# A TWO-LEVEL STRUCTURE FOR ADVANCED SPACE POWER SYSTEM AUTOMATION

Final Report

**Kenneth A. Loparo and Vira Chankong**
Principal Investigators

**Igor Vaks, Michael Talbot, and Jack Martin**
Graduate Students

Department of Systems Engineering
Case School of Engineering
Case Western Reserve University
Cleveland, Ohio 44106-7070

Submitted to

NASA Lewis Research Center
Power Technology Division
Aerospace Technology
21000 Brookpark Road
Cleveland, Ohio, Ohio 44135

ATTN: Dr. Karl Faymon and Mr. Jim Kish

Grant No.: NAG3-800
June 1987 - May 1990

## TABLE OF CONTENTS

# A Two -Level Structure for Advanced Space Power System Automation

Final Report

## 1. INTRODUCTION

This report describes the work accomplished during the three year period of the project from 1987 to 1990.

The overall goal of the project is to investigate the use of a two-level structure for fault detection and diagnosis of space power systems. The proposed hierarchical structure consists of model-based algorithmic procedures augmented by production rule type procedures at the lower level and more sophisticated problem solving and reasoning strategies at the higher level. This is deemed an appropriate framework for building fault detection and diagnosis schemes for space power systems for various reasons. Power systems are integral components of a spacecraft. They must operate reliably and efficiently. Any potential or incipient faults must be detected and diagnosed quickly and accurately so that appropriate remedial actions can be taken. Speed, accuracy, and the ability to access, integrate and interpret information from diverse sources are essential attributes of the power system fault detection and diagnosis system aboard a space craft. It is in response to such key attributes that the proposed two-level structure holds the most promise. In the events of disturbances and contingencies, quantitative data are gathered and analyzed with speed and precision at the appropriate level of technical detail by the lower level components. Priority loads are then determined and services to them maintained. The higher level components integrate results from the lower level with other qualitative information and develop modified strategies to improve the long term performance and survivability of the system. The intelligence of the system lies in this higher level where learning, sophisticated reasoning and creative strategy development take place. Thus the proposed two-level structure provides an ability to balance detailed analysis with innovative problem solving. The right mix will, of course, depend on the problem encountered.

Following the basic ideas outlined above, we identified the following tasks to be carried out during the three-year project period: 1)performing extensive simulation using existing mathematical models to build a specific knowledge base of the operating characteristics of space power systems; 2) carrying out the necessary basic research on hierarchical control structures, real-time quantitative algorithms, and decision-theoretic procedures; 3) developing a two-level automation scheme for fault detection and diagnosis, maintenance and restoration scheduling, and load management; and 4) testing and demonstration. The first task was carried out in Year 1. Tasks 2 and 3 occupied Year 2 and most of Year 3. Demonstrations of the developed system were carried out during the end of Year 3, and preparation to test the system on the Testbed at LeRC was made. The project ended before such testing could be completed.

The remainder of this report is organized as follows. Section 2 outlines the proposed system structure that served as a master plan for this project. Section 3 describes work accomplished. Section 4 provides concluding remarks and ideas for future work.

## 2. PROPOSED TWO-LEVEL STRUCTURE

Figure 1(a) and 1(b) show the proposed system structure under investigation in this work. The fault detection and diagnostic part contains six functional block which can be grouped into two stages. The fault detection and identification stage is composed of four blocks that determine what is wrong with the power system without specifying any corrective action. These may include state estimation, alarm processing, trouble call analysis and system diagnosis. The isolation stage, which is responsible for the synthesis of the corrective actions to rectify an abnormal operating situation, is performed in the system operating policy and remedial action selection blocks. This structure effectively integrates the myriad of quantitative and qualitative information which is, or will be, available. It should be noted that the aim of this tool is not to exclude the system operator: instead, it is to assist the operator in the performance of real-time monitoring and decision making. A more detailed description of these components and an illustration of how they work are given in Attachment 1. This research focused on fault detection and identification aspects of this structure as opposed to the isolation aspect.

In Figure 1(a), in addition to the six functional blocks mentioned above, there is a supporting block and a related functional block. According to our proposed two-level scheme, a knowledge-base is an important "supporting" block. The load scheduler is a related functional block which is closely linked to the system operating policy, and can thus provide useful information to and benefit from the fault detection and diagnosis blocks. Figure 1(a) shows typical information flow among various blocks. Figure 1(b) shows information flow from various blocks to the knowledge-base during the learning or knowledge acquisition phase. Attachment 1 illustrates typical information that may be available and how it might be used to perform fault detection and diagnosis.

## 3. WORK ACCOMPLISHED

Year 1:

It is clear from Figures 1(a) and 1(b) that the knowledge base is an integral component of our proposed system structure which includes the use of expert system techniques in the system diagnosis and load scheduler blocks. But since the design of space power systems was, and still is, on-going, such knowledge was not readily available. The first year work was therefore devoted to building the necessary knowledge-base through simulation.

At the time when this task was initiated, the best available mathematical model of a prototype system was the model of the Direct Energy Transfer (DET) spacecraft power system (Figure 2) developed by a team from Virginia Polytechnic Institute and State University [1]. The DET model consists of basic electric circuit models of a solar array, a simple battery charger and a discharger, a shunt regulator, cable/filter, and DC resistive load. Although computer programs written in EASY5 of the above components were available, it was decided to carry out the simulation in Turbo Pascal 4. This is due partly to accessibility and partly to greater flexibility of the input interface of Turbo Pascal. For example, variations in load profile, illumination, and/or temperature over the entire orbit (or planning horizon) could be easily entered and manipulated graphically through the screen interface. These features were particularly useful in the simulation exercise carried out subsequently.

Later in January 1988, a Rocketdyne report on Power Management Control for the Power System Testbed [2] became available. This report built on the DET model by refining existing component models and adding other components to make it more realistic in the context of space power systems. Components that were improved were switching shunt units for regulating power generated by solar arrays, the battery charger and discharger and their control units, and some power management distribution and control units. Additional components included a solar dynamic unit to provide an alternative form of power generation, DC and AC transmission lines, remote bus isolators (RBIs), and remote power control (RPC) ( protective devices in the instance of transmission line faults), DC-to-20MHz inverter, load converters (AC-to-DC and 20MHz-to-1.4KHz), and battery. Several of these components were still primitive and, according to the report, would be improved upon when more design information became available. For example, at the time when the Rocketdyne report was written, TRW and GD were still improving their designs of the DC-to-20MHz inverters.

The original simulation models, implemented in Turbo Pascal, were refined based on the Rocketdyne document. Component models of the battery charger and discharger and their control units were improved, the battery model was added, and other components were included.

Each component model described above, with the exception of the battery model, produced transient behavior which reached steady-state in milliseconds. The battery model, on the other hand, took several hours for charging and discharging operations between its minimum and maximum allowable charge levels. This marked difference in time scales, plus the fact that one complete orbit takes about 90 minutes, indicated the need for a hierarchical approach for integrating component models for simulating the operations of the prototype space power system during one or more orbits. We adopted a simple hierarchical structure as illustrated in Figure 3(a) which consists of models operating at essentially two time scales. The time scale used in simulating the component models was in milliseconds, whereas the time step used in the orbit level and battery models was in minutes or hours.

Figure 3(b) illustrates how the proposed orbit-level model works. Assume that at a particular time instance during the orbit (e.g. time A in Figure 3(b)), the system is operating at a steady state level. Assume also that, at time B, the power load profile drops sharply from one steady state level to another. The component models are executed at this time instant. But since the load power is not constant during C and D, no steady state may be found before D is reached. One obvious strategy is to run the component models continuously from C to D. This, however, may be computationally very taxing particularly if the time interval between C and D is long. To overcome this difficulty, we proposed the use of interpolation. The component models are first activated at C until a definite trend of all the relevant operating characteristics is obtained. The models are then reactivated at D to determine a new steady state level. A linear interpolation is then used where the change is occurring at a linear rate, the component models may be run at a few intermediate points and piecewise linear interpolation may be used to join these points.

There are thus two main functions of the orbit-level model: one is bookkeeping (keeping track of changes), and the other is interpolation. These two functions were incorporated in the model. Another possible function, which could be investigated in future work, is the interface with the PMAD and other control units (e.g. a load scheduling unit which was developed in a related research project).

All components described above were packaged into a complete simulator with a user's manual (See Attachment 2). This simulator was then used to perform the necessary simulation work. Attachment 3 shows typical simulation runs and results. The primary goal of this exercise was to build a specific knowledge base of the operating characteristics of space power systems for the subsequent work on fault detection and diagnosis. The simulation results were used to learn how each component and the system as a whole would operate under normal as well as faulted conditions, By "fault" in this case we meant not only an electrical fault but also an equipment failure and all other contingencies. A consultation meeting was convened between the project team and experts at NASA Lewis Research Center to discuss possible "faults" that might arise during the operation of space power systems. Since the design of the prototype system was, at the time of the meeting, (and still is ) on-going and not much was known about faults, it was recommend that the project team develop, in consultation with NASA personnel, a generic list of faults based on terrestrial experience and the Rocketdyne report. Such a list was compiled (See Attachment 2, p.6) and was used to perform simulation experiments. The results of these experiments highlighted the functional behavior of individual components as well as their interactions under normal and faulted conditions. This information was organized into a rule-based knowledge system. The use of this system in performing fault detection and diagnosis tasks is described in Attachment 3.

Year 2:

The DET model used to develop a knowledge base as described above was seen to be too simplistic to serve as a base for developing a useful and realistic fault detection and diagnosis system for space power systems. It was used because it represented the best model available at the time. Through our first year experience, we felt that until a relatively stable design of the space power system was available and many of the performance and failure characteristics of such system were properly understood, it would not be a well spent effort to investigate specific algorithms for an effective integration of the four fault detection and identification components as originally planned. On consulting our project manager at LeRC, the research plan for the rest of the project was slightly revised: we concentrated our efforts on the construction of a more powerful and more realistic device to help learn about faults of space power systems, and on how to combine such a device with appropriate quantitative algorithms to create a useful fault detector and diagnoser for a given design of a space power system. The key feature of this revised project goal is that whatever system we develop, it must be adaptable to any final design of the space power system.

Accordingly, we spent the remaining time of the project designing, developing, testing and demonstrating a computer-based system called FAULTS that can be used to:

* graphically BUILD a space power system to any desired configuration and any level of complexity for the purpose of analyzing reliability and power flow of the whole power system as well as its components;
* LEARN failure characteristics of the power system under various operating states of its components and subsystems by performing reliability analysis and power flow analysis;
* GENERATE recognizable fault patterns and TRAIN the system to recognize such patterns;
* PERFORM fault detection and diagnosis based on the training obtained.

Thus the system we sought to develop was a computer-aided tool that consisted of a SYSTEM BUILDER that would allow the user to easily build a space power system customized to any specific configuration, a KNOWLEDGE GENERATOR that would identify and learn about possible "patterns" of faults and their likelihood through **a system Reliability and Power Flow Simulator (RPFSim)** and **a fault patterns identifier/trainer**, and a FAULT DETECTOR AND DIAGNOSER (FD&D). Year 2 was spent on designing the system and on building the foundation for SYSTEM BUILDER and RPFSim. The final work on SYSTEM BUILDER and RPFSim, and the remaining pieces of the system (the fault patterns identifier/trainer and the patterns identifier/trainer and FD&D) were completed during Year 3 along with the testing and demonstration of the complete system.

The information used to build the system reliability and power flow simulator (RFPSim) along with the associated SYSTEM BUILDER is from a 1987 NASA document called the Power System Description Document (PSDD) [3] which represented the latest design information of the power system for Space Station Freedom at that time. The goal was to build a tool for assessing the reliability of the complete power system and its individual components and to simulate the power flow through the system. Reliability is a measure of the ability of the power system to deliver a given level of power after it operates for some time period without external disturbances or contingencies. Specifically, based on the design specifications on mean-time-between-failures (MTBF) of various components and on the proposed interconnections among components, the proposed simulator could calculate the probability that a given level of power will be delivered by the system during an operating period without external disturbances. Reliability of a system therefore depends on the design specifications of its components (MTBF, failure distributions, and mean-time-between-repairs MTTR), and the ages and interconnections of these components.

For the purpose of reliability analysis and the associated power flow simulation, the Reliability Block Diagram (RBD) is used as the underlying modeling method. Using RBD, the overall system is represented by interconnections of blocks (representing subsystems, assemblies, subassemblies etc.), where each block itself may be comprised of interconnections of lower level blocks, etc., down to the most primitive blocks (representing the smallest replacement units of the systems). Figures 4(a)-4(d) show, respectively, the reliability block diagrams of the overall power system of Space Station Freedom as envisioned at that time, the photovoltaic (PV) subsystem, the solar dynamic (SD) subsystem, and the Power Management and Distribution (PMAD) subsystem. The smallest units (components used in these RBDs) are the proposed Operating Replacement Units (ORUs). Each ORU, which contains electrical, mechanical and/or thermal components serving a common purpose, represents the smallest replaceable unit in the power system. MTBFs and failure characteristics of these ORUs are design specifications given in the PSDD [3]. Two or more ORUs make a subassembly, two or more subassemblies make an assembly, and two or more assemblies make a subsystem. A brief description of ORUs, subassemblies and assemblies (electrical or otherwise) is given in Attachment 4.

The RBDs shown in Figures 4(a)-4(d) are the results of several iterations of refinements with the advice and updated information provided by David Hoffman of the Systems Engineering and Integration Division, Systems Engineering and Analysis Branch, NASA/LeRc. The interconnections used in the RBDs of Figures 4(a)-4(d), and hence in SYSTEM BUILDER and RPFSim are the traditional series connections, parallel connections, star or delta connections, bridge connections, and any combinations thereof. Methods for analyzing reliability of blocks connected by these interconnection schemes are well documented in the literature. For the specific application to space power systems, it was necessary to include modified (partitioned) parallel connections to allow more accurate and convenient representations of solar arrays, battery packs, and thermal radiators. These connections are discussed in Attachment 5.

Associated with a block is a reliability description (e.g. MTBF, MTTR, and failure distributions) and an input-output description (e.g. input power - output power relationships during sunlight and eclipse cycles): the former is used for reliability analysis while the latter is used for throughput (e.g. power flow) simulation. In general, only the information for the primitive blocks need be provided by the user. The relevant description of each "derived" block are computed from the information in the lower level blocks.

There are two distinct features that set the RBDs in Figures 4(a)-4(d) apart from ordinary RBDs. First, the failure characteristics of some ORUs depend not only on their own failure characteristics but are also conditional on other ORUs. The ORUs with such properties are indicated in the diagrams by stars (*). Second, is the need to introduce a "partitioned" series connection. This feature arises only subsequent to simplification of a parallel connection (described in Attachment 5). An example is the series connection of ORUs 101 and 102: the output delivered by a component in this connection depends not only on its own failure characteristics but also on the failure characteristics of the preceding partitioned parallel connection.

As said earlier, due to the expected continual improvement and revision of the design of the space power system for Space Station Freedom, a key criterion in designing SYSTEM BUILDER and RPFSim is flexibility---an ability to quickly and conveniently accommodate changes and reconfigure the system.

Because of the desired flexibility and because of the distinct features of the

RBDs described above, a modular object oriented approach is adopted in designing SYSTEM BUILDER and RPFSim. The RBDs are viewed as simple series and parallel connections of primitive units. A primitive unit is either a normal block or a special block. A special block is either a group of ORUs connected by one of the modified parallel or series connections or a conditional ORU described above. A special computational rule is developed for each primitive unit in addition to general computational rules for ordinary series and parallel connections of primitive units. With this modular approach, it is therefore capable of assembling and reassembling ORUs into primitive units, and primitive units into a module or the overall system. Future design changes of the power system can thus be conveniently accommodated. RPFSim will also provide an effective tool to experiment with various system configurations for design purposes.

Based on these RBDs, SYSTEM BUILDER and RFPSim were developed and implemented in LISP on the TI Explorer. Object oriented programming on the TI explorer was chosen as the environment to develop this simulator. Because of the symbolic computing and graphic capabilities, an interface with FAULT DETECTOR AND DIAGNOSER was developed. SYSTEM BUILDER allows the user to create and modify RBDs through a graphic interface, adding yet another important attribute to enhance its potential.

In summary, SYSTEM BUILDER and RPFSim form a powerful device which can be used to:

* simulate system/equipment failures under normal operating conditions;
* study the effects of an equipment failure on the performance (failure) of other equipment and of the whole system;
* simulate system/equipment failures under various operating conditions, when combined with a model of external disturbances;
* study the performance of various design specifications (MTBF) of components and other design aspects;
* generate prior probabilities of failures for use in any Bayesian-based information fusion scheme for fault detection and diagnosis.

The first three items above will help improve our knowledge base (by acting as a "trainer") and generate data useful for testing the fault detection and diagnosis scheme to be discussed next.

Year 3:

The work in Year 3 was mainly to complete the development and refinements of

SYSTEM BUILDER and RPFSim, and to develop the fault patterns identifier/trainer and the FAULT DETECTOR AND DIAGNOSER (FD&D) itself. Testing and demonstrations of the complete system FAULTS were also conducted at LeRC.

The fault patterns identifier/trainer that was to be developed would take all available information relevant to the detection and diagnosis of faults, and combine these information in such a way that the result would be useful for detecting and diagnosing faults. For a general engineering system, typical information that may be available includes scientific understanding of how various components work individually and together as subsystems and the system as a whole; expert knowledge gained from experience through similar systems or situations; and past records of faulted states which show possible faulted conditions, their likelihoods and their associated observable symptoms. The first two types of information above could be purely qualitative or a mixture of qualitative and quantitative, whereas the third type is mostly quantitative. For the case of space power systems, good scientific knowledge and expert knowledge were minimal at the time of this research project due to the fact that the system was still in a preliminary design stage and no previous experience of similar systems existed. With SYSTEM BUILDER and RPFSim, the third type of information could be easily made available through a well planned simulation. Accordingly, the fault patterns identifier and the subsequent FD&D were developed based solely on this type of information. We left the question of how to incorporate qualitative information for future work, since a suitable procedure would depend very much on the specific form of the qualitative data available.

As used in our earlier Pascal version (see Attachment 3), the quantitative technique used to identify fault patterns based on a sample of simulated past records is a self-organizing (memory) network [4,5]. To make the description more concrete, we assume the following situation: A complex system consisting of many components (e.g. the space power system) is operating. Sensors (e.g. RPBs) are placed at strategic locations throughout the system to measure key variables (e.g. bus and load voltages and currents). The operator observing these sensor readings must determine whether there is a fault in the system, and if so, where. We assume that a sample of (simulated) past records have been generated using RPFSim. Each record consists of a set of sensor readings (e.g. measurements of voltages and currents at RPBs in the power system), and a description of the corresponding actual operating states of all the faulted ORUs, subassemblies, and assemblies. The basic idea is to use generated past records as a trainer to classify those records into groups based on some measure of "similarity": two records will be put into the same group if the sets of sensor readings in the two records are "similar". Recognizable patterns of faults within each group are then identified for further use in the fault detection and diagnosis. In this work, the measure of similarity between two records is measured by a weighted distance between the two records.

Specifically, record i---($s_{i1}$, $s_{i2}$,....,$s_{in}$) and record j---($s_{j1}$, $s_{j2}$,....,$s_{jn}$), n being the number of sensors used, are similar if the weighted distance $d_{ij}(w) = d_{ij}(w_1,w_2,...,w_n) = \Sigma w_k| s_{ik}-s_{jk}|^2$ is sufficiently small, where $w_k$ is the relative weight assigned to the reading at sensor k. A self organizing network was used to classify simulated past records. This is an iterative procedure for adjusting the set of weights $\mathbf{w} = (w_1,w_2,...,w_n)$ until distinct and meaningful groups are formed. The distance d of any pair of records in the same group is relatively small compared to the distance between any two records taken from two different groups. When a record consists only of two sensor readings ($s_1$, $s_2$), it can be represented as a point in a plane with $w_1s_1$ and $w_2s_2$ serving as the coordinates for a given set of weights $w_1$ and $w_2$. Thus the classification scheme involves adjusting the set of weights $w_1$ and $w_2$ so that distinct clusters of points are formed on the plane. In our implementation, locations of points (records) on the plane are shown at each iteration to show how points pull together to form clusters as the weights $w_1$ and $w_2$ change. By examining the actual operating states of ORUs, subassemblies and assemblies corresponding to each record in a cluster, fault patterns characteristic to that cluster can then be identified and recorded. This completes the description of fault patterns identifier/trainer implemented in our system.

To describe FD&D, we show how to use what we have described above to detect and diagnose faults. Assume that the operator observes a suspect set of sensor readings ($s_1$,....,$s_n$). The idea is to see how "similar" this observed record is to records in each cluster. This can be assessed by computing the weighted distance between the observed record to each of the records in a cluster. The average of these distances reflect how "close" or "similar" the observed record is to records in that cluster: the smaller the average distance, the more likely that the observed system is operating at a similar faulted state to the one identified for that cluster. By computing the average distance from the observed record to each cluster and ranking these distances from small to large, a list of possible faults can be presented to the operator in order of their likelihoods.

A complete system FAULTS (consisting of SYSTEM BUILDER, RPFSim, Fault Patterns Identifier/Trainer, and FD&D) was implemented on the TI Explorer platform and installed on a TI Explorer at LeRC. Two demonstrations of the system were given, one at CWRU and one at LeRC. More specific features of the system and how to utilize them are described in Attachment 6.

## 4. CONCLUDING REMARKS AND RECOMMENDATIONS FOR FUTURE WORK

We began the research project with the goal of exploring the use of qualitative reasoning (as in expert systems etc.) in combination with quantitative algorithms in a two-level structure to develop an autonomous system for detecting and diagnosing faults in space power systems. Lack of a definitive design and expert (qualitative) knowledge of the space power system forced us to concentrate our initial task on the building of a knowledge base. The fact that the design of the space power system was in the state of flux prevented us from building a knowledge base specific to any design. Instead, with agreement from our project manager at LeRC, we focused our research work on developing a computer-assisted tool that could help build a knowledge base, once the final design is in place. Such a tool could even be used for evaluating various design configurations to aid in the selection of the final design. Based on the type of information that could be generated by this knowledge base generator, a scheme for detecting and diagnosing faults was also developed. As it turned out, the system developed should be a powerful tool for both the space power system designer and for the developer of a fault detection and diagnosis system, perhaps more so than was originally envisioned.

The most obvious areas needing further work are the identification and acquisition of appropriate qualitative information (expert knowledge), once available, and how to effectively use such knowledge in combination with quantitative knowledge generated by the system developed here to detect and diagnose faults. Even with regard to the developed system, the work described in this report is by no means complete. Several key issues have not been addressed and require further attention:

* Maintenance and replacement options: Inclusion of the ability to include maintenance and replacement options in SYSTEM BUILDER and RPFSim would increase the utility and capability of the system tremendously;
* Sensor placement design: How many sensors should be used and where should they be placed?
* Training set design: How many "training" records should be simulated and how should they be selected? Also, how often and how this training set should be updated?
* Training method: Are there training procedures other than the self organizing network that should be used?
* Fault detection and diagnosis scheme: Are there other ways of using the information obtained to detect and diagnosing faults? In particular, if some qualitative data are also available, what data fusion scheme (Dempster-Schafer's theory of evidence, Bayesian rule, fuzzy logic, MYCIN-like rules, or other reasoning techniques under uncertainties) would be appropriate?

Other interesting issues concern the possible generation of qualitative knowledge through qualitative simulations.

## 5. CITED REFERENCES

1. Computer-Aided Modeling and Analysis of Power Processing Systems (CAMAPPS)- Phase I, S. Kim, J. Lee, B.H. Cho, and F.C. Lee, Research Report prepared for NASA/Goddard Space Flight Center, *NAGS-518*, 1986

2. Power Management Control For Power System Test Bed, Task Order 2 Final Report, Volumes 1 and 2, Rockwell Internation, Rocketdyne Division, *RI/RD87-171-1*

3. Power System Description Document (PSDD), NASA/LeRC, *SE-01*, September 1987.

4. *Adaptive Pattern Recognition and Neural Networks*, Y.H. Pao, Addison-Wesley, Reading, Mass, 1989

5. *Pattern Recognition: Statistical, Structural and Neural Approaches*, R. Schalkoff, Wiley , New York, 1992

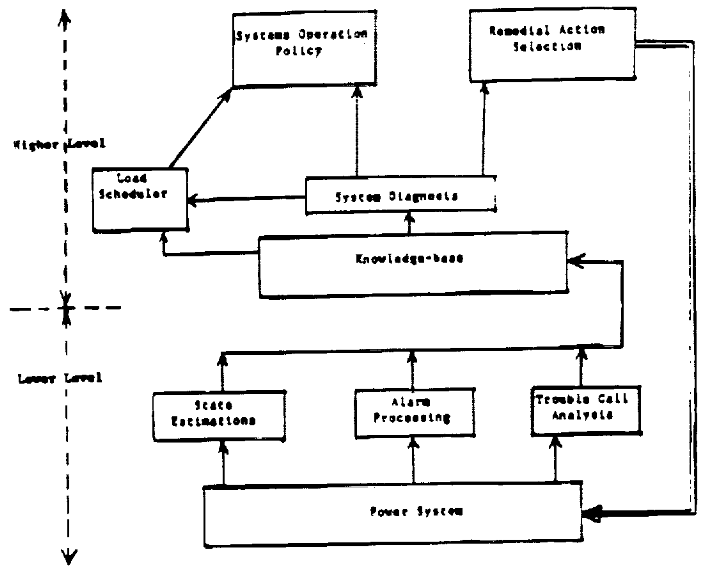


Figure 1(a) Proposed System Structure

——→ "Active" information flow for decision–making

⟹ Action

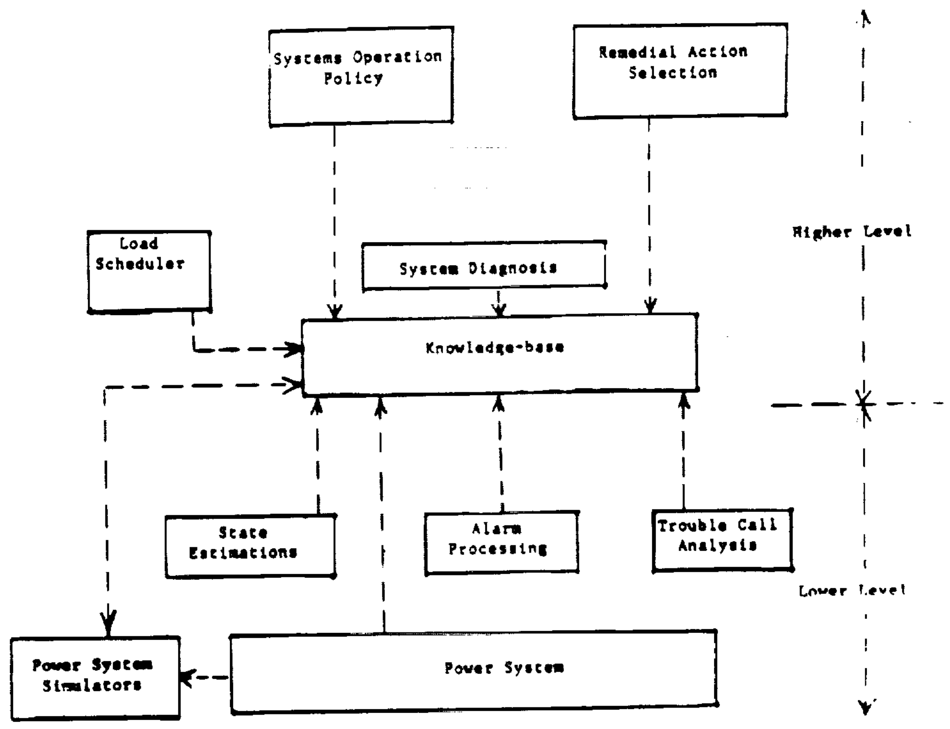


Figure 1(b) Proposed System Structure

- - - → "Passive" information flow for learning and knowledge acquisition and storage.

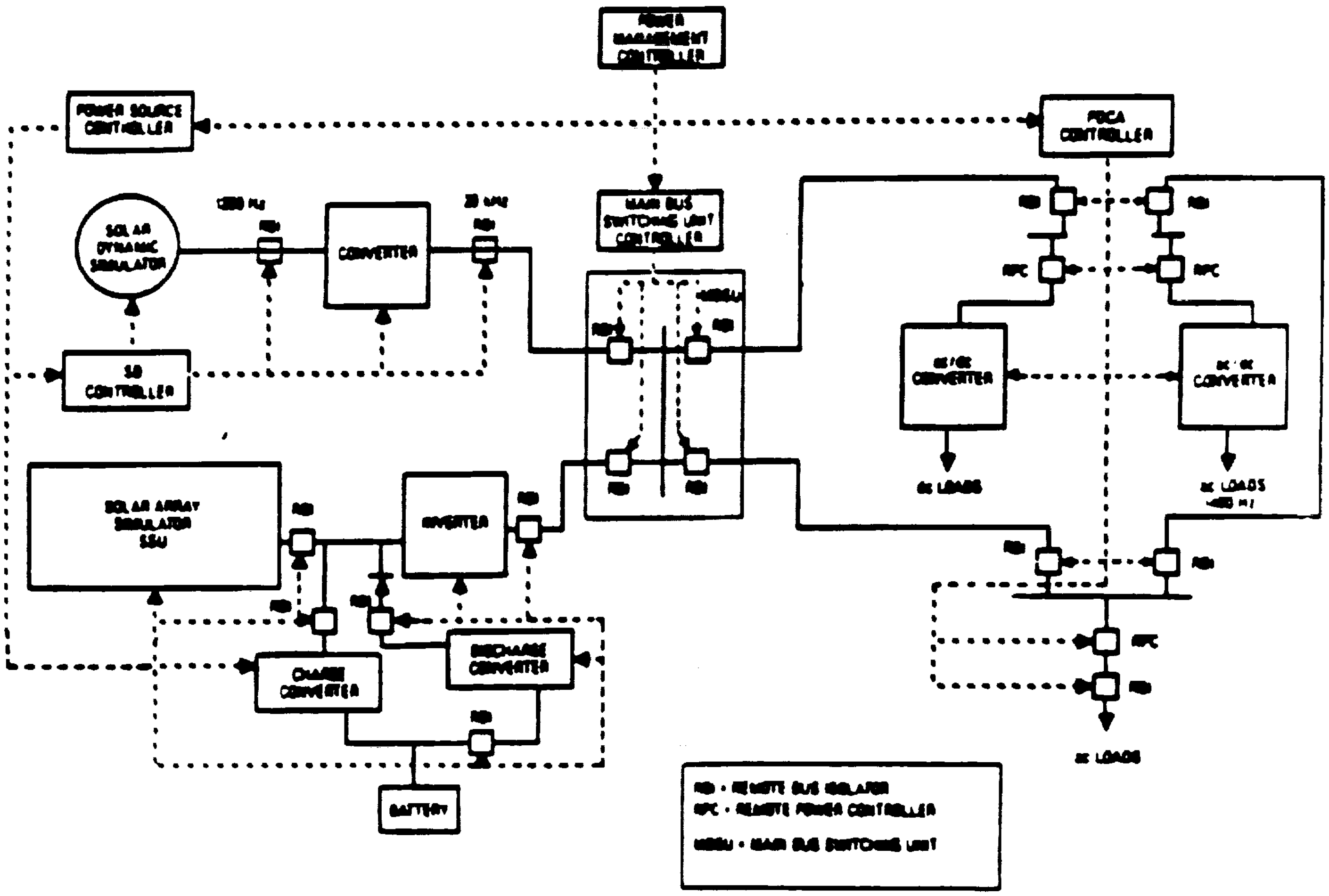

POWER MANAGEMENT CONTROLLER
POWER SOURCE CONTROLLER
PDCA CONTROLLER
MAIN BUS SWITCHING UNIT CONTROLLER
SOLAR DYNAMIC SIMULATOR
SD CONTROLLER
CONVERTER
RBI
RPC
MBSU
SOLAR ARRAY SIMULATOR SSU
INVERTER
CHARGE CONVERTER
DISCHARGE CONVERTER
BATTERY
CONVERTER
dc LOADS
ac LOADS
RBI - REMOTE BUS ISOLATOR
RPC - REMOTE POWER CONTROLLER
MBSU - MAIN BUS SWITCHING UNIT

Figure 2 Schematic of the Power System Test Bed

(Based on Direct Energy Transfer (DET) System)

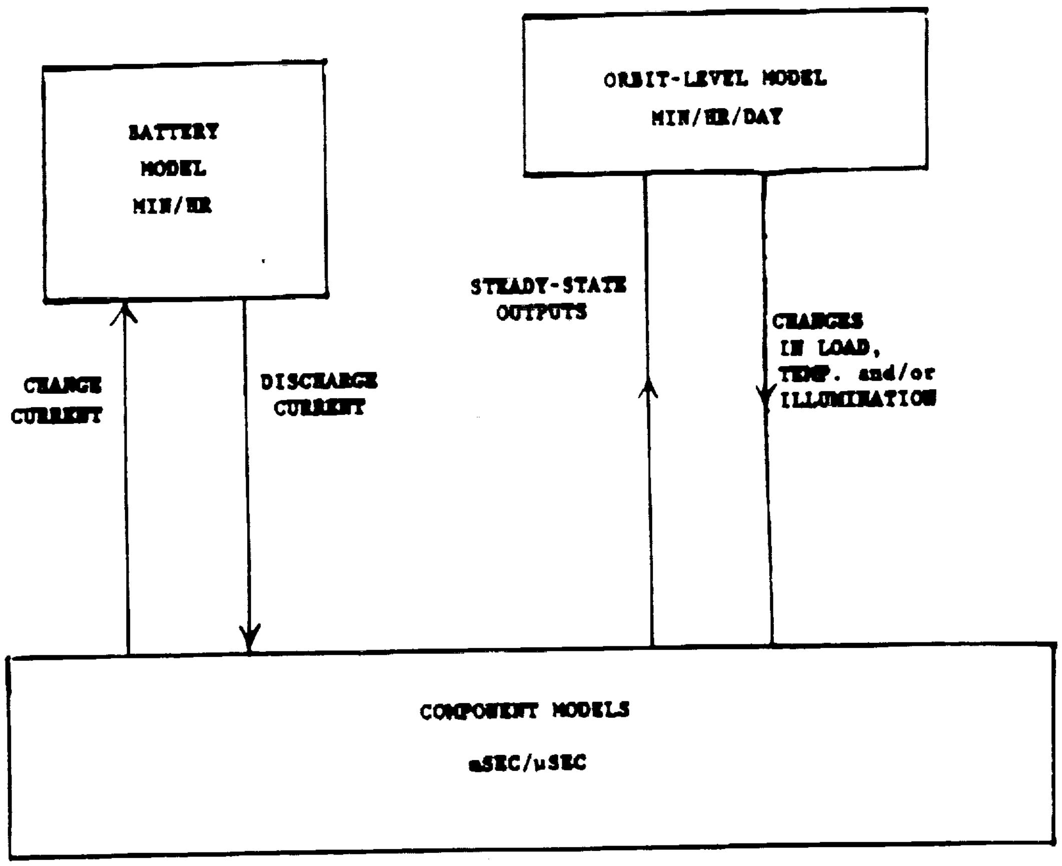


Figure 3(a) Two–level Model Integration

Profile of Power Load, Temperature or Illumination

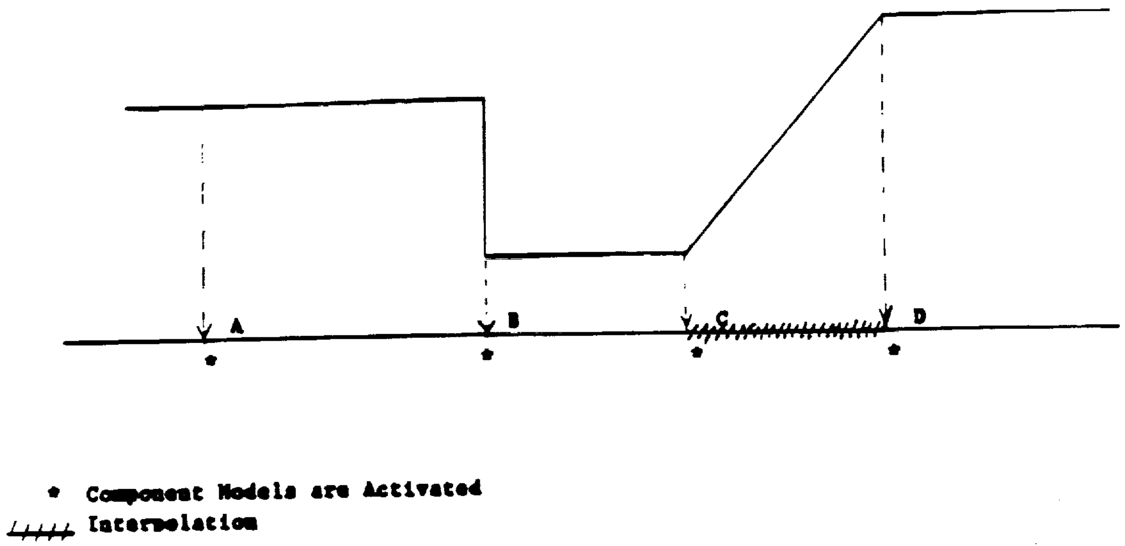


Figure 3(b) An Illustration of How Orbit–Level Model Works

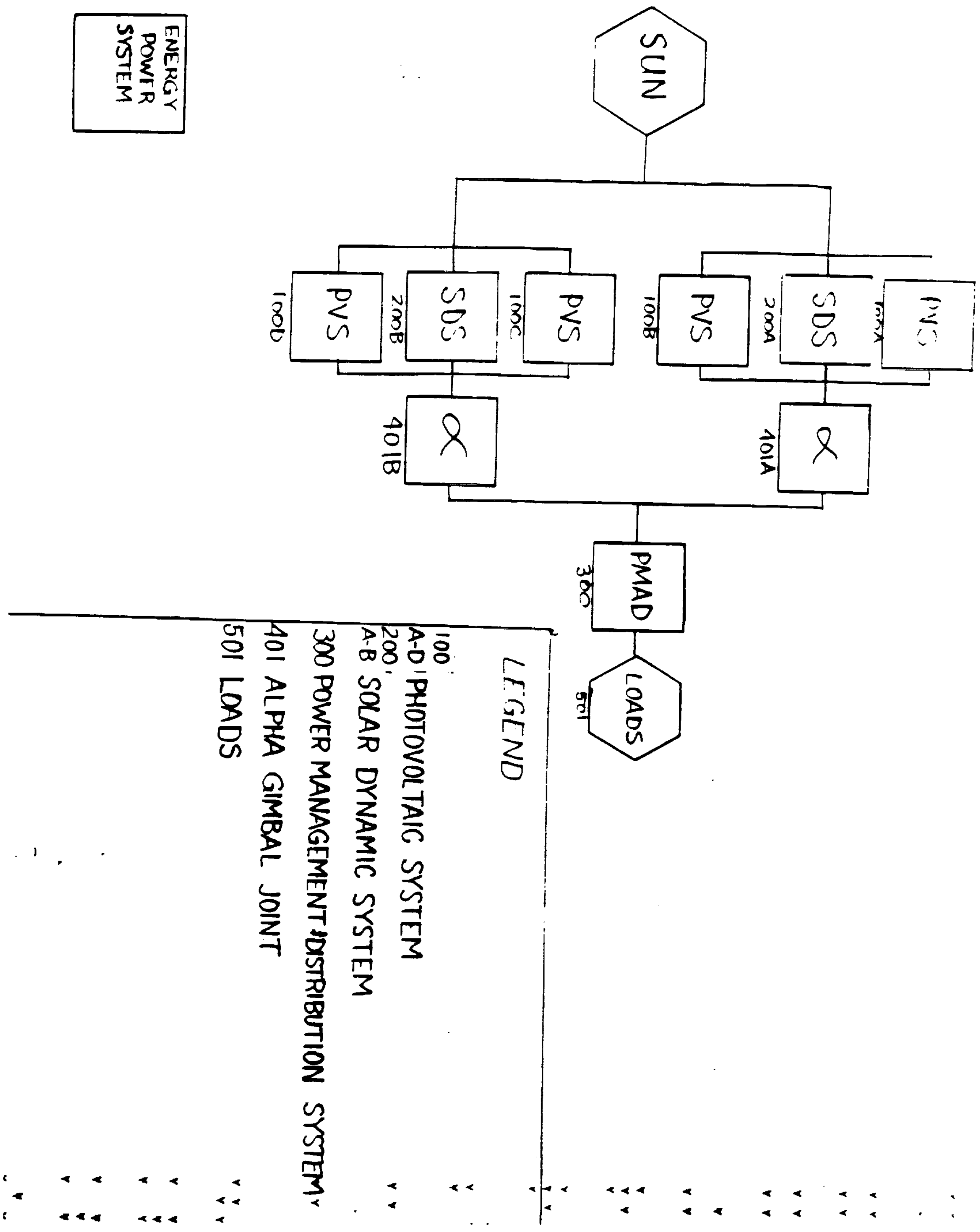


Figure 4(a) RBDs of Energy Power System (EPS)

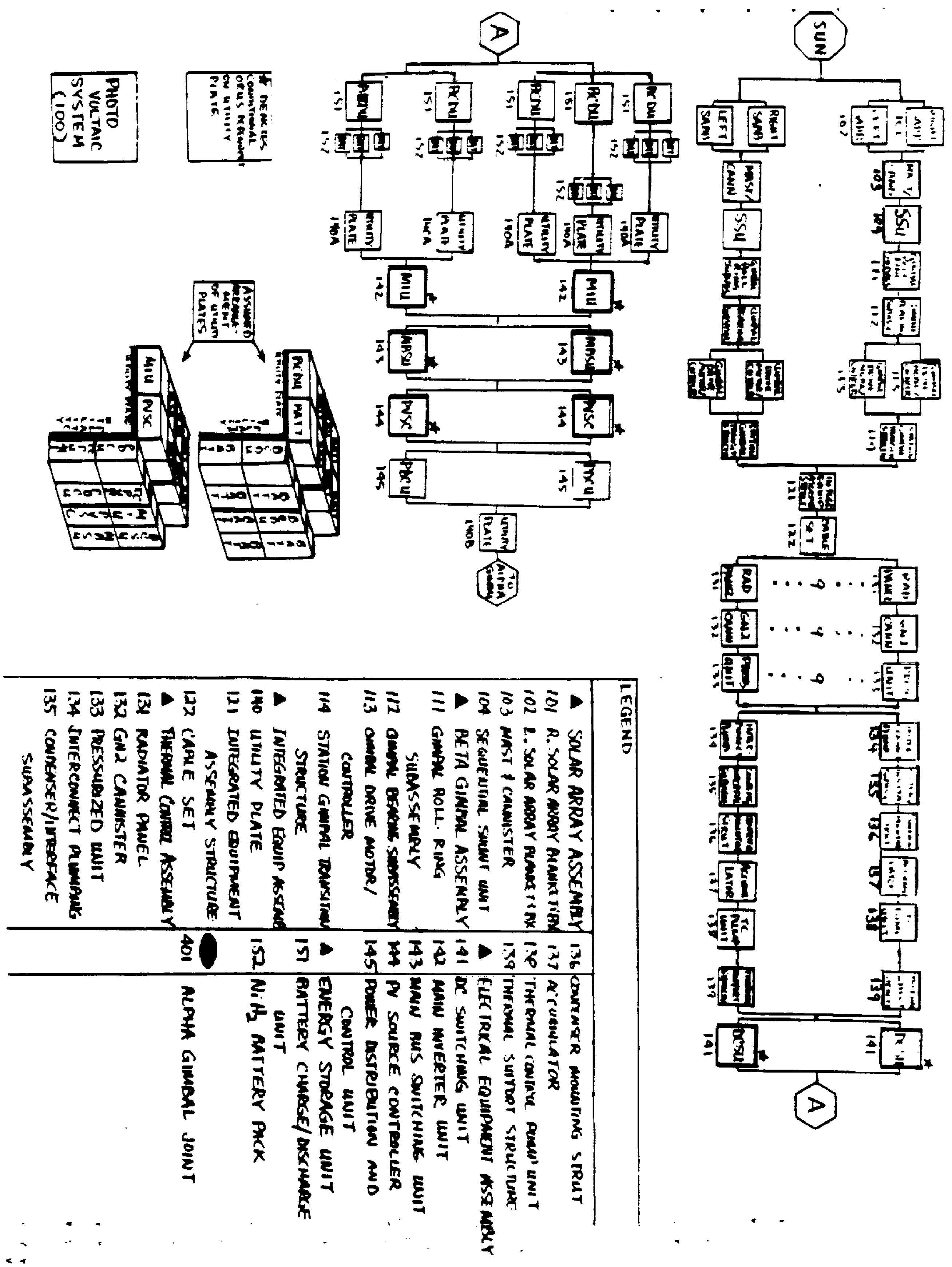
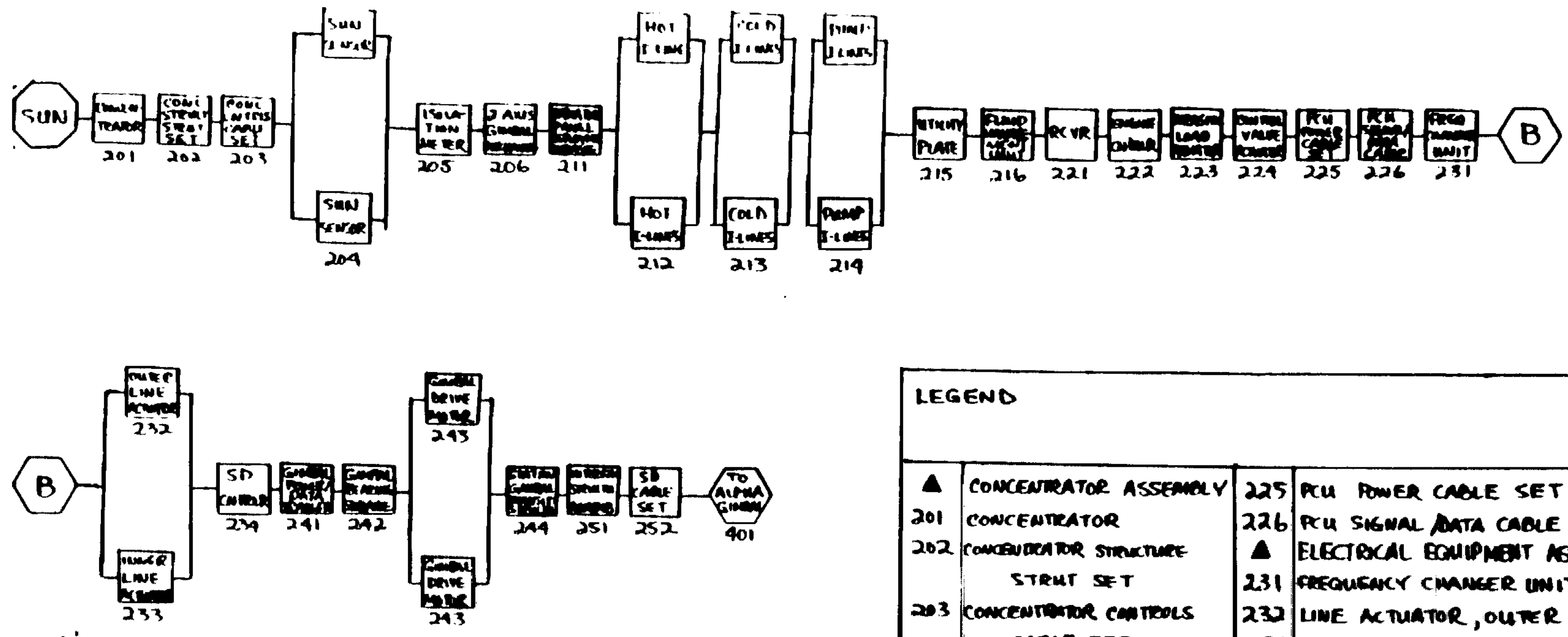

SUN
A
TO ALPHA GIMBAL
PHOTO VOLTAIC SYSTEM (100)
* DENOTES CONVENTIONAL ORUS MOUNTED ON UTILITY PLATE
ASSUMED ARRANGEMENT OF UTILITY PLATES
LEGEND
▲ SOLAR ARRAY ASSEMBLY
101 R. SOLAR ARRAY PLANETARIUM
102 L. SOLAR ARRAY PLANETARIUM
103 MAST & CANNISTER
104 SEQUENTIAL SHUNT UNIT
▲ BETA GIMBAL ASSEMBLY
111 GIMBAL ROLL RING SUBASSEMBLY
112 GIMBAL BEARING SUBASSEMBLY
113 GIMBAL DRIVE MOTOR/CONTROLLER
114 STATION GIMBAL TRANSITION STRUCTURE
▲ INTEGRATED EQUIP ASSEMBLY
140 UTILITY PLATE
121 INTEGRATED EQUIPMENT ASSEMBLY STRUCTURE
122 CABLE SET
▲ THERMAL CONTROL ASSEMBLY
131 RADIATOR PANEL
132 GN2 CANNISTER
133 PRESSURIZED UNIT
134 INTERCONNECT PLUMBING
135 CONDENSER/INTERFACE SUBASSEMBLY
136 CONTAINER MOUNTING STRUT
137 ACCUMULATOR
138 THERMAL CONTROL PUMP UNIT
139 THERMAL SUPPORT STRUCTURE
▲ ELECTRICAL EQUIPMENT ASSEMBLY
141 DC SWITCHING UNIT
142 MAIN INVERTER UNIT
143 MAIN BUS SWITCHING UNIT
144 PV SOURCE CONTROLLER
145 POWER DISTRIBUTION AND CONTROL UNIT
▲ ENERGY STORAGE UNIT
151 BATTERY CHARGE/DISCHARGE UNIT
152 NiH2 BATTERY PACK
● 401 ALPHA GIMBAL JOINT

Figure 4(b) RBDs of Photo Voltaic System (PV)



| LEGEND | | | |
|---|---|---|---|
| ▲ | CONCENTRATOR ASSEMBLY | 225 | PCU POWER CABLE SET |
| 201 | CONCENTRATOR | 226 | PCU SIGNAL/DATA CABLE SET |
| 202 | CONCENTRATOR STRUCTURE STRUT SET | ▲ | ELECTRICAL EQUIPMENT ASSEMBLY |
| 203 | CONCENTRATOR CONTROLS CABLE SET | 231 | FREQUENCY CHANGER UNIT |
| 204 | SUN SENSOR | 232 | LINE ACTUATOR, OUTER |
| 205 | ISOLATION METER | 233 | LINE ACTUATOR, INNER |
| 206 | 2 AXIS GIMBAL MECHANISM | 234 | SD CONTROLLER |
| ▲ | HEAT REJECTION ASSEMBLY | ▲ | BETA GIMBAL ASSEMBLY |
| 211 | RADIATOR PANEL DEPLOYMENT SUBASSEMBLY | 241 | GIMBAL POWER/DATA TRANSFER ASS |
| 212 | HOT INTERCONNECT LINES | 242 | GIMBAL BEARING SUBASSEMBLY |
| 213 | COLD INTERCONNECT LINES | 243 | GIMBAL DRIVE MOTOR |
| 214 | PUMP INTERCONNECT LINES | 244 | STATION GIMBAL TRANSITION STRUCTURE |
| 215 | UTILITY PLATE | ▲ | INTERFACE STRUCTURE/INTEGRATION HARDWARE |
| 216 | FLUID MANAGEMENT UNIT | 251 | INTERFACE STRUCTURE ASSEMBLY |
| ▲ | RECEIVER/POWER CONTROL UNIT | 252 | SD CABLE SET |
| 221 | RECEIVER | ● | |
| 222 | ENGINE CONTROLLER | 401 | ALPHA GIMBAL JOINT |
| 223 | PARASITIC LOAD RADIATOR | | |
| 224 | CONTROL VALVE ACTUATOR | | |

SOLAR DYNAMIC SYSTEM (200)

Figure 4(c) RBDs of Solar Dynamic System (SD)

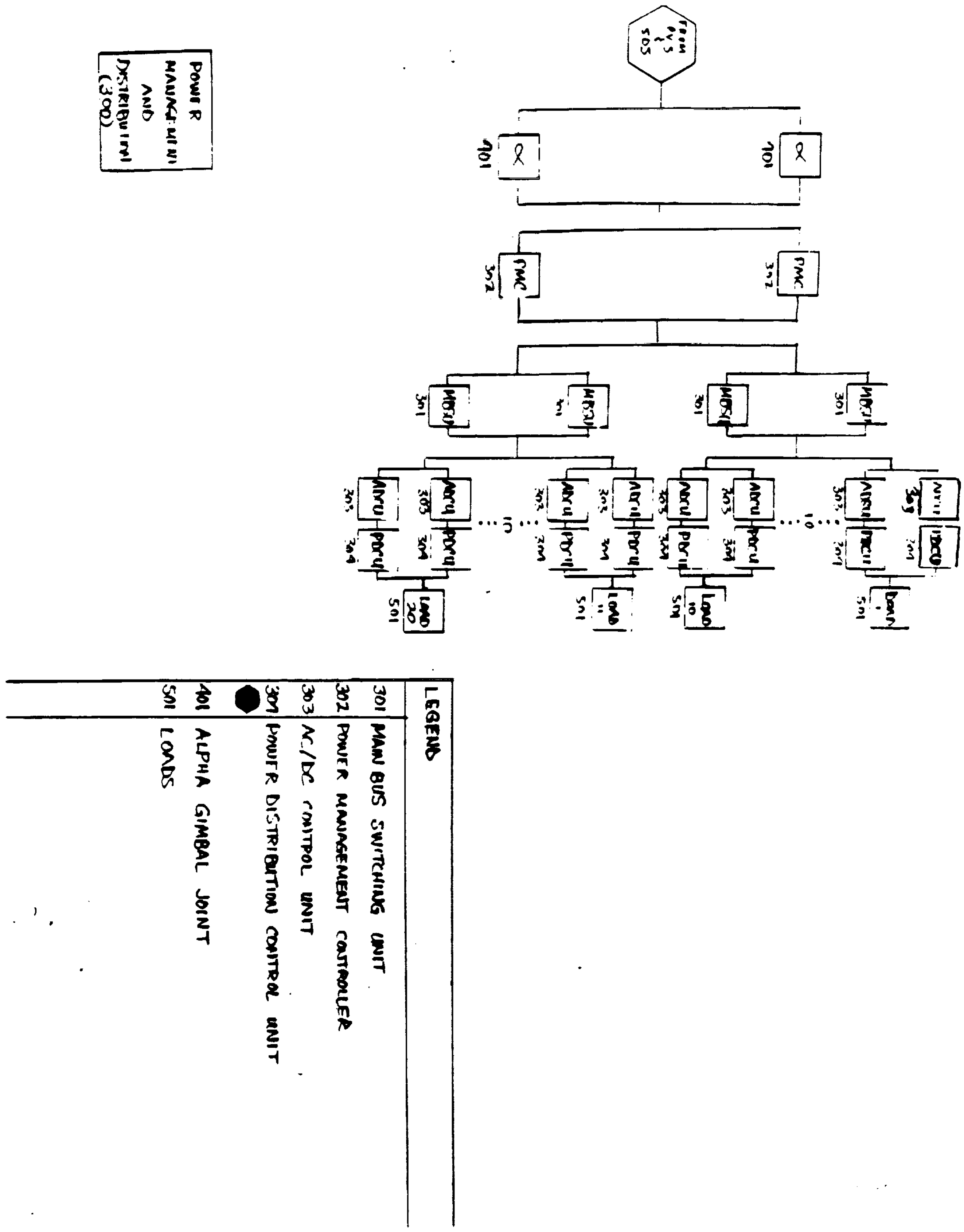

POWER MANAGEMENT AND DISTRIBUTION (300)
LEGEND
301 MAIN BUS SWITCHING UNIT
302 POWER MANAGEMENT CONTROLLER
303 AC/DC CONTROL UNIT
304 POWER DISTRIBUTION CONTROL UNIT
401 ALPHA GIMBAL JOINT
501 LOADS
MBSU
PMC
ACCU
PDCU
LOAD

Figure 4(d) RBDs of Power Management and Distribution (PMAD)

# ATTACHMENT 1

## An Illustration of Fault Detection And Diagnosis Using State Estimation, Alarm Processing, Trouble Call Analysis and Quantitative Data

With the assumption that a good knowledge base would be available for our project (which turned out not to be the case), specific issues that we originally planned to investigate were:

State Estimation:
The determination of the particular estimation method most suited for this application and the frequency with which the state should be estimated.

Alarm Processing:
The determination of alarm processing rules for combining elementary events and the length of time that the processor should wait for the occurrence of a companion event.

Trouble Call Analysis:
The determination of which equipment should be available to register complaints, the type of conditions about which equipment could complain, and how the complaints would be generated and handled.

System Diagnosis:
The determination of the knowledge representation, the method by which to combine quantitative and qualitative information, the resolution of conflicts, and the method of learning.

Although, the lack of a good knowledge base forced us to concentrate our efforts on the development of a KNOWLEDGE GENERATOR, some thoughts about the above issues were made at least during the initial stage of the project, with special emphasis on the system diagnosis component. These thoughts are summarized here. We first summarize our ideas on how these components work individually and collectively in a general setting, and illustrate these ideas with an example.

In the state estimation module, power system data including voltages, currents, and power system data including voltages, currents, and power measurements are processed to estimate unmeasurable internal states of the system and to detect bad data or faulted conditions. Here, a static or dynamic model of the power system, including the system interconnection topology is used to estimate the system state. A least squares estimation technique is commonly used.

Bad data or faulted conditions can be, for example, identified using statistical hypothesis testing techniques based on the properties of an "optimal" least squares estimator. Detection is usually determined reliably, diagnosis is a more difficult problem and generally requires some a priori modeling of faulted conditions. The system reliability simulator discussed previously, for example, can be used to serve this purpose. Techniques for diagnosis include innovations–based methods, Bayesian techniques, and observer–based model matching and generalized likelihood ratio techniques.

The alarm processing module correlates status information from various subsystems of the space power system with regard to critical conditions. For example, most physical devices such as regulators, inverters, etc. have constraints on internal variables which must be maintained for safe and reliable operation. As operating limits are exceeded, alarms are set and processed by this module. The alarm data is processed to identify possible sources of faults which are consistent with the observed data. This module is responsible for reporting alarm conditions to the operator,prioritizing and classifying the alarms according to type, common features, physical location, etc.

The trouble call analysis module handles complaints from various components in the system with regard to the current operating configuration and state. A typical example for the space power system could be that a load which was scheduled to receive electrical energy at a specified voltage and frequency level was not serviced properly, i.e., the quality of the service was unacceptable or the power was never delivered. During faulted conditions where abnormal behavior of the generating or distribution system is expected, information from the trouble call analysis module will be used to develop a prioritized list of possible fault conditions which correlate with the observed phenomena.

The system diagnostics module is responsible for fusion of the data received from the state estimation, alarm processing, and trouble call analysis modules.

The first step in the fusion procedure is to classify the current operating state. A convenient framework has been introduced in the work of T.E. Dyliaco for terrestrial power systems. Here, operating states are classified according to normal, alert, emergency and restorative. This decomposition of the operating states is determined by equality and inequality constraints which govern the steady state operation of the network. Modifications of these ideas to the space power system are required because of the differences in operating characteristics when compared to the terrestrial system.

If the operating state is classified as being normal, no additional action is required. If the operating state is not normal this is an indication of either the existence of a faulted condition on the network or the vulnerability of the current operating state and network configuration to possible contingencies. The alert condition refers to a situation where the system operating state and network topology are such that a probable (or possible) contingency could force the system into an undesirable operating state.

The emergency operating state refers to an operating condition where equipment malfunctions and faults on the network have impaired the ability of the power system to service the required loads. Automatic fault detection, isolation, and active control are required to maneuver the system to an acceptable operating state and configuration. It is in this operating mode where data from the three modules: state estimation, alarm processing, and trouble call analysis must be integrated in a timely manner. A hierarchical detection and isolation methodology is required where the functionality is decomposed according to temporal considerations. That is, algorithmic or simple rule–based procedures for integrating the information, e.g. Bayesian, Dempster–Shafer, and fuzzy set procedures, are implemented at the lowest level of the hierarchy. This information is then used for the automatic implementation of remedial actions to at least stabilize the system until the root cause can be determined. The next level of the hierarchy implements problem solving techniques to determine the cause of the fault and this information is used to reconfigure the system through the implementation of additional remedial control actions.

To illustrate the basic ideas of the approach, consider the problem of detecting and isolating a breaker failure in the system shown in Figure1.1. All circuit breakers are assumed to be initially closed. A fault is postulated on line A which would normally be cleared by opening breakers 2, 3, and 10; but, if breaker 3 failed to open, then bus 1 – section 2 would be cleared by a breaker failure scheme involving the additional opening of breakers 6 and 9. The problem reports that would be generated by this sequence of events are summarized in Table 1. Note that the "Event" column is not part of the problem report; it is included here only for clarification. Also note that the numbers given for the state estimation entries are in a per unit format for illustration purposes only and so do not correspond to a particular system.

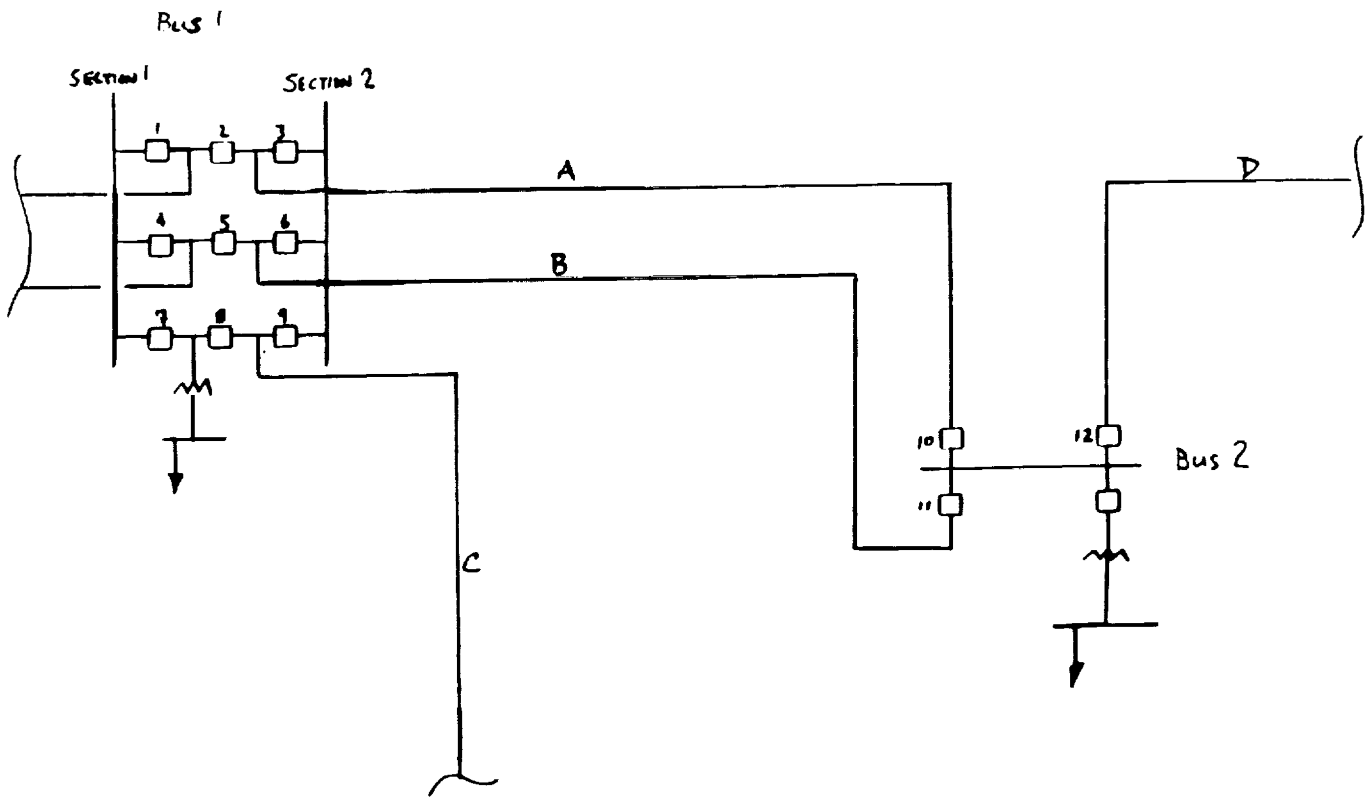


Figure 1.1 Example Power System

Table 1. Problem Report Summary for Breaker Failure Example

| Event | Location | SE | AP | TCA |
|---|---|---|---|---|
| Fault | Line A | I=10.0 | High current flow | none |
| | Bus 1 | V=0.65 | Low voltage | none |
| | Bus 2 | V=0.65 | Low voltage | none |
| | Load 2 | V=0.65 | none | Low voltage |
| Primary Protection | Line A | I=0 | High current flow Line deenergized | none |
| | Bus 1 | V=0.70 | Low voltage | none |
| | Bus 2 | V=0.75 | Low voltage | none |
| | Load 2 | V=0.70 | none | Low voltage |
| Backup Protection | Line A | I=0 | Line deenergized | none |
| | Bus 1 | V=0.95 | Breaker failure | none |
| | Bus 2 | V=0.92 | Voltage normal | none |
| | Load 2 | V=0.90 | Voltage normal | Voltage acceptable |

When the fault initially occurs, the short circuit current flowing in line A would result in the state estimation (SE) and alarm processing (AP) outputs shown for that location. The fault would depress the voltage at buses 1 and 2 and also at the load fed from bus 2, resulting in the reports shown under SE, AP, and trouble call analysis (TCA) for those locations. The fault would normally be cleared by the primary protection system by opening breakers 2, 3, and 10; But, if we postulate that breaker 3 does not actually open and is incorrectly reported open by the instrumentation, then the SE would remove line A from its model and report zero flow in the line; AP would take as input the opening of the three breakers and report the line as deenergized. However, current sensors on the line would still measure short circuit current, causing AP to continue to report the seemingly contradictory high flow alarm. Voltages would continue to be depressed, producing the reports under SE, AP, and TCA for the locations shown. The backup protection, the breaker failure scheme, would now react, tripping breakers 6 and 9 to isolate the fault. AP would combine these breaker trips with the previous three trips, using the relaying schemes in the knowledge base, to produce the resulting breaker failure suggestion at the bus 1 location. SE, AP, and TCA reports for all locations would show that all voltages had recovered.

The system diagnosis component takes the information in these problem reports and the fault types and component functional descriptions in the knowledge base to form hypotheses. These hypotheses are used, for example using the Dempster–Shafer theory of evidence technique, to determine the support for the hypotheses provided by the information within individual problem reports, among reports received in the same time frame, and among reports received from the same location.

Typical Alarm Processing Rules:

If FROM–BREAKER and TO–BREAKER of LINE X are OPEN then issue alarm LINE X DEENERGIZED.

If FROM–BROKER and NOT (TO–BREAKER) of LINE X are OPEN or NOT (FROM–BREAKER) and TO–BREAKER of LINE X are OPEN then issue alarm LINE X DISCONNECTED AT OPEN–BREAKER BUS.

Functional Description of Components in Example

Bus: distribute power; every component connected to bus is constrained to same voltage.

Transmission line: distribute power/conduct current; voltage at terminals related to current in line by certain model; model is dependent on line length in relation to wavelength.

Circuit breakers: connect/disconnect lines and loads to buses; two modes
–open: no current through breaker
–closed: no voltage drop across breaker

Relay circuits: (general) measure voltages and currents and send operate signals to breakers when certain combinations occur, dependent on specific type of relay circuit. Special types include overcurrent (send trip signals when current exceeds setpoint), backup (send trip signals when primary protection has failed), breaker failure (special type of backup relay circuit; sends trip signals to isolate bus section when primary protection fails to clear bus or line fault).

ATTACHMENT 2

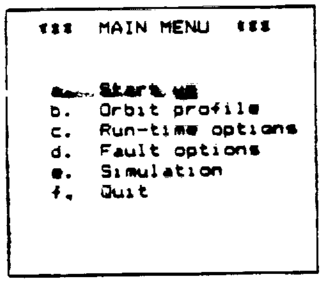
# Space Power System Prototype:* A Simulation Model

User's Manual

Vira Chankong, Kenneth A. Loparo, and Igor B. Vaks
Department of Systems Engineering
Case Western Reserve University
Cleveland, Ohio 44106

This note gives step-by-step instructions for initializing and running the simulation of the Space Power System model written in Turbo Pascal 4.0. Options available to the user are listed and explained below.

The program is called STATION.EXE. All other files in the diskette must also be in the same directory. The program is activated by typing STATION and pressing <RETURN>. At this point the user will be presented with the main menu:



Use <Up> and <Down> arrow keys to move from one option to another. Alternatively, you can type in a letter that corresponds to the option on the screen. For instance, to choose "Run-Time options" type in <c> or press <down arrow> twice and then <Return>. To choose the highlighted option press <Return>.

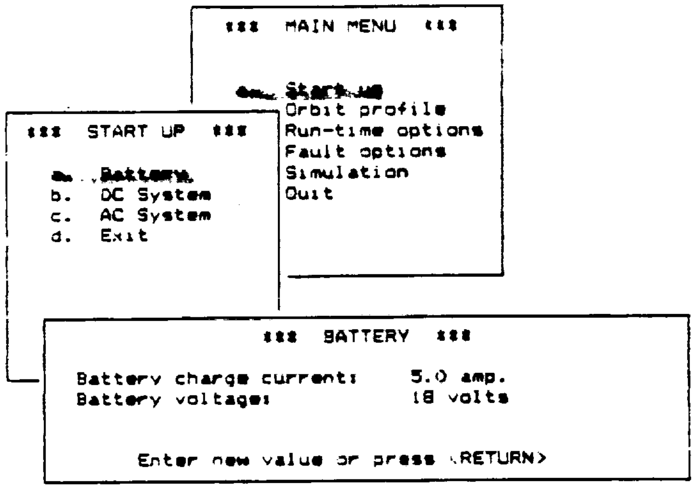
To run the simulation (Option e), the following items need to be provided:

1. Initialization of the battery (Option a)
2. Temperature and illumination profiles for the solar array module (Option b)
3. Power load profile (Option b)
4. Run-time parameters, e.g. time-step and simulation time horizon, etc (Option c)
5. Specification of faults for simulating the system response to faults (Option d)

*Developed under Grant NAG3-800 from NASA Lewis Center, Power Technology Division, Cleveland, Ohio.

Default settings are given for some of the above items. Default values can be entered by pressing <RETURN> when asked for input. A short description of each option in the main menu follows.

## A. Start up menu



```
Battery charge current:     5.0 amp.
Battery voltage:            18 volts

     Enter new value or press <RETURN>
```

This option allows the battery to be initialized (Option "Battery"), and the steady-state behavior (Option "DC System") and AC behavior (Option "AC System") of the system to be investigated under normal operating conditions (i.e. no changes internally and externally to the system).

Battery

The battery option should be run to initialize the battery to a specified battery voltage level. The default value of this voltage is 18 Volts. If the user want to choose another level, a value not exceeding 20 volts is recommended. Choosing a larger value may result in a very long charging time, since the charge/voltage curve levels off at around 19-20 volts.

The battery charging current must also be provided. The default value, which is highly recommended, is 5 amperes. Choosing a lower value will lengthen the charging time and a high value will most likely result in a voltage drop during low-current operation due to the charge/discharge characteristics of the battery model.

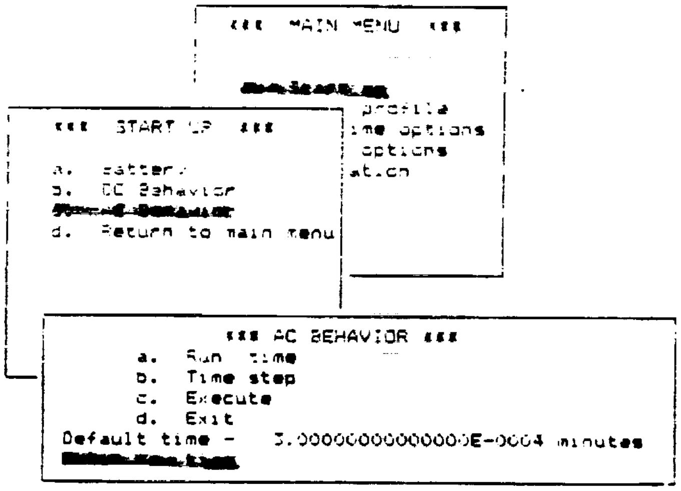
The initialized battery voltage level and the battery charging current can be entered through Option 'a' of the Start-up menu. If this option is not run, the battery voltage and the battery charge level will be zero.

DC Behavior

When there are no internal and external changes in the system, the steady-state DC behavior can be examined using this option.

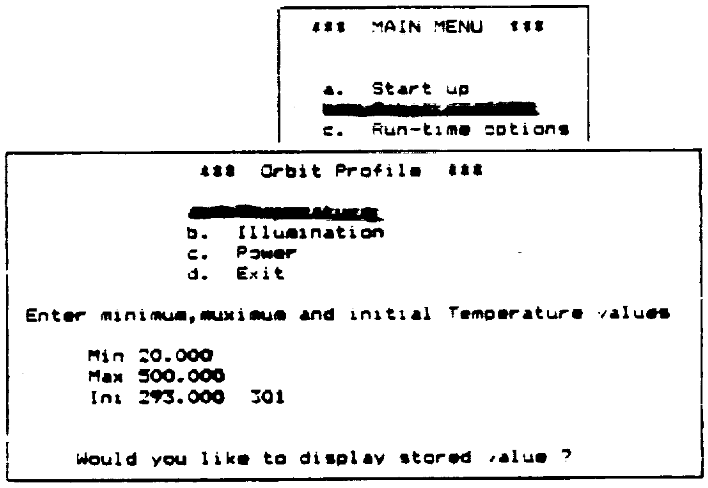
AC Behavior

This option allows the user to view the start-up AC characteristics of the system. Before running this, two parameters have to be specified. Run-time is the length of the time horizon in seconds for the simulation run and time step is the integrated time step (in seconds) used in the computations. The default value for run-time is 3E-4 seconds and for the time step is 1E-7 seconds. The default run-time will be used if option <a> is not activated. Likewise, the default time step will be used if option <b> is bypassed.



```
*** MAIN MENU ***

[illegible]
  profile
  ime options
  options
 at.cn

*** START UP ***

a.  Battery
b.  DC Behavior
[illegible]
d.  Return to main menu

*** AC BEHAVIOR ***
     a.  Run  time
     b.  Time step
     c.  Execute
     d.  Exit
Default time -   3.0000000000000E-0004 minutes
[illegible]
```

## B. Orbit Profile



```
*** MAIN MENU ***

a.  Start up
[illegible]
c.  Run-time options

*** Orbit Profile ***

[illegible]
b.  Illumination
c.  Power
d.  Exit

Enter minimum,muximum and initial Temperature values

     Min 20.000
     Max 500.000
     Ini 293.000  301

     Would you like to display stored value ?
```

This option allows the user to specify temperature, illumination and power profiles for a simulation. The first two profiles are the characteristics of the solar array module, whereas the third profile is for the load. Use arrow keys or press an appropriate letter key to move the cursor to highlight the desired option and press <RETURN>. In response to "Enter minimum, maximum and initial <Parameter name> values", type in the values of the lower and upper limits of the profile to be entered as well as the initial level to start the profile. The default values of these quantities will be shown on the screen. To choose any of these default values, press <RETURN> when asked for input. The rest of the profile will be entered graphically. At this point, the user will be prompted "Would you like to display stored values?". If you have a profile already stored on the disk and wish to rewiew and/or change it, press <y>. Otherwise, press <RETURN> or any other key.

The profile is generated graphically by moving the arrowhead from left (initial value) to right. The speed of the arrowhead can be controlled by pressing keys <1> (slowest) through <9> (fastest). The key <0> corresponds to the stationary position. When the arrowhead starts moving forward, its direction can be changed by the arrow keys and PgUp, PgDn, F1 and F2 keys:

| Key | Direction of Movement |
|---|---|
| Arrow keys | As indicated on the key |
| PgUp | 45 degrees upward |
| PgDn | 45 degrees downward |
| F1 | Gradual change upward |
| F2 | Gradual change downward |

When finished with this option, the new input data is stored on the system disk and doesn't have to be re-entered next time the program is used.

## C. Run-Time Options

This option allows the user to modify the Run-time parameters and choose different ways to run the simulation.

a.

This option gives the user a choice to view the data after every change in external conditions (illumination profile, etc) or internal conditions (faults,change from charge to discharge mode, etc) conditions. This "display" mode is the default mode. The alternative mode is to run the simulation continuously for the complete simulation time horizon without display. Pressing <a> switches from one mode to the other.

For example, if the program is running in the default (display) mode, pressing <a> causes the program to run in the no-display mode. Pressing <a> again switches the simulation back to the display mode.

b.

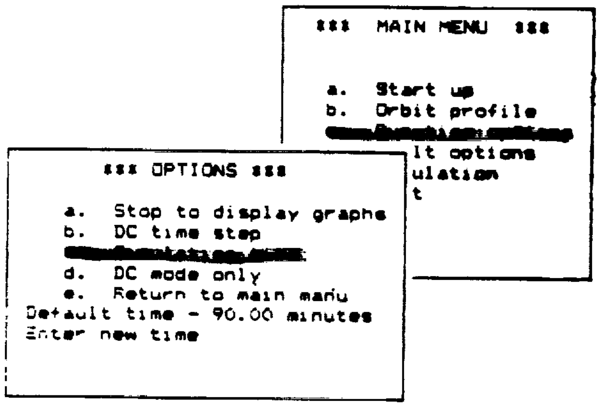
The default time step for the DC mode is one microsecond. Option b. can be used to change the default setting. However, a smaller step size will slow down the execution time even though it may result in a more accurate simulation. Increasing the time step will speed up the simulation but may make the system unstable.

c.

The default value of the simulation run-time is 90 minutes. You can change this by choosing <c> and entering the desired simulation time when prompted. When this is done, be sure to modify the orbit profiles (Option b in the main menu) accordingly.

d.

The simulation can be run in either DC mode (default) or DC & AC mode. Pressing <d> in this menu switches from one mode to the other.



```
*** MAIN MENU ***

a.  Start up
b.  Orbit profile
[redacted]
lt options
ulation
t
```

```
*** OPTIONS ***

a.  Stop to display graphs
b.  DC time step
[redacted]
d.  DC mode only
e.  Return to main manu
Default time - 90.00 minutes
Enter new time
```

D. Fault menu

The program allows various kinds of faults to be simulated. These faults are listed in the following table.

| Fault | Comments |
|---|---|
| Solar Array | |
| Panel breakedown | A specified % of solar panels fail |
| DC loads | |
| Leak | Resistance of R in the LRC circuit is decreased |
| Switch | The on/off switch in the power stage is stuck in the wronf position |
| Sensor | Wrong value is reported by the voltage and/or current feedback modules |
| AC load | The Resistor value in the inverter module is decreased, resulting in power loss |
| Battery | |
| Charger | The charger switch is stuck in either ON or OFF position |
| Discharger | same as above |
| Cells | A specifyied % of battery cells fail |
| RBI error | The specified RBI is stuck in an incorrect state (e.g. OFF instead of ON, etc.) |

This option allows the user to configure form any desired fault pattern. First the time (in minutes) at which a particular fault occurs is entered. Then the type of fault is chosen from the given list. Finally when appropriate, the severity of fault is entered. Multiple faults occuring at the same or different times can be easily formed. For example, if we wish to simulate the following faults:

A. 20% Solar panel breakdown at the 20th minute
B. Moderate equipment short at the 20th minute, and
C. Degradation of battery cells resulting in the drop of battery voltage to 15 volts at the 50th minute.

The following sequence of keys are pressed:

From the main menu, choose <d>
From the Fault Schedule menu, choose <a>
In response to prompt, type in 20 (meaning that a fault will occur at 20 minutes)

A menu of possible faults will appear on the screen.
In response to "Chose <1> through <5>" press <1>
In response to "Choose <a> through <b>" press <a>
In response to "Enter % power loss" type in 20
(meaning that there is 20% power loss due to panel breakdown.

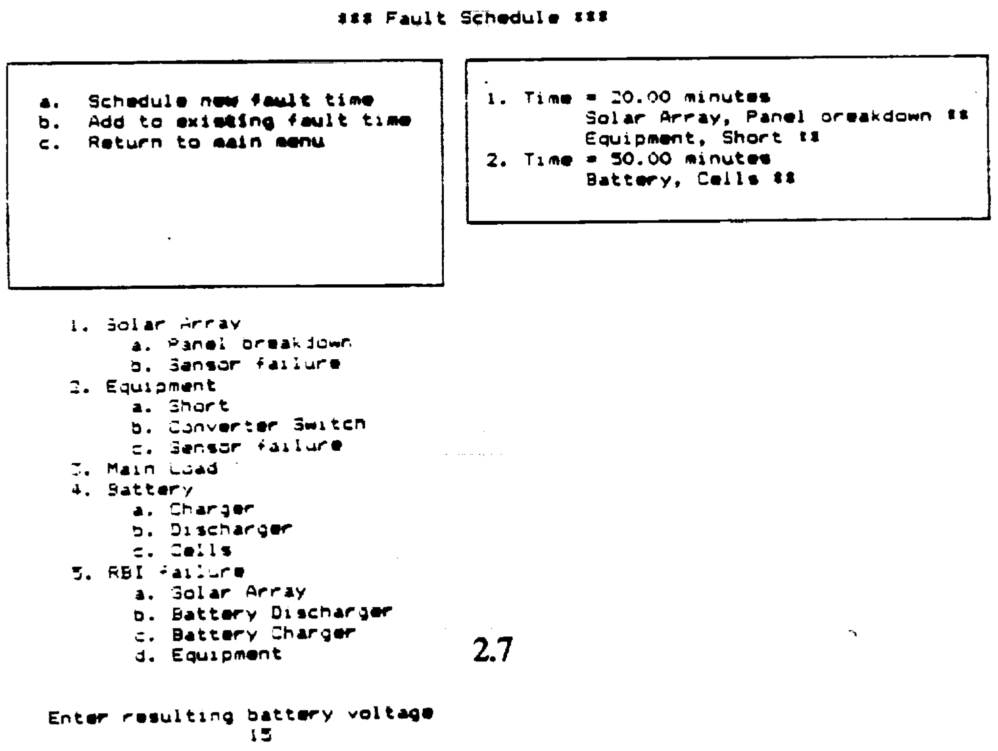
This completes the specification of Fault A. and you are returned to the Fault Schedule menu. An entered fault should be listed on the right part of the screen.

To specify fault B, press <b>.
In response to "Choose <1> through <1>" press <1>
In response to "Choose <1> through <5>" press <2>
In response to "Choose <a> through <c>" press <a>
In responce to "Choose one", press <b>

This completes the specification of Fault B which occurs at the same time as Fault A. At this time the original Fault Schedule menu should return to the screen.
Finally, to specify fault C, press <a>
In responce to "Enter time between 0 and .." tipe in <50>
(meaning that fault C occurs at time 50 minutes.)
In responce to "Choose <1> through <5> press <4>
In responce to "Choose <a> through <c> press <c>
In responce to "Enter resulting battery voltage" type in 15 and press <RETURN>

This completes the specification of all faults. Again, the first Fault Schedule menu should return to the screen. Press <c> to return to the main menu.

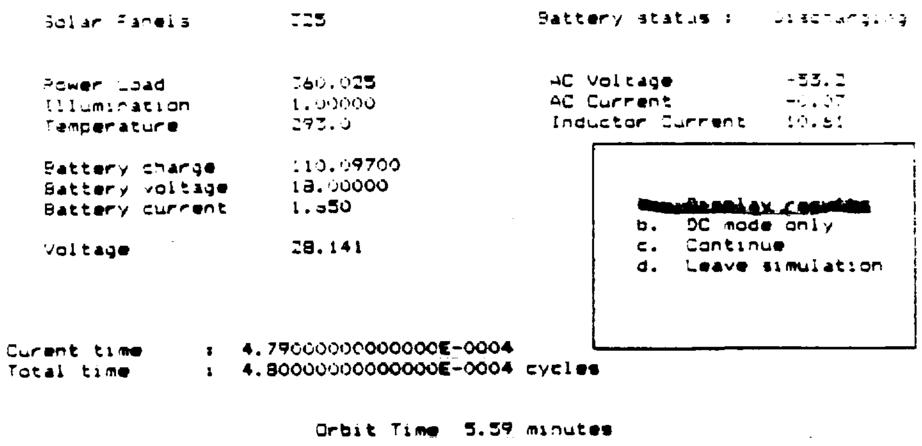
Note : At this time faults can not be erased once enterred. If you make a mistake, the only way to correct it is to restart the program. This inconvenience will be corrected later.



```
                        *** Fault Schedule ***

 a.  Schedule new fault time         1. Time = 20.00 minutes
 b.  Add to existing fault time             Solar Array, Panel breakdown **
 c.  Return to main menu                    Equipment, Short **
                                     2. Time = 50.00 minutes
                                            Battery, Cells **

     1. Solar Array
          a. Panel breakdown
          b. Sensor failure
     2. Equipment
          a. Short
          b. Converter Switch
          c. Sensor failure
     3. Main Load
     4. Battery
          a. Charger
          b. Discharger
          c. Cells
     5. RBI Failure
          a. Solar Array
          b. Battery Discharger
          c. Battery Charger
          d. Equipment

 Enter resulting battery voltage
             15
```

## E. Simulation

This command executes the simulation. If the continuous simulation option was chosen (<a> from run-time options menu) then the program will run without interruption for the length of the orbit. Otherwise, after every change the user will be prompted with the following menu:



```
Solar Panels         225              Battery status :   Discharging

Power Load           360.025           AC Voltage         -53.2
Illumination         1.00000           AC Current         -0.37
Temperature          293.0             Inductor Current   10.51

Battery charge       110.09700
Battery voltage      18.00000
Battery current      1.550                  a.  Display results
                                            b.  DC mode only
Voltage              28.141                 c.  Continue
                                            d.  Leave simulation


Curent time     :  4.79000000000000E-0004
Total time      :  4.80000000000000E-0004 cycles

                      Orbit Time  5.59 minutes
```

At this time the user can display results from the previous run on the screen (Option <a>), switch between DC mode and DC & AC mode (Option <b>), continue the simulation (Option <c>), or exit to the main menu (Option <d>).

Displaying the results (Option <a>):
After choosing this option the user will be presented with the list of DC variables that can be displayed. Select one and enter the minimum and maximum values for that variable. Ideal values for the variables are given below:

| | | |
|---|---|---|
| Bus Voltage | : | 28.144 |
| Bus Current | : | 15 through 40 |
| Battery Current | : | -10 through 20 |
| Shunt Current | : | 0 through 10 |
| Equipment 1 V | : | 20 |
| Equipment 1 I | : | 10 |

Choosing option <h> from the display menu allows the user to view AC variables. This option is available only if running the entire simulation (AC and DC components). The ranges of value for AC variables are given below:

| | | |
|---|---|---|
| AC Voltage | : | -77 to 77 |
| AC Current | : | -0.5 to 0.5 |
| Inductor Current | : | 0 to 20 |

## ATTACHMENT 3

### An Illustration of How to Use Pascal Version of Fault Detection and Diagnosis System Based On the DET Model

To illustrate a more detailed implementation of the Pascal Version of the Fault Detection and Diagnosis system developed in Year 1, consider the problem of detecting and diagnosing faults on the DET system shown in Figure 2 in the text. Assume that are a state estimation procedure and an alarm processing procedure are in place to monitor the system operating state and to report any abnormal operating conditions. The specific approach described here does not explicitly consider trouble call analysis, but an extension to include such a function should be easy.

Suppose further the following information is also available:

(i) Past records of faulted states which show not only the faulted conditions but also the associated symptoms of such conditions. This type of knowledge is usually referred to as data or empirical knowledge which is quantitative in nature. It is stored in a data base. Instead of or in addition to historical records, this type of knowledge can also be generated by simulation if a good simulation model of the system is available.

(ii) Good scientific understanding of how primitive components of the system work individually as well as collectively. This is theoretical or scientific knowledge.

(iii) In addition, (i) and (ii) may be combined with judgmental knowledge gained from experience with this or other related systems to form what we call procedural (rule-based or inferencing) knowledge. This involves the understanding of the functions of components or groups of components as opposed to detailed scientific properties or behaviors. This type of knowledge is more readily suitable for problem solving or reasoning and is usually qualitative in nature. It is stored in a knowledge base.

In our example, each type of knowledge was generated by using the DET simulation model developed in Year 1. In Particular (i) and (ii) were created by running the DET model several times as part of the experiments performed in Year 1.

Given the existence of a data base containing past or generated records in (i), the knowledge base containing procedural knowledge (rules) in (ii) and (iii), and the information from the state estimation procedure and the alarm processing procedure reporting abnormal operating conditions, how can the faulted conditions be diagnosed? It is clear that an approach that combines both quantitative and qualitative reasoning is needed. This is consistent with the two-level "structure" theme of this research.

In the structure described below, a memory network is used to process quantitative information, a Prolog rule-based system is used to process qualitative information, and backward and forward chaining procedures are used as a means for fusing information processed by the two procedures. This is illustrated in Figure 3.1. We show below how this structure works, and how attractive and effective it is for this type of problems.

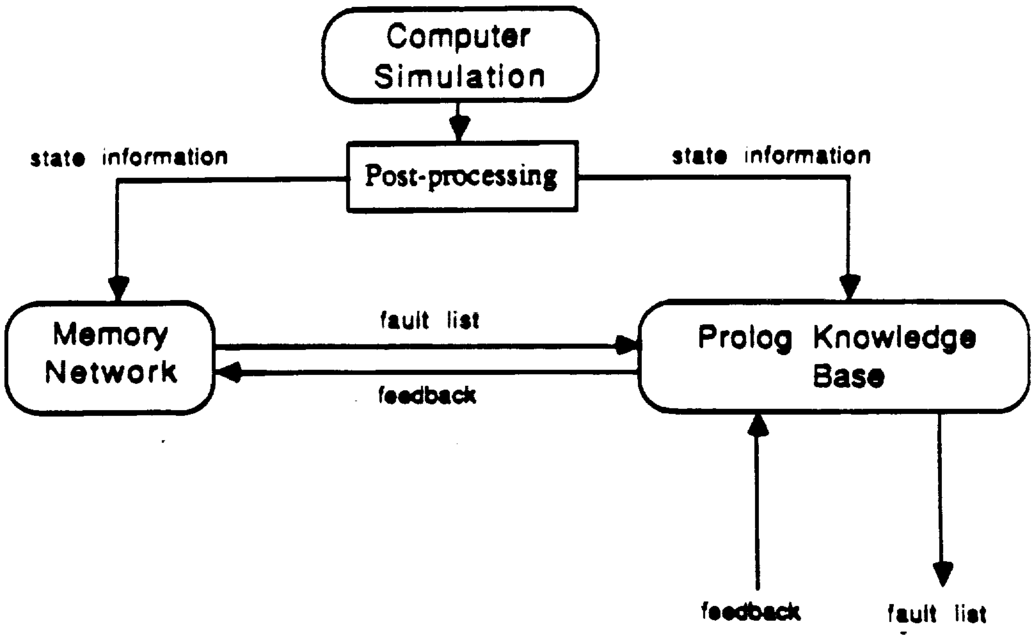


Figure 3.1 Structure of a Fault Detection and Diagnosis System based on DET

The Prolog model incorporates two basic types of information, static and dynamic. Static knowledge refers to the functional description of the components, the connections between the components and the fault protection scheme, e.g. which actions should follow as a consequence of which alarm signals. Dynamic knowledge includes operating conditions of the power system, alarm signals, input from the memory structure itself and feedback from the user. Dynamic information is inserted into the Prolog knowledge during execution by using a file stream. The data can be changed by the rule–based system by inserting new facts over those which previously existed in the database. For instance, the data could be inconsistent due to a sensor failure or poor identification of the situation by the memory structure. In that case, the program will disregard unnecessary information which may lead to inconsistencies in the database. Inconsistency could be caused by one of the three conditions: incorrect/incomplete data, bad rule in the data base or a sub–optimal similarity function. The conditions can be classified and last two inconsistencies will be corrected.

The information is organized into several structures, called predicates. These structures provide a functional way for the Prolog deduction mechanism to determine state estimates and to perform fault analysis given the qualitative information about current system parameters. The Prolog language allows this in a very natural way. The facts about the system under study are declared, clauses to manipulate these facts are built, and the deduction is done automatically by the Prolog inference. For instance, several rules would be used to describe the battery unit:

Maintains bus voltage in normal range
Should be in discharge mode if bus voltage is low
Should switch to charge mode if there is excess power
Should be in charge or neutral mode if shunt regulator is on
Battery current should not change unless a fault or change in conditions occur
Battery current should be increasing (up to max. value) as long as the bus voltage is low

If bus voltage is low and slope of battery current curve is less than <value>
    then {battery voltage is low} or {a problem with error feedback in discharge unit}

If bus voltage is low and battery current is zero
    then {a problem with error feedback in discharge unit} or {incorrect RBI state}

The inference engine can use both forward or backward chaining to arrive at the solution. Forward chaining refers to working toward the goal from the premisses. Backward chaining is picking a possible solution and checking if it is deducible from the available information. This particular program uses both methods. To come up with the first list of possible faults the program produces a forward chain, from state conditions, through component functions and alarms to faults. A second list of possible faults is read in as an input from the memory structure. Prolog works backwards, attempting to show that these faults are deducible from the data base. Next, the program checks if the two lists are consistent, producing the final answer.

The purpose of the memory network is to provide a concise and meaningful representation of past experiences. This allows the decision making system to exhibit a certain amount of adoption to a changing environment. This capability will help to ease the amount of apriori information which must be built into a system.

The memory structure is composed of a variable number of interconnected cells. Each cell is a record of a particular event which has occurred in the system's past. Below are examples of some of the information that may by stored in a typical cell record:

**Cell number 23**

| | |
|---|---|
| Operating condition | *Fault, recovering* |
| Type of fault | *Battery Discharger* |
| Outside conditions | *Unchanged* |
| Power demand | *Unchanged* |
| Solar array | *Changed from* ***Shunt Mode*** *to* ***Full Power Mode*** |

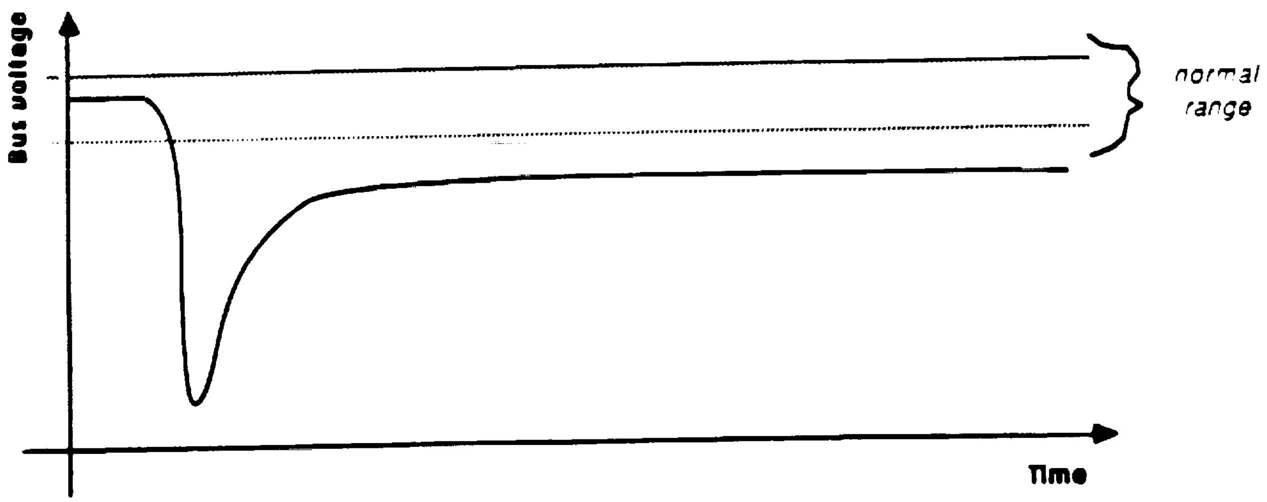
**List of adjacent cells**

| **Number** | **Fault** | **Distance** |
|---|---|---|
| 12 | Battery Discharge | 3.7 |
| 35 | Battery Discharge | 3.95 |
| 37 | Inverter | 4.1 |

**and so on — any** number of cells can be included

**Record of state parameters**

Graphs for Bus Current, Battery Current, Equipment Voltage, Solar Array Current and Shunt current were included. An example of one of the parameters is given below.



The strength of the connections (or alternatively, the distance) between two cells is a reflection of the similarity between events stored in these cells. Ideally, all experiences which are similar will be grouped together. For example, all power generation faults will be placed in a distinct cluster. Within that cluster it should be possible to identify two groups: solar array and battery faults. Each of the groups can be divided still further, e.g. charger, discharger and battery cell failures. These groupings can be made by using any one of a number of topological clustering methods given that a similarity function has been defined. the metric or similarity function simply provides a way of specifying which aspects of the experiential information are important when determining the degree of closeness among experiences. In this implementation, the metric itself is determined with the help of the Prolog database.

Each cell contains information about several important parameters traced over time. Comparisons among the cells may be achieved by matching all of the parameters in two cells and adding the differences. Some parameters are more important than others, this can be represented in the memory structure by varying the weights associated with each term. Even more distinctions can be made: the fact that there was a sudden surge in voltage could be far more important than the actual value of the surge or initial voltages. To implement this, over a dozen comparison functions were defined. Each function is designed to test for a certain pattern: similar initial or final conditions, slopes of graphs, sudden surges or drops, etc. The function takes two cells as an input and returns one if the cells are dissimilar, zero if the cells are similar with respect to that particular function. Some of the functions return a scalar value of the difference instead of one or zero. The distance between cells is the weighted sum of all the functions:

$$\text{Distance}\,(\text{cell1}, \text{cell2}) \overset{\Delta}{=} \sum_{i=1}^{n} W_i f_i\,(\text{cell1}, \text{cell2})$$

Some examples of the functions are given below:

| | |
|---|---|
| $f_1$ | Normal initial conditions (all parameters in range) |
| $f_2$ | Normal final conditions |
| $f_3$ | Unstable final conditions |
| $f_{4-10}$ | Surge/Drop in {bus voltage, load voltage, battery current} |
| $f_{11}$ | Euclidian distance between vectors formed by array currents, battery currents, shunt currents and equipment currents of the two cells |

These functions together with the assigned weights define the metric. Determining the weights properly is the most important part of the identification process; this is described next.

Initial training of the network consists of entering a number of known conditions into the network and trying out different weights until distinct and meaningful groups are formed. Since we know the states, it is easy to determine the *distinct and meaningful* criteria, identical conditions are grouped into distinct clusters. The search for weights is guided by the rule base which restricts the number of weight permutations. First, the clustering method is applied using weight values of one. The resulting groupings are scanned to check which mistakes are committed most often. For example, equipment and inverter faults could be confused and grouped together regularly. The program will then scan the Prolog database and determine characteristics that separate the two faults. As a result, it will increase the weights associated with shunt current, equipment voltage and power stage status. The clustering method would be applied again to check if the weight changes resulted in an improvement. Eventually, the program hits the performance ceiling, changes in weight values do not result in additional improvements.

Once the training is complete, the network can take a state as an input and match it against previously stored experiences. The result is a list of cells the input state most closely resembles. This list is passed on to the Prolog rule–base for further classification. If the matching was not successful, the answer determined by the Prolog rule–base is fed back to the network and the weights are reshuffled in an attempt to come up with a better matching. The process is identical to the one performed during the training period, however only minor adjustments should be needed. The program also has to decide if the new state is sufficiently distinct from previous experiences to warrant being added to the structure.

For example, the memory network reports a battery cell failure. The expert system, after examining state conditions, alarm calls and the location of the fault in the memory structure decides that a power stage in the battery discharge unit failed. Next, the functions of the two components are compared to help modify the similarity function. The weights associated with the slope of battery current, slope of bus voltage current and magnitude of the battery current are changed slightly.

The program keeps modifying the weights until the last fault is identified correctly. At the same time, the overall structure of memory is monitored to make sure that weight permutations do not disturb the existing clusters and cause other faults to be identified incorrectly.

Experimental validation of the fault detection and diagnosis system was accomplished when the data for the memory network was generated using the model simulation. Originally it was planned to use several hundred cells to represent the system. However, the system performed surprisingly well with as few as 50 cells. Since the intent was to study the way the system improved with time, it was desirable to start with a practical implementation, therefore the original training was performed using only 50 cells. Each cell was represented by a Pascal record with variable and static fields. Static fields contain cell numbers and information pertaining to the state stored in the cell. Variable fields contain state classification data and a list of cell numbers which are within a certain distance from this one in the network. An evaluation function penalizes the net whenever the list of similar cells contains cells with different states. Determining the metric is an optimization problem — choosing a set of weights which minimizes this evaluation function.

As expected, there were two ways to improve the solution. The first involved using more information about the states and the second was to add more cells to the structure. If all of the relevant information was known apriori and all of the important states listed, then the memory network would exhibit perfect performance. While this is not possible in practice, the available information was enough to identify most of the faults correctly when tested on simple problems. Out of the twenty novel faults tested, nineteen were identified correctly. Since the states diagnosed incorrectly are added to the memory, the network is taught not to make the same mistakes again. When problems that were caused by multiple faults were simulated, the list of possible faults included one of the problems in all cases and both problems in approximately 3/4 of the cases. The performance seems acceptable considering the small size of the network and could be improved significantly by training. It is worthwhile to note that in cases where the problem was diagnosed incorrectly by the memory network, the solution was in the correct "neighborhood" — the most common mistake was confusing battery cells and battery charger faults, equipment short and power stage short faults.

A problem with rule–base knowledge systems is the tendency of the search to diverge. Here for instance, solar array faults were frequently mistaken for inverter faults. Using the memory structure helps to eliminate this problem. While the alarms for the two faults could be similar, the quantitative state information is quite distinct. Hence, the memory will never confuse the two faults when it passes the solution to the database, only one of the faults will be on the list.

## Example (using incomplete/partial information)

The DET system is operating in shunt mode during the solar cycle. The battery is in neutral mode and 7 solar panels are turned off. An alarm reports sudden drop in bus voltage with no other fault conditions. The solar array is turned to full power and the battery is switched to discharge mode. Bus voltage stops decreasing, however it is still below normal. No other alarms are reported.

The alarms and conditions are presented to the rule based system and the memory network. Two things have to be accounted for: drop in bus voltage and inability of the system to recover.

Rule base

I. Voltage drop could have been caused by:

1. Energy generation problem in solar array, i.e. panel failure
2. Power leak in the Inverter module
3. Power leak in the Equipment Power Stage

In case 1. there should have been an alarm on solar array current. However, a problem with current error feedback could have caused both the low voltage condition and absence of the alarm.
In cases 2. and 3. there could have been a problem with supplying sufficient power to the loads. That problem might not be detected for a relatively long time (more than 20 milliseconds).

II. Stable but low final voltage:

a. Low battery voltage — there is a limit on how much power can be supplied
b. Problem with error feedback in the battery discharge module
c. On/Off switch is in the incorrect position — the battery and/or discharge unit are disconnected from the system
d. RBI malfunction

Logically consistent combinations of faults I & II: 1a,1d, 2a,2b, 3a,3b.

The Memory Network reports a list of past experiences that resemble the fault the most:

| Cell Number | Condition | Similarity metric |
|---|---|---|
| cell 17 | Solar Array | 6.51 |
| cell 46 | Solar Array | 7.8 |
| cell 18 | Battery Discharger | 12.53 |
| cell 24 | Solar Array | 14.65 |
| cell 25 | Battery | 19.08 |

Single fault conditions are easily identifiable by the network. The list of similar faults should be uniform and contains a number of cells at a distance of less than 5 units. Since this is not the case, the rule base will identify the fault as a new condition: it does not fall into any one of the existing clusters. Despite that, the information is far from useless. The Prolog program concludes that a multiple fault occurred: solar array failed, causing a drop in power supply and the switch in the battery discharger is not working properly, preventing the system from recovering. The fault conditions are added to the memory structure, so that the next time this multiple failure occurs, it can be identified immediately. In this case, information was treated simply as additional evidence in support of scenario lc. Given more powerful computational facilities, a much deeper reasoning is possible.

The fault condition has something in common with fault conditions stored in cells 17, 46 and 24 — the solar array faults. The fault is also placed in the neighborhood with cells containing battery faults. The parameters that are responsible for this matchings can be easily identified, both explaining the process and suggesting, perhaps even adding new rules to the expert system. In the same way, the memory can "explain" why other choices suggested by the expert system were incorrect. This example differed from cells storing inverter and equipment faults in solar array current characteristics. Switch from shunt mode to full power mode was responsible for partial recovery of system, suggesting that the fault occurred in the battery discharger or the RBI, not the battery itself.

To summarize, this implementation of the proposed two–level structure has several interesting features:

a. The similarity function is flexible and is constantly updated to account for new experiences. This feature is extremely useful when the true metric is not known: the system can learn new patterns in a way similar to neural networks. the search for the weights, however, is guided by rules in the knowledge base instead of a fixed learning algorithm. This leads to a very useful feature.

b. Unlike a neural network, the system has explanation capabilities. The clustering of data in memory is performed systematically by the rule–base. To explain a match, the system can look at the aspects of the similarity function that were responsible for making the match. Since the rules used in forming the similarity function are known, all of the information needed to explain a match is available.

c. Some rules in the knowledge base could be redundant or incorrect. the system is able to identify these rules automatically: removing them from the knowledge base improves the clustering of data in the memory network. On the other hand, some evaluation criteria could be missing from the similarity function. Correcting this problem in the neural network involves additional training, often extensive, and not always a successful process. The system described here can use this method or attempt to add new rules to the knowledge base, a much quicker and more efficient process.

Once the training is completed, both the rule–based system and memory network can function independently of each other. When a problem is presented, both systems try to find the solution, one using qualitative reasoning, the other quantitative information. A rule–based expert guides the two systems to make sure that both solutions are consistent. The capability also exists for the expert to evaluate performance of the system and suggest when changes in the rule base or the similarity function are needed.

## ATTACHMENT 4
## Operating Replacement Units (ORUs)

### PHOTOVOLTAIC MODULE (PV)

SOLAR ARRAY ASSEMBLY - converts solar insolation to DC electrical power.

101,102 RIGHT or LEFT SOLAR ARRAY BLANKET and BOX

-structure that houses the large area silicon solar cells used in converting solar energy into electrical energy. The solar cells are fixed to an accordion-folded flexible blanket stored in a box during launch.

103 MAST/CANNISTER

-support mechanism that deploys and provides support for the solar blanket. The cannister houses the mast mechanism during launch.

104 SEQUENTIAL SHUNT UNIT

-a switch that regulates the solar array wing output voltage in response to control signals generated by photovoltaic (PV) control elements in the DC switching units (DCSU). It shunts array current that exceeds the load demand.

BETA GIMBAL ASSEMBLY - provides structural support, structural attachment, articulation for sun P&T and the transfer of electrical signals and power to and from the station PV power module.

111 GIMBAL ROLL RING SUBASSEMBLY

-delivers electric power command signals from solar array assembly through th beta gimbal assembly to the IEA. The power and data that flows through this subassembly is destined for the electrical equipment assembly to be converted, stored and distributed as required.

112 GIMBAL BEARING SUBASSEMBLY

-provides rotational capabilities and structural support for the roll ring subassemly, drive motor subassembly and PV array. It provides path between the power generation device and the transition structure while allowing for orientation to the sun. It consists of a pair of bearings, ring gear and housings.

113 GIMBAL DRIVE MOTOR SUBASSEMBLY

-controls the position, velocity and acceleration of the beta gimbal. It provides control commands to the gimbal drive motors. It imparts drive torque to the bearing subassembly via the ring gear enabling the beta gimbal to rotate.

114 STATION GIMBAL TRANSITION STRUCTURE

-positions the beta gimbal as designed.

INTEGRATED EQUIPMENT ASSEMBLY - provides structural attachment for energy storage, and electrical equipment assemblies, requiring on-orbit maintenance and cooling.

121 INTEGRATED EQUIPMENT ASSEMBLY STRUCTURE

-serves as the structure to which utility plates, thermal manifolds and electrical cables are attached. It consists of a rectangular aluminum box frame with attachment holes.

122 CABLE SET AND TRAY

-provides cabling connecting components requiring data or power. It integrates the PV cable set to the truss and the structural framework.

THERMAL CONTROL ASSEMBLY - acquires and transfers excess heat from PV module and rejects it to space.

131 RADIATOR PANEL

-provides the medium to reject excess heat acquired by the utility plates. It consists of nine panels, a condenser section that actually rejects the heat into space, and an evaporator section that interfaces with the main condenser. Its design allows for two panels to fail before affecting ORUs on the utility plates.

132 $GN_2$ CANNISTER

-contains nitrogen gas used to pressurize the bellows which are used to provide a high contact force evenly distributed over the contact area between the condenser and radiator panels.

133 PRESSURIZATION UNIT

-regulates the pressure of the nitrogen gas in the radiator panels to ensure continual and maximum contact between the condenser and radiator panels.

134 INTERCONNECT PLUMBING

-piping that connects the elements of the thermal control assembly and provides a vessel for the transport of excess heat and heat transport fluids. It includes piping taking the heat from the utility plates and connects the heat acquisition, heat transport and heat rejection components.

135 CONDENSER/INTERFACE SUBASSEMBLY

-receives super-heated ammonia vapor from the utility plates and releases it to the evaporator of the radiator panels. It condenses the vapor into sub-cooled liquid which enters the cold end pumping chamber of the rotating fluid management device (RFMD).

136 CONDENSER MOUNT STRUT

-provides a mounting location for the condenser to keep it in contact with the radiator panels.

137 ACCUMULATOR

-regulates the amount of liquid ammonia in the RFMD of the pump unit. It is part of the heat transport subassembly that is activated when transients arise in the various heat transfer processes. The accumulator volume is controlled by the vapor pressure of the ammonia.

138 THERMAL CONTROL PUMP UNIT

-provides force to maintain proper fluid flow in the thermal control assembly.

139 THERMAL SUPPORT STRUCTURE

-provides the support framework for the thermal control assembly.

140 UTILITY PLATES

-provides surface to mount battery packs and electronic equipment ORUs. It provides thermal, electrical and fluid interfaces between energy storage assembly ORUs and the PV electrical equipment ORUs. The plates are mounted to the integrated equipment assembly.

ELECTRICAL EQUIPMENT ASSEMBLY - controls DC power, converts DC to AC power and provides electrical interface to PMAD and Solar Dynamic (SD) modules.

141 DC SWITCHING UNIT (DCSU)

-regulates and controls the source power so input power is always within acceptable limits. It provides the power switching function for interconnection of solar array and battery source power to the main inverter units (MIUs) for conversion to 20 kHz AC power. DCSUs always come in redundant pairs.

142 MAIN INVERTER UNIT (MIU)

-converts DC power obtained by the solar array or energy storage assembly to 20kHz AC power. MIUs are always used in redundant pairs.

143 MAIN BUS SWITCHING UNIT (MBSU) -allows AC power to find the best path to PMAD. It is a combination of remote bus interface (RBIs) switches. It provides switching between redundant AC busses. MBSUs are always used in redundant pairs.

144 PV SOURCE CONTROLLER (PVSC or PVC)

-provides communication between the PMC and the PV module functional controllers. It controls the generation, storage and regulation of the PV source power. It controls and monitors the PV wings. It can operate PV module in the event of certain PMAD failures which increases system reliability. PVSCs are always used in redundant pairs.

145 POWER DISTRIBUTION and CONTROL UNIT (PDCU)

-provides lower voltage AC power to PV module electrical equipment. It resembles a house's wall outlet by function (i.e.,it does not look like a wall outlet, but operates similar to an outlet). PDCUs are always found in redundant pairs.

ENERGY STORAGE ASSEMBLY - stores, conditions, controls and distributes electrical power produced by solar arrays.

151 BATTERY CHARGE/DISCHARGE UNIT (BCDU)

-controls charge and discharge rates of the battery packs. In conjunction with commands from the PVSC, measures and compares parameters and adjusts the charge rates to match the target charge rate values given by the PVSC. It regulates current flow to the battery and boosts battery voltage to the level of the DC bus voltage. It isolates the battery pack from both the charge/discharge unit and the control power bus to protect the batteries against downstream faults. It consists of a battery controller, charge regulator, discharge regulator and DC fault interrupters and DC cables. One BCDU is found with every three battery packs. Peak power output is 6.5 kw.

152 $NiH_2$ BATTERY PACK

-meets all energy requirements for the station, including safety, performance commonality and modularity. It is a nickel hydrogen rechargeable battery consisting of thirty 81 A-hr cells in series.

401 ALPHA GIMBAL

-provides solar array and SD pointing on the alpha axis. It allows for passage across its assembly for data and electrical energy. It is not a part of the PV module.

## SOLAR DYNAMIC MODULE (SD)

CONCENTRATOR ASSEMBLY - acquires and concentrates radiation into receiver.

201 REFLECTIVE SURFACE SUBASSEMBLY

-offset parabolic design of 19 hexagonal truss panels of lightweight graphite epoxy construction with multiple triangular reflective facets mounted in the hexes. It reflects incoming solar radiation through the dynamic receiver aperture.

202 CONCENTRATOR STRUCTURE STRUT SET

-provides framework to provide a fixed reference distance between the reflector vertex and the receiver aperture and firmly supports the reflector.

203 CONCENTRATOR CONTROLS CABLE SET

-provides connections within the concentrator for commands and data.

204 SUN SENSOR

205 ISOLATION METER

206 TWO-AXIS FINE-POINTING MECHANISM

-provides vernier-poinitng capability and acts as the structural transition between the interface structure and reflector support strut set.

HEAT REJECTION ASSEMBLY - acquires and transfers excess heat waste from SD module and rejects it to space so as to maintain appropriate SD module component and electrical equipment within required temperature limits during active modules operation.

211 RADIATOR PANEL DEPLOYMENT SUBASSEMBLY

-provides the heat rejection function for SD module by radiating heat into space. It consists of eight panels, a deployment mechanism and the support structure.

212 HOT INTERCONNECT LINES

-fluid lines with disconnects that provide the flow path between the power control unit (PCU) assembly and the radiator array subassembly. Two pairs of lines exist.

213 COLD INTERCONNECT LINES

-two pairs of fluid lines with disconnects that provide the flow path between the SD utility plate and the PCU assembly.

214 PUMP INTERCONNECT LINES

-fluid lines with disconnects that provide the flow path between the radiator array subassembly and the fluid management units. Two pairs of lines exist.

215 UTILITY PLATE

-contains electronics cooling cold plate, thermal interface with electronics ORUs and fluid interfaces with other coolant management subassembly ORUs and fluid interfaces with other coolant management subassembly ORUs. Under all operating conditions the utility plate coolant outlet temperature remains less than 20 C.

216 FLUID MANAGEMENT UNIT

-contains all of the active components for each loop of the closed Brayton cycle (CBC) heat rejection assembly.

RECEIVER/POWER CONTROL UNIT (PCU) - make up the power generation subsystem. It exchanges heat from the heat source, stores thermal energy and converts thermal energy in the cycle working fluid to electrical energy.

221 RECEIVER

-admits concentrated solar flux through its aperture. It consists of 82 tubes carrying the working fluid that absorbs the solar energy. It serves as the heat source exchanger and thermal energy storage device for the SD module.

222 ENGINE CONTROLLER

-controls operation of the PCU.

223 PARASITIC LOAD RADIATOR

-performs the function of an electrical sink for excess power. It provides effective spee d control for the turbo-alternator rotor while managing the excess power in a way that allows fast response to changes in user demand.

224 CONTROL VALVE ACTUATOR

-provides a closed center, three-way diverter valve designit isolates the accumulator from the compressor allowing pressure to be stored until needed for peak power operation.

225 PCU POWER CABLE SET

-provides power connections from within and from the PCU.

226 PCU SIGNAL/DATA CABLE SET

-provides data and signal connections within the PCU.

ELECTRICAL EQUIPMENT ASSEMBLY - containd most of the major electric components required to operate and control the SD module and electric power it generates.

231 FREQUENCY CHANGER

-solid state power electronic component which converts the mid-frequency, 3-phase AC power from the SD alternator at its output to 20 kHz single phase as power at its output for transmission to the SD module/PV module interface.

232 LINEAR ACTUATOR OUTER

-in conjunction with the inner linear actuator translates and rotates the reflective surface subassembly independently of the receiver/PCU assemblies resulting in a low gimbalad mass and modest coarse and fine-pointing parasitic power requirements.

233 LINEAR ACTUATOR INNER

-in conjunction with the outer linear actuator translates and rotates the reflective surface subassembly independently of the receiver/PCU assemblies resulting in a low gimbalad mass and modest coarse and fine-pointing parasitic power requirements.

234 SD CONTROLLER

-controls the total operation of the SD module, excluding the PCU operations. It includes controls for concentrator pointing, beta gimball pointing, radiator deployment/retraction, fluid pumps and overall SD supervisory control.

BETA GIMBAL ASSEMBLY - accounts for the seasonal motion of the sun and procession of the orbit plane, the beta gimbal rotates the the SD module a nominal +52 degrees about the beta axis.

241 GIMBAL POWER/DATA TRANSFER SUBASSEMBLY

-delivers electric power from the electrical equipment assemblies (output from the frequency changer) and data signals from the SD module controller, across the rotating interface to be delivered to and received from the PV module.

242 GIMBAL BEARING SUBASSEMBLY

-provides the structural path between the power generation device and the transition structure while allowing for rotation for orientation to the sun. It consists of a pair of bearings, wing gear and housings.

243 GIMBAL DRIVE MOTOR

-controls the position, velocity and acceleration of the beta gimbal. It provides control commands to the beta gimbal drive motors. It provides the position of the beta gimbal, drive motor status and health monitoring status to the SD controller. It provides torque to point and slew the attached power generation device.

244 STATION GIMBAL TRANSITION STRUCTURE

-positions the beta gimbal within the 5-meter truss so that it is centered within an outboard face of the 5-meter cube and positioned into the cube interior to allow for proper clearance of the interface structure assembly with the transverse boom.

INTERFACE STRUCTURE/INTEGRATION HARDWARE -

251 INTERFACE STRUCTURE ASSEMBLY

-high rigidity interface structure to which all other SD module assemblies are ultimately attached. It supports the receiver assembly, the PCU assembly and the Heat Rejection assembly as well as the SD equipment box subassembly. It provides part of the interface between the Receiver/PCU/Heat rejection assemblies launch package and the shuttle.

252 SD CABLE SET

-conducts 20kHz AC, single phase power from the output of the frequency changer in the SD module to intermediate control equipment such as an RBI and from there to the SD module/PV module interface.

301 MAIN BUS SWITCHING UNIT (MBSU)

-serves as the primary tie tie point for utility power feeder protection and supplies utility power to the distribution system. It turns internal RBIs on and off under normal operation and interrupts faults and reconfigures the system as necessary. MBSUs are always found in redundant pairs.

302 POWER MANAGEMENT CONTROLLER (PMC)

-primary controller of PMAD system. It provides overall power system coordination and EPS data flow. It is the interface between the power system and the data management system network and interfaces with subordinate controllers via the PMAD control bus. It conducts built-in test functions. It controls functions for power distribution system coordination and monitoring of power generation system. It has data management/communications functions with external systems and ground to include providing DMS with specific power system data that has been received from the lower level control processors and actual sensor data. It coordinates and commands the lower level control processors in order to provide users power.

303 AC/DC CONTROL UNIT (ADCU)

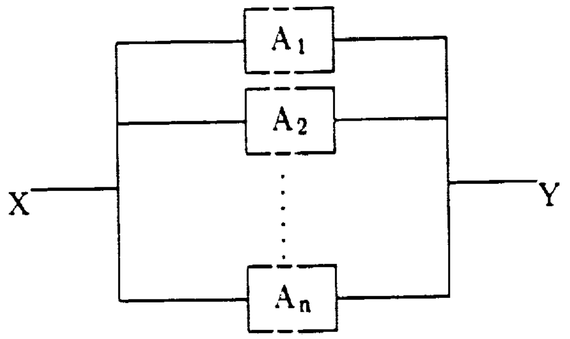
-converts AC power to DC power.

304 POWER DISTRIBUTION CONTROL UNIT (PDCU)

-serves as the final distribution point to all user loads.

## ATTACHMENT 5
**Special Connection Blocks**

There are some special block connections useful in developing the RBDs ot engineering systems, for example the space station freedom power systems. First, there are two types of modified parallel connections as illustrated below:



| | |
|---|---|
| 1) Normal parallel connection: (1–out–of–n) | Power W is delivered to Y if at least one of n units $A_1,\ldots,A_n$ works |
| 2) Partitioned parallel connection: | Power $\frac{kW}{n}$ is delivered to Y if k units of the n units $A_1,\ldots,A_n$ work |
| 3) k–out–of–n parallel connection: | Power W is delivered to Y if at least k units of the n units $A_1, \ldots , A_n$ work. No power is delivered to Y otherwise. |

Secondly, the failure characteristics of some primitive blocks depend not only on their own failure characteristics, but are conditionally dependent on other primitive blocks. Special computational rules are provided to account for this dependency.

The third type of special connection is the partitioned series connection. This structure arises subsequent to the simplification of a partitioned parallel connection. The output delivered by a component in this connection depends not only on its own failure characteristics but also in the preceding partitioned parallel connection.

## ATTACHMENT 6
**Functions and Features of FAULTS**

There are five main functions in FAULTS : Build; Reliability Analysis; Power Flow Analysis; Training; and Fault Detection and Diagnosis.

**Build** is the function of SYSTEM BUILDER described in the report. It allows the user to construct a complex system by providing (i) a list of primitive blocks along with their reliability, operating states, and input-output descriptions, and (ii) a list of customized block connections (if any) together with the corresponding computation rules. The user can then create blocks at different levels of aggregation by specifying primitive and/or previously created blocks to be used and the type of connection required. Once created, each block can be stored in a library so that it can be used to create other blocks. Auxiliary functions such as MODIFY, ADD, and DELETE add power and flexibility to the system building capability by allowing the contents and characteristics of the created blocks to be modified, enhanced or deleted when needed.

**Power Flow Analysis** is one of the two main functions of RPFSim. It allows power levels at any ORUs, subassemblies, assemblies, or the whole system to be computed for any input conditions, operating states of ORUs, and input-output relationships. Essentially any form of input-output model for each ORU, subassembly, and assembly can be entered including simple efficiency models (e.g. output = efficiency * input), function/logical models (e.g. output = $f_s$(eff, input) in sunlight cycle and = $f_e$(eff, input) in eclipse cycle), and arbitrary dynamic models.

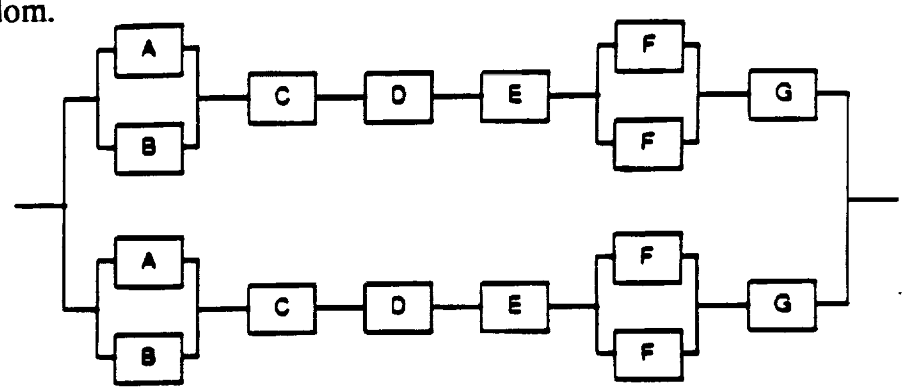
**Reliability Analysis** is the other main function of RPFSim, which is used to compute reliability measures for any part of the system in terms of power delivered. In particular, when combined with Power Flow Analysis to form RPFSim, the following can generated: (i) the probability that a given level of power (measured in terms percentage of maximum deliverable power level) will be delivered by the system or to a particular unit in the system at a given time; and (ii) the power profile (delivered power v.s. time) during a time period of interest for any given scenario concerning component failures and repair. Both these results can be displayed in tabulated or graphical forms.

RPFSim and SYSTEM BUILDER together allow extensive "what-if" experiments to be performed to examine the effects of system configurations, reliability parameter specifications of components, and input/output characteristics of the components on the overall reliability performance of the system. The ability to quickly and easily accommodate changes and reconfigure the system is deemed as a key criterion in designing FAULTS, particularly if it is to be used as a system design tool.

**Training** is a function that allows the system to learn about common and novel faults from data generated from RPFSim and expert knowledge (rules), if available. Single mode and multimode faults can be handled. At the end of each training session, recognizable patterns of faults are identified. In the current implementation, a self organizing (memory) network is used as a training tool, and data generated from RPFSim (model-based) is the main training set. If in the future, rules and other forms of qualitative information are available, the fault patterns identifier/trainer should be upgraded. For example, rules or expert knowledge might be used to assist the weight adjustment scheme in the self organizing network algorithms to identify clusters more efficiently and effectively.

**Fault Detection and Diagnosis** uses results from **Training** in combination with expert knowledge to detect and diagnose faults corresponding to a given set of observed sensor measurements. The data fusion scheme used is a simple evidence weighing procedure: First, the degree of membership (of the observed set of sensor readings) to each cluster is computed. Only clusters with high degrees of membership (i.e. most similar to the observed data) are maintained, and fault patterns typifying those clusters are considered further. This helps narrow down the scope of search. Second, a fault index associated with each retained fault pattern is computed by multiplying the likelihood (reliability) of that pattern to the distance from the observed data to that pattern. The smaller the index, the more likely that faults in the system giving rise to the observed data are characterized by that particular fault pattern. Thus, possible faults can be ranked according to their fault indices. Other data fusion schemes such as Dempster-Shafer theory of evidence, Bayesian rules, and fuzzy logic could be investigated in future work.

To illustrate, we will show how to build and analyze the following subsystem representing one wing of the Solar Array Assembly in the power system of Space Station Freedom.



A through G represent the primitive units (ORUs). ORUs A and B are connected through a partitioned parallel connection to form a subassembly, while 2 ORUs of type F are connected in parallel to form another subassembly. These two assemblies are connected in series with C, D, E, and G to form a Solar Array Assembly.

Two Solar Array Assemblies are then connected in parallel to form a Solar Array Wing as shown in the diagram.

On activating the system and select POWER SYSTEM option from the system menu, the following menu will appear:

MAIN MENU

1. Component Menu
2. Power Flow Simulation
3. System Diagnosis
4. Maintenance Policy
5. File Menu
6. Display Defaults
7. Clean Graphic Panes
8. Clean Simulation Pane
9. Exit

BUILD

In the BUILD mode, ORUs A through G have to be individually entered as primitive blocks, each block with its own reliability description (e.g. MTBF and operating states) and a power input-output description. For example, to create "A", choose "Component Menu" and then choose "Add". The following pop-up window will appear:

COMPONENT WINDOW

| | | |
|---|---|---|
| Name: | | A |
| MTBF:(hr.) | | 87000 |
| Normal Output: | | (0.9995 *) |
| Faulted Output: | | (0.17 *) |
| Display | | |
| Label: | | A |
| Length: (pixels) | | 30 |
| Width; (pixels) | | 18 |
| Color: | | Green |
| Operating Condition: | | Normal |
| Power Input: | | 77.3081 |
| Power Output: | | 77.2695 |
| System Type: | | nil |
| List of Subcomponents: | | nil |
| TOP LEVEL | for Graphics? | YES NO |
| | for Reliability? | YES NO |
| | for Power Flow? | YES NO |

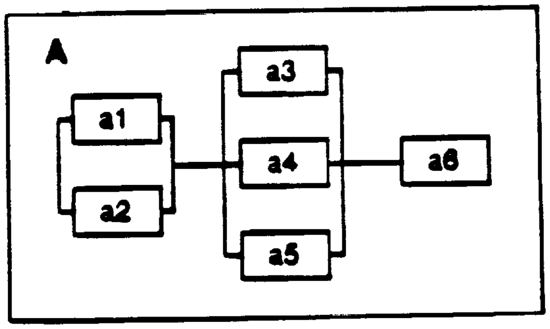
"Name" is the name of the block to be listed in the library for general reference (e.g. A or Battery A); "MTBF" is the mean-time-between-failure of the block in hours (if the block represents an ORU, enter the designed MTBF, otherwise either leave it blank or fill the MTBF for the whole block if known.); "Normal Output" is the output level as a function of input (input-output relationship) if the block is operating in the normal mode (expressed in KW for power generating blocks such as solar array and battery, and possibly as a fraction of the input power for other blocks such as in the above example where Normal Output = 0.9995*Input); "Faulted Output" is the output level if the block is operating in the faulted mode; "Display" items define the desired label, dimensions and color of the screen display of the block; "Operating Conditions" is the simulated operating scenario of the block (e.g. normal until the 8th hour and fully repaired or replaced at the 9th hour); "Power Input" is the power level at the input side of the block (leave blank, if the user does not know); "Power Output" is the power level at the output side of the block (leave blank, if the user does not know); "System Type" represents the connection type if the block has subcomponents (type "nil" if no subcomponent); and "List of Subcomponents" is a list of subcomponents of the block if any (type "nil" if no subcomponent). The bottom part of the window will be described later.

The process is repeated for each primitive unit (A to G) and the results stored in the library of components for further use. Now the subassemblies A-B and F-F can be created. For example, to create A-B, again the "Add" command is selected from the menu in "Component Menu". The same pop-up window appears. The name "Battery Pack" and label "A-B" might be entered for the "Name" and "Label", respectively. No entries are needed for the "MTBF" through "Power Output" as these will be automatically determined from the subcomponents and the interconnection structure. To enter subcommponents, first specify the "Partitioned Parallel" connection type in "System Type", then activate "List of Subcomponents" (by a mouse) to show a list of names and blocks already created. Subcomponents A and B can be selected from the list by highlighting the desired components using a mouse. Subassembly A-B is then formed and stored in the library of components. The process is repeated for subassembly F-F.

Next, the Solar Array Assembly is created by activating the "Add" command from the "Component Menu", entering the term "Solar Array Assembly" as the assembly name and/or label, and activating the "List of Subcomponents" command from the same window. Blocks A-B, C, D, E, F-F and G are then selected from the displayed list of created components. Finally the series connection is specified in the "System Type" slot. A block for Solar Array Assembly has now been created and stored. The complete Solar Array Wing can now be created by again using the "Add" command from the "Component Menu", using the "List of Subcomponents" command to select Solar Array Assembly twice as subcomponents, and specifying the parallel connection in "System

Type" as the connection between the two subcomponents. This completes the building of the Solar Array Wing as shown in the diagram.

Other commands in the "Component Menu" are "Modify", "Copy", "Remove", "Display", "Power Flow" and "Reliability". By using the "Modify" command, a created block can be modified by adding or deleting subcomponents and/or modifying the reliability description and input-output description. For example, suppose Block A is to be modified to include subcomponents as shown in the diagram below.



First, build the primitive blocks a1 through a6 and the combined blocks a1-a2 and a3-a4-a5 using the "Add" command as before. Then choose the "Modify" command, and specify Block A as the block to be modified. The component window similar to the one shown earlier now appears. Choose the "Add Component" command from the window and select blocks a1-a2, a3-a4-a5, and a6 from the component library, and finally select the series connection from the list of available connection types. Block A has now been modified and all other blocks containing A as a subcomponent will also be automatically modified accordingly.

The "Copy" command is used to copy a block for use in another part of the system. The "Remove" command is used to remove any created block. And the "Display" command is used to display a created block.

By activating appropriate commands at the bottom of the component window, a block that contains subcomponents can be displayed or analyzed at the TOP level as a single block or at a lower level as interconnections of subcomponents.

## ANALYSIS

The "Power Flow" and "Reliability" commands in the "Component Menu" are for performing "single event" simulation on reliability and power flow for a selected block. (Multiple events and more complex scenarios for simulation are handled through commands in the "Power Flow Simulation" option activated from the MAIN MENU).

For reliability analysis, upon choosing the "Reliability" command, the following window appears:

RELIABILITY WINDOW

| | |
|---|---|
| System Name: | Solar Array Wing |
| Time: | 8700 |
| Display: | PRINT PLOT BAR |
| Interval Between Points | 10 |
| Refresh Previous Graphics Screen | YES NO |
| Line Color to Use: | red |

The user enters the name of the system (a block from the library of components) to be analyzed, and the time in hours at which the reliability of the system is to be computed. The user will also choose whether to display the results in a tabulated form (PRINT), an exceedence probability plot (PLOT), or a bar chart (BAR). If the PLOT option is selected, the user can also choose whether to remove old plots before a new line is plotted (YES). If user chooses to plot a new line without erasing the old lines (NO), they can also choose a color for the new line.

The idealized results would consist of the probability value for each discrete power level deliverable by the system. For a system consisting of many components and subcomponents, the number of possible discrete power levels would be large. To simplify the computation, display, and data storage requirements, an approximation is made to limit the maximum number of discrete power levels analyzed for each system. The approximation is based on aggregating several successive points (power levels) into a single point. The number of points to be aggregated is specified by the user by entering the desired "Interval Between Points". The user can also specify the absolute maximum number of power levels to be considered by specifying the "Maximum List Size".

For the power analysis, the following window appears once the "Power Flow" command is activated from the "Component Menu".

POWER FLOW WINDOW

| | |
|---|---|
| System Name: | Solar Array Wing |
| Power Input: | 77.3081 |
| Illumination: | (Sunlight or Eclipse) |
| Load: | (Peak or Reduced) |

Again the user enters the name of the system to be analyzed and the input power level to the system. The user also selects the illumination cycle (sunlight or eclipse) to analyze and the load level (peak or reduced) to the system. The average power delivered to various components and subcomponents of the system as well as to the load are then computed and displayed.

Finally, dynamic power flow under various operating states of components and subcomponents can be simulated using the "Power Flow Simulation" command from the MAIN MENU. Upon selecting a particular system to be analyzed and activating the command, the following window appears.

POWER FLOW SIMULATION WINDOW

---

Choose Events:
Clear Past Events:
Run Simulation:
Display Results:

---

The "Choose Events" command allows contingencies (failure/recovery scenarios) to be developed for simulation. A typical scenario may read: Component A fails at the $t_1$ hour and is fully repaired at the $t_2$ hour; Component B fails at the $t_3$ hour etc. Events from the created scenarios can be deleted by using the "Clear Past Events" command. Once an acceptable contingency scenario is created, the simulation can be executed using the "Run Simulation" command, and the results displayed using the "Display Results" command. Results which show the output power v.s. time can be displayed in either tabular or graphical form.

In addition to defining simulation scenarios by using "Choose Events" to specify events to be simulated, maintenance and replacement policies can also be incorporated as part of those scenarios. This is done through the "Maintenance Policy" command in the MAIN MENU. When the command is activated, the following menu appears.

MAINTENANCE POLICY WINDOW

Choose
Remove
Display

The "Choose" command allows the user to select a component or subsystem and describe (in LISP) the maintenance and replacement scenario to be associated with that component/system. A typical maintenance/replacement scenario may read: If component A is older than 2 years old, do maintenance and upgrade work so that it looks like new etc. The "Remove" command is used to delete a selected maintenance and replacement scenario already entered. The "Display" command gives the list of all components having maintenance and replacement scenarios defined.

## SYSTEM DIAGNOSTIC

This is where the training and fault detection and diagnosis functions are performed. However, although all features have been performed and included in the demonstration, the current version is not yet suitable for implementation. Not every piece is fully developed and not all pieces are tied together. To use this feature, the user will specify the system or subsystem to be investigated, a list of sensors, the current sensor readings to be analyzed, and any fault patterns known to the user. Additional information required are the sensor readings under normal operating conditions, and additional list of known fault patterns. The first type of information could be generated by running the system using "Power Flow Simulation" with all components operating at the normal states (however, in this version, the results has to be transferred to this function manually.) The knowledge base of verified fault patterns can be generated (off-line) from simulated data and the results kept in a separate file. The result will then be manually transferred to this function to perform the final diagnosis task.

We end this attachment with a brief description of other options in the MAIN MENU.

FILE MENU: When activated, the following menu appears.

FILE MENU
- Component Files
- Simulation Data Files
- System Defaults
- Graphics

The "Component Files" command is used to READ (LOAD) or WRITE (SAVE) files describing components and subsystems created by the BUILD function described above.

The "Simulation Data Files" command is used to READ (LOAD) or WRITE (SAVE) simulated data files generated using the "Reliability", "Power Flow", and "Power Flow Simulation" commands.

The "System Defaults" command is used to define/modify all default attributes for FAULTS other than graphics display.

The "Graphics" command is used to define/modify (and save) defaults for graphic display. The following window shows typical default values used:

| | |
|---|---|
| X distance between components | 4 (pixels) |
| Y distance between components | 12 (pixels) |
| Starting X location for Display | 25 (pixels) |
| Starting Y location for Display | 280 (pixels) |
| Width of connecting lines | 2 (pixels) |
| Graphic Screen Background | Black |
| Graphic Screen Foreground | Yellow |
| Lisp-Listener-Background | 75% Gray |
| Lisp-Listener-Foreground | White |
| Menu-Pane-Background | White |
| Menu-Pane-Foreground | Blue |
| Parallel Connection Colors: | |
| Redundant Parallel | Green |
| Partitioned Parallel | Red |
| k-of-n parallel | Blue |
| Series Connection Color | Green |

DISPLAY DEFAULTS:
This command performs the same function as the "Graphics" command in the FILE MENU option.

CLEAN GRAPHIC PANE:
This is used to clear the screen

CLEAR:
This is used to clear LISP window.

EXIT:
This is used to exit the program.