, and STARBOARD is "C".</p> <p>The Mode Window will display "NORMAL".</p>
<p>2. Activate Schedule "Mikes Test 4" Select "System Options" Select "Schedule Functions". Select "Activate Schedule" Select "Soft Fault Test"</p> <p>Q_____</p>	<p>The Power Application screen will display power to the buses (beige) and the switches will display current being drawn when the schedule is activated.</p>

Action	Response
3. Select "Hypothetical Scheduler".	The hypothetical scheduler interface application screen will appear. The user will then see a workbox which requests when the user will be done making hypothetical changes.
Q_____	
Action	Response
4. Take the mission time and add 10 minutes and enter this as Hypothetical completion time. Then click "Done".	The time finished? workbox will disappear.
Q_____	
Action	Response
5. Select "Add Activity"	The activity menu will appear.
Q_____	
Action	Response
6. Click "Biotechnology Facility 2"	The activity menu will disappear. The add activity workbox will appear.
Q_____	
Action	Response
7. Click "Done"	

The add activity workbox will disappear. The hypothetical scheduler will then add the activity. This may be seen by changing the window of time to a 12 hour width.

Q _____

Action

8. Click on the gantt bar for "Biotechnology Facility 1"

Response

The gantt bar highlights.

Q _____

Action

9. Click "Delete Performance"

Response

The hypothetical scheduler will delete the highlighted gantt bar from the hypothetical schedule and redisplay the schedule. (In this case the entire activity is removed from the schedule)

Q _____

Action

10. Click "Implement"

Response

The hypothetical schedule will now become the active schedule and the user interface will bring up FELES user interface screen.

Q _____

Action

11. Go to Idle Mode.

Response

Select "System Options"
Select "Mode Changes"
Select "Go to Idle"

System goes to idle mode of operation.

Q_____

19 Return to Idle Mode

Action	Response
1. Halt the active schedule. Select "System Options" Select "Schedule Functions" Select "Halt the Active Schedule"	FELES screen will show all activities halted.
2. Go to Idle Mode Select "System Options" Select "Mode Changes" Select "Go to Idle"	The breadboard will be shut down and the LLP's will be returned to ground zero. The Mode Window will display "IDLE".

Q _____

(This Page Intentionally Left Blank)

20 Shutdown

20.0.1 Power System

Action	Response
1. Turn the LLP's off.	LLP's are turned off.
2. Turn the main power off.	Main power is turned off.
3. Turn the Housekeeping power off to the rack.	Housekeeping power is turned off.

Q_____

20.0.2 Solbourne

To quit the Solbourne, perform the following steps:

Action	Response
1. Stop the system on the SSM/PMAD interface.	The system is stopped.
2. Exit the system.	UNIX prompt will appear.
3. Type "logout".	User is logged off the system.

(This Page Intentionally Left Blank)

Appendix G

SSM/PMAD User Manual

PRECEDING PAGE BLANK NOT FILMED

(This Page Intentionally Left Blank)

G SSM/PMAD User Manual

SSM/PMAD User Manual

Compiled by
Barry Ashworth
Michael Elges
Chris Myers
Janet Weinz

(This Page Intentionally Left Blank)

Contents

1	Purpose of User Manual	508
1.1	Acronyms and Definitions	508
2	Installation of SSM/PMAD from the Solbourne	512
2.1	Loading the Software	512
2.2	Changing the X Server	513
2.3	Copying the Initialization Files	513
2.4	Creating an LLP Software Diskette	513
3	Overview of the SSM/PMAD System	516
3.1	Purpose of the SSM/PMAD System	516
3.2	Hardware Configuration	517
3.3	Distributed Control and Process Management	517
3.4	Contingencies	519
3.5	Fault Diagnoses	519
3.5.1	Types of Faults Diagnosed	522
4	The User Interface	524
4.1	Bringing up the system	524
4.1.1	The Power System	524
4.2	Logging onto the System	525
4.2.1	Running SSM/PMAD from a Remote Location	526
4.3	Screen Layout and Interface Components	527
4.3.1	Applications Selection Window	531
4.3.2	Clock Bar Window	531
4.3.3	Mode Indicator Window	531
4.3.4	System Flow Window	532
4.3.5	The Four Fault Diagnoses and Detection Windows	532
4.3.6	Application Window, Menu and Scratchpad	535
4.3.7	Status Line Window	535
4.4	System Options	536
4.4.1	Mode Changes	536
4.4.2	Schedule Functions	542
4.4.3	Other System Operations	544
4.4.4	Utilities	549
4.4.5	Help	550
4.4.6	Exit SSM/PMAD	550
4.4.7	Cancel	550

4.5	The Power System Application Menu	550
4.5.1	Power System Topology	550
4.5.2	Power System Representation	552
4.5.3	Power System Application Options	553
4.6	The Power Utilization Application Menu	555
4.6.1	Change Window of Time	555
4.6.2	System Power Usage	555
4.6.3	Load Center Power Usage	555
4.6.4	Component Power Usage	557
4.6.5	System Power Availability	557
4.6.6	Load Center Power Availability	557
4.7	The FELES Application Menu	559
4.7.1	Change Window of Time	559
4.7.2	View Schedule by Activity	559
4.7.3	View Schedule by Load Center	561
4.7.4	Activity Info	561
4.7.5	Load Info	561
4.8	The Activity Editor	564
4.8.1	Background and Purpose	564
4.8.2	Definition of Activity	565
4.8.3	What MAESTRO Needs	566
4.8.4	What are the <i>SSM/PMAD</i> Requirements	567
4.8.5	Window Layout	568
4.8.6	Viewing and Editing the Data at Different Levels	571
4.8.7	Layout of an Activity, Subtask, Requirement and Powered Equipment	574
4.8.8	Scheduling	583
4.8.9	Deleting the Data	586
4.9	ILA Interface	591
4.9.1	ILA System Overview	591
4.9.2	Groups	592
4.9.3	Manual Seizure	592
4.9.4	Data Collected for Seizing Switches	594
4.9.5	ILA Seizure Workbox	595
4.9.6	Editing Switch Seizure Data	600
4.9.7	The Add/Drop Option	600
4.9.8	The Calculate Option	601
4.9.9	Manual Release	601
4.10	The Hypothetical Scheduler Application	604
4.10.1	Window Layout	605
4.10.2	Adding an Activity to the Hypothetical Schedule	606

4.10.3 Deleting an Activity Performance from the Hypothetical Schedule . .	606
4.10.4 Implementing the Hypothetical Schedule	606

List of Figures

1	SSM/PMAD Hardware Configuration	518
2	Distributed Control and Process Management	520
3	Architecture Hierarchy	521
4	Screen Layout	528
5	Sample Workbox	529
6	Single Selection Menu with a Pull Right	530
7	Multiple Choice Menu with a Pull Right	530
8	Screen Selections	531
9	Clock Bar Window	531
10	Mode Indicator	532
11	System-Flow	533
12	User Interface Fault Windows and Source Power Change Window	534
13	Go to Normal	537
14	Go to Maintenance	540
15	Go to Idle	542
16	Activate a Schedule	543
17	Schedule Activation Time Workbox	544
18	Schedule Contingencies Workbox	545
19	Active Schedule Halt Time Workbox	545
20	Source Power Change Message Window	548
21	Source Power Change Workbox	549
22	Power System Screen	551
23	Power System Detailed Data	553
24	Power System Load Information	554
25	Power Utilization Screen	556
26	Power Utilization Change Window of Time	557
27	Power Utilization Component Power Usage	558
28	FELES Screen	560
29	FELES Change Window of Time	561
30	FELES View Schedule by Load Center	562
31	FELES Activity Info	563
32	FELES Load Info	564
33	Activity Editor Window	569
34	Activity Editor Status Line Category I	570
35	Activity Editor Status Line Category II	570
36	Activity Editor Status Line Category III	570
37	Activity Editor Edit Menu	571
38	Activity Editor Activity Pull Right	573

39	Activity Editor Expansion of a Subtask	575
40	Activity Editor Subtask Options Menu	576
41	Activity Editor Subtask Options Menu	577
42	Activity Editor Availability Edit Workbox	580
43	Activity Editor Updated Availability List	581
44	Activity Editor Scheduling Menu	584
45	Activity Editor Scheduling Changes	585
46	Schedule Changes due to Delete	587
47	Delete Subtask Pull Right Menu	589
48	Subtask Deletion Information	589
49	Switch Grouping	593
50	Seizure Select Menu	594
51	ILA Seizure Information Menu	596
52	Switch Description Lines	597
53	Switch Description Line Reference	598
54	Switch Seizure Workbox	600
55	Add Menu with Load Center Pull Right	602
56	Updated ILA Seizure Information Menu	603
57	Manual Release Menu	604

1 Purpose of User Manual

The purpose of this manual is to give the user instructions and guidance in the general operation of the breadboard from the Solbourne. The Solbourne provides the primary user-interface for the operation of the *SSM/PMAD* breadboard known as the *SSM/PMAD* Interface. This interface will be the primary focus of the user manual.

The manual is organized into four sections. The rest of section one defines the acronyms and some brief definitions that are used throughout this manual. Section two describes the installation of the *SSM/PMAD* interface on the Solbourne. Section three provides a background on key elements of the system. Section four is the main part of the user manual and provides the user with the necessary details for operating the *SSM/PMAD* interface.

1.1 Acronyms and Definitions

AI Artificial Intelligence. The field of Computer Science studying aspects of human intelligence and how they can be implemented on a computer.

CLOS Common LISP Object System. An object-oriented programming framework for Common LISP.

DBMS Database Management System.

Fault Isolation Fault isolation determines the cause of a fault in normal mode of operation.

Fault Reporting When a fault occurs in the power system hardware, the fault is reported to the user and FRAMEs. Diagnosis and recovery occurs as a result.

FELES Front End Load Enable Scheduler. A schedule translator and associated functions for the *SSM/PMAD* system. It defines a list of events for turning on and off switches of the breadboard.

FRAMEs Fault Recovery and Management Expert System. The set of expert systems and deterministic algorithms for determining and diagnosing hard, soft and incipient faults during normal operation of the system.

GC Generic Controller. An element of the hardware of the *SSM/PMAD* breadboard for interfacing to the controlling RPCs.

H/W Hardware. The *SSM/PMAD* breadboard hardware components.

ICD Interface Control Document.

ILA Intermediate Levels of Autonomy.

KANT Knowledge and Negotiation Tool. The collection of planning and negotiation software activities within the *SSM/PMAD* System.

KHz KiloHertz.

KNOMAD Knowledge Management and Design Environment. The supporting software for the FRAMES knowledge base and advanced AI programming for the *SSM/PMAD* system.

LC Load Center. An element of the *SSM/PMAD* breadboard where loads may be connected to 1K-RPCs.

Limit Checking This function determines if the amount of current going through a switch is within the allowed amount.

LISP List Processing. The programming language of choice for developing complex software systems utilizing AI technology.

LLP Lower Level Processor. A computer system for controlling LCs or PDCUs.

Load Shedding This function shuts off switches in the load centers if the amount of current going through the switch is more than allowed as determined by the limit checking function. In the PDCU, the illegal use of current is reported to the user and FRAMES if appropriate.

LPL Load Priority List. The load priority list is used to determine which loads are more important than others in the event of power system contingencies.

LPLMS Load Priority List Management System. An expert system load priority list generator used to define priorities on switches that map to scheduled events. These priorities are used by the *SSM/PMAD* system to determine switch allocations in contingency operations.

MAESTRO Master of Automated Expert Scheduling Through Resource Orchestration. The scheduler is responsible for taking a set of activities as input and using a model of the power system topology, producing an efficient schedule of activities for an output.

MSFC Marshall Space Flight Center.

NASA National Aeronautics and Space Administration.

Performance Monitoring The performance monitoring function allows the user to observe how the power is being utilized compared to how it was scheduled. The function includes monitoring and averaging power over time.

PDCU Power Distribution Control Unit. A power distribution unit of the *SSM/PMAD* hardware.

Redundant Switching If a load being powered by a switch has been declared to be important and is scheduled in a redundant fashion, then the redundant switch to the load will be turned on in a contingency situation.

RBI Remote Bus Isolator. A remotely controllable relay for carrying large amounts of power (e.g. 15KW).

RCCB Remote Control Circuit Breaker. A relatively smart switch for switching 10KW of power.

RPC Remote Power Controller. A smart switch that may switch either 3KW or 1KW of power.

RS-423 Electronic Industries Associates RS standard for communications. The RS-423 communications between the LLPs and the SICs.

Schedule Execution The events generated by the schedule are executed and cause switches to be turned on and off.

System Operation This function is the main controlling function of the *SSM/PMAD* system. It manages all the other functions of the system.

SSM/PMAD Interface This interface is used to operate the system in both normal and maintenance modes. It initializes and stops the system.

SIC Switchgear Interface Controller. The element of the hardware for interfacing to a bus of switches. In the case of a PDCU, it controls all the switches (only one bus in a PDCU).

SSM Space Station Module.

SSM/PMAD Space Station Module Power Management and Distribution. A testbed for trying and evaluating mechanisms for autonomously and manually operating the power system for Space Station Freedom.

S/W Software. Those elements of the *SSM/PMAD* breadboard functions that are not hardware.

TCP/IP Transport Control Protocol / Internet Protocol. The communications medium used between the LLPs, the Solbourne and the Symbolics.

UI User Interface. The place where the user interacts with the *SSM/PMAD* breadboard.

VCLR Visual Control Logic Representation. A means for representing and documenting the logical flow of an algorithm.

2 Installation of *SSM/PMAD* from the Solbourne

This section describes the installation of *SSM/PMAD* from the Solbourne. This includes a subsection on installing the LLP software.

The installation of the *SSM/PMAD* software involves loading the requisite application software onto the Solbourne and copying the necessary initialization files into the user account. This installation assumes that the Solbourne is installed and running in a normal network environment.

This installation assumes that UNIX and X windows are installed. The only actual changes that will be necessary to make to the existing installation of the X windows software on the Solbourne are:

1. Change the X server to one provided by FRANZ, Inc.
2. Edit Hosts File for LLP locations.

2.1 Loading the Software

The *SSM/PMAD* software distribution includes the LLP software, FRAMES, LPLMS, FELLES, KNOMAD, pbm-plus, and the FRANZ, Inc. X server as well as Franz's Allegro Common LISP.

A few legal disclaimers are necessary. NASA has bought two Solbourne 5/501 computers under contract NAS8-36433. Each has a license for Franz Allegro Common LISP and Common Windows on X. This installation includes the Allegro Common LISP that is licensed to NASA on their Solbourne 5/501 computers. It includes FRAMES and KNOMAD, developed by Martin Marietta, and PbmPlus. The X server provided by Franz, Inc as part of this distribution is the public domain X server (Version 11 Release 4) as distributed by MIT. PbmPlus is a public domain set of programs for converting between various picture formats such as GIF, SUN Raster, X, bitmap, etc. PbmPlus is in the public domain and was developed by Jef Poskanzer.

The software distribution is located in the "/usr/local" directory. To load the software you must be in the "/usr/local" directory. If you have any difficulty understanding the below steps, have your system administrator read this and load it for you. A familiarity with the UNIX operating system is assumed.

To load the software, insert the distribution tape (5 1/4") in the local tape drive and perform the following steps where '%' is the unix prompt:

1. %cd /usr/local
2. Rewind the tape. %mt -f /dev/rst8 rew
3. Copy the file. %tar -xvf /dev/rst8

4. Remove the tape when copying is completed.

2.2 Changing the X Server

The next step is to change the X server, by performing the following steps:

1. `%cd /usr/bin/X11`
2. `%rm X`
3. The link command (`ln`) gives two names to the same file. `%ln -s /usr/local/bin/Xsun /usr/bin/X11/X`

2.3 Copying the Initialization Files

The final step to installing the software is to install the initialization files into your user account and edit your `.cshrc` file. You should be in your local user directory. The installation of the initialization files will install `.clinit.cl`, `.xinitrc`, and `.twmrc` into your home directory.

If you already have some of these files, you may want to save them to another name and merge them together with the newly installed files. However, you should understand exactly what the commands in these files are doing in relationship to operating the *SSM/PMAD* interface before you do this.

Assuming you are working in the cshell, from your local directory, type the following commands exactly as shown:

1. `%cd`
2. `%cp /usr/local/ssmpmad-if/dot-files/.??* .`

2.4 Creating an LLP Software Diskette

To create a new LLP software disk, perform the following steps:

1. Take the LLP executable diskette (part of the *SSM/PMAD* delivery) and make a diskcopy of it onto a blank 1.2 MByte diskette. This is done by typing at the DOS prompt of the 80386 PC LLP computer:

```
diskcopy A: A: <cr>
```

The computer will respond by prompting for either the Source diskette or the Target disk. The Source diskette is the LLP executable disk. The Target diskette is the new, uninitialized diskette. The computer will repeatedly prompt for one diskette or the other until the copy is completed.

2. Once the copy is made, the LLP executable diskette is no longer needed. All that is required is to give the new LLP software diskette an appropriate host table. Make sure the new LLP software diskette is in drive A.

The following command will move the proper host table to the correct directory for normal operations for LLP B.

```
copy A:\tables\hosts.B A:\cmcnet\hosts <cr>
```

For LLP C the command is the same except:

```
copy A:\tables\hosts.C A:\cmcnet\hosts <cr>
```

For LLP A the command is the same except:

```
copy A:\tables\hosts.A A:\cmcnet\hosts <cr>
```

Once the host table is in the correct directory, the new LLP software diskette is ready for operation.

(This Page Intentionally Left Blank)

3 Overview of the *SSM/PMAD* System

The *SSM/PMAD* system consists of the hardware and software for efficiently managing and operating a space station module like power system. This includes the necessary scheduling of power, the control of hardware switches, and the overall control of the system during normal and contingency operations.

3.1 Purpose of the *SSM/PMAD* System

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 personnel who planned and scheduled 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 appropriately 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 associated deterministic processing supports it.

The *SSM/PMAD* user interface possesses the capability to add a human user to its autonomously executing plan and schedule. A feature called "Intermediate Levels of Autonomy" (ILA) effectively manages the complex user input activities with a minimal impact to the system. ILA encompasses directed autonomous system activities which range between fully manual and fully autonomous, but not including the end points. The system consists of a run-time planning system which is reactive to the user's needs and functions independently with the priority management and scheduling environments to provide alternatives to the user. The system provides simultaneous multi-agent knowledge system processing, allowing a user to remain free from having to possess specific knowledge about system state in a rapidly changing environment. The system presents an user environment in which the user is

not overwhelmed by presentations of vast amounts of information from a highly distributed process and control system.

3.2 Hardware Configuration

The *SSM/PMAD* hardware configuration is distributed intelligent control from the LLPs. The arrangement is a "STAR" bus with hierarchical data handling as depicted in Figure 1.

The power system includes the following hardware components:

- The switches are one kilowatt (1K), three kilowatt (3K), and fifteen kilowatt (15K) remotely operable devices.
- The remote bus isolator (RBI) is a switch that can be remotely commanded on or off but will not trip.
- The 1K and 3K switches are remote power controllers (RPCs) that are smart switches, and will trip on a low-impedance short.
- Sensors are implemented in different sizes (50 and 100 amps) depending on the rating of the sensed current. All voltage sensors are 120 volt DC sensors.
- An analog to digital converter (A/D) is used to collect analog data from sensors and digitize it.
- 80386 PCs function as lowest level processors (LLPs). Each LLP is responsible for controlling either a load center or a power distribution and control unit (PDCU). This involves communicating to the SIC and A/D cards of the hardware for that LLP. Each SIC is then responsible to control a set of generic cards for operating the RPCs.
- The switchgear interface controller (SIC) allows communications between the hardware and software components. There is one SIC for each bus of a load center and one SIC for each PDCU. Each SIC controls a set of generic cards (GC).
- The switch controller is a generic controller (GC) that adds further logic to the switches. For an RPC, the generic controller will detect high impedance shorts that will result in an i^2t trip as well as low voltage situations that will result in an under voltage trip. The GC allows both AC and DC switches to be controlled.

3.3 Distributed Control and Process Management

The simplified physical organization of the *SSM/PMAD* system depicted in Figure 2 provides important conceptual principles for the general operation of the system. Timing constraints

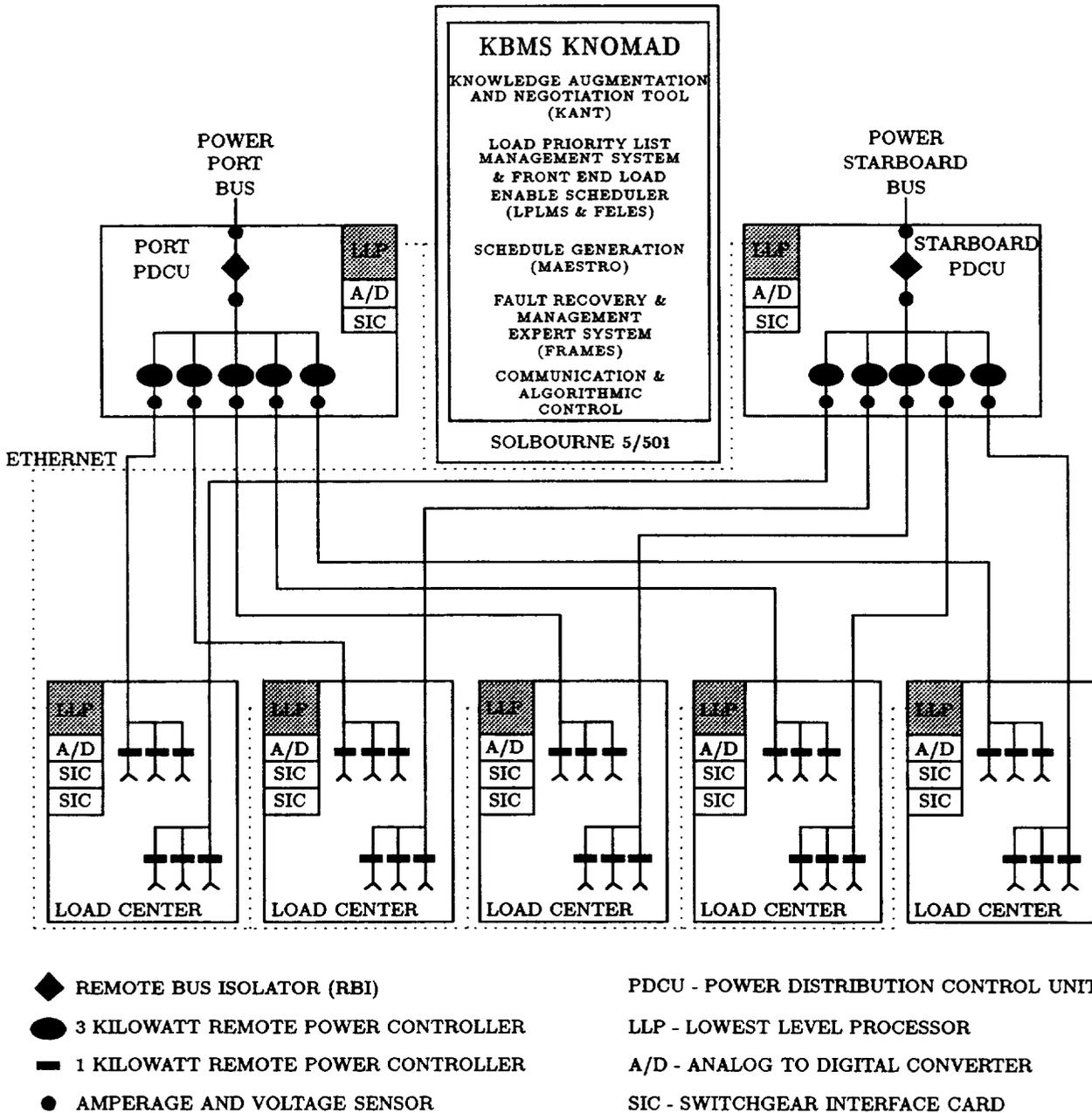


Figure 1: SSM/PMAD Hardware Configuration

helped dictate the physical organization of the *SSM/PMAD* system. The critical operations of tripping a switch due to low-impedance shorts in the power system are carried out in the millisecond region and are performed in the switchgear itself. Those operations which need to be performed within seconds are partitioned to the LLPs. Fault diagnosis and system control operations are partitioned in the next level up and operate in the seconds to minutes region. Schedule generation and regeneration in a contingency situation are also partitioned at the this level of the system and operate in the minutes region.

In Figure 2, notice that there are three management functions; the KNOMAD knowledge manager, the FRAMES fault management system, and LPLMS priority manager. There is one planning function, KANT; one scheduler, MAESTRO; and one user interface consisting of the FELES and the U/I functions. All of these functional elements, whether knowledge based or procedural have a common interface to rule bases and a common interface to databases. With common rule base and database interface capability, KANT, placed strategically between the user and the data, can help the user plan and decide which activities and resources to obtain for required periods of ILA activity. The LLPs report data and state changes from the distributed 80386 processors up to FRAMES for processing at the upper level.

The diagram in Figure 3 depicts the *SSM/PMAD* System architecture hierarchy in still another way.

3.4 Contingencies

In the *SSM/PMAD* test bed, contingencies may be caused from different sources and may be handled in many ways. The user requesting ILA activities may cause a contingency based upon interruption of a scheduled or planned activity. If the contingency can be handled, the contingency receives a maximum priority. Any contingencies generated by a user requesting ILA are handled as perturbations to the planned schedule causing rescheduling following the ILA activity.

3.5 Fault Diagnoses

The expert systems of the *SSM/PMAD* breadboard that handle the fault diagnosis aspect of the system is called FRAMES. The FRAMES knowledge base is a very complex part of the *SSM/PMAD* breadboard. It logically consists of a domain file that defines all the data that the expert systems use to reason about the power system switchgear. It consists of domain functions used by the expert systems to compute algorithmic functions. It also consists of the main knowledge base and the various rule groups of the expert systems. These are presented in their entirety in the *SSM/PMAD* Technical Reference, Interim Final Report, Volume 2, MCR-89-516.

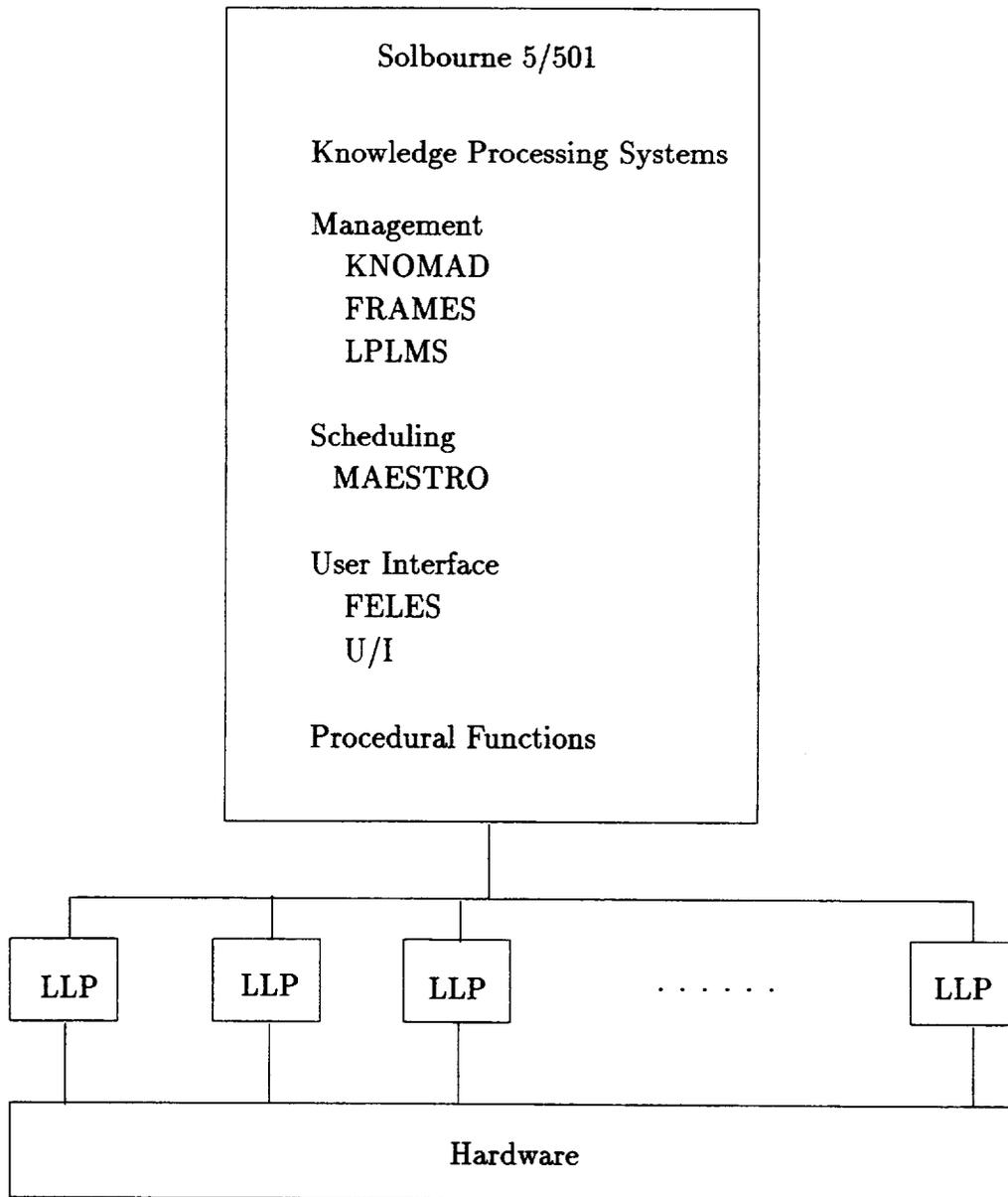


Figure 2: Distributed Control and Process Management

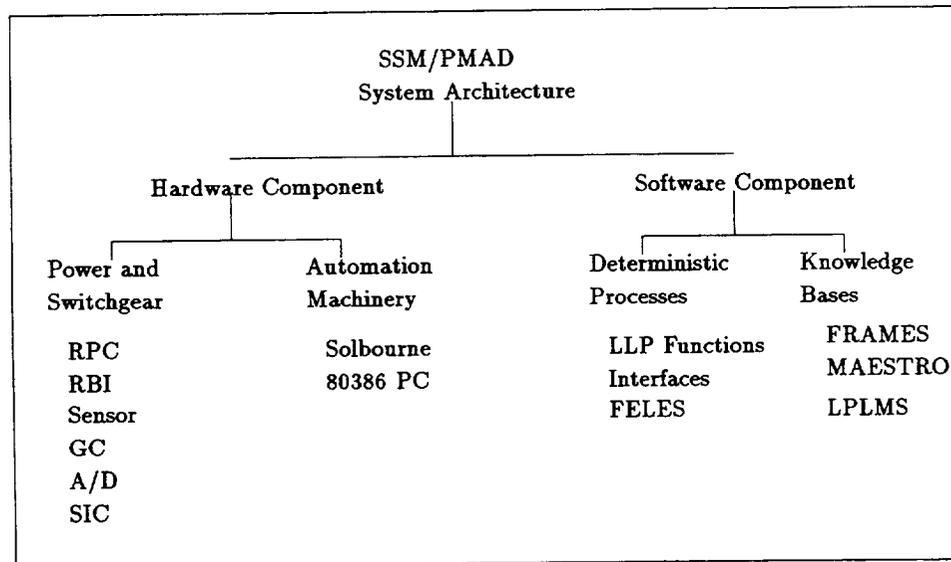


Figure 3: Architecture Hierarchy

The FRAMES system is partitioned into three major divisions based upon the response time needed at the different partitions. The three partitions are the distributed lower level processors for controlling the hardware, the fault isolation and diagnosis expert systems, and the scheduling system.

- The first partition, the switch hardware is controlled by the lower level processes which reside on PC clones. The algorithmic processes at this level control the operation of turning switches on and off as well as monitoring power levels and performing limit checking. The lower level processes detect fault conditions which include a switch physically tripping off due to an over current or under voltage situation in the hardware. These fault symptoms are communicated to the fault isolation and diagnosis expert systems. Additionally, scheduled operations may no longer be performed on the tripped switches.

The response time of the lower level processors is necessarily fast. Typically, limit checking operations to shut a load off if it is using too much power, for example, are done within a one second period. The speed of the lower level processes also has implications on later fault isolation. It is quite possible that if an I²t short is occurring in the hardware at a level of approximately 120% of the switch's rating, that it could take the switch up to five seconds to trip. The lower level processor will shut the switch off much sooner.

- The second partition is the fault isolation and diagnosis part of FRAMES. This part consists of a number of traditional expert systems for diagnosing different types of problems in the power hardware as well as maintaining other knowledge intensive states such as the load priority list. Currently three expert systems are defined to exist at this level: The Load Priority List Management System (LPLMS), the fault diagnosis expert system, and the soft fault expert system.

The fault isolation and diagnosis expert system requires many supporting functions in addition to the rules that make it up. Embedding knowledge intensive applications into real world complex systems require many parts for a successful system. In the FRAMES system these additional functions include detecting and monitoring the hardware (done by the lower level processors); communication algorithms for communicating with the distributed processors; algorithmic processes for logging data, updating database values, and the like; and user interface functions to make the system useful.

- The third partition, the scheduling system is not required to be nearly as responsive. Its role is to create a schedule for operating the switches in advance and to maintain that schedule during contingencies in the power system. In the present system, schedules are shipped to the lower level processors in thirty minute blocks. This allows for a semi-graceful degradation of the overall system performance if the scheduler becomes inoperable for some reason. When a fault has been diagnosed in the power system and a set of switches has been determined unusable, the scheduler is expected to reschedule its activities in a reasonable amount of time. The scheduler has been partitioned at the highest level and is expected to perform in a period of minutes.

3.5.1 Types of Faults Diagnosed

There are three types of faults that can occur in the *SSM/PMAD* system. Each is described briefly below.

- Hard Faults
- Soft Faults
- Incipient Faults

The four types of hard faults in the *SSM/PMAD* system are:

- OT Over temperature occurs when an RPC is over the specified temperature.
- UV An under voltage occurs if a less than nominal voltage is registered at the input to the RPC.

- FT Fast trip occurs when the current exceeds 130% of the rating for the RPC.
- I²t Over current occurs when the current is greater than 100% of the rating for the RPC and less than 130% of the rating for the RPC. An over current seldom occurs because the LLP software is fast enough to sense the current is out of range for the schedule constraints, and the switch is "shed".

The two types of soft faults in the *SSM/PMAD* system are:

- The first type of soft fault occurs when power usage exceeds a budgeted limit but does not exceed an absolute power limit.
- The second type of soft fault occurs during periods of ILA when a user can utilize switches or loads known to trip under known conditions. This will be identified to the system as a resource allocation for the ILA period. The user is made aware of the potential problems of using the resource, and the system will record the situation such that if the problem occurs again, appropriate action is taken.

The three types of incipient faults in the *SSM/PMAD* system are:

- The first type of incipient fault occurs when whenever the mean time between failures (MTBF) is within delta time of occurring. Delta is an planner-chosen variable introduced at schedule generation time.
- The second type of incipient fault occurs when a soft fault has happened which has not caused an LLP recognized overlimit condition.
- The third type of incipient fault occurs when a user chooses any number of faulted switches to be operational during intermediate levels of autonomy control periods. An incipient fault condition exists for switch activities on previously faulted switches which were soft faults due to internal load anomalies.

4 The User Interface

The *SSM/PMAD* system user interface allows a power system user the ability to perform power management tasks with a minimum of computer knowledge. The present user interface is the culmination of several years of development and interface iteration.

4.1 Bringing up the system

This section describes the procedure for starting the power system and the LLPs.

To operate the *SSM/PMAD*, it is recommended that the operator bring up the system in the order given here, that is, the power system, and then the Solbourne interface.

4.1.1 The Power System

The power system is the simplest component to operate. There are two important steps to starting the power system. The first step is to turn on the power to the *SSM/PMAD* breadboard. The second step is to turn on the LLPs that will be operated. It is assumed that all the wiring of the breadboard is correct, and that the cabling of the LLPs and power system hardware is correct.

Breadboard Power Turning on the power to the *SSM/PMAD* breadboard is site dependent. The only substantial requirement is that 120 Volt DC power be supplied to each bus that will be operated. For bus A (controlled by PDCU a), 120 Volt DC power should be supplied to the RBI of PDCU A. Similarly for bus B.

Starting the LLPs Before operating the *SSM/PMAD*, decide which LLPs will be a part of the running system. Confirm that the proper system diskette is in each LLP (The 5 1/4" system diskette for Port LLP should be in the Port LLP, Load Center 1 in the Load Center 1 LLP, etc) Then turn the LLP on and let it boot.

It is possible to add an LLP to the already operating system in Maintenance mode, if the new LLP is a load center. Simply turn it on and let it boot. The *SSM/PMAD* user will need to connect to the new LLP. If you add a first or second PDCU, it is recommended that you exit the *SSM/PMAD* system, and bring up the system again with the PDCU as a part of the system.

Shutting Down the Power System Power down the system by doing the following:

- Turn off the 120V power supplies for the Port and Starboard buses.
- Turn off the housekeeping power supplies that supply power to the switchgear.

- Remove each LLP diskette from the 80386 PCs.
- Turn off the power to each LLP.

4.2 Logging onto the System

The section describes the initial operation of *SSM/PMAD* system from the Solbourne interface. Type in the following commands to login to the *SSM/PMAD* system in Denver:

1. Login to the Solbourne At "Daneel login:" prompt, type in user name provided by your system administrator.
2. After the "password" prompt, type in the user's password.
3. % Type: startx <return>
4. The "*scratch*" window and "minibuffer" window come up. To go to the buffer window, type: <ESC> x (This moves the cursor to the minibuffer window) fi:common-lisp <return>

Two significant initialization files are ".clinit.cl" and ".twmrc". The .clinit.cl file defines a number of initialization steps necessary for LISP networking. It also includes a *hardware-location* variable which is very important for talking to the correct set of hardware. The .twmrc file is used primarily for the purpose of setting border colors of windows and foreground colors of menu selections.

5. Lisp is up. Load the necessary files to run *SSM/PMAD*.
(load "/usr/local/ssmpmad-if/ssmpmad.sys")
(load-system 'ssmpmad-system)
(go-to-created *ssmpmad-if*)
If running the system in Huntsville from Denver, type: (setf (ssmpmad-if-location *ssmpmad-if*) :huntsville)
(load-system 'ui-system)
(run-startup-functions) The *SSM/PMAD* screen will appear.

Type in the following commands to login to the *SSM/PMAD* system in Huntsville:

1. Login to Solbourne. At "Marvin login:" prompt, type: PMAD
2. % Type: PMAD <return> The *SSM/PMAD* screen will appear.

4.2.1 Running *SSM/PMAD* from a Remote Location

To run the *SSM/PMAD* system from a remote location on a compatible workstation running X-windows and the TWM window manager. The user must have files: `ssmpmad.cl` and `ssmpmad`. The file `ssmpmad.cl` is the executable file. The file `ssmpmad` contains other essential non-compressed files. They must be located in the same directory.

The remote workstation must have a screen resolution of at least 1152 x 900 pixels. The user must also have all of the *SSM/PMAD* database files: `equipment-mode.db`, `resources.db`, `subtask.db`, `activity.db`, and `sched-ind.cl`. These files are normally located in `/usr/local/ssmpmad-if/database/`. The user may set command line variables or environment variables to point to the database files, the database path, the X-display type, and the X-display. See Table 1.

Environment Variable	Command-Line Variable	Default Value
DISPLAY	DISPLAY	:0
X-DISPLAY-TYPE	X-DISPLAY-TYPE	COLOR
EQUIPMENT-DB	EQUIPMENT-DB	equipment-mode.db
RESOURCE-DB	RESOURCE-DB	resources.db
SUBTASK-DB	SUBTASK-DB	subtask.db
ACTIVITY-DB	ACTIVITY-DB	activity.db
SCHEDULE-LIBRARY	SCHEDULE-LIBRARY	sched-ind.cl
DATABASE-PATH	DATABASE-PATH	/usr/local/ssmpmad-if/database/

Table 1: Environment and Command-Line Variables for *SSM/PMAD*

The environment variable names, like `DISPLAY`, are case-sensitive and must be all capital letters. The command-line variable names are case insensitive. The `DISPLAY` variable refers to the name of the display screen on which to display the user interface. The variable `X-DISPLAY-TYPE` can either be `color` or `greyscale` or `grayscale`. `EQUIPMENT-DB`, `RESOURCE-DB`, `SUBTASK-DB`, `ACTIVITY-DB`, and `SCHEDULE-LIBRARY` variable are only used if the database file names have been changed. The `DATABASE-PATH` variable is used to specify a different path for finding the database files, the path name must be absolute and must contain a closing `/`.

The following examples show how to set equivalent environment variables and command-line variables. An example of setting up the environment is as follows:

```
setenv DISPLAY fubar:0
```

This sets `DISPLAY` to `fubar:0`

```
setenv X-DISPLAY-TYPE greyscale
```

This sets `X-DISPLAY-TYPE` to `greyscale`

setenv EQUIPMENT-DB equipment-mode-test.db
This sets EQUIPMENT-DB to equipment-mode-test.db

setenv RESOURCE-DB resources-test.db
This sets RESOURCE-DB to resources-test.db

ssmpmad.cl
This command line runs SSM/PMAD System

All the above commands are equivalent to:

```
ssmpmad.cl -- -display fubar:0 -x-display-type greyscale -equipment-db equipment-mode-test.db -resource-db resources-test.db
```

This runs the SSM/PMAD System with the display set to fubar:0, the x-display-type set to greyscale, the equipment database set to equipment-mode-test.db, and the resources database set to resources-test.db. Notice that the environment variables are not case-sensitive in a command-line format. The "--" must be used in the command-line. The variables before "--" go to the lisp start-up, and the variables behind "--" go to the application start-up.

4.3 Screen Layout and Interface Components

The *SSM/PMAD* system user interface allows a power system user the ability to perform power management tasks with a minimum of computer knowledge. The present user interface is the culmination of several years of development and interface iteration.

In designing the current version of *SSM/PMAD* user interface, several types of interface objects needed for system interaction were defined. These objects include windows, workboxes, buttons, and menus. See Figure 4.

- A window is defined as a rectangular area on the user interface screen where graphic or textual information may be displayed.
- A workbox is defined as a window within which the system prompts the user for information. Workboxes are dynamic with their context being driven transparently to the user. A workbox is broken down into three regions. The first region is a scrolling graphics window in which the user interactively enters data. The second region is below the scrolling window region and is used to display messages to the user. The third region contains all appropriate buttons for the workbox. See Figure 5.

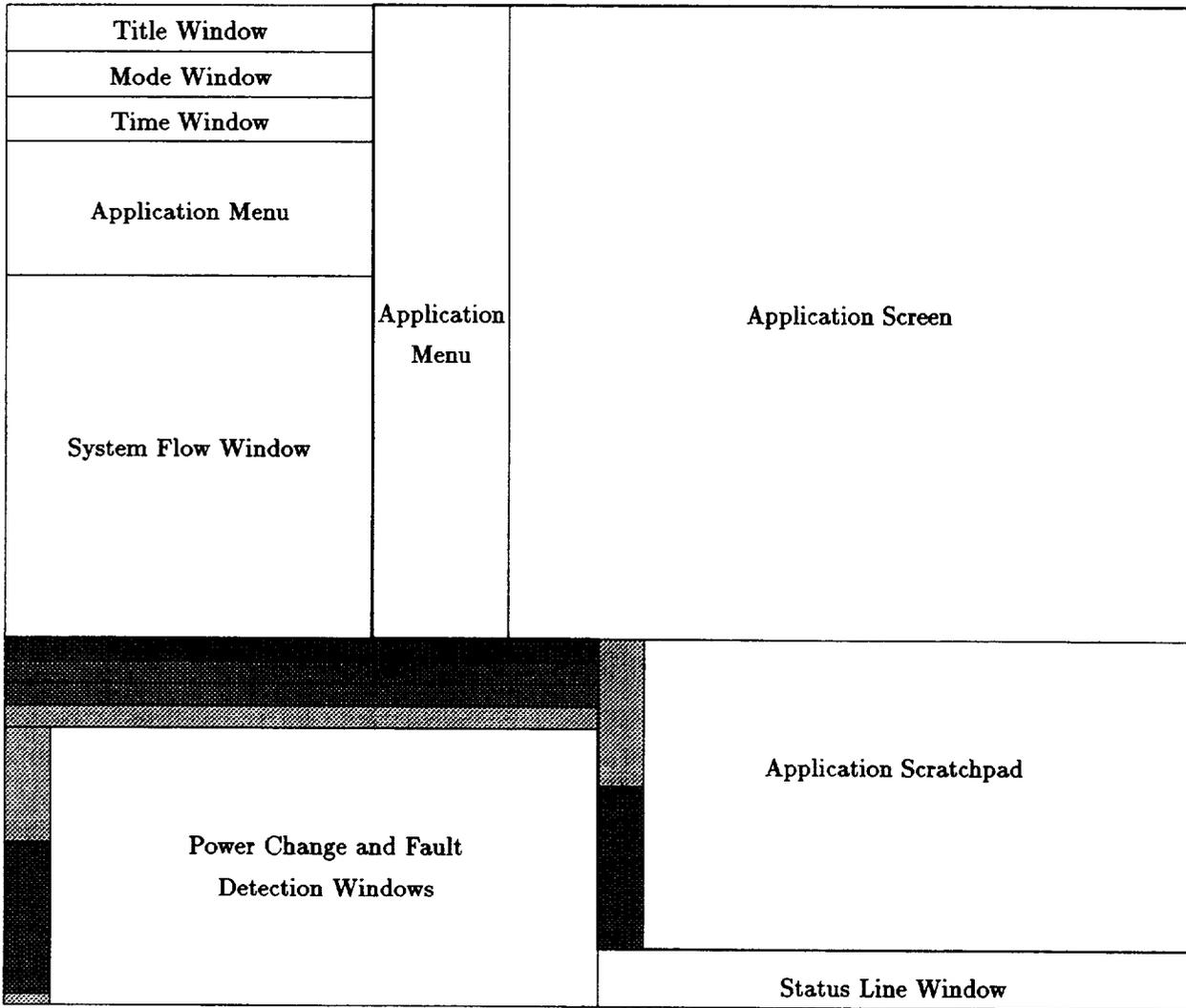


Figure 4: Screen Layout

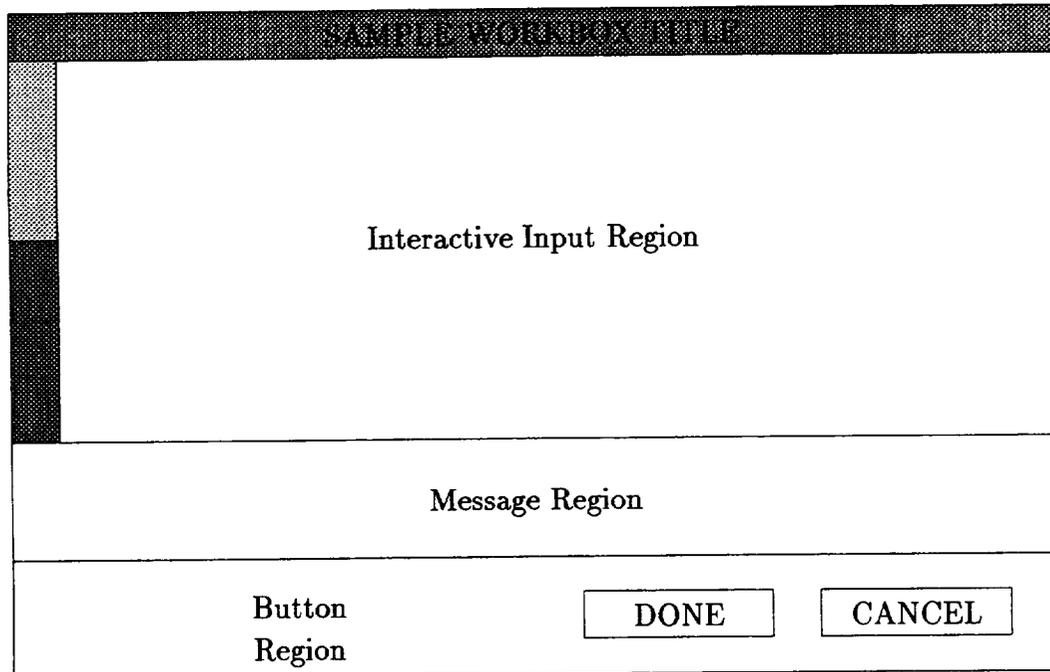


Figure 5: Sample Workbox

All workboxes have "Done" and "Cancel" buttons. Any button click outside the workbox is interpreted as a button waveoff and has the same effect as mouse clicking the "Cancel" button.

- A button is defined as a rectangular region on the user interface which, when selected, initiates an interface response. "Done" and "Cancel" are specific instances of buttons inside a workbox.
- A menu is defined as a list of possible selections which are accessed through mouse cursor movement and button clicks.

Pop up menus are used throughout the *SSM/PMAD* interface application. Pop up menus are either single selection or multiple choice menus. Examples of each are shown in Figure 6 and Figure 7. A pull-right submenu is display upon entering a selection area and is removed when the selection area is exited. A button click outside the pop up menu is interpreted as a button waveoff and the menu is removed.

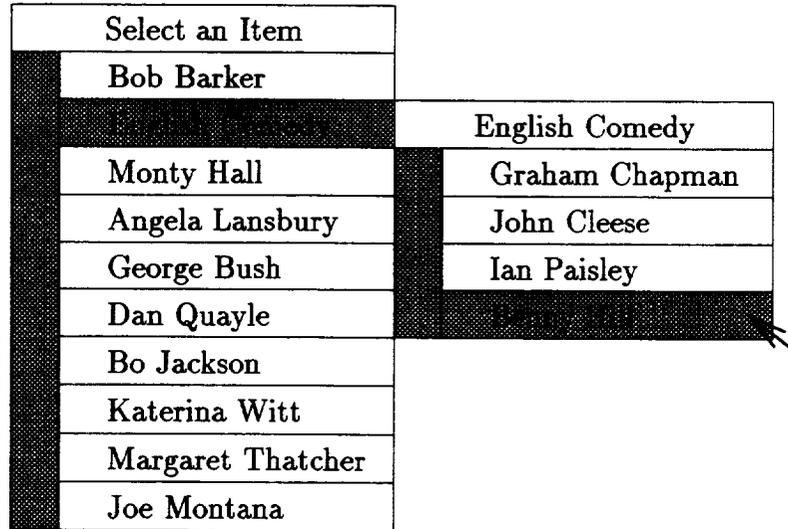


Figure 6: Single Selection Menu with a Pull Right

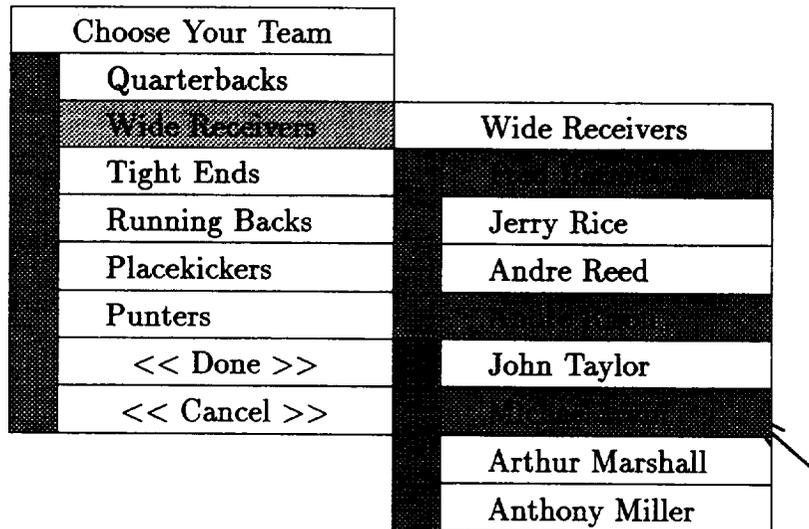


Figure 7: Multiple Choice Menu with a Pull Right

APPLICATIONS	
POWER SYSTEM	POWER UTILIZATION
FELES	VIEW FILES
ACTIVITY EDITOR	

Figure 8: Screen Selections

GMT	Mission Time
14.17:12	00.01.54

Figure 9: Clock Bar Window

4.3.1 Applications Selection Window

This window is a menu of application screen available to the operator. See Figure 8. Each item in the menu may be selected by clicking the mouse on it. When this is done, the associated application screen becomes the active one available to the operator. When the system first comes up, the application window will be the Power System screen.

4.3.2 Clock Bar Window

This window displays both the Greenwich Mean Time abbreviated GMT time and the *SSM/PMAD* mission time. Mission Time displays how far the system is into the current "run". See Figure 9.

All times in the system are displayed in terms of mission time. The format is always "dd.hh:mm", where dd is a day number, hh is an hour, and mm is a minute. These times shall be updated on a minute basis.

4.3.3 Mode Indicator Window

This window displays the current *SSM/PMAD* mode. See Figure 10. The mode can be one of the following: Idle, Normal, or Maintenance, and is always visible.

When the system first comes up, the mode is "Idle". Each of the modes is discussed in detail under section 4.4.1.



MAINTENANCE

Figure 10: Mode Indicator

4.3.4 System Flow Window

The System Flow window describes the state or status of various components of the *SSM/PMAD* system. See Figure 11. Every component has a state associated with it (from the perspective of the software on the Solbourne). The software components that are displayed are MAESTRO, FELES, LPLMS, FRAMES, KANT, and the User Interface. An example state might be "Computing Priority List" displayed by LPLMS.

This window displays interaction within the *SSM/PMAD* system's software components by highlighting the component and displaying a brief message about what is happening.

The System Flow window displays communication connections between software components not resident on the same machine as the user interface. The non-resident components with connectivity which are displayed are APEX and the Lowest Level Processors (LLPs). Those components have a status of either "c" or "nc" for connected or not connected, respectively.

The APEX component box is mouse sensitive and allows the user to attempt to connect or disconnect with/from the APEX system with a simple mouse click. This feature is available in all modes of operation of the *SSM/PMAD* system.

The LLP component boxes are also mouse sensitive allowing the user to connect or disconnect with/from the LLP with a mouse click, but only in maintenance mode of operation.

The system flow window contains four mouse sensitive boxes: hard faults, soft faults, incipient faults, and power changes. When a box is selected, it makes visible the appropriate fault diagnosis and detection window by placing it on top of the others. This window is always visible.

4.3.5 The Four Fault Diagnoses and Detection Windows

The four fault diagnosis and detection windows are "Hard Fault Diagnosis", "Soft Fault Diagnosis", "Incipient Fault Detection", and "Source Power Changes". Each of these windows is a scrolling window with at least two window's worth of scrollability, which occupies the same space at the lower left hand corner of the User Interface Screen. See Figure 12.

Each of these windows has its title bar visible. The title bar is mouse selectable with the selection moving the selected window to the top layer thereby making its scrolling window

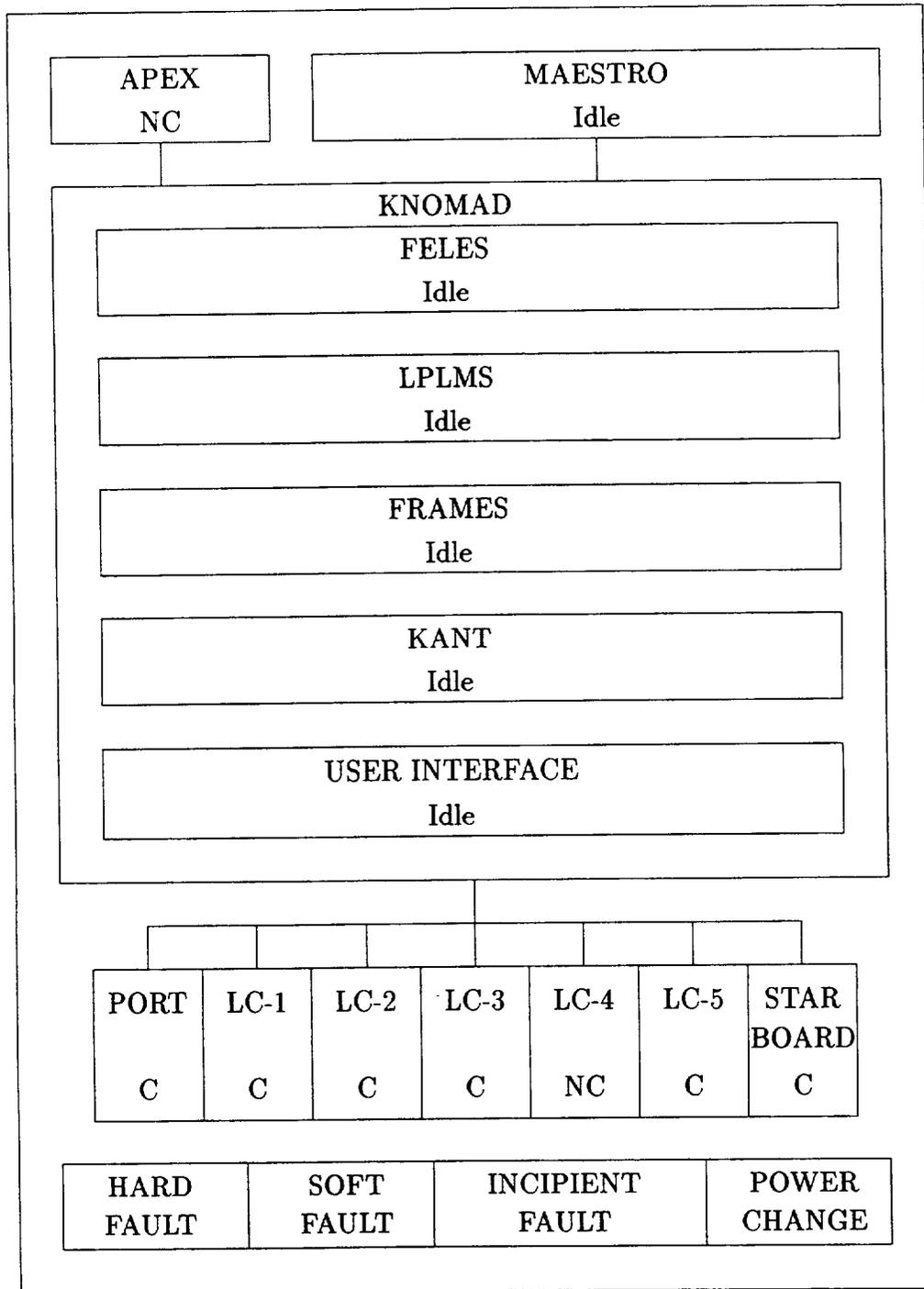


Figure 11: System-Flow

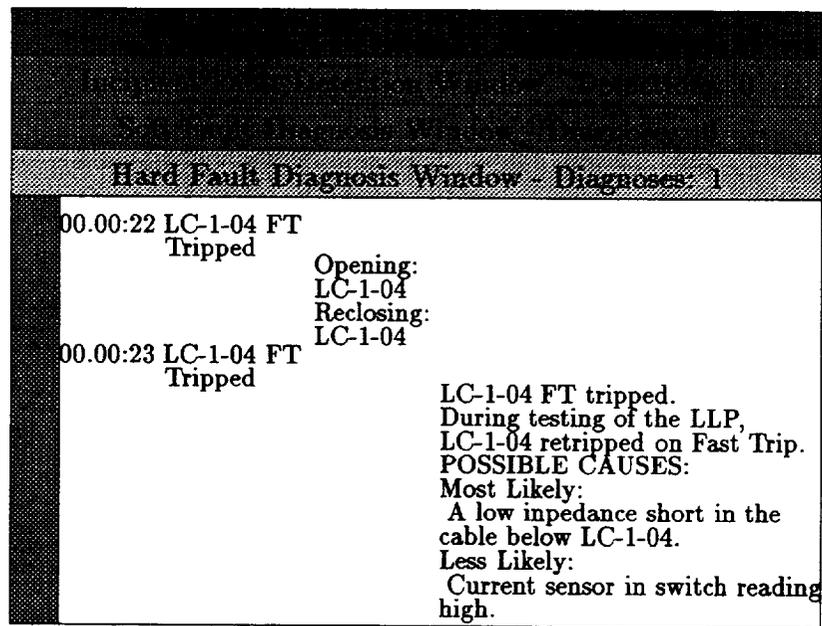


Figure 12: User Interface Fault Windows and Source Power Change Window

visible. The title bar lists the number of hard fault diagnoses, soft fault diagnoses, incipient fault detections, and source power changes discovered since the beginning of the mission (mission time 0).

As an added feature, a fault diagnosis and detection window being written to automatically rotates to the top. This feature is capable of being disabled/enabled from the system options pop up menu.

4.3.6 Application Window, Menu and Scratchpad

The application screen is comprised of the application window, the application menu and the application scratchpad window. The *SSM/PMAD* User Interface has four application screens which may be accessed through the applications selection menu. Only one application screen is visible on the *SSM/PMAD* User Interface at a given time. For this reason, application screens are dynamic and may be switched from one to another as a user sees fit. The four applications that are available are:

- Power System
- FELES
- Power Utilization
- Activity Editor

The Power System application screen is examined in detail in section 4.5. The FELES application screen is examined in detail in section 4.7. The Power Utilization application screen is examined in detail in section 4.6. The Activity Editor application screen is examined in detail in section 4.8.

The application menu contains mouse button selected functionality in support of the application. These mouse sensitive areas are grayed out if the functionality is not available. The first item in the application menu is always the system options menu. The application menu always contains a help button for more information on the application.

Ancillary information resulting from application menu button functionality is displayed in the application scratchpad window. The application scratchpad window is a scrolling window with several windows' worth of scrolling area.

4.3.7 Status Line Window

This window supplies the user information about the possible actions using the mouse.

4.4 System Options

System Options appears at the top of every application menu. The following menu options are described below:

- Mode Changes
- Schedule Functions
- Other System Operations
- Utilities
- Help
- Exit *SSM/PMAD*
- Cancel

4.4.1 Mode Changes

This option brings up a submenu allowing the user to change the *SSM/PMAD* system mode of operations. Each mode change is described in detail below.

- Go to Normal Mode
- Go to Maintenance Mode
- Go to Idle Mode

The Normal Mode of Operation To put the system into normal mode, the operator should select this option. See Figure 13. Any switches that may have been on due to maintenance operations will be turned off in preparation for manual operations. This mode allows for both autonomous scheduling and manual intervention concurrently.

The effects of the various mode changes to "normal" are:

Idle → Normal: This mode change puts the user in Normal mode.

Maintenance → Normal: This operation will remove Maintenance activities from the active schedule causing any switches that were on to be turned off in preparation for Normal mode. The system will then go to Normal mode.

When the "Go to Normal Mode" is selected,

- In the system flow window on the screen, KANT will say "Go to Normal".
- When the system is in normal mode, the mode indicator window will say: "NORMAL"

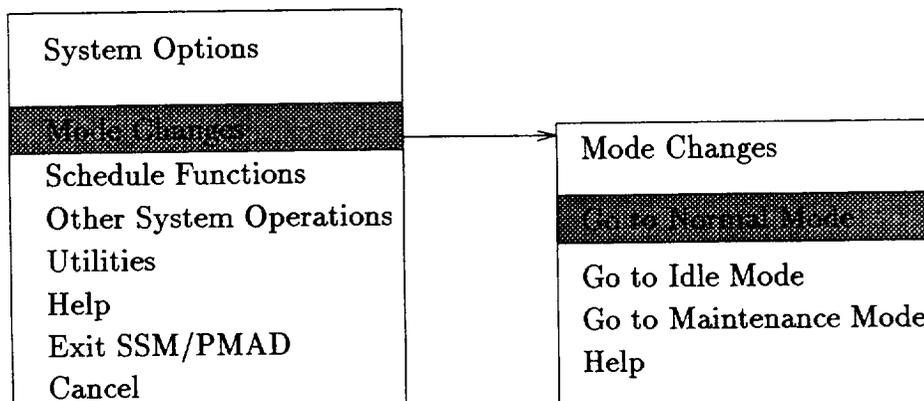


Figure 13: Go to Normal

Although the following description of what happens in normal mode is not necessary for an understanding of the system, it is included for the user who wants a more in depth knowledge of the system. The following discussion may be skipped for the first time reader of this manual. In Normal mode, the user can choose a schedule to read and activate for operating the switches of the breadboard. The user is also prompted to indicate the start of mission time. During the execution of the schedule, the user can manually seize control of a switch to initiate an activity that is not in the schedule. When the user completes the task which uses the switch, control of the switch is returned by manually releasing the switch back to the system.

After the schedule is activated, the user does not need to do anything to operate the system. It will autonomously operate itself. Switches will be turned on and off according to the schedule as prepared by MAESTRO. If a fault occurs, the user interface will reflect the status of the system and perform fault diagnosis to determine which switches may no longer be used because of the fault. The scheduler will then be notified of the new state of the system and will perform contingency scheduling to continue efficient use of the power system. All these actions occur autonomously.

There is not much for a user to do during autonomous operations except to observe the schedule's progress and possibly monitor any interesting switch and sensor data. At any time the user may choose to go back to Maintenance mode by selecting "Go to Maintenance Mode" from Mode Changes under the System Options Menu. If the user does go back to Maintenance mode, the system will remain in the last state in Normal mode. It is up to the user to turn off any "on" switches.

In Normal system operation the FRAMES fault diagnosis software and scheduling soft-

ware are present. Selecting normal mode will cause FRAMES to reinitialize the connected LLPs, and start the fault diagnosis expert system. The software for normal system operation is much more complex due to the necessity of finding out what fault really occurred in the power system switchgear and the need to update the existing schedule so that an efficient use of the power system may be made with the remaining switches.

In terms of actual LLP and switchgear operations, normal mode operations look almost identical to maintenance mode operations. All that is really done is to add a lot of reasoning mechanisms to the upper layers of the *SSM/PMAD* breadboard in order to more intelligently maintain control of the power system without requiring user intervention. During initialization, the following system transactions occur:

1. The Solbourne sends an initialize transaction to each LLP.
2. Each LLP responds with a switch and sensor configuration transaction as well as switch and sensor conversion values.
3. The Solbourne starts the fault diagnosis expert system software.

Normal operations then include the periodic switch and sensor performance transactions sent from the LLPs to the Solbourne. However, in addition to these normal transactions, the following additional transactions may occur:

1. Every five minutes the Solbourne will compute the utilization of power of the breadboard for each switch and for the overall power usage. This utilization data is made available for reference purposes.
2. Periodically, FELES and LPLMS will send updated versions of the schedule and load priority list to the Solbourne.
3. The Solbourne will then distribute the various parts of these lists to the respective LLPs.

As in maintenance mode, the user may query the LLPs for data at any time. When a fault occurs in the power system switchgear, the sequence of transactions and operations is much more involved:

1. When an LLP first detects a fault in the switchgear, it sends a fault transaction to the Solbourne.
2. The Solbourne then sends a quiescent transaction to each LLP.
3. Each LLP responds with a quiescent "is true" transaction when their data is in a stable state.

4. The Solbourne then sends a query for switch status information to each LLP.
5. Each LLP then sends the switch status transaction as requested.
6. The Solbourne then checks to see if the snapshot of the power system it has just collected is stable. If an LLP has had new fault data during the data collection stage, it will set a non-quiescent bit in the switch status transaction. If the snapshot is not stable, the Solbourne will repeat steps 2-5 until a quiescent snapshot is achieved.
7. The Solbourne also collects the symptoms from the snapshot and initiates the fault diagnosis expert system to perform the fault diagnosis.
8. The fault diagnosis expert system may need to manipulate the switches to gain more information about the fault in order to isolate it. This process involves sending a switch control transaction to each of the LLPs with switches that need to be turned on or off.
9. The commanded LLPs will respond with switch and sensor configuration transactions to keep the Solbourne updated as to the state of the power system.
10. The Solbourne will then repeat steps 2-6 each time a fault isolation step occurs.
11. When the fault diagnosis expert system concludes a diagnosis, any switches determined to be unusable are taken out of service.
12. Steps 7-11 are then repeated if there are further faults represented by symptoms not accounted for by the diagnoses so far. When all the current faults are isolated, MAE-STRO is commanded to perform contingency scheduling, taking activities affected by out of service switches off the schedule and possible adding other activities to the schedule.
13. The Solbourne distributes the new schedule and load priority list to the LLPs.

The Maintenance Mode of Operation To put the system into maintenance mode, the operator should select this option. See Figure 14.

The effects of the various mode changes to "maintenance" are:

Idle → Maintenance: This mode change puts the user in Maintenance mode.

Normal → Maintenance: This operation will modify the active schedule so that any Normal activities are stopped. This mode change leaves the system in the current state and lets the user operate it as desired.

When the "Go to Maintenance Mode" is selected,

- On the system message section of the screen, KANT will say "Go to Maintenance".

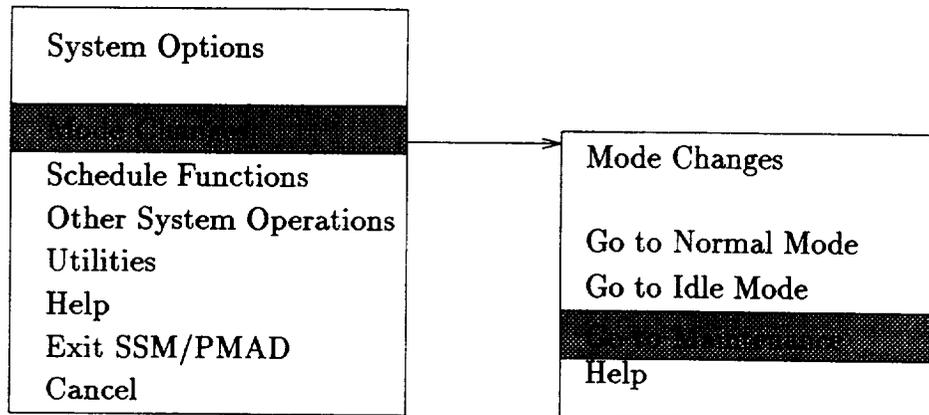


Figure 14: Go to Maintenance

- When the system is in maintenance mode, the Mode window will display: "MAINTENANCE"

Although the following description of what happens in maintenance mode is not necessary for an understanding of the system, it is included for the user who wants a more in depth knowledge of the system. The following discussion may be skipped for the first time reader of this manual. Maintenance mode enables the user to manually operate the switches and observe the operation of the power system switchgear, perhaps for the purpose of calibrating the switchgear as well as testing new components. When the system is in maintenance mode, MAESTRO and the FRAMES diagnostic system are not present. The transactions that occur between the System Operation function and an LLP are as follows during initialization.

1. An initialization message is sent to the LLP.
2. The LLP responds with three messages about its configuration.
 - (a) The switch and sensor configuration of the LLP.
 - (b) The switch conversion values of the LLP used to convert amperage readings from the switch to amps. These values may be adjusted for calibration.
 - (c) The sensor conversion values of the LLP used to convert sensor readings to the appropriate units and values. These values may be adjusted for calibration.
3. The Solbourne also sends a time synchronization message to the LLP so that the LLP may initialize its clock.

After the LLPs have been initialized, the user may command a switch or request data from an LLP. If a switch is commanded the following transactions occur:

1. The Solbourne sends a switch control list to the LLP to command the switch.
2. The LLP sends a switch and sensor configuration transaction back to the Solbourne to reflect the new state of the switches as a result of the command.
3. The LLP also updates its performance data on the switches and sends performance data to the Solbourne.

If a request for data is initiated, the following transactions occur:

1. The Solbourne sends a query for data transaction to the LLP.
2. The LLP responds with a transaction that fulfils the request.

Every five minutes the LLPs update their performance data for switches and sensors. This information is then sent in two transactions to the Solbourne.

1. The LLP sends a switch performance transaction to the Solbourne.
2. The LLP sends a sensor performance transaction to the Solbourne.

Finally, if a hard fault occurs in the power system switchgear, the following transactions occur:

1. The LLP detecting the fault sends a fault transaction to the Solbourne.
2. The Solbourne then sends a quiescent transaction to all the LLPs. This transaction is a request for the LLPs to send a quiescent "is true" transaction back to the Solbourne when their data is quiescent.
3. Each LLP sends a quiescent "is true" transaction to the Solbourne.
4. The Solbourne then sends a request for switch status information to each LLP.
5. Each LLP responds with the switch status transaction.

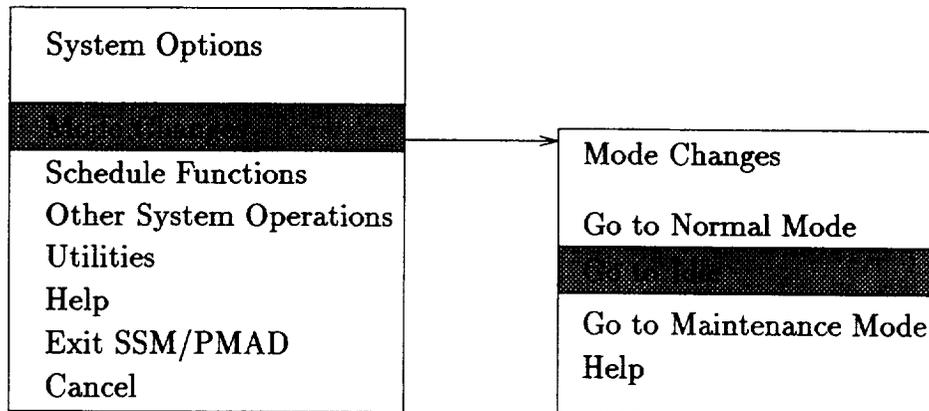


Figure 15: Go to Idle

The Idle Mode When the *SSM/PMAD* is first started, it is in “idle” mode. The user will not be operating a schedule and no connection will be available to any of the lower level processors. This is the mode the *SSM/PMAD* must be in to exit *SSM/PMAD*. To put the system into idle mode, the operator should select this option. See Figure 15.

The effects of the various mode changes to “idle” are:

Maintenance → Idle: This operation will halt the active schedule and leave the user in an Idle Mode.

Normal → Idle: This mode change will turn off all switches that were on by halting the active schedule and leave the user in the Idle mode.

When the “Go to Idle Mode” is selected,

- On the system message section of the screen, KANT will say “Go to Idle”.
- When the system is in idle mode, the Mode window will display: “IDLE”

4.4.2 Schedule Functions

This option brings up a sub-menu of system level scheduling functions.

The idea of an “Active Schedule” is used throughout this section. When the user is in maintenance mode there is no active schedule.

To operate a schedule from normal mode is accomplished in two ways. The general way to do this is to get the system into normal mode and activate a schedule. If it is desired to start a different schedule while in normal mode, it is necessary to halt the active schedule and subsequently activate a new schedule.

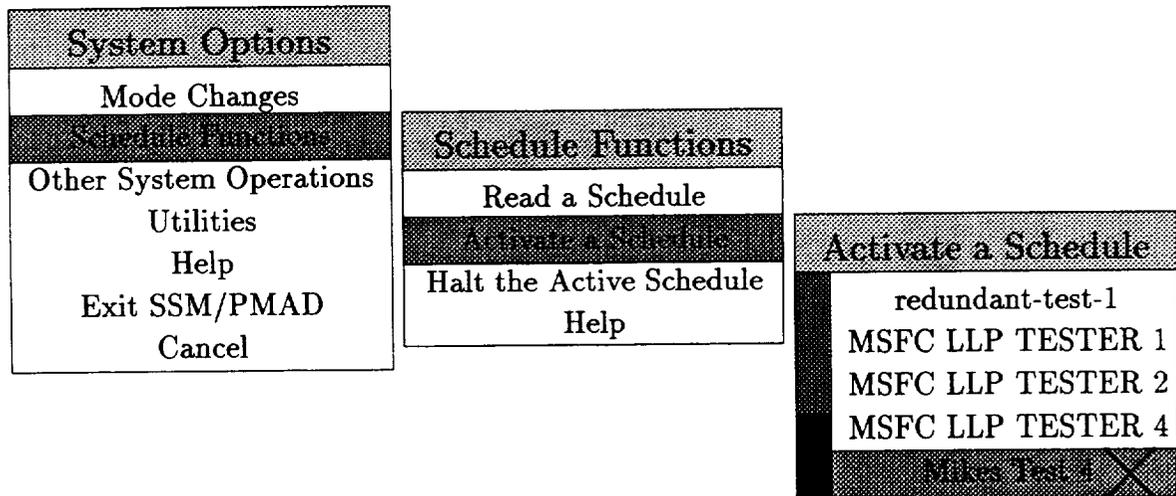


Figure 16: Activate a Schedule

Read A Schedule To “read a schedule” into the KNOMAD database and the *SSM/PMAD* system, this option should be selected. This will allow the operator to select from a menu of available schedules. It will not activate the schedule, but will allow the operator to view the activities on the schedule from the FELES screen. Any number of schedules can be read.

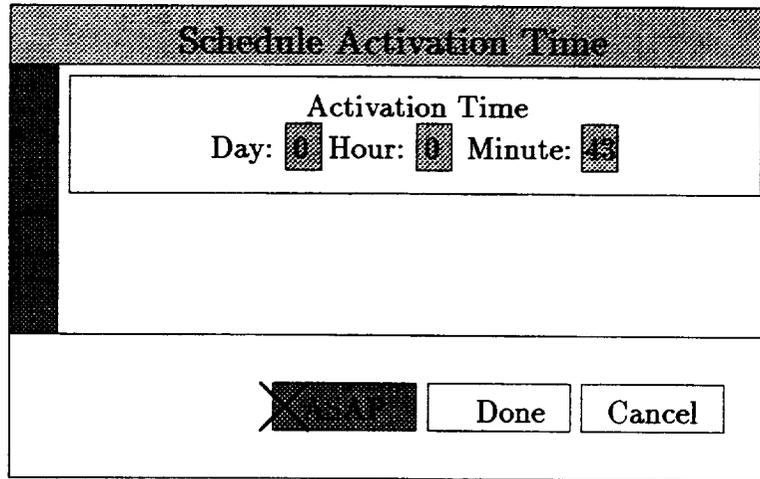
When “Read a Schedule” is selected, on the System Flow window, KANT will say “Read a Schedule”.

Activate a Schedule To “activate a schedule”, the operator should select this option. The activate both reads and activates the schedule. This option will allow the user to select from a menu of available schedules. See Figure 16.

The operator will also be prompted for a time at which the schedule should be put into operation. This time is currently input as a start time in terms of mission time. Activating a schedule requires quite a bit of computation. The system will estimate the amount of time required before it can be put into operation. If the user asks to activate a schedule at a time that the system thinks cannot be met, the user will be notified in workbox window. To activate a schedule as soon as possible, the user should button click the ASAP button instead of entering start time data. See Figure 17.

When “Activate a Schedule” is selected, in the System Flow window of the screen, KANT will say “Activate Schedule”. LPLMS will say “Computing Priority Weightings”, and then FELES will say “Sending Event List”.

If any of the switches on the requested schedule are unavailable, a workbox entitled



Screenshot of the "Schedule Activation Time" dialog box. The title bar reads "Schedule Activation Time". Inside, there is a section titled "Activation Time" with three input fields: "Day: 0", "Hour: 0", and "Minute: 43". Below this section are three buttons: a disabled button (indicated by a diagonal slash), "Done", and "Cancel".

Figure 17: Schedule Activation Time Workbox

"Schedule Activation Confirmation" is displayed with a list of all the activities which will be affected. The user can proceed to activate the schedule by button clicking "Done" or cancel the request by button clicking "Cancel". See Figure 18.

Halt the Active Schedule This option allows the user to halt the currently active schedule. Halting a schedule simply cuts off all scheduled events at a specified halt time.

If the operator is in normal mode and wishes to operate a different schedule than the one currently executing, the operator would first halt the active schedule, and then read and activate a new one.

When halting the active schedule, the operator will be prompted for a mission time at which the schedule should be halted (again, in terms of mission time). If the operator wants to halt the schedule immediately, the operator should button click the ASAP button, instead of entering a halt time. See Figure 19.

When "Halt the Active Schedule" is selected, on the System Flow window of the screen, KANT will say "Halt Active Schedule".

4.4.3 Other System Operations

This option brings up a submenu of "other" system functions:

- Connect to LLP

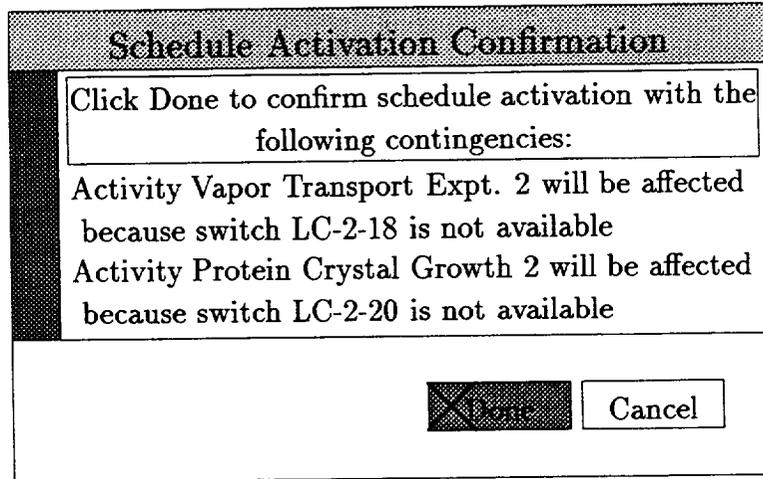


Figure 18: Schedule Contingencies Workbox

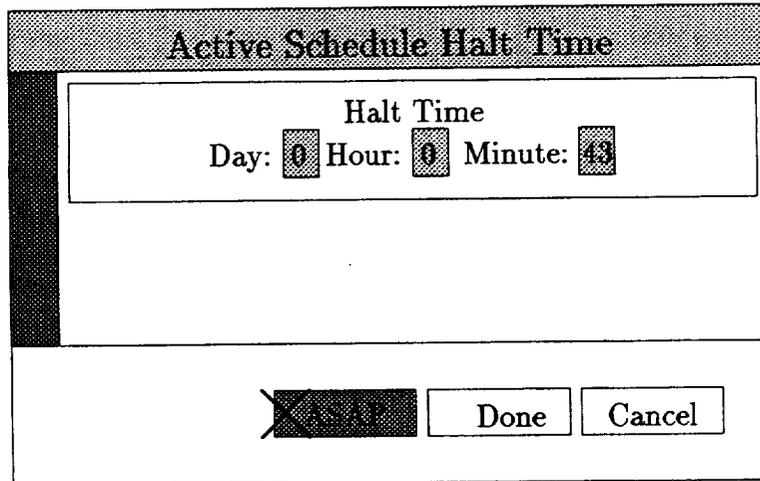


Figure 19: Active Schedule Halt Time Workbox

- Disconnect from LLP
- Connect or Disconnect to Apex
- Source Power Change
- Return RPCs to Service

Connect to LLP This option allows the user to connect an existing TCP connection to a selected LLP. This option is not available in normal mode, only in idle or maintenance mode.

When the system first comes up, this option is used if all the LLPs do not connect when the mode is changed from idle to normal or maintenance mode. If a LLP is not connected, a "nc" will appear in the System Flow section of the screen, and the corresponding graphical representation will not exist on the Power System Application screen.

On the System Options menu, you may select Other System Operations, then select Connect to LLP and choose the load center which is not up, for example, Load Center 2.

After connecting, a "c" will appear first in the System Flow section of the screen, and then the corresponding graphical representation will appear in the Power System Application screen.

The above menu driven method of connecting to a LLP has been replaced with an easier method. Now the user only has to button click the LLPs he wants to activate on the system flow screen.

Disconnect from LLP This option allows the user to disconnect an existing TCP connection to a selected LLP. This option is not available in normal mode, only in idle or maintenance mode.

If LC-1 is connected, for example, a "c" will appear in the System Flow section of the screen, and the corresponding graphical representation will exist on the Power System Application screen.

If Load Center 2 needs repair, on the System Options menu, select Other System Operations, then select Disconnect to LLP and choose Load Center 2. After disconnecting, a "nc" will appear first in the System Flow section of the screen, and then the corresponding graphical representation will disappear in the Power System Application screen.

The above menu driven method of disconnecting to a LLP has been replaced with an easier method. Now the user only has to button click the LLPs he wants to deactivate on the system flow screen.

Connect/Disconnect to Apex This option is not activated at this time, but may have future use.

Source Power Change An integral part of power management and scheduling is allowing for power changes. Under normal operation, the Port bus and the Starboard bus have 12500 Watts of power available on each bus for powering the load centers. It could happen that less than 12,500 Watts per bus is available, and the schedule must be adjusted to accommodate the decrease in power.

When a source power change occurs, the system determines how to reschedule the activities on the current schedule. The priorities associated with activities are used to reschedule the activities. The high priority activities are scheduled first, and the lower priority activities are scheduled next. For example, operating the lights and maintaining the air supply are high priority activities, and will be scheduled before non-essential experiments.

When the source power change code is executed, the algorithm sorts the activities by priority. A high priority activity is scheduled, and its corresponding maximum power is added to the amount of power being used on the bus. The next highest priority activity is scheduled, and its corresponding maximum power is added to the power being used on the bus. This continues until the power limitation is exceeded, and the activity is inserted on a list of activities to be shed or all activities have been scheduled. The system advises the user of the source power change by highlighting in red the message bar that says source power change. Clicking the bar displays information concerning the source power. See Figure 20.

Several underlying system concepts influence the way the source power change works. When source power change is selected, a workbox is displayed and the start time of the source power change is requested. See Figure 21. If more than 5 minutes is allowed for the source power change to take effect, MAESTRO has time to reschedule the activities and execute the change just like any other contingency. However, if less than 5 minutes is allowed for the source power change to take effect, *SSM/PMAD* software handles the load shedding itself. There is not enough time for MAESTRO to handle the contingency.

A second concept that influences the way the source power change works is the effect of groups in the system. If switches are grouped and the operator seizes manual control of the group, the FELES screen shows the manual activities for the grouped switches. These manual activities are also part of the schedule. When the source power change is executed, if one switch in the group causes the power limitation to be exceeded, the entire group is shed, not just the one switch. Shedding the entire group of switches makes sense, because the switches act as a unit. The system is intelligent enough to know the groups in the system, and the switches therein.

A third concept that influences the way the source power change works is the use of redundant switches. Any switch that is a redundant switch, even though it may be drawing zero Watts, must be left "on" because of the redundant possibility on that bus.

One can view the effects of the source power change on the activities in the schedule by viewing the FELES screen. The FELES screen will show the shed activities are halted for a designated duration, and then reactivated at the end of the source power change. The window of time may need to be enlarged to view the entire source power change.

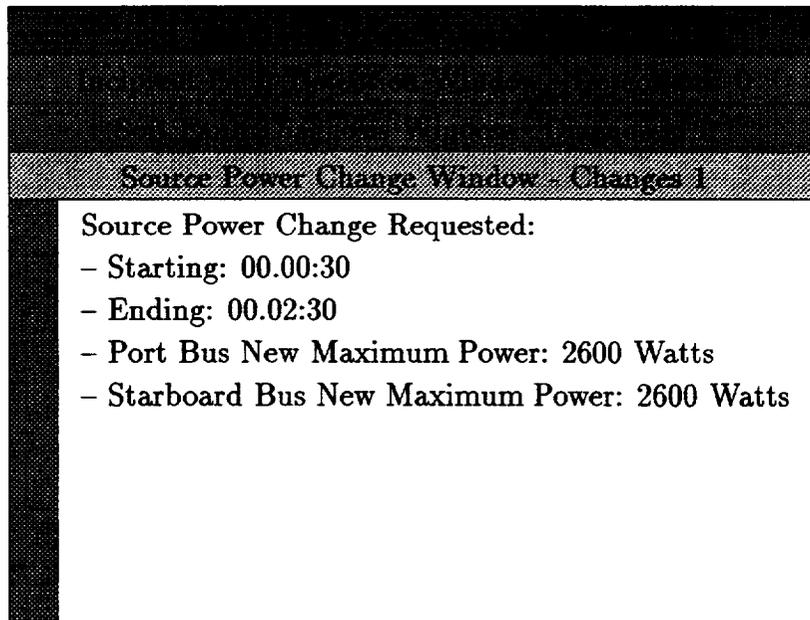


Figure 20: Source Power Change Message Window

Source Power Change					
Start Time			Duration		
Day: <input type="text" value="0"/>	Hour: <input type="text" value="0"/>	Minute: <input type="text" value="30"/>	Day: <input type="text" value="0"/>	Hour: <input type="text" value="2"/>	Minute: <input type="text" value="0"/>
Power Allotment Port Bus: <input type="text" value="2600"/>					
Power Allotment Starboard Bus: <input type="text" value="2600"/>					
			<input type="button" value="Done"/> <input type="button" value="Cancel"/>		

Figure 21: Source Power Change Workbox

Return RPCs to Service After testing a switch or a group of switches, this option is used to return the switches to service.

4.4.4 Utilities

This option brings up a submenu of utility functions accessible from the user interface:

- Clear Scratchpad
- Toggle System Flow Highlighting
- Screendump

Clear Scratchpad This option clears the current application screen scratchpad window.

Disable System Flow Highlighting This option allows the user to toggle the System Flow highlighting feature of the user interface. When highlighting is enabled, elements of the System Flow window will be highlighted as their state changes from an "idle" state.

Disable Garbage Collection Cursor

Screendump This option allows the user to perform a screen dump of the user interface. The resulting screen dump will appear in the “ssmpmad-if-archive” subdirectory of the user’s directory (i.e. “/ssmpmad-if-archive/”). The file will be a Sun rasterfile, compressed and labeled with the current date and time, e.g. “Screen-mm-dd-yy-hh-mm-ss.Z”.

4.4.5 Help

This option brings up a description of the user interface and its component windows as well as the screen layouts.

4.4.6 Exit *SSM/PMAD*

To Exit *SSM/PMAD*, go to idle mode, and then select the Exit *SSM/PMAD* option. Idle → Exit *SSM/PMAD* This mode change exits the *SSM/PMAD* system, and returns the system to the machine prompt.

This option is used to completely exit the *SSM/PMAD* software. LISP will be terminated and the operator will be left at the prompt from which the *SSM/PMAD* system was started. This option has an “Are you sure?” submenu to prevent a user from accidentally exiting.

4.4.7 Cancel

This option cancels the request for the system options menu to be displayed.

4.5 The Power System Application Menu

The Power system screen contains a topological map of the power system. The application screen presents power buses, switches, and sensors as representative graphics and icons. See Figure 22.

The Power System screen was designed to be the primary application on the display during operation of the *SSM/PMAD*. Because of this, its application menu contains options with limited capabilities of some of the other applications. When a user observes something interesting on the Power System screen, ancillary information may be obtained using the options in the application menu. If the user still has not obtained enough information on the event observed, switching to a more appropriate application is still an option for more information.

4.5.1 Power System Topology

The *SSM/PMAD* autonomous power system contains two power buses, port and starboard, and each feeds a Power Distribution Control Unit (PDCU). Both PDCUs distribute power to each load center and the interface is capable of displaying up to five load centers. Up to

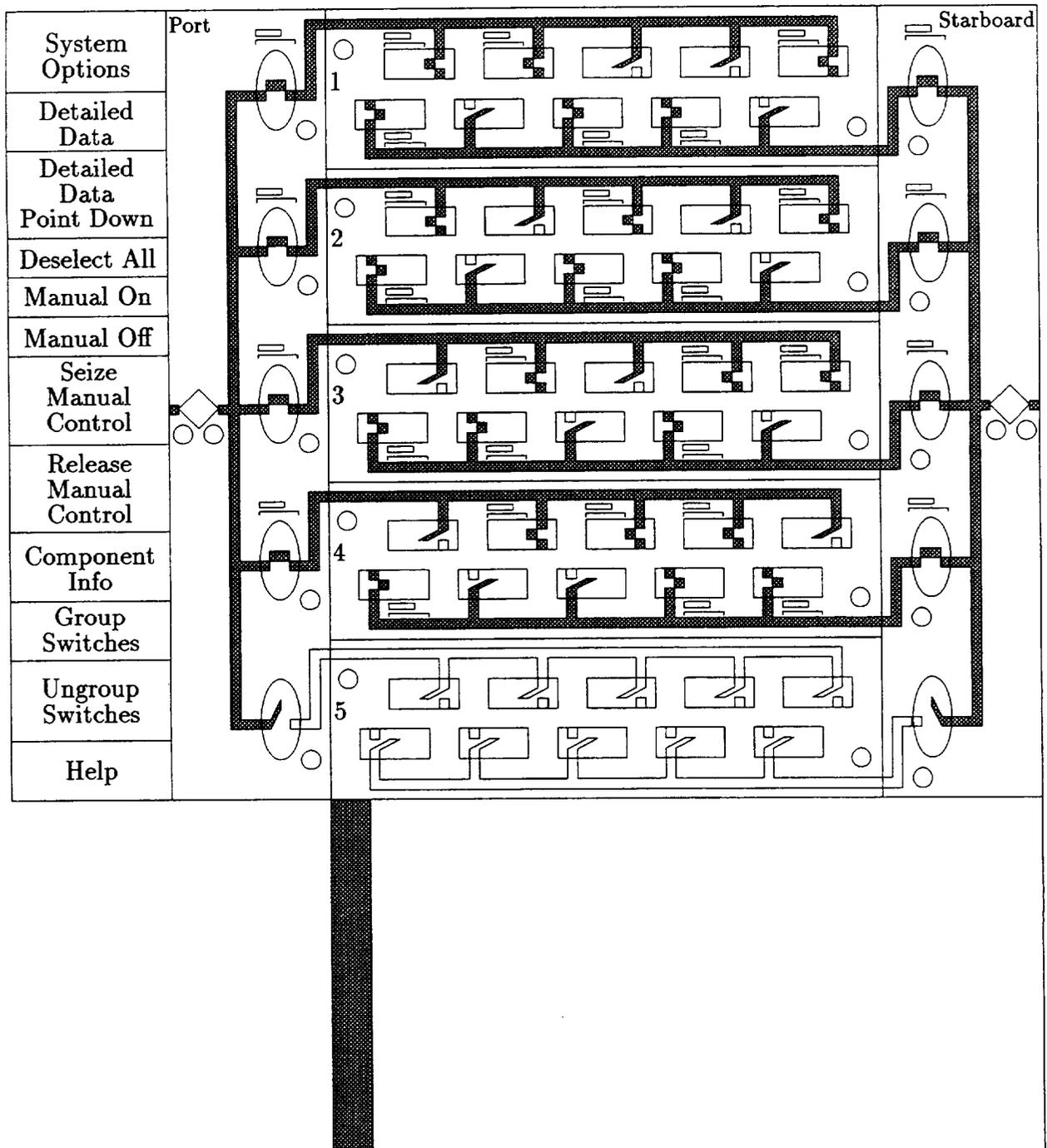


Figure 22: Power System Screen

five switches may be displayed per bus per load center on the interface. Redundant sourcing of an individual load is only permitted within a load center and is displayed that way at the user interface.

4.5.2 Power System Representation

See Figure 22 for a depiction of the following aspects of the power system screen. Graphical pipes represent the power buses on the interface. A filled graphical pipe represents an energized bus, and an empty pipe represents a non-energized bus. Every switch icon contains a multiplicity of information, including its type, state, status, amount of current flowing through the switch, scheduled current to a redundantly sourced load, and whether it has been selected for interface operations. Each switch looks like a relay made with graphical pipes which overlays a geometric shape. The geometric shapes indicate the type of switch, a diamond indicates a Remote Bus Isolator (RBI), an oval indicates a three kilowatt Remote Power Controller (RPC), and a rectangle indicates a one kilowatt RPC. The color of the geometric shape expresses the state of the switch, with green being normal, red being faulted, brown being out of service and off-green being unavailable.

Each closed switch icon contains an analog bar which represents the current flowing through the RPC versus maximum current capacity. This analog bar also has a schedule current indicator for the switch. The switch label and current flowing through the switch are displayed on the geometric background.

If the switch is tripped or out of service, an abbreviation for the fault type which caused the problem is displayed in place of the current.

If the switch has been placed under manual control, a hand icon points to the switch and partially overlays the switch.

When a switch supplies current to a redundantly sourced load, a redundant icon will appear below the switch and its redundant partner within a load center. The primary switch in a redundant pair will have a redundant icon with black foreground and tan background. The secondary switch in a redundant pair will have the same icon pattern, but a black background and a tan foreground.

If the switch has been selected for interface operations, a checkmark icon will partially overlay the switch.

Hollow circles represent sensors on the interface. Sensors may be selected with a button click in the same manner as switches. When a sensor is selected, the sensor is overlaid with a selection checkmark.

The following Power System menu options are discussed below:

- Detailed Data/Detailed Data Point Down
- Deselect All

```
Time: 00.01:11 LC-1-5 State CLOSED Current 3.3 AMPS
0 1000 Fault Flags -> FT FALSE OC FALSE UV FALSE OT FALSE Tag 0
```

Figure 23: Power System Detailed Data

- Manual On
- Manual Off
- Seize Manual Control
- Release Manual Control
- Component Information
- Group Switches
- Upgroup Switches

4.5.3 Power System Application Options

Detailed Data/Detailed Data Point Down The function Detailed Data will return a more complete set of switch data to the user for each switch on the interface which has been selected. Selection is denoted with a check mark. To select a switch or sensor on the Power System screen, move the mouse cursor over the switch or sensor object and click a button on the mouse.

For switches, the user will retrieve information which includes the switch's state, the current going through it, if it is tripped (and how).

If LC-1-05 were selected, the time, load center, status of the switch, current going through it, and fault flags are displayed for LC-1-05 in the scratchpad area. See Figure 23.

The user may similarly get information from sensors. For sensors the user will be able to monitor the current and voltage of the sensor.

The function Detailed Data Point Down returns the same information as "Detailed Data:", but it will return data for every switch selected and every switch hierarchically below every selected switch within the power system topology.

Deselect All This function will deselect all presently selected switches indicated by check marks on the Power System interface screen. All check marks will be cleared.

<p>RPC: LC-1-05 is being used by activity Exobiology Data Acquisition From 00.01:00 minutes to 00.03:00 minutes With a maximum of 550.0 W and a minimum of 0.0 W Load does not have redundant power Frames has permission to test load in fault isolation</p>

Figure 24: Power System Load Information

Manual On/Manual Off The Manual On function allows the user to manually turn on a switch within the Power System. This function is not available when the system is in normal mode.

The Manual Off function allows the user to manually turn off a switch within the Power System. This function is not available when the system is in normal mode.

Seize Manual Control The Seize Manual Control function allows the user to manually seize control of a switch and execute tests on the switch.

Release Manual Control The Release Manual Control function allows the user to manually release control of a switch back to the system.

Component Information The function Component Information returns information about a load presently assigned to a selected switch. If a switch is open there is no load assigned to it. If the switch is faulted or out of service, there may or may not be a load assigned to it. And if the switch is closed, there is definitely a load assigned to it. The load information returned is taken from the schedule.

In Figure 24, load information for LC-1-05 is displayed in the scratchpad area. Load information includes the activity using the RPC, the time period of usage, the minimum and maximum power, redundancy information, and whether FRAMES has permission to test the load in fault isolation.

The function Load Information Point Down returns the same information as "Load Information", but it will also return the load information for every switch hierarchically below the selected switches within the Power System.

Group Switches/Ungroup Switches

4.6 The Power Utilization Application Menu

The Power Utilization application screen displays plots of system power usage and system power availability versus time. The power usage plots contain a plot of scheduled power usage versus time which is overlaid by the actual system power usage. See Figure 25.

The following Power Utilization screen options are discussed below:

- Change Window of Time
- System Power Usage (default)
- Load Center Power Usage
- Component Power Usage
- System Power Availability
- Load Center Power Availability

4.6.1 Change Window of Time

This function will change the time axis of all the plots presently being viewed. Two hours is the default. This time can be increased or decreased. In Figure 26, the Change Window of Time work box is displayed for the Power Utilization screen.

4.6.2 System Power Usage

This function will cause the Power Utilization screen to display the power usage in the system with actual power used plotted over scheduled power. The data is displayed in two columns on the screen with load centers displayed by bus from top to bottom. The Port bus of a given load center is displayed in the left column and the Starboard bus is in the right column. The bottom row of plots contain the overall Port and Starboard bus power usage.

In Figure 25, in the left column, Power Utilization for LC-1 Port, LC-2 Port, LC-3 Port, LC-4 Port, and LC-5 Port is displayed. In the right column, Power Utilization for LC-1 Starboard, LC-2 Starboard, LC-3 Starboard, LC-4, and LC-5 Starboard is displayed.

4.6.3 Load Center Power Usage

This function prompts the user for the Load Center to display. For a given load center, actual power usage is plotted over scheduled power for all the given RPCs. The data is displayed in two columns on the screen with Port connections on the left and Starboard connections on the right. The bottom row of plots contain the overall load center's Port power usage and the overall Load Center's Starboard power usage.

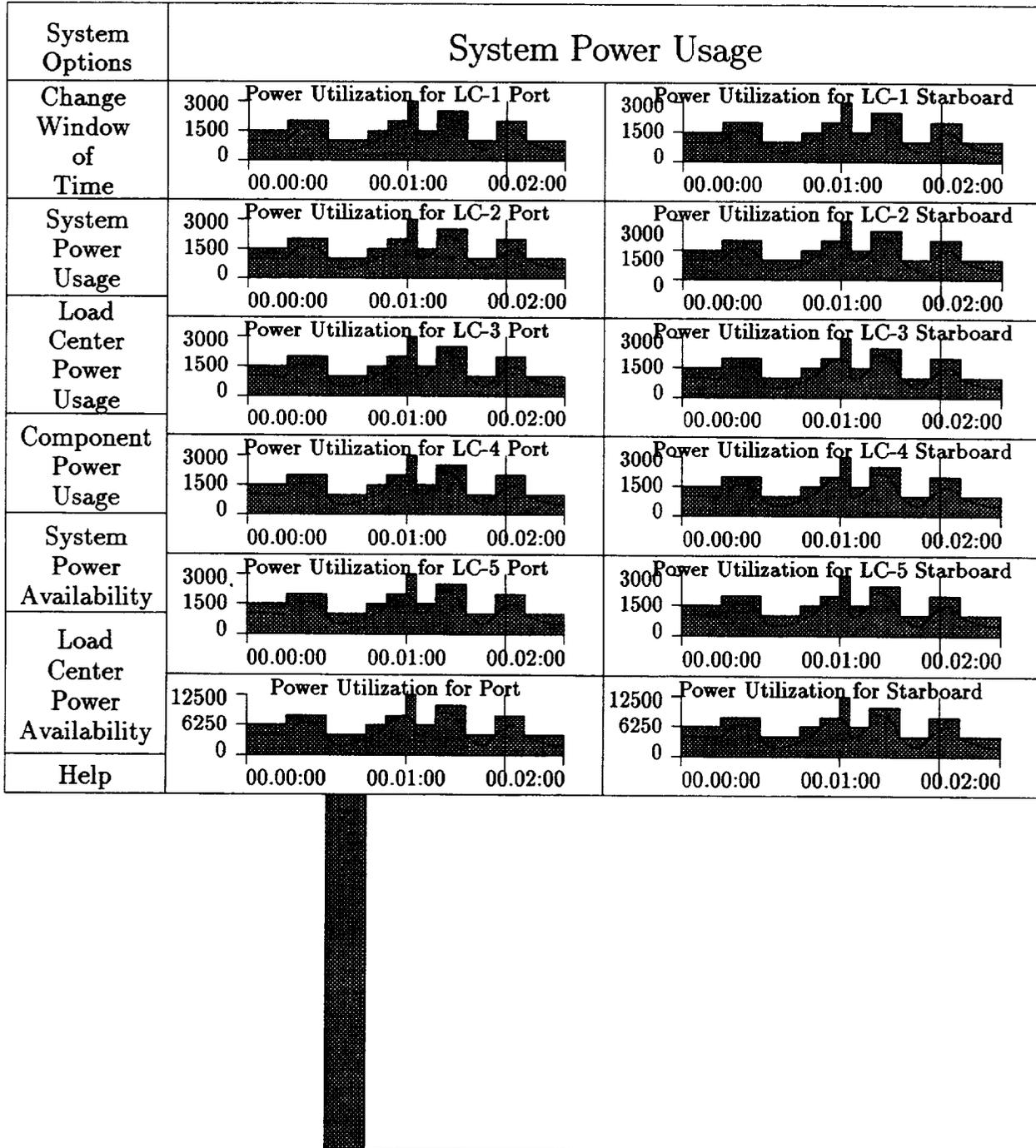


Figure 25: Power Utilization Screen

Start of Window of Time			Duration of Window of Time				
Day:	<input type="text" value="0"/>	Hour: <input type="text" value="0"/>	Minute: <input type="text" value="30"/>	Day:	<input type="text" value="0"/>	Hour: <input type="text" value="3"/>	Minute: <input type="text" value="0"/>
<input type="button" value="Done"/> <input type="button" value="Cancel"/>							

Figure 26: Power Utilization Change Window of Time

4.6.4 Component Power Usage

This function will prompt the user for the load center and RPC to display. The selected RPC plot will be displayed in the scratchpad. See Figure 27.

4.6.5 System Power Availability

This function will display the power available for scheduling in the system. If the graphs for system power usage and system power available were combined, maximum power available would be the result. The data will be displayed by load center and bus. The Port load centers are displayed on the left and the starboard switches are displayed on the right. The bottom row of plots contain the overall Port and Starboard power availability.

4.6.6 Load Center Power Availability

This function prompts the user for the load center to display. This function will display the power available for scheduling in the system. If the graphs for load center power usage and load center power available were combined, maximum power available would be the result. For a given load center, actual power available is plotted for all the given RPCs. The data is displayed in two columns on the screen with Port connections on the left and Starboard

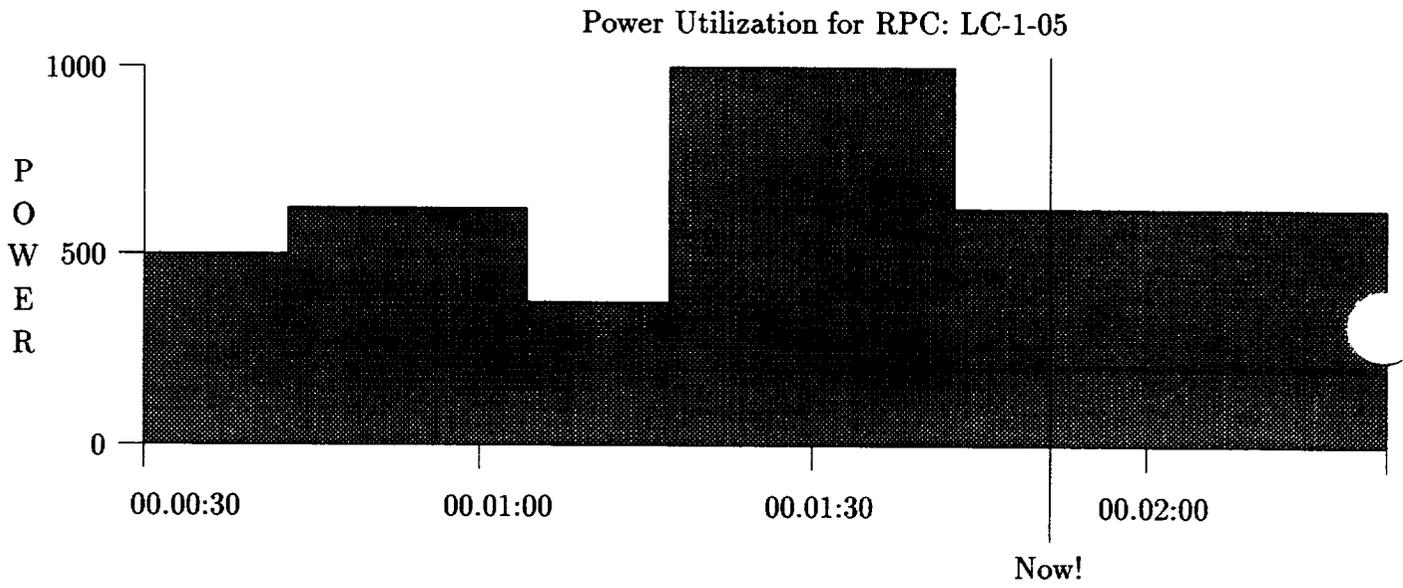


Figure 27: Power Utilization Component Power Usage

connections on the right. The bottom row of plots contain the overall load center's Port power availability and the overall load center's Starboard power availability.

4.7 The FELES Application Menu

The Front End Loads Enable Scheduler (FELES) application screen allows the user to display various views of the active schedule and get information on selected items. The default view is a Gantt Chart representation of the active schedule by activity. Information about an activity may be obtained by selecting an activity and asking for load information. See Figure 28.

Activities are displayed with respect to time in a scrollable screen using "View Schedule by Activity". The small tick marks below the bar indicate different performances within an activity.

The following FELES screen options are discussed below:

- View Active Schedule (Default)
- View Alternative Schedule
- Change Window of Time
- View Schedule by Activity (Default)
- View Schedule by Load Center
- Activity Info
- Load Info

4.7.1 Change Window of Time

This option will allow the user to display the Gantt Chart with a different time interval. The default window is 2 hours. The user will be prompted for the beginning and ending times. When the new time view is entered, the Gantt Chart will be redisplayed accordingly.

In Figure 28, the window of time is set to the two hours default and in the "Change Window of Time", three hours is requested. See Figure 29.

4.7.2 View Schedule by Activity

This is the default view, and when this view is active is not selectable. If this view is selected, the schedule is displayed in terms of activities. In Figure 28, the activities for the active schedule are shown. The display is scrollable.

System Options	Schedule: Test Schedule					
View Active Schedule	Cabin Air Supply					
	Fire Safety System					
View Alternative Schedule	Main Cabin Lights					
	Avionics					
Change Window of Time	Biotechnology Facility 1					
	Exobiology Data Acquisition					
View Schedule by Activity	Glovebox Housekeeping					
	Life Sciences Glovebox					
View Schedule by Load Center	Microbiology Monitor 1					
	Microbiology Monitor 2					
Activity Info	Modular Combustion Facility 2					
Load Info	00.00:00	00.00:30	00.01:00	00.01:30	00.02:00	00.02:30
Help				Now!		

Figure 28: FELES Screen

Start of Window of Time			Duration of Window of Time								
Day:	<input type="text" value="0"/>	Hour:	<input type="text" value="0"/>	Minute:	<input type="text" value="30"/>	Day:	<input type="text" value="0"/>	Hour:	<input type="text" value="3"/>	Minute:	<input type="text" value="0"/>
<input type="button" value="Done"/> <input type="button" value="Cancel"/>											

Figure 29: FELES Change Window of Time

4.7.3 View Schedule by Load Center

This option allows the user to view the schedule in terms of the switches of any particular load center. The user will be prompted for the particular load center. In Figure 30, Load Center 1 was selected. All the RPCs for Load Center 1 are shown.

4.7.4 Activity Info

This option will display information about the selected activities, what activity it is, its subtasks, equipment being used, etc.

In Figure 31, the current time period for RPC 05 is selected for display. Past and future times could also be selected. The load information displayed includes the times, the activity, the subtask, the powered equipment used, the maximum and minimum power, redundant switch status, and permission to test. The load information is given for all all subtasks active for the given time: Initialize, Collect, etc.

4.7.5 Load Info

This option allows the user to display information about what loads (powered equipment) are using the selected switches.

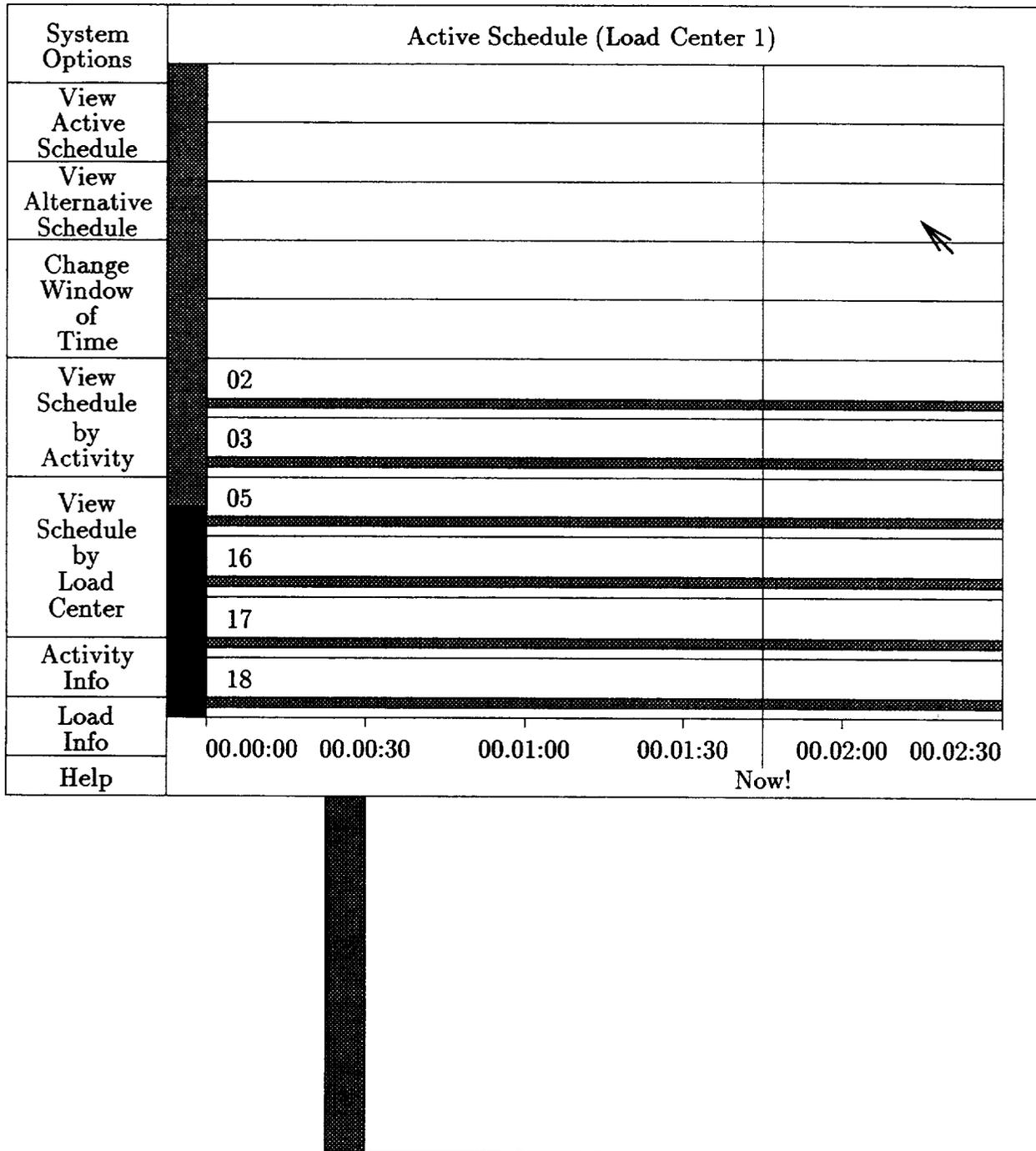


Figure 30: FELES View Schedule by Load Center

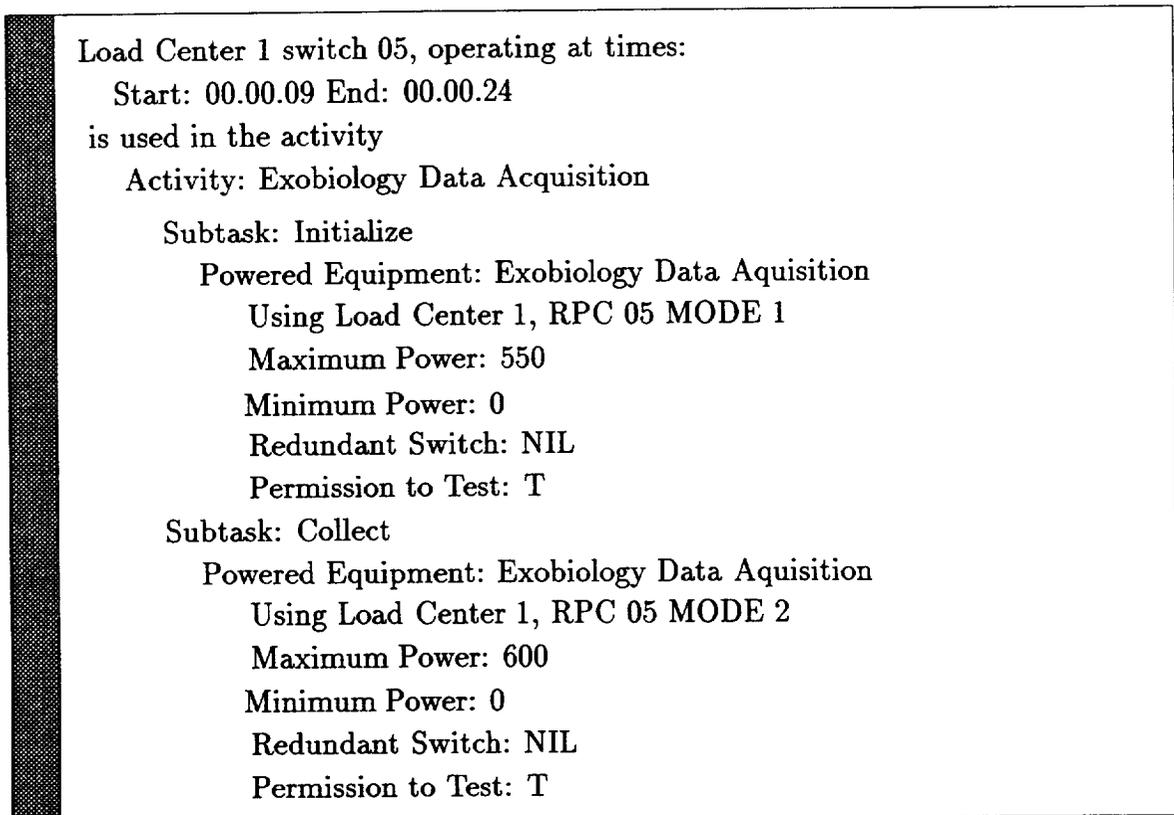


Figure 31: FELES Activity Info

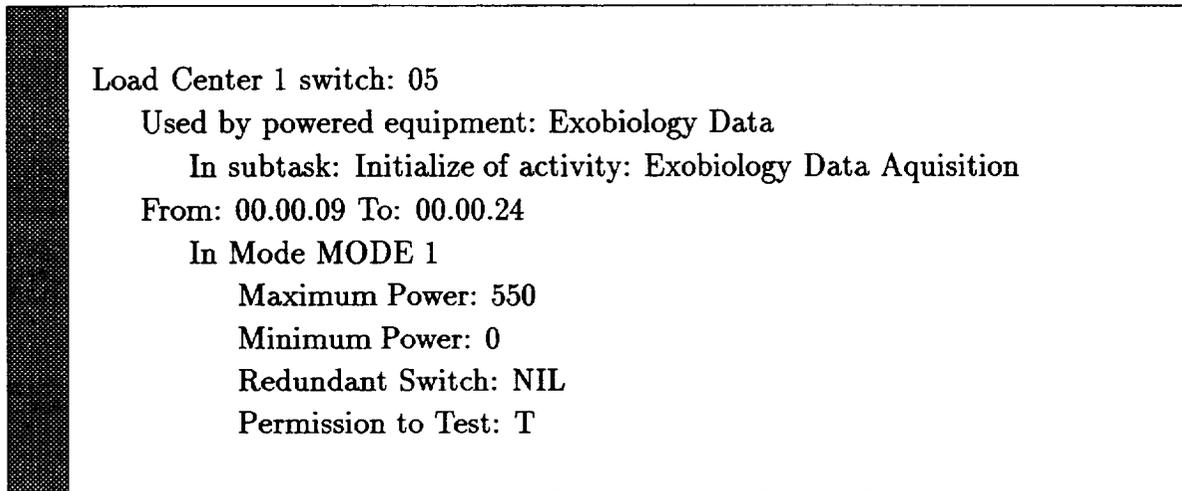


Figure 32: FELES Load Info

In Figure 32, RPC 05 of Load Center 1 is used by Exobiology Data Acquisition for the requested time period. Maximum power, minimum power, redundant switch, and permission to test information are provided for the single subtask active at the current time: "Initialize".

4.8 The Activity Editor

The user of the *SSM/PMAD* system will want to create and modify schedules that run on the *SSM/PMAD* testbed. The activity editor provides this capability and is the focus of this section of the user manual. Subsection one discusses the purpose for an activity editor, and the reason it was installed on the Solbourne. Subsection two defines an activity and discusses aspects of an activity. Subsection three discusses the MAESTRO requirements for an activity. Subsection four discusses the implementation of the activity editor in MAESTRO and *SSM/PMAD*.

4.8.1 Background and Purpose

The ability to create new activities became increasingly more important as the *SSM/PMAD* testbed evolved. Modifying existing schedules to incorporate changes that model the Space Station more accurately was equally important. The creation and modification of activities allow the user to generate and demonstrate Space Station mission scenarios using the *SSM/PMAD* testbed.

Intermediate Levels of Autonomy (ILA) must effectively manage complex user input activities with a minimal impact to the system. To facilitate interactive planning provided within KANT, the activity editor builds activities and modifies functions of the ILA.

To support the activity editor and its purpose, MAESTRO was ported to the Solbourne. Common data structures were created for MAESTRO and the rest of the system to share. These data structures allow changes made by the activity editor to be seen immediately by the scheduler without having to read the scheduling files into memory.

Future development includes moving the database files to "ingres", an SQL database. This database will use the common data structure for storage, and give the system more flexibility. For example, currently the whole database needs to be saved when a change is made to any element of the data. With the proposed design change, data will be incrementally written to the database.

4.8.2 Definition of Activity

The activity editor creates and modifies activities. Each activity accomplishes a goal using a set of resources and a set of actions which execute in some sequence. MAESTRO defines an object called a subtask for contiguous actions on a uniform set of resources. This subtask may require that a particular environment state be maintained and may be dependent on the execution of other subtasks. An activity is thus defined as a linear, non-overlapping sequence of one or more subtasks.

Scheduling will specify the start and end times for the subtasks of activities. A valid schedule will be one with start and end times for subtasks such that all activities may be successfully executed by the scheduler.

The user may want an activity performed more than once. This is typically the case with experiments in space, where the cost of getting equipment in orbit is so expensive that a single execution of an activity is not cost-effective. MAESTRO has the ability to schedule multiple performances of activities.

An activity group in MAESTRO contains multiple activities which accomplish the same goal. An activity group in MAESTRO has a primary model and zero or more alternative models for accomplishing the same goal. To reduce complexity, a design decision was made to choose only the primary model in the initial design of the activity editor.

One of the goals of the activity editor interface is to minimize input without curtailing the ability to create complex activities. To minimize input, defaults are used for some of the subtask fields. Examples of defaults are given in section 4.9.3, a subsection on the activity editor.

4.8.3 What MAESTRO Needs

MAESTRO needs certain pieces of information in order to effectively schedule activities for the user. It assumes that with these pieces of information the placement of activities on a schedule can be calculated without trial and error. This is called "opportunity calculation". MAESTRO performs an opportunity calculation via a two step process.

- Resource Opportunity Calculation schedules subtasks with respect to resource and environmental constraints.
- Temporal Constraint Propagation schedules subtasks with respect to timing constraints with some subtasks running before other subtasks.

For the resource opportunity calculation, the user can enter resource and environmental constraint information into an activity. This is done by adding powered equipment and requirements for non-powered resources to accomplish the activity.

Since powered equipment is the most important resource that we schedule on the *SSM/PMAD* testbed, the majority of all subtasks that are created have at least one piece of powered equipment. Because of the primary need for scheduling power, powered equipment was broken out from the other resources into its own group.

The resources that fall under the requirements category are rate controlled, equipment, consumable, condition, and alterable condition. Currently, rate controlled and equipment are the only resources that can be created using the activity editor. Condition and alterable conditions are ways of modeling the user's environment such as day and night. Condition and alterable conditions have not been incorporated yet because they have not been used, and their complexity of creation did not warrant inclusion into the current activity editor. The consumable resource will be incorporated in the activity editor when time permits.

It should be noted that there is only a very crude way to represent environmental constraints in the *SSM/PMAD* system, and the activity editor does not currently have the ability to enter this information.

The second part of the opportunity calculation is the temporal constraint propagation. The only timing constraints currently implemented are delays between subtasks and the duration of the subtask. An attribute field called temporal constraint has been added to activities which allows the user to specify temporal constraints between activities. For example, a time dependency between activity FOO and activity BAR can be created by declaring a time dependency of ten minutes. This means that activity FOO must run within ten minutes of activity BAR. Initially, the time dependency field is set to nil in all activities, because time dependency is a relatively new feature to MAESTRO and its usefulness has not been fully determined. The user is able to do the same thing with a subtask within an activity, so the concept is being abstracted to another area of the system.

4.8.4 What are the *SSM/PMAD* Requirements

From the previous section, it is apparent that an activity can be quite complex. The main goal of the activity editor was to try to give the user an easy way to create and update schedules that are run on the *SSM/PMAD* testbed. In order to do this, the activity editor must meet three basic criteria:

- Minimize the input
- Layer the data
- Reuse the basic building blocks

Minimizing input is achieved by allowing the user to have access only to those characteristics of an activity that are associated with scheduling power. This meant that a lot of activity information that MAESTRO could use was either set to a default values or omitted altogether.

Layering the data is a natural extension of the current activity editor data structure. A method called expand/unexpand has been devised to allow the user to see the different layers that are requested. This allows the user to view the data in a very broad sense, for example, the subtasks within an activity. It also allows the user to view the data in a very narrow sense, for example, what are the mode parameters for a piece of powered equipment. Expand and Unexpand are discussed in greater detail in section 4.9.3.

Reusing the basic building blocks is harder to implement because the idea of reusing subtasks is not a part of the MAESTRO scheduling system. There are four basic types of data that can be re-used to build a schedule.

- Powered equipment whose use is essential in building subtasks.
- Resources whose use is essential in building subtasks.
- Subtasks which are the essential part of an activity.
- Activities which are used in building a schedule.

To reuse any of the above data items, new fields had to be added to the current MAESTRO data structures. The new fields allow the user to keep track of who is using what and where and guarantee the integrity of the schedules. An example would be a subtask that is used in two different activities and those activities are used in five different schedules. If the user were to edit this subtask and change some of the information, this would change the meaning of the two activities which might consequently invalidate all five schedules. This information must be maintained along with the activity in order to inform the user of any conflicts and resolve them before they crash the system.

Implementing the above three criteria simplifies the creation of a schedule by the user. The new method of creating schedules is much simpler than the old method of creating and maintaining schedules that existed on the MAESTRO system. Considerable effort was made to meet the needs of the power system and at the same time utilize the functionality that exists in MAESTRO.

The activity editor is still in a development stage. Therefore we encourage user feedback in order to determine how the functionality of the activity editor can be improved. A future consideration might be to add the power functionality that MAESTRO has for scheduling. At the current time, the user chooses the activities they would like to have scheduled and the time interval. The latest MAESTRO tools allow placement of a chosen activity onto the schedule and then scheduling around the placement along with many other features.

4.8.5 Window Layout

To get to the activity editor, the user selects the "Activity Editor" menu item from the available applications. The Activity Editor is only available during idle and maintenance mode. Because the same data structures are used by MAESTRO and *SSM/PMAD*, allowing the user to edit during normal mode could denigrate the integrity of the system. When the activity editor is first invoked, it will look like Figure 33.

Like all other applications, the activity editor has an applications menu and a scratchpad window. The scratchpad window provides the user with textual data that is either user requested information or activity editor error information. The application menu contains the main functions for the activity editor. These function are:

- Edit
- Create
- Delete
- Save
- Create Schedule
- Delete Schedule
- Help

The implementation of each function is discussed in subsequent sections of this manual. The main window of the activity editor screen is broken down into four parts.

- The first part contains the title "Activity Editor" and is only information.

System Options	Activity Editor	
Edit	Description: <input type="text" value="The operation of the pumps to supply cabin air."/>	
Create	Name: <input type="text" value="Cabin Air Supply"/>	
Delete	Base Priority: <input type="text" value="0"/>	
Save	Performances Requested: <input type="text" value="100"/>	
Create Schedule	Minimum Performances Needed: <input type="text" value="1"/>	
Delete Schedule	Subtask: Main Cabin Air Supply 1	
Help	Max Duration: <input type="text" value="129600"/>	
	Min Duration: <input type="text" value=""/>	
	Maximum Delay: <input type="text" value="0"/>	
	Minimum Delay: <input type="text" value="0"/>	
	Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/>	
	Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input checked="" type="radio"/> Location: <input type="text" value="LC-3-4"/>	
	Powered Equipment: <input type="text" value="*****Undefined*****"/> Mode: <input checked="" type="radio"/> Location: <input type="text" value="*****"/>	

Figure 33: Activity Editor Window

Left, Middle, or Right Mouse Button to Edit This Field

Figure 34: Activity Editor Status Line Category I

Left, Middle, or Right Mouse Button NOT VALID

Figure 35: Activity Editor Status Line Category II

- The second part is the main window used by the activity editor. It is a scrolling window that is used to display the data that the user can edit.
- The third part is a window that displays the status information about the mouse functions that are available for the particular fields within the main window. An example of the status line can be found in Figure 34. When the mouse is located within a data field, the status line shows the user what each mouse button will do for that field. This status line appears when the description field is entered. Figure 35 appears when the name field is entered. Not valid indicates this is not an editable field. Figure 36 appears when the subtask field is entered. All data fields will have one of the three different status line shown in Figure 34 through Figure 36.

All data displayed in the main window of the activity editor screen comes in the same general format. First comes the title for the data followed by the data itself. The bounds of the data is discriminated using a box that has a blue background color. A feature known as highlighting is used to let the user know when the mouse has entered the editable region of a data field. The highlight causes the background color of the box to turn red. When the mouse leaves the editable region, the background color will return to its red color. At the same time as the background color of the box is turning red, the status line is being updated

Left: Edit, Middle: Expand/Unexpand, Right: Menu Options

Figure 36: Activity Editor Status Line Category III

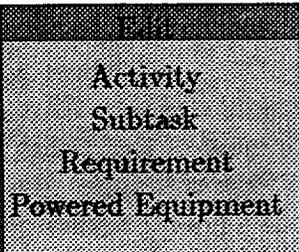
System Options	Activity Editor	
Edit		
Create		
Delete		

Figure 37: Activity Editor Edit Menu

to reflect the editing information for that editable region.

4.8.6 Viewing and Editing the Data at Different Levels

To understand what editing capabilities the activity editor has, the user chooses data to edit. To do this the user selects the Edit option on the applications menu. This will cause a menu to appear in the upper left hand corner of the application main window. Refer to Figure 37 to see an example of this menu.

Notice that there are four entries in the menu:

- Activity
- Subtask
- Requirements
- Powered Equipment

From an earlier discussion, the user will remember that requirements and powered equipment are resource building blocks used in the creation of subtasks, which are used to build activities, which are used to build schedules. The activity editor gives the user the ability to edit each item directly or as a part of an encompassing item. For example, if there is a subtask called FOO which is used in activity BAR, the user can edit the subtask FOO as an

independent object or as part of the activity BAR. The user cannot edit a schedule directly, only the components that make up a schedule. This is why schedule is not a menu item in the edit window.

Each item in the edit menu has a pull-right menu associated with it. So the user first chooses the category to edit, like activity, and then moves the mouse onto it. This will cause the pull-right to appear and the user uses any of the three mouse buttons to select that item. The item in this case is an activity that the user would like to edit. Some pull-right menus are scrollable in order to accommodate the large number of items that must fit into a fixed size menu. Once the selection has been made, the item will appear in the main window of the activity editor. An example of this is in Figure 38 where the activity "Cabin Air Supply" has been selected for editing. The activity is shown first in order to show the layering in the system from activity to resource.

Now that the item of interest is displayed in the activity editor, it is time to begin updating it. All the data fields fall into basically three categories.

- The first category is a data field that can only be changed directly from keyboard inputs. See Figure 34. All three mouse buttons edit the field. The user selects the data field with any of the mouse buttons and a cursor will appear at the beginning of the data field. The user then uses standard emacs commands for cursor movements to move the cursor around in the data field.
- The second category is for the data fields that are not editable. See Figure 35. All mouse buttons are invalid. Since the user cannot update this field, any attempt to select the field will be ignored.
- The third category is the most complex. Each of the mouse buttons performs a different function. See Figure 36. The left mouse button allows the user to edit the field directly. The right mouse button brings up a menu of options that the user can choose from to update this field. Later in this manual, we will cover the menu options in detail when we discuss each data field. The middle mouse button performs the expand/unexpand of the data field.

The expansion of a data field shows the rest of the information that makes up that data field. The middle mouse button provides both the function of expanding and unexpanding the data object based on the current state of the data object. For example, if the data object is already expanded, then the middle mouse button will cause an unexpand of that data object.

Once the data object is expanded, the user will see the data indented just below the data field that was expanded. Figure 33 shows a subtask that has been expanded within an activity. Indentations are used to show levels of expansion. Figure 39 shows a good example of nested expansion. First the subtask is expanded, then a piece of powered equipment within

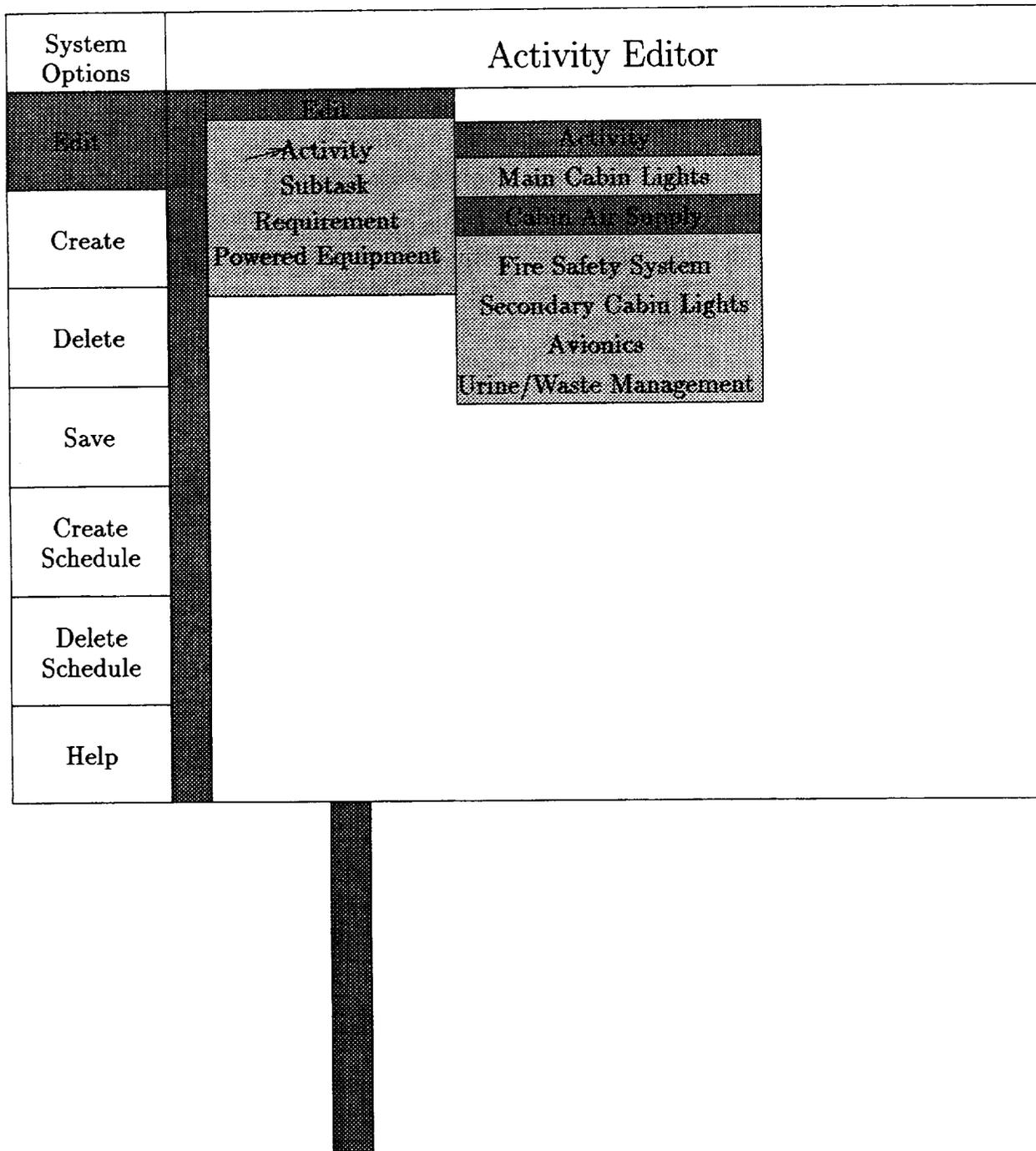


Figure 38: Activity Editor Activity Pull Right

that subtask, and finally a mode is expanded within that piece of powered equipment. By following the indentation, the user can easily tell where an expansion starts and ends and to whom that expansion belongs. Also note how overwhelming the data would become if all the data associated with the activity were automatically displayed to the user. It would make it very difficult if not impossible to figure out what the activity was doing!

4.8.7 Layout of an Activity, Subtask, Requirement and Powered Equipment

Up to now, editing data fields has been discussed in the broad sense. This section examines in detail the individual fields that make up the activities, subtasks, requirements and powered equipment. We will do this by stating the field name and the category that it belongs to and in case of category II, we will discuss what the menu options do.

Activity Fields

- Description (Category I) An alpha-numeric string that describes the task of the activity.
- Name (Category I) An alpha-numeric unique string that represents the name of the activity.
- Static Priority (Category I) A number between zero and three that represents the static priority of the activity.
- Performances Requested (Category I) A number representing how many times the user would like to have this activity scheduled.
- Minimum Performances Needed (Category I) The minimum number of performances that must be scheduled for this activity to be placed in the schedule. This number must be less than the number of performances requested.
- Subtask (Category III) The individual sequential tasks that must be accomplished by the activity. The number of subtasks is unlimited and the ordering is important. The order that the subtasks appear in the activity is the order of the subtask execution in that activity.

This is a category II field so using the right mouse button on this data field produces a menu. See Figure 40. There are five items in this menu. The first four items have pull-right menus associated with them that contain all the subtasks that currently exist. See Figure 41.

- The first menu item "Inserting Existing Subtask Above" will insert the subtask selected above the current highlighted subtask (the initial subtask that was selected to bring up the menu).

System Options	Activity Editor
Edit	Description: <input type="text" value="The operation of the pumps to supply cabin air."/>
Create	Name: <input type="text" value="Cabin Air Supply"/> Base Priority: <input type="text" value="0"/>
Delete	Performances Requested: <input type="text" value="100"/> Minimum Performances Needed: <input type="text" value="1"/>
Save	Subtask: Main Cabin Air Supply 1 Max Duration: <input type="text" value="129600"/> Min Duration: <input type="text" value="1"/>
Create Schedule	Maximum Delay: <input type="text" value="0"/> Minimum Delay: <input type="text" value="0"/> Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/>
Delete Schedule	Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input type="text" value="On"/> Location: <input type="text" value="LC-3-4"/> Default Location: <input type="text" value="LC-3-4"/>
Help	Start: <input type="text" value="0"/> End: <input type="text" value="525600"/> Quantity: <input type="text" value=""/> Mode: On Maximum Power: <input type="text" value="800"/> Minimum Power: <input type="text" value="0"/> Switch to Redundant: <input type="text" value="Y"/>

Figure 39: Activity Editor Expansion of a Subtask

System Options	Activity Editor	
Edit	Insert Existing Subtask Above Insert Existing Subtask Here Copy an Existing Subtask Above Copy an Existing Subtask Here Remove This Subtask Entry	
Create		
Delete		

Figure 40: Activity Editor Subtask Options Menu

- The second menu item “Insert Existing Subtask Here” will replace the current subtask that is highlighted with the one that is selected.
- The third menu item is very similar to the first menu item except that it will first make a copy of the subtask selected and then insert it above the highlighted subtask.
- Likewise the fourth menu item is similar to the second menu except that it will first make a copy of the subtask selected and then insert it into the highlighted subtask.
- The fifth menu item “Remove This Subtask Entry” will remove the highlighted subtask from the activity.

The copy options in the third and fourth menu items are added for flexibility. This allows the user to select a subtask that needs slight modification, but the user does not want the modification to affect other activities that might use this subtask.

Subtask Fields

- **Maximum Duration (Category I)** The maximum number of minutes that the subtask is allowed to run.
- **Minimum Duration (Category I)** The minimum number of minutes that the subtask must run. This number must be less than or equal to the Maximum Duration number.

System Options	Activity Editor	
Edit	<div style="background-color: #cccccc; padding: 5px;"> Subtask Options Insert Existing Subtask Above Insert Existing Subtask Here Copy an Existing Subtask Above Copy an Existing Subtask Here Remove This Subtask Entry </div>	
Create		
Delete		
		<div style="background-color: #cccccc; padding: 5px;"> Copy an Existing Subtask Here Operate Main Cabin Lights Main Cabin Air Supply 1 Main Cabin Air Supply 2 Operate Secondary Cabin Lights Operate Avionics Air Process Waste Heat Waste </div>

Figure 41: Activity Editor Subtask Options Menu

- **Maximum Delay (Category I)** The maximum delay in minutes between the start of this subtask and the ending of the preceding subtask in this activity. The field is not applicable if this is the first subtask of an activity.
- **Minimum Delay (Category I)** The minimum delay in minutes needed between this subtask and the preceding subtask in this activity. This number must be less than or equal to the Maximum Delay and is not applicable if this is the first subtask in the activity.
- **Requirement (Category III)** The individual non-powered equipment resources that the subtasks uses. The number of requirements is unlimited and the ordering is not important. This is a category III field so the right mouse button on this data field will cause a menu of options to be displayed. This menu has five items in the menu. They are exactly like the subtask where the user can insert and remove requirements from the subtask. The big difference is that there is another field associated with the requirement field that is called the amount field. The amount field is located just left of the requirement field on the same line. See the requirement field in Figure 39. This was done to let the user know that the two fields work as a group and should be treated as one. When the user inserts a new requirement into the subtask, the amount field will default to 1. It is the user's responsibility to change the amount field to reflect how much of that resource will be used. The amount field is a category I field. Currently there are two types of requirements that are used: rate controlled (Crew) and equipment (Wrench).
- **Amount (Category I)** The number representing how much of the requirement will be used for this subtask. This number cannot be larger than the maximum amount that the requirement can have. This value is located in the availability list for the requirement.
- **Powered Equipment (Category III)** The individual powered equipment that will be used by this subtask. The number of pieces of powered equipment for a subtask is unlimited. The order of the powered equipment is not important because by definition it represents what pieces of powered equipment the subtask uses over a given time interval. This is a category II data field that contains a menu option that acts just like the menu options in the subtask menus. See Figure 40 and Figure 41. The user uses the menu options to insert or remove pieces of powered equipment from the subtask.

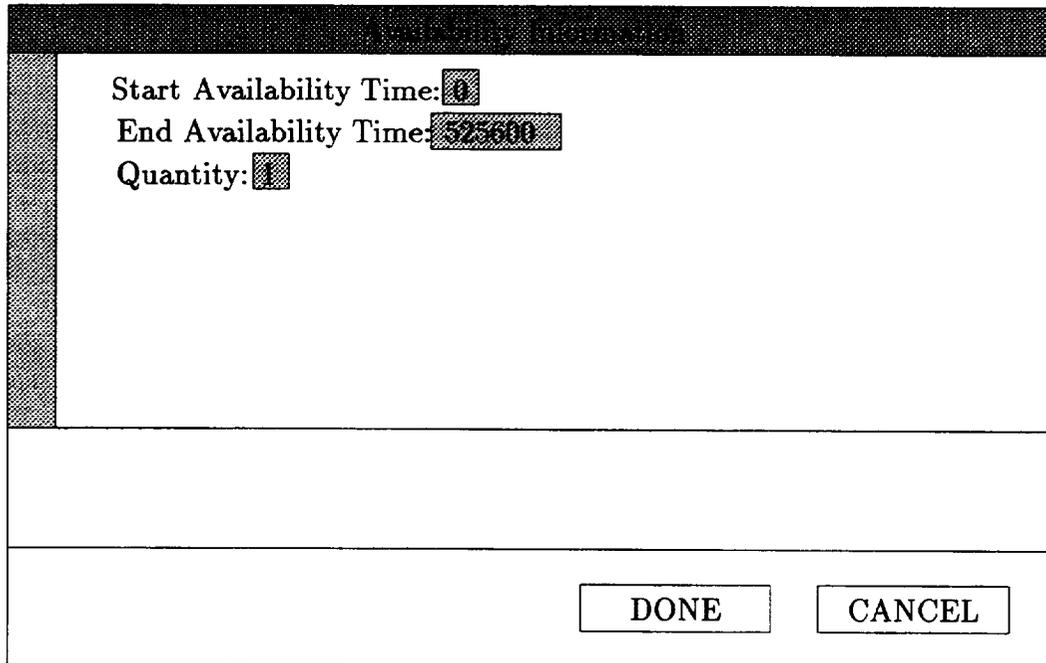
There are two other pieces of data that are associated with every piece of powered equipment that are inserted in the mode and location fields. These two data fields are on the same line as the piece of powered equipment to show they belong together. Each is discussed below.

- **Mode (Category I*)** The mode in which the piece of powered equipment operates. The mode represents in which operational configuration the piece of powered equipment operates. When the user inserts a new piece of powered equipment into a subtask, the mode default is the first mode in the modes list for that piece of powered equipment (very non-deterministic). It is the user's responsibility to change this field to support the goal of the subtask. The I* means that the user does not edit it directly. When the user selects the data field with any mouse button, it will provide a menu of valid modes for that particular piece of powered equipment, from which the user selects one of the options.
- **Location (Category I*)** The location data field represents which switch the piece of powered equipment will be plugged into. When the powered equipment is inserted into the subtask, this data field defaults to the equipment's default location. This is the location that the piece of powered equipment will reside when the subtask is running. This is an I* category because the update is done through a menu that contains all the 1K switches in the system. The location of 3K switches is not currently implemented. When the user inserts a piece of powered equipment into the system, this data field is set to the default location specified in the powered equipment definition structure.

Requirement Fields

- **Name (Category II)** An alpha-numeric unique string that represents the name of this requirement. The name will appear at different indentation levels. In the case where the user has expanded the requirement from within a subtask, the name will be the requirement name seen in the subtask. It will not be prefaced by name. If the user edits the requirement directly, the name will be preceded by the word "NAME:" and the indentation level for all the data will be the same.
- **Resource Type (Category I*)** The type for this requirement. There are currently only two valid responses: rate controlled and equipment. MAESTRO will handle five and future versions of the activity editor will be upgraded to handle all five. The category type is a I* because the only way to change the data is through a menu that contains only valid resource types.
- **Start End Quantity (Category I)** The three data fields on one line represent an entry in the availability list for the requirement. The availability list is made up of a list of lists. Each element of the list contains a start time, end time and quantity. This is the way the user describes how much of the resource there is over time.

Figure 39 shows an example of a very basic availability list that just covers one interval with the quantity 1. The time span must cover from 0 minutes to 525,600 minutes. To



Start Availability Time: 0
End Availability Time: 525600
Quantity: 1

DONE CANCEL

Figure 42: Activity Editor Availability Edit Workbox

change the availability list the user selects any of the data fields. A work box will appear in the scratch pad window. See Figure 42. This workbox will contain three data fields: start, end and quantity. There will be default values in each one of the data fields: start is 0, end is 525,600, and quantity is 1.

When the user edits the availability data fields, a specific interval is created to insert in the availability list. Suppose the user enters a start time of 10 minutes, an end time of 100 minutes, and a quantity of 2. The new availability list would look like Figure 43. Notice in Figure 43 that the new availability list now has three entries. The first goes from 0 to 10 minutes at a quantity of 1, the next entry goes from 10 minutes to 100 minutes at a quantity of 2 (the one just entered), and the third entry goes from 100 minutes to 525,600 minutes at a quantity of 1. When the user adds a new interval, the system knows to fill out the availability list to cover the time span from 0 to 525,600 minutes. Note that if the user had entered the quantity value of 1 instead of 2 then there would have been no change in the availability list because all the quantities would have the same value. By changing the interval, the user can create any required complexity for the availability list.

System Options	Activity Editor
Edit	Description: <input type="text" value="The operation of the pumps to supply cabin air."/>
Create	Name: <input type="text" value="Cabin Air Supply"/> Base Priority: <input type="text" value="0"/>
Delete	Performances Requested: <input type="text" value="100"/> Minimum Performances Needed: <input type="text" value="1"/>
Save	Subtask: Main Cabin Air Supply 1 Max Duration: <input type="text" value="129600"/> Min Duration: <input type="text" value=""/>
Create Schedule	Maximum Delay: <input type="text" value="0"/> Minimum Delay: <input type="text" value="0"/> Requirement: <input type="text" value="*****Undefined*****"/> Amount: <input type="text" value="*****"/>
Delete Schedule	Powered Equipment: <input type="text" value="Main Cabin Air Pump"/> Mode: <input type="text" value="On"/> Location: <input type="text" value="LC-3-4"/> Default Location: <input type="text" value="LC-3-4"/>
Help	Start: <input type="text" value="0"/> End: <input type="text" value="10"/> Quantity: <input type="text" value="1"/> Start: <input type="text" value="0"/> End: <input type="text" value="100"/> Quantity: <input type="text" value="2"/> Start: <input type="text" value="100"/> End: <input type="text" value="525600"/> Quantity: <input type="text" value="1"/> Mode: On

Figure 43: Activity Editor Updated Availability List

To remove an interval, extend the interval preceding it, and extend it to cover the interval you want to remove. This may seem confusing, but just try it and it will become second nature in just a few clicks of the mouse.

Powered Equipment Fields

- **Default Location (Category I*)** This is the default switch that the piece of powered equipment will default to when selected for use in a subtask. The selection is made through a menu that contains a list of available 1K switches from which the user can choose. When the user creates a new piece of powered equipment, they specify this location.
- **Start End Quantity (Category I)** This is exactly the same as the availability discussed in Section 4.8.7.
- **Mode (Category III*)** These represent all the currently defined modes for this piece of powered equipment. This is a category III* because the left and right mouse buttons do different actions. The left mouse button has no action at all because there is no list of other modes that can be placed into this list of modes. All valid modes are in the list.

The right mouse button is also different. It brings up a menu of options, but this menu only has two options while the others had five. The two options are creating a new mode for this piece of powered equipment and deleting a mode from this piece of powered equipment. The others allowed the user to insert existing or copied data into the particular data type, but because the mode is unique to powered equipment, this paradigm had to be changed to handle this. In creating a new mode, the name must be unique within one piece of powered equipment. That is, two pieces of powered equipment could each have a mode named "ON" without causing a problem.

Mode Fields Modes are only editable through a piece of powered equipment. This allows the user to re-use the same mode name (i.e. ON) in more than one piece of powered equipment. It is not possible to edit a mode from the application menu, because the user needs the context within which the mode is being edited.

- **Maximum Power (Category I)** The maximum power in watts that the piece of powered equipment may use in this mode. This number may not exceed 1000 or be less than 0. This number must also be greater than the minimum power. Equal is not allowed.
- **Minimum Power (Category I)** The minimum power in watts that a piece of powered equipment uses. This number must be between 0 and 1000 and must be less than the maximum power.

- **Switch to Redundant (Category I)** A single alpha-character, “Y” or “N”, that specifies whether the mode for this piece of powered equipment has a switch to redundant capability. “Y” means it has redundant capability for this mode, and “N” means there is no redundant capability for this mode.
- **Permission to Test (Category I)** A single alpha-numeric character, “Y” or “N”, that specifies whether the mode for this piece of powered equipment is testable during a fault. A “Y” means the piece of equipment is testable, and “N” means that the piece of equipment is not testable in this mode.

4.8.8 Scheduling

The activity editor’s job is to create and update activities and other building blocks used to create activities. These activities are then used to run schedules on the *SSM/PMAD* system.

The user can create schedules from the activity editor by using the “Create Schedule” option on the activity editor application menu. Upon selection of this item the system will pop up a scrollable menu window. See Figure 44. This menu is different from any we have seen so far in the activity editor. By selecting any item that is not enclosed by << >>, the item will become selected. The item is highlighted after it is selected. The user will then go through and select the activities to be scheduled. If the user erroneously selects an activity, selecting it again will cause it to be de-selected and the menu item will be de-highlighted. This menu is also scrollable because the number of activities may be very large. Once the user is done selecting activities, choosing the menu item titled <<Schedule Selected Activities>> will start the scheduling processes.

The first step of the scheduling process is to determine if the database has been modified or not. If any of the data in the activity database has been changed, then the database must be saved prior to scheduling. Also if any of the data has been changed, the scheduling process determines if any of these changes will affect already existing schedules and informs the user of the conflicts. Any change to the data that affects a schedule must be dealt with immediately, so the system will put up a workbox of all the effected schedules and let the user decide what to do with them. Figure 45 shows an example of the workbox. In this case the changes to the activity database have invalidated three schedules.

If a schedule is invalidated due to activity changes to the activity database, the user can do one of three things to the schedule:

- Reschedule the activities that make up the schedule
- Delete the schedule
- Ignore the problem

System Options	Activity Editor
Edit	<<Schedule Selected Activities>> <<Cancel>> Main Cabin Lights Cabin Air Supply Fire Safety System Secondary Cabin Lights Avionics Urine/Waste Management Urine/Waste Heaters Life Sciences Glovebox
Create	
Delete	
Save	
Cancel Schedule	

Figure 44: Activity Editor Scheduling Menu

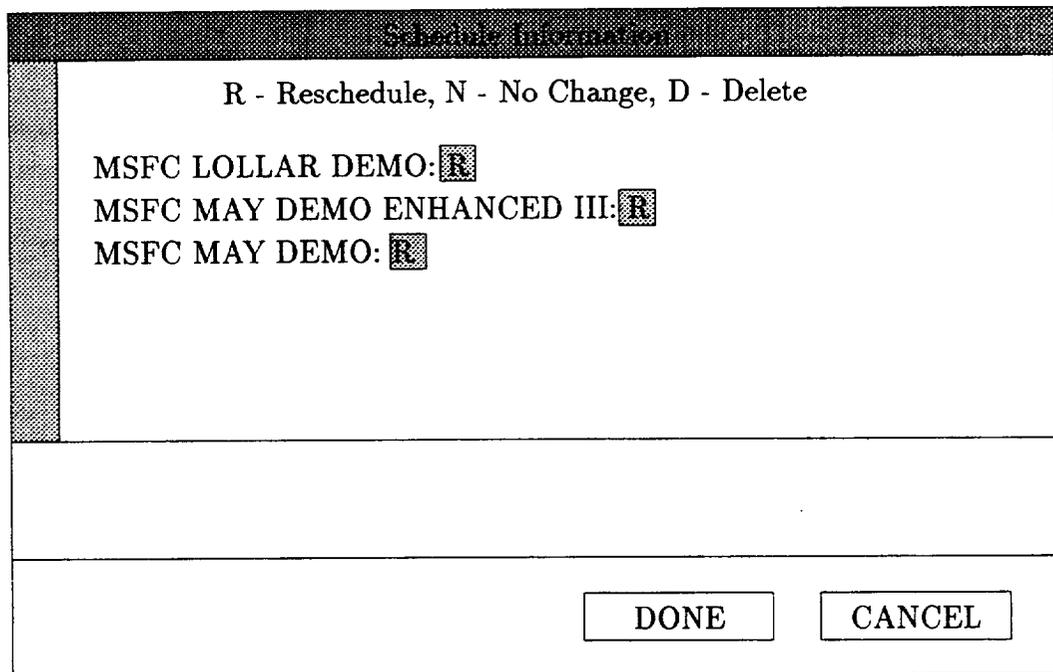


Figure 45: Activity Editor Scheduling Changes

The third option is very dangerous because if the user invalidated the schedule and decides not to rectify the problem, there is no guarantee that the schedule will run correctly. This option is included for flexibility, but the user should exercise caution when using it.

After the user has saved the database, the system will pop up another workbox to obtain non-activity pertinent information. At the present time there are only three pieces of non-activity information needed for scheduling:

- Schedule name
- Start time for the schedule in minutes
- End time for the schedule in minutes

For now, leave the start time at zero. The effect of changing it to a non-zero value is still being investigated. The end time must be greater than the start time and less than 525,600 minutes. Once the user has finished entering this data, the data will be sent to MAESTRO for scheduling.

When scheduling is completed, look at the scratch pad window to see if any problems may have occurred. Scheduling information concerning activities which were not scheduled for some reason will appear in the window, and can be used to correct the activity.

4.8.9 Deleting the Data

To use the deleting capabilities of the activity editor, the user selects the Delete option on the applications menu. This causes a menu to appear in the middle portion of the left hand side of the application main window. Notice that there are four entries in the menu:

- Activity
- Subtask
- Requirements
- Powered Equipment

From an earlier discussion, the user will remember that requirements and powered equipment are resource building blocks used in the creation of subtasks, which are used to build activities, which are used to build schedules.

The four choices for deletion (activity, subtask, requirements, and powered equipment) are discussed in the following four sections. Each of the four choices in the delete menu has a pull-right menu associated with it. These pull-rights contain all the items that have been created for that category. Some pull-right menus are scrollable in order to accommodate the large number of items that must fit into a fixed size menu.

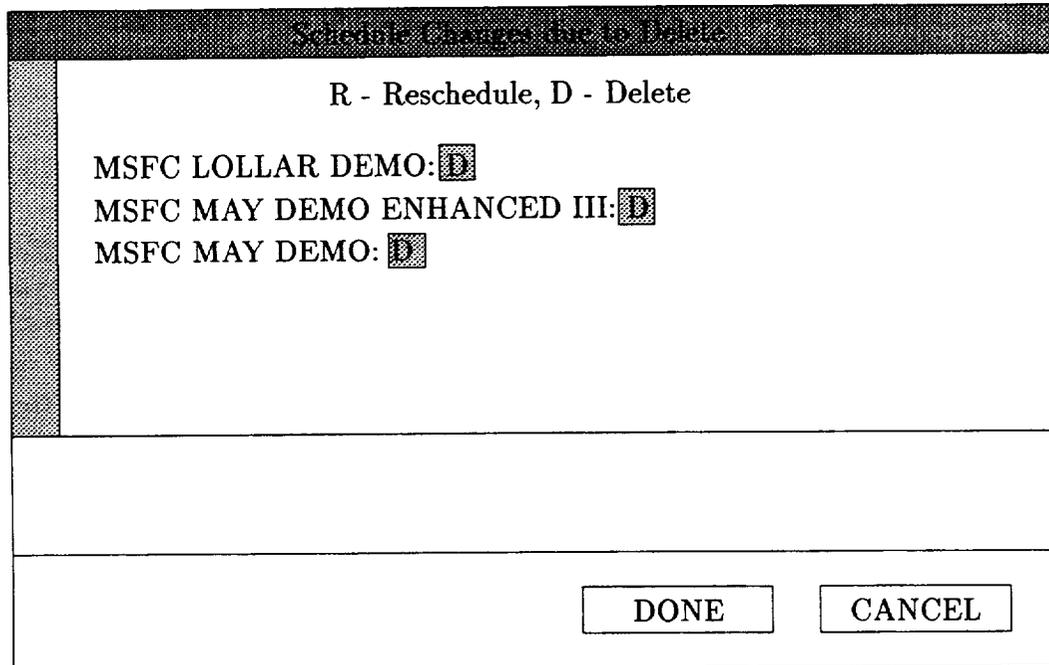


Figure 46: Schedule Changes due to Delete

Delete Activity One of the four choices for deletion is an activity. The user can delete an activity by selecting the Delete Option of the Application menu. Selecting Activity will produce a pull-right with all the available activities for deletion. The user uses any of the three mouse buttons to select an activity for deletion. After selecting an activity to delete, one of two workboxes will appear in the scratchpad area of the screen. If the activity is not a part of any schedule, a workbox entitled "Confirmation!" will appear. If any schedules are affected by the deletion of the selected activity, a workbox entitled "Schedule Changes due to Delete" will appear.

If the workbox entitled "Schedule Changes due to Delete" appears, the changes to the schedule must be dealt with immediately, and the user decides if the schedule should be rescheduled or deleted. The user types R for reschedule or D for delete for each of the schedules affected. Clicking "Done" will initiate the changes or clicking "Cancel" will cancel the request. Figure 46 shows an example of the workbox where schedules are affected. In this case the deletion of activity Main Cabin Lights has invalidated the listed schedules.

After clicking "Done", the request is processed, and a report will be printed to the scratch pad area displaying the success rate of rescheduling the activities in the schedule.

The activity will not appear in the revised schedule. If the schedule is requested for deletion, the schedule will be deleted from the list of all schedules in the database.

If the activity to be deleted is not a part of any schedule, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox cancels the request and erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user's intention before really deleting an activity.

Several slots of the Frames database are affected by deleting an activity. As the result of the deletion of an activity, the activity is deleted from each of the subtasks that comprise the activity. This relationship between activities and subtasks is recorded in the subtask frame in a slot called activities. The activity and its execution sequence number are also deleted from the subtask frame slot called sequence. The activity group is deleted from the list of all activity groups in the system. The activity is deleted from the list of activities on the edit and delete pull-right menus. The activity is deleted from the list of activities on the create schedule menu.

Delete Subtask Another choice for deletion is a subtask. The user can delete a subtask by selecting the Delete Option of the Application menu. Selecting Subtask will produce a pull-right with all the available subtasks for deletion. See Figure 47.

The user uses any of the three mouse buttons to select a subtask for deletion. After selecting a subtask to delete, a workbox or a message will appear in the scratchpad area of the screen. If the subtask is a part of an activity, a scratchpad message entitled "Subtask Deletion Information" will appear. See Figure 48. If the subtask is not a part of any activity, the confirmation workbox appears.

The message entitled "Subtask Deletion Information" lists all the activities that contain the subtask to be deleted. The message explains that the subtask can not be deleted because there are activities that use the subtask. In general, the user must first delete the activities that use the subtask, and then delete the subtask. At this point in time, the system will not do that automatically for the user, because the connection between subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the subtask to be deleted is not a part of any activity, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox cancels the request and erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user's intention before really deleting an subtask.

As the result of the deletion of a subtask, the subtask is deleted from a list of all subtasks in the system, from each of the pieces of powered equipment that use the subtask, from

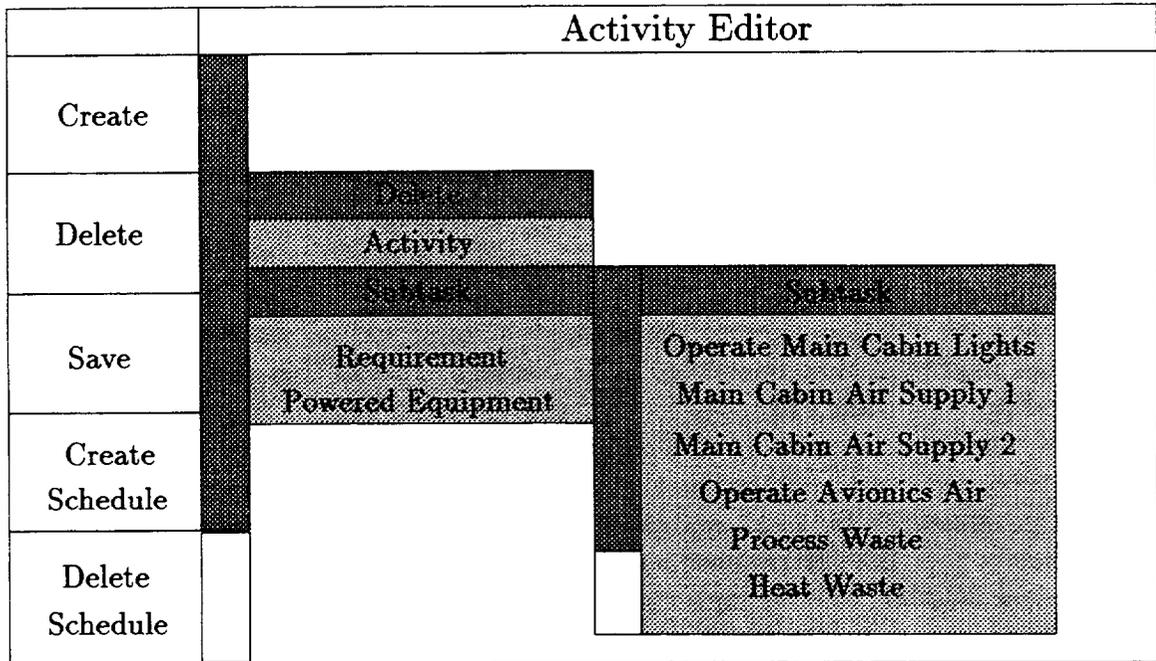


Figure 47: Delete Subtask Pull Right Menu

Subtask Deletion Information:
 Unable to delete requested subtask, Operate Main Cabin Lights,
 because the following activity/activities use the subtask:
 Main Cabin Lights

Figure 48: Subtask Deletion Information

the list of subtasks on the edit and delete pull-right menus, and from the list of subtasks displayed when "Insert an Existing Subtask Above" is selected while editing subtasks.

Delete Requirement Another choice for deletion is a requirement. The user can delete a requirement by selecting the Delete Option of the Application menu. Selecting Requirement will produce a pull-right with all the available requirements for deletion. The user uses any of the three mouse buttons to select a requirement for deletion. After selecting a requirement to delete, a workbox or a message will appear in the scratchpad area of the screen. If the requirement is a part of a subtask, a scratchpad message entitled "Requirement Deletion Information" will appear. If the requirement is not a part of any subtask, the confirmation workbox appears.

The message entitled "Requirement Deletion Information" lists all the subtasks that contain the requirement to be deleted. The message explains that the requirement can not be deleted because there are subtasks that use the requirement. In general, the user must first delete the subtasks that use the requirement, and then delete the requirement. At this point in time, the system will not do that automatically for the user, because the connection between requirements, subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the requirement to be deleted is not a part of any subtask, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox cancels the request and erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user's intention before really deleting a requirement.

As the result of the deletion of a requirement, the requirement is deleted from a list of all requirements in the system. The list of all requirements is a part of the the list of all resources in the system. The requirement is deleted from the list of requirements on the edit and delete pull-right menus. The requirement is deleted from the list of requirements displayed when "Insert an Existing Requirement Here" is selected while editing requirements.

Delete Powered Equipment Another choice for deletion is a piece of powered equipment. The user can delete a piece of powered equipment by selecting the Delete Option of the Application menu. Selecting Powered Equipment will produce a pull-right with all the available pieces of powered equipment for deletion. The user uses any of the three mouse buttons to select a piece of powered equipment for deletion. After selecting a piece of powered equipment to delete, a workbox or a message will appear in the scratchpad area of the screen. If the piece of powered equipment is a part of a subtask, a scratchpad message entitled "Powered Equipment Deletion Information" will appear. If the piece of powered equipment is not a part of any subtask, the confirmation workbox appears.

The message entitled "Powered Equipment Deletion Information" lists all the subtasks that contain the piece of powered equipment to be deleted. The message explains that the piece of powered equipment can not be deleted because there are subtasks that use the piece of powered equipment. In general, the user must first delete the subtasks that use the piece of powered equipment, and then delete the piece of powered equipment. At this point in time, the system will not do that automatically for the user, because the connection between pieces of powered equipment, subtasks, activities and schedules is very intricate. The rule employed in deletions is that the system does not delete higher objects when a lower object has been requested for deletion, and the higher object uses the lower object. The user must first delete the higher objects using the menu options before deleting the lower level object.

If the piece of powered equipment to be deleted is not a part of any subtask, the confirmation workbox will be displayed. The only action required from the user is to click "Done" to validate the deletion, or click "Cancel" to cancel the request. Clicking anywhere outside the workbox also erases the workbox from the scratchpad area. The confirmation workbox is used to confirm the user's intention before really deleting a piece of powered equipment.

As the result of the deletion of a piece of powered equipment, the piece of powered equipment is deleted from a list of all resources in the system. The piece of powered equipment is deleted from the list of pieces of powered equipment on the edit and delete pull-right menus. The piece of powered equipment is deleted from the list of requirements displayed when "Insert an Existing Powered Equipment Here" is selected while editing pieces of powered equipment.

4.9 ILA Interface

4.9.1 ILA System Overview

ILA activities are initiated when the user requests control of some switches through the "Seize Manual Control" menu. These requests are displayed in workbox entitled "ILA Seizure Workbox".

The user can edit the information about the switches being seized. This information includes when the switch is to be under manual control and for how long, what priority the manual seizure is compared to other activities on the schedule, the power requirements of the switch in watts, and whether FRAMES has permission to test the switch if any faults occur in the system while the switch is under manual control.

After the user updates the switch seizure information to reflect what is needed, the user asks the system to calculate the effects of the manual seizure on the current system. The effects are calculated and displayed to the user. The user can continue to change switch information or add or drop switches until the effects on the system are acceptable.

The ILA functions determine if it is possible to place the switch on the schedule. The ILA functions notify MAESTRO of the unavailability of the switch in autonomous operations,

and MAESTRO generates the necessary adjustments. After the changes to the schedule are made and the seizure of the switch is represented in the schedule as an activity, the switch is automatically turned on under manual control.

The switch is available for manual use when the hand appears on the user interface. When the user has the switch under manual control, the user may turn the switch on or off using the "Manual On" and "Manual Off" menu options on the user interface. Manually turning the switch off does not indicate to the rest of the *SSM/PMAD* system that the user is finished testing the switch. These manual ons and offs are completely controlled by the user while the switch is under manual control. However, while the switch is "on" under manual control, the allocated power originally requested by the user is what is available. Furthermore, even if the switch is "off" under manual control, the FELES screen will indicate that the activity on the manually controlled switch is still active.

When the period of time requested for the manual seizure has elapsed, the switch will automatically be turned off and given back to the autonomous operations and made available to MAESTRO. The hand icon will be removed. It is possible to release the switch from manual control earlier by utilizing the "Release Manual Control" option on the user interface.

4.9.2 Groups

The ability to group switches gives the user a way to tell ILA that a set of switches must be obtained in order to accomplish a task. A failure to obtain one switch in the set negates the entire group.

Grouping switches is performed through a menu option on the power system screen entitled "Group Switches". Selecting this option causes a menu to appear with all the load centers, and from the load centers, pull-rights of all the switches in a load center. The user goes through the menu selecting the switches for a group. Every time a selection is made that menu item is highlighted. To deselect any switch, select the highlighted item and it is dehighlighted. Switches are selected and grouped by button clicking the <Group Selected Items>. A workbox entitled "New Switch Group Information" is displayed. The workbox contains a field for the unique name of the group. After giving the group a name, the user button clicks "Done" and the workbox disappears from the scratchpad area. See Figure 49.

4.9.3 Manual Seizure

The "Seize Manual Control" initiates the dialogue between a user and the system to negotiate the transfer of control of the requested switches. The "Seize Manual Control" option displays a menu of the individual switches as well as the groups that are available. See Figure 50.

The menu items in "Seize Manual Control" are of two types. The first type of menu item is a load center item. These menu items have pull-right menus that contain the individual switches for that load center. So if the user wants to select switch 4 of load center 3, move

SWITCH GROUPING	
Group Name:	<input type="text" value="Experiment 14"/>
Switches In The Group:	
	LC-01 SW: 03
	LC-02 SW: 14
	LC-02 SW: 12
	LC-03 SW:21
	<input type="button" value="ADD/DROP"/> <input type="button" value="DONE"/> <input type="button" value="CANCEL"/>

Figure 49: Switch Grouping

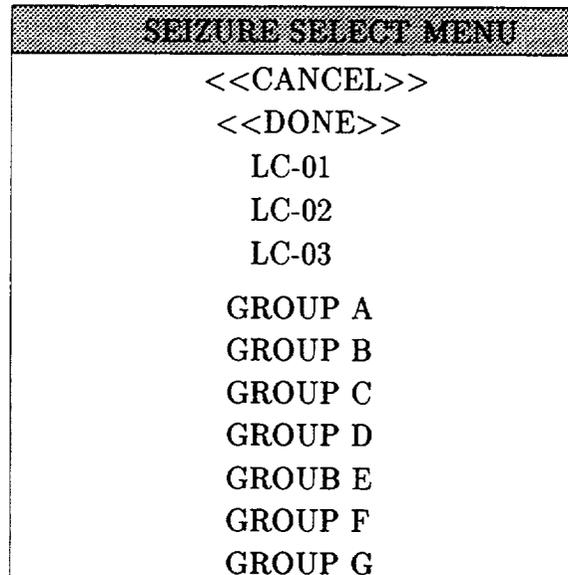


Figure 50: Seizure Select Menu

the mouse to the load center 3 menu item and then select switch 4 of the pull-right menu. The second type of menu item in the “Seize Manual Control” menu is the group item. The user selects the group, and the group item will become highlighted. If the user wants to deselect a switch or group, button click on the item to be deselected.

The menu also includes two action items. The user can cancel from the “Seize Manual Control” by button clicking the “cancel” option or initiate the seizure by button clicking <Seize Selected Items>.

4.9.4 Data Collected for Seizing Switches

The data collected for each switch being seized consists of the following items whose definitions are detailed below:

- **Switch Name** This is a symbol identical to the identification of a switch in the *SSM/PMAD* database.
- **Starting Time** This is the mission time that the seizure of the switch is to begin. The number must be between 0 and 525600. Default: 0 (Now)
- **Duration** This is the length in minutes that the user wants to seize the switch. This number must be between 1 and 525600. Default: 525600 (Forever)

- **Static Priority** This is the priority category for the seizure of the switch. The integer must be between 0 and 3 with 0 being the highest priority and 3 being the lowest. Default: 3 (Lowest)
- **Relative Priority** This is the users ordering of the switches for a manual seizure event within a particular priority category. The user is required to make sure the ordering is correct or the calculation routine will error on bad input. Default: Next value in the static level.
- **Power Consumption** This is an integer value between 0 and 1000 that represents the maximum amount of power that will be consumed by this switch. Default: 1000 (Highest)
- **Redundant?** This value is either T or NIL and represents if the switch has redundant capability. Default: NIL (No)
- **Testable?** This value is either T or NIL and represents whether the switch is testable by the FRAMES system if a fault occurs. Default: NIL (No)
- **Diagnosable?** This value is either T or NIL and it represents if the FRAMES system should try to diagnose a fault on this switch if one were to occur. Default: T (Yes)
- **Bumpable?** This value is either T or NIL and represents whether this switch can bump other switches within the same priority category when trying to seize this switch. Default: NIL (No)
- **Switch Position** This value is either T or NIL and represents the position or the switch upon seizure. T means the switch will be turned on while NIL means the switch will be turned OFF. Default: NIL (Off)

When the data is initially presented in the “ILA Seizure Workbox”, all the fields contain default values except the name field and the group field, if applicable. All entries prompt “Run Calculate for Effects”.

4.9.5 ILA Seizure Workbox

To interpret the information of the ILA Seizure Workbox, notice that the data is broken into discrete groups which will be referred to as entries. They are called entries to prevent confusion with the concept of grouping switches. Each entry contains either a switch or a group of switches. Each entry is broken down into a description and an effects part. See Figure 51.

ILA Seizure Information Workbox	
LC-01 SW: 02 NONE [00.00:00 - 99.99:99] [3 - 1] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS	
LC-02 SW: 03 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF LC-02 SW: 04 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF LC-03 SW: 21 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS	
LC-01 SW: 23 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D OFF LC-03 SW: 02 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS	
LC-03 SW: 04 NONE [00.00:00 - 99.99:99] [3 - 4] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS	
LC-01 SW: 07 NONE [00.00:00 - 99.99:99] [3 - 5] 1000W —D OFF REASON: RUN CALCULATE TO SEE EFFECTS	
<input type="button" value="CALCULATE"/> <input type="button" value="HELP"/> <input type="button" value="DROP/ADD"/> <input type="button" value="DONE"/> <input type="button" value="CANCEL"/>	

Figure 51: ILA Seizure Information Menu

ILA Seizure Information Workbox

SINGLE → LC-01 SW: 02 NONE [00.00:00 - 99.99:99] [3 - 1] 1000W —D ON
EFFECTS: ACTIVITY GLOVEBOX HOUSKEEPING DISPLACED BY SWITCH SEIZURE
EFFECTED SWITCH(S): LC-01 SW: 02
LC-02 SW: 06
ACTIVITY STAR GAZER DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCHE(S): LC-01 SW: 04

GROUP → LC-02 SW: 03 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON
LC-02 SW: 04 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF
LC-03 SW: 21 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON
EFFECTS: ACTIVITY EXOBIOLGY EXP DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCH(S): LC-01 SW: 05
LC-03 SW: 03
ACTIVITY MICRO COUNTER DISPLACED DUE TO POWER CONSTRAINT
EFFECTED SWITCH(S): LC-02 SW: 06

LC-01 SW: 23 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON
LC-03 SW: 02 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON
REASON: LC-01 SW: 23 STATIC PRIORITY TO LOW
LC-03 SW: 03 RELATIVE PRIORITY TO LOW

LC-03 SW: 18 NONE [00.00:00 - 99.99:99] [3 - 4] 1000W —D ON
EFFECTS: ACTIVITY STAR COUNTER DISPLACED DUE TO LOSS OF REDUNDANT
SWITCH LC-03 SW: 18
EFFECTED SWITCH(S): LC-03 SW: 04

LC-01 SW: 07 NONE [00.00:00 - 99.99:99] [3 - 5] 1000W —D ON
REASON: LC-01 SW: 07 POWER REQUIREMENT EXCEEDS MAX ALLOWD ON 3K

CALCULATE HELP ADD/DROP DONE CANCEL

Figure 52: Switch Description Lines

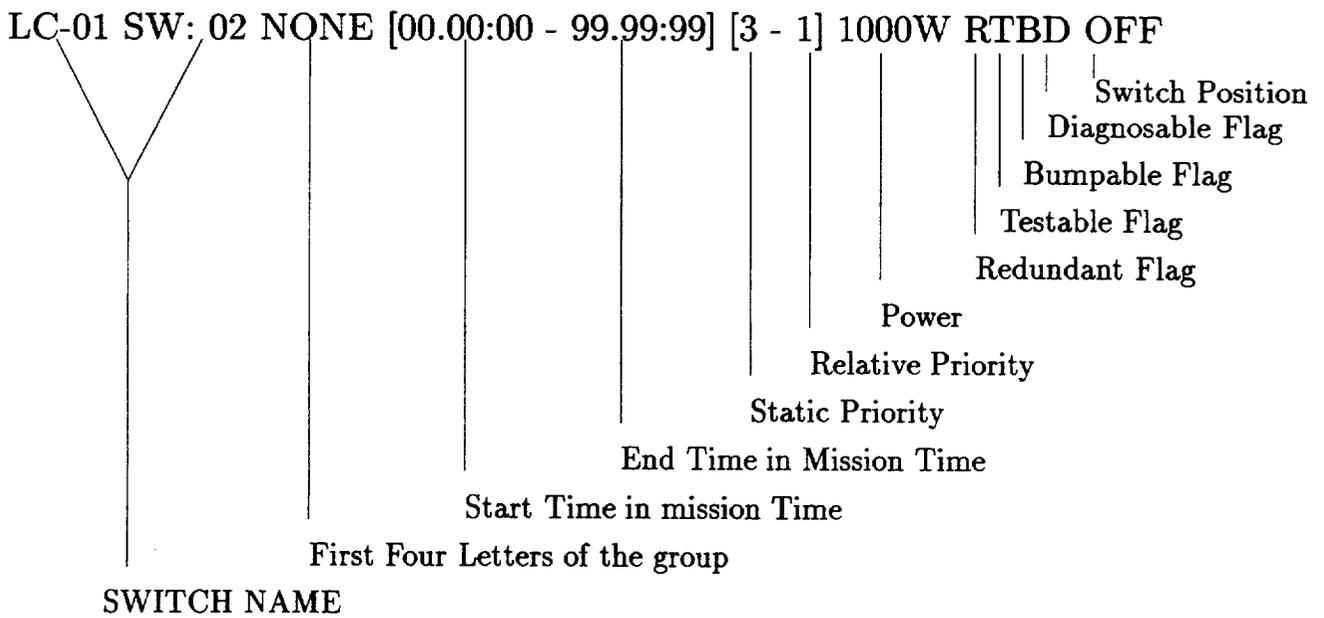


Figure 53: Switch Description Line Reference

The first part is the switch description line. Figure 52 shows an example of an entry with a single switch and grouped entries. See Figure 53 for a dissection of the switch description line.

The switch description line displays the defaults in a shorthand notion. "RTBD" is short for redundant, testable, bumpable and diagnosable respectively. If a "-" appears for one of the values, it means that the flag value has not been set.

The effects field is the textual information that describes either the effects of manual seizure on the system, or a reason the seizure is not allowed for the entry. A list of all the possible effects are listed below:

- Activity xxxx interrupted due to seizure of switch LC-xx SW: xx
- Activity xxxx interrupted due to seizure of bumpable switch LC-xx SW: xx
- Activity xxxx interrupted due to redundant needs on switch LC-xx SW: xx
- Activity xxxx interrupted due to loss of redundant switch LC-xx SW:xx

The affected switches will probably get shed. MAESTRO will try to reschedule them or move them to other switches depending on the activity. ILA is giving the worst case scenario of what might happens. The switches that are grabbed during ILA activities will not be shed due to MAESTRO rescheduling associated with the manual seizure. But the switches may be shed later due to a source power change.

There are cases when the system is unable to seize a switch. The switch seizure failure reasons are listed below:

- Static priority equal but not bumpable for switch LC-xx SW: xx
- Static priority too low for switch LC-xx SW: xx
- Power requirement exceeds max for switch PDCU SW: xx
- Unable to grab redundant switch for switch LC-xx SW: xx
- Run calculate for effects

The "ILA Seizure Workbox" has five buttons to select. They are Calculate, Help, Add/Drop, Done and Cancel. The "Cancel" button is like the cancel button on any workbox. The Done button will calculate any seizures that are allowed.

LC-01 SWITCH: 02	
STARTING TIME (dd.hh:mm):	00.18:30
DURATION (dd.hh:mm):	0.05:20
STATIC PRIORITY(0 - 3):	1
RELATIVE PRIORITY(1 - 99):	1
POWER CONSUMPTION (Watts):	500
IS SWITCH REDUNDANT (Y/N):	N
IS SWITCH TESTABLE (Y/N):	Y
IS SWITCH DIAGNAOSABLE BY FRAMES (Y/N):	Y
PERMISSION TO BUMP OTHER SWITCHES (Y/N):	Y
INITIAL SWITCH POSITION (O/F):	O
<input type="button" value="DONE"/> <input type="button" value="CANCEL"/>	

Figure 54: Switch Seizure Workbook

4.9.6 Editing Switch Seizure Data

To edit the data for a switch, move the mouse to the switch that is to be updated in the ILA Seizure Workbook. The name of the switch is mouse sensitive, so when the mouse enters the switch name region the name will be highlighted. When the name is highlighted, a mouse click will cause a workbook containing the data for the switch to be displayed. The user can update any field in the workbook using standard workbook editing commands. See figure 54 for an example of the switch information which could be updated.

4.9.7 The Add/Drop Option

The user can add or drop a switch from the list of switches to be seized by clicking the ADD/DROP option in the "ILA Seizure Workbook". The "Seize Manual Control" menu will

appear with the switches that have been seized highlighted. See Figure 55. The user uses the pull-rights to select the individual switches to add or drop from the switch seizure list. An item is selected if it is highlighted. Selecting a selected item will deselect it. The user can also drop a group.

When the user has finished adding and dropping switches, the user selects <seize selected items>, and the “ILA Seizure Workbox” appears with the list of updated switches and groups. Any switch that has been added to the list will have the “Run Calculate for Effects” prompt. The user will have to re-run calculate to get the updated results..

4.9.8 The Calculate Option

The “Calculate” button will invoke a function to calculate the effects of the switch seizures. The function will order the data based on the static and relative priority. It uses the static priority to put the switch request into four categories, and then uses the relative priority to order the switches within that category. After the switches have been ordered, the function will take the highest priority switch and try to determine if that seizure is legal. If the seizure is not allowed, the function will collect the reasons why the seizure is disallowed and store the information. The information consists of three elements. The first is the switch identifier, the second is the keyword :EFFECTS or :REASON, and the third is a string that contains the effects or reason. When the function is finished with all the switches it will send the data back to the ILA Seizure Workbox for display. See Figure 56.

The user decides if the effects are acceptable or not. If the effects are unacceptable, the existing switch information can be edited, a new switch can be added or a switch can be dropped. The changes must be run for effects again. This sequence of steps continues until the user is satisfied with the results.

When the results are acceptable, the user button clicks the “Done” button in the workbox to implement the seizure. MAESTRO is informed of the seizures and pseudo activities are created for the seized switches. The pseudo activities and the activity schedule are merged so that the seizures occur at the right time. If the user requests control of the switches immediately, then the switch will be under user control as soon as the Done button is clicked.

4.9.9 Manual Release

When switches are seized to perform a task, the switches remain under manual control for the user-specified duration. To release that switch back to the system prior to the specified time, the user can select “Release Manual Control”.

When the “Release Manual Control” option is selected, a menu is popped up that contains all the manually controlled switches. See Figure 57. Some of these entries will occur as individual switches while others will be groups. Selecting an item from the menu that is an

ADD/DROP Menu		
<<DONE>> <<CANCEL>> SINGLE SWITCHES		
GROUP A	Load Center	SWITCH
GROUP B	LC-01	Switch 01
GROUP C	LC-02	Switch 02
GROUP D	LC-03	Switch 03
GROUB E		Switch 04
GROUP F		Switch 05
GROUP G		Switch 06
		Switch 07
		Switch 08
		Switch 14
		Switch 15
		Switch 16
		Switch 17
		Switch 18
		Switch 19
		Switch 20
		Switch 21
		Switch 22

Figure 55: Add Menu with Load Center Pull Right

ILA Seizure Information Workbox						
<p>LC-01 SW: 02 NONE [00.00:00 - 99.99:99] [3 - 1] 1000W —D ON EFFECTS: ACTIVITY GLOVEBOX HOUSKEEPING DISPLACED BY SWITCH SIEZURE EFFECTED SWITCH(S): LC-01 SW: 02 LC-02 SW: 06 ACTIVITY STAR GAZER DISPLACED DUE TO POWER CONSTRAINT EFFECTED SWITCHE(S): LC-01 SW: 04</p> <p>LC-02 SW: 03 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON LC-02 SW: 04 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D OFF LC-03 SW: 21 STAR [00.00:00 - 99.99:99] [3 - 2] 1000W —D ON EFFECTS: ACTIVITY EXOBIOLOGY EXP DISPLACED DUE TO POWER CONSTRAINT EFFECTED SWITCH(S): LC-01 SW: 05 LC-03 SW: 03 ACTIVITY MICRO COUNTER DISPLACED DUE TO POWER CONSTRAINT EFFECTED SWITCH(S): LC-02 SW: 06</p> <p>LC-01 SW: 23 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON LC-03 SW: 02 CYCL [00.00:00 - 99.99:99] [3 - 3] 1000W —D ON REASON: LC-01 SW: 23 STATIC PRIORITY TO LOW LC-03 SW: 03 RELATIVE PRIORITY TO LOW</p> <p>LC-03 SW: 18 NONE [00.00:00 - 99.99:99] [3 - 4] 1000W —D ON EFFECTS: ACTIVITY STAR COUNTER DISPLACED DUE TO LOSS OF REDUNDANT SWITCH LC-03 SW: 18 EFFECTED SWITCH(S): LC-03 SW: 04</p> <p>LC-01 SW: 07 NONE [00.00:00 - 99.99:99] [3 - 5] 1000W —D ON REASON: LC-01 SW: 07 POWER REQUIREMENT EXCEEDS MAX ALLOWD ON 3K</p>	<table border="1" style="width: 100%; border-collapse: collapse; text-align: center;"> <tr> <td style="width: 20%; padding: 5px;">CALCULATE</td> <td style="width: 20%; padding: 5px;">HELP</td> <td style="width: 20%; padding: 5px;">ADD/DROP</td> <td style="width: 20%; padding: 5px;">DONE</td> <td style="width: 20%; padding: 5px;">CANCEL</td> </tr> </table>	CALCULATE	HELP	ADD/DROP	DONE	CANCEL
CALCULATE	HELP	ADD/DROP	DONE	CANCEL		

Figure 56: Updated ILA Seizure Information Menu

Manual Release Menu	
LC-01 SW: 04	
LC-01 SW: 16	
LC-02 SW: 03	
LC-02 SW: 06	
LC-03 SW: 19	
GROUP1	
GROUP2	
	SWITCHES
	LC-01 SW: 04
	LC-02 SW: 20
	LC-02 SW: 18

Figure 57: Manual Release Menu

individual switch will cause that switch to be returned to the system. Selecting a group will cause the switches in that group to be released back to the system.

When a switch is released back to the system, MAESTRO is notified that the switch is available. MAESTRO tries to reschedule any activities that might have been displaced due to the manual seizure of that switch.

4.10 The Hypothetical Scheduler Application

The hypothetical scheduler application was developed to allow a user to make changes to a currently executing schedule. It features a window to display, edit, create, and delete activities, subtasks, requirements, and powered equipment. This editor window functions in a manner similar to the main window on the activity editor. The hypothetical scheduler application also features a gantt chart window for displaying the hypothetical schedule in terms of its activities.

Upon activation of the hypothetical scheduler screen, the user is prompted with a "Time Finished?" workbox to allow the hypothetical scheduler to know when scheduling changes may start. This workbox will default to mission time plus 30 minutes.

4.10.1 Window Layout

To get to the hypothetical scheduler, the user selects the “Hypothetical Scheduler” menu item from the available applications. The hypothetical scheduler is designed for use in normal mode of operation. As with the other application screens, the hypothetical scheduler has an application menu and a scratchpad window. The scratchpad window provides the user with requested information or ancillary data to requested operations. The application menu contains the following functions:

- Edit
- Create
- Delete
- Save
- Add Activity
- Delete Performance
- Change Window of Time
- Start Over
- Help
- Implement

Application menu functions: Edit, Create, Delete, and Save operate in the same manner as the corresponding buttons on the Activity Editor screen. The only difference is that any scheduling object in the currently active schedule may not be edited. In other words, changing the parameters of a currently scheduled activity, subtask, requirement, or piece of powered equipment is not allowed. The Add Activity menu item allows the user to insert any available activity into the hypothetical schedule. The Delete Performance menu item allows the user to delete a selected activity performance from the hypothetical schedule. The Change Window of Time menu item provides the user with a way to expand the gantt chart window to a larger or smaller time window. The Start Over menu item supplies the user with the ability to reset the hypothetical scheduler screen to the presently active schedule’s present state. The Help menu option provides help to the user in the scratchpad region of the screen. The Implement menu item moves the hypothetical schedule into reality by making it the active schedule.

4.10.2 Adding an Activity to the Hypothetical Schedule

The "Add Activity" application menu button initiates the dialogue between a user and the system to effect the placement of a selected activity. The "Add Activity" option displays a menu of all activities defined in the SSM/PMAD system. Upon selecting an activity, the system computes a set of placement windows for the activity. If there are placement windows, the "Activity START Placement Selection" workbox appears. This workbox provides the user with activity placement capability. The user may click upon a time window or may edit the time fields directly. The user may change the priority by clicking on "Up Priority" or "Dn Priority" which will result in a new set of placement windows for the activity being calculated. The user may also click on "Calculate" after manually changing the priority field and compute a new set of placement windows for the priority specified. The user may also click on help which will pop up a help workbox. The user may exit from "Add Activity" by waving off the "Activity START Placement Selection" workbox or clicking cancel. To actually add an activity to the hypothetical schedule, the user must select a placement start time window and click done. Upon clicking done, the system will add the activity and its activity performance will appear in the gantt chart window.

4.10.3 Deleting an Activity Performance from the Hypothetical Schedule

The "Delete Performance" application menu button will cause a single selected activity performance on the gantt chart to be removed from the hypothetical schedule. If more than one gantt bar is selected in the gantt chart window, the system will pop-up an error workbox informing the user that only one activity performance may be deleted at a time. If no gantt bars are selected in the gantt chart window, the system will pop up an error workbox to inform the user that no activity performances have been selected. Selecting a gantt bar on the gantt chart window is as simple as clicking on the gantt bar causing it to highlight. Deselecting a selected gantt bar is accomplished in the same way by clicking on the selected bar. Once a single activity performance has been selected and "Delete Performance" is clicked upon, the system will remove or cut-off the activity performance selected. Note that it is not possible to remove an activity performance which has already been completed. Cutting off an activity performance occurs when the activity performance has already started and therefore must be stopped.

4.10.4 Implementing the Hypothetical Schedule

The "Implement" application menu button causes the following sequence of events to occur. The hypothetical schedule is saved. Then the hypothetical schedule is read in as the active schedule. Then the event lists and priority lists are updated and sent to the LLPs. And last, the FELES screen is made active showing the new hypothetical schedule as the active schedule.