OPEN CONTROL/DISPLAY SYSTEM FOR A TELEROBOTICS WORK STATION

Saul Keslowitz, PhD
Grumman Corporation
Bethpage, NY 11714

ABSTRACT

Grumman Space System's Controls and Displays (C&D) Laboratory has developed a working Advanced Space Cockpit that integrates advanced control and display devices into a state-of-the-art multimicroprocessor hardware configuration, using "window" graphics and running under an object-oriented, multitasking real-time operating system environment. This Open Control/Display System supports the idea that the operator should be able to interactively monitor, select, control, and display information about many payloads aboard the Space Station using unique sets of I/O devices with a single, software-reconfigurable workstation. This is done while maintaining system consistency, yet the system is completely open to accept new additions and advances in hardware and software.

As we see it, operators aboard the Space Station will be required to monitor and maintain all of the Station's subsystems from a single, small, shared work area. We have designed displays that provide consistency between operations. The information displayed to the operator is easily accessible, understandable, and useful. There is no need to train an operator on each specific display since each display page follows a consistent format. Typical displays include graphic aids such as docking reticles, force/moment and motor torque plots, joint angle parameter displays, and graphic handbook information such as schematics. We have demonstrated the usefulness of infrared touch-sensitive color graphic switches and of voice recognition systems, in addition to conventional dedicated input devices.

The hardware/software architecture is configured so that when new technologies become available, it will be easy to modify the system to handle the new requirements. The system was made hardware-independent by applying the concept of "software layering." This entailed writing generic software modules that can communicate with hardware-specific software "device drivers" in a consistent manner. The concept of "object-oriented programming" makes a large complicated software task manageable by breaking it down into tasks or "objects" that communicate by passing uniformly defined messages to each other. Applying these modern approaches makes the resulting software manageable, easily modifiable, and transportable. We have made our software "data driven," so that changes in program requirements only require a modification of the data file rather than a program rewrite, thus avoiding a tedious reverification and validation effort.

The Advanced Space Cockpit, linked to Grumman's Hybrid Computing Facility and Large Amplitude Space Simulator (LASS), has been used to test the Open Control/Display System via full-scale simulations of the following tasks: telerobotic truss assembly, RCS and thermal bus servicing, CMG changeout, RMS constrained motion and space constructible radiator assembly, HPA coordinated control, and OMV docking and tumbling satellite retrieval. The proposed man-machine interface standard discussed below has evolved through many iterations of the tasks, and is based on feedback from NASA and Air Force personnel who performed those tasks in the LASS.

1 - INTRODUCTION

(1 Space Systems' Advanced Space Cockpit Laboratory is used il for investigating advanced state-of-the-art technologies as they ɔ the man-machine interface. With this laboratory, it is possibemonstrate a variety of ways that humans can communicate more ely and naturally with the computer. A principal function of thisl station laboratory is to demonstrate methods of monitoring and cɪg simulated Space Station payloads, remote vehicles, and manipɪystems. The workstation has been developed, as should a typica Station workstation. It has been designed to interface with v any input/output peripheral device, while imposing a minimum in software or hardware redesign. To accommodate the rapidly cl technology in the areas of hardware, software, and artificial ince, it should be possible to easily upgrade the workstation with tɔdern approaches as proven research results become available. ɪg the limitations of dedicated workstations for each payload (possiling weight and requiring more space in the Space Station) makes It is possible to have a generic workstation with a windowingility so that the operator can control and display anything from aɪstation. This also makes sense in terms of reliability.

2 - MAN-MACHINE INTERFACE

Cuirements should be imposed on future workstations so that it is ead more natural for the operator to communicate with all aspects ohisticated computer-based system. Some guidelines in the followiɪs should be considered:

- devices
- ɪy formats
- ɔnality among workstations
- rotocol
- [color
- [shape and texture
- ology trends
- e of automation.

ave addressed these issues and, based on several iterations of changwe have developed our own requirements based on feedback from h factors tests, Air Force, and NASA personnel suggestions[5].

2.1 STATION FEATURES

ad of having a separate workstation to monitor and control Spaceion payloads, which adds weight and uses more space, the AdvarSpace Cockpit was designed so that almost all C&D functions can bntrolled from any console and that interaction with the user is consis between workstations. Using infrared touchable color graphic switcach workstation can be reconfigured by simply redefining the touchches displayed on the workstation. At any given time only the pertinswitches are displayed. The software is set up so that the displaompts the user and guides him through a diagnostic scenario when at-of-tolerance condition or failure occurs. The master alarm switchhe upper right hand corner turns red, blinks, and an audible tone ls until this switch is touched to signify acknowledgement of the pn.

al functions available to our workstations are clearing and callink the graphics in a zone on the screen. This is useful if the graphbstruct the image of the live target. Other functions on the bottothe screen are touch switch control of brightness and contrast of thV image, calling up docking reticles, parameter displays, and real-lots in designated zones.

2.2 H INPUT

rincipal input device is an infrared touch bezel. It allows a fast,, and natural way for the user to interact with a computer. The reasons for selecting the touch bezel are as follows[6]:

special skills required, such as typing

u touch what you see

No coordinated hand movement is necessary such as with a trackball or joystick

All options are available on the screen

st response

- Faster than entering commands on a keyboard or other input devices
- Can be programmed for immediate user feedback (aural or visual)

- Only valid options displayed
  - Computer guides user step by step
  - Reduce operator search time
- Touch targets can be located at different points on display
  - Can touch desired element in schematic (electrical, thermal, etc)
- Touch system displays can be interesting, illustrated and visually pleasing as well as functional
- Input errors are significantly reduced, since only valid selections are on screen
- User training time reduced as system becomes more complex
  - Not necessary to learn a computer language or key sequences to memorize

- It does not require desk space in work area
  - Large number of options can be made available by redefining touch points.

2.3 VOICE INPUT

An attractive input device is a voice recognition system. This allows hands-free advantage along with little operation training required. However, voice recognition systems presently suffer from limited recognition success along with limitations such as speaker dependence and finite number of words. As a result, voice control should not be used to control critical functions. We have successfully used it for voice control of cameras. However, statistical and expert system enhancements using syntactic and semantic considerations in increasing recognition accuracy will be studied in the future.

2.4 DISPLAY FORMATS

Each cockpit display has a consistent screen format developed by Grumman so that the operator knowns exactly where to look, no matter

at which ution he is working. The displays are partitioned (Fig. 1) such thtouch switches on the lower portion of the display control displaytions, the left switches control mission functions, the upper portsplays status information, and below that is a message area. The right corner of the display is reserved for caution and warning infon, and the right side is for real-time plots and parameter displ The center of the screen displays live video camera images withability to select graphic aids (such docking reticles, force displayoverlay the live video. Figure 2 is a picture of an RMS displayen live video is not required, the center and left side of the displayused for windows.

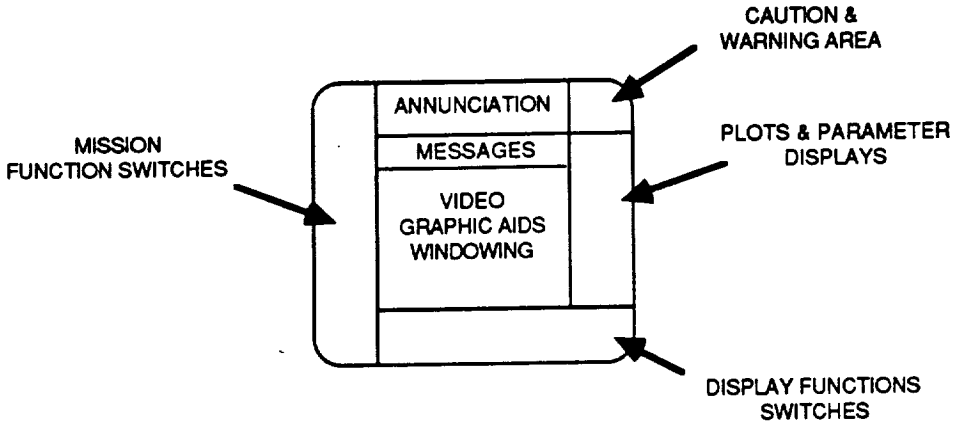
2.5 WINDOW

To monithe status of various components and perform tasks on board the SpStation, we have developed a windowing system in which the operator orms an orderly database search to retrieve any information he wa The windowing system also allows him to perform almost any tadesired by interacting with the touch sensitive graphic switches in thndow.

The use windowing has proven to be a powerful display capability in termf improving the man-machine interface. Multiple windows displayedn a CRT are basically equivalent to multiple "software CRTs". That i instead of having many bulky CRTs displaying various types of infornion, one CRT can display the same information with multiple window each corresponding to a CRT. In addition, a given window of displ information can be "opened" or "closed", interactively. Typically, with nultiple windows on a CRT, one window can display real-time informaion such as plots of a selected pump's pressure and temperature, anher can display thermal control system schematics, and another EVA status, all simultaneously and interactively. Figure 3 shows a typical example of multiple windows displayed simultaneously.

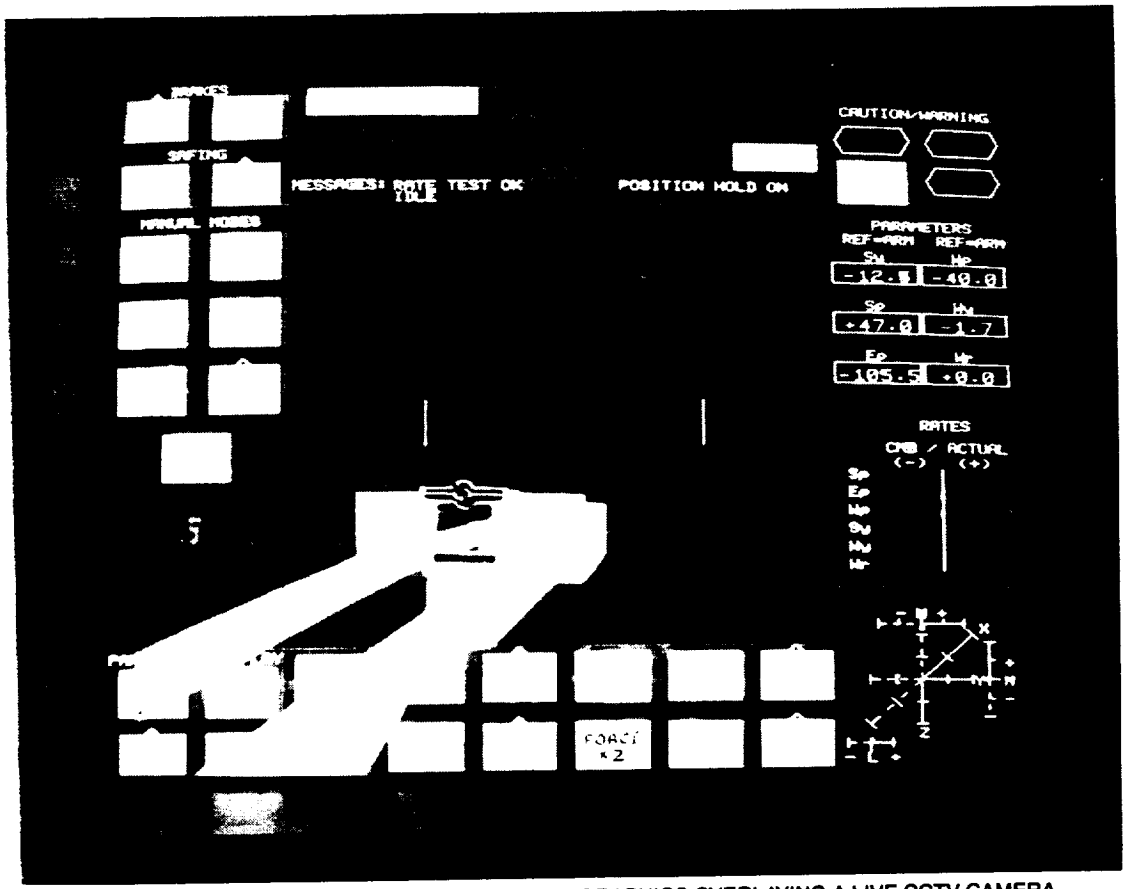
2.6 TOUCH SWITCH STATES

Since "touch switch" switches are different from conventional switches, we have developed consistent representations of switch states



R87-3538-006G

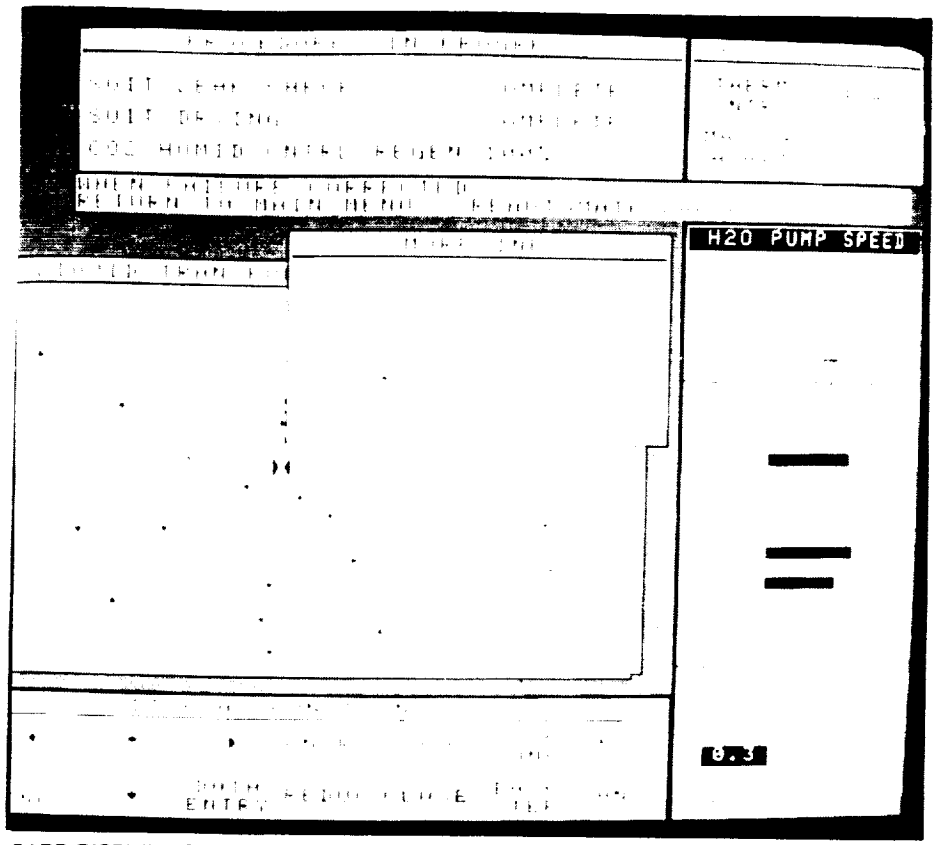
Fig. 1 Basic Display Format



NOTE: STANDARD DISPLAY FORMAT SHOWING THE GRAPHICS OVERLAYING A LIVE CCTV CAMERA IMAGE FROM THE CAMERA MOUNTED ON THE LASS DURING A SPACE CONSTRUCTIBLE RADIATOR SIMULATION. WHEN THE RADIATOR MAKES CONTACT WITH THE MANIFOLD WITH THE HELP OF THE DOCKING RETICLE, THE OPERATOR PRESSES FORCE X 2 AND THE FORCE/MOMENT DISPLAY IN THE LOWER RIGHT CORNER IS MAGNIFIED AND DISPLAYED IN THE CENTER OF THE SCREEN, OVERLAYING CAMERA IMAGE OF THE CONTACT AREA.

R87-3538-008G

Fig. 2 RMS Display



IDARD DISPLAY FORMAT SIMULTANEOUSLY SHOWS PROCEDURES IN PROGRESS STA-
ROLLABLE WINDOW, A MESSAGE IN ITS DESIGNATED AREA, AN OUT OF TOLERANCE
D SPECIFIED BY A REAL TIME H_2O PUMP SPEED PLOT, A MORE INFO WINDOW, A RED
JMP IN THE LIQUID TRANSPORT CIRCUIT SCHEMATIC AND HIGHLIGHTED EVA IN THE
ARNING AREA.

Fig. 3 Interactive Windowing Display

by)lor, texture, and shape. A normal switch available for selapproximately a 1.25 in. x 0.75 in. light blue filled rectangle witction written or abbreviated inside it. A brick face unfilled swesents a function that is unavailable at present. When a ligwitch is touched, it turns yellow and an unfilled arch is plaop of it. This represents an intermediate state, signifying thaD computer is waiting for further operator action or a confirmaıl that the device, mode, or function selected was activated. For, when "ACCEL" mode is selected on an OMV display, the swis yellow, and the C&D computer causes a message to be tra into space to the OMV computer and then waits for an actignal. When the signal is received, the switch turns green anc green arch is drawn on top of the switch, signifying the actate. The arches are used to distinguish between the states of hes if color is not available.

A short audible tone occurs each time the screen is touched, indicating to the operator that his touch registered. The possibility exists that a wrong switch was selected. Therefore, in order to prevent false activation of some switches, "ENTER" and "CANCEL" switches are available. Usually, these switches are used in conjunction with the mission function switches. For example, in the RMS display, if "SINGLE" is touched it turns yellow as do "ENTER" and "CANCEL", signifying that a choice should be made.

3 - HARDWARE/SOFTWARE OBJECTIVES

The hardware/software architecture of the workstation was originally configured with the following objectives:

- Must be powerful and general enough to accommodate all potential real-time requirements that it may have to satisfy with respect to controlling and monitoring payloads considering the human interface point of view
- Drive a number of color graphics monitors and plasma displays, perform camera control functions, manage real-time communications and color displays (such as docking, rendezvous, berth curves, plots), and accept voice, touch bezel, switches, and hand controller inputs
- Use state-of-the-art hardware and software, and use modern concepts
- Easily expandable, so as to not become obsolete
- Should use the "open system" concept and not be tied to one vendor
- Demonstrate a highly reliable system
- Hardware and operating system software should be characteristic of the flyable system to minimize redesign.

The workstation system architecture put together in this laboratory was chosen with these goals in mind. Our laboratory is built around a powerful multimicroprocessor configuration which uses a real-time multitasking operating system. This flexible architecture, which is representative of most modern advanced workstations, allows tasks to run in

real time and with each microprocessor performing its particular function simultaneously. This greatly increases the performance from a single processor system.

4 - OPEN SYSTEM CONCEPT

With rapidly advancing technology over many generations of Space Station existence, and in the face of increasing and changing requirements, it becomes necessary to design an initial workstation that can be upgraded, expanded, or reconfigured quickly and easily, in order to prevent obsolescence. Therefore, when developing a workstation, consideration should be given to choosing an open system architecture which is easily expandable, with hardware/software products available from many manufacturers.

The C&D laboratory configuration consists of three Intel 80286-based multibus microcomputers running under the iRMX-86 real-time operating system. The 80286 microprocessor was chosen because it was the most powerful available at the time. A multibus system was selected because it is an IEEE standard and over 2000 board level products from over 200 manufacturers support it. As a result, we are not locked into one vendor, our system will not become obsolete, and is easily expandable. Some multibus cards perform functions such as D/A, A/D conversion, color graphics, and serial communication.

5 - HARDWARE DESCRIPTION

Figure 4 shows a block diagram of the Advanced Space Cockpit and its associated computer control room (C&D laboratory) driving it. The C&D laboratory consists of three Intel 80286-based multibus computers, each with 30 megabyte hard drives, a 1 megabyte floppy disk drive with at least 512K of Random Access Memory (RAM), and a terminal. Each computer runs under the iRMX-86 real-time multitasking operating system.

Messages are passed between computers over an RS-232 link, which will be upgraded to Intel's Bitbus. The computers are populated with

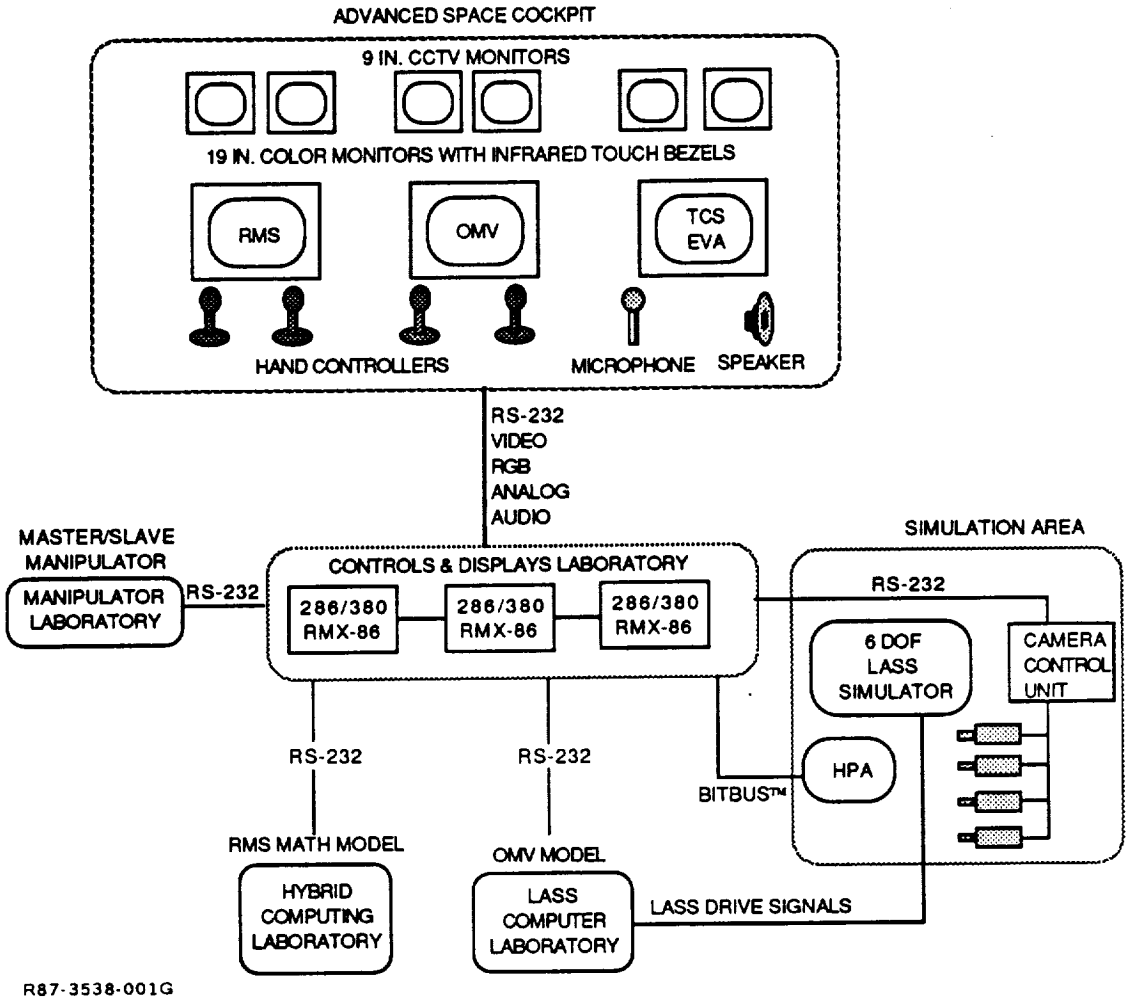


Fig. 4 Advanced Space Cockpit Block Diagram

multibus-compatible RS-232 cards, high and medium resolution Matrox color graphics cards, voice recognition/synthesis cards, discrete I/O, analog/digital converter cards, and Bitbus cards. The medium resolution graphics systems are capable of overlaying color graphics and text over a real-time CCTV camera image. This is how we overlay a docking reticle over a live camera image of a target during an OMV docking simulation. In addition, some of the cards have their own microprocessor on board, allowing concurrent processing within a microcomputer itself.

The main input device is a Carroll infrared touch bezel mounted on all our 19 in. color monitors. Graphic touch switches are used for input selection capability. Three workstations are presently active in the cockpit, however it can accommodate at least eight workstations. Figure 5 shows two of the workstations.

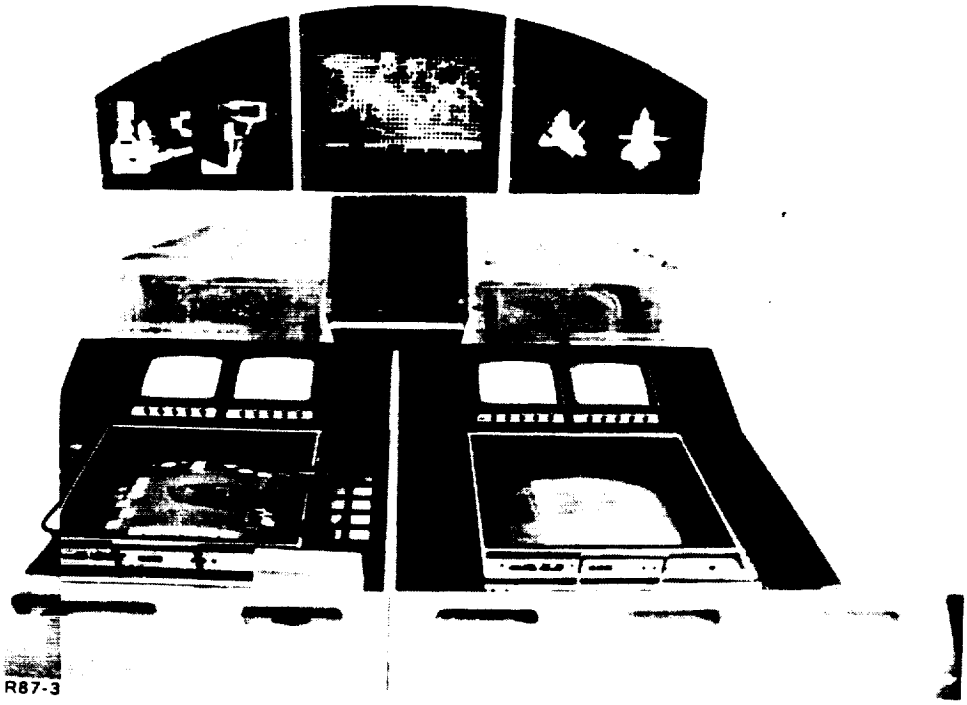


Fig. 5 Mockup of Two Bays of Prototype Workstation

neric LED dot matrix switches and plasma displays with touchare also in the cockpit. Three and six degree-of-freedom hand ers are available for OMV and RMS simulations. Headsets with nes are available for communications with the control room and oratories involved with the simulations, in addition to suppoice control of the cameras and displays using a Votan voice recog stem.

dback is achieved by using the SpeechPlus' Prose 2000 voice s system or human computer compressed speech available from th system. Tones are used for caution, warning, and failed condit

6 - SOFTWARE DESIGN PHILOSOPHY

Ooach is to develop an easily reconfigurable workstation that has a software configuration built around the iRMX-86 operating system s the following principle of good design.

"A system should be built with a minimum set of unchangeable parts; those parts should be as general as possible; and all parts of the system should be held in a uniform framework."[1]

Any part of the system that is not easy to change, is not sufficiently general, or works differently from the others will require additional effort and will impede development. Also, the software models used for this "rapid prototyping" configuration should be based on mechanisms that are familiar, easily understood, and have desirable characteristics such as simplicity, speedy development, extendibility, and reusability.

As the field of software engineering develops, various architectural concepts are emerging for developing portable reusable and maintainable software. These include the "virtual machine"[2] concept which protects the software from hardware changes and "information hiding" throughout the layering of software functions (Fig. 6). Also, "Anthropomorphic Programming is a proven technique for building systems that work using a multitask structuring and message passing scheme to simplify the analysis of complex systems".[3] Our approach keeps these modern concepts in mind.

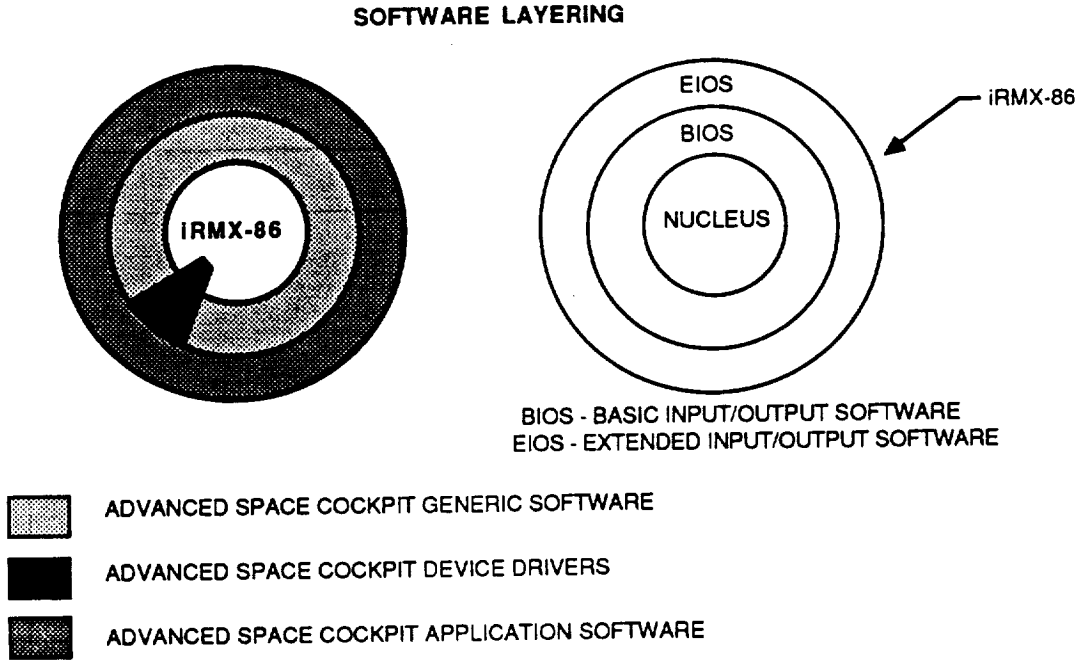
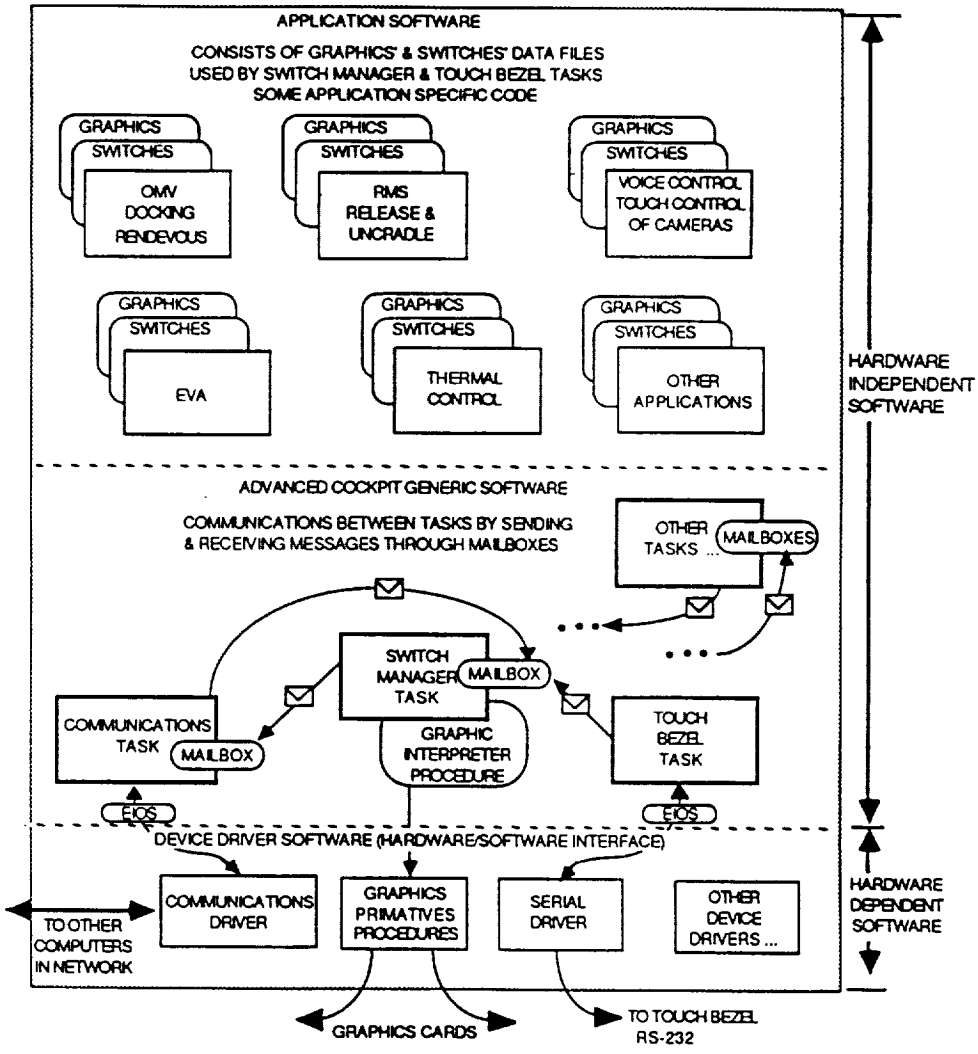


Fig. 6 Outer Layers Are Built On Inner Layers

Figure 7 shows a typical software architecture that exists on each microcomputer. Based on our design of a generic workstation of this type, we have defined the requirements of each iRMX-86 task in addition to a specific message format that each module expects to see and produce. As a result of this general architecture, additional modules can be easily added as required with little or no impact on the other modules. They can be written independently by separate programmers by simply defining the task's functions and the messages it wants to receive and send. Later, modules can be optimized internally, if nec-



NOTE: ADVANCED SPACE COCKPIT GENERIC SOFTWARE IS DATA DRIVEN, HARDWARE INDEPENDENT AND RUNS IN A MULTITASKING ENVIRONMENT. WITH THIS ARCHITECTURE THE SOFTWARE/HARDWARE IS EASILY RECONFIGURABLE AND EXPANDABLE.

R87-3538-004G

Fig. 7 Typical Software Architecture On Each Microcomputer In Network

essary, as long as the external specifications are unchanged. Note that this general specification can be applied to any workstation that uses a multitasking operating system.

7 - iRMX-86 MULTITASKING OPERATING SYSTEM

The IRMX-86 operating system[4] used in our workstation has capabilities representative of any real-time operating system. The multitasking capability allows the programmer to factor a problem into simple processes or tasks. It makes available the usual assortment of object types to the software developer. It allows the developer to create/delete tasks, mailboxes, semaphores, and segments. Intertask communication is performed by sending and receiving messages to and from a given task's mailbox. Although it appears that all the tasks are running concurrently, in reality, with only one host 80286 microprocessor in a computer, only one task can be running at a time. If a task is not running then its state is put on either the ready, asleep, or suspended queue (Fig. 8) until the present task stops running, and then the operating system starts running the next available task on the ready queue.

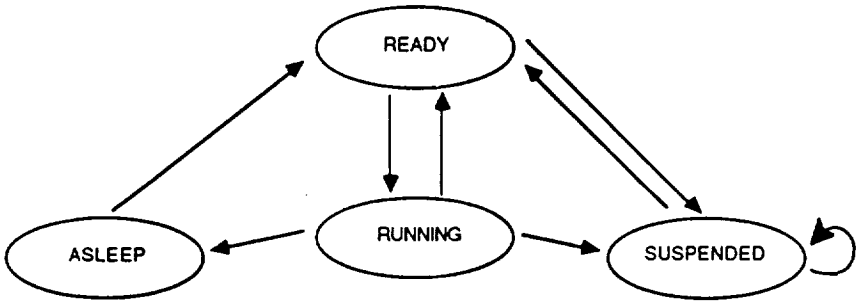


Fig. 8 Task State Transition Diagram

In Fig. 6, it is seen that iRMX-86 uses the software layering concept. The outer software layers use the inner ones. The nucleus makes the object types mentioned above available to the outer layers. The Basic Input Output System (BIOS) layer allows the outer layer to communicate with any peripheral input/output device in a uniform manner. This assumes the hardware-specific device driver was written for the device using Intel's prescribed format. Devices can therefore be

opened, closed, read from, and written to. The next higher layer, the EIOS, makes it even easier to communicate with the I/O devices. Although there are more outer layers available, our software is only built on those mentioned.

8 - DESCRIPTION OF SOFTWARE

As mentioned above, the laboratory is to be used as "rapid prototyping" tool for investigating the workstation design with respect to the man-machine interface. With this in mind, most of the application software consists of switches and graphics data files. This is consistent with our goals of minimizing the amount of new code we have to write for a new application or modification to an existing one.

Figure 7 shows a typical software configuration that allows rapid reconfiguring of the hardware and software. Each 80286-based microcomputer running under iRMX-86 can be considered a processor node linked to each other over a communication network (presently RS-232 and later upgraded to Intel's Bitbus).

When the touch-sensitive display is touched, the position of the touch point is sent over the RS-232 link to the host microcomputer. The touch bezel task receives this position, converts it to screen (pixel) coordinates, consults the displayed switches file, and determines which switch is touched. It then sends a message identifying the switch to the switch manager task's mailbox. When the switch manager gets the message it consults the switches file, finds the switch's data structure, and executes the prescribed action to be taken based on the selected state machine's outputs, as described below.

This software configuration is advantageous from the reliability and fault-tolerance point of view. If the same or similar hardware exists on two or more systems and if it fails on one, it is possible to still perform the function if a failure detection task routed the messages to the other system. Also, if a task requires a resource (hardware) that is busy or is only on another system, then a task manager can assign it to that resource on another system, if it exists.

Our configuration is hardware-independent. Each peripheral of the same type has associated with it a set of well-defined primitives and, similarly, each task has messages that it understands. Whenever a new peripheral is interfaced to our workstation, we simply write a software device driver for it. This device driver is the software to hardware interface that performs the hardware functions specified by the primitives. For example, all graphics systems in our workstation recognize primitives such as DrawLine, DrawCircle, etc. With this capability, any application display can be run on any monitor. Another result of this configuration is that it is easy to control any function from any input device (touch bezel, voice control, etc) by causing the same messages to be generated when the inputs from different devices are considered the same.

9 - FINITE STATE MACHINE CONCEPT

Common to all workstations are ways of inputting information to control functions such as selecting modes of a system and activation or deactivation of a function. In order to simplify the management of a large number of input touch switch selections, it is necessary to apply the same general scheme for all switches, rather than program the operation of each separately. In addition, it is advantageous to keep information about the switches in a data file rather than in the code. This makes it easier to make modifications to a given switch without recompiling the program. Typical fields in our touch switch data structure are described as follows:

```
Type
   TouchSwitch=Record
      PositionOnScreen:Point;
      SwitchLength, SwitchWidth:Integer;
      NameInSwitch:String;
      SwitchDisplayed:Boolean;
      PresentState:Integer;
      StateMachineToUse:Integer;
   End;
```

With this information, if a switch is touched, then the software searches the switches file, finds the switch that was touched, and then uses the StateMachineToUse field to determine what action to take. Each switch is modelled as a state machine, which is an extremely powerful and useful mathematical concept having roots in autonoma theory. This finite state automation model describes the behavior of a system as follows: the next state of a system is determined by the present state and input; the output of a system is determined by the present state and input. Figure 9 shows how this model can be graphically represented by a state diagram and by a matrix when applying it to the "ACCEL" touch switch on the OMV display. Each one of the touch

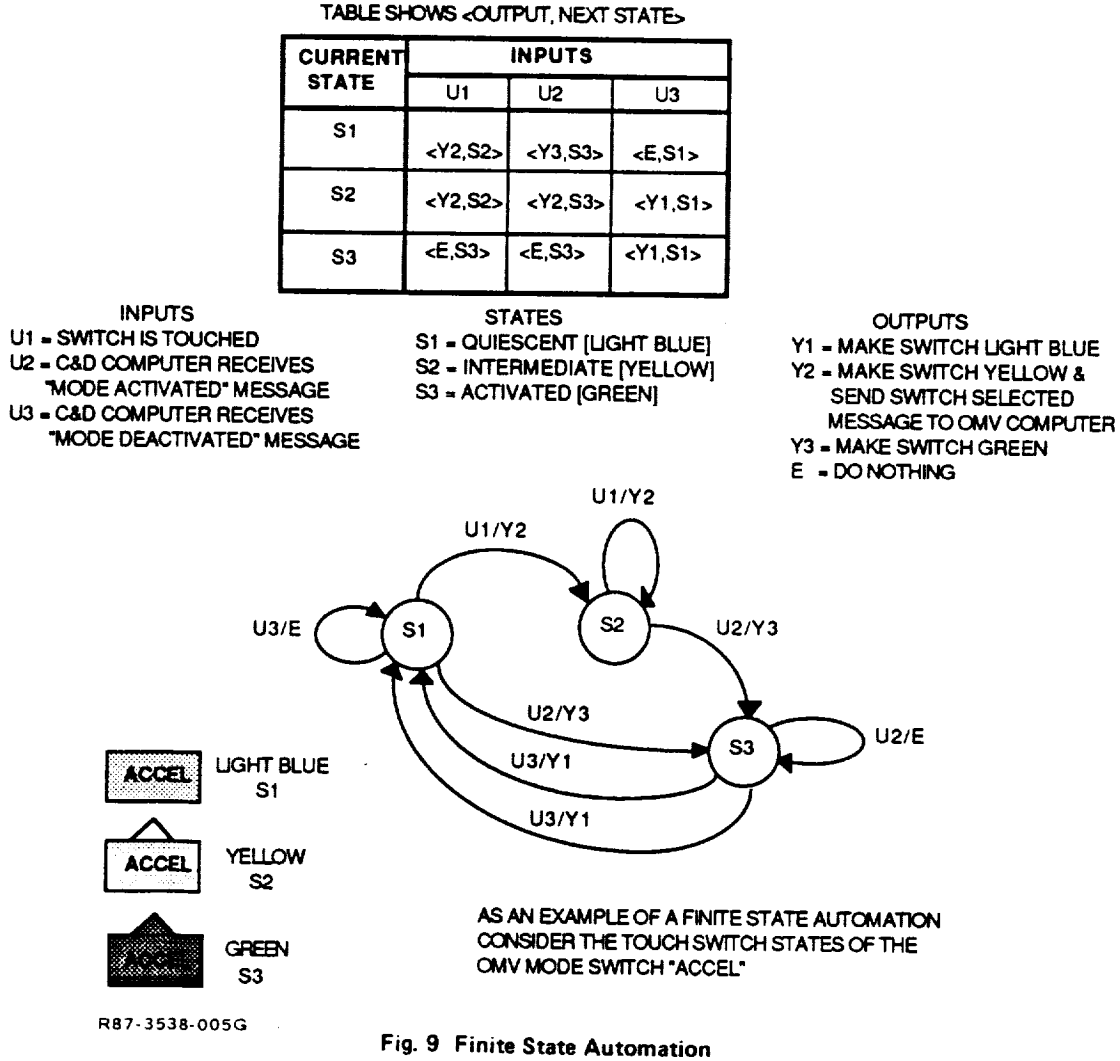


Fig. 9 Finite State Automation

switches has its own state machine associated with it. Typical state machine outputs are:

1. Load and display a switches file
2. Display graphics file
3. Set a switch to a color
4. Send a message to a computer
5. Send a message to a task's mailbox
6. Suspend/resume a task.

Outputs associated with windows are open, close, move, scroll, and scale.

10 - GRAPHICS INTERPRETER

Although not all our graphics hardware is the same, we can run all our software on any one of the graphics systems. This is made possible by defining and writing the same low-level graphics primitives (such as MoveCursorTo(x,y); DrawLineTo(x,y); DrawCircle(x,y,r)) for each system. Therefore, graphics applications software just calls these primitives and the pictures can be presented on any one of the graphics systems. In addition, by putting our display list, consisting of a sequence of primitives, in a data file, our software can interpret this list and draw the picture. This is a tremendous advantage, for graphics pictures can be changed immediately by changing the data file.

11 - EXPERT SYSTEMS

This popular subfield of artificial intelligence allows a program to be broken up into a knowledge base and an inference engine. Using schemes such as forward and backward chaining, the inference engine searches the knowledge base in an attempt to satisfy the present goal. This knowledge base contains if/then rules about the subject at hand, which is called the domain of discourse.

We plan to apply expert system schemes in order to assist the operator at the workstation by automating some of the intelligent decisions that the operator normally makes, such as displaying relevant data at

certain times when necessary. Also, we would like to increase the reliability of the voice recognition system by having the expert system consider the semantics of the sequence of voice commands rather than basing an acceptance/rejection decision only on a recognition threshold.

12 - SIMULATION

A number of different RMS and OMV simulations were performed using the LASS facility's 6 degree-of-freedom motion base. The Advanced Space Cockpit was used as the control station for all the simulations. Maneuvering of the simulated OMV and RMS was controlled by hand controllers. In some of the simulation scenarios, two 3 degree of freedom hand controllers were interchanged with a 6 degree of freedom hand controller for comparison purposes. Operations were conducted using views from a CCTV camera mounted on the motion base, in addition to other camera views. All modes were selected using the touch switches. Docking reticles and force/moment displays, overlaying the live camera image, were available in the center of the screen so that the operator never had to move his eyes from that point.

Real dynamical RMS and OMV math models were used to drive the LASS motion base so that it responded like the real system. Computer communications update times were approximately 50 milliseconds, and display update times were about 250 milliseconds.

Other simulation scenarios use the windowing display. Thermal Control and EVA monitoring, checkout, and servicing scenarios were performed. In Fig. 3, a typical situation using a multiple window display shows an out-of-tolerance pump speed in the Liquid Transport Circuit, in the EVA subsystem. Due to consistency of workstations, operators can easily move from OMV to RMS to TCS and finally to EVA without having to relearn the MMI protocol.

Note that although our workstations are used in simulations, based on our software architecture, it is possible, with a minimum software change, to connect the workstations to the hardware used in an actual

system. This can be done by replacing the "simulation tasks" with tasks that communicate with a host computer and real payloads.

13 - CONCLUSIONS

Traditional programs can be viewed as consisting of data and procedures operating on the data. However, modern programming systems using abstract models (such as the concept of "objects", which hides implementation, software layering, or finite state automations) makes the design of a complicated software project easier and more manageable.[7] Use of other popular modern concepts such as windows, expert systems, voice recognition and touch input improves the man-machine interface by decreasing operator training time and allows him to more naturally communicate with the computer. Future workstations should be easily reconfigurable, have a consistent human-machine communication protocol, and be intelligent, so that much of the operator's decision-making becomes automated.

14 - REFERENCES

[1] Daniel Ingalls, "Design Principles Behind Smalltalk", Byte, August 1981, Vol. 6 No. 8

[2] Ramachendra P. Batni, "Software Development Evolves Into Software Engineering", Computer Design, September 1984, Vol. 23, No. 10

[3] K.S. Booth, et al, "Anthropomorphic Programming", University of Waterloo computer Science Dept. monograph CS-82-47

[4] Intel Corporation, "iRMX^{TM} 86 Nucleus Reference Manual"

[5] A.D. D'Amico, "Simulation Test of Man-Machine Interface for Remote Manipulator System Operations in the Space Station", Proceedings of the Conference on Remote Systems and Robotics in Hostile Environments, March 31, 1987

[6] "An Introduction to Touch Technology", Carroll Touch, PO Box 1309, Round Rock, Texas, 78680

[7] R.E. Filman and D.P. Friedman, Coordinated Computing:Tools and Techniques for Distributed Software, McGraw-Hill, 1984