N90-20661

APPLICATIONS OF GRAPHICS TO SUPPORT A TESTBED FOR AUTONOMOUS SPACE VEHICLE OPERATIONS

K. R. Schmeckpeper, J. P. Aldridge, S. Benson, S. Horner, A. Kullman, T. Mulder, W. Parrott, D. Roman, G. Watts
McDonnell Douglas Space Systems Company
Houston, Texas 77062

Daniel C. Bochsler
LinCom Corporation - Houston Operations
Houston, Texas 77058

ABSTRACT

We describe our experience using graphics tools and utilities while building an application, AUTOPS, that uses a graphical Macintosh (TM)-like interface for the input and display of data, and animation graphics to enhance the presentation of results of autonomous space vehicle operations simulations. AUTOPS is a test bed for evaluating decisions for intelligent control systems for autonomous vehicles. Decisions made by an intelligent control system, e.g., a revised mission plan, might be displayed to the user in textual format or he can witness the effects of those decisions via "out of the window" graphics animations. Although a textual description conveys essentials, a graphics animation conveys the replanning results in a more convincing way. Similarily, iconic and menu-driven screen interfaces provide the user with more meaningful options and displays. We present our experiences with the SunView and TAE Plus graphics tools that we used for interface design, and the Johnson Space Center Interactive Graphics Laboratory animation graphics tools that we used for generating our "out of the window" graphics.

INTRODUCTION

For several years, much effort has gone into the development and application of enabling and enhancing technologies for support of space operations. Many new technologies and methods, such as artificial intelligence and expert systems, have been applied to flight design software, user interface problems, ground and flight crew training, ground based mission control operations, robotic operations, flight systems management, etc. [1] The AUTOPS (autonomous operations) test bed integrates many of these technologies into a single framework to develop effective operations management, an element of mission success that is equal in importance to reliable hardware and software. [2]

AUTOPS is an evolving tool that has thus far been developed to the point of a feasibility demonstration that makes considerable use of animated graphics and screen interaction graphics. The animations are used for demonstrating proximity operations autonomy in operation planning, mission monitoring, and fault management. Screen graphics additionally assist in demonstrating vehicle monitoring and health maintenance expert systems and rendezvous planning activities. Because these items form uniquely informative means to convey system behavior to an analyst, they form an important feature of AUTOPS.

Although the importance of good graphics is unquestionable, their development has previously represented a significant commitment of time and effort. The availability of graphics tools has significantly changed this level of commitment. In this paper, we discuss our recent experience with using some of these tools.

AUTOPS CONCEPT

Figure 1 illustrates the architecture of AUTOPS. The test bed consists of a collection of objects dedicated to specific activities: a test bed controller and vehicles that contain subobjects such as intelligent vehicle control systems, orbital and hardware simulations, and data management capabilities. The graphic capabilities are isolated from the computational capabilities in the graphics and operator interface objects controlled by the test bed controller. This architecture permits the reuse of code developed by others or the use of tools developed by others to produce the desired interfaces. Intelligent control is accomplished through cooperating expert systems that perform mission direction, mission monitoring, operations planning, and system health monitoring and fault recovery.

Currently, "vehicles" use software simulation as the means for providing orbital motion parameters and consistent sensor response to the orbital environment and vehicle subsystem operation. It is our intent to provide the capability to integrate hardware into the test bed to provide some of these data. For example, if it were desirable to test the ability of a vision sensor for use in close proximity operations, a television picture could be generated using the animated graphics and fed back to the vision hardware for the appropriate vehicle. A more immediate example is to use a fuzzy logic hardware chip to provide engine firings in place of the fuzzy logic controller software used in the feasibility demonstration.

Finally, other features of AUTOPS include the execution of the operation in real time and integration of currently available programs, especially simulation software. Real-time operation means here that the simulation computations will occur often enough to reflect actual behavior of an autonomous space vehicle and that time spent by expert systems in arriving at a decision for action will be taken into account.

SCREEN INTERFACES

Our feasibility demonstration required three screen interface designs: a main Operator Interface (OI), an interface to the electrical power system expert system (EPSYS), and an interface to the propulsion system expert system (PROPSYS). These interfaces were constructed over a period of time in which we were significantly increasing our graphics tool capability. The first to be built, the EPSYS interface, was created with SunView which is system software for our SUN network. The OI and PROPSYS interfaces were created with TAE Plus software obtained from Goddard Space Flight Center.

EPSYS is a prototype diagnostic expert system for monitoring the electrical power system of an autonomous shuttle-like space vehicle. Its function is to detect and explain anomalies and generate plans to recover from system faults. EPSYS supports a window- and menu-based user interface. The user-interface is composed of a base window that is subdivided among a group of graphic and text subwindows (Figure 2). Each graphic subwindow represents a control panel for a physical subsystem. The control panels are composed of parameter headings and a matrix of associated status lights and trend symbols. A command button and hierarchical menu system were designed to allow the user to easily communicate with the expert system. The final component of the interface is a scrollable text subwindow. The function of this window is to organize and display the textual representation of the high-level interactions and conclusions within the expert system.

Our choices for the development of the EPSYS interface were SunView and X. We chose SunView largely because we had access to the source code of a SunView-based interface which supported many of the same functional requirements that EPSYS possessed. Also, SunView is well-documented. At this time, our in-house version of X had several bugs and lacked complete and accurate documentation. In addition, the documentation we possessed supplied few examples. Also, our version of TAE Plus, an X code generator, was an early release and did not support many of the functions we needed to implement. The EPSYS interface was completed in three weeks by two programmers, including learning the SunView system.

The second interface we built was for the OI for inputting orbital parameters and showing calculational results. We elected to use TAE Plus for this task. TAE Plus allows the user to build a graphics interface with a Macintosh (TM)-like feel by using a graphics workbench tool with a mouse. It adds a layer of programming over standard X code, such that the developer is required to have little, if any, X programming knowledge. Once the developer has the interface screen or panels designed, the workbench tool can generate code that implements it. Currently, the workbench will generate code in the C and Ada languages with Fortran and C++ generators under development.

Approximately one week was spent in learning how to use TAE Plus and how to integrate its generated code into an application. The original OI design was completed and implemented in four days by one programmer. An additional week was spent in editing the interface by "tweaking" the placement of items in a panel. Figure 3 presents the prototype OI master control panel and vehicle states output panels. The graphics workspace is the only panel that requires direct X programming.

Figure 4 shows an overlayed Initialization panel where the user can select one of ten rendezvous cases and either accept default data or modify any of the orbital elements. This panel required the most time to complete, as all work was performed on a SUN 3/50. TAE Plus was designed to run on a SUN 3/60. A twenty-four character limitation on display text length required that the titles on the rendezvous case selection buttons be created in halves and dragged to their locations on the panel.

The Propulsion Expert System (PROPSYS) is another prototype for a fault management system. PROPSYS will be a part of a distributed network of cooperating expert systems forming the System Monitor for an autonomous vehicle. It is a rule-based system written in CLIPS. Its user interface was developed using TAE Plus and X. The user interface is composed of a main control panel which is used to generate subsystem faults (Figure 5). The subsystem chosen brings up other panels with menus to enter parameters necessary for fault generation. After fault generation is complete, display panels that are appropriate for monitoring the subsystem during fault analysis and recovery appear (Figure 6). A standard X window displays text provided by the expert system during its operation. The text provides information on high-level interactions and conclusions made by the expert system.

The PROPSYS interface was completed in about four weeks by two programmers. This included learning TAE Plus and integrating its generated code with the application code. Access to existing TAE Plus code provided invaluable assistance and reduced our development time.

ANIMATION GRAPHICS EXPERIENCE

The integration of the AUTOPS Testbed Prototype with an existing graphics package was a simple, straight-forward procedure. In order to connect the prototype to the graphics, the AUTOPS Testbed Prototype software was loaded onto a Sun workstation located in the NASA Interactive Graphics Lab. This Sun contained Raster Technologies' graphics boards to provide a graphics engine and was connected to a high-resolution color monitor.

The modification of code in order that AUTOPS could be integrated with the graphics was also a minor procedure that consisted of customizing three routines and a data file. The three routines and the data file were copied from the graphics package into the AUTOPS simulation code. They were then modified to fit our requirements. This consisted of picking the vehicle models that we were using, in this case, models of the Shuttle Orbiter and the Orbital Maneuvering Vehicle, choosing information such as eye-point position, background models for the stars and the Earth, lighting, and size of the vehicle models. After the modifications were completed, the resulting code was compiled and linked into the simulation code. The Prototype was then executed in the same way that it was before the graphics was integrated into it. The procedure for integrating the AUTOPS prototype with the graphics required two programmers for two days.

Figure 7 shows one of the runs made with this system. The asterisks show positions of the orbiter at constant time intervals. Speed is thus indicated by the separation of successive indicators. This example illustrates the triggering of a replan by an expert system planner in response to an anomaly, in this case, a loss of general purpose computer redundancy. Flight rules specify that the vehicle shall back straight out to a 200 foot range in this event. The graphics emphasize and record this behavior.

CONCLUSION

We have found that graphics tools provide a practical solution to quickly building excellent interfaces, that the tools are rapidly improving, and that the time for changes is growing sufficiently short that timely modifications of the interfaces to accomodate user preferences is now practical. Also, animated graphics can be easily adapted to enhance computational results without extensive modification of an application that does not support such capacity.

REFERENCES

1) Wang, Lui and Bochsler, Daniel, "Space Shuttle Onboard Navigation Console Expert/Trainer System," p. 11, NASA CONFERENCE PUBLICATION 2491, First Annual Workshop on Space Operations Automation and Robotics (SOAR 87), Houston, TX, August 5-7, 1987.

2) Beck, Harold,"Application of Technology to Space Flight Operations," Workshop on Space Launch Management and Operations,

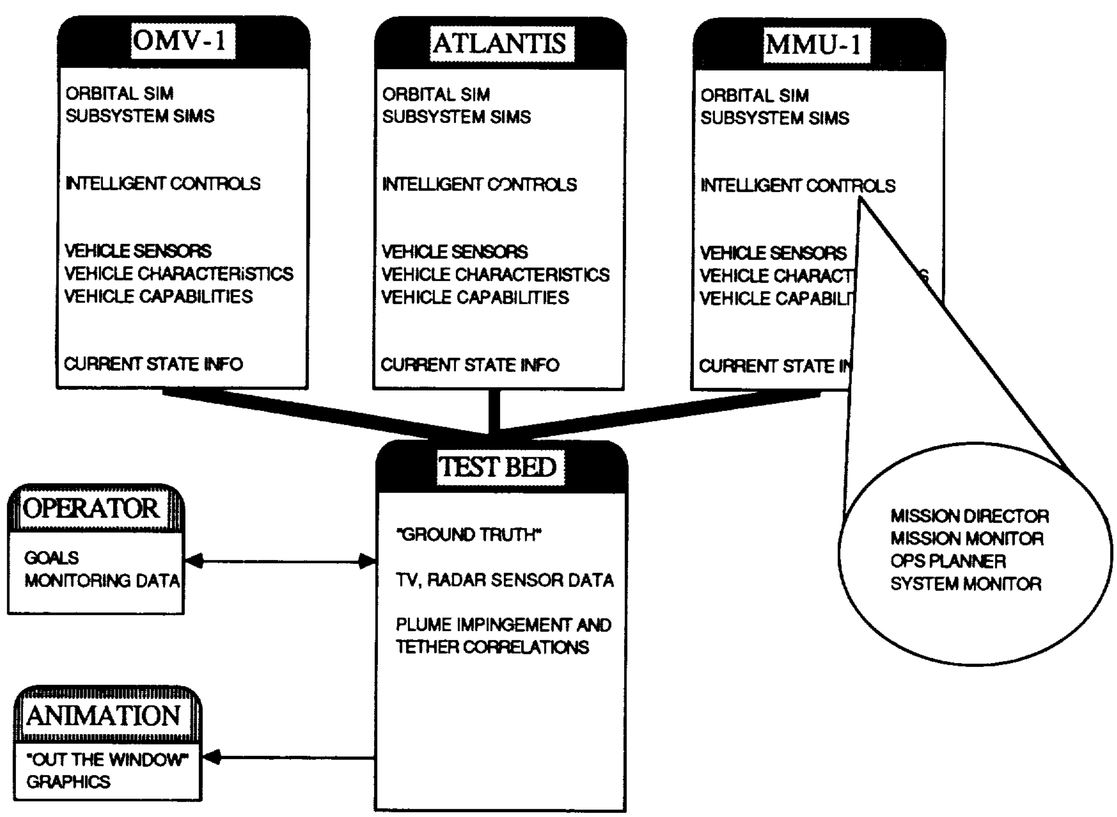

Figure 1. AUTOPS architecture

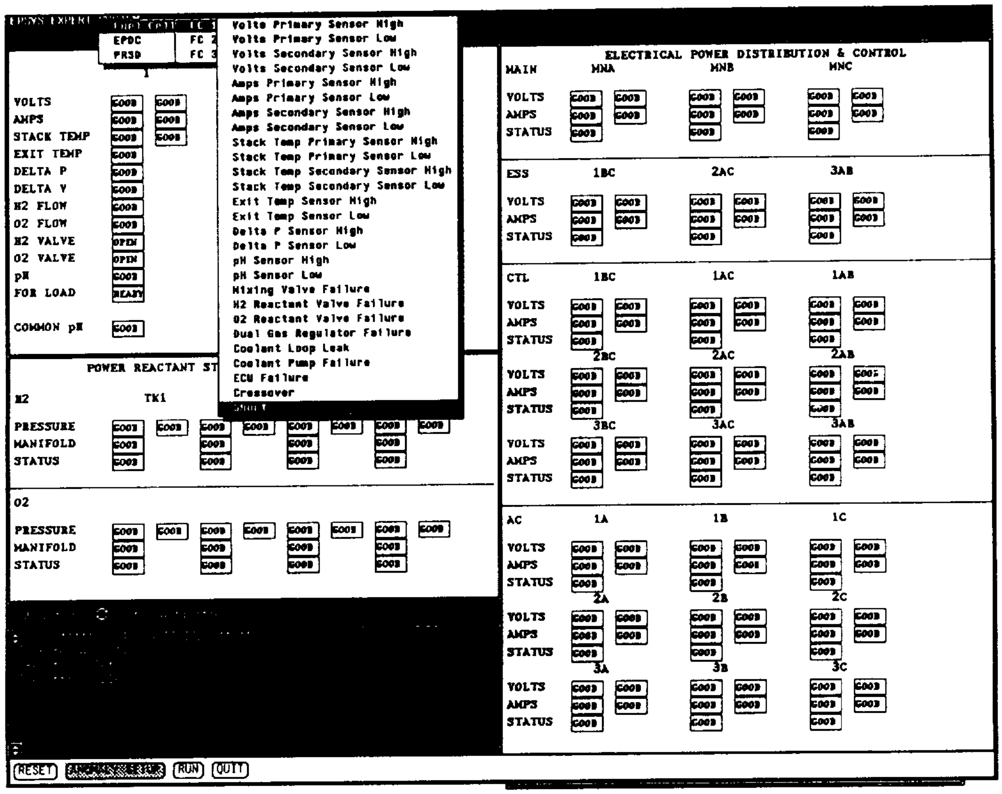

Figure 2 EPSYS User Interface

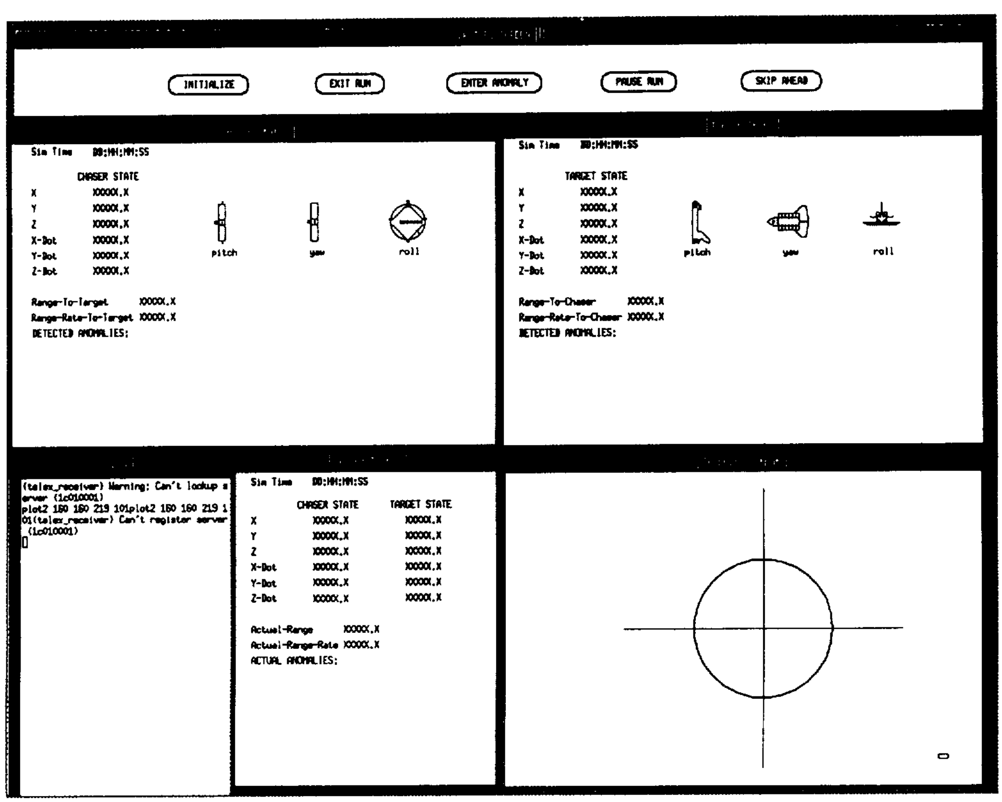

Figure 3 AUTOPS Operator Interface

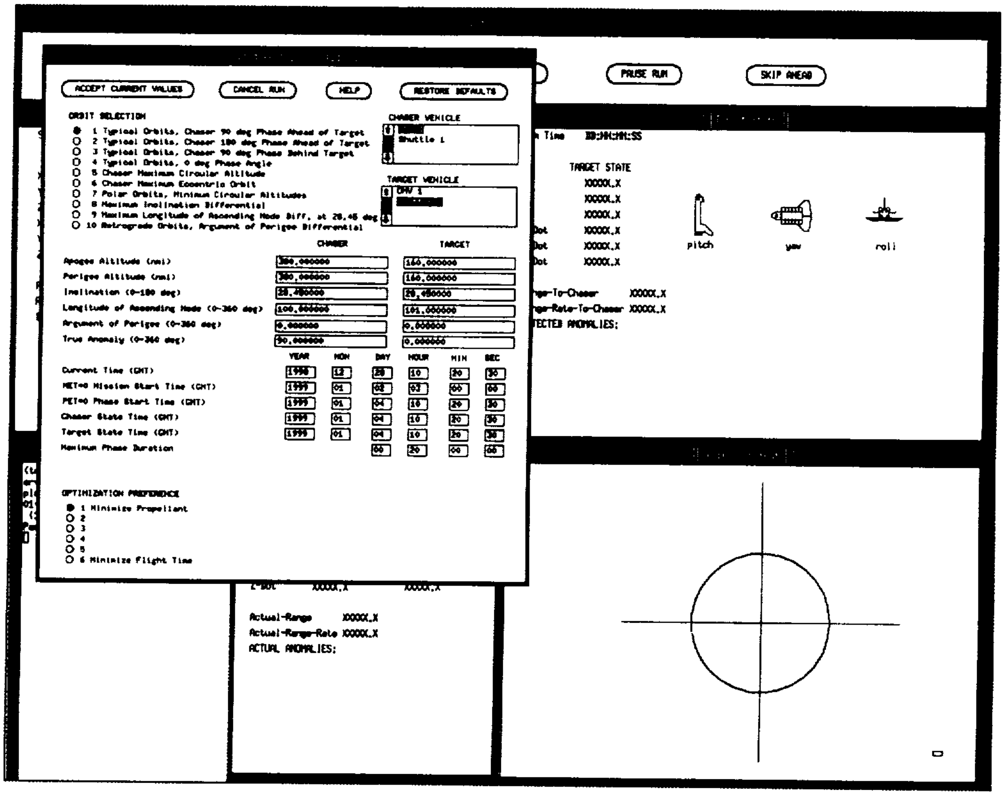

Figure 4 AUTOPS Initialization Interface

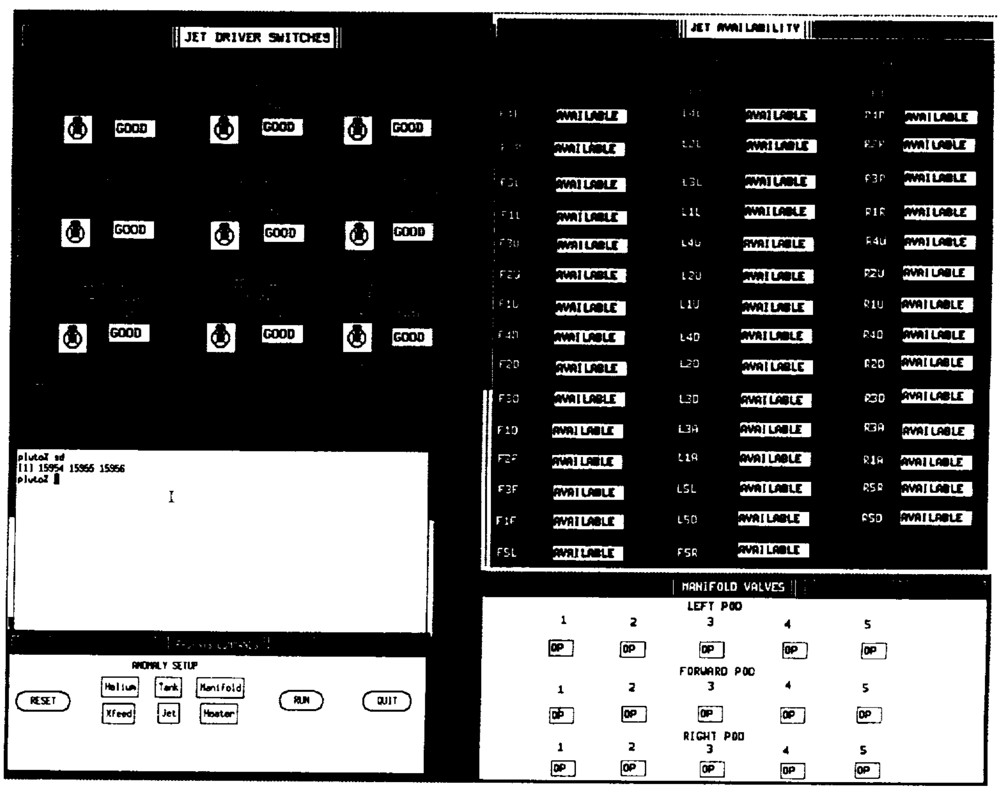

Figure 5 PROPSYS Main Control Panel

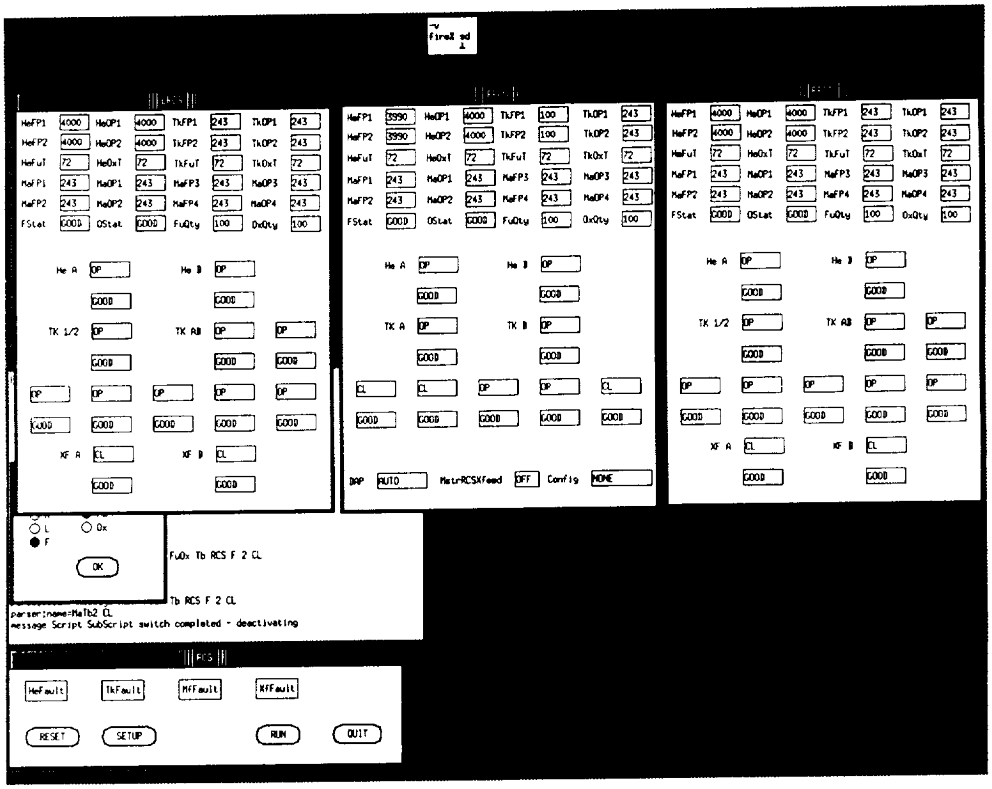

Figure 6 PROPSYS Fault Monitoring Panel

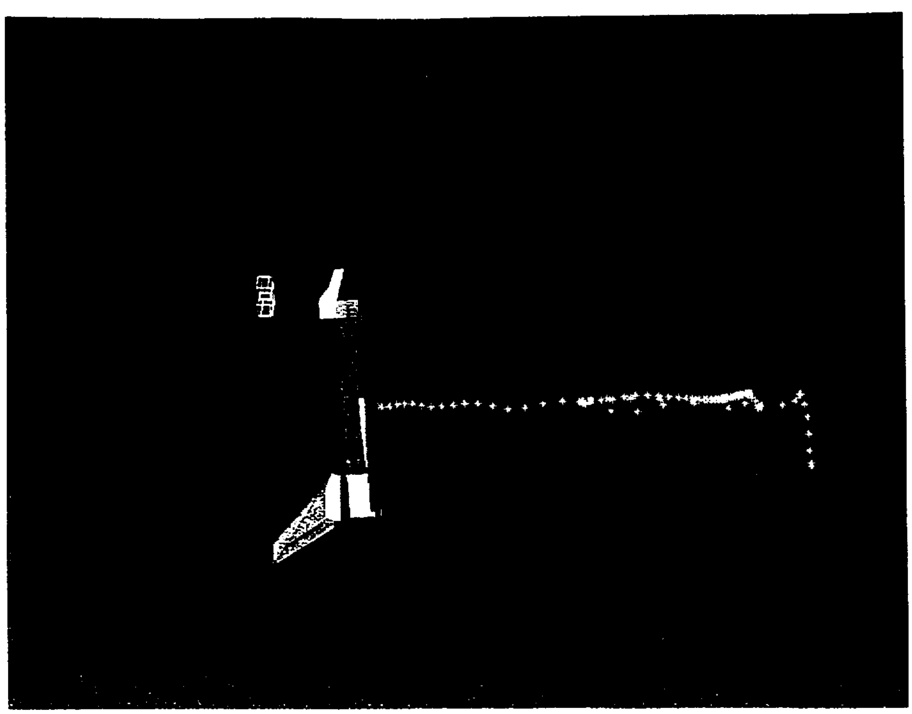

Figure 7 AUTOPS Animation Graphics Output

DESTINATION MARS

Mike Remus
Morton Thiokol, Inc.

(Paper not provided by publication date.)