Integration/Evaluation of a HCI Prototyping Environment

Final Report

Delivery Order No. 25
Basic NASA Contract No. NAS8-39131

Department of Computer Science and Engineering
Auburn University, AL 36849-5347

Contact: Loretta A. Moore, Ph.D.
Principal Investigator
(205) 844-6330
moore@eng.auburn.edu

October 17, 1994

Integration/Evaluation of a HCI Prototyping Environment

Final Report

Delivery Order No. 25
Basic NASA Contract No. NAS8-39131

Loretta A. Moore, Ph.D.
Principal Investigator

October 17, 1994

ABSTRACT

Components of a Human Computer Interface (HCI) prototyping environment have been integrated and evaluated. This environment will be valuable in developing and refining HCI standards and evaluating program/project interface development, especially the International Space Station Alpha's on-board displays for payload operations. This environment, which allows for rapid prototyping and evaluation of graphical interfaces, includes four components: (1) a HCI format development tool, (2) a test and evaluation simulator development tool, (3) a dynamic, interactive interface between the HCI prototype and simulator, and (4) an embedded evaluation capability to evaluate the adequacy of a HCI based on a user's performance. The objective of the research was to determine whether or not the functional components could be integrated and could provide the needed functionality for a rapid prototyping environment. In order to evaluate the rapid prototyping environment two prototypes were developed. The system chosen for initial evaluation was an automobile. Following the automobile prototype development, the Hopkins Ultraviolet Telescope (HUT), a Spacelab/Space Station payload, prototype was developed. This report will discuss the architecture of the environment, the prototypes developed within it, and results of an evaluation of the environment based on usability, functionality, and performance.

ACKNOWLEDGMENTS

We appreciate the assistance provided by NASA personnel, especially Mr. Joseph P. Hale, whose guidance has been of great value. The following is an alphabetical listing of the team members who have participated on the project.

Principal Investigator:

Loretta A. Moore, Ph.D.

Graduate Research Assistants:

Terri Bester
Shannon Price
Thomas Rix

Undergraduate Research Assistants:

Roderick Chaney
Trina Knight

The following trademarks are referenced in the text of this report:

Sammi is a trademark of Scientific Software-Intercomp.
PERCNET is a registered trademark of Perceptronics, Inc.

TABLE OF CONTENTS

LIST OF FIGURES

1.0 INTRODUCTION

The Crew Systems Engineering Branch of the Mission Operations Laboratory of NASA Marshall Space Flight Center is interested in a dynamic Human Computer Interface Prototyping Environment for the International Space Station Alpha's on-board payload graphical displays. On the Space Station, new payloads will be added to the on-board complement of payloads in ninety day increments. Although a payload starts its development and integration processes from two to four years before launch, a set of new payloads' displays are due every ninety days. Thus, this drives the need for an efficient and effective prototyping process. The functional components of a dynamic prototyping environment in which the process of rapid prototyping can be carried out have been integrated and evaluated.

Most Graphical User Interface toolkits allow designers to develop graphical displays with little or no programming, however in order to provide dynamic simulation of an interface more effort is required. Most tools provide an Application Programmer's Interface (API) which allows the designer to write callback routines to interface with databases, library calls, processes, and equipment. These callbacks can also be used to interface with a simulator for purposes of evaluation. However, utilizing these features assumes programming language knowledge and some knowledge of networking. Interface designers may not have this level of expertise and therefore need to be provided with a friendlier method of producing simulations to drive the interface.

A rapid prototyping environment has been developed which allows for rapid prototyping and evaluation of graphical displays [5]. The components of this environment include: a graphical user interface development toolkit, a simulator tool, a dynamic interface between the interface and the simulator, and an embedded evaluation tool. The purpose of this environment is to support the process of rapid prototyping, so it is important that the tools included within the environment provide the needed functionality, but also be easy to use.

In order to evaluate the usability, functionality, and performance of the environment two prototypes were developed. The specific tasks which were performed for this statement of work include:

(1) Integration and testing of the automobile interface, simulator, and evaluation component.

(2) Derivation of the payload prototype requirements from the Hopkins Ultraviolet Telescope (HUT) simulator requirements document.

(3) Development of the payload simulator using PERCNET, a graphical modeling and knowledge-based simulation development environment.

(4) Development of a graphical interface for operation of the payload simulator using SAMMI, a graphical interface development toolkit.

(5) Integration and testing of the payload simulator and interface.

The products of this effort include an understanding of the requirements for a rapid prototyping environment, an integrated and working prototype of an automobile prototype with embedded evaluation, and an integrated and tested version of a payload prototype. The following sections of this paper describe the prototyping environment, the development of the prototypes, and results of the evaluation.

2.0 ARCHITECTURE OF THE PROTOTYPING ENVIRONMENT

The Human-Computer Interface Prototyping Environment with Embedded Evaluation capability is designed to allow a developer to create a rapid prototype of a system and to specify correct procedures for operating the system [6]. The first component of the architecture is the Graphical User Interface (GUI) development tool. This tool allows the designer to graphically create the interface of the system and specify a data source for each object within the display. The simulator tool provides the capability to create a low-fidelity simulation of the system to drive the interface. The embedded evaluation tool allows the designer to specify which actions need to be taken to complete a task, what actions should be taken in response to certain events (e.g., malfunctions), and the time frames in which these actions should be taken. Each of these components is a separate process which communicates with its peers through the network server. Figure 1 shows the architecture of the HCI Prototyping Environment.

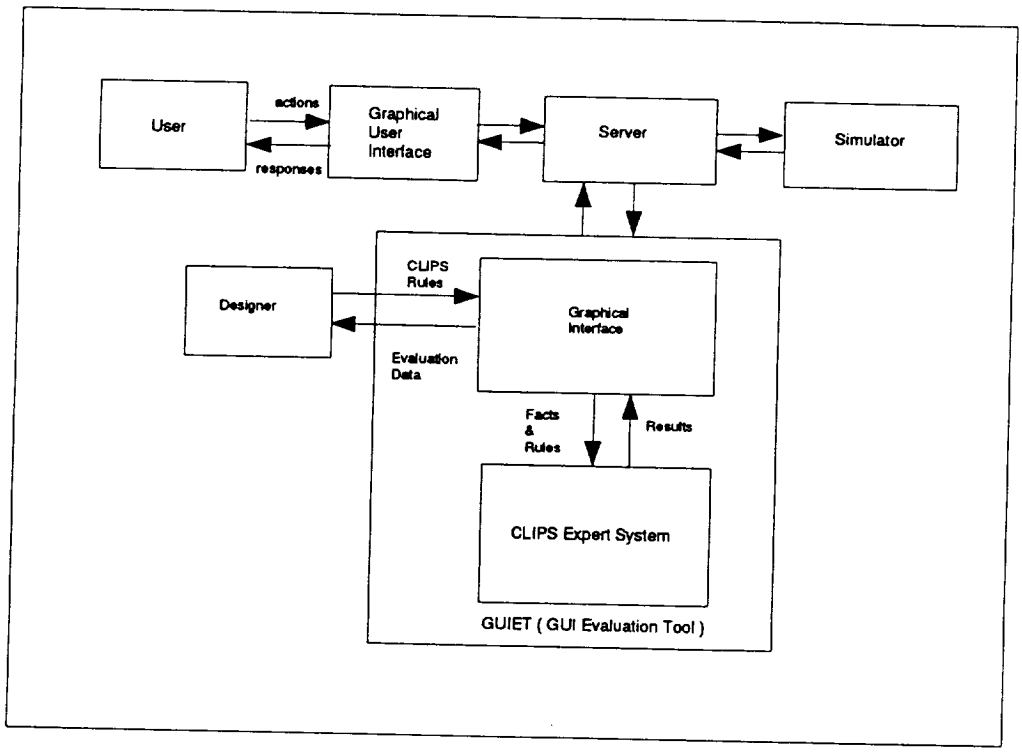


Figure 1 - Architecture of the HCI Prototyping Environment

During execution of the system, the interface objects send and receive data and commands to and from the simulator by way of the data server and the simulator provides realistic feedback to the interface based on user inputs. The server sends the embedded evaluation tool the actions which the user has taken, all events and activities which have occurred, and the times associated with these items. The embedded evaluation tool analyzes the actions which have been performed by the user, that is, the user's model of the system, against the predefined conceptual model of the designer. The system identifies which tasks were completed correctly, or not, and provides information to the designer as to the points in the interaction in which the user's model of the system did not correspond to the designer's conceptual model of the system.

2.1 Graphical User Interface Development Tool

The Graphical User Interface (GUI) tool for the prototyping environment will allow the designer to create the display through direct manipulation. This includes the creation of static and dynamic ob-

jects, windows, menus, and boxes. The tool also allows objects created to be linked to a data source. During execution, the interface objects send and receive data and commands to the simulator by way of the data server. The user interface objects and their associated data access description are defined independent of the actual source of data. This first allows the development of the interface and the simulator to occur concurrently. Second, an interface developed with the GUI tool can later be connected to a high fidelity simulator and then to the actual flight software. The tool used in this study was SAMMI [9].

2.2 Simulator Development Tool

An evaluation of a prototyped user interface is best supported by a simulation of the system. A simulation allows for dynamic evaluation of the interface rather than just a static evaluation of the screen's appearance. This allows potential users to evaluate both the look (in terms of the screen layout, color, objects, etc.) and feel (in terms of operations and actions which need to be performed) of a system's interface. Because of the need to provide dynamic evaluation of an interface, there must be support for producing active simulations. The high-fidelity training simulators are normally delivered too late to be effectively used in prototyping the displays. Therefore, it is important to build a low fidelity simulator, so that the iterative cycle of refining the human computer interface based upon a user's interactions can proceed early in software development. The simulator development tool provides the capability to develop a low fidelity simulation of a system or process. In addition to producing active simulations, the simulator tool helps the designer identify and define basic system requirements. The tool used for simulation in this study was PERCNET [8].

2.3 Graphical User Interface Evaluation Tool

An important aspect of the prototyping process is the ability to evaluate the adequacy of the developed graphical user interfaces. The embedded evaluation tool communicates with the server to receive information on the interaction between the user and the system. The types of data collected include user actions, simulator events and activities, and the times associated with these items. The collected data is analyzed to determine task correctness, task completion times, error counts, and user response times. The data is then analyzed to provide feedback as to which features of the interface the user had problems with and therefore need to be redesigned. The graphical user interface evaluation tool used was GUIET [6].

2.4 Dynamic, Interactive Interface

This interface will handle communication between the GUI prototyping tool and the simulation tool during execution. The interface is a server which has been developed using the GUI's Application Programmer's Interface. Messages and commands can be sent and received both ways between the GUI and the simulator. The server also services requests from the embedded evaluation process, providing information as to which actions the user has taken and which events and activities have fired.

3.0 AUTOMOBILE PROTOTYPE

In order to evaluate the architecture, an automobile system was prototyped in the environment. An automobile was chosen because it has sufficient complexity and subsystems' interdependencies to provide a moderate level of operational workload. Further, potential subjects in the empirical studies would have a working understanding of an automobile's functionality, thus minimizing pre-experiment training requirements.

An automobile can be considered a system with many interacting components that perform a task. The driver (or user) monitors and controls the automobile's performance using pedals, levers, gauges, and a steering wheel. The dashboard and controls are the user interface and the engine is the main part of

the system. Mapping the automobile system to the simulation architecture calls for a model of the dashboard and driver controls and a separate model of the engine. The main component of the automobile is the engine which responds to inputs from the driver (e.g. the driver shifts gears or presses the accelerator pedal) and factors in the effects of the environment (e.g. climbing a hill causes a decrease in the speed of the car). The driver changes inputs to obtain desired performance results. If the car slows down climbing a hill, pressing the accelerator closer to the floorboard will counteract the effects of the hill.

The dashboard and controls have been modeled using Sammi [9], a graphical user interface development tool developed by Kinesix. Two options have been investigated for simulation: petri nets and rules. Petri nets provide a graphical model of concurrent systems. The petri net system which has been used is PERCNET [8], developed by Perceptronics. PERCNET is designed to be used as a knowledge-based graphical simulation environment for modeling and analyzing human-machine tasks. With PERCNET, task models are developed using modified petri nets, a combination of petri nets, frames, and rules. The rule based system which has been used is CLIPS [1], a rule based language primarily used for the design of expert systems, developed by NASA. CLIPS executes in a non-procedural fashion making it ideal for representing random and concurrent events. The automobile system has been prototyped using both the petri net and rule-based systems as simulators and comparisons were made based upon functionality, usability, and performance [7].

3.1 Graphical User Interface

The interface was implemented using Sammi, which provided a graphical means by which to develop the interface. Sammi combines the functions of a graphical user interface toolkit with full network communications support, providing both client/server and peer-to-peer communication options. The format editor of Sammi was used to develop the static automobile display. The display's functionality was tested by building a simulator of the engine and allowing the two processes to communicate through interprocess communication. The graphical interface for the automobile prototype is shown in figure 2.

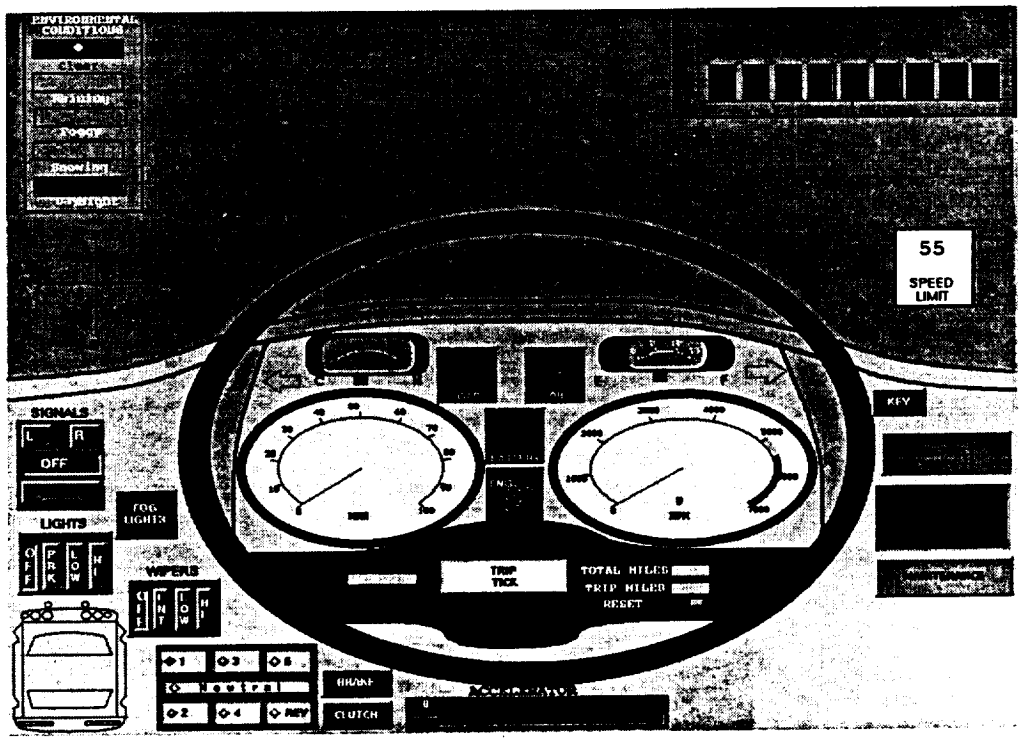


Figure 2 - Graphical Interface for the Automobile Prototype

3.2 Low Fidelity Simulator

The automobile simulator was modeled using PERCNET, a very powerful system analysis software package designed by Perceptronics, Inc. It provides an easy-to-use, graphical interface which allows users to quickly lay out a petri net model of the system. PERCNET uses "modified" petri nets, which allow each state to describe pre-conditions for state transitions, modify global variables, perform function calls and maintain a global simulation time. Figure 3 shows the top-level petri net of the automobile simulator.

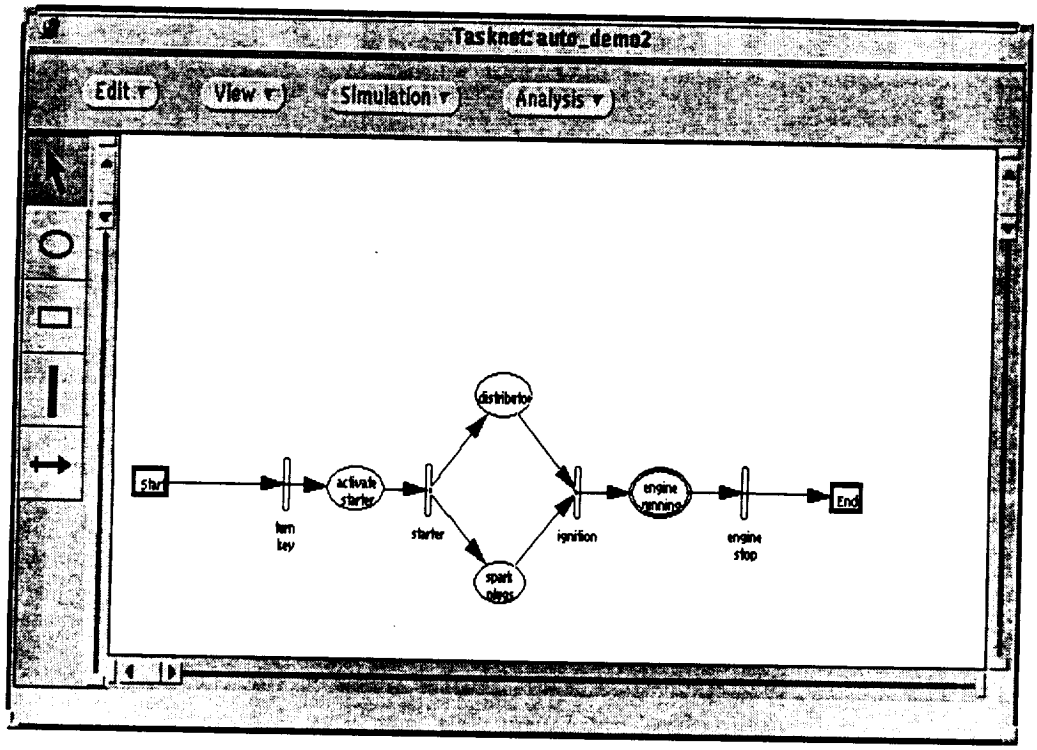


Figure 3 - Top-Level Petri Net of the Automobile Simulator

The starter is the component that is activated by the turning of the key. Before the starter can begin working, however, the key should be turned on, the driver must be wearing his/her seat belt, the car must be in neutral and the battery must have a sufficient charge to start the starter. When all three pre-conditions are true, the starter is activated and control advances to the right in the Petri net. Once the starter has been activated, it must do its part to start the automobile. The starter allows electricity to flow into the distributor where it is channeled into the spark plugs. As long as the starter is functioning, the distributor and spark plugs are activated. Finally, as long as the spark plugs and distributor are working properly and there is gasoline, the spark from the spark plugs ignites the gasoline mixture in the engine and ignition is achieved. Now that ignition has been accomplished, the engine is running. The concentric circles representing the engine_running activity in Figure 3 indicate that the state is shown in a sub-net.

The petri net representing the automobile passes from the ignition portion to the engine running state and remains in the running state until some condition causes the engine to stop running. The engine will stop running if the engine runs out of gas, runs out of oil, the temperature rises above a certain threshold, the key is turned off, the engine stalls (when the automobile is in some gear and the rpms fall below a threshold amount), the battery loses its charge or the fuel pump, oil pump, spark plugs or alternator fail.

The major components of the engine modeled are: fuel pump, oil pump, water pump, distributor, spark plugs, starter, battery, alternator, and fan. The condition of these components is modeled using a boolean variable indicating either that they are functioning or they are not. The boolean variables are then used as conditions within events occurring during the simulation. Details of the Petri Net implementation can be found in [5].

3.3 Interface Evaluation Rules

Formative evaluation is conducted through usability studies. Given a functional prototype and tasks that can be accomplished on that prototype, the designer observes how users interact with the prototype to accomplish those tasks in order to identify improvements for the next design iteration. Evaluation of the interaction is measured in terms of specific parameters including: time to learn to use the system, speed of task performance, rates and types of errors made by users, retention over time, and subjective satisfaction [10]. Analysis of this information will assist in redesign of the system.

The conceptual model of a designer is a description of the system and how the user should interact with it in terms of completing a set of tasks [4]. The user's mental model is a model formed by the user of how the system works, and it guides the user's actions [2]. Most interaction problems occur when the user has an inaccurate model of the system or when the user's model of a system does not correspond with the designer's conceptual model of the system. The evaluation approach which will be discussed in this paper evaluates the user's mental model of the system against the designer's conceptual model.

A rule-based evaluation approach, implemented using CLIPS, was used to develop the conceptual model. The model outlines the specific actions that the user must take in order to complete a task. Evaluation criteria which are embedded in the rules include the existence of certain actions, the sequencing of actions, and the time in which actions should be completed. Throughout the evaluation process, user actions are continuously associated with a set of possibly changing goals. Once a goal has been identified, the user's action in response to that goal are evaluated to determine if a user has performed a task correctly. Tasks may be performed at three levels: expert, intermediate, and novice.

The goal of GUIET [6] is to provide for dynamic evaluation of user actions within the HCI Prototyping Environment. Using GUIET, the process of formative evaluation has more flexibility and takes less time for analysis. The main advantage is that the evaluation of most of the participant's actions are automated. The evaluation is performed at runtime by an expert system. The knowledge base of the system contains the designer's conceptual model of how he/she thinks the user should interact with the prototyped system. Because the knowledge base is not hard coded into the application, it can be dynamically changed according to the needs of the evaluator. This provides the flexibility to evaluate different interfaces with the same evaluation criteria or one interface with different evaluation criteria. This design saves time because the data is automatically collected and analyzed based on the rule based conceptual model. If a new interface is prototyped, the only change that needs to be made with GUIET is changing the knowledge base.

The tasks which the user are asked to perform with the automobile prototype can be divided into two categories: driving the car (i.e., using the controls) and responding to events (e.g., environmental and maintenance). The tasks measured include:

- Starting the car
- Driving forward (including changing gears)
- Driving backward
- Turning
- Stopping (at stop signs, lights, etc.)
- Parking the car

- Increasing and decreasing speed [Responding to speed limit changes]
- Driving uphill and downhill [Responding to hill events]
- Performing maintenance [Responding to maintenance events]
- Responding to environmental conditions

The events which can occur while the user is driving include environmental condition events (e.g., rain, snow, fog, and clear weather), time of day events (e.g., day and night), terrain changes (uphill and downhill), speed limit changes, and maintenance problems (e.g., gas, oil, battery, alternator, and engine). In addition to the events, the participant is given a set of instructions that must be followed. These are in the form of driving directions (e.g., drive 5 miles north and park the car).

Driving the car consists of manipulating graphical objects on the screen. For each of the tasks described above, the designer has determined a set of correct actions that must be made to complete the task. For example, the actions which must be taken for starting the car include:

1. Lock the seatbelt
2. Release emergency brake
3. Depress the brake
4. Depress the clutch
5. Put the gear in neutral
6. Turn the key on

Task correctness is evaluated based mainly on three evaluation criteria: the existence of certain actions, the sequencing of actions, and the time associated with the completions of the actions or task. An integer clock counter is used to indicate the action or event sequence. In the beginning of evaluation, the clock is reset to zero. Every subsequent action taken by the driver would increment the clock by one. Action sequence is important for many driving maneuvers. For example, clutch must be engaged before shifting gears. The evaluation process evaluates the correctness and effectiveness of a driver's interactions with the graphical user interface. User performance can be classified into three levels for most tasks - expert, intermediate, and novice. There may also be no response to a task. A counter is designated for each performance level. Every time a sequence of user actions is classified at a particular level, the associated counter will be incremented by one. The purpose of the evaluation is not to classify or evaluate users, but to evaluate the interface. The classification of users into categories is done to identify the level at which the users are interacting with the system. The goal is to have most, if not all, interactions at what the designer would consider the expert level. If users are not interacting at this level, it is the interface which must be enhanced to improve user performance.

An evaluation rule is designed for each performance level. After a sequence of actions is completed, it will be evaluated based on the rules for the three performance levels. However, only one of the rules would succeed. The rules are organized in a way that the expert level would be tried first, then the intermediate level, and then the novice level. Once a rule has been successfully fired, this sequence of actions will be discarded. The prioritization of these rule is achieved through the *salience* values of CLIPS. Rules for different tasks may contain different evaluation criteria. It depends on the designer's conceptual model of how he/she feels the task needs to be completed.

4.0 HOPKINS ULTRAVIOLET TELESCOPE (HUT) PROTOTYPE

4.1 Interface and Simulator Requirements

An iterative Human Computer Interface Prototyping Process was used to develop the onboard payload displays for the Hopkins Ultraviolet Telescope (HUT). The process began with the identification of known requirements. The requirements gathering phase allowed the development team to collect many of the documents explaining the components and operation of the HUT experiment. These

descriptions of the system differed in the amount of detail presented and in the intended audience. The documents included the Astro Observatory System Overview, the Hopkins Ultraviolet Telescope Handbook, the HUT Payload Operating Procedures for Astro 2, and the Payload Crew Training Complex Experiment Simulator Requirements Document. They described the purpose and history of the experiment, explained in detail each system component, and outlined crew procedures. It was found that while this wide range of information did not always appear pertinent to the payload display development, a more complete understanding of the system was very insightful when faced with design decisions during implementation.

Integration and analysis of the requirements proceeded using the available documents to develop a single, coherent view of each component and the functions of the experiment. The relationships between components and functions were identified and the general operating procedures were outlined. User-centered functional analysis, task analysis, user analysis and operational flows were each used to develop this complete view.

User-centered functional analysis looked at the functions of the system and the expectations of user performance. A deliberate effort was made to keep analysis sufficiently general so that a high-level view of the system could be developed. The main functions of the HUT experiment are to initialize the telescope system and then operate the telescope. The user will be expected to manipulate GUI objects to monitor, calibrate and adjust the telescope subsystems (i.e., power, heaters, vacuum pumps, mirrors, and doors), the spectrometer and camera.

During task analysis, the main functions of the system were described in greater detail. System procedures were also incorporated in analysis and this gave a clearer understanding of the importance of the relationships between components and procedures. For example, it was at this stage that developers understood the difference between the active and inactive states of the telescope. During launch and de-orbit, the telescope is in an inactive state; however, in this state, certain components remain active (e.g., some power is provided to vacuum ion pumps which maintain the vacuum within the spectrometer at all times). This implies that there is some experiment monitoring that goes on even when the telescope is inactive. This also implies that certain display components may not be available at all times (i.e., when the experiment is inactive).

User analysis identified the users of the system as astronauts who are necessarily astronomers. There would be a small number of potential users and these users would undergo rigorous training and evaluation with the interface later in development. The users main focus will be on performing the operating procedures. An on-line procedure reference will be helpful in the final interface. A great deal of time is spent monitoring values (e.g., temperatures, pressure and electrical currents). Much of this monitoring can be built into the display. Simple color changes or popup windows might be used to reduce the amount of monitoring.

The documentation was also used to develop operation flows of the system. These flows were used in the implementation phase of prototyping to develop the simulator. The simulator provides "live" feedback to the user interface during evaluation.

A formal design representation was created using the User Action Notation (UAN) as described in [3]. The UAN provided a means of describing the user actions required to perform tasks identified during task analysis. UAN also allows developers to describe interface feedback and identify interface states. This is the first formal requirements document produced in the process. The complete notation for operation of HUT is provided in Appendix A.

4.2 Graphical User Interface

The graphical interface for HUT was implemented using SAMMI. Two steps were used to implement the display prototypes. The first of these steps involved deciding on the design of the background and

placement of display object groupings. Using the UAN and previous analysis results, the main system components were identified as: power, heaters, mirrors, pumps, spectrometer, camera and television. Each of these components had sub-components that required arrangement within the component display. For example, study of the operating procedures revealed that the spectrometer has fields for mode, mask, minimum/maximum amplitude and minimum/maximum width. These fields were not often accessed but when accessed, the user is asked to view all six values in order. These fields were grouped within the spectrometer display. Component grouping proceeded similarly. The vacuum pumps, which maintain the vacuum within the spectrometer, must be constantly monitored and are often switched on and off while calibrating the spectrometer. The vacuum pump and spectrometer displays were placed next to each other.

The second prototype iteration involved deciding exactly how to represent each of the display objects. Factors contributing to the final design included frequency of access, type and range of values, and aesthetic appearance. The initial telescope display appeared very "busy" after the initial implementation. An attempt was made at this stage to modularize the display and reduce the number of objects visible at one time. Infrequently accessed objects were implemented with pop-up windows. The complexity of the operating procedures encouraged the addition of a "procedure window" that would serve as a quick-reference describing the steps of each procedure. The graphical interface for HUT along with the various pop-up windows can be found in Appendix B.

4.3 Low Fidelity Simulator

The simulator for the HUT prototype was developed using PERCNET. The requirements for this simulator were extracted from the Payload Crew Training Complex Experiment Simulator Requirements Document. The functions of the system were identified along with their inputs and outputs. This activity of requirements gathering occurred concurrently with the interface requirements gathering and analysis discussed in section 4.1. The main system components modeled were power, heaters, mirrors, pumps, spectrometer, camera, and the dedicated experiment processor. The inputs an outputs of each component were modeled, in addition to the system state changes which result from user actions (e.g., survival, observation, etc.). Detailed design of the simulator can be found in Appendix C.

5.0 EVALUATION

The functional components of the environment were integrated successfully and provided the needed functionality. The suite of tools provided for development within of the environment were easy to learn and use. Although they were all different interfaces, they were all graphical. The ideal environment would provide a consistent interface; however, there are no currently available tools which provide all of the support needed for rapid prototyping. The only real problem within the environment was performance. The next sections describe the usability, functionality, and performance of the system in more detail.

5.1 Usability

Most features of the environment are easy-to-learn and use While some knowledge of the X windowing system, petri-net theory, and rule-based languages would benefit designers, much could be done with very minimal knowledge. One difficulty in working with the simulator tool, PERCNET, was the lack of available documentation on the Tool Command Language (TCL). All function calls, calculations, communication and ad-hoc programming are done using this language. Perceptronics provides only minimal documentation on the use of the language within PERCNET making it very difficult to perform anything more than the most basic operations.

5.2 Functionality

The functionality for developing prototypes within the environment is provided. As this project began, PERCNET was a closed package, that is, there was no provision for communicating with other applications. NASA contracted Perceptronics to modify PERCNET to allow for such a feature. The final result was a revision of PERCNET which would allow communication with other applications through the use of sockets. Applications are allowed to request that global variables be retrieved and/or modified. PERCNET essentially opened it's blackboard (i.e., global data store) to other applications. The other application in this case being the server. With this modification and the development of the tool GUIET, all needed functionality was complete.

5.3 Performance

The performance within the Petri Net architecture was not acceptable for real-time interface simulation. Interfaces running within this architecture exhibit a very slow response rate to user actions when PERCNET is executing within its subnets. The PERCNET execution also used excessive amounts of swap space and memory which also affected the refreshing of displays.

Early analysis attempted to find the exact cause of the poor performance; however, only limited work could be done without access to PERCNET's source code. Since PERCNET's code was unavailable, we could only speculate about what was actually happening to cause the slow responses. It was determined that the cause of much of the problem was that PERCNET was trying to do too much. In the PERCNET simulation architecture, PERCNET is actually the data server for the environment. The global blackboard is maintained within PERCNET. The server only provides a mechanism for passing information between PERCNET and other applications. The server is connected to PERCNET by a socket and the server is actually on the "client" end of the connection-oriented socket. The server establishes connections with PERCNET and Sammi and then alternately receives information from each. Any data or commands received from Sammi are passed immediately to PERCNET. Commands from PERCNET for Sammi are passed immediately through, as well. Finally, the server sends Sammi copies of all variables. Since PERCNET is the blackboard server, as well as the simulator, PERCNET's performance would naturally be affected by the added burden. Lastly, the method provided for sending variables to the server was terribly inefficient. When a calculation was performed in the simulation model for a variable that was needed by the interface, that variable was passed to the server whether or not it's value had changed from the previous iteration. No mechanism was provided for restricting the number of redundant values passed across the communication link. As a result, PERCNET passed every value back to the server when only a few had actually changed.

In order to investigate other alternatives to simulation, a rule based architecture was developed [7]. Each of the limitations discussed above was addressed in the design of the server and blackboard in the rule-based architecture. The server program was divided into three portions: blackboard management, Sammi routines, CLIPS routines. The Sammi and CLIPS routines are provided to communicate with the respective applications. These routines map data into a special "blackboard entry" form and pass the data to the blackboard management routines. The blackboard routines also return information to the Sammi and CLIPS routines for routing back to the applications. The blackboard management routines require that each application (many more applications may be supported) register itself initially. Applications are assigned application identification numbers which are used for all subsequent transactions. This application number allows the blackboard to closely monitor which variable values each application needs to see. It also provides a mechanism for installing a priority scheme for updates.

After several functions were added to the rule based model, it performed the same functions as the Petri Net simulator. If a new system is prototyped, the only changes which would be needed are to the knowledge base. The communication link developed for the Sammi-CLIPS architecture uses the blackboard paradigm to improve modularity, flexibility, and efficiency. This form of data manage-

ment stores all information in a central location (the blackboard), and processes communicate by posting and retrieving information from the blackboard. The server manages the blackboard, allowing applications to retrieve current values from the board and to request that a value be changed. The server accepts write requests from valid sources and changes values. The comparison of the two architectures goes much further than comparing the two simulation designs. The design of the communication link significantly affects the flexibility and performance of the architecture.

The overwhelming advantage of the CLIPS and blackboard combination is the flexibility and potential they provide. Features are provided that allow modifications which can affect performance. The ability to tune the performance has allowed the simulation architecture to be tailored to specific running conditions (e.g., machine limitations, network traffic and complexity of the interface being simulated). Several parameters may be modified to alter performance. Tuning tests have improved performance. More detailed performance testing is needed to verify the results.

6.0 Conclusions

Various applications are provided for development within the HCI Prototyping Environment. A developer can rapidly create a prototype of a system and evaluate it with end users by conducting usability tests. The interface can then be iteratively refined through redesign of components which caused performance problems. In the environment a graphical interface for the system can be created, a model or simulation of the system can be built, and evaluation criteria can be specified for the operation of the system. These three components are automatically linked and can be executed. During execution the components send messages to each other through the server. Evaluation of the environment proved that integration of a suite of tools within a prototyping environment is possible and that the needed functionality can be provided. However, evaluation also showed the importance of examining the performance of the tools. It is concluded that the framework be adopted, but there should be additional investigation into the actual applications which would serve as the simulator tool.

REFERENCES

1. CLIPS Reference Manual, NASA Johnson Space Flight Center, Houston, Texas, 1993.

2. Eberts, Ray, *User Interface Design*, Prentice Hall, Englewood Cliffs, New Jersey, 1994.

3. Hix, Deborah, and Hartson H. Rex, *Developing User Interfaces: Ensuring Usability Through Product and Process*, John Wiley & Sons, Inc., New York, 1993.

4. Mayhew, Deborah, *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, New Jersey, 1992.

5. Moore, Loretta, "Assessment of a Human Computer Interface Prototyping Environment," Final Report, Delivery Order No. 16, Basic NASA Contract No. NAS8-39131, NASA Marshall Space Fight Center, Huntsville, AL, 1993.

6. Moore, Loretta A., Chang, Kai, Hale, Joseph P., Bester, Terri, Rix, Thomas, and Wang, Yaowen, "Rule Based Design of Conceptual Models for Formative Evaluation," Paper presented at the Third Conference on CLIPS, Houston, Texas, September 12-14, 1994.

7. Moore, Loretta A., Price, Shannon W., and Hale, Joseph P., "Simulation in a Dynamic Prototyping Environment: Petri Nets or Rules?" Paper presented at the Third Conference on CLIPS, Houston, Texas, September 12-14, 1994.

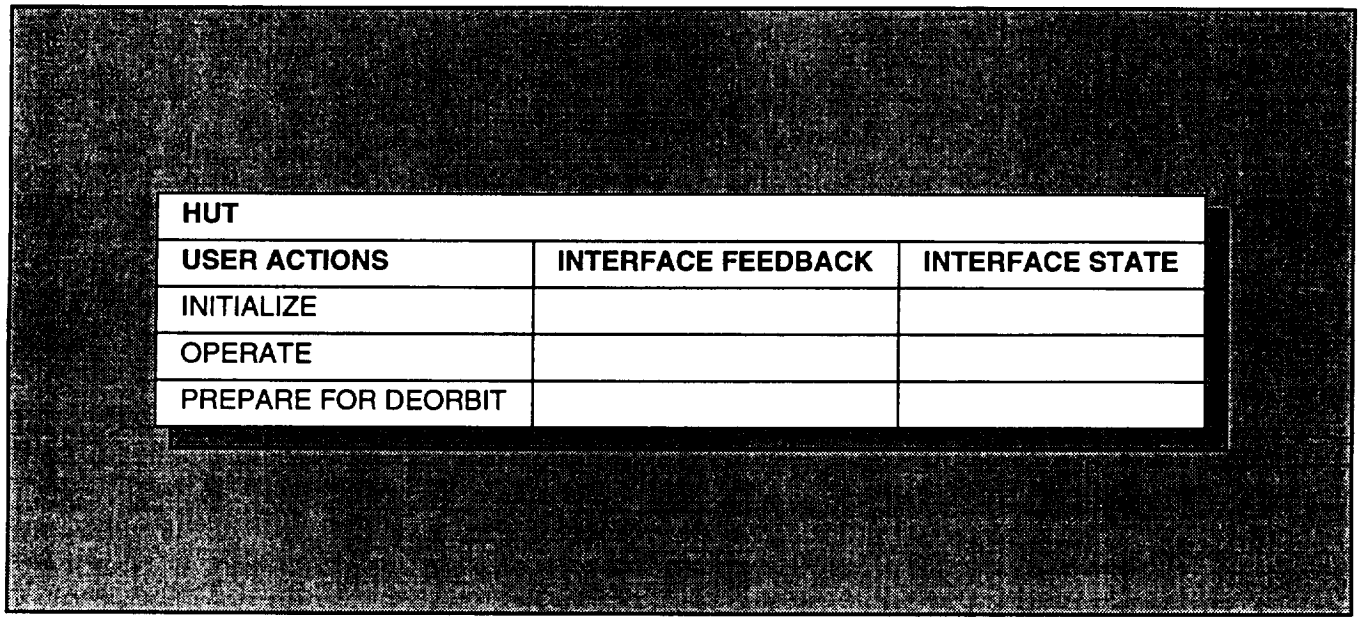
8. PERCNET User's Manual, Perceptronics Inc., Woodland Hills, California, 1992.

9. Sammi API Manual, Kinesix Corporation, Houston Texas, 1992.

10. Shneiderman, Ben, *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Second Edition, Addison-Wesley, Reading, Massachusetts, 1992.

APPENDIX A - User Action Notation for the Hopkins Ultraviolet Telescope

HUT



HUT		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
INITIALIZE		
OPERATE		
PREPARE FOR DEORBIT		

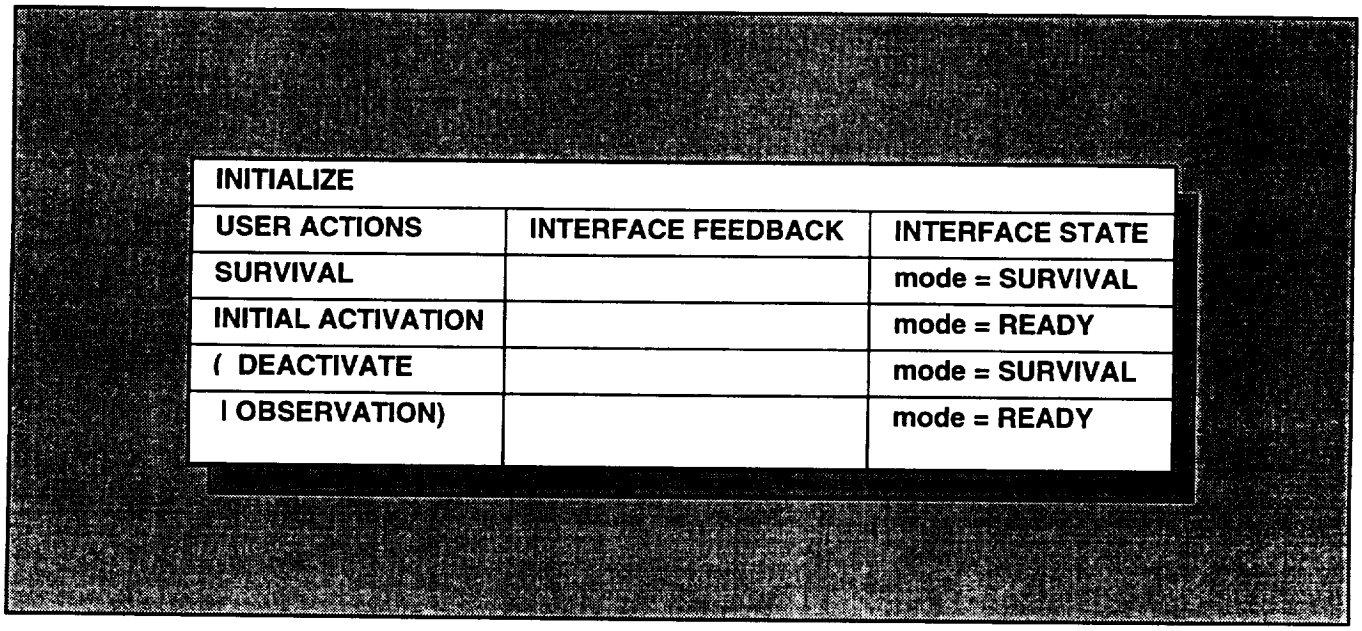
HUT

INITIALIZE		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
SURVIVAL		mode = SURVIVAL
INITIAL ACTIVATION		mode = READY
(DEACTIVATE		mode = SURVIVAL
I OBSERVATION)		mode = READY

OPERATE		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
(REACTIVATE		mode = READY
(OBSERVATION		
I TEST) *		
DEACTIVATE)		mode = SURVIVAL

PREPARE FOR DEORBIT		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
command (+28V BUS OFF)	+28V BUS OFF	
command (HEATERS OFF)	HEATERS OFF	mode = SHUTDOWN
check_vacuum_pump (1, ON, -5.00 torr, 0.15A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00A)		

HUT - INITIALIZE



INITIALIZE		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
SURVIVAL		mode = SURVIVAL
INITIAL ACTIVATION		mode = READY
(DEACTIVATE		mode = SURVIVAL
I OBSERVATION)		mode = READY

INITIALIZE

SURVIVAL		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
HEATER_POWER_ON		
command (+28V BUS ON)	+28V BUS = ON	mode = SURVIVAL
check_vacuum_pump (1, ON, -5.00t, 0.15A		
check_vacuum_pump (2, OFF, -7.13t, 0.00A)		

INITIAL_ACTIVATION		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(mode = SURVIVAL && check (+Y SHUTTER_DOOR CLOSED) && check (-Y SHUTTER_DOOR CLOSED) && check (SMALL_APERTURE_ DOOR CLOSED) && check (CCTV source HUT)):		
MAIN_POWER_ON	MAIN POWER = ON	
command (RESET DEP)	DEP_STATE = RESET	
command (LOAD DEP)	DEP_STATE = LOAD DEP = ACTIVE	Inactive 8 Minutes DEP = ACTIVE
command (SP POWER ON)	SP POWER = ON	
LOAD_SP	SP = LOADING	Inactive 3 MInutes SP = READY
check (HEATER_MODE = SLAVED)		
check (ELECTRONICS_HEATER = ON)		
TEST_CAMERA		TV_FUNCTIONALITY _TEST = COMPLETE
MIRROR_BACKLASH_MOVEMENT		MIRROR_BACKLASH = COMPLETE
OBSERVATION_VERIFY		
EXPERIMENT_OUTGASSING		OUTGASSING = COMPLETE
TEST_ SPECTROMETER		mode = READY

INITIALIZE

DEACTIVATE		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(MAIN_POWER = ON && DEP = ACTIVE) :		
command (SHUTDOWN)	DEP = INACTIVE DEP_STATE = OFF	
check (ILLUMINATION_LAMP OFF)		
command (SP_POWER OFF)	SP_POWER = OFF	
command (MAIN_POWER OFF)	MAIN_POWER = OFF	
check_vacuum_pump (1, ON, -5.00 torr, 0.15 A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00 A)		

OBSERVATION		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(TV_FOCUS_ALIGN		
I SPECTROMETER_FOCUS		
I CALIBRATE_APERTURE_&_BOS		
I CALIBRATE_12ARCSEC_CTR		
I CALIBRATE_11x60ARCSEC_CTR)		
SCIENCE_OBSERVATION		

INITIALIZE-SURVIVAL

HEATER_POWER_ON		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
check_vacuum_pump (1, ON, -5.0 torr, ---)		
command (HEATERS ON)	HEATERS ON	
check (HEATER_CURRENT < 16.5 A)		
check (HEATER_CONV_ CURRENT 0.110 - 0.180 A)		
check (STRUCTURE_ TEMPERATURE > 0°C)		
check (ELECTRONICS_ TEMPERATURE >= -10°C))		
check_vacuum_pump (1, ON, ----, 0.35 A)		
WAIT (30 seconds)		

INITIALIZE-INITIAL_ACTIVATION

MAIN_POWER_ON		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (MAIN POWER ON)	MAIN POWER = ON	
check (+5V (5.0 - 5.5 V))		
check (+12V (11.5 - 12.7 V))		
check (-12V (-11.9 - -12.9 V))		
check (+18V (+17.8 - 19.6 V))		

LOAD_SP		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (LOAD SP)	SP = LOADING	Inactive 3 Minutes SP = READY
check (SP MODE HISTOGRAM)		
check (SP MASK 0)		
check (MIN AMP (8))		
check (MAX AMP (62))		
check (MIN WIDTH (2))		
check (MAX WIDTH (20))		

INITIALIZE-INITIAL_ACTIVATION

TEST_CAMERA		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (EARTH_BOS DISABLE)	EARTH_BOS = DISABLED	
check (FILTER OFF)		
check (SUN_BOS ENABLED)		
check (SMALL_APERTURE_ DOOR CLOSED)		
check (+Y_SHUTTER_DOOR CLOSED)		
check (-Y_SHUTTER_DOOR CLOSED)		
command (CAMERA_POWER ON)	CAMERA_POWER = ON	
check (EXPOSURE 0, CAMERA_HV 0, ZOOM = OFF, WHITE_LEVEL = 0, BLACK_LEVEL = 0, SOFT_INT = OFF, FORCE_SYNC = OFF)		
command (TV_MODE FIELD)	TV_MODE = FIELD	
command (FILTER ND2)	FILTER = ND2	
command (CAMERA_HV 5)	CAMERA_HV = 5	
command (ILLUMINATION_LAMP ON)	ILLUMINATION_LAMP = ON	
WAIT (FOCC aknowledge)		
command (EXPOSURE 5)	EXPOSURE = 5	
command (BLACK_LEVEL x) /*adjust*/	BLACK_LEVEL = x	
command (WHITE_LEVEL x) /*adjust*/	WHITE_LEVEL = x	
command (SOFTWARE_INTEGRATION ON)	SW_INTEGRATION = ON	
MONITOR CCTV /* visual */	Center of Images Brightens	
command (SOFTWARE INTEGRATION OFF)	SW_INTEGRATION = OFF	
command (TV_MAGNITUDE 10)	TV_MAGNITUDE = 10	
MONITOR CCTV /* visual */	Fiducials Dim	
command (TV_MAGNITUDE 12)	TV_MAGNITUDE = 12	
TEST_TV_MODES		
command (ILLUMINATION_LAMP OFF)	ILLUMINATION_LAMP = OFF	
command (TV_MAGNITUDE = -5)	TV_MAGNITUDE = -5	
command (EARTH_BOS ENABLE)	EARTH_BOS = ENABLED	TV_FUNCTIONALITY_ TEST = COMPLETE

INITIALIZE - INITIAL_ACTIVATION

OBSERVATION_VERIFY		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (PREVIEW activation sequence 15)	SEQ 15 Display	
command (DISPLAY current status)	CURRENT SEQ Display	
command (SETUP "dummy" observation)	OBS SEQ Form Display	
command (LOCATE_TYPE MANUAL)	LOCATE_TYPE = MANUAL	
MOVE_CURSOR	CURSOR_MOVEMENT	
command (BEGIN)	DEP_STATE = OBSERVE	
check (time decreasing)	time = 600 seconds	
WAIT (150 seconds)	time = 450 seconds SP_MASK and SP_MODE change automatically	
WAIT (150 seconds)	time = 300 seconds SP_MASK and SP_MODE change automatically	
command (QUIT)	DEP_STATE = SLEW	
command (TV_MODE SPECTRUM)	TV_MODE = SPECTRUM	
command (ENABLE_DOORS)	NONE	

EXPERIMENT_OUTGASSING		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(DEP = READY):		
command (SMALL_APERTURE_DOOR 50cm^2)	SMALL_APERTURE_DOOR = OPEN	
command (+Y_SHUTTER_DOOR OPEN	+Y_SHUTTER_DOOR = OPEN	
command (-Y_SHUTTER_DOOR OPEN)	-Y_SHUTTER_DOOR = OPEN	OUTGASSING = COMPLETE

INITIALIZE - INITIAL_ACTIVATION

TEST_SPECTROMETER		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(OUTGASSING = COMPLETE && TV_FUNCIONALITY_ TEST = COMPLETE) :		
check (SP_POWER ON, SP_MODE 4, SP_MASK 0, MIN_AMP 8, MAX_AMP = 62, MIN_WIDTH = 2, MAX_WIDTH = 20)		
check_vacuum_pump (2, ON, -5.70 torr, 0.12A)		
command (VACUUM_PUMP1 OFF)		
check_vacuum_pump (1, OFF, -7.03 torr, 0.00 A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00 A)		
command (DETECTOR ON)	DETECTOR = ON	
check (MCP_HV_ADJUSTMENT 0 2.63 ±0.050)		
check (PHOS_HV_ADJUSTMENT 0 6.0 ±0.075)		
check (CCTV noise level) /*auricular */		
check (PHOTON < 10)		
WAIT (30 seconds)		
command (MCP_HV x) /* adjust */	MCP_HV = x	
WAIT (30 seconds)		
command (PHOS_HV x) /* adjust */	PHOS_HV = x	
WAIT (30 seconds)		
command (CALIBRATION_LAMP ON)	CALIBRATION_LAMP = ON	
WAIT (30 seconds)		
command (SP_MODE SINGLE_SCAN)	SP_MODE = SINGLE_SCAN	
WAIT (5 minutes)		
command (SP_MODE = HIGH_ TIME_RESOLUTION)	SP_MODE = HIGH_TIME RESOLUTION	
command (CALIBRATION_LAMP OFF)	CALIBRATION_LAMP = OFF	
command (DETECTOR OFF)	DETECTOR = OFF	
command (SET DETECTOR DEFAULTS)		
command (DETECTOR ON)	DETECTOR = ON	

INITIALIZE - INITIAL_ACTIVATION

TEST_SPECTROMETER (CONTINUED)		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.00 torr, 0.15A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00 A)		
(command (SLIT_WHEEL x)) x=7-4	SLIT_WEEL = x	
command (VACUUM_PUMP_1 OFF)		
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.00 torr, 0.15 A)		
WAIT (10 minutes)		
(command (SLIT_WHEEL x)) x = 3-0	SLIT_WHEEL = x	

INITIALIZE - OBSERVATION

TV_FOCUS_ALIGN		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(WUPPE = inoperative && CCTV souce is HUT) :		
command (PREVIEW SEQ ##)	Display F&A Target #1	
command (EARTH_BOS DISABLE)	EARTH_BOS = DISABLED	
IPS MOVEMENT		
command (LOCATE_TYPE NONE)	LOCATE_TYPE = NONE	
command (TV_MODE DOWN ZOOM)	TV_MODE = DOWN ZOOM	
command (SOFTWARE_ INTEGRATION ON)	SW_INTEGRATION = ON	
RECORD IMAGE SIZE DATA		
(command (Δx #) /* adjust */	Δx = #	
command (MIRROR_MODE MOVE)	MIRROR_MODE = MOVE	
command (MIRROR_MODE STOP))*	MIRROR_MODE = STOP	
SET_MIRRORS		
command (MIRROR_MODE FOCUS)	MIRROR_MODE = FOCUS	
command (EARTH_BOS ENABLE)	EARTH_BOS = ENABLED	

SET_MIRRORS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(command (-Z = nnnn)	-Z = nnnn	
I command (-Y+Z = nnnn)	-Y+Z = nnnn	
I command (+Y+Z = nnnn))*	+Y+Z = nnnn	
command (START)		Inactive during motion
command (MIRROR_MODE 0)	MIRROR_MODE = 0	

INITIALIZE - OBSERVATION

SPECTROMETER_FOCUS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(HUT spectrometer focus target acquired && DEP_STATE = OBSERVE) :		
command (SOFTWARE_ INTEGRATION ON)	SOFTWARE_INTEGRATION = ON	
record (+Y+Z)		
record (START_TIME)		
WAIT (200 seconds)		
record (STOP_TIME)		
command (Δx = #)	Δx = #	
command (MIRROR_MODE MOVE)	MIRROR_MODE = MOVE	
(command (MIRROR_MODE STOP))*	MIRROR_MODE = STOP	
command (MIRROR_MODE FOCUS)	MIRROR_MODE = FOCUS	
command (SOFTWARE_ INTGRATION OFF)	SOFTWARE_INTEGRATION = OFF	

CALIBRATE_APERTURE_&_BOS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(EARTH_BOS = ENABLED && DETECTOR = OFF) :		
command (SOFTWARE_INTEGRATION ON)	SW_INTEGRATION = ON	
command (TV_MAGNITUDE = #) /*adjust */	TV_MAGNITUDE = #	
(command (SLIT_WHEEL n)		
command (WHITE_LEVEL = x) /*adjust */	WHITE_LEVEL = x	
command (BLACK_LEVEL = x) /* adjust */	BLACK_LEVEL = x	
WAIT (100 seconds)) n = 3,2,1,7,6,5,6,7,1,2,3		Aperture_Centers = COMPLETE

INITIALIZE - OBSERVATION

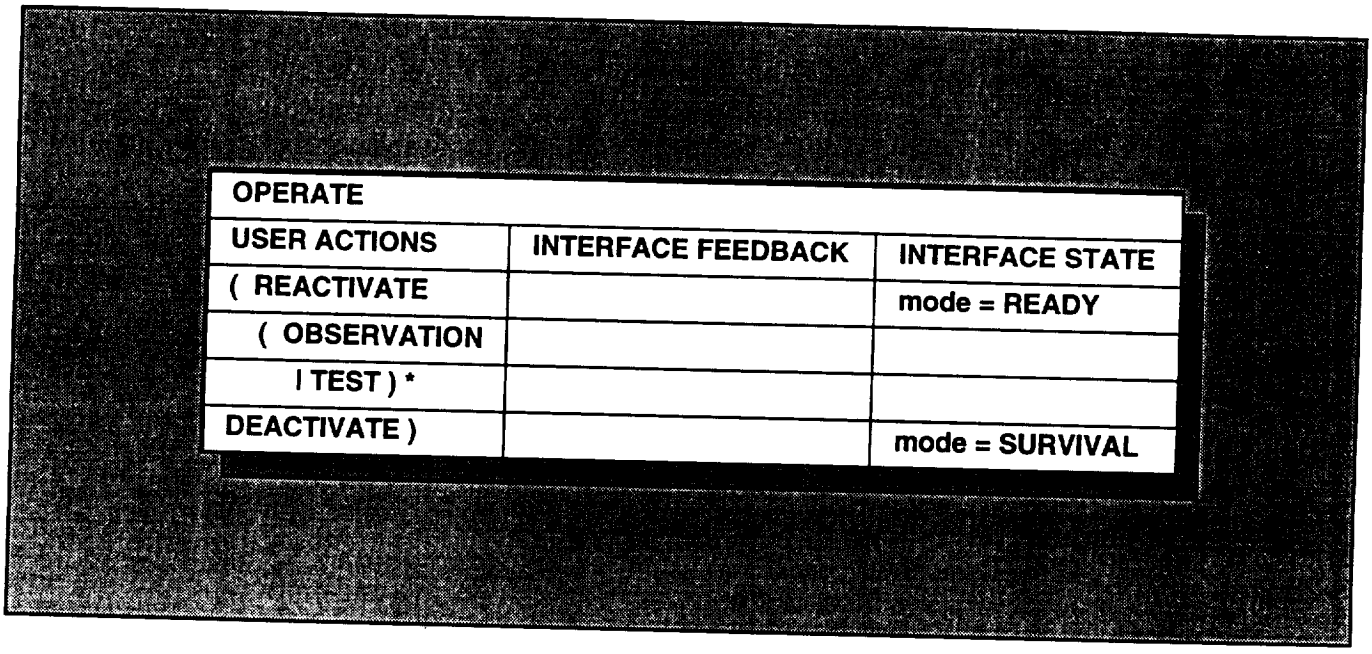
CALIBRATE_12ARCSEC_CTR		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(Aperture_Centers = COMPLETE && DEP_STATE = OBSERVE && Aperture 1 has moved into place) :		
IPS MOVEMENT		
command (SLIT_WHEEL 9x120)	SLIT_WHEEL = 9x120	
command (SLIT_WHEEL 30 diam)	SLIT_WHEEL = 30 diam	
check (PITCH/YAW ERROR <= ±0.5)		
IPS MOVEMENT		

CALIBRATE_11x60ARCSEC_CTR		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(Aperture_Centers = COMPLETE && DEP_STATE = OBSERVE && Aperture 6 has moved into place) :		
IPS MOVEMENT		
command (SLIT_WHEEL 18x120 CAF2 FILTER)	SLIT_WHEEL = 18x120 CAF2 FILTER	
command (SLIT_WHEEL 18x120)	SLIT_WHEEL = 18x120	
check (PITCH/YAW ERROR <= ±0.5)		
IPS MOVEMENT		

INITIALIZE - OBSERVATION

SCIENCE_OBSERVATION		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(mode = READY && HUT - only operations && Payload Bay floodlights OFF		
(command (PREVIEW SEQ ##))*	Display SEQ ##	
command (CURRENT)		
command (SETUP)	DEP_STATE = LOCATE	
IPS MOVEMENT		
command (BEGIN)	DEP_STATE = OBSERVE	
((command (PAUSE)	DEP_STATE = PAUSE	
command (PROCEED))	DEP_STATE = OBSERVE	
! command (PREVIEW SEQ ##))*	Display SEQ ##	
((time = 0)	DEP_STATE = SLEW	
! (command (PAUSE)	DEP_STATE = PAUSE	
command (QUIT))	DEP_STATE = SLEW	
! command (QUIT))	DEP_STATE = SLEW	
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.00 torr, 0.15 A)		

HUT - OPERATE



OPERATE		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(REACTIVATE		mode = READY
(OBSERVATION		
I TEST) *		
DEACTIVATE)		mode = SURVIVAL

OPERATE

REACTIVATE		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(mode = SURVIVAL && +28V_BUS = ON && +28V_CURENT (0.2-0.3A) && HEATERS ON && HEATER_CURRENT < 16.5A && check_vacuum_pump (1, ON -5.00 torr, 0.15A) && check_vacuum_pump (2, OFF, -7.13 torr, 0.00A) :		
MAIN_POWER_ON	MAIN_POWER = ON	
command (RESET DEP)	DEP_STATE = RESET	
command (LOAD DEP)	DEP_STATE = READY	Inactive 8 minutes DEP = ACTIVE
command (SP_POWER ON)	SP_POWER = ON	
LOAD_SP	SP = LOADED	
check (HEATER_MODE = SLAVED)		
check (ELECTRONICS_HEATER = ON)		
CAMERA_POWER_ON	CAMERA_POWER = ON	
DETECTOR_POWER_ON	DETECTOR = ON	
SET_MIRROR		mode = READY

OBSERVATION		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(TV_FOCUS_ALIGN		
I SPECTROMETER_FOCUS		
I CALIBRATE_APERTURE_&_BOS		
I CALIBRATE_12ARCSEC_CTR		
I CALIBRATE_11x60ARCSEC_CTR)		
SCIENCE_OBSERVATION		

OPERATE

TEST		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(CAMERA_SENSITIVITY_TEST		
| REDUCED_APERTURE_50		
| REDUCED_APERTURE_1		
| EXPERIMENT_OUTGASSING		
| MONITOR_SAA)		

DEACTIVATE		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(MAIN_POWER = ON && DEP = ACTIVE) :		
command (SHUTDOWN)	DEP = INACTIVE DEP_STATE = OFF	
check (ILLUMINATION_LAMP OFF)		
command (SP_POWER OFF)	SP_POWER = OFF	
command (MAIN_POWER OFF)	MAIN_POWER = OFF	
check_vacuum_pump (1, ON, -5.00 torr, 0.15 A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00 A)		

OPERATE - REACTIVATE

MAIN_POWER_ON		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (MAIN POWER ON)	MAIN POWER = ON	
check (+5V (5.0 - 5.5 V))		
check (+12V (11.5 - 12.7 V))		
check (-12V (-11.9 - -12.9 V))		
check (+18V (+17.8 - 19.6 V))		

LOAD_SP		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (LOAD SP)	SP = LOADING	Inactive 3 Minutes SP = READY
check (SP MODE HISTOGRAM)		
check (SP MASK 0)		
check (MIN AMP (8))		
check (MAX AMP (62))		
check (MIN WIDTH (2))		
check (MAX WIDTH (20))		

OPERATE - REACTIVATE

CAMERA_POWER_ON		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (CAMERA_POWER ON)	CAMERA_POWER = ON	
check (EXPOSURE = 0, CAMERA_HV = 0, ZOOM = OFF, WHITE_LEVEL = 0, BLACK_LEVEL = 0, SOFTWARE_INTEGRATION = OFF, FORCE_SYNC = OFF)		
command (ENABLE_DOORS)	NONE	
command (+Y_SHUTTER_DOOR OPEN)	+Y_SHUTTER_DOOR = OPEN	
command (-Y_SHUTTER_DOOR OPEN)	-Y_SHUTTER_DOOR = OPEN	
WAIT (3 minutes)		

DETECTOR_POWER_ON		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (TV_MODE SPECTRUM)	TV_MODE = SPECTRUM	
check_vacuum_pump (1, ON, -5.00 torr, 0.15A)		
command (VACUUM_PUMP_1 OFF)		
check_vacuum_pump (1, OFF, -7.03 torr, 0.00 A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00 A)		
command (SET DETETOR DEFAULTS)		
command (DETECTOR ON)	DETECTOR = ON	
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON -5.00 torr, 0.15 A)		

SET_MIRRORS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(command (-Z = nnnn)	-Z = nnnn	
I command (-Y+Z = nnnn)	-Y+Z = nnnn	
I command (+Y+Z = nnnn))*	+Y+Z = nnnn	
command (START)		Inactive during motion
command (MIRROR_MODE 0)	MIRROR_MODE = 0	

OPERATE - OBSERVATION

TV_FOCUS_ALIGN		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(WUPPE = inoperative && CCTV souce is HUT) :		
command (PREVIEW SEQ ##)	Display F&A Target #1	
command (EARTH_BOS DISABLE)	EARTH_BOS = DISABLED	
IPS MOVEMENT		
command (LOCATE_TYPE NONE)	LOCATE_TYPE = NONE	
command (TV_MODE DOWN ZOOM)	TV_MODE = DOWN ZOOM	
command (SOFTWARE_ INTEGRATION ON)	SW_INTEGRATION = ON	
RECORD IMAGE SIZE DATA		
(command (Δx #) /* adjust */	Δx = #	
command (MIRROR_MODE MOVE)	MIRROR_MODE = MOVE	
command (MIRROR_MODE STOP))*	MIRROR_MODE = STOP	
SET_MIRRORS		
command (MIRROR_MODE FOCUS)	MIRROR_MODE = FOCUS	
command (EARTH_BOS ENABLE)	EARTH_BOS = ENABLED	

SET_MIRRORS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(command (-Z = nnnn)	-Z = nnnn	
I command (-Y+Z = nnnn)	-Y+Z = nnnn	
I command (+Y+Z = nnnn))*	+Y+Z = nnnn	
command (START)		Inactive during motion
command (MIRROR_MODE 0)	MIRROR_MODE = 0	

OPERATE - OBSERVATION

SPECTROMETER_FOCUS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(HUT spectrometer focus target acquired && DEP_STATE = OBSERVE) :		
command (SOFTWARE_ INTEGRATION ON)	SOFTWARE_INTEGRATION = ON	
record (+Y+Z)		
record (START_TIME)		
WAIT (200 seconds)		
record (STOP_TIME)		
command (Δx = #)	Δx = #	
command (MIRROR_MODE MOVE)	MIRROR_MODE = MOVE	
(command (MIRROR_MODE STOP))*	MIRROR_MODE = STOP	
command (MIRROR_MODE FOCUS)	MIRROR_MODE = FOCUS	
command (SOFTWARE_ INTGRATION OFF)	SOFTWARE_INTEGRATION = OFF	

CALIBRATE_APERTURE_&_BOS		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(EARTH_BOS = ENABLED && DETECTOR = OFF) :		
command (SOFTWARE_INTEGRATION ON)	SW_INTEGRATION = ON	
command (TV_MAGNITUDE = #) /*adjust */	TV_MAGNITUDE = #	
(command (SLIT_WHEEL n)		
command (WHITE_LEVEL = x) /*adjust */	WHITE_LEVEL = x	
command (BLACK_LEVEL = x) /* adjust */	BLACK_LEVEL = x	
WAIT (100 seconds)) n = 3,2,1,7,6,5,6,7,1,2,3		Aperture_Centers = COMPLETE

OPERATE - OBSERVATION

CALIBRATE_12ARCSEC_CTR		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
(Aperture_Centers = COMPLETE && DEP_STATE = OBSERVE && Aperture 1 has moved into place) :		
IPS MOVEMENT		
command (SLIT_WHEEL 9x120)	SLIT_WHEEL = 9x120	
command (SLIT_WHEEL 30 diam)	SLIT_WHEEL = 30 diam	
check (PITCH/YAW ERROR <= ±0.5)		
IPS MOVEMENT		

CALIBRATE_11x60ARCSEC_CTR		
USER ACTIONS	**INTERFACE FEEDBACK**	**INTERFACE STATE**
(Aperture_Centers = COMPLETE && DEP_STATE = OBSERVE && Aperture 6 has moved into place) :		
IPS MOVEMENT		
command (SLIT_WHEEL 18x120 CAF2 FILTER)	SLIT_WHEEL = 18x120 CAF2 FILTER	
command (SLIT_WHEEL 18x120)	SLIT_WHEEL = 18x120	
check (PITCH/YAW ERROR <= ±0.5)		
IPS MOVEMENT		

OPERATE - OBSERVATION

SCIENCE_OBSERVATION		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(mode = READY && HUT - only operations && Payload Bay floodlights OFF		
(command (PREVIEW SEQ ##))*	Display SEQ ##	
command (CURRENT)		
command (SETUP)	DEP_STATE = LOCATE	
IPS MOVEMENT		
command (BEGIN)	DEP_STATE = OBSERVE	
((command (PAUSE)	DEP_STATE = PAUSE	
command (PROCEED))	DEP_STATE = OBSERVE	
I command (PREVIEW SEQ ##))*	Display SEQ ##	
((time = 0)	DEP_STATE = SLEW	
I (command (PAUSE)	DEP_STATE = PAUSE	
command (QUIT))	DEP_STATE = SLEW	
I command (QUIT))	DEP_STATE = SLEW	
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.00 torr, 0.15 A)		

OPERATE - TEST

CAMERA_SENSITIVITY_TEST		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(HUT Camera Sensitivity Target Acquired && DEP_STATE = OBSERVE)		
command (TV_MODE DOWN FIELD)	TV_MODE = DOWN FIELD	
(command (TV_MAGNITUDE x)) x = 10, 11, 12, 13	TV_MAGNITUDE = x	
command (TV_MODE DOWN SPECTRUM)	TV_MODE = DOWN SPECTRUM	
(command (TV_MAGNITUDE x)) x = 14, 15, 16	TV_MAGNITUDE = x	

REDUCED_APERTURE_50		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(+Y_SHUTTER_DOOR OPEN && -Y_SHUTTER_DOOR OPEN) :		
check_vacuum_pump (1, ON, -5.5 torr, 0.15 A)		
check_vacuum_pump (2, OFF, -7.0 torr, 0.00 A)		
command (SMALL_APERTURE_DOOR 50cm^2)	SMALL_APERTURE_DOOR = 50cm^2	
command (SLIT_WHEEL 175 diam)	SLIT_WHEEL = 175 diam	
WAIT (2 minutes)		
check_vacuum_pump (1, ON, -5.5 torr, 0.15A)		
command (+Y_SHUTTER_DOOR CLOSE)	+Y_SHUTTER_DOOR = CLOSED	
command (-Y_SHUTTER_DOOR_CLOSE)	-Y_SHUTTER_DOOR = CLOSED	
command (VACUUM_PUMP_1 OFF)		
WAIT (10 minutes)		
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.5 torr, 0.15 A)		
command (SLIT_WHEEL BLANK)	SLIT_WHEEL = BLANK	
command (FILTER = ND6)	FILTER = ND6	
command (+Y_SHUTTER_DOOR OPEN)	+Y_SHUTTER_DOOR = OPEN	
command (-Y_SHUTTER_DOOR OPEN)	-Y_SHUTTER_DOOR = OPEN	
command (SMALL_APERTURE_DOOR CLOSE)	SMALL_APERTURE_DOOR = CLOSED	

OPERATE - TEST

REDUCED_APERTURE_1		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(+Y_SHUTTER_DOOR OPEN && -Y_SHUTTER_DOOR OPEN) :		
check_vacuum_pump (1, ON, -5.5 torr, 0.15 A)		
check_vacuum_pump (2, OFF, -7.0 torr, 0.00 A)		
command (SMALL_APERTURE_DOOR 1cm^2)	SMALL_APERTURE_DOOR = 1cm^2	
command (SLIT_WHEEL 18 diam)	SLIT_WHEEL = 18 diam	
check_vacuum_pump (1, ON, -5.5 torr, 0.15A)		
command (+Y_SHUTTER_DOOR CLOSE)	+Y_SHUTTER_DOOR = CLOSED	
command (-Y_SHUTTER_DOOR CLOSE)	-Y_SHUTTER_DOOR = CLOSED	
command (-Y_SHUTTER_DOOR CRACK)	-Y_SHUTTER_DOOR = CRACKED	
command (VACUUM_PUMP_1 OFF)		
WAIT (5 minutes)		
command (VACUUM_PUMP_1 ON)		
check_vacuum_pump (1, ON, -5.5 torr, 0.15 A)		
WAIT (2 minutes)		
command (SLIT_WHEEL BLANK)	SLIT_WHEEL = BLANK	
command (FILTER = ND6)	FILTER = ND6	
command (+Y_SHUTTER_DOOR OPEN)	+Y_SHUTTER_DOOR = OPEN	
command (-Y_SHUTTER_DOOR OPEN)	-Y_SHUTTER_DOOR = OPEN	
command (SMALL_APERTURE_DOOR CLOSE)	SMALL_APERTURE_DOOR = CLOSED	

EXPERIMENT_OUTGASSING		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(DEP = READY):		
command (SMALL_APERTURE_DOOR 50cm^2)	SMALL_APERTURE_DOOR = OPEN	
command (+Y_SHUTTER_DOOR OPEN	+Y_SHUTTER_DOOR = OPEN	
command (-Y_SHUTTER_DOOR OPEN)	-Y_SHUTTER_DOOR = OPEN	OUTGASSING = COMPLETE

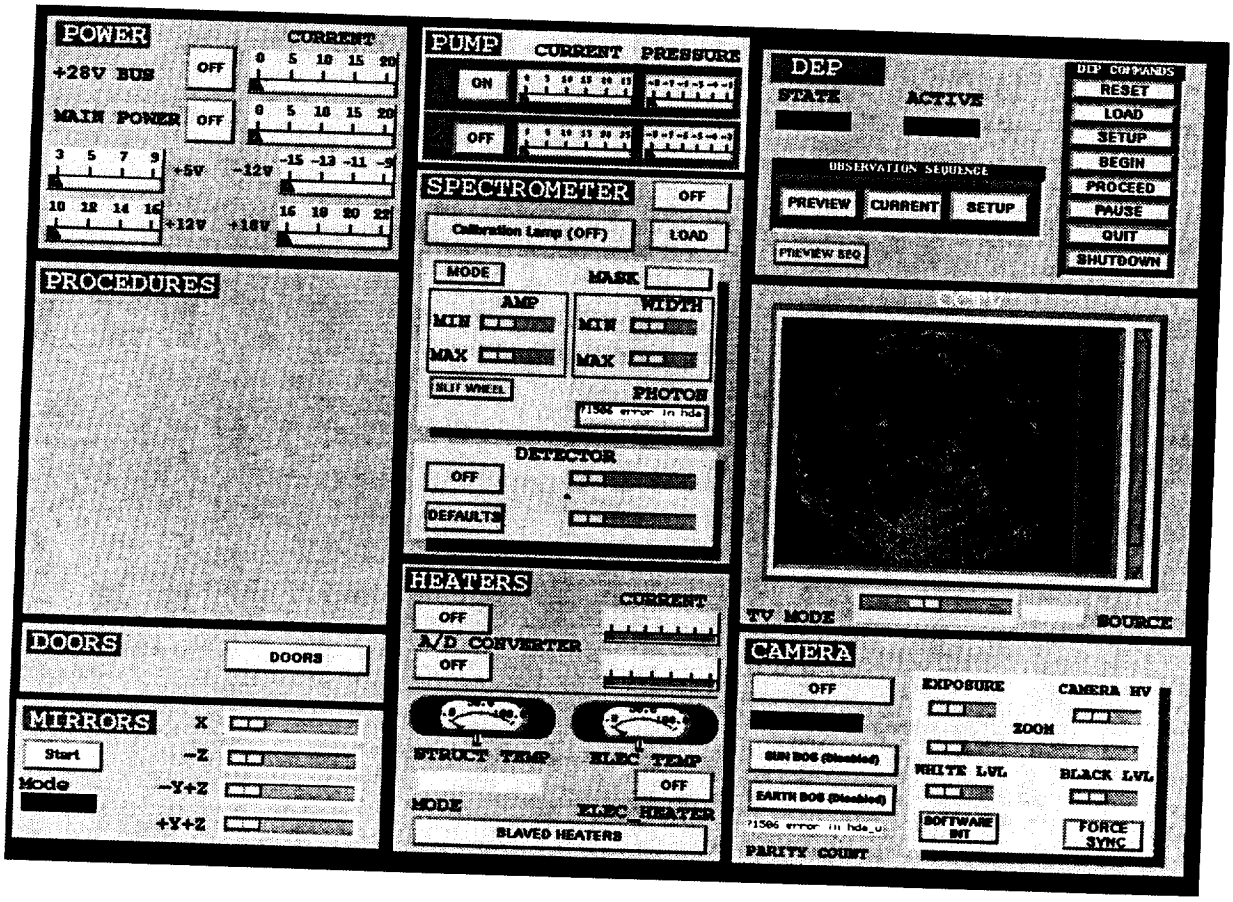
OPERATE - TEST

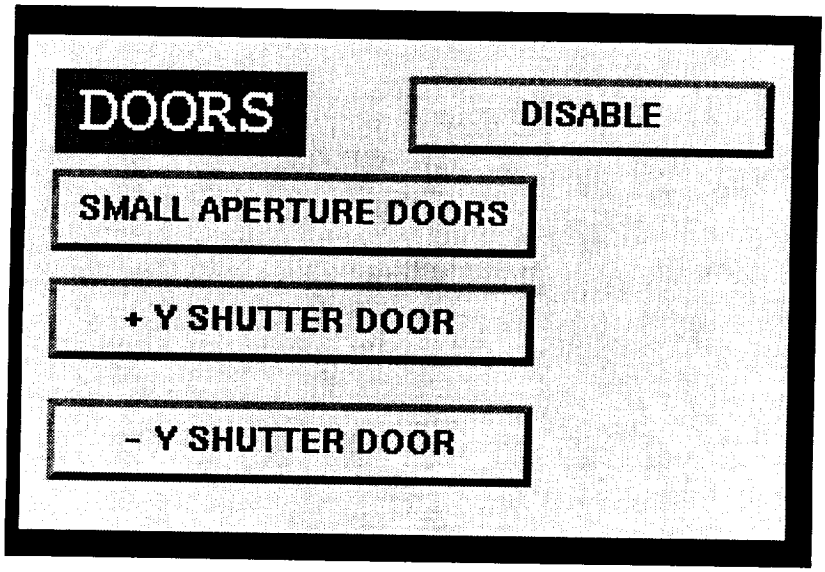
MONITOR_SAA		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
(DEP_STATE = READY && SP = READY) :		
check (PARITY_COUNT = 0)		
check (STATUS = "DET OFF -> SP HIBER)		
command (EARTH_BOS ENABLE)	EARTH_BOS = ENABLED	
command (TV_MODE FIELD)	TV_MODE = FIELD	
command (CAMERA_HV x) /* adjust */	CAMERA_HV = x	
command (EXPOSURE x) /* adjust */	EXPOSURE = x	
check (STATUS = "OFF -> - DET")		
command (TV_MAGNITUDE -5)	TV_MAGNITUDE = -5	
check (PARITY_COUNT 0)		
command (EARTH_BOS ENABLE)	EARTH_BOS = ENABLED	

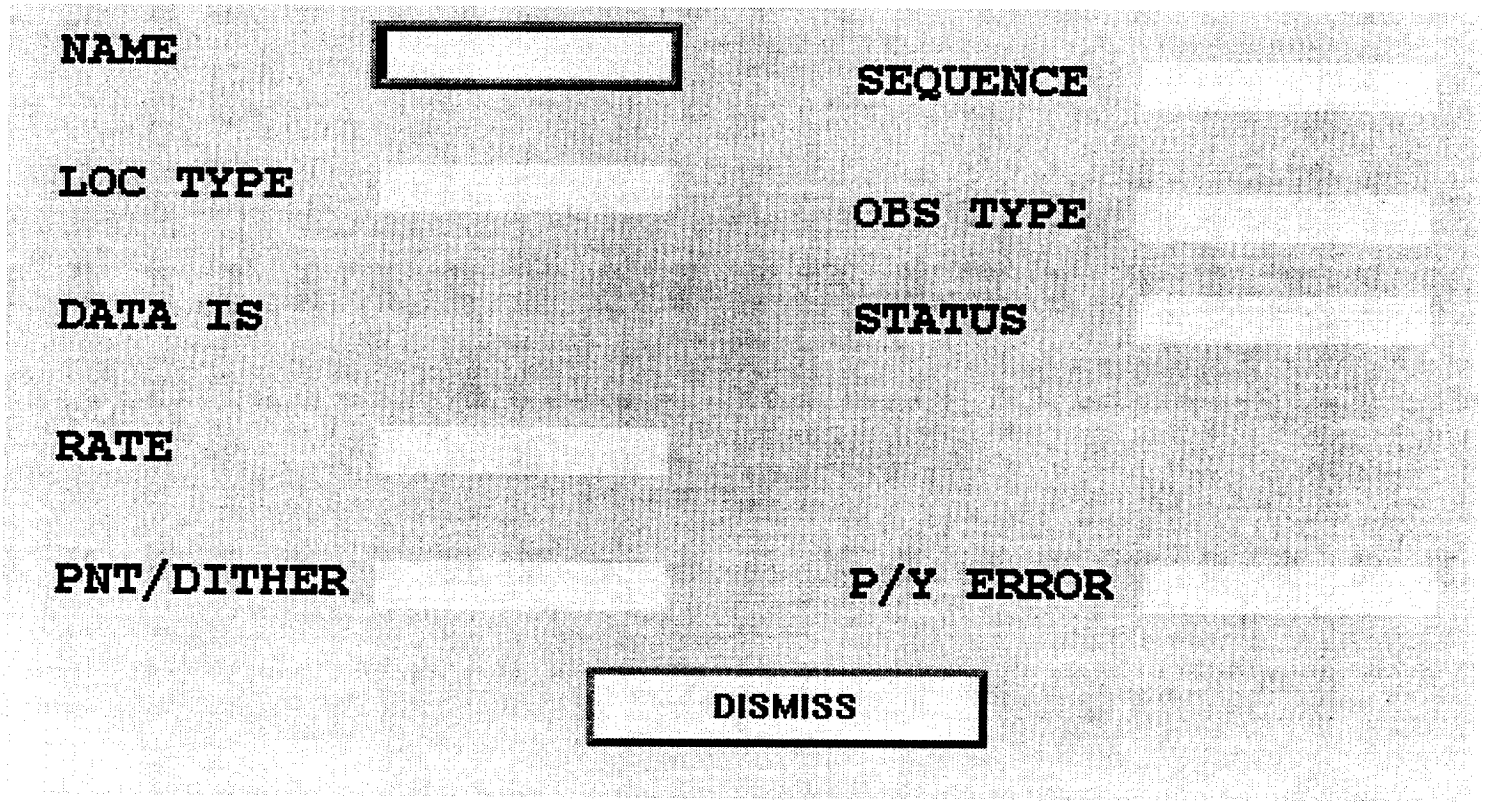
HUT - PREPARE FOR DEORBIT

PREPARE FOR DEORBIT		
USER ACTIONS	INTERFACE FEEDBACK	INTERFACE STATE
command (+28V BUS OFF)	+28V BUS OFF	
command (HEATERS OFF)	HEATERS OFF	mode = SHUTDOWN
check_vacuum_pump (1, ON, -5.00 torr, 0.15A)		
check_vacuum_pump (2, OFF, -7.13 torr, 0.00A)		

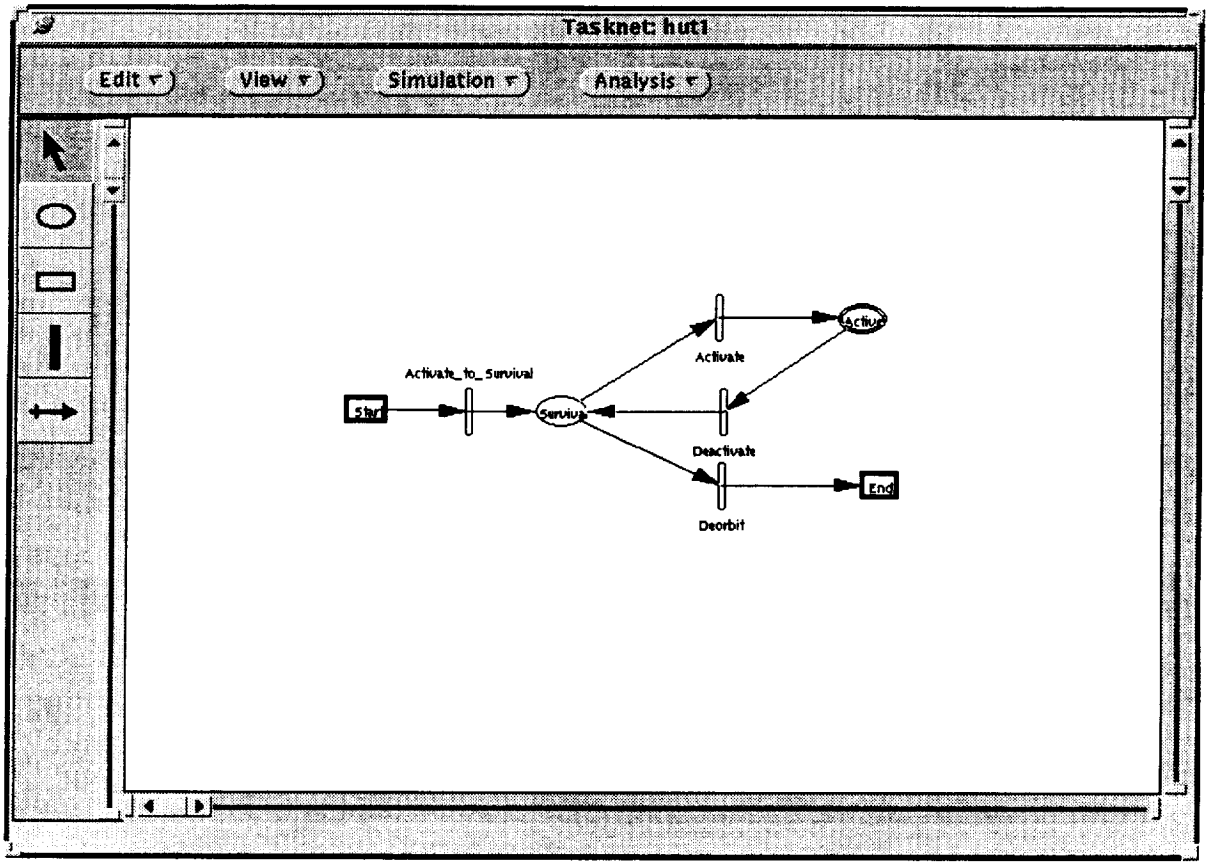
Appendix B - Graphical Interface for the Hopkins Ultraviolet Telescope


POWER
CURRENT
+28V BUS
OFF
MAIN POWER
OFF
+5V
-12V
+12V
+18V
PUMP
CURRENT
PRESSURE
ON
OFF
SPECTROMETER
OFF
Calibration Lamp (OFF)
LOAD
MODE
MASK
AMP
MIN
MAX
WIDTH
MIN
MAX
SLIT WHEEL
PHOTOM
DETECTOR
OFF
DEFAULTS
DEP
STATE
ACTIVE
DEP COMMANDS
RESET
LOAD
SETUP
BEGIN
PROCEED
PAUSE
QUIT
SHUTDOWN
OBSERVATION SEQUENCE
PREVIEW
CURRENT
SETUP
PREVIEW SEQ
PROCEDURES
TV MODE
SOURCE
DOORS
DOORS
MIRRORS
X
-Z
-Y+Z
+Y+Z
Start
Mode
HEATERS
OFF
CURRENT
A/D CONVERTER
OFF
STRUCT TEMP
ELEC TEMP
OFF
MODE
ELEC HEATER
SLAVED HEATERS
CAMERA
OFF
EXPOSURE
CAMERA HV
ZOOM
WHITE LVL
BLACK LVL
SUN BOS (Disabled)
EARTH BOS (Disabled)
SOFTWARE INT
FORCE SYNC
PARITY COUNT


DOORS
DISABLE
SMALL APERTURE DOORS
+ Y SHUTTER DOOR
- Y SHUTTER DOOR


NAME
SEQUENCE
LOC TYPE
OBS TYPE
DATA IS
STATUS
RATE
PNT/DITHER
P/Y ERROR
DISMISS

Appendix C - Simulator for the Hopkins Ultraviolet Telescope


Tasknet: hut1
Edit
View
Simulation
Analysis
Activate_to_Survival
Activate
Deactivate
Deorbit
End

Survival

Alphanumeric Id	ACT5
Tasknet	hut1
Last Modified	qqswp -- Mon Jun 13 09:01:54 1994
Time	
Mean	0.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
heatercurrent	SendMsg saturn "set heater_current [set $heater_current 9]"
heaterconvcurrent	SendMsg saturn "set heater_conv_current [set heater_conv_current 14] "
pump1current	SendMsg saturn "set pump1_current [set pump1_current 15]"

pump2current	SendMsg saturn "set pump2_current [set pump2_current 0] "
electronicstemp	SendMsg saturn "set electronics_temp [set electronics_temp 45]"
mainpowercurrent	SendMsg saturn "set main_power_current [set main_power_current 0]"
structuretemp	SendMsg saturn "set structure_temp [set structure_temp 0] "
heatermode	SendMsg saturn "set heater_mode [set heater_mode 1]"
bus28vcurrent	SendMsg saturn "set bus28v_current [set bus28v_current 1]"

Activate_to_Survival

Alphanumeric Id	EVENT4
Tasknet	hut1
Last Modified	qqswp -- Thu May 19 11:13:17 1994
Event	
Condition	$IC && ($heaters) && ($bus28v) && ($pump1) && (!$pump2)
Probability	1.0
User Defined Slots	

Activate

Alphanumeric Id	EVENT6
Tasknet	hut1
Last Modified	qqswp -- Fri May 20 17:10:05 1994
Event	
Condition	$IC && ($main_power)
Probability	1.0
User Defined Slots	
inactive	$FALSE

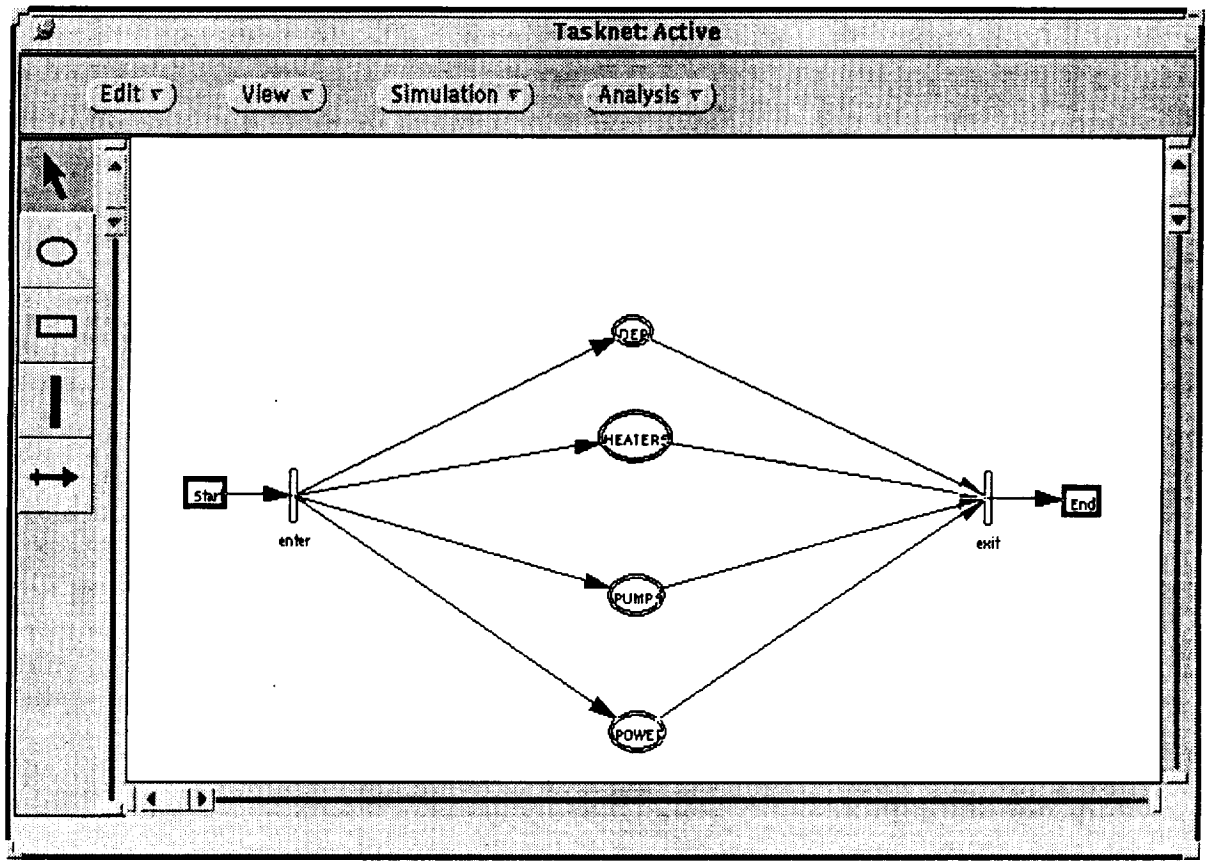
Deactivate

Alphanumeric Id	EVENT10
Tasknet	hut1
Last Modified	qqswp -- Fri May 20 17:10:26 1994
Event	
Condition	$IC && ($inactive)
Probability	1.0

User Defined Slots

Deorbit

Alphanumeric Id	EVENT24
Tasknet	hut1
Last Modified	qqswp -- Mon Jun 13 07:58:25 1994
Event	
Condition	$IC && (!$bus28v) && (!$heaters) & & ($pump1) && (!$pump2)
Probability	1.0
User Defined Slots	
heatercurrent	SendMsg saturn "set heater_current [set heater_current 0] "
heaterconvcurrent	SendMsg saturn "set heater_conv_cu rrent [set heater_conv_current 0]"
bus28vcurrent	SendMsg saturn "set bus28v_current [set bus28v_current 0] "
electronicstemp	SendMsg saturn "set electronics_te mp [set electronics_temp 0]"

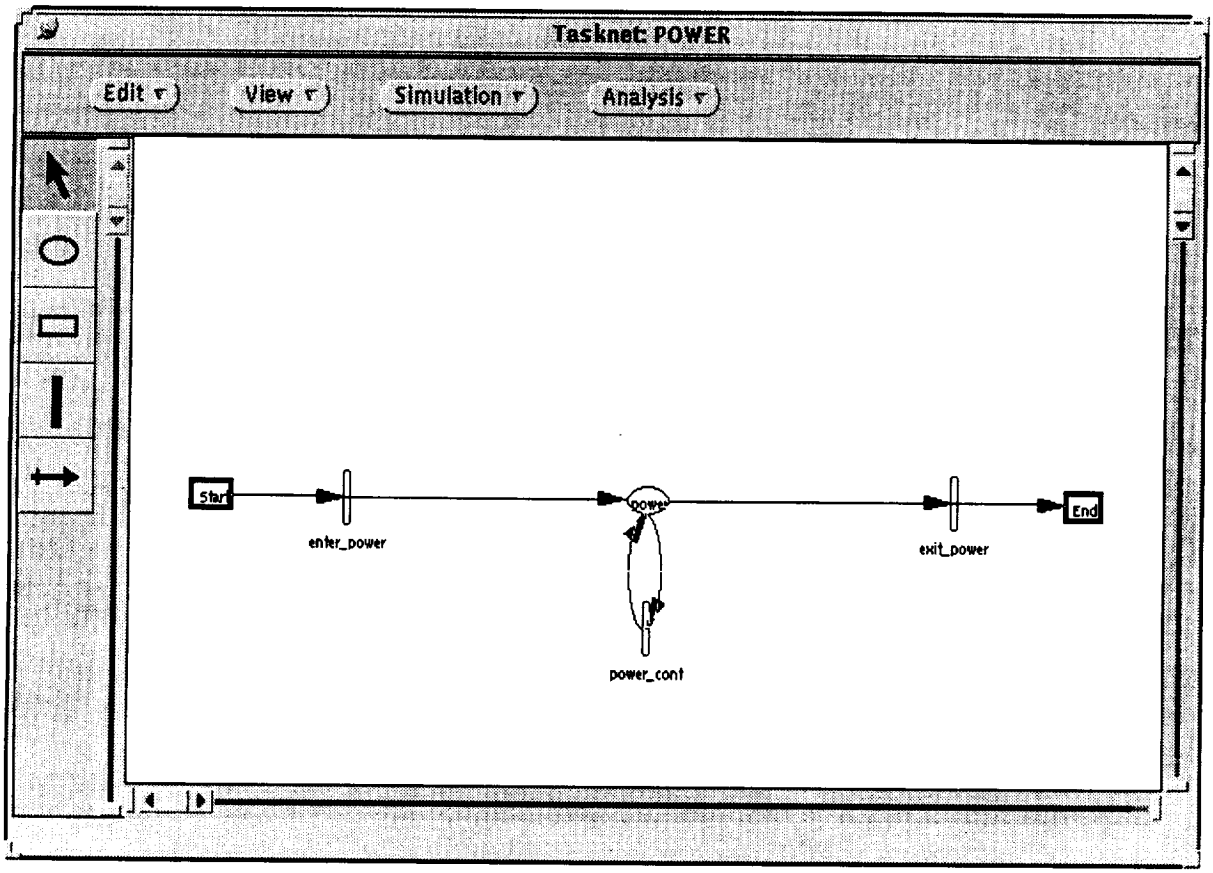

Tasknet: Active
Edit
View
Simulation
Analysis
Start
enter
HEATER
PUMP
exit
End

enter

Alphanumeric Id	EVENT106
Tasknet	Active
Last Modified	qqswp -- Tue May 24 20:37:09 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	
inactive	$FALSE

exit

Alphanumeric Id	EVENT107
Tasknet	Active
Last Modified	qqswp -- Fri May 20 17:27:15 1994
Event	
Condition	$IC && $inactive
Probability	1.0
User Defined Slots	


Tasknet: POWER
Edit
View
Simulation
Analysis
Start
enter_power
power
exit_power
End
power_cont

power

Alphanumeric Id	ACT319
Tasknet	POWER
Last Modified	qqswp -- Thu Jun 16 12:46:11 1994
Time	
Mean	3.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
mainpowercurrent	SendMsg saturn "set main_power_cur rent [set main_power_current [expr {3 + $camera_current}]]"
inactive	if {!$main_power} then {$TRUE} els e {$FALSE}
calcbuscurr	0.14 + $main_power_current + $spec _current + ($pump1_current/100) +

bus28vcurrent

```
($pump2_current/100)
SendMsg saturn "set bus28v_current
 [set bus28v_current [expr { (($ca
lc_bus_curr > 0) && ($calc_bus_cur
r < 1)) ? 1 : $calc_bus_curr } ] ]
 "
```

pos5vcurrent

```
SendMsg saturn "set pos5v_current
[set pos5v_current 5]"
```

pos12vcurrent

```
SendMsg saturn "set pos12v_current
 [ set pos12v_current 12 ]"
```

neg12vcurrent

```
SendMsg saturn "set neg12v_current
 [set neg12v_current -12]"
```

pos18vcurrent

```
SendMsg saturn "set pos18v_current
[set pos18v_current 19]"
```

enter_power

Alphanumeric Id	EVENT320
Tasknet	POWER
Last Modified	qqswp -- Mon Jun 13 22:52:09 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	

exit_power

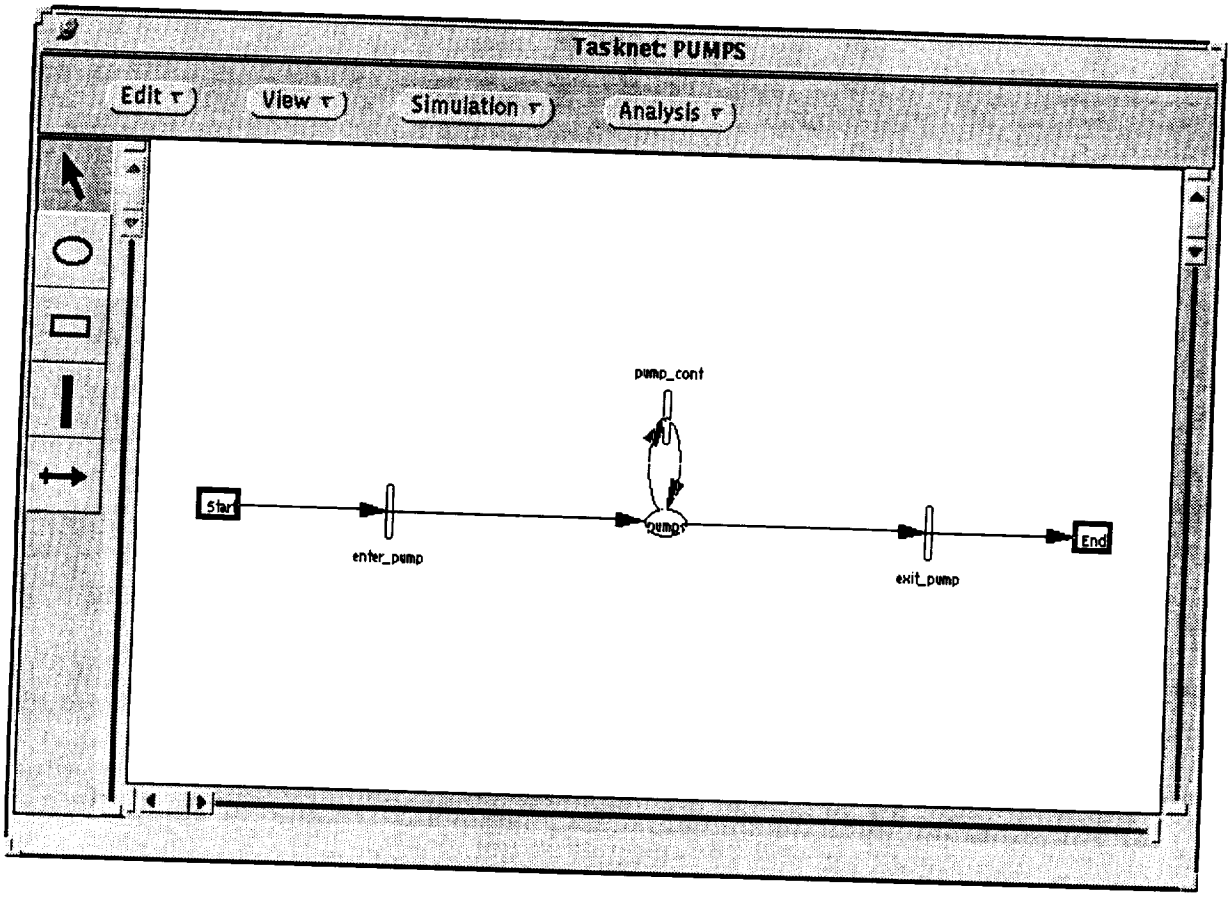
Alphanumeric Id	EVENT321
Tasknet	POWER
Last Modified	qqswp -- Wed Jun 15 11:23:32 1994
Event	
Condition	$IC && $inactive
Probability	1.0
User Defined Slots	
bus28vcurrent	SendMsg saturn "set bus28v_current [set bus28v_current [expr {$bus28v_current - $main_power_current}]] "
mainpowercurrent	SendMsg saturn "set main_power_current [set main_power_current 0]"
pos5vcurrent	SendMsg saturn "set pos5v_current [set pos5v_current 0]"
pos12vcurrent	SendMsg saturn "set pos12v_current [set pos12v_current 0]"
neg12vcurrent	SendMsg saturn "set neg12v_current

pos18vcurrent

[set neg12v_current 0]"
SendMsg saturn "set pos18v_current
[set pos18v_current 0]"

power_cont

Alphanumeric Id	EVENT322
Tasknet	POWER
Last Modified	qqswp -- Wed May 25 09:04:57 1994
Event	
Condition	$IC && !$inactive
Probability	1.0
User Defined Slots	


Tasknet: PUMPS
Edit
View
Simulation
Analysis
pump_cont
Start
enter_pump
Pump
exit_pump
End

pumps

Alphanumeric Id	ACT338
Tasknet	PUMPS
Last Modified	qqswp -- Mon Jun 13 22:54:41 1994
Time	
Mean	3.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
pump1current	SendMsg saturn "set pump1_current [set pump1_current [expr {$pump1 * 12}]] "
pump2current	SendMsg saturn "set pump2_current [set pump2_current [expr {$pump2 * 13}]] "
pump1pressure	SendMsg saturn "set pump1_pressure

enter_pump

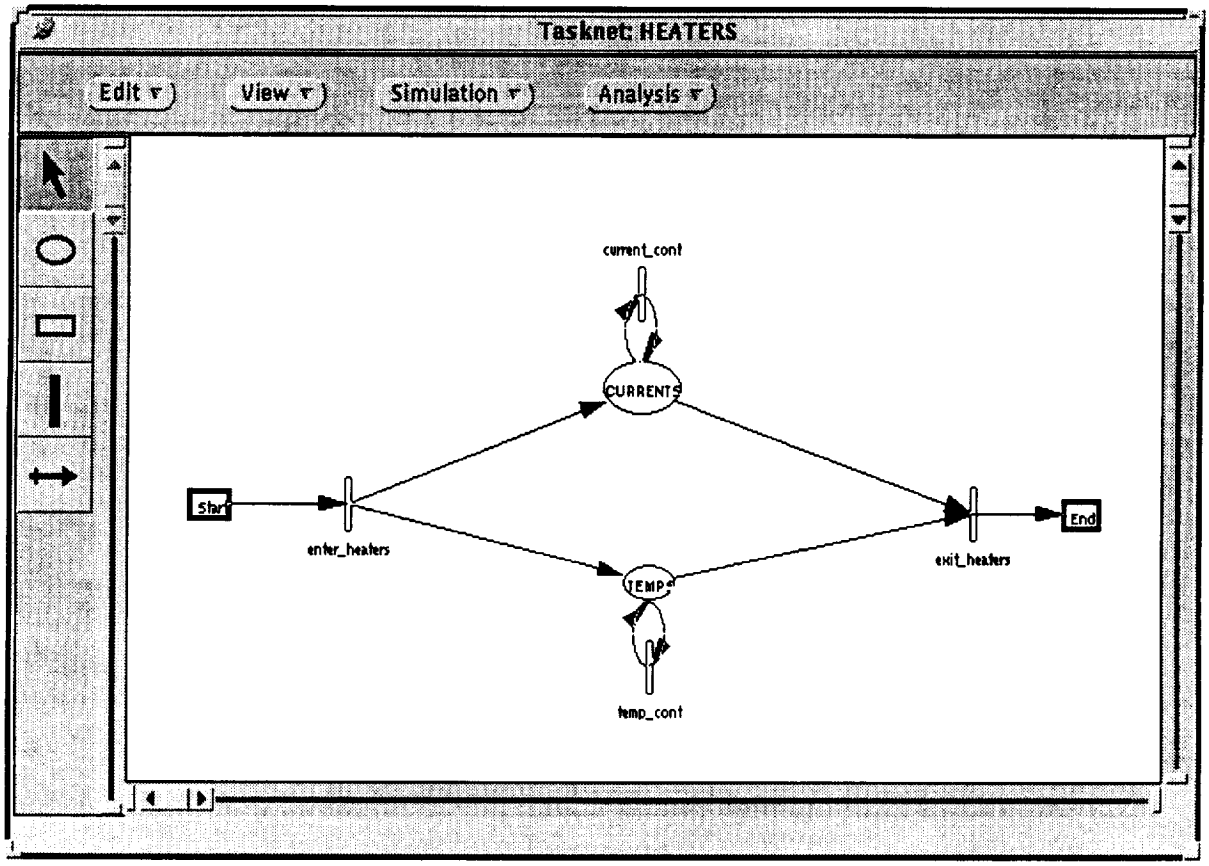
Alphanumeric Id	EVENT335
Tasknet	PUMPS
Last Modified	qqswp -- Mon Jun 13 22:51:53 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	

pump_cont

Alphanumeric Id	EVENT336
Tasknet	PUMPS
Last Modified	qqswp -- Thu Jun 2 14:58:39 1994
Event	
Condition	$IC && !$inactive
Probability	1.0
User Defined Slots	

exit_pump

Alphanumeric Id	EVENT337
Tasknet	PUMPS
Last Modified	qqswp -- Mon Jun 13 22:52:43 1994
Event	
Condition	$IC && $inactive
Probability	1.0
User Defined Slots	


Tasknet: HEATERS
Edit
View
Simulation
Analysis
current_cont
CURRENTS
Start
enter_heaters
TEMP
exit_heaters
End
temp_cont

CURRENTS

Alphanumeric Id	ACT355
Tasknet	HEATERS
Last Modified	qqswp -- Mon Jun 13 23:02:09 1994
Time	
Mean	3.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
heatercurrent	SendMsg saturn "set heater_current [set heater_current 9]"
heaterconvcurrent	SendMsg saturn "set heater_conv_current [set heater_conv_current 14] "

TEMPS

Alphanumeric Id	ACT356
Tasknet	HEATERS
Last Modified	qqswp -- Mon Jun 13 23:00:42 1994
Time	
Mean	3.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
electronicstemp	SendMsg saturn "set electronics_te mp [set electronics_temp 45] "
structuretemp	SendMsg saturn "set structure_temp [set structure_temp [expr {!$dep _active ? 0 : 10 }]] "

enter_heaters

Alphanumeric Id	EVENT351
Tasknet	HEATERS
Last Modified	qqswp -- Mon Jun 13 22:56:38 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	

exit_heaters

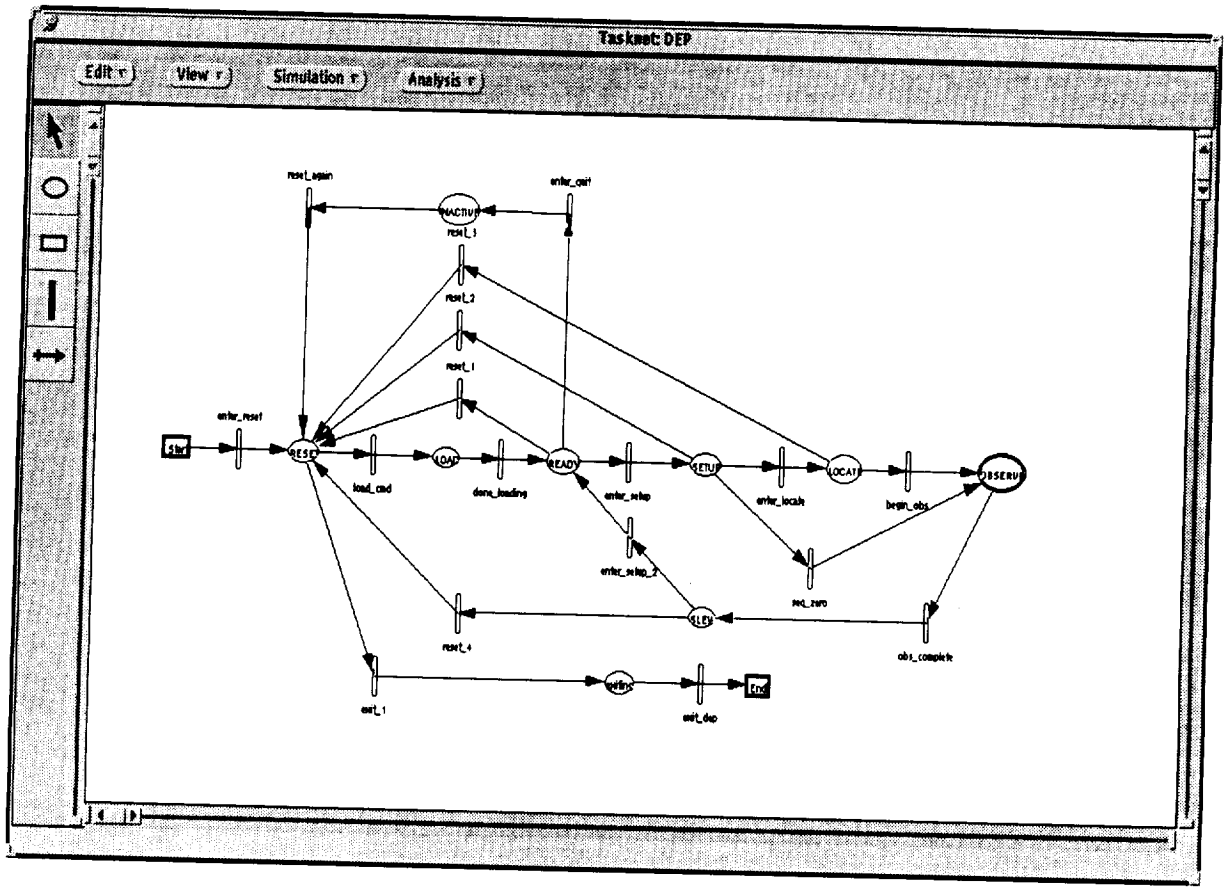
Alphanumeric Id	EVENT352
Tasknet	HEATERS
Last Modified	qqswp -- Mon Jun 13 22:59:18 1994
Event	
Condition	$IC && $inactive
Probability	1.0
User Defined Slots	

current_cont

Alphanumeric Id	EVENT353
Tasknet	HEATERS
Last Modified	qqswp -- Thu Jun 2 15:22:08 1994
Event	
Condition	$IC && !$inactive
Probability	1.0
User Defined Slots	

temp_cont

Alphanumeric Id	EVENT354
Tasknet	HEATERS
Last Modified	qqswp -- Thu Jun 2 15:22:41 1994
Event	
Condition	$IC && !$inactive
Probability	1.0
User Defined Slots	


Tasknet: DEP
Edit
View
Simulation
Analysis
reset_again
enter_rest
reset_3
reset_2
reset_1
enter_reset
load_cmd
done_loading
enter_setup
enter_locate
begin_obs
enter_setup_2
rest_zero
reset_4
obs_complete
exit_1
exit_dep

RESET

Alphanumeric Id	ACT374
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 08:34:44 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
depstate	SendMsg saturn "set dep_state [set dep_state 1]"

LOAD

Alphanumeric Id	ACT378

Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 10:27:56 1994
Time	
Mean	50.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
depactive	SendMsg saturn "set dep_active [se t dep_active 1]"
depstate	SendMsg saturn "set dep_state [set dep_state 2]"
previewtvmag	0
previewtvmode	0
previewseqname	0
previewspmode	0
previewspmask	0
previewlocatetype	0
previewobstype	0

previewslitpos	0
previewdoorpos	0
previewfilterpos	0
previewsrcmag	0
previewguidemag	0

READY

Alphanumeric Id	ACT394
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 08:48:58 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	

depstate	SendMsg saturn "set dep_state [set dep_state 3]"

INACTIVE

Alphanumeric Id	ACT400
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 08:50:05 1994
Time	
Mean	2.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
depactive	SendMsg saturn "set dep_active [se t dep_active 0]"
depstate	SendMsg saturn "set dep_state [set

dep_state 0]"

exiting

Alphanumeric Id	ACT408
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 09:33:48 1994
Time	
Mean	2.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	

SETUP

Alphanumeric Id	ACT422

Tasknet	DEP
Last Modified	qqswp -- Thu Jun 16 22:07:32 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
currentdoorpos	$preview_door_pos
currentfilerpos	$preview_filter_pos
currentobstype	$preview_obs_type
currentseqname	$preview_seq_name
currentslitpos	$preview_slit_pos
currentspmask	$preview_sp_mask
currentspmode	$preview_sp_mode
currentsrcmag	$preview_src_mag
currenttvmag	$preview_tv_mag
currenttvmode	$preview_tv_mode

currentseq	$preview_seq
currentguidemag	$preview_guide_mag
currentlocatetype	$preview_locate_type
currentdatais	$preview_data_is
currentrate	$preview_rate
currentpntdither	$preview_pnt_dither
currentpyerror	$preview_p_y_error
currentobsstatus	$preview_obs_status
depstate	SendMsg saturn "set dep_state [set dep_state 4]"

LOCATE

Alphanumeric Id	ACT427
Tasknet	DEP
Last Modified	qqswp -- Thu Jun 16 22:08:17 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0

Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
depstate	SendMsg saturn "set dep_state [set dep_state 5]"

SLEW

Alphanumeric Id	ACT436
Tasknet	DEP
Last Modified	qqswp -- Thu Jun 16 22:09:52 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	

enter_reset

Alphanumeric Id	EVENT373
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 10:28:28 1994
Event	
Condition	$IC && ($main_power) && (!$inactive)
Probability	1.0
User Defined Slots	
depactive	SendMsg saturn "set dep_active [set dep_active 0]"
depcmd	1

exit_dep

Alphanumeric Id	EVENT375
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 09:34:05 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	
depactive	SendMsg saturn "set dep_active [set dep_active 0]"
depstate	SendMsg saturn "set dep_state [set dep_state 0]"

load_cmd

Alphanumeric Id EVENT376
Tasknet DEP
Last Modified qqswp -- Wed Jun 15 10:28:31 1994
Event
Condition $IC && ($dep_cmd == 2) && (!$inactive)
Probability 1.0
User Defined Slots

---

done_loading

Alphanumeric Id EVENT377
Tasknet DEP
Last Modified qqswp -- Wed Jun 15 11:07:22 1994
Event
Condition $IC
Probability 1.0
User Defined Slots

---

reset_again

Alphanumeric Id EVENT390
Tasknet DEP
Last Modified qqswp -- Tue Jun 14 09:38:32 1994
Event
Condition $IC && (($dep_cmd == 1) || ($inactive))
Probability 1.0
User Defined Slots

enter_quit

Alphanumeric Id	EVENT398
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 11:01:39 1994
Event	
Condition	$IC && ($dep_cmd == 8) && (!$inactive)
Probability	1.0
User Defined Slots	

exit_1

Alphanumeric Id	EVENT406
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 09:28:13 1994
Event	
Condition	$IC && $inactive
Probability	1.0
User Defined Slots	

reset_1

Alphanumeric Id	EVENT415
Tasknet	DEP
Last Modified	qqswp -- Tue Jun 14 09:39:26 1994
Event	
Condition	$IC && (($dep_cmd == 1) || ($inactive))
Probability	1.0

User Defined Slots

---

enter_setup

Alphanumeric Id	EVENT423
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 11:01:54 1994
Event	
Condition	$IC && ($dep_cmd == 3) && (!$inactive)
Probability	1.0
User Defined Slots	

---

enter_locate

Alphanumeric Id	EVENT426
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 09:56:11 1994
Event	
Condition	$IC && ($current_seq != 0) && (!$inactive)
Probability	1.0
User Defined Slots	

---

reset_2

Alphanumeric Id	EVENT429
Tasknet	DEP
Last Modified	qqswp -- Thu Jun 16 12:48:09 1994

Event
Condition $IC && (($dep_cmd == 1) || ($inact ive))
Probability 1.0
User Defined Slots

---

seq_zero

Alphanumeric Id EVENT434
Tasknet DEP
Last Modified qqswp -- Thu Jun 16 22:48:50 1994
Event
Condition $IC && ($preview_seq == 0) && (!$i nactive)
Probability 1.0
User Defined Slots
depcmd 4

---

reset_3

Alphanumeric Id EVENT439
Tasknet DEP
Last Modified qqswp -- Wed Jun 15 10:00:23 1994
Event
Condition $IC && (($dep_cmd == 1) || ($inact ive))
Probability 1.0
User Defined Slots

---

begin_obs

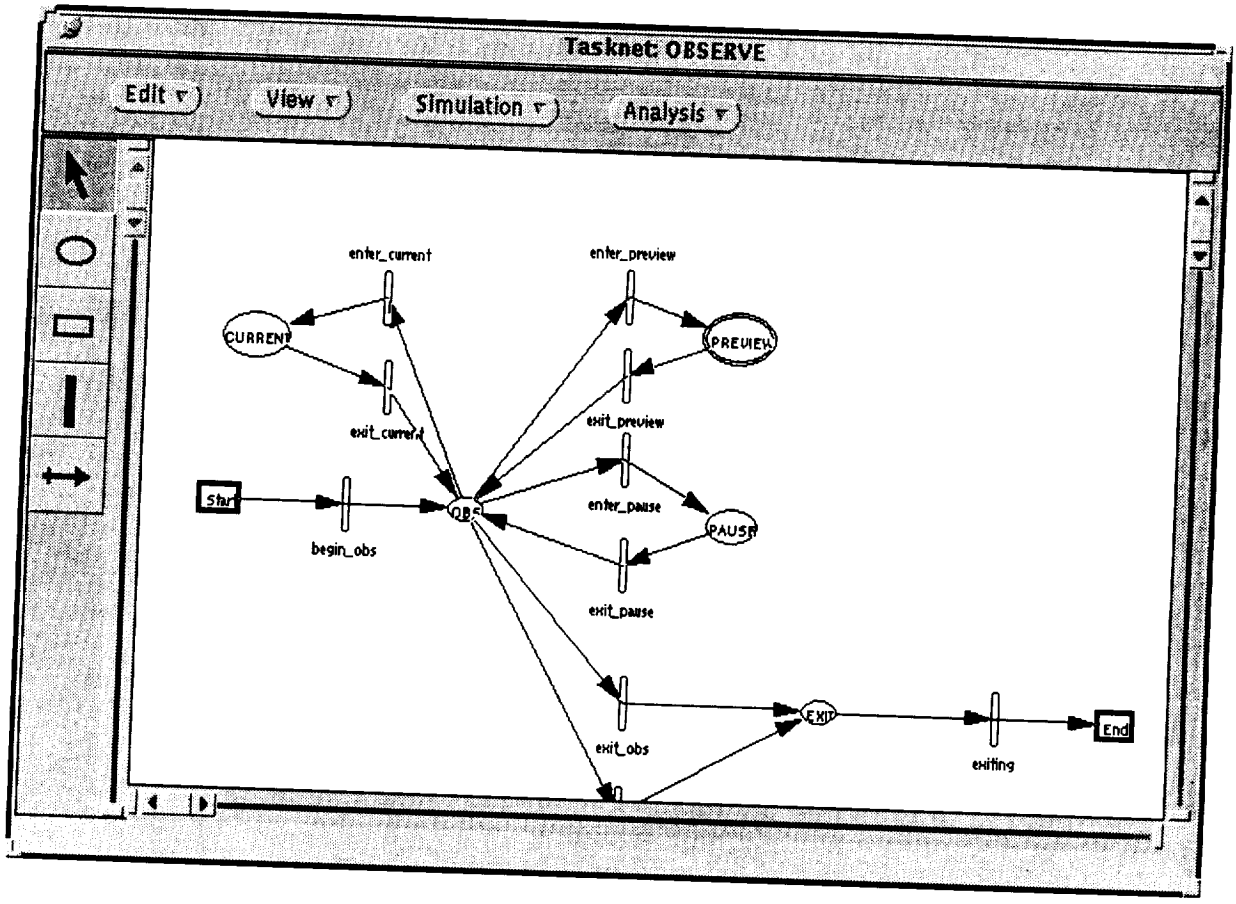
Alphanumeric Id	EVENT442
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 11:02:51 1994
Event	
Condition	$IC && ($dep_cmd == 4) && (!$inactive)
Probability	1.0
User Defined Slots	

reset_4

Alphanumeric Id	EVENT460
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 10:12:07 1994
Event	
Condition	$IC && (($dep_cmd == 1) || ($inactive))
Probability	1.0
User Defined Slots	

obs_complete

Alphanumeric Id	EVENT463
Tasknet	DEP
Last Modified	qqswp -- Wed Jun 15 10:12:59 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	


Tasknet: OBSERVE
Edit
View
Simulation
Analysis
enter_current
enter_preview
CURRENT
PREVIEW
exit_current
exit_preview
Start
OBS
enter_pause
PAUSE
begin_obs
exit_pause
exit_obs
EXIT
End
exiting

OBS

Alphanumeric Id	ACT471
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 22:09:01 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
depstate	SendMsg saturn "set dep_state [set dep_state 6]"

PAUSE

Alphanumeric Id	ACT474

Tasknet	OBSERVE
Last Modified Time	qqswp -- Wed Jun 15 10:18:46 1994
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
pausetime	$TIME

EXIT

Alphanumeric Id	ACT484
Tasknet	OBSERVE
Last Modified Time	qqswp -- Wed Jun 15 10:30:09 1994
Mean	4.0
Devtation	0.0

Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	
Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	

CURRENT

Alphanumeric Id	ACT510
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 13:47:09 1994
Time	
Mean	4.0
Devtation	0.0
Distribution	NORMAL
Unit	MINUTES
Crew	
Crew	Crew1
UModel Workload	

Demand Value	0.0
Multiple Resource Model	
Window Display	0.0
Displays & Controls	0.0
Auditory Processing	0.0
Verbal Processing	0.0
Spatial Processing	0.0
Continuous Motor	0.0
Discrete Motor	0.0
Communication	
Message	
User Defined Slots	
currentdoorpos	SendMsg saturn "set current_door_pos $current_door_pos"
currentfilerpos	SendMsg saturn "set current_filter_pos $current_filter_pos"
currentguidemag	SendMsg saturn "set current_guide_mag $current_guide_mag"
currentlocatetype	SendMsg saturn "set current_locate_type $current_locate_type"
currentobstype	SendMsg saturn "set current_obs_type $current_obs_type"
currentseq	SendMsg saturn "set current_seq $current_seq"
currentseqname	SendMsg saturn "set current_seq_name $current_seq_name"
currentslitpos	SendMsg saturn "set current_slit_pos $current_slit_pos"
currentspmask	SendMsg saturn "set current_sp_mask $current_sp_mask"
currentspmode	SendMsg saturn "set current_sp_mode $current_sp_mode"

begin_obs

Alphanumeric Id	EVENT469
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 14:25:57 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	
obstimeexpire	SendMsg saturn "set obs_time_expir e [set obs_time_expire [expr { $TI ME + 200.0 }]] "

exiting

Alphanumeric Id	EVENT470
Tasknet	OBSERVE
Last Modified	qqswp -- Wed Jun 15 10:30:17 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	

enter_pause

Alphanumeric Id	EVENT472
Tasknet	OBSERVE
Last Modified	qqswp -- Wed Jun 15 11:03:15 1994
Event	
Condition	$IC && ($dep_cmd == 6) && (!$inact

	ive)
Probability	1.0
User Defined Slots	

exit_pause

Alphanumeric Id	EVENT473
Tasknet	OBSERVE
Last Modified	qqswp -- Wed Jun 15 11:03:34 1994
Event	
Condition	$IC && (($dep_cmd == 5) || ($dep_c md == 7) \|\| ($dep_cmd == 8) || ($i nactive))
Probability	1.0
User Defined Slots	
obstimeexpire	$obs_time_expire + ($TIME - $pause _time)

exit_obs

Alphanumeric Id	EVENT482
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 13:06:54 1994
Event	
Condition	$IC && (($dep_cmd == 7) || ($dep_c md == 8) \|\| ($obs_time_expire > $T IME) \|\| ($inactive))
Probability	1.0
User Defined Slots	

seq_zero_exit

Alphanumeric Id	EVENT498
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 10:04:53 1994
Event	
Condition	$IC && (($dep_cmd == 1) || ($dep_c md == 3)) && ($preview_seq != 0)
Probability	1.0
User Defined Slots	

exit_preview

Alphanumeric Id	EVENT501
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 10:09:55 1994
Event	
Condition	$IC
Probability	1.0
User Defined Slots	

enter_preview

Alphanumeric Id	EVENT502
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 10:09:12 1994
Event	
Condition	$IC && ($dep_cmd == 9) && (!$inact ive)
Probability	1.0

User Defined Slots

---

exit_current

Alphanumeric Id	EVENT508
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 13:11:44 1994
Event	
Condition	$IC && (($dep_cmd == -11) || ($inactive))
Probability	1.0
User Defined Slots	
currentwindow	SendMsg saturn "cmd dw hut1_seq"

---

enter_current

Alphanumeric Id	EVENT509
Tasknet	OBSERVE
Last Modified	qqswp -- Thu Jun 16 13:49:51 1994
Event	
Condition	$IC && ($dep_cmd == 11) && (!$inactive)
Probability	1.0
User Defined Slots	
currentwindow	SendMsg saturn "cmd aw hut1_seq"