NAVAL POSTGRADUATE SCHOOL

Monterey, California

THESIS

ARGOS: DESIGN AND DEVELOPMENT OF OBJECT-ORIENTED, EVENT-DRIVEN MULTIMEDIA DATA BASE TECHNOLOGY IN SUPPORT OF THE PAPERLESS SHIP

by

Kevin F. Duffy and B. B. Giannotti

December 1988

Thesis Advisor: C. Thomas Wu

Security Classification of this page

REPORT DOCUMENTATION PAGE

1a Report Security Classification: Unclassified	1b Restrictive Markings
2a Security Classification Authority	3 Distribution Availability of Report: Approved for public release; distribution is unlimited.
2b Declassification/Downgrading Schedule	
4 Performing Organization Report Number(s)	5 Monitoring Organization Report Number(s)
6a Name of Performing Organization: Naval Postgraduate School; 6b Office Symbol (*If Applicable*): 37	7a Name of Monitoring Organization: Naval Postgraduate School
6c Address (*city, state, and ZIP code*): Monterey, CA 93943-5000	7b Address (*city, state, and ZIP code*): Monterey, CA 93943-5000
8a Name of Funding/Sponsoring Organization; 8b Office Symbol (*If Applicable*)	9 Procurement Instrument Identification Number
8c Address (*city, state, and ZIP code*)	10 Source of Funding Numbers: Program Element Number; Project No; Task No; Work Unit Accession No

11 Title (*Include Security Classification*): Argos: Design and Development of Object-Oriented, Event-Driven MultiMedia Data Base Technology in Support of the Paperless Ship

12 Personal Author(s): Kevin F. Duffy and B. B. Giannotti

13a Type of Report	13b Time Covered	14 Date of Report (*year, month,day*)	15 Page Count
Master's Thesis	From To	December 1988	185

16 Supplementary Notation: The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

17 Cosati Codes			18 Subject Terms (*continue on reverse if necessary and identify by block number*)
Field	Group	Subgroup	Object-Oriented, Event Driven, Multi-Media Data Base, Paperless Ship, HyperMedia

19 Abstract (*continue on reverse if necessary and identify by block number*

Argos is a prototype multimedia database developed as both a Battle Group Commander's assesment tool and a shipboard data management tool. The current prototype developed by using HyperCard/Macintosh demonstrates an effective utilization of off-the-shelf technology to solve real world problems commonly faced by the United States Navy. The ultimate goal of Argos is to provide database support for the "Paperless Ship".

20 Distribution/Availability of Abstract: [X] unclassified/unlimited [ ] same as report [ ] DTIC users	21 Abstract Security Classification: Unclassified
22a Name of Responsible Individual: C. Wu	22b Telephone (*Include Area code*): (408) 646-3391; 22c Office SymbolT.: 52Wq

DD FORM 1473, 84 MAR 83 APR edition may be used until exhausted

All other editions are obsolete

security classification of this page: Unclassified

Approved for public release; distribution is unlimited

Argos: Design and Development of Object-Oriented, Event-Driven MultiMedia Data Base Technology in Support of the Paperless Ship

Kevin F. Duffy
Lieutenant, United States Navy
B.S., Purdue University, 1980

and

B. B. Giannotti
Commander, United States Navy
B.S., United States Naval Academy, 1972

Submitted in partial fulfillment of the requirements for the degree of

MASTERS OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1988

ABSTRACT

Argos is a prototype multimedia database developed as both a Battle Group Commander's assesment tool and a shipboard data management tool. The current prototype developed by using HyperCard/Macintosh demonstrates an effective utilization of off-the-shelf technology to solve real world problems commonly faced by the United States Navy. The ultimate goal of Argos is to provide database support for the "Paperless Ship".

TABLE OF CONTENTS

I. INTRODUCTION

United States Naval Warships by design are self-sufficient entities in both their war fighting capabilities and their ability to support sustained open ocean operations. It has been determined that an Oliver Hazard Perry guided missile frigate (FFG-7 class) with a crew of 185 men, nominal 3500 ton displacement, and overall 445 foot length carries in excess of 20 tons of paper in direct support of the ship's mission and her crew [Ref. 1:pp. 157-159]. The onboard paper includes executive correspondence, required reports, training records, performance evaluations, medical/dental records, financial management records, technical manuals, equipment operator manuals etc. The reduction of this bulk of paper is the quickest way for the U.S. Navy to significantly improve its war fighting capability. Besides the obvious increase in war fighting potential of the ship by replacing the 20 plus tons of paper with missiles, torpedoes, projectiles, and other offensive or defensive equipment, the reduction in manpower required to maintain the paperwork infrastructure would be substantial. This thesis will explore the feasibility of the "paperless ship."

Simple elimination of paper from the ship, of course, is not a solution. We must maintain the informational content of the current onboard paperwork while physically removing it from the ship. In other words, we would like to convert the entire paperwork infrastructure into some electronic media and provide an automated system that manages all of this newly stored data. The concept of a "paperless ship" was first made popular by a former director of Surface Warfare, VADM J. Metcalf,USN (Retired). It was generally believed that the paperless ship could not be developed. It was a sweet dream that would not happen. The reason for this

pessimism, we believe, stems from the mind set of the people who attempted to synthesize a solution by utilizing the wrong technology.

In the DoD community, relational database management systems (DBMS) are widely held as the solution to the information management problem. In fact, many agencies under the DoD umbrella require that future (non-numerical) application software must be developed by using a relational DBMS that supports the SQL query language. The relational DBMS is an improvement over older database management systems such as hierarchical and network DBMS, but it is not a panacea for all data management problems. The relational DBMS is definitely not a solution for the paperless ship. It is the wrong technology.

There are several reasons why the relational DBMS is the wrong technology. First, the semantics involved in the non-tactical operations (supported by the onboard paper infrastructure) are far too complex to be modeled by the relational DBMS. Second, information conveyed by onboard paper comes in various forms such as images, variable-length text, graphics, etc., while the relational DBMS is only capable of handling fixed-size records. Third, the query language supported by a relational DBMS is too difficult, if not impossible, for the average sailor to utilize. Lastly, incremental modular construction (i.e., building piece by piece) of the system is not practical with the relational DBMS because of the difficulty involved in redefining the relationally defined information structure.

Failure of the relational DBMS does not mean that the paperless ship cannot be realized, just as Babbage's failure to use a steam engine to run his first computer did not mean that the computer was beyond reach. By applying the right technology, we can take a substantial step toward fulfilling the dream of the paperless ship. But is there a technology today that can make the dream come true? After all, it took about

100 years since the idea was conceived to apply the right technology (electricity) in building the first real computer. We believe there is a technology today and have built a prototype system to demonstrate that this technology is a viable solution.

What we need to build the paperless ship is a multimedia DBMS, a database management system capable of handling multimedia data including images, signals, and text. The ideal multimedia DBMS for the paperless ship must have a easy-to-learn and easy-to-use interface for querying and retrieving data, allow modular construction of a database, and be operable under various operating systems. Unfortunately, there is no multimedia DBMS, capable of supporting traditional database functions such as concurrency, security, recovery, etc., beyond managing multimedia data, available today. We, therefore, decided to employ HyperCard™[1] for our prototype.

The HyperCard choice was predicated on several features that other development systems do not offer. Specifically, HyperCard's object-oriented properties support ease of development and portability and reusability of modules. In addition, these same object-oriented capabilities provide the developer with a rapid, interactive prototyping environment that greatly enhances debugging and culminates in significantly increased robustness than in conventional programming environments.

HyperCard provides the developer with a great deal of power and a rich set of development tools, that when combined establish a degree of compatibility and cognitive richness found in few other environments. The human factors engineering principles and human interface technology found within the Macintosh operating

---

[1]HyperCard™, HyperTalk™, Stackware™ and Macintosh™ are all trademarks of Apple Computer Incorporated.

system have been extended into HyperCard. This inclusion allows easy development of applications that provide the user with an instinctively friendly look and feel. The importance of the look and feel is fundamental to a significantly reduced learning curve and a dramatically increased level of usability for applications developed in the HyperCard environment. The rich set of development tools that extend beyond HyperCard enable the developer to easily acquire, manipulate and import text, sound and graphics into HyperCard without data conversion.

The availability, compatibility and cost of peripheral hardware for the Macintosh was also a major consideration. All graphics utilized in this application were scanned utilizing a two hundred dollar Thunderware™ ThunderScan® and the human voice speech was digitized using a two hundred dollar Farallon™ MacRecorder®[2]. CD ROM technology is also available and HyperCard has built in capabilities for exploiting this technology. Video driver software, which interfaces with HyperCard is also commercially available for many models of VideoDisk.

To demonstrate the viability of a multimedia database in support of the paperless ship project, a functional program called **Argos**, named after the most famous ship builder in Greek mythology, has been developed and tested. Due to limited time and resources, this implementation was aimed primarily at the maintenance function: specifically the LM2500 gas turbine engine. However, it must be noted that **Argos** is not designed to demonstrate applicability of a specific function, rather its purpose is validate the integration possibilities across the entire functional spectrum of military units both at sea and ashore.

---

[2]Thunderscan® is copyrighted by Thunderware™, and MacRecorder® is copyrighted by Farallon Computing Incorporated.

This thesis is organized as follows. Chapter II discusses the paperless ship problem in detail. A brief overview of the programming environment and its inherent language is presented in chapter III. Chapter IV provides implementation details. Conclusions and recommendations for follow-on thesis work are presented in chapter V. All code and graphical images are contained in the appendices.

II. THE PROBLEM STATEMENT

In the age of long-range over-the-horizon tactical weapon systems, three dimensional radars and state of the art propulsion plants, one of the least glamorous and most easily overlooked tasks is that of automating the nontactical functional areas in a shipboard environment. Efforts that have been ongoing for a number of years to implement a Navy wide program to directly support shipboard supply and maintenance functions have had limited success; additionally, very little has been done to automate routine shipboard administration, personnel, medical and operational functions. From an ISIC's (Immediate Superior in the Chain of Command) perspective, whether that be the Battle Group Commander or a Desron (Destroyer Squadron) Commander etc, limited information access on a real time basis is a significant deficiency. As has been demonstrated time and again, self sufficiency and integrated Battle Group Logistics during any operational contingency is a corner stone to the overall success of deployed units. The purpose of **Argos** is to enhance the ISIC's information access by giving him instant access to the nontactical functional data bases from which his ships are operating. Whether the issue is locating a spare part in one ship of the force to correct a casualty in another, identifying a trained technician to provide a technical assist, collecting retention statistics, checking the training posture of the ships under his command, or assembling information to answer any of a myriad of questions that are typically solved by cluttering already overburdened communications nets, **Argos** has the potential to immediately alleviate if not solve the information access problem facing the ISIC. This alone would positively contribute to reducing a ship's administrative overhead of responding to multiple queries.

While the problem of information access is obviously exacerbated by geographic location, it is not limited to the ISIC. Each ship's Commanding Officer faces similar problems when attempting to assess his maintenance related support posture; training status; operational readiness; medical and dental readiness etc. By eliminating the inherent time delays of manually collating information, individual productivity will sharply increase.

Other issues faced by the Navy include the per installation hardware costs. These costs quickly become a major issue when a single installation for the SNAP II system utilizing the Harris mini computer currently installed in most combatant ships today costs $250,000 and it provides a mere four work stations, where as the system we are proposing runs on micro computers that cost $500 to $1000 per copy. Training costs are another consideration. These costs in terms of both time and money are significantly higher with the SNAP II Harris installation. For example in 1983 when this system was first introduced to the fleet, sailors designated as "system users" had to be sent to a facility in Texas for two weeks of hands on training to guarantee proficiency at no more than the user level. While the absolute costs associated with user friendliness are reflected somewhat in training costs, they actually go much further and are more difficult to categorize. Specifically we do not define menu driven or command line systems as user friendly. The psychological resistance of the user to a non user friendly system is particularly difficult to quantify, although the bottom line result is a substantial loss in productivity. The lack of user friendliness of currently installed shipboard systems results in an inability of the user to realize the full potential of the system which in many cases ultimately creates more work than the older manual system. Because most systems purchased for fleet introduction do not represent proven off-the-shelf technology,

user base to verify the system hardware and software. This alone detracts from system robustness; additionally, further degradations are experienced as a result of attempts to model complex data with systems that are technologically incapable of performing this function. Consequently, the lack of robustness manifests itself to the user as an inability to perform as advertised or desired.

Correctness of data is crucial to the success of any database system. A key element in verification of the data's correctness is the ability of designated personnel to easily view the data in a discrete isolated format that is familiar to the viewer. Current systems do not provide this capability, thus a preponderance of the data that fleet personnel are working with is either incomplete or incorrect. A specific example is the SNAP II system. Five years after its introduction to the fleet, substantial amounts of money are still being dedicated to correcting the SNAP II databases.

III. THE PROGRAMMING ENVIRONMENT: HYPERCARD

In this chapter we describe our development tool HyperCard/HyperTalk and show how conducive it is for modeling complex data types and developing multimedia databases. A relatively new programming environment called HyperCard was developed by Apple Computer® Incorporated to run on the Macintosh family of computers. It is presently being marketed as an extension to the Macintosh operating system. Version 1.2 is used for the development of this thesis. HyperCard is an event-driven, object-oriented programming environment that is driven by messages to and from objects. Actions are initiated in response to events which then send a chain reaction of messages from one object to another. HyperCard, which contains a general purpose programming language called HyperTalk, provides powerful painting tools, editing functions and semiautomatic program development. HyperCard is a multimedia development system that allows developers to easily integrate graphics, text and audio into an object-oriented environment.

HyperCard's interface is very intuitive and easy to learn; both for the developer and the user. The low end of the development spectrum allows non programmers to create very professional looking stacks without writing a line of code. At the high end of the development spectrum, programmers may create powerful functions and commands written in Pascal, C, or Assembler which may not be currently available in HyperTalk's rich instruction set. HyperCard is a tremendous labor saving intuitive developmental environment that has extended the concepts of integration in software development.

HyperCard, which uses the metaphor of a stack as an object that can hold both processes and data, exists only in the context of a stack. A stack should not be confused with the classical data structure stack in which elements may only be placed on or removed from the top. A stack is more analogous to a stack of three by five cards in which cards may be accessed at any point. HyperCard supports development of stacks that allow data, which may be any combination of text, graphics and sound, to be stored, linked, searched and viewed. This is the basis for multimedia database applications. Information may be linked relationally within a stack or from one stack to another. HyperCard was not designed to replace traditional relational databases; however, if integrated properly it can greatly enhance their capabilities as a front end processor. As a stand alone multimedia database developmental tool, HyperCard is clearly a powerful user friendly system that allows applications to be constructed in minutes that would require a monumental effort in a conventional programming language.

As an object-oriented programming environment HyperCard has pre-defined five objects. They are buttons, fields, backgrounds, cards and stacks. Some objects have a unique HyperCard generated ID which will never be repeated in the same stack. All HyperCard objects can send and receive messages; have properties including script which is code associated with that particular object; and have a visible representation that may be set on or off. A button is an area on the card that is accessible with the mouse pointer. Buttons may be graphical, textual, a combination of both or totally invisible. When the user clicks the mouse pointer on a button, a message will be sent to the button and the script of the button will be executed. A field is an area to store textual data on a given card. Fields are not static. They may be adjusted to any size or appearance desired. Field scripts, like all scripts

scripts in HyperCard, are also event driven. Backgrounds are objects that cards oftentimes share giving them a homogeneous look. An example might be the visual representation of a ROLODEX card [Ref. 2:p. 112]. Cards are the objects on which fields, buttons and backgrounds reside. One to several million cards form a stack. The stack along with the four objects it contains (i.e., cards, backgrounds, buttons and fields), and any attached resources is the executable program.

Modularity is a property of objects in HyperCard. Once an object is created it may be moved in its entirety to another stack with its graphical appearance, scripts (program code) and resources (predefined object code i.e., icons, dialogs, sound etc.) This was extremely important in facilitating rapid prototype development and code reusability.

Message sending is a characteristic of an object-oriented programming environment. HyperCard generates messages, called system messages, which are sent to objects in response to certain program events. Consequently, the HyperCard environment allows development of databases with procedural attachment which readily lends itself to modeling of real world data. Procedural attachment is the attachment of procedures to data elements and containers which provides behavioral characteristics in the form of code. It enables development of multiple data associations via dynamic and static linking; thereby resulting in the best of conventional database (relational and hierarchical) technology associations. Additionally, as a direct result of procedural attachment's ability to support modular code in a fully specified high level language the developer enjoys the freedom of semantic data modeling. Search, browse and reporting capabilities are established in a similar manner.

Whenever script is executed a message is generated. The first object to receive the message is the sending object and if it has a message handler (a subroutine in HyperTalk) it will execute the handler. The script can also call the same message handler from which it originated. This is recursion in HyperTalk. HyperTalk is also capable of nesting. which would occur if handler1 in ObjectA calls handler2. Capabilities that allow recursion and the creation of procedures as well as push down data structures provide the developer with a full range of programming tools found in more conventional languages such as Pascal or C.

HyperCard has two types of objects: transparent and opaque. Transparent objects are virtually invisible, that is they allow the viewer to look down to layers below the actual top layer. Opaque objects are solid; consequently, they block the viewer from observing objects situated directly below. Every HyperCard object is created in its own layer. Layers can best be visualized as ultra-thin sheets of clear plastic [Ref. 2:pp. 109-110]. Since every object gets created in its own layer, the layers are placed one on top of the other as the objects get added to the stack. Opaque objects are visible through all layers of the stack regardless of their relative stack position; unless however, they get covered by another opaque object in a subsequent layer which would render the lower object impossible to be seen by anyone looking down. Transparent objects allow the viewer to observe opaque objects below. Buttons, which are a type of HyperCard object, can be layered into a stack like any other object: whether transparent or opaque, buttons will react to mouse clicks regardless of depth in a layer. If buttons are layered on top of each other only the top button will respond to a mouse click. Visibility is a property which many HyperCard objects possess. It is clearly different from transparency. That is, when an object's visibility is set to false the object not only cannot be seen but is also inactive.

but is also inactive. However, attributes of an object, whose visibility is set to false may be obtained or changed through the scripting language. Visibility and layering together provide the developer with the ability to construct complex data structures and establish inheritance of code by layering buttons on top of one another and passing discrete commands on to various layers. This is a very powerful tool that in general facilitates compactness and reusability of code. Specifically, the invisible button was essential for the development of **Argos** because it is the mechanism by which the user is able to define a pathway to a desired piece of equipment, subcomponent, or piece part. All that is required is for the user to click on a graphic and **Argos** will respond by displaying a blowup of the selected graphic.

Background and card layers are the second category of layer in HyperCard. Everything assigned to a background is active and visible on every card of that same background in the stack, that is anything placed into a background design gets copied onto every other card with the same background: this includes graphics, text, and buttons. The programmer may choose to place graphics, text, or buttons onto a specific card and have them visible only when that particular card is the top card in the stack, by placing those objects into the card domain. All objects in the card domain are in the very top layers of the stack with background objects lying below. The card domain can also be called the foreground. Conceptually, it is very important to note that card objects are visible and active only in their respective layers, whereas background objects are visible and active for all cards sharing a particular background. In terms of creating applications, this subtle difference between foreground cards and backgrounds becomes an indispensable tool for the programmer to hide certain action buttons from the user at different points of the program by covering up an action button on a background card with an opaque object

on a foreground card. Of interest, background buttons can be created only one at a time, each in its own layer; however, the user can not discern any difference between the two buttons or the two layers because both opaque buttons are readily visible and show no obvious indications of being in two completely different layers. Careful manipulation of background and card layers enables the programmer to develop a particular look and feel that results in very user friendly interfaces. This allows complex modeling of data structures that are analogous to everyday metaphors.

System protection in **Argos**, which is important for establishing program and data integrity, is easily supported by HyperCard because of its inherent stack protection mechanism. Stack protection is provided by the system. The level of protection is determined by the programmer and is assigned via the PROTECT STACK menu which ostensibly allows the programmer to choose any level of protection desired. Passwords are available options that can be used to protect a particular stack. The password need only be entered once during a session to allow access to a given stack. The system will remember that the user knows the correct password once it is entered to eliminate the tedious requirement of reentering a password on every attempted entry to protected stacks during a single session. A more advanced level of protection exists by using the scripting language HyperTalk which allows the programmer to limit the data which may be accessed down to the data element level. This capability may be extended to password protection which can be applied to protect a specific data element or a specific function vice the more traditional login password protection schemes currently in use on Navy systems such as SNAP II. The traditional login password protection schemes tend to be more limited than the **Argos** password system because the HyperTalk scripting language allows complete flexibility in limiting users to specific functions or data elements.

Another extremely important aspect of HyperCard is its linking ability. HyperCard links are a method of establishing a unidirectional pathway from one card to another. Links may be between cards in the same or different stacks regardless of either card's relative stack position. Bi-directional links can also be programmed by inserting a unidirectional button on each card such that each card has a pathway to the other. To establish links between cards in the same stack either the unique card identification number is used as a destination address, or the card name can be used. Links to cards in different stacks are exactly the same with the addition of the new stack name to the destination address. HyperCard linking enables the programmer to implement true conceptual relational database applications, that is, data never needs to be duplicated as in most other conventional data base systems, i.e., there is no data redundancy. This is accomplished by HyperCard's ability create links via unique identification numbers that are independent of data content.

HyperTalk is a general purpose programming language that contains a robust set of commands and functions. It is also a special purpose language that tends to be better for some programming tasks than most other languages, such as construction of visual databases and educational systems. It is a very intuitive and natural language which tends to favor nonprogramers in its grammatical style. This concept is further extended by the language property that provides programmers world wide the capability to program in their native language. The object-oriented nature of HyperTalk makes the scripting portion of the programs compact, extremely easy to debug and very portable from one program to the other. The finished programs tend to be very intuitive for the user to operate and have a visual look and feel that in other languages would be very difficult to obtain. This makes HyperTalk a very labor saving programming language. One of the most powerful features of HyperTalk are

the XCMD's and XFNC's that allow virtual unlimited extendability. When HyperTalk was created two very powerful interface capabilities were installed called XCMD (external command) and XFNC (external function). These two items enable HyperTalk to search the resource fork of the stack for a command or function if it is not found within the stack script. This capability provides virtual unlimited extendability to the HyperTalk language. HyperTalk will search the resource fork of the stack for an unknown command of type XCMD and likewise will search for an unknown function of type XFNC for an unknown function. Therefore when an author of a HyperTalk script wishes to extend the language of HyperTalk, he can write the function or command in Pascal, C or assembler and move it into the resource of the Stack where he wishes to use it. Consequently, extensions to the HyperTalk language are always carried with the individual stacks that require them. Selected commands and functions from a library of XCMD's and XFNC's are easily moved in and out of Stacks as desired.

The HyperTalk scripting language is totally unique among programming languages. However, it does derive its basic approach from Smalltalk and Pascal resulting in an overall appearance very similar to that of a natural language. Command structures are english-like sentences or phrases such as repeat five times or put the answer into it. HyperTalk is extremely forgiving in syntax and it allows multiple command structure variations. This is a very important distinction in terms of ease of programming and project implementation of the **Argos** prototype because it will allow utilization of in-house Navy programming assets which would be a significant cost savings.

Functions in HyperCard may be one of three types: HyperTalk defined, user defined, or XFNC. HyperTalk functions behave in the same fashion as

conventional programming language functions. When a function is invoked in HyperTalk, the HyperTalk scripts are searched in a hierarchical fashion until it finds a match. If it doesn't find a match then the resource fork is checked to determine if a XFNC is available. This method of determining function location allows the programmer to redefine system functions as well as define entirely new ones. The ability to redefine the environment proved to be invaluable as this project progressed. For example, amongst others, the "find string" command had to be redefined to maintain the "look and feel" we were attempting to create in **Argos**. The importance of XCMD and XFNC can not be over stated. While HyperTalk is powerful enough to handle most programing requirements, the ability to write XCMD and XFNC in higher level languages such as pascal or C provides an extremely powerful tool. This capability allows discrete external functions and procedures to be executed from within **Argos**.

There are two sound commands available in HyperTalk, play and beep. Play requires a SND type resource to be available in the stack for the voice parameter. HyperCard originally came with four sound resources, Harpsichord, Boing, silence, and Dialing tones. These resources are the voice in which the command operates. The play command is used to play digitized sound or to play music from a string of notes. Beep is used to invoke the system beep. Another common sound command which is a XCMD called Talk[3]. Talk, which uses another program called MacinTalk[4], converts text or phonemes into speech. Both SND and the XCMD

---

[3] The Talk XCMD was create by James L. Paul of Paul Software Engineering.

[4] MacinTalk is a product of Apple Computer Incorporated.

"talk" were utilized extensively throughout **Argos** to appeal to the user's audio sense as we continued to develop very user friendly "look and feel."

The most significant criticism of HyperCard is that its language HyperTalk is an interpretive language. The interpretive nature of HyperTalk in several instances noticeably slows execution; however, in other cases, such as searching or card selection, HyperCard is phenomenally fast. HyperCard is a very large developmental system and requires a minimum of 1 megabyte of RAM to operate. It can display only one card of a fixed size, in black and white on the screen at a time; however, the illusion of a smaller size card is very easy to implement. HyperCard strays from the standard Macintosh interface guidelines, in that the applications written in HyperCard are driven primarily by buttons vice the pull down menu style of all other Macintosh applications. Some of the built in features which allow easy access to the system routines, such as the answer command, have a one line text limitation in the question variable. Many of these weaknesses have been remedied by individual programmers and are readily available in the public domain in the form of XFNC's (external function) and XCMD's (external command), which are written in Pascal, C or Assembler and extend the HyperTalk language.

IV. IMPLEMENTATION

As designed and implemented the **Argos** prototype is based upon a series of graphical stacks that allow the user to visualize images of what he or she is dealing with. For example, rather than relying upon conveying the meaning of an LM2500 gas turbine engine and all of its component parts through a textual description, as is required by conventional database schemes, **Argos** provides the user with a visible graphic interface that is coupled with a technical textual description thereby reducing the technical knowledge required to be proficient in using this software.

The stack construct allows strict adherence to modular design. This is critical to any dynamic software system that is expected to be responsive to periodic updates resulting from organizational changes, equipment additions and removals, personnel receipts and detachments etc. By maintaining strict modularity, changes can be implemented readily as they occur with no adverse side effects to other modules or perturbations to other system users.

The user enters **Argos** at the Battle Group level which enables one to choose a ship by name and then to choose one of the functional areas each of which is a discrete stack. Six functional areas were modeled in **Argos**: maintenance, supply, operations, medical, administration and personnel. This by no means was intended to be all inclusive or static, rather it merely represents the types of functional areas that may be modeled in the system. There are conceptually six virtual stacks (one per functional area) for each ship in the Battle Group. Each functional area is represented as a virtual stack because within each virtual functional area stack are the actual individual component stacks. That is, each piece of equipment installed in any ship modeled by **Argos** has its own stack that represents the myriad of subcomponents

that are integral to that piece of equipment. This demonstrates the modularity referred to above. As equipment upgrades occur and components change, only one stack needs to be updated, not the entire software system. This concept can be taken a step further. When a new ship is under construction, a complete software system such as **Argos** can also be installed by simply taking copies of the equipment modules along with modules for the other functional areas and simply placing them aboard the ship upon delivery to the Navy.

Argos has been developed to represent discrete functional areas common to every ship of the Navy rather than limiting the implementation to a single facet or functional area. Underlying the **Argos** system is the ability for complete integration of data. This facilitates data sharing amongst the functional areas and enables the user to make ad hoc queries because of the uniform interface that acts as a template for the **Argos** system. From the Battle Group level this allows module integration for such queries (i.e., list by name and pay grade all of the gas turbine technical experts currently in the Battle Group). As a result of the modularity this entire system is very dynamic and can be expanded to model anything from a 172 foot diesel Minesweeper to a 1092 foot nuclear propulsion aircraft carrier by adding and deleting modules as appropriate. The modularity aspect of **Argos** makes it reusable software because discrete modules can be changed to reflect actual shipboard changes. Since a small desk top microcomputer is being utilized, the software and its computer can be literally hand carried or mailed any where in the world and be used within minutes of receipt.

Modular design was crucial to the success of this project. HyperCard was very important in this regard because it easily enabled us to achieve complete modularity due to its ability to represent a hierarchical scheme, relational scheme and

object-oriented scheme. Hierarchically it is able to represent the organizational infrastructure of the Navy (ISIC to Unit). Relationally it is able to represent any of the normal relations typically associated with conventional shipboard database applications. **Argos** also contains object-oriented database capabilities because each card contains both data and instructions.

Argos represents a mutimedia database. Graphics are utilized to represent objects that heretofore were solely represented by textual descriptors or attributes. Textual information is utilized to enhance the meaning of and further define whatever object the user is currently viewing. Audio is utilized as an additional feedback mechanism which is normally implemented solely in text if at all. When taken in aggregate, **Argos** makes use of the underlying features of multimedia to create a significantly more intuitive environment for the uninitiated or novice computer user.

Background buttons, which appear on every card of a stack that utilizes a given background, are an integral part of the look and feel of **Argos**. Examples of these background buttons include HELP, COSAL, Equip, CSMP, APL, Forms, Order, Techman etc. See Appendices A thru H for further amplification. The HELP button allows the user to quickly refer to a system reference manual should he or she get lost while navigating through any portion of this system. The ORDER button enables the user to automatically fill in and print out a requisition document for any piece of equipment or sub component modeled. The AWR button provides the user with the capability to automatically generate an equipment degradation report and to update the CSMP (Current Ship's Maintenance Project) once the required parts have been ordered. The TECHMAN button gives the user instant access to the pertinent sections of the technical manual that apply to the piece of equipment he or she is currently viewing. The COSAL, CSMP, EIC, and APL buttons allow the user

instant access to the entire ship's COSAL, CSMP, EIC, or APL for browsing should that be desired. The miniature card button located in the upper left of every card enables the user to return to the previous card viewed and in this manner can literally back out of the graphical path just navigated. The ship button located in the upper right corner of every card provides the user with the ability to return to the beginning of the graphical hierarchy of the maintenance stack so that multiple components can be investigated without having to exit and reenter the program. See Appendix F (Help Stack) for further information on buttons and icons.

All graphic buttons are invisible so that they can be positioned over the various graphics found on each card. For example, on each component card of the maintenance stack the graphic in the upper left corner changes from card to card yet the functionality of the button does not change i.e., to return the user to the previous card viewed. Special buttons are also utilized throughout this program. On the COMPRESSOR REAR FRAME/COMBUSTOR card the AIR SEALS button allows the user to advance to a different card that displays the air seals hidden between the compressor rear frame and the combustor and therefore not visible. Similarly, on the HP COMPRESSOR STATOR cards the ALT VIEW button enables the user to view different aspects of the HP compressor stator in order to see all sub components.

The HELP stack is an integral part of this entire program. It has been developed to provide the user with an intuitive look and feel that will answer any program specific questions that may arise at any level or point within **Argos**. HELP has a search function that eliminates the need to page through the entire stack to answer a single question and it fully defines the functionality of all background buttons.

Argos also includes all of the nontactical data files required by the maintenance functional area as stacks. The COSAL (Coordinated Ship's Allowance List)

delineates every piece of equipment that ship is allowed to have installed. The APL (Allowance Parts List) provides every repair part allowed to be carried in support of the installed equipment. The EIC (Equipment Index Code) provides part numbers for every subcomponent on a given ship. The FORMS stack has a card for every form required by the Maintenance functional area for the conduct of day to day business onboard any U.S.Naval warship.

In addition to the functionalities described above, several scripts were created in order to automate the development process. For example one developmental button was designed to rewrite all of the scripts on the graphics buttons on an individual card. Another script was written to automatically retrieve card id's for linking graphical parts to an associated card in the APL stack. See Appendix H for developer script listings. This capability significantly enhanced the development effort of **Argos**.

V. CONCLUSIONS

Argos has demonstrated that multimedia technology can be effectively used to create functional useable systems with significant advantages over conventional technology. Development time utilizing this technology is dramatically reduced due to its object-oriented nature, system environment, and rich set of development tools readily available throughout the development process. Prototype testing and demonstrations have verified the user friendliness of this system. Software maintenance on the Argos system tends to be relatively easy primarily due to its modular design. Consequently, the ripple effect so often associated with making changes to conventional software systems is totally avoided. Single component or functional stack updates or replacements are quick and quite easy to implement because there is very little data, if any, carried from one module to the next.

Unequivocally, the cost of this technology is a tremendous advantage. As mentioned earlier, development, maintenance, and training time are greatly reduced compared to software systems developed using conventional technology. The Argos development software itself was also very inexpensive. For example, the HyperCard software cost only fifty dollars and all other software tools utilized throughout prototype development cost less than five hundred dollars. Argos was developed on a Macintosh II with a 40 megabyte hard drive and a Macintosh SE with a 20 megabyte hard drive. The Macintosh II, however was not necessary to develop or run the completed system. A Macintosh plus or SE is quite sufficient to develop and operate the system; however, a slight speed advantage is noticeable between the Macintosh II and the SE and between the Macintosh SE and the plus. The Macintosh II system of this configuration may be purchased commercially for less than $6,000

and the SE for less than $3,000. An Apple imagewriter printer was also required to operate the scanning device. An image writer printer is commercially available for less than $500. The audio and graphic digitizers cost less than $500 combined. In order to fully implement this technology a mass storage device such as ROEM (removable optical erasable media) must be incorporated and these are available for approximately $2000 to $5000.

The level of user friendliness achieved in this implementation greatly contributes to significantly reduced training time and associated costs. There is no complex query language or command set for a user to remember, consequently a new user with rudimentary keyboard skills could be proficient with the **Argos** prototype system literally in a matter of minutes. This system requires only semantic or recognition knowledge rather than syntactic or rote knowledge for the user to become fully functional. Additionally, there is a uniform interface throughout **Argos** that further reduces training requirements and enhances usability.

The data has been optimized as a result of the system design. It is not redundant and may be further broken down into discrete stacks so that only the subset of interest must be traversed to find a particular data element. Data relations such as components or personnel must only be represented once and they may be freely linked to other functional areas for access and retrieval. Most of the functional operations are predefined by the implementation and new or altered operations may easily be added.

The data for the most part is stored as an integral part of the program stacks and are not discrete files. However, data is easily exported and imported from the stacks. At this stage of development the search algorithm is fully developed to support ad hoc textual queries; however, the ability to extract data from multiple stacks

simultaneously does not exist and is dependent upon the implementation of a complex query language. Also, there is no system facility to allow multiuser accesses and updates as is typically found in conventional data base technology. All data can be stored in the ROEM format. Password protection in HyperCard can be implemented to protect individual stacks as well as specific user functions such as the approval feature in Argos to control the expenditure of funds when ordering repair parts. While this technology is not designed to replace conventional database technology, it does have certain advantages that make it very desirable for applications such as Argos. Conversely, an application that requires thousands of transactions a second would not be appropriate for development in a multimedia programming environment.

Argos also offers a simplistic graphical interface to the shipboard supply support system. While current shipboard systems such as SNAP II are attempting to provide an interface to the supply system for maintenance related interactions, Argos takes this a step further by providing the technician with a mutimedia interface that uses the concept of user friendliness as its point of departure. By making the system as easy as possible to utilize, individual productivity for all users will significantly increase.

In terms of cost, Argos compares favorably. While systems currently installed use relatively expensive one of a kind mini computers built upon older technology, Argos is predicated on strictly off-the-shelf current, yet advanced, technology that is readily available in the market place. In terms of hardware, Argos requires a micro computer that is considerably less expensive as a capitol investment, yet for the application under consideration micros are every bit as functional as the larger more expensive mini computers. Micro computers have the added advantage of

significantly reduced space requirements for installation. With a much smaller footprint, the entire micro installation can easily sit an a desk which is a major consideration when discussing small combatants.

When considering unique one of a kind mini computer installations one must also consider the availability or nonavailability of hardware system repair parts. Unique hardware systems typically have unique repair parts which can lead to sole source contracting. Sole source contracting normally is not advantageous to the government and can lead to increased down time due to repair parts nonavailbility and to increased costs due to a lack of fair and open competition. Unique computer hardware systems that have very small user bases often times are dependent upon the end user to identify bugs that have evaded the design engineers. This in itself becomes a major factor in reduced productivity when utilizing a system that was supposed to increase, not decrease, productivity.

When discussing off the shelf micro computer hardware systems, world-wide parts and technical support are quickly identified as very important benefits. Contracts for repair parts are easily bid utilizing full and open competition because now there are third party vendors who can compete with the parent manufacturer to supply repair parts at the best possible price to the government. Given the mobility of a true blue water Navy such as our own, hardware technical support becomes an issue if problems that can not be solved by indigenous assets become dependent upon contractor support that must literally travel half-way around the world often times under adverse geo-political conditions. When assessing the desirability of a unique minicomputer hardware system versus an off the shelf micro computer hardware system one must also consider the limited user base the mini computer system has as opposed to the exceptionally large user base of the micro computer

system. With a user base that is orders of magnitude larger, inherent hardware design problems are typically well identified and corrected long before their installation in Naval ships. Additionally, as a direct result of having a world wide user base, the parent manufacturer is considerably more dynamic in providing technical support because that mechanism is already in place to support his other customers. In essence the parent company must survive in the market place on its reputation.

In terms of software development for the **Argos** system, the HyperCard programming environment coupled with its indigenous programming language HyperTalk have provided a means of easily developing complicated software systems. Simply put, HyperCard in effect places layers of abstraction between the programmer and the actual bit manipulation that goes into creating complex software systems. Now for the first time government agencies are no longer tied by an umbilical cord to software development consultants and their respective companies because complex database applications can be developed using in house assets. In an age of increased congressional scrutiny and obvious fiscal austerity it only makes good common sense to utilize in-house assets for areas such as software development, particularly when the Department of Defense continues to order mid-grade military officers to the Naval Postgraduate School to follow a course of study in any number of technical curriculums, one of which is Computer Science. Utilization of military officers holding advanced degrees in computer science for development of software systems similar to **Argos** rather than relying upon contractors would have significant cost reducing ramifications. Development time would also be sharply reduced because now the developer has a more intimate

knowledge of the end users of the system under development and a better feel for how this system should interface to the big picture.

Another advantage of using off the shelf technology for software system development is that by not having to completely develop a new system for each application, development costs even for contractors would be sharply reduced thereby passing real dollar savings on to the Department of Defense. As a prototype and when fully implemented **Argos** is designed to be completely modular. This will allow all new equipment procurement contracts to stipulate that the contractor must deliver an **Argos** application for the equipment being delivered to the military thereby eliminating the need for continued software development.

In a fully developed **Argos** system the capability will exist for allowing multiple ad hoc queries of the various non tactical data bases for each ship individually, or the Battle Group as a whole entity. The obvious advantages of developing similar HyperCard systems for the military in general easily outweigh the disadvantages. Specific areas that would support continued research in multimedia technology include but are not limited to the following:

- Transport **Argos** to MS DOS, UNIX or some other operating system.
- Development of a multimedia semantic data model (MSDM).
- Conceptual design of the Navy's data requirement using MSDM to fully support the paperless ship concept.
- Development of an **Argos** capability to share data files with an operational MICRO-OMMS system in Foxbase.
- Extension of **Argos** to fully develop one or more of the functional areas.
- Development of a complex query language interface capability.
- Conduct a feasibility study that would project the cost of a fully operational **Argos** system.
- Investigate the size requirements for different configurations of the **Argos** system.

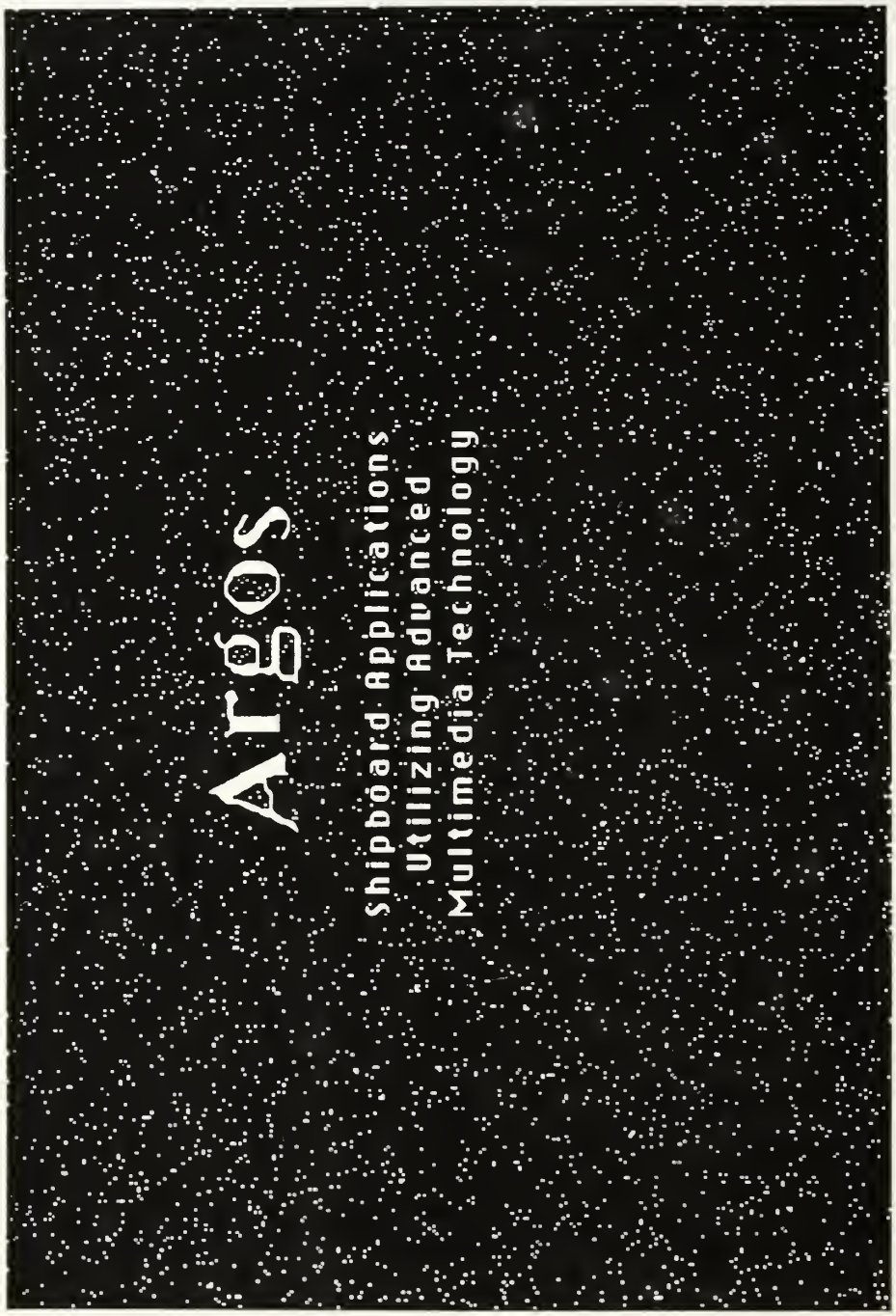
- Conduct a study that would determine implementation and cost factors associated with inclusion of ROEM technology.
- Develop **Argos** on a Next Computer in the Next Step Object-Oriented Operating System which is also available on other than Next computer hardware.

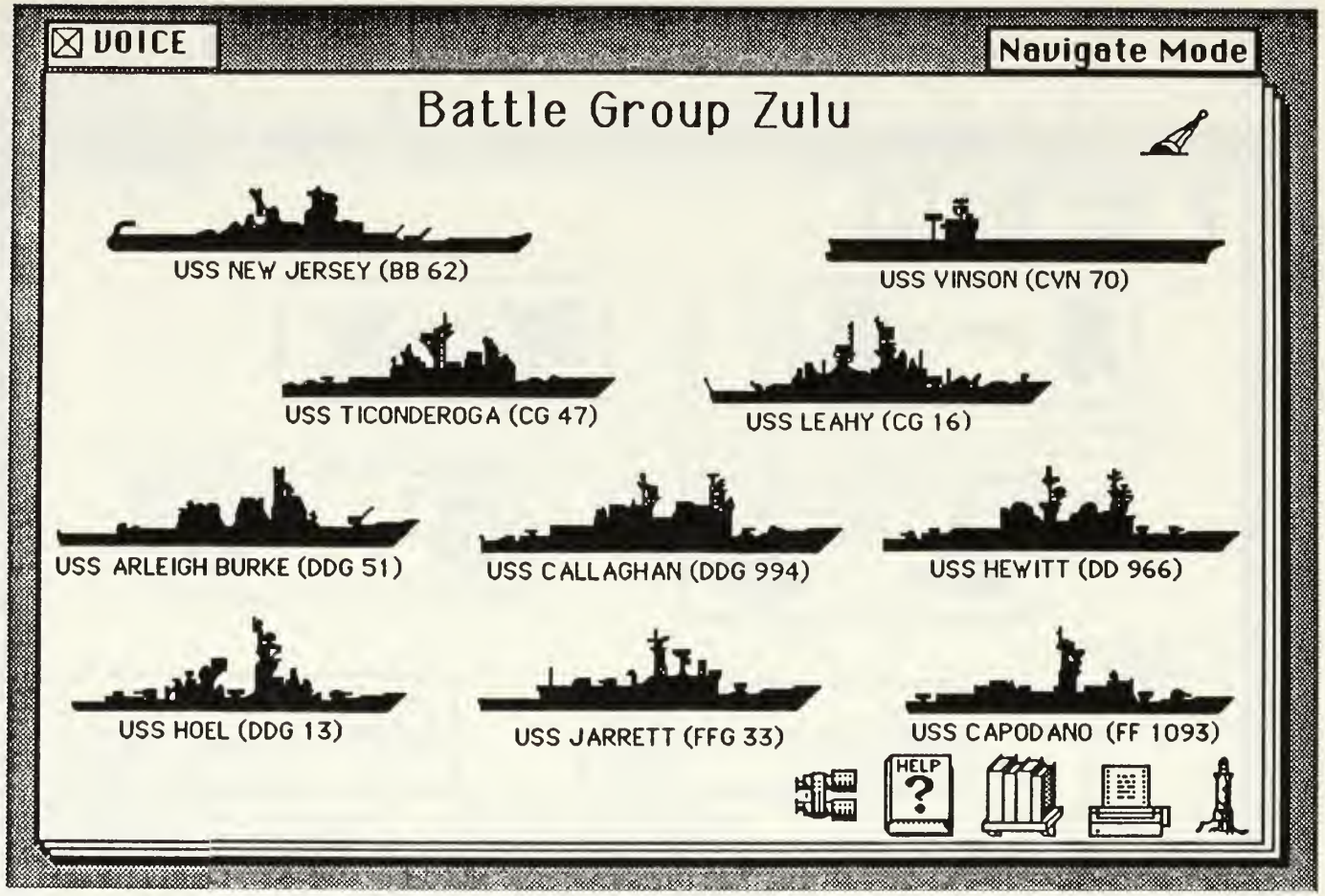
REFERENCES

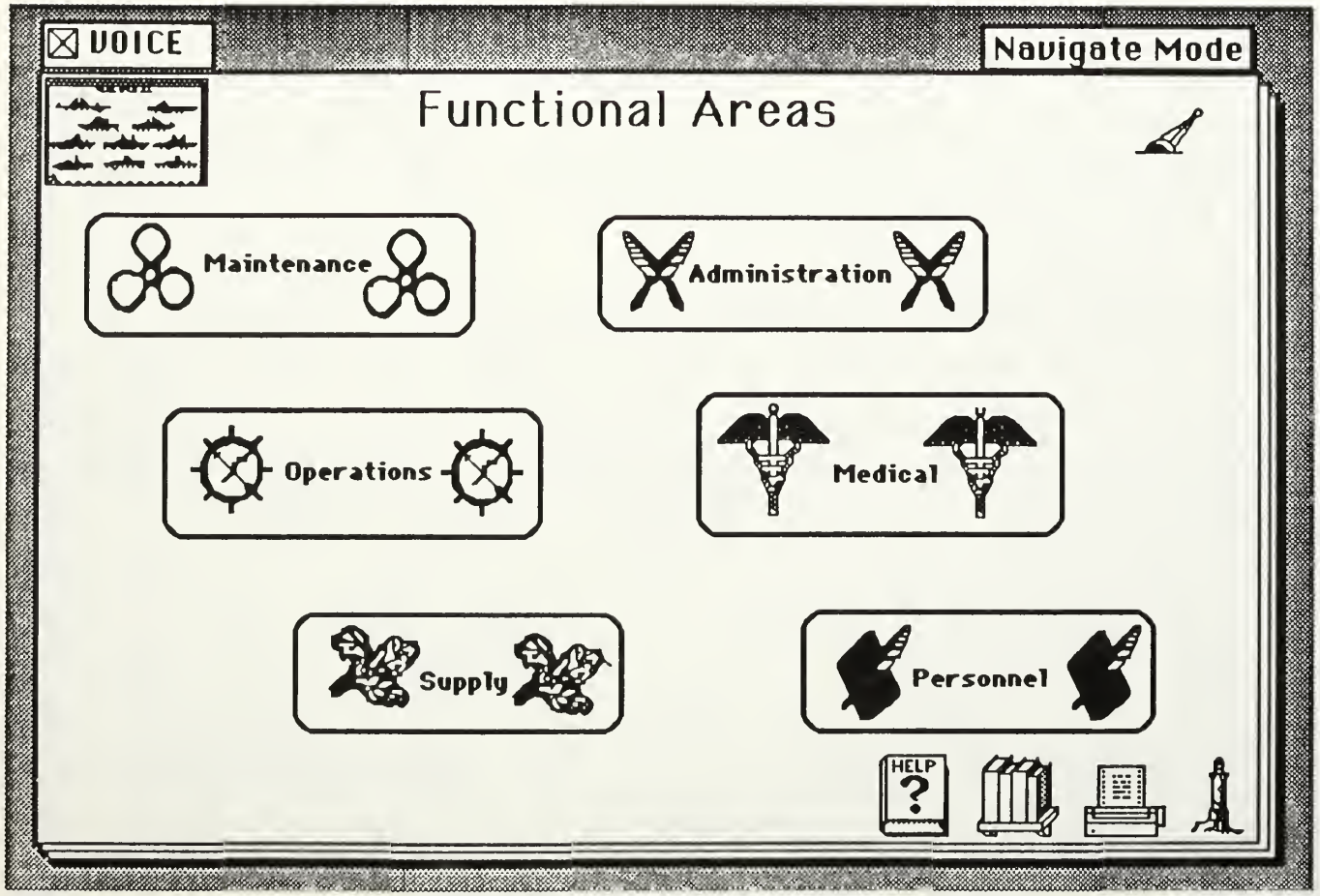
1. Ruff, D., LCdr, USN, from: "The Advent of the Paperless Ship," *Naval Engineers Journal*, July 1988, pp 157-159.

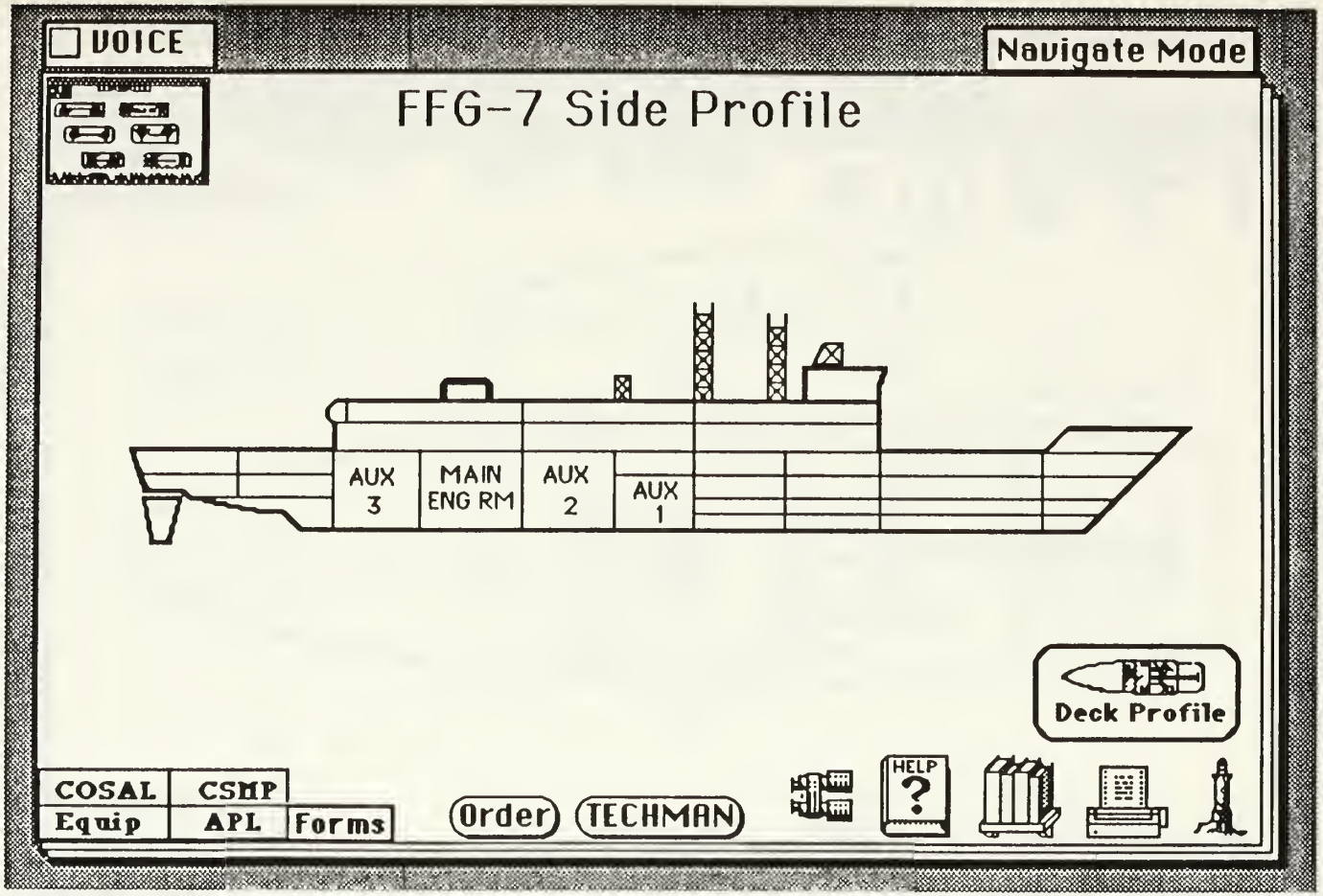
2. Goodman, D., *The Complete HyperCard™ Handbook*, Bantam Computer Books, 1987.

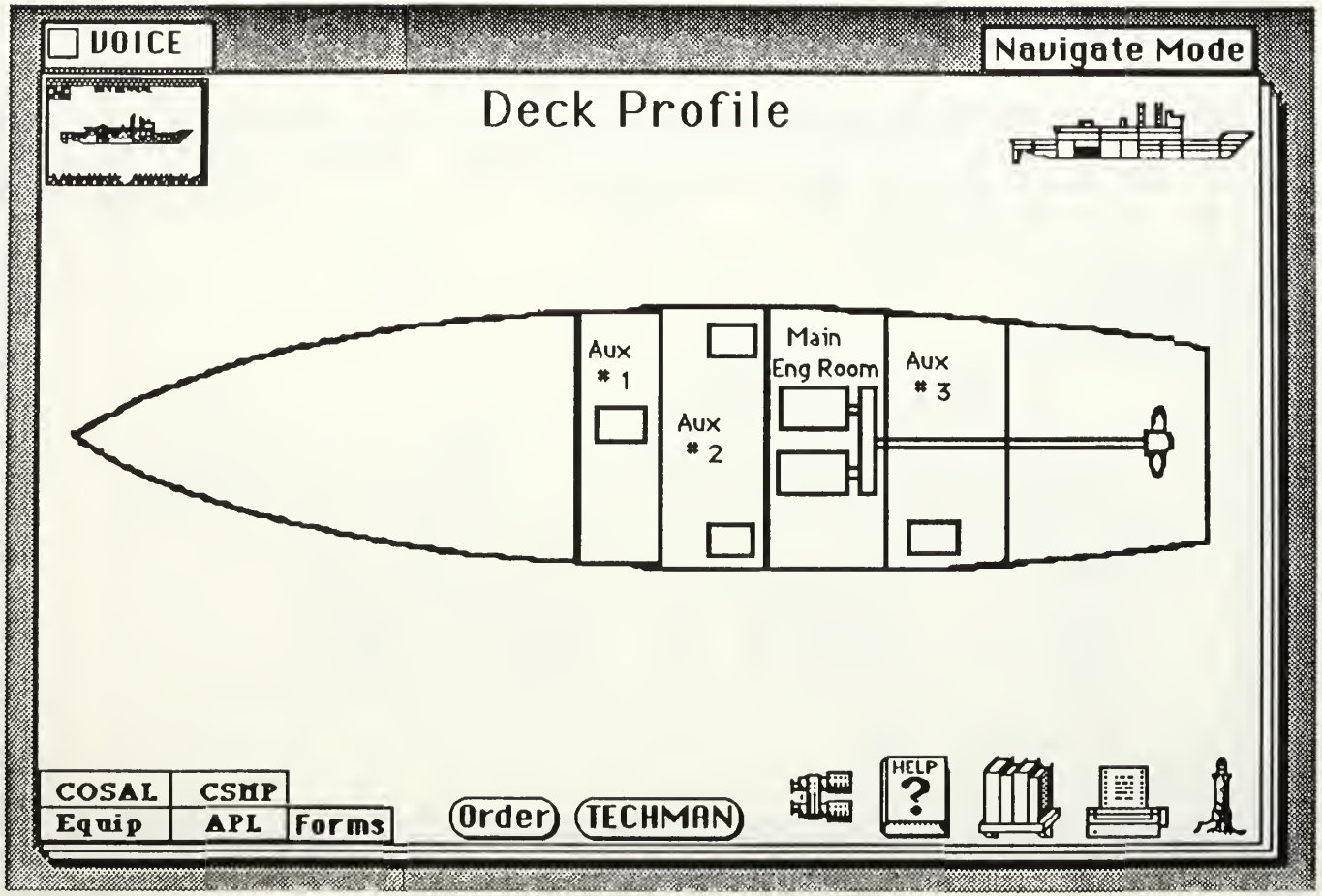
APPENDIX A
ARGOS STACK

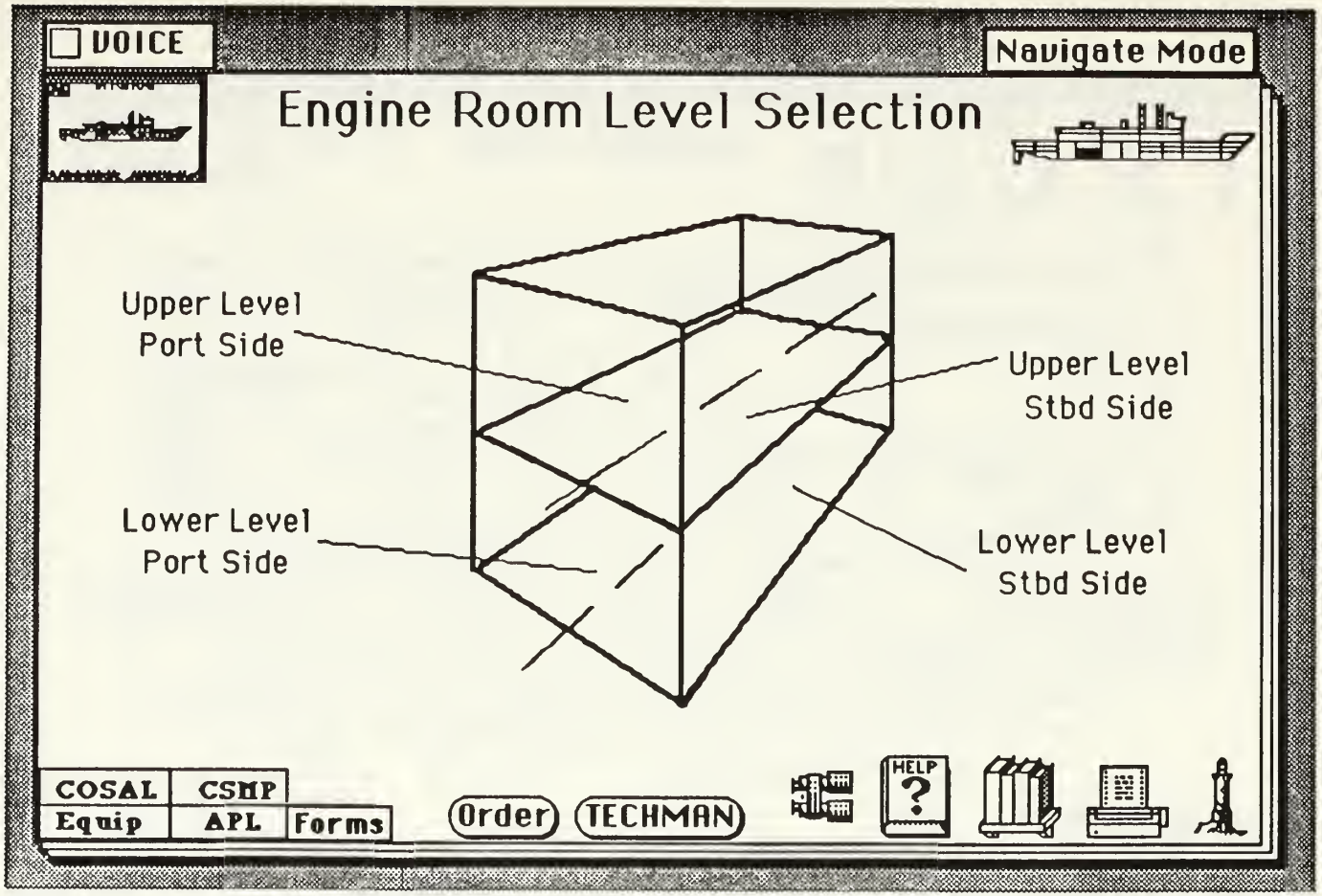


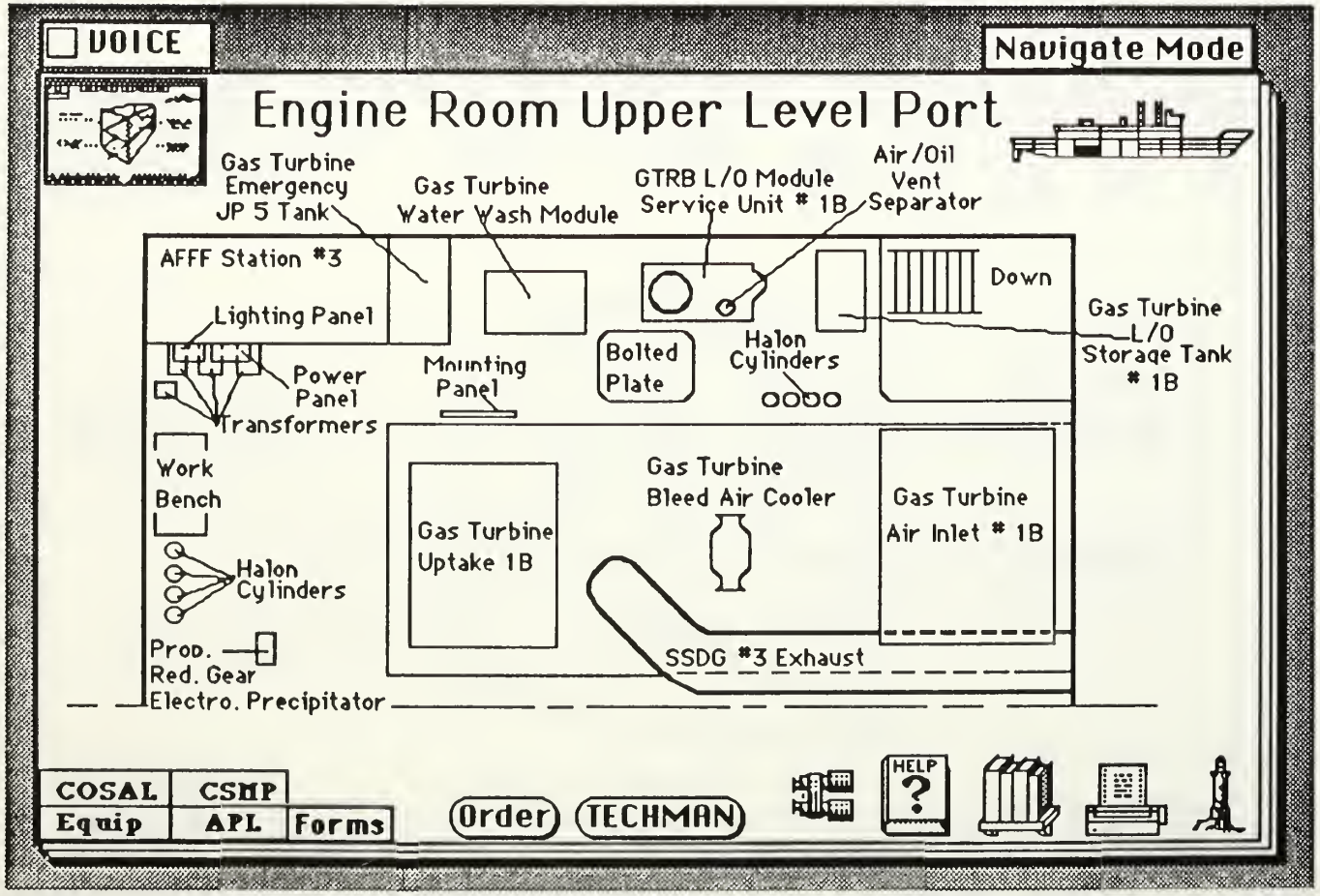

VOICE
Navigate Mode
Battle Group Zulu
USS NEW JERSEY (BB 62)
USS VINSON (CVN 70)
USS TICONDEROGA (CG 47)
USS LEAHY (CG 16)
USS ARLEIGH BURKE (DDG 51)
USS CALLAGHAN (DDG 994)
USS HEWITT (DD 966)
USS HOEL (DDG 13)
USS JARRETT (FFG 33)
USS CAPODANO (FF 1093)
HELP
?

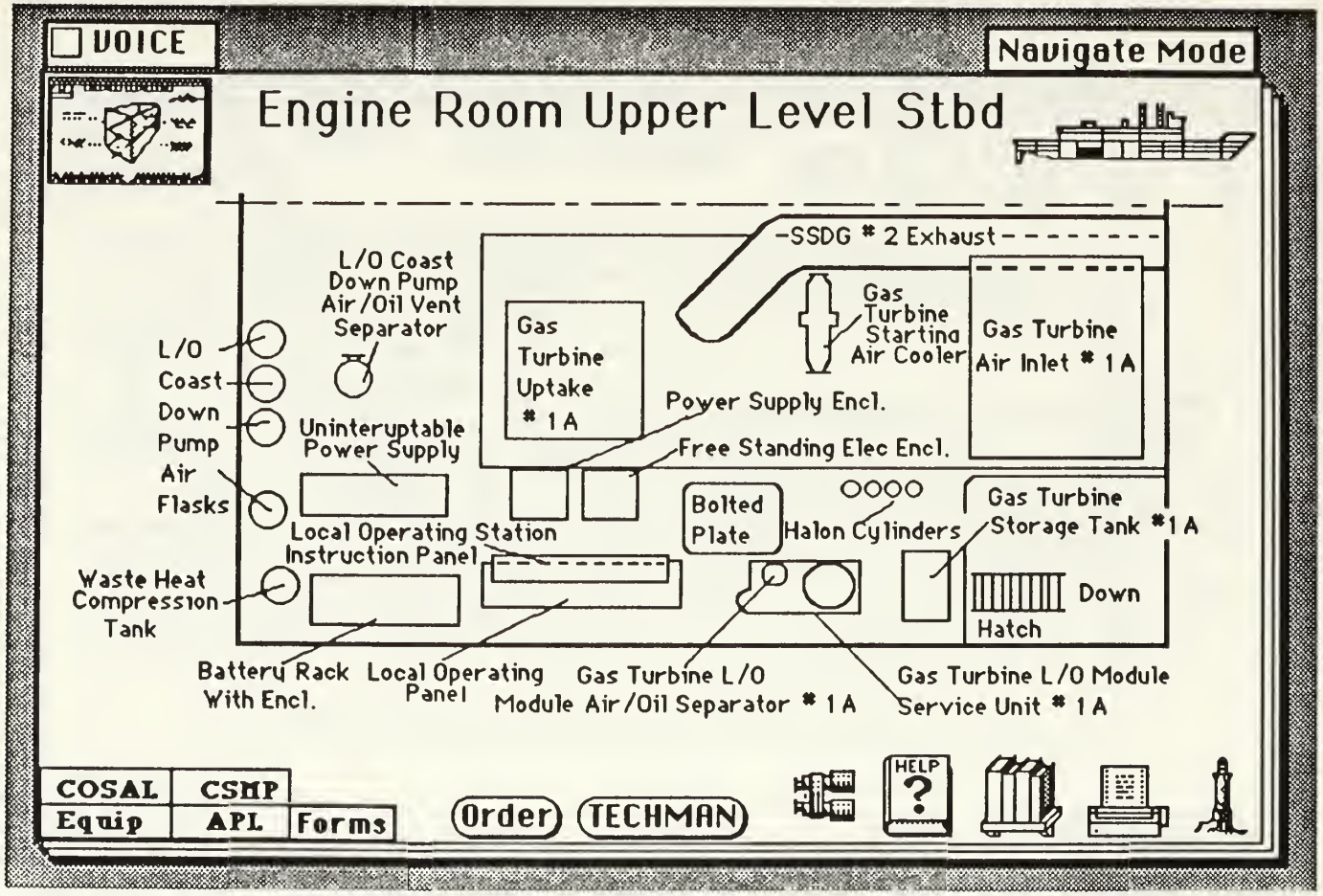

VOICE
Navigate Mode
Functional Areas
Maintenance
Administration
Operations
Medical
Supply
Personnel
HELP
?

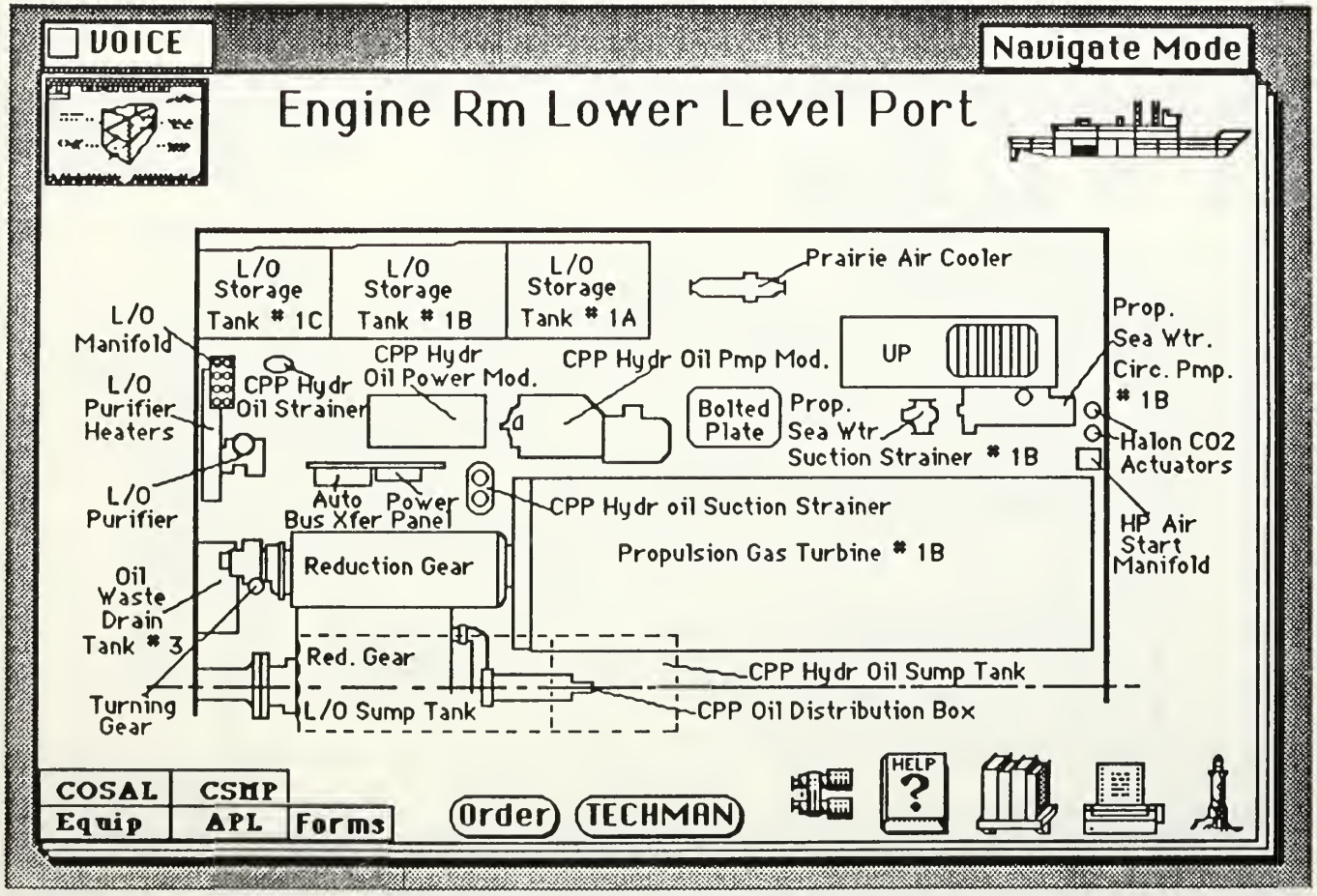

VOICE
Navigate Mode
FFG-7 Side Profile
AUX 3
MAIN ENG RM
AUX 2
AUX 1
Deck Profile
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

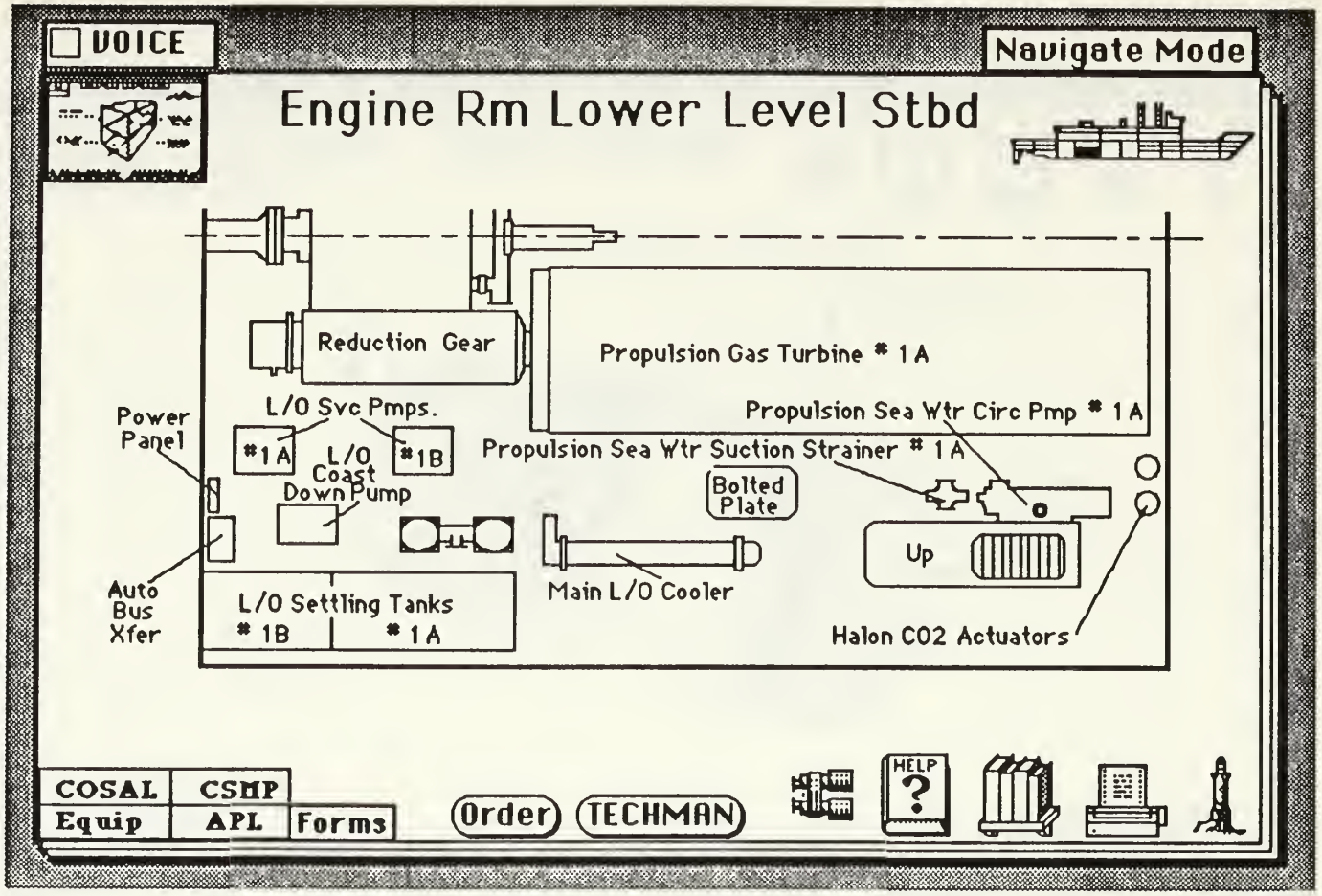

VOICE
Navigate Mode
Deck Profile
Aux
1
Aux
2
Main
Eng Room
Aux
3
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

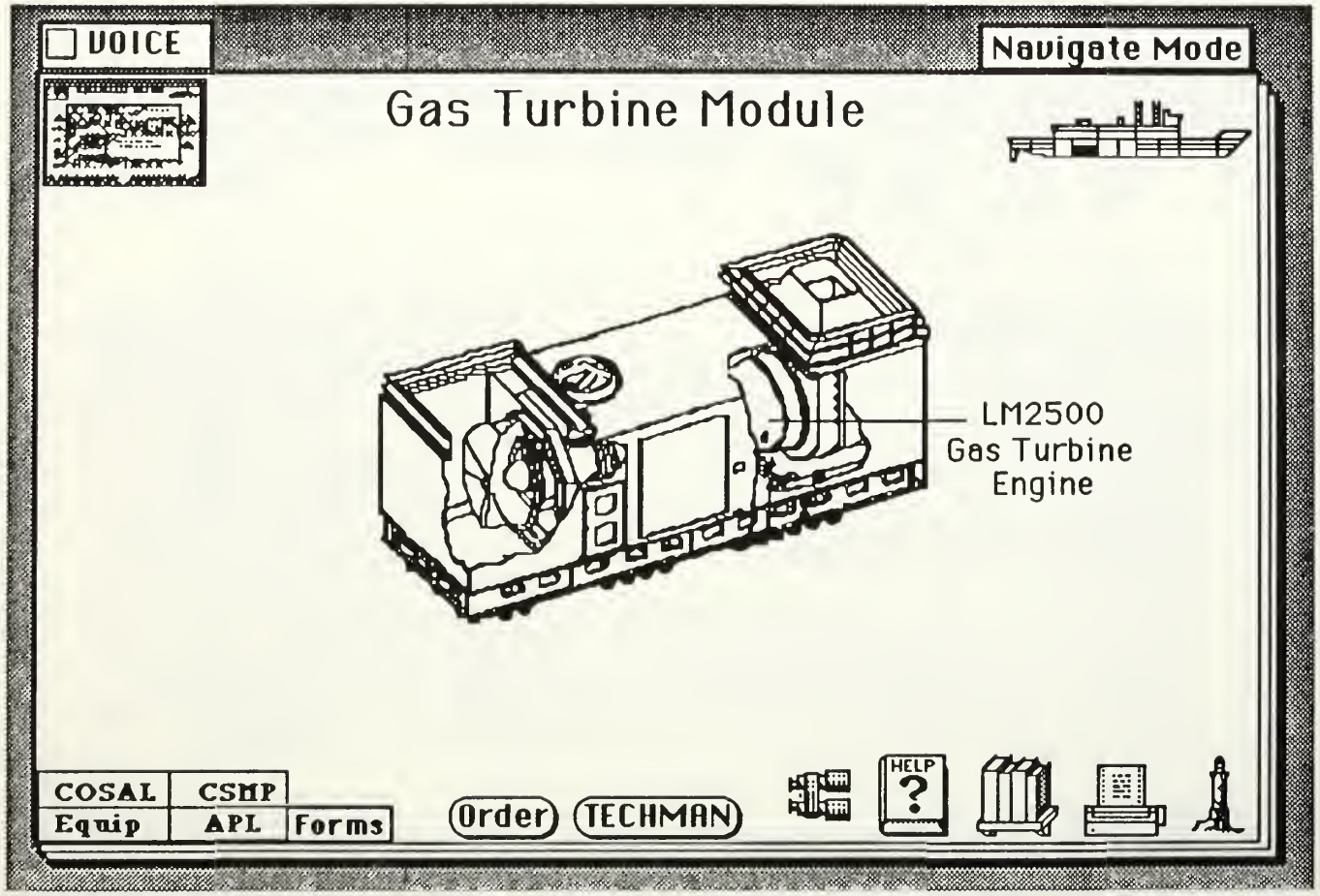

VOICE
Navigate Mode
Engine Room Level Selection
Upper Level
Port Side
Upper Level
Stbd Side
Lower Level
Port Side
Lower Level
Stbd Side
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

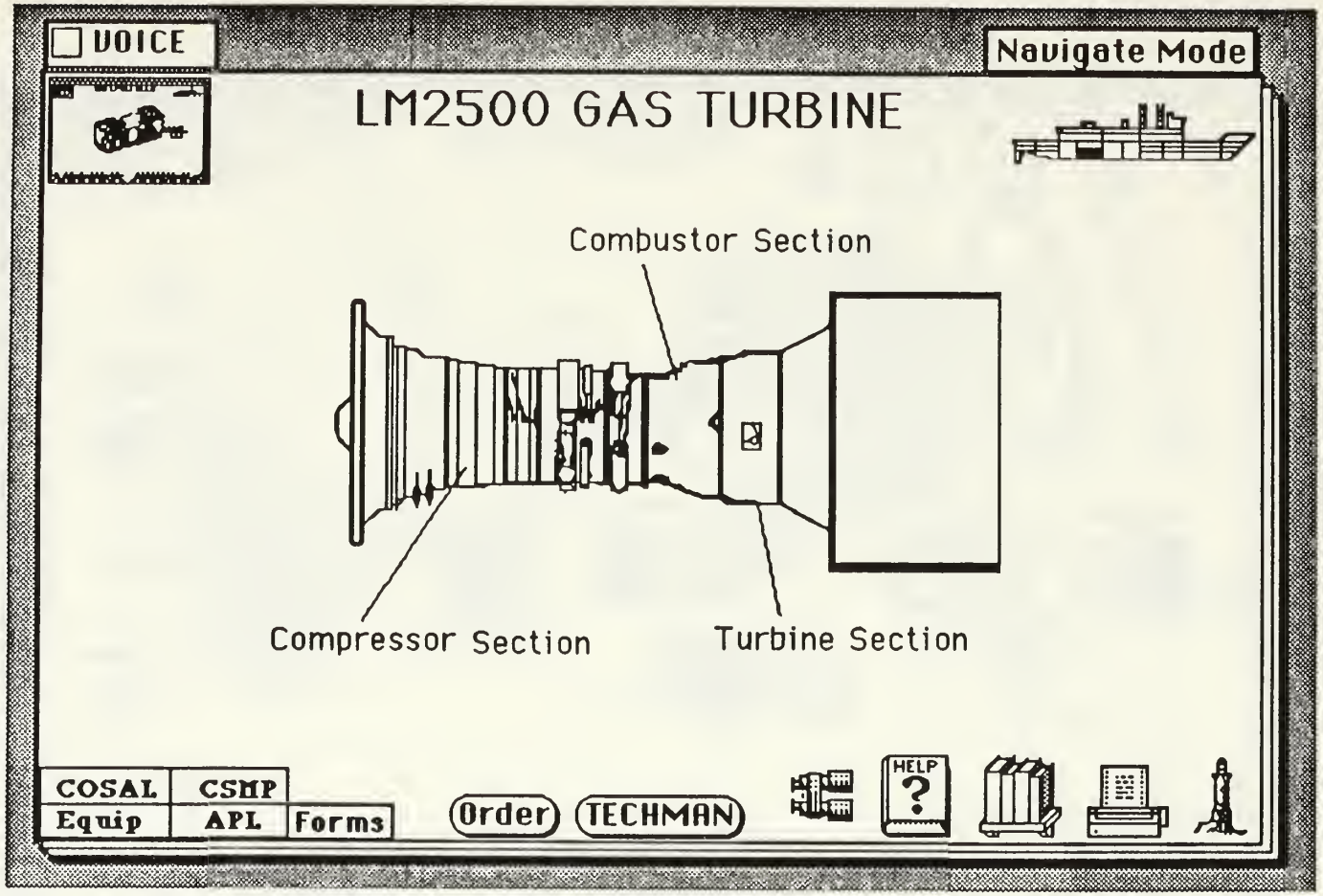

VOICE
Navigate Mode
Engine Room Upper Level Port
Gas Turbine Emergency JP 5 Tank
Gas Turbine Water Wash Module
GTRB L/O Module Service Unit # 1B
Air/Oil Vent Separator
AFFF Station #3
Lighting Panel
Power Panel
Transformers
Mounting Panel
Bolted Plate
Halon Cylinders
Down
Gas Turbine L/O Storage Tank # 1B
Work Bench
Halon Cylinders
Prop. Red. Gear
Electro. Precipitator
Gas Turbine Uptake 1B
Gas Turbine Bleed Air Cooler
Gas Turbine Air Inlet # 1B
SSDG #3 Exhaust
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

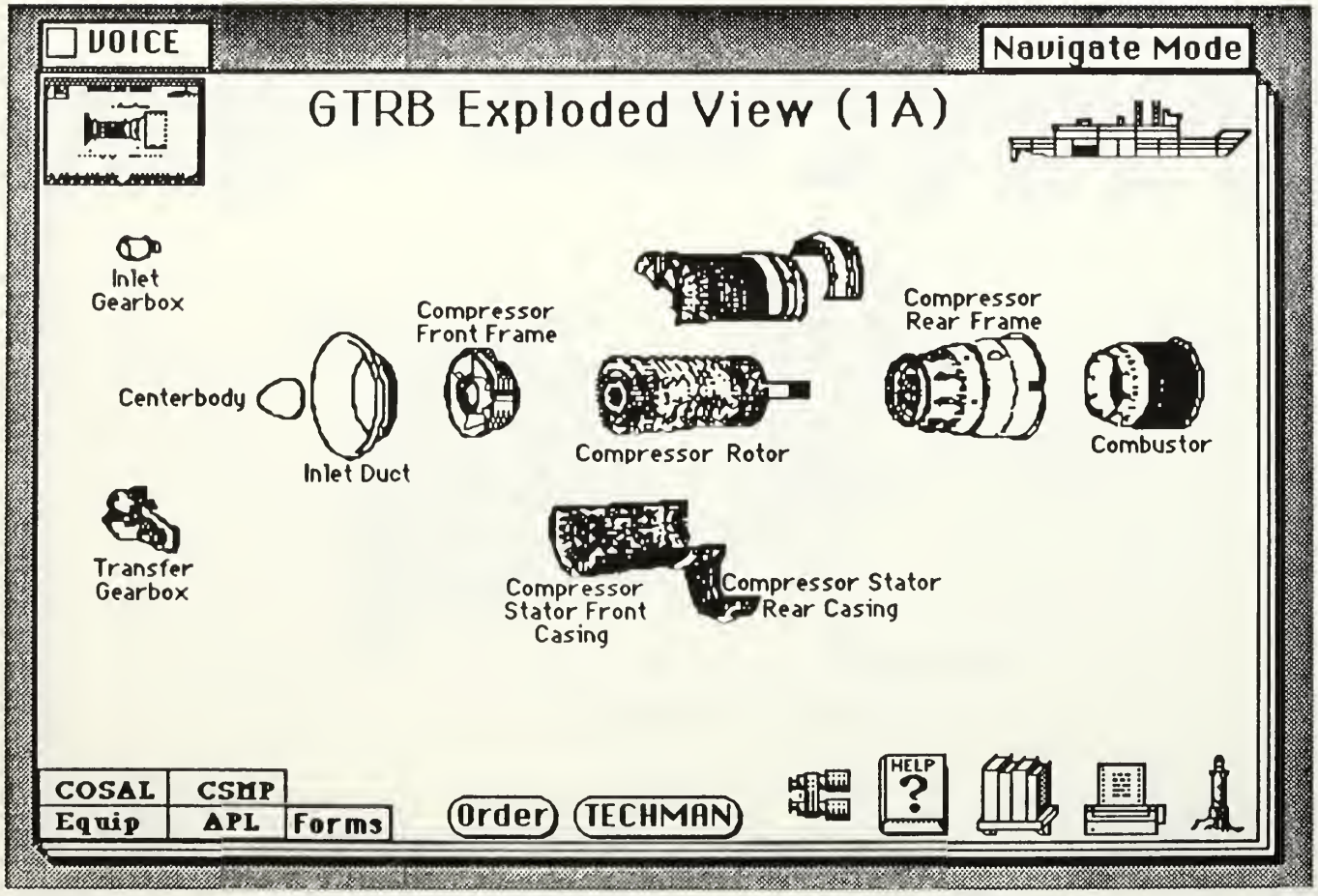

VOICE
Navigate Mode
Engine Room Upper Level Stbd
SSDG * 2 Exhaust
L/O Coast Down Pump Air/Oil Vent Separator
L/O Coast Down Pump Air Flasks
Uninteruptable Power Supply
Gas Turbine Uptake * 1A
Gas Turbine Starting Air Cooler
Gas Turbine Air Inlet * 1A
Power Supply Encl.
Free Standing Elec Encl.
Bolted Plate
Halon Cylinders
Gas Turbine Storage Tank *1A
Local Operating Station Instruction Panel
Waste Heat Compression Tank
Down Hatch
Battery Rack With Encl.
Local Operating Panel
Gas Turbine L/O Module Air/Oil Separator * 1A
Gas Turbine L/O Module Service Unit * 1A
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP

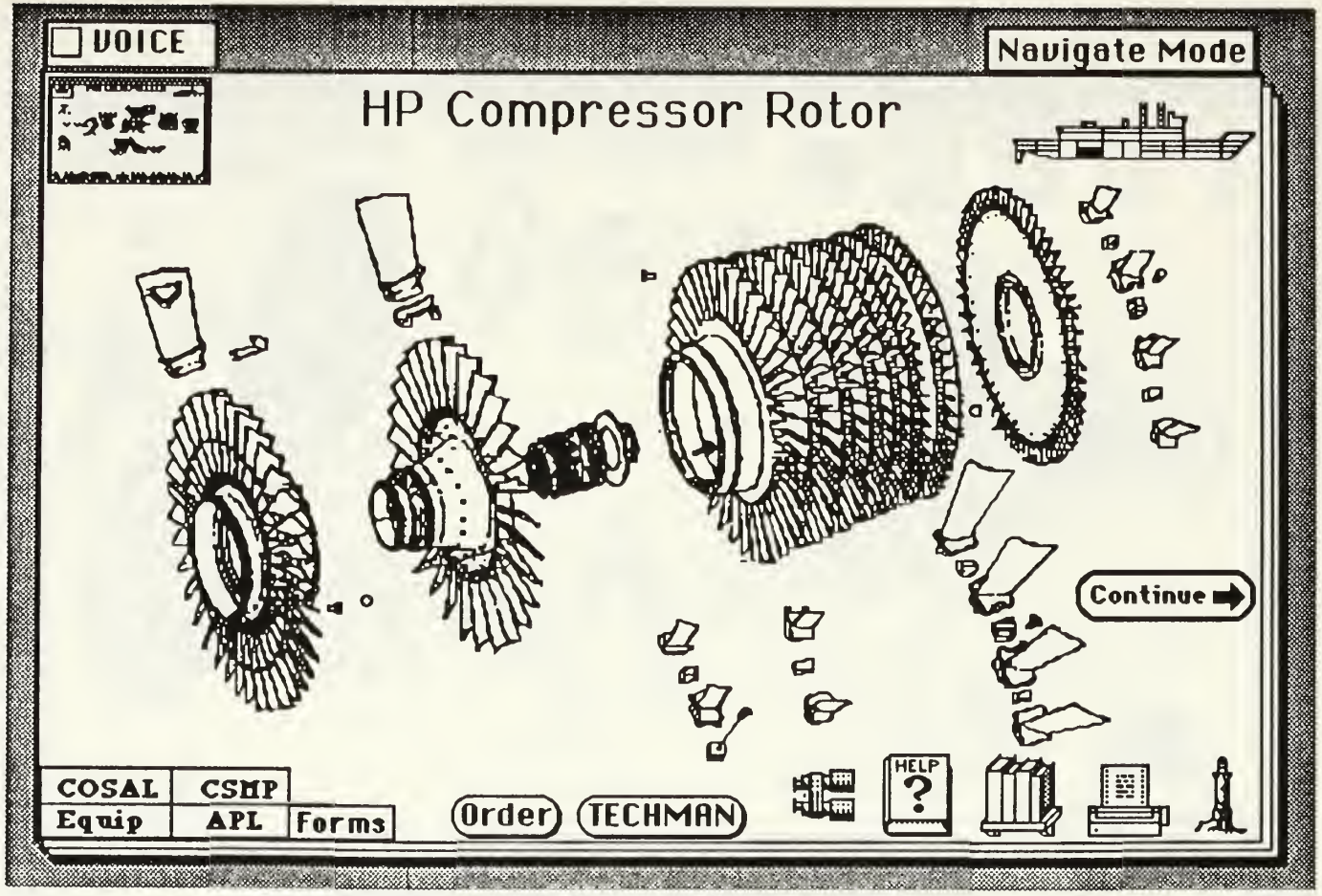

VOICE
Navigate Mode
Engine Rm Lower Level Port
L/O Storage Tank * 1C
L/O Storage Tank * 1B
L/O Storage Tank * 1A
Prairie Air Cooler
L/O Manifold
CPP Hydr Oil Strainer
CPP Hydr Oil Power Mod.
CPP Hydr Oil Pmp Mod.
UP
Prop. Sea Wtr. Circ. Pmp. * 1B
L/O Purifier Heaters
Bolted Plate
Prop. Sea Wtr Suction Strainer * 1B
Halon CO2 Actuators
L/O Purifier
Auto Power Bus Xfer Panel
CPP Hydr oil Suction Strainer
HP Air Start Manifold
Reduction Gear
Propulsion Gas Turbine * 1B
Oil Waste Drain Tank * 3
Red. Gear
CPP Hydr Oil Sump Tank
Turning Gear
L/O Sump Tank
CPP Oil Distribution Box
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP


VOICE
Navigate Mode
Engine Rm Lower Level Stbd
Reduction Gear
Propulsion Gas Turbine * 1A
Propulsion Sea Wtr Circ Pmp * 1A
Propulsion Sea Wtr Suction Strainer * 1A
L/O Svc Pmps.
*1A
*1B
L/O Coast Down Pump
Power Panel
Bolted Plate
Up
Auto Bus Xfer
L/O Settling Tanks
* 1B
* 1A
Main L/O Cooler
Halon CO2 Actuators
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP

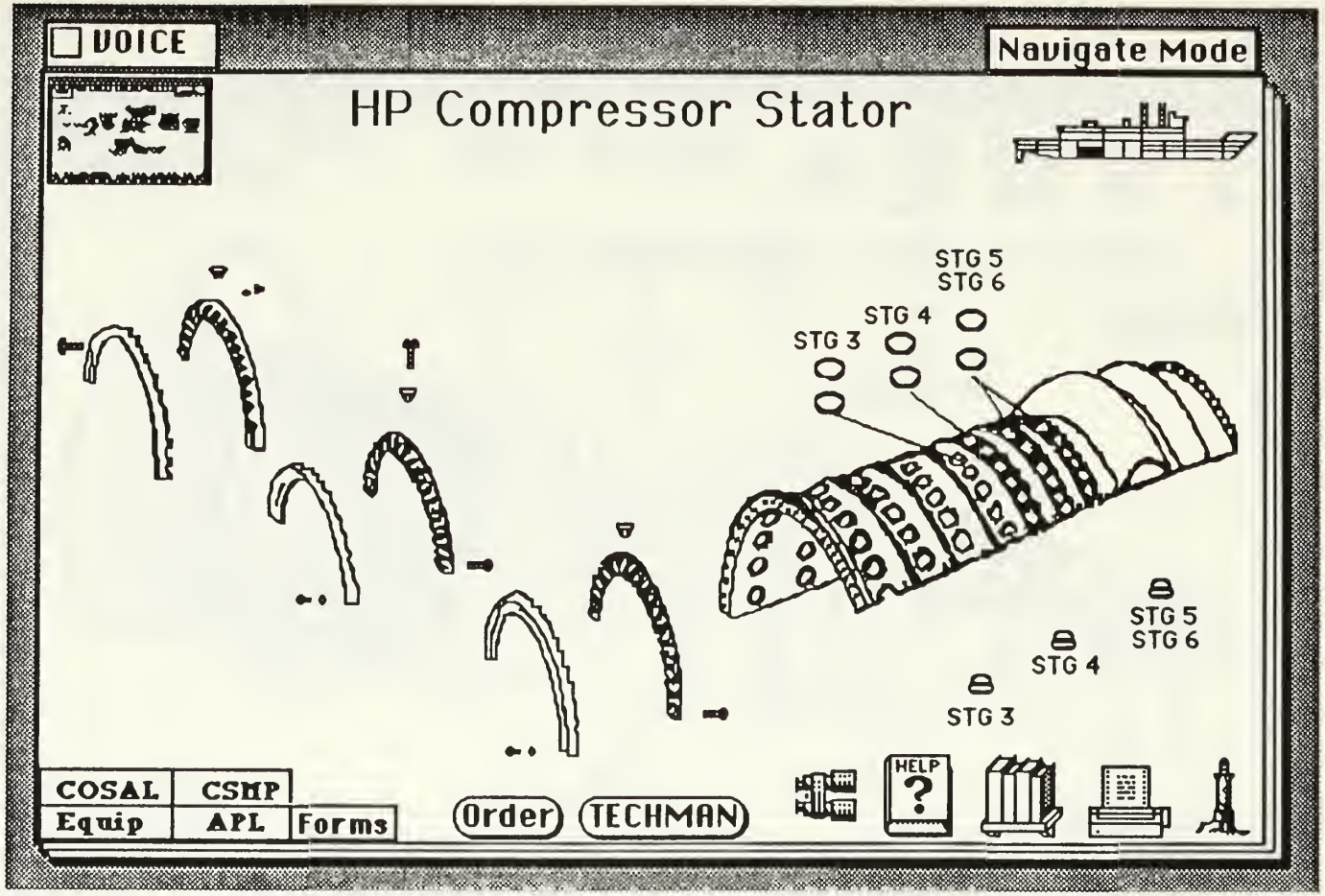

VOICE
Navigate Mode
Gas Turbine Module
LM2500
Gas Turbine
Engine
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

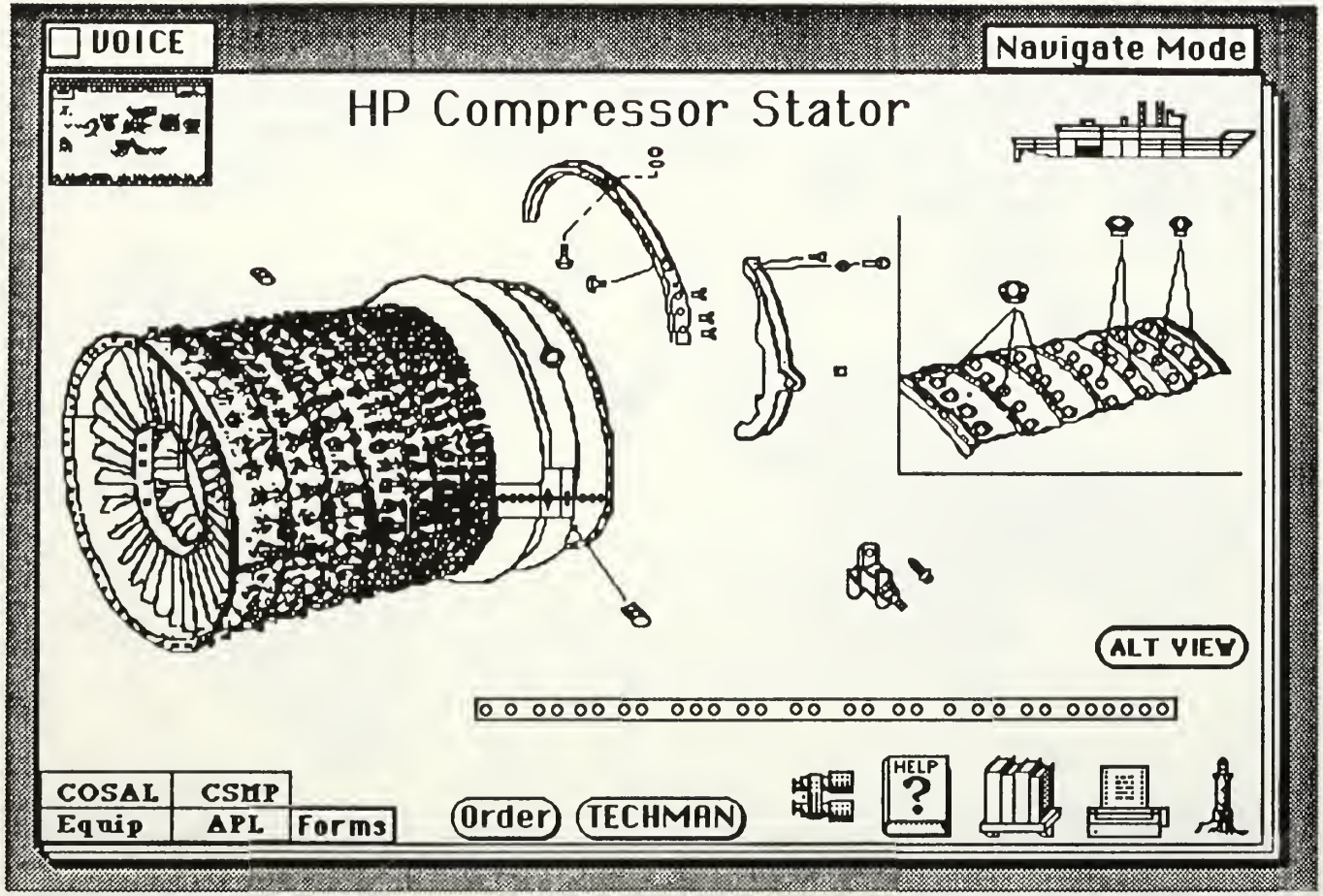

VOICE
Navigate Mode
LM2500 GAS TURBINE
Combustor Section
Compressor Section
Turbine Section
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

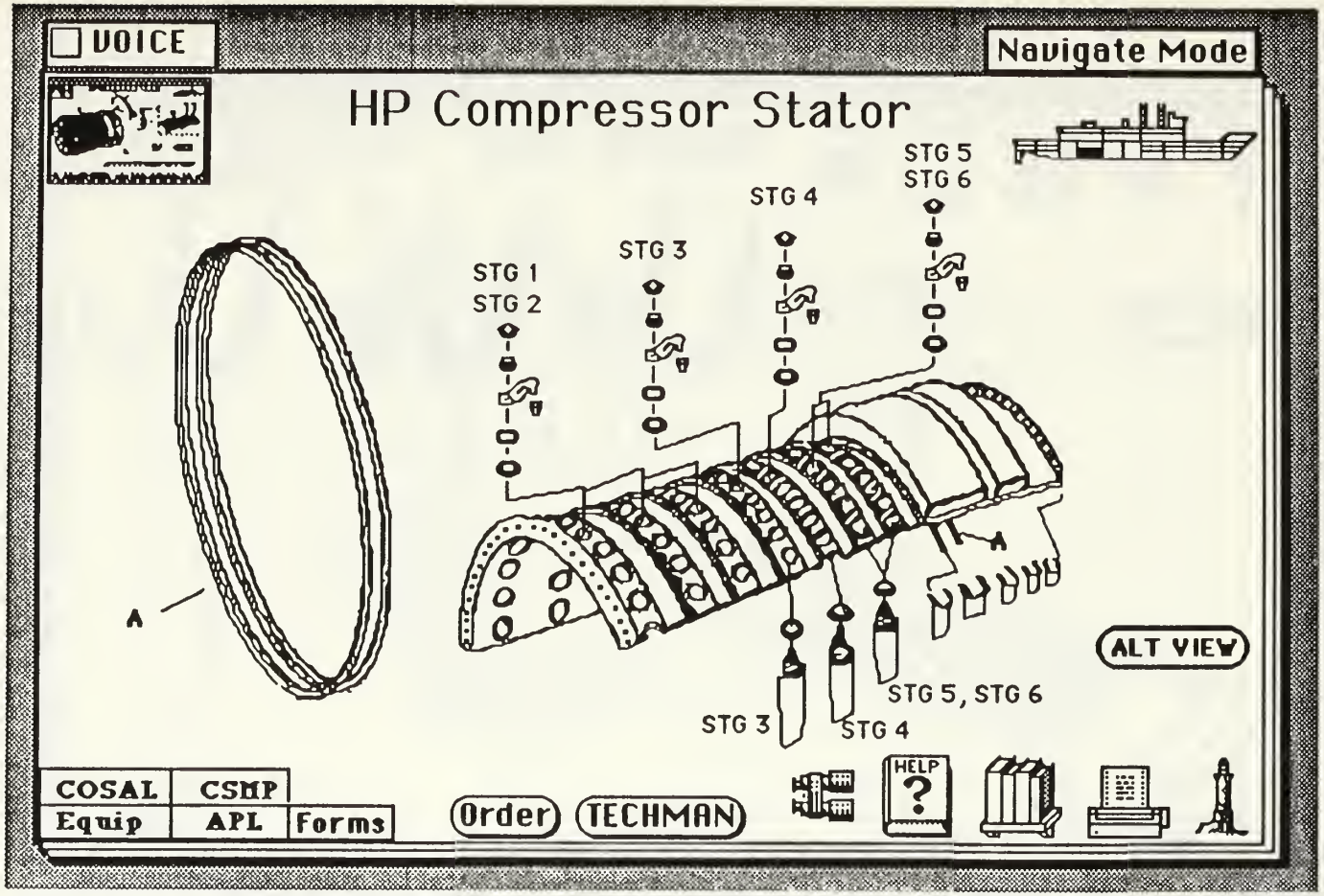

VOICE
Navigate Mode
GTRB Exploded View (1A)
Inlet Gearbox
Compressor Front Frame
Compressor Rear Frame
Centerbody
Inlet Duct
Compressor Rotor
Combustor
Transfer Gearbox
Compressor Stator Front Casing
Compressor Stator Rear Casing
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

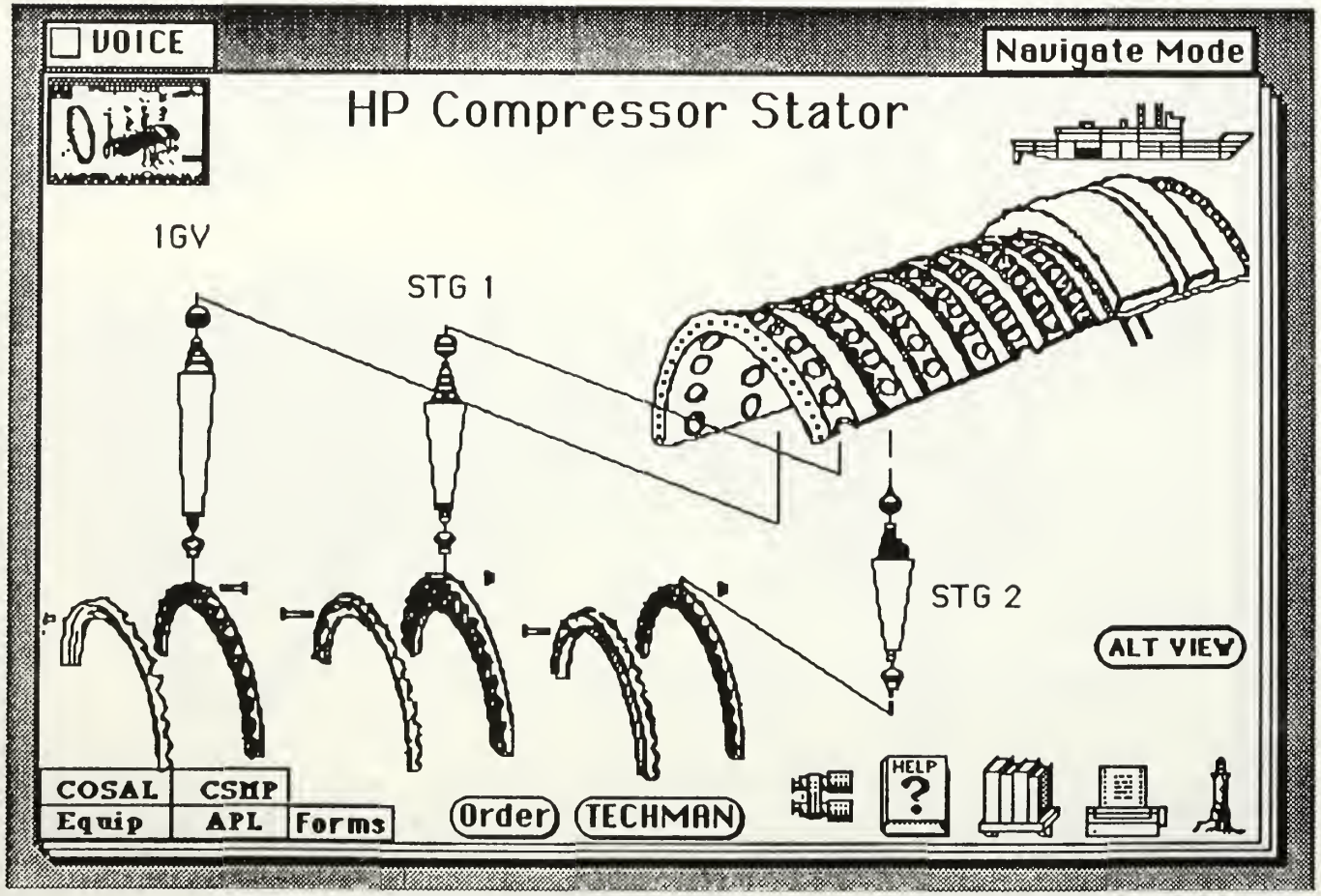

VOICE
Navigate Mode
HP Compressor Rotor
Continue
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

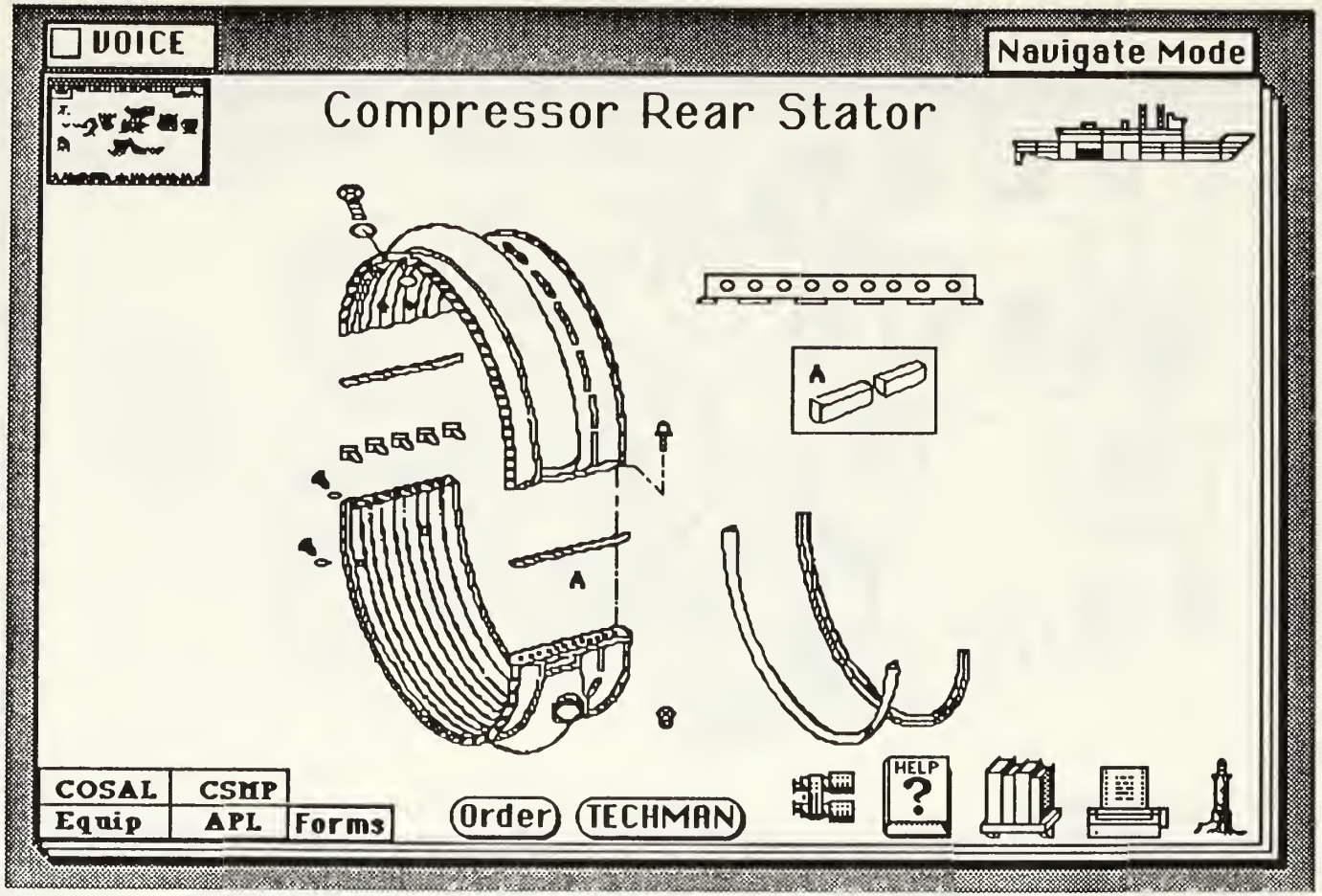
VOICE
Navigate Mode
HP Compressor Rotor
HELP
?
COSAL
CSHP
Equip
APL
Forms
Order
TECHMAN

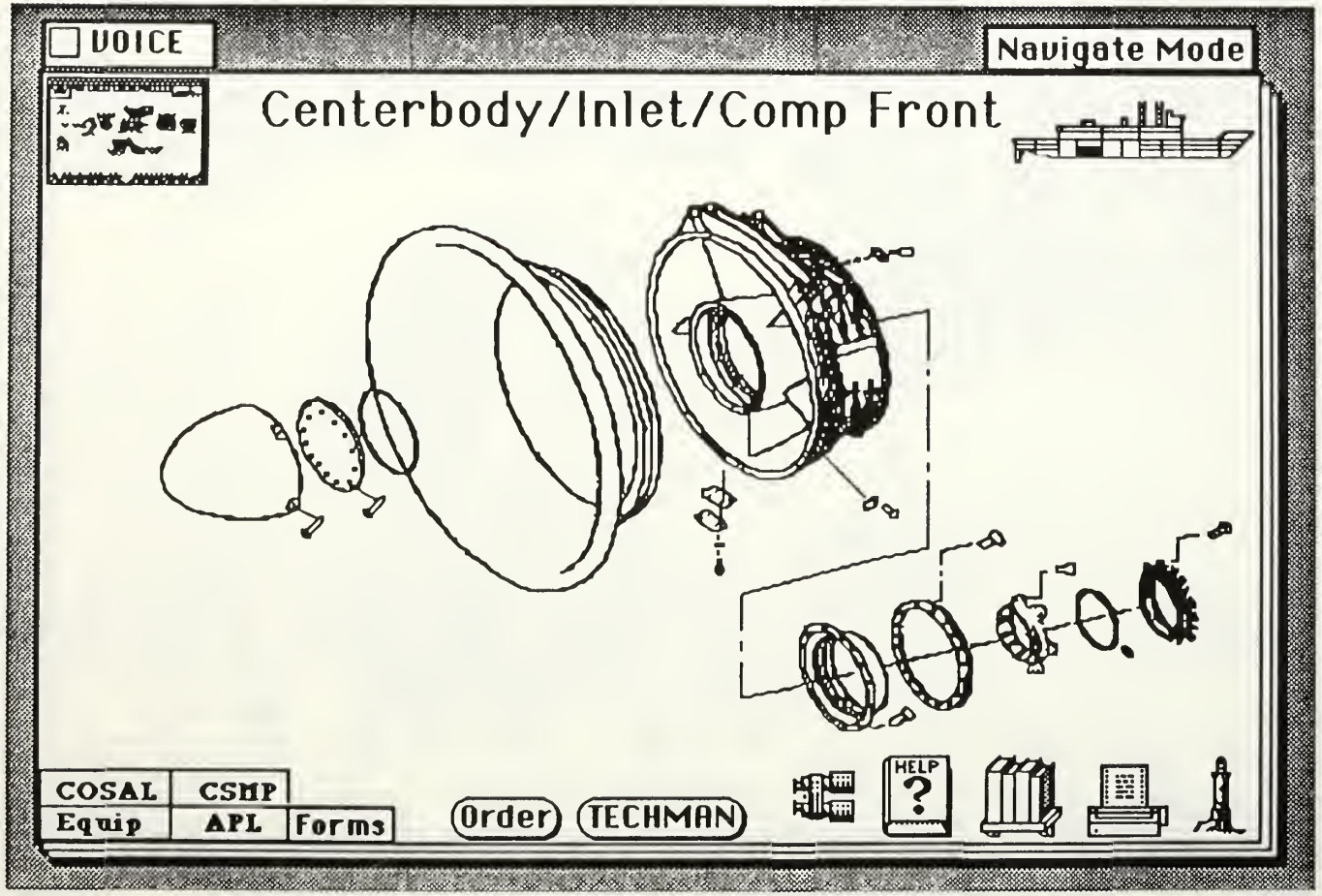

VOICE
Navigate Mode
HP Compressor Stator
STG 5
STG 6
STG 4
STG 3
STG 5
STG 6
STG 4
STG 3
HELP
?
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN

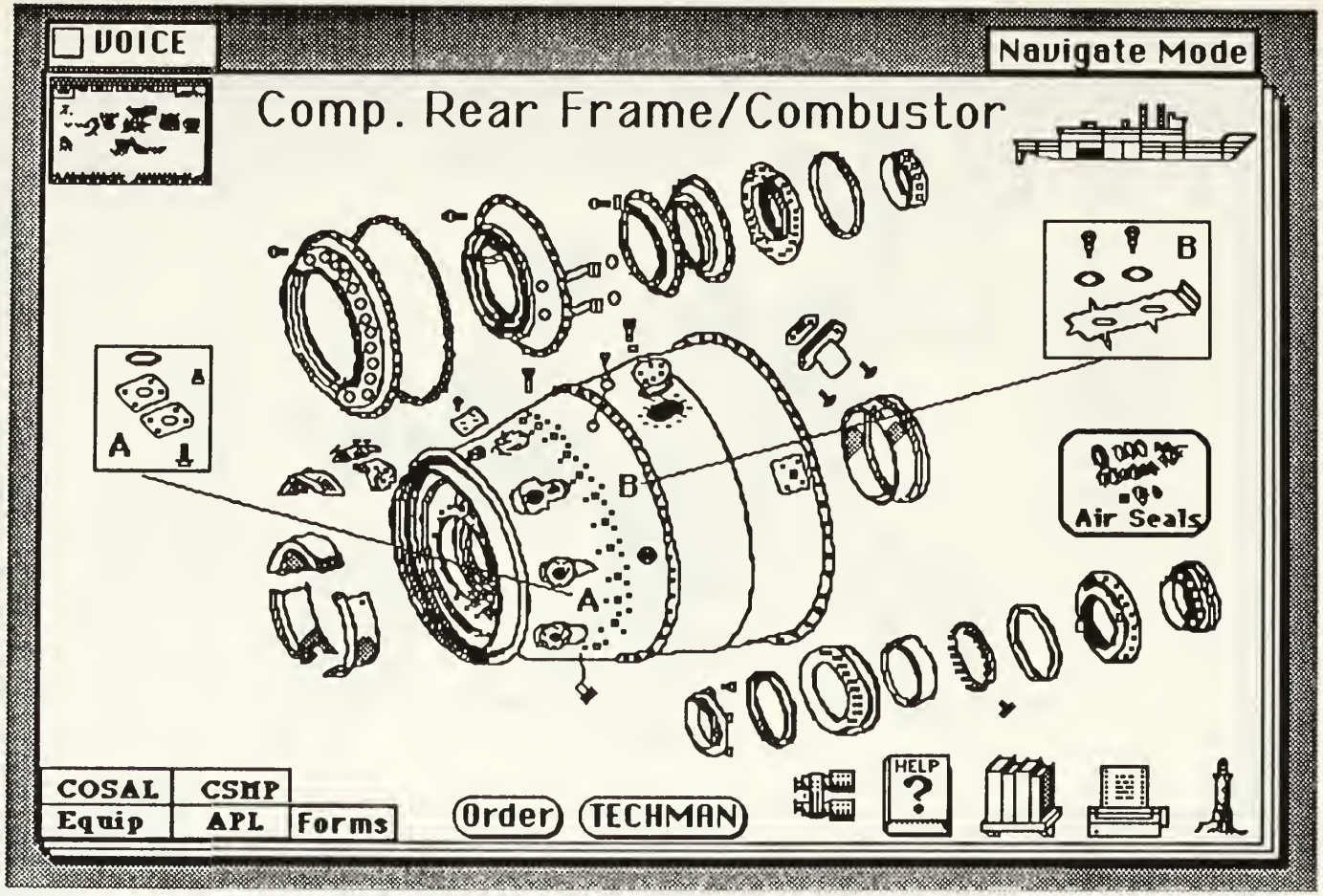

VOICE
Navigate Mode
HP Compressor Stator
ALT VIEW
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

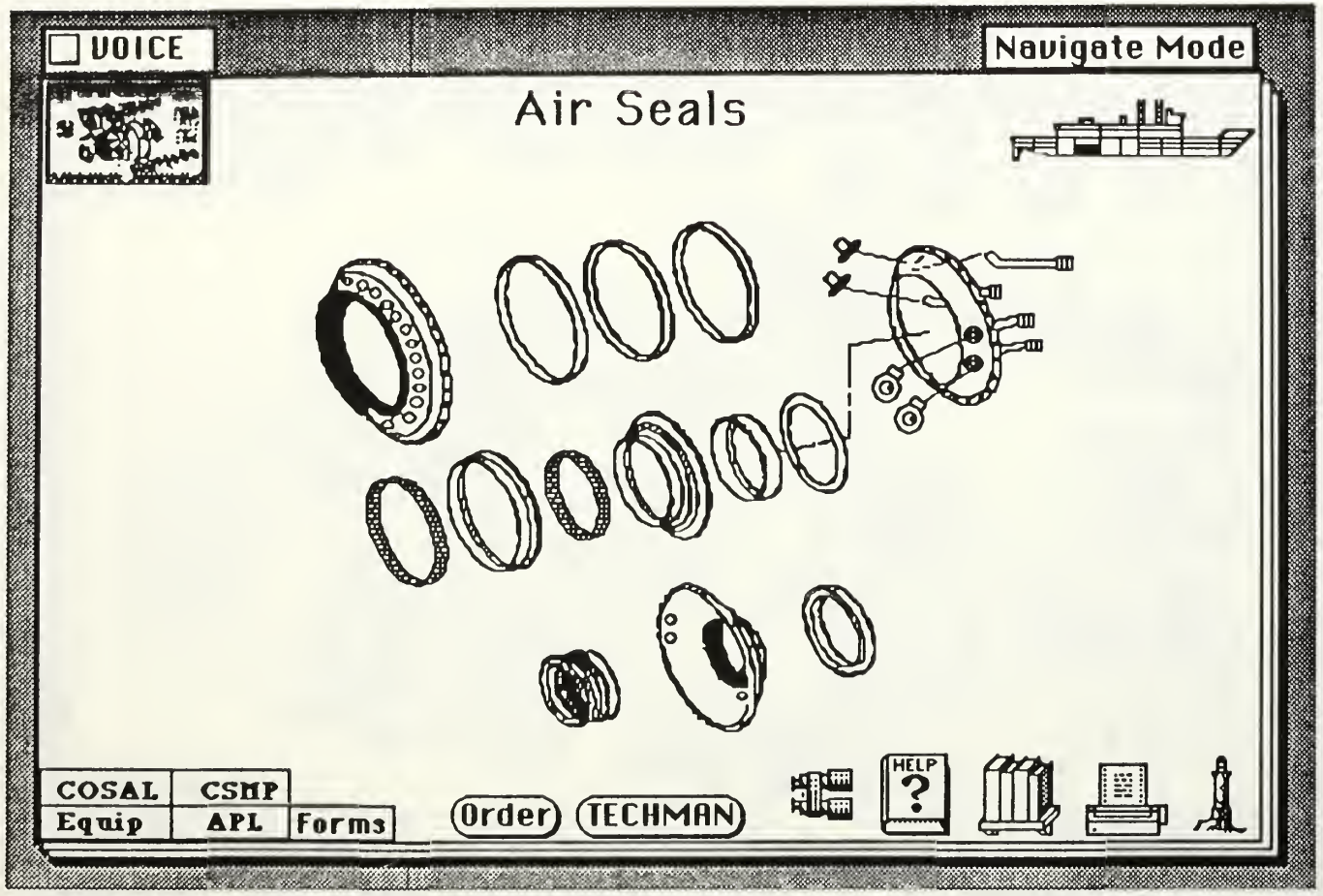

VOICE
Navigate Mode
HP Compressor Stator
STG 1
STG 2
STG 3
STG 4
STG 5
STG 6
A
A
ALT VIEW
STG 3
STG 4
STG 5, STG 6
HELP
?
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN

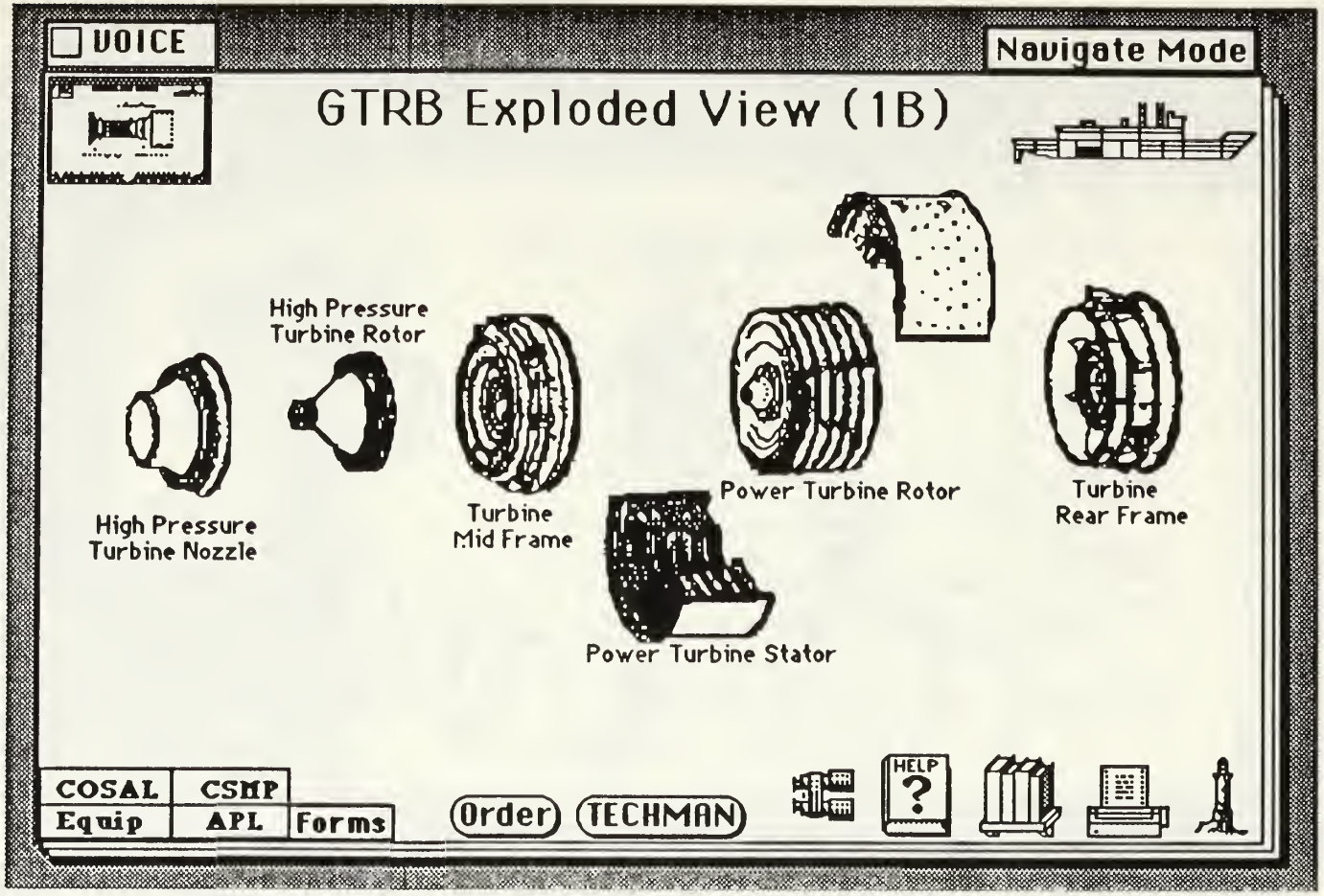

VOICE
Navigate Mode
HP Compressor Stator
IGV
STG 1
STG 2
ALT VIEW
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

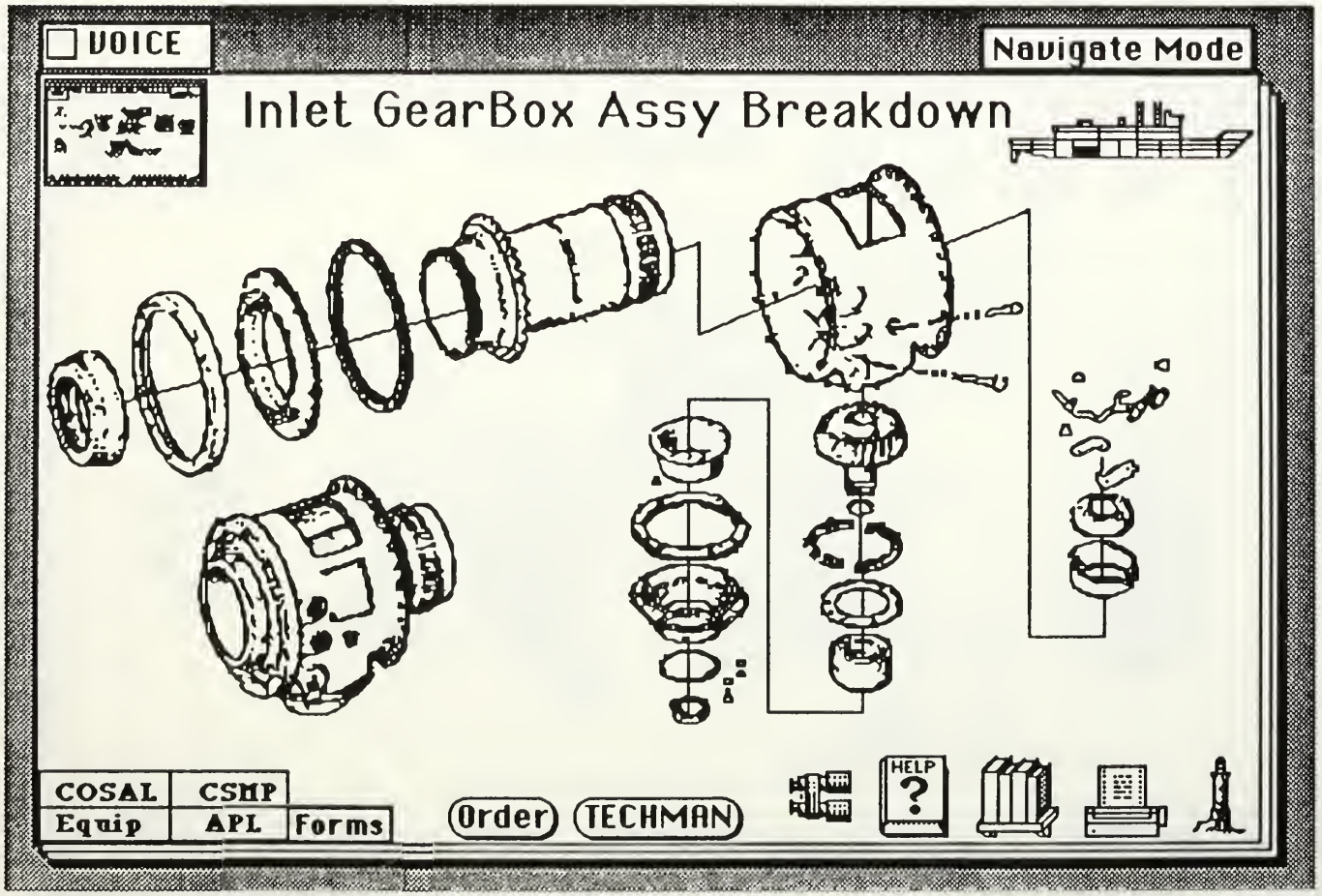

VOICE
Navigate Mode
Compressor Rear Stator
A
A
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

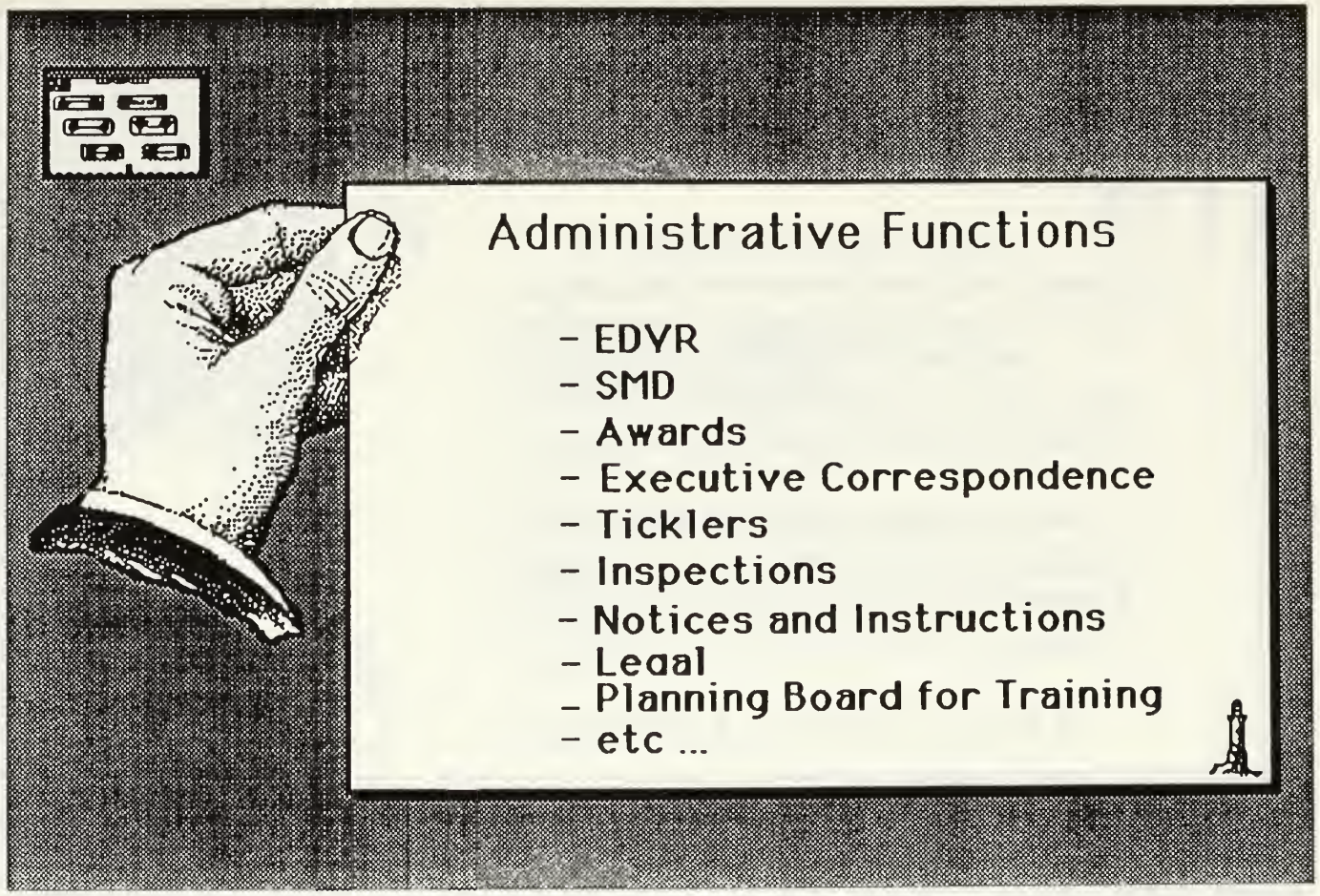

VOICE
Navigate Mode
Centerbody/Inlet/Comp Front
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

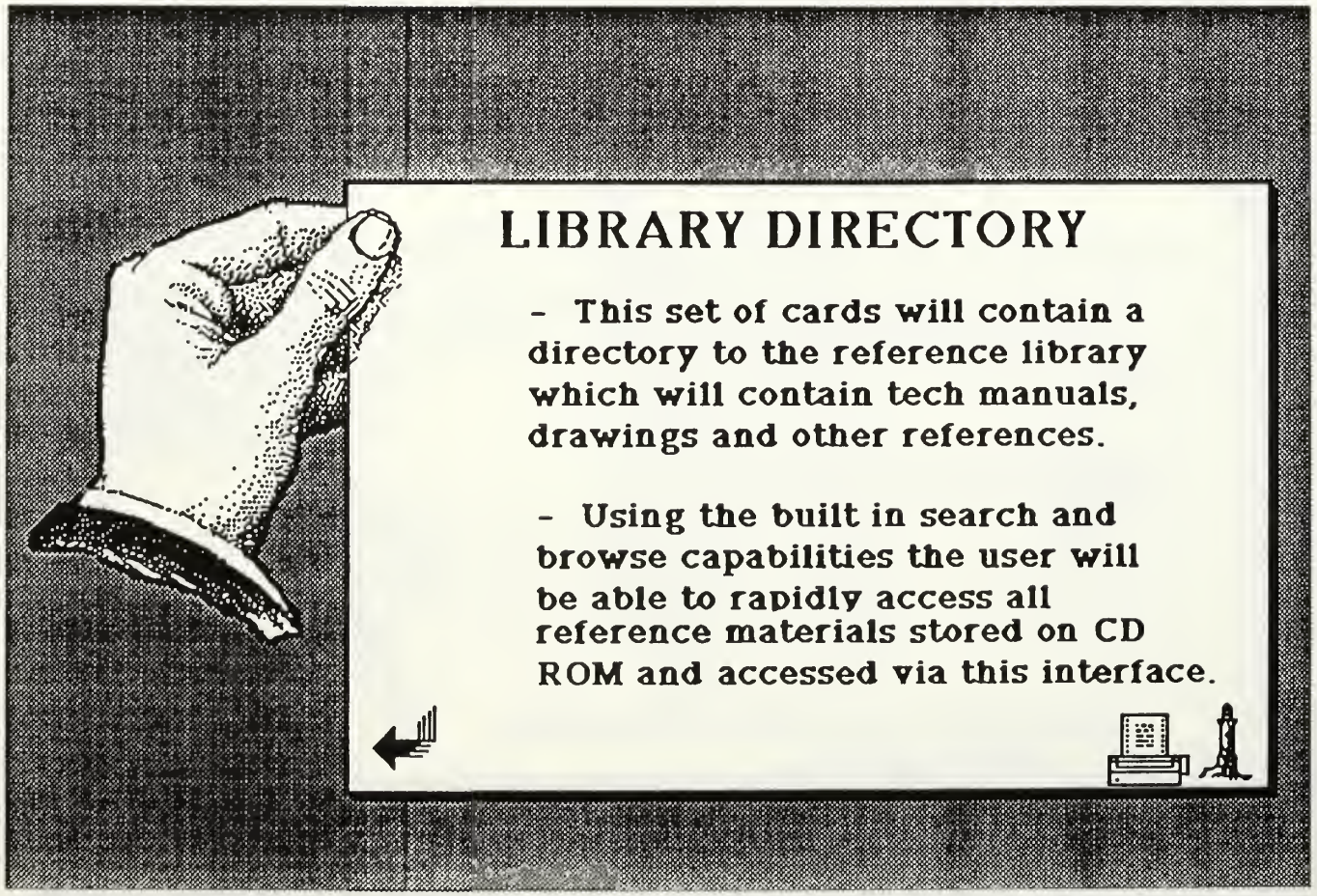

VOICE
Navigate Mode
Comp. Rear Frame/Combustor
A
B
Air Seals
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

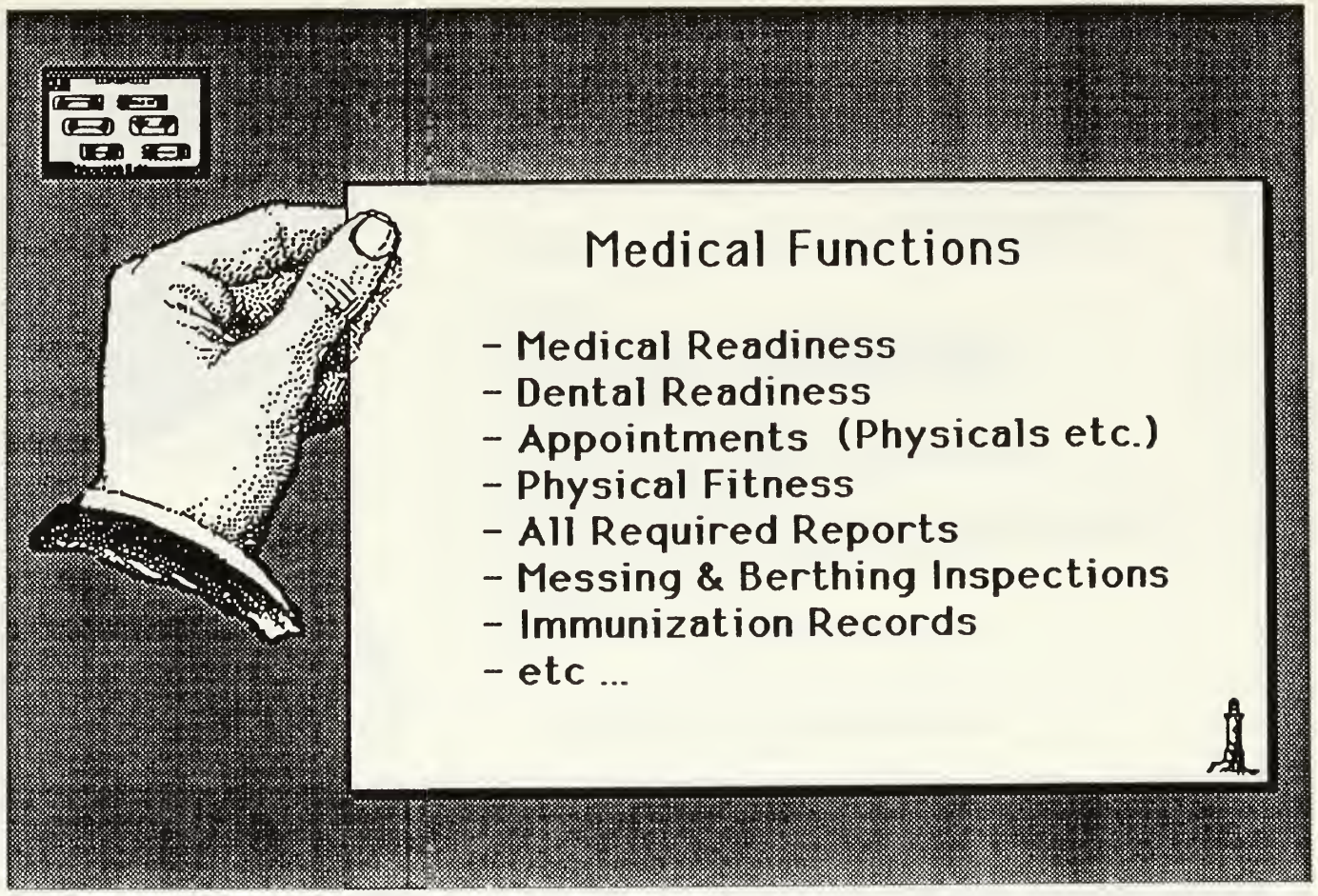

VOICE
Navigate Mode
Air Seals
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?


VOICE
Navigate Mode
GTRB Exploded View (1B)
High Pressure
Turbine Rotor
High Pressure
Turbine Nozzle
Turbine
Mid Frame
Power Turbine Rotor
Power Turbine Stator
Turbine
Rear Frame
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

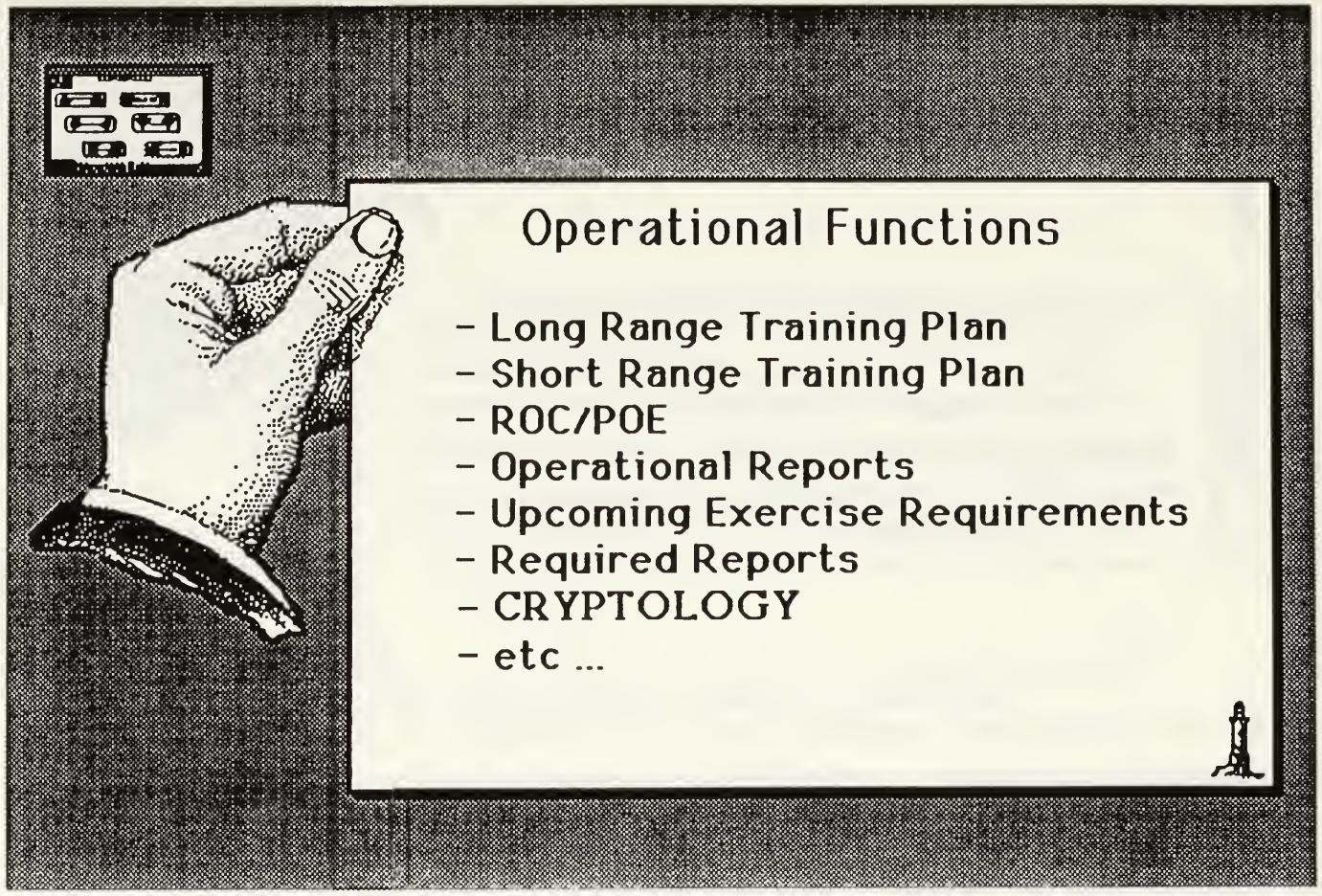

VOICE
Navigate Mode
Inlet GearBox Assy Breakdown
COSAL
CSMP
Equip
APL
Forms
Order
TECHMAN
HELP
?

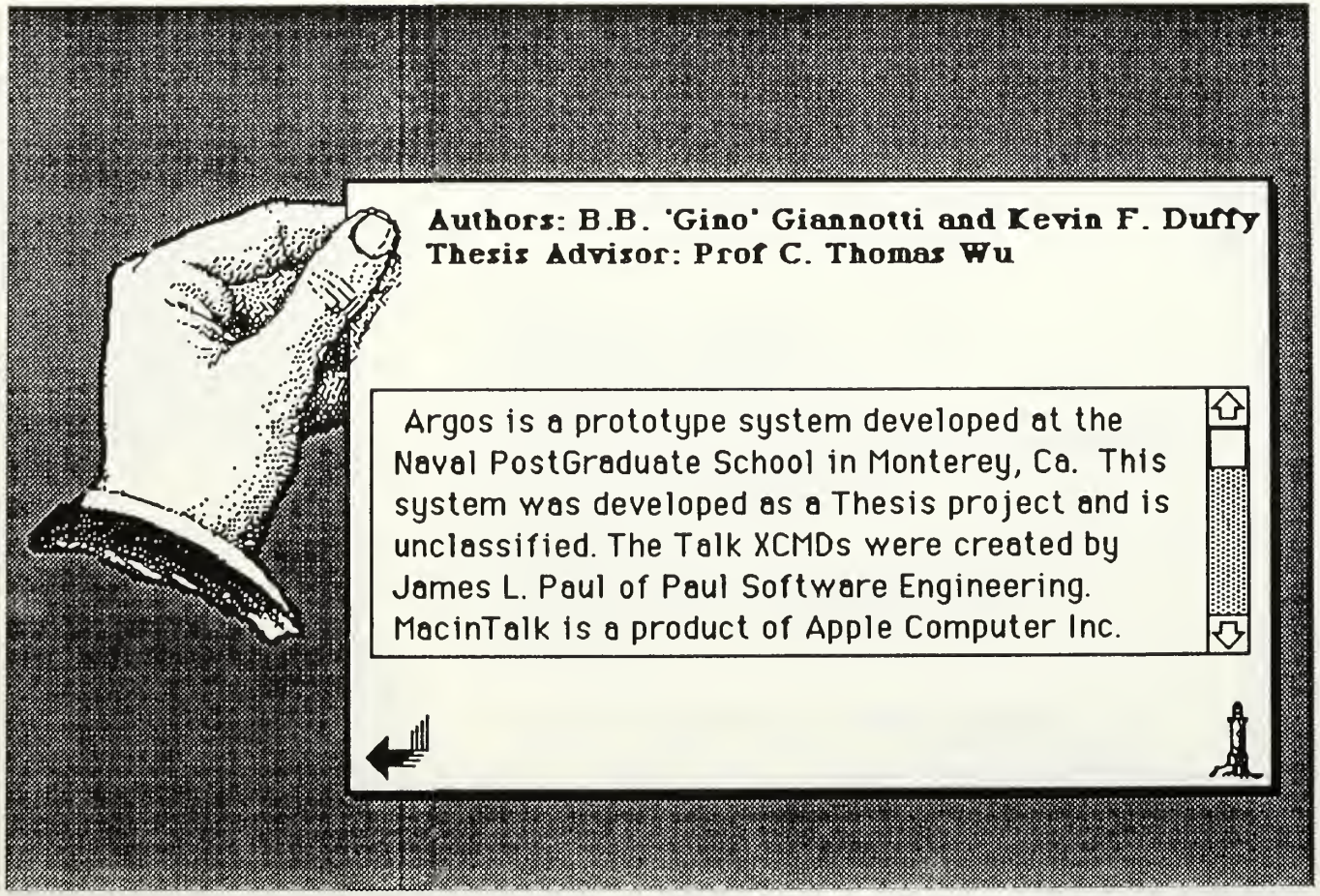

Administrative Functions
- EDVR
- SMD
- Awards
- Executive Correspondence
- Ticklers
- Inspections
- Notices and Instructions
- Legal
- Planning Board for Training
- etc ...

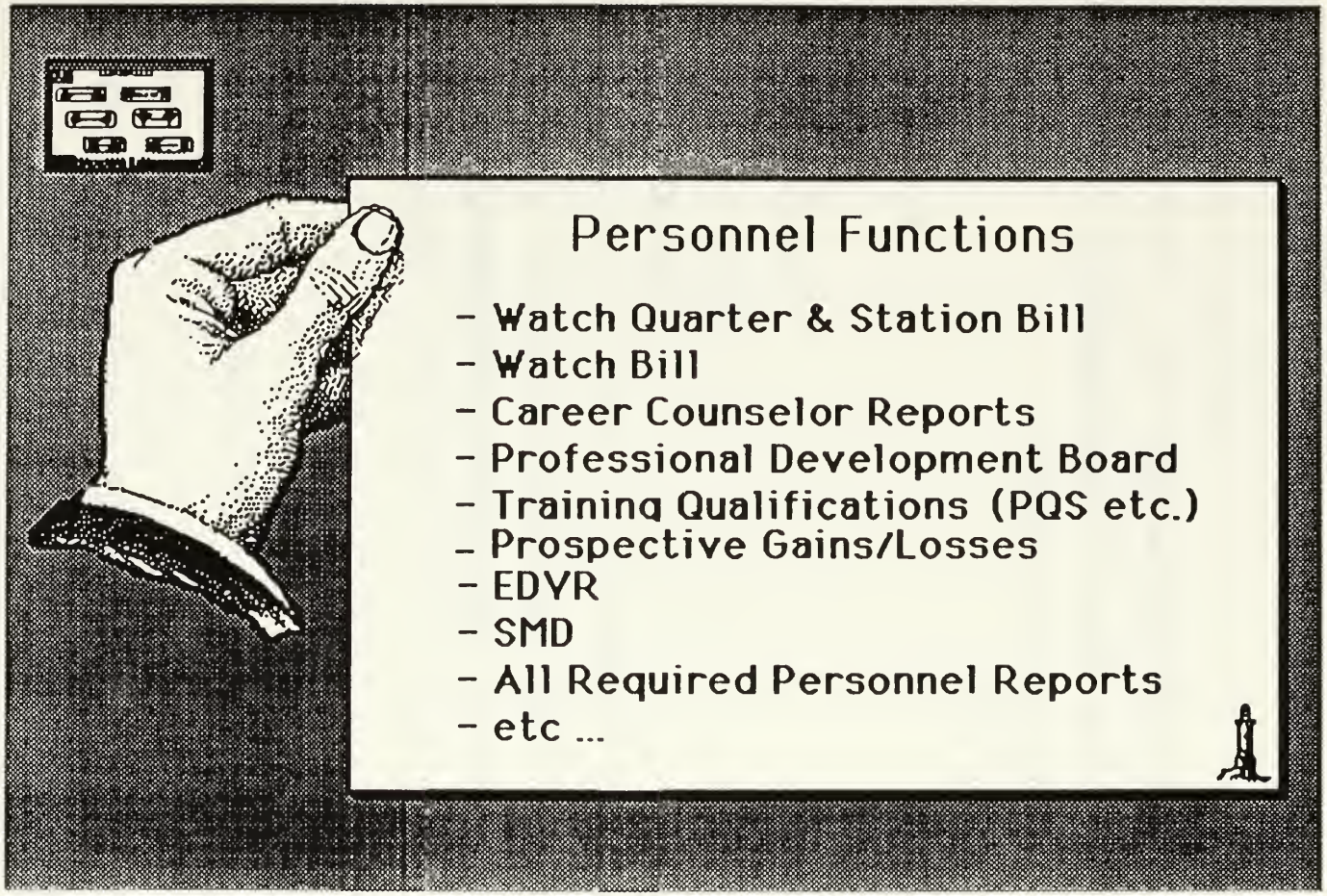

LIBRARY DIRECTORY
- This set of cards will contain a directory to the reference library which will contain tech manuals, drawings and other references.
- Using the built in search and browse capabilities the user will be able to rapidly access all reference materials stored on CD ROM and accessed via this interface.


Medical Functions
- Medical Readiness
- Dental Readiness
- Appointments (Physicals etc.)
- Physical Fitness
- All Required Reports
- Messing & Berthing Inspections
- Immunization Records
- etc ...


Supply Functions
- Financial Management
- Optar
- Food Service Records
- Parts Support
- All Disbursing Activities
- Imprest Fund
- Ship's Store Inventories
- etc ...
REVIEW REQN


Operational Functions
- Long Range Training Plan
- Short Range Training Plan
- ROC/POE
- Operational Reports
- Upcoming Exercise Requirements
- Required Reports
- CRYPTOLOGY
- etc ...


Authors: B.B. 'Gino' Giannotti and Kevin F. Duffy
Thesis Advisor: Prof C. Thomas Wu
Argos is a prototype system developed at the Naval PostGraduate School in Monterey, Ca. This system was developed as a Thesis project and is unclassified. The Talk XCMDs were created by James L. Paul of Paul Software Engineering. MacinTalk is a product of Apple Computer Inc.


Personnel Functions
- Watch Quarter & Station Bill
- Watch Bill
- Career Counselor Reports
- Professional Development Board
- Training Qualifications (PQS etc.)
- Prospective Gains/Losses
- EDVR
- SMD
- All Required Personnel Reports
- etc ...

```
** STACK SCRIPT ************************************
on openStack
  global mode
  -- mode may be any of the following types:
   -- navigate - traverse through the graphical hierarchy
  -- order - for ordering an item via graphics
 put "NAVIGATE" into MODE
  hide message box
  hide menubar
  set userlevel to 5
end openStack

on closestack
  -- this handler will automatically compact stack
  if the freesize of this stack > 0.15 * the size of this stack then
    doMenu "Compact Stack"
 end if
end closestack

on gohome
 play "BYE"
end gohome
```

```
function CLICKLINE
   -- this function will return the line number of a field where a
   --  mousedown event has occurred

    return trunc(((scroll of the target)¬
   + (item 2 of the clickloc) - (item 2 of the rect of the target¬
   )) div the textheight of the target) + 1
end CLICKLINE

function JULIANDATE
   -- returns julian date
  put the date into CURRENT
  convert CURRENT to seconds
  put "1/1/" into FIRSTDATE
  put char 4 to 5 of third item of the long date after FIRSTDATE
  convert FIRSTDATE to seconds
  put char 5 of third item of the long date into YEAR
  return YEAR & (CURRENT - FIRSTDATE) div 86400 + 1
end JULIANDATE

** BACKGROUND #1: GRAPHIC ************************************
on closecard
   -- this handler will automatically reset cards to original state
   --  if field "Description" is visible
```

```
  if visible of field "Description" = true then
    set lockscreen to true
    hide field "Description"
    repeat with i=1 to the number of buttons
     show button i
    end repeat
     show background button "sorry"
    set lockscreen to false
   hide msg
  end if
end  closecard

on ARGOSTALK x
   -- this handler will speak in computer voice the text contained in
   -- x.  This procedure requires several TALK XCMD's and MacinTalk
   -- must be in the system folder.
   if hilite of background button "VOICE" = true then
   TALK x, 160, 115
  end if
end ARGOSTALK

on  closebackground
   -- reset Mode button
   set name of background button id 40 to "Navigate Mode"
end  closebackground
```

```
on  opencard
   -- this handler will speak in computer voice the text conatined in
   -- x.  This procedure requires several TALK XCMD's and MacinTalk
   -- must be in the system folder. This procedure will be invoked
   -- only if the individual card does not have an OpenCard Handler.
   if hilite of background button "VOICE" = true then
   TALK FIELD "NOMENCLATURE", 160, 115
  end if
   show card picture
end  opencard

on  returnkey
   -- this is a redefinition of the returnkey function
   -- for the purposes of automating the find string command
   -- so the user may simply hit return in order to find the next
   -- occurence of a find string in both the description field or
   -- the nomenclature field. HyperCard doesn't support this without
   -- a custom handler.

  if (char 1 to 11 of msg) = "find string" then
    put the id of this card into tempid
     if visible of field "Description" then
      set lockscreen to true
       send returnKey to Hypercard
```

```
      if tempid <> id of this card then
        go recent card
        hide card picture
        set visible of field "Description" to true
        repeat with i=1 to the number of buttons
        hide button i
       end repeat
         hide background button "sorry"
     end if
      set lockscreen to false
   else
       send returnKey to Hypercard
   end if
  else
     send returnKey to Hypercard
  end if
end  returnKey


On GRAPHIC linenumber
   -- this handler provides the functionality for the transparent
   -- graphics buttons found on the illustrated parts breakdown.
   -- This handler provides for both the navigate and the order mode.
   -- The parameter linenumber corresponds to the sequence number of
   -- the card button from which it was called. On the the equivalent
```

```
-- line number of the field "DATA" there are two elements which
-- are the GRAPHIC data link id of the next card in the Hierarchy
-- the second element is the id of the card in the COSAL stack which
-- contains some of the data associated with a given part.

put first item of line linenumber of field "DATA" into GRAPHICLINK
put second item of line linenumber of field "DATA" into DATALINK
GLOBAL MODE
if MODE = "ORDER" then
 if DATALINK = " NONE" then
  play "ITEM NOT FOUND"
  exit GRAPHIC
 end if
  push card
  set the cursor to 4
  set LockScreen to TRUE
 go to CARD ID DATALINK of STACK "COSAL"
  put field "ITEM NAME" into ITEMNAME
  put field "APL" into APL
 put field "SOURCE CODE" into SOURCECODE
 put field "COG CODE" into COGCODE
 put field "MATl CONTROL CODE" into MATLCONTROLCODE
  put field "STOCK NUMBER" into STOCKNUMBER
  put field "UNIT OF ISSUE" into UNITOFISSUE
   go to stack equipment
```

```
find APL in field "APL"
put field "UIC" into UIC
put field "WC EQPT" into WCEQPT
put field "EIC" into EIC
go to stack forms
go to first card of background "NAVSUP 1250"
domenu new card
put JULIANDATE() into field "REQN DATE"
put ITEMNAME into field "NOUN NAME OR REF SYM"
put APL into field "APL/AEL/CID"
put UIC into field "UIC"
put WCEQPT into field "WC"
put WCEQPT into field "DEPT NO"
put EIC into field "EIC"
put SOURCECODE into field "SC"
put COGCODE into field "COG"
put MATLCONTROLCODE into field "MCC"
put char 1 to 4 of STOCKNUMBER into field "FSC"
put char 5 to 13 of STOCKNUMBER into field "NIIN"
put char 14 to 15 of STOCKNUMBER into field "SMIC"
put UNITOFISSUE into field "U/I"
put "X" into field "YES"
set the cursor to 1
set lockscreen to false
select text of field "URGY"
```

```
    -- get data and stick into order card of stack forms
    -- go to other stacks
    -- go to order card  (pop card to come back)

  put "NAVIGATE" into MODE
  play "NAVIGATE MODE"
 else
  if GRAPHICLINK = "NONE" THEN
   PLAY "YOU HAVE REACHED"
  ELSE
   if GRAPHICLINK = "INCOMPLETE" THEN
    PLAY "SORRY THIS AREA HAS NOT"
   ELSE
     PUT THE SHORT id of this CARD into UPLINK
    VISUAL EFFECT ZOOM OUT
      go to card ID GRAPHICLINK
        -- the next line of code dynamically sets the uplink path of
       -- the hierarchy
      PUT UPLINK INTO field "UPLINK"
      end if
   end if
 END IF
end GRAPHIC
```

```
** BKGND #1, FIELD #1: Description
************************************
on mouseup
   -- this handler turns show field "description" off and
   -- show the card picture with associated buttons on.
   show card picture
   set the highlight of background btn "VOICE" to true
   set visible of field "Description" to false
   repeat with i=1 to the number of buttons
    show button i
   end repeat
   show background button "sorry"
end mouseup

** BKGND #1, FIELD #5: data ***********************************
-- EACH LINE NUMBER OF THE FIELD CONTAINS TWO DATA ITEMS WHICH
-- CORRESPOND TO A BUTTON NUMBER. I.E. LINE 1 CONTAINS DATA FOR BUTTON 12
-- THE FIRST ITEM IS THE CARD ID OF THE CHILD OF THIS ITEM
-- THE SECOND ITEM IS THE CARD ID OF THE CARD IN THE COSAL STACK WHICH
-- CORRESPONDS TO THIS ITEM

** BKGND #1, BUTTON #1: TECHMAN
************************************
on mouseUp
   -- this handler toggles between showing field "description" and
```

```
   -- showing the card picture with associated buttons.
  PLAY "TECHMAN"
   if visible of field "Description" = true then
      set the highlight of background btn "VOICE" to true
     show card picture
     hide field "Description"
     repeat with i=1 to the number of buttons
      show button i
    end repeat
     show background button "sorry"
  else
     hide card picture
     show field "Description"
     repeat with i=1 to the number of buttons
      hide button i
    end repeat
     hide background button "sorry"
  end if
end  mouseUp

** BKGND #1, BUTTON #2: Order ************************************
on  mouseUp
 PLAY "PASSWORD PLEASE"
 play "ORDER MODE"
 GLOBAL MODE
```

```
  put "ORDER"into MODE
   set name of background button id 40 to  "Order Mode"
  play "MAKE A SELECTION"
  ANSWER "MAKE SELECTION" with OK or CANCEL
  if it = "CANCEL" then
    Put "NAVIGATE" into MODE
   PLAY "NAVIGATE MODE"
      set name of background button id 40 to  "Navigate Mode"
  end if
end  mouseUp

** BKGND #1, BUTTON #3: HELP ************************************
on  mouseUp
  PLAY "HELP"
   push this card
  go to stack "ARGOS HELP"
end  mouseUp

** BKGND #1, BUTTON #4: COSAL ************************************
on  mouseUp
 PLAY "COSAL"
   push this card
  go to stack "COSAL"
end  mouseUp
```

```
** BKGND #1, BUTTON #5: EQUIP ************************************
on mouseUp
 PLAY "EQUIPMENT"
 PUSH THIS CARD
 GO TO STACK "EQUIPMENT"
end mouseUp

** BKGND #1, BUTTON #6: APL ***************************************
on mouseUp
  PLAY "APL"
   push this card
  go to stack "APL"
end mouseUp

** BKGND #1, BUTTON #7: NAVIGATE
***********************************
on mouseUp
  PLAY "CSMP"
   push this card
  go to stack "CSMP"
end mouseUp

** BKGND #1, BUTTON #8: Sorry *************************************
on mouseUp
  PLAY "SORRY THIS AREA HAS NOT"
```

```
end  mouseUp

** BKGND #1, BUTTON #9: New Button
************************************
on  mouseUp
  -- goes back to ship side view level
  visual effect zoom out
  go to card id 22260
end  mouseUp

** BKGND #1, BUTTON #10: UP ************************************
on  mouseUp
  -- goes up the hierarchy
  visual effect zoom out
  go to card id field "Uplink"
end  mouseUp

** BKGND #1, BUTTON #11: Find ************************************
on  mouseUp
  -- this handler provides for a modified search.
  put the id of this card into tempid
 PLAY "SEARCH"
  ask"Please enter Search String."
  if visible of field "Description" then
    set lockscreen to true
```

```
    set the highlight of background btn "VOICE" to false
    put "find string" && quote & it & quote && "in field Description"¬
   into msg
   hide msg
     send returnkey to hypercard
   if tempid <> id of this card then
     go recent
       set the highlight of background btn "VOICE" to true
     set lockscreen to false
   end if
  else
   hide msg
   put "find string" && quote & it & quote && "in field NOMENCLATURE" into
msg
   hide msg
     send returnkey to hypercard
  end if
end mouseUp

** BKGND #1, BUTTON #12: LIBRARY
************************************
on mouseUp
  PLAY "LIBRARY"
  push card
  go to card library OF STACK "ARGOS"
```

```
end  mouseUp

** BKGND #1, BUTTON #13: EXIT ***********************************
on  mouseUp
   gohome
  go home
end  mouseUp

** BKGND #1, BUTTON #14: PRINT
***************************************
on  mouseUp
  play "PRINT"
   doMenu  Print  Card
end  mouseUp

** BKGND #1, BUTTON #16: FORMS
***************************************
on  mouseUp
 PLAY "FORMS"
   push  this  card
  go to stack "FORMS"
end  mouseUp

** BKGND #1, BUTTON #18: VOICE
***************************************
```

```
on  mousedown
   -- toggles  voice  on/off
   if the hilite of me then
   ARGOSTALK "VOICE ON"
   else
   TALK "VOICE OFF", 160, 115
   end if
end  mousedown

**  BACKGROUND  #2  ************************************

on  openBackground
end  openBackground

** BKGND #2, BUTTON #1: RETURN
**************************************
on  mouseUp
   visual effect zoom out
   go to card id field "Uplink"
end  mouseUp

**  BKGND  #2,  BUTTON  #2:  Sorry  ************************************
on  mouseUp
  PLAY "SORRY THIS AREA HAS NOT"
    answer "This AREA has not been modeled"
```

```
end  mouseUp

** BKGND #2, BUTTON #3: EXIT ************************************
on  mouseUp
 GOHOME
  go home
end  mouseUp

** BKGND #3, BUTTON #1: Return
**************************************
on  mouseUp
  pop card
end  mouseUp

** BKGND #3, BUTTON #2: EXIT ************************************
on  mouseUp
   gohome
  go home
end  mouseUp

** BKGND #3, BUTTON #3: PRINT ***********************************
on  mouseUp
  doMenu Print Card
end  mouseUp
```

```
** CARD #1 ************************************
on opencard
  visual effect iris close very slowly
 PLAY START
 PLAY "WELCOME TO ARGOS"
  wait 4 seconds
  go to next card
end opencard

** CARD #2 ************************************
on opencard
end opencard

** CARD #2, BUTTON #1: Halyburton
**************************************
on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 10931
  put it into field "uplink"
end mouseUp

** CARD #2, BUTTON #5: ABOUT ARGOS
**************************************
```

```
on  mouseUp
   visual effect zoom out
 PLAY "ABOUT ARGOS"
   push card
  go to card id 5287
end  mouseUp

**  CARD  #3  ************************************

ON OPENCARD
  play "FUNCTIONAL AREAS"
END OPENCARD

** CARD #3, BUTTON #1: Deck Profile
****************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 20294
   put it into field "UpLink"
end  mouseUp

** CARD #3, BUTTON #2: MAINTENANCE
****************************************
```

```
on  mouseUp
   visual effect zoom out
  PLAY "MAINTENANCE"
   get short id of this card
   go to card id 22260
   put it into field "UpLink"
end  mouseUp


** CARD #3, BUTTON #3: ADMINISTRATION
*************************************

on  mouseUp
   visual effect zoom out
  PLAY "ADMINISTRATION"
   get short id of this card
   go to card id 11414
   put it into field "UpLink"
end  mouseUp

** CARD #3, BUTTON #4: Operations
*************************************

on  mouseUp
```

```
   visual effect zoom out
  PLAY "OPERATIONS"
   get short id of this card
   go to card id 14330
   put it into field "UpLink"
end  mouseUp

** CARD #3, BUTTON #5: Medical ************************************

on  mouseUp
   visual effect zoom out
  PLAY "MEDICAL"
   get short id of this card
   go to card id 13530
   put it into field "UpLink"
end  mouseUp

** CARD #3, BUTTON #6: Supply *************************************

on  mouseUp
   visual effect zoom out
  PLAY "SUPPLY"
   get short id of this card
   go to card id 13860
   put it into field "UpLink"
```

```
end  mouseUp

** CARD #3, BUTTON #7: Personnel
**************************************

on  mouseUp
   visual effect zoom out
  PLAY "PERSONNEL"
   get short id of this card
   go to card id 14878
   put it into field "UpLink"
end  mouseUp

** CARD #3, BUTTON #9: ABOUT ARGOS
**************************************

on  mouseUp
   visual effect zoom out
  PLAY "ABOUT ARGOS"
   push card
   go to card id 5287
end  mouseUp
```

```
** CARD #5, BUTTON #1: Engine Room
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 2790
   put it into field "UpLink"
end  mouseUp

** CARD #5, BUTTON #2: Deck Profile
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 20294
   put it into field "UpLink"
end  mouseUp

** CARD #6, BUTTON #1: New Button
**************************************

on  mouseUp
   visual effect zoom out
```

```
  get short id of this card
  go to card id 2790
  put it into field "UpLink"
end  mouseUp

** CARD #7: Main Eng Rm Perspective
************************************

ON OPENCARD
 ARGOSTALK "ENGIN ROOM LEVEL SLECTION"
END OPENCARD

** CARD #7, BUTTON #1: UPPER LEVEL PORT
************************************

on  mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 24891
  put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #2: Lower Level Port
************************************
```

```
on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 26121
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #3: Lower Level Stbd 1 of 4
**************************************
on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 26624
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #4: Upper Level Stbd 1 0f 7
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 25681
   put it into field "UpLink"
end  mouseUp
```

```
** CARD #7, BUTTON #5: Upper Level Stbd 3 of 7
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 25681
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #6: Upper Level Stbd 4 of 7
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 25681
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #7: Upper Level Stbd 5 of 7
**************************************

on  mouseUp
```

```
  visual effect zoom out
  get short id of this card
  go to card id 25681
  put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #8: Upper Level Stbd 6 of 7
**************************************

on  mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 25681
  put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #9: Upper Level Stbd 7 of 7
**************************************

on  mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 25681
  put it into field "UpLink"
end  mouseUp
```

```
** CARD #7, BUTTON #10: Upper Level Stbd 2 of 7
************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 25681
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #11: Lower Level Stbd 3 of 4
************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 26624
   put it into field "UpLink"
end  mouseUp

** CARD #7, BUTTON #12: Lower Level Stbd 2 of 4
************************************

on  mouseUp
```

```
  visual effect zoom out
  get short id of this card
  go to card id 26624
  put it into field "UpLink"
end mouseUp

** CARD #7, BUTTON #13: Lower Level Stbd 4 of 4
**************************************

on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 26624
  put it into field "UpLink"
end mouseUp

** CARD #8 ***********************************

ON OPENCARD
 ARGOSTALK "ENGIN ROOM UPPER LEVEL PORT SIDE"
END OPENCARD

** CARD #9 ***********************************

ON OPENCARD
```

```
 ARGOSTALK "ENGIN ROOM UPPER LEVEL STARBIRD SIDE"
END OPENCARD

** CARD #10 ************************************

ON OPENCARD
 ARGOSTALK "ENGIN ROOM LOWER LEVEL PORT SIDE"
END OPENCARD

** CARD #10, BUTTON #1: Propulsion Gas Turbine 1B
**************************************

on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 28371
  put it into field "UpLink"
end mouseUp

** CARD #11 ************************************

ON OPENCARD
 ARGOSTALK "ENGIN ROOM LOWER LEVEL STARBIRD SIDE"
END OPENCARD
```

```
** CARD #11, BUTTON #1: Propulsion Gas Turbine 1A
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 28371
   put it into field "UpLink"
end  mouseUp

**  CARD  #12  ***********************************

ON OPENCARD
 ARGOSTALK "GAS TURBIN MODULE"
END OPENCARD

** CARD #12, BUTTON #1: Gas Turbine in Module
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 8729
   put it into field "UpLink"
end  mouseUp
```

```
** CARD #13: LM2500 ************************************

ON OPENCARD
 ARGOSTALK "LM TWENTY 5 HUNDRED GAS TURBIN ENGIN"
END OPENCARD

** CARD #13, BUTTON #1: LM2500 GT Turbine Section
**************************************

on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 21321
  put it into field "UpLink"
end mouseUp

** CARD #13, BUTTON #2: LM2500 Gas Turbine Eng
**************************************

on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 23316
  put it into field "UpLink"
```

```
end  mouseUp

**  CARD  #14  ************************************
ON OPENCARD
 ARGOSTALK "GAS TURBIN EXPLODED VIEW 1 AL FA"
END OPENCARD

** CARD #14, BUTTON #1: New Button
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #14, BUTTON #2: Centerbody Assembly
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #14, BUTTON #3: comp rear fr/combustor
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
```

```
  GRAPHIC (number of me)
end  MouseUp

** CARD #14, BUTTON #4: comp. rear stator 1 of 2
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #14, BUTTON #5: comp. rear stator 2 of 2
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #14, BUTTON #6: HP COMP STATOR 1 of 2
************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp
```

```
** CARD #14, BUTTON #7: HP Comp Stator 2 of 2
**************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #14, BUTTON #8: Compressor Rotor
**************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #15: HP COMp. Rotor 1 of 2
**************************************
ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR ROTER"
END OPENCARD

** CARD #15, BUTTON #1: continue
**************************************
On MouseUp
  visual effect zoom out
  get short id of this card
```

```
  go to card id 12411
   put it into field "uplink"
end  MouseUp

**  CARD  #16  ************************************
ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR ROTER CONTINUATION"
END OPENCARD

**  CARD  #17  ************************************
ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR STATER ALTERNET VIEW 3"
END OPENCARD

**  CARD  #18:  HP  Comp  Stator  ************************************

ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR STATER"
END OPENCARD

** CARD #18, BUTTON #1: CLOSE UP
**************************************

on  mouseUp
   visual effect zoom out
```

```
  get short id of this card
  go to card id 9423
  put it into field "uplink"
end mouseUp

** CARD #19 ************************************

ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR STATER ALTERNET VIEW 1"
END OPENCARD

** CARD #19, BUTTON #1: Close up
**************************************

on mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 9967
  put it into field "uplink"
end mouseUp

** CARD #20: HP COMP STATOR ***********************************

ON OPENCARD
 ARGOSTALK "HIGH PRESSURE COMPRESSOR STATER ALTERNET VIEW 2"
```

```
END OPENCARD

** CARD #20, BUTTON #1: Close Up
**************************************

on  mouseUp
   visual effect zoom out
   get short id of this card
  go to card id 7832
   put it into field "uplink"
end  mouseUp

**  CARD #21: compressor rear stator
**************************************

ON OPENCARD
 ARGOSTALK "COMPRESSER REAR STATER"
END OPENCARD

** CARD #21, BUTTON #1: CASE ASSY, CSR1
**************************************
On  MouseUp
   GRAPHIC (number of me)
end  MouseUp
```

```
** CARD #21, BUTTON #2: CASE ASSY, CSR2
************************************
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #21, BUTTON #3 ************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #4: LINER, VANE DOVETAIL, LH1
************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #5: LINER, VANE DOVETAIL, LH2
************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #6: LINER, VANE DOVETAIL, RH1
************************************
```

```
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #7: LINER, VANE DOVETAIL RH2
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #8: VANE, STG 12, CRS
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #9: VANE, STG 13, CRS
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #10: VANE, STG 14, CRS
************************************
on  mouseUp
```

```
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #11: VANE, STG 15, CRS
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #12: VANE, STG 16, OGV, CRS
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #13: GASKET, METAL, O-RING1
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #14: PLUG, MACH-BORESCOPE1
**************************************
on  mouseUp
  GRAPHIC (number of me)
```

```
end  mouseUp

** CARD #21, BUTTON #15: BOLT, MACH, DBL HEX HD
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #16: BOLT, BODY BOUND, DBL HEX
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #17: BOLT, MACH, DBL HEX
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #18: BOLT, BODY BOUND, DBL HEX
**************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp
```

```
** CARD #21, BUTTON #19: NUT, SELF LOCK, DBL HEX
**************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #20: KEY, CASING
**************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #21 **************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #22 **************************************
on mouseUp
  GRAPHIC (number of me)
end mouseUp

** CARD #21, BUTTON #23: GASKET, METAL, O-RING2
**************************************
```

```
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #25: PLUG, MACH-BORESCOPE2
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #26: PLUG, MACH-BORESCOPE3
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #27: CASE, ASSY, CSR3
************************************
on  mouseUp
  GRAPHIC (number of me)
end  mouseUp

** CARD #21, BUTTON #28: CASE, ASSY, CSR4
************************************
on  mouseUp
```

```
  GRAPHIC (number of me)
end  mouseUp

**  CARD  #22  ************************************
ON OPENCARD
 ARGOSTALK "CENTER BODY, INLET, COMPRESSER FRONT FRAME"
END OPENCARD

** CARD #23: comp rear fr/combustor
**************************************
ON OPENCARD
 ARGOSTALK "COMPRESSER REAR FRAME, COMBUSTER"
END OPENCARD

** CARD #23, BUTTON #1: Air Seals
**************************************

on  mouseUp
  visual effect zoom out
  get short id of this card
  go to card id 2535
  put it into field "uplink"
end  mouseUp

**  CARD  #25  ************************************
```

```
ON OPENCARD
 ARGOSTALK "GAS TURBIN EXPLODED VIEW 1 BRA VO"
END OPENCARD

** CARD #26 ************************************
ON OPENCARD
  ARGOSTALK "Inlet Gearbox Assembly Breakdown"
END OPENCARD

** CARD #26, BUTTON #1: GEARBOX ASSY, INLET
**************************************
 -- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #2: ADAPTER, SHAFT, GEAR
**************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp
** CARD #26, BUTTON #3: SHIM ************************************
-- Graphic Handler may be found in this cards background
On MouseUp
```

```
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #4: ADAPTER PLATE
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #5: ADAPTER BLOCK
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #6: ADAPTER, BLOCK, INTERNAL
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp
```

```
** CARD #26, BUTTON #7: HOUSING ASSY
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #8: HOUSING, BRNG-GEAR SHAFT
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #9: SCREW, MACHINE,10-32UNF
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #10: SHIM ASSY, LAMINATED
************************************
-- Graphic Handler may be found in this cards background
```

```
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #11: ADAPTER ASSY-INLETGEARBX
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #12: PACKING,PREFORMED, VITON
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #13: NUT, PLAIN, SHAFT
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp
```

```
** CARD #26, BUTTON #14: GEARSHAFT, BEVEL
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #15: RING, RETAINING
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #16: NUT, SELFLOCKING,GANG CH
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #17: RETAINER, BRNG, DUPLEX
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
```

```
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #18: BRNG, BALL
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #19: NOZZEL, OIL1
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #20: NOZZEL,OIL2
**************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp
```

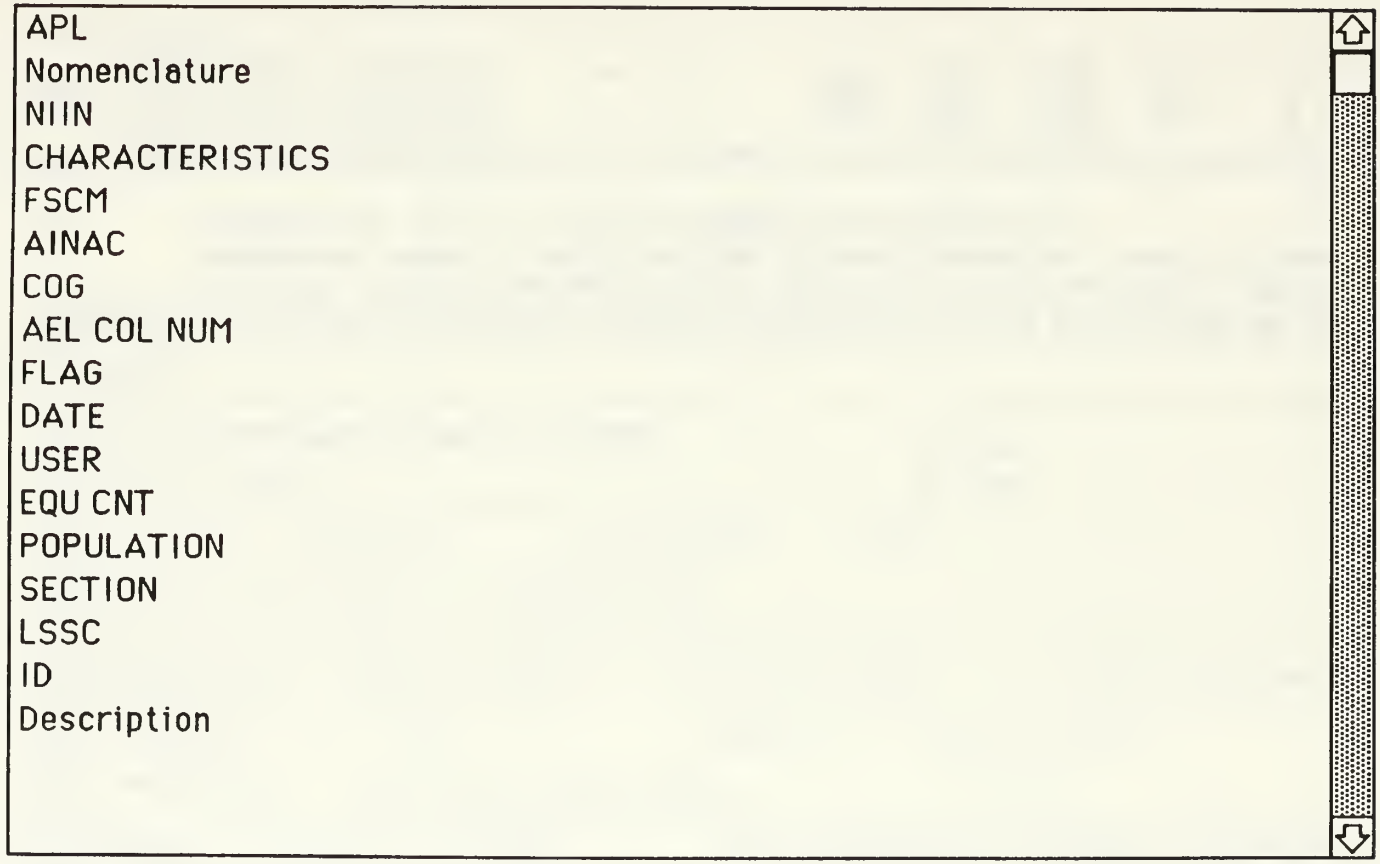
```
** CARD #26, BUTTON #21: HOUSING, BRNG, ROLLER
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #22: BRNG, ROLLER, CYLINDRICAL
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #23: RETAINER, BEARING
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #24: NUT, SELF LOCKING,DBLHEX2
************************************
-- Graphic Handler may be found in this cards background
On MouseUp
```

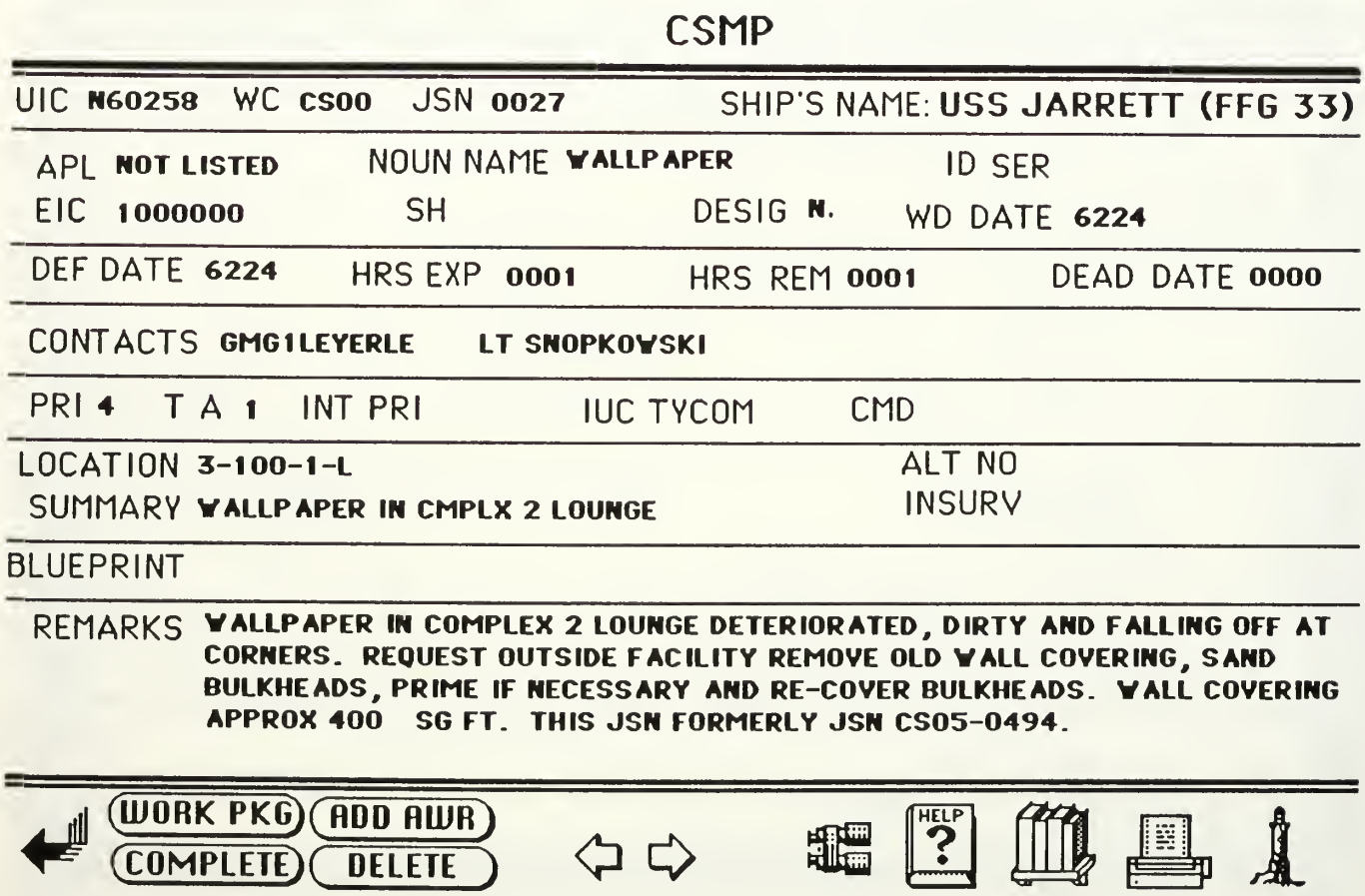
```
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #25: NUT, SELF LOCKING DBLHEX1
*************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #26: NOZZEL ASSY, OIL
*************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp

** CARD #26, BUTTON #27: BOLT, MACH, DBLHEX
*************************************
-- Graphic Handler may be found in this cards background
On  MouseUp
  GRAPHIC (number of me)
end  MouseUp
```

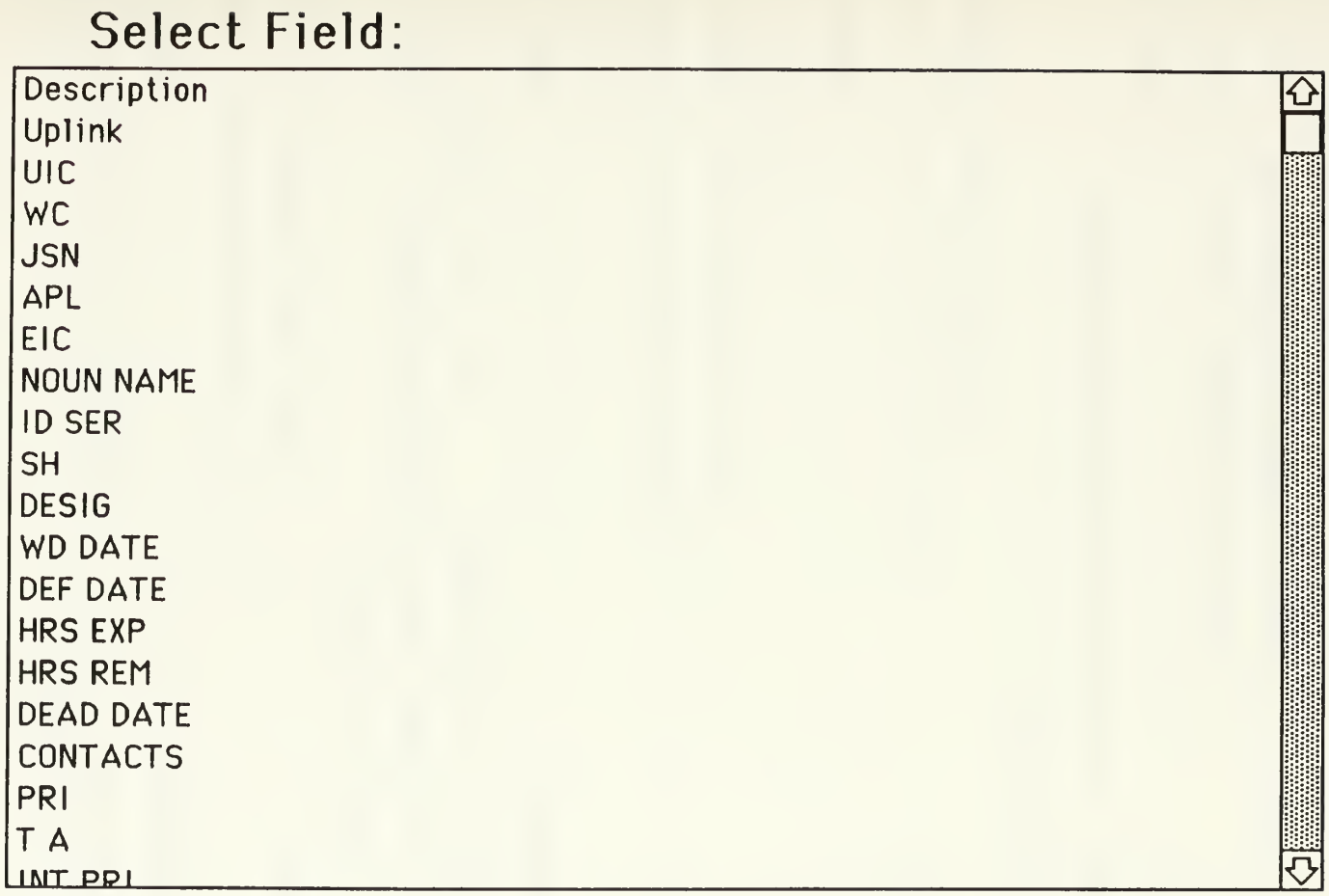
```
** CARD #26, BUTTON #28: NUT, SELF LOCK, DBLHEX
**************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #26, BUTTON #29: BOLT, MACHINE, AMS 6322
**************************************
-- Graphic Handler may be found in this cards background
On MouseUp
  GRAPHIC (number of me)
end MouseUp

** CARD #34, BUTTON #1: Return
**************************************

on mouseUp
  pop card
end mouseUp
```

APPENDIX B

APL STACK


Select Field:
APL
Nomenclature
NIIN
CHARACTERISTICS
FSCM
AINAC
COG
AEL COL NUM
FLAG
DATE
USER
EQU CNT
POPULATION
SECTION
LSSC
ID
Description


CSMP
UIC N60258 WC CS00 JSN 0027 SHIP'S NAME: USS JARRETT (FFG 33)
APL NOT LISTED NOUN NAME WALLPAPER ID SER
EIC 1000000 SH DESIG N. WD DATE 6224
DEF DATE 6224 HRS EXP 0001 HRS REM 0001 DEAD DATE 0000
CONTACTS GMG1LEYERLE LT SNOPKOWSKI
PRI 4 T A 1 INT PRI IUC TYCOM CMD
LOCATION 3-100-1-L ALT NO
SUMMARY WALLPAPER IN CMPLX 2 LOUNGE INSURV
BLUEPRINT
REMARKS WALLPAPER IN COMPLEX 2 LOUNGE DETERIORATED, DIRTY AND FALLING OFF AT CORNERS. REQUEST OUTSIDE FACILITY REMOVE OLD WALL COVERING, SAND BULKHEADS, PRIME IF NECESSARY AND RE-COVER BULKHEADS. WALL COVERING APPROX 400 SG FT. THIS JSN FORMERLY JSN CS05-0494.
WORK PKG
ADD AWR
COMPLETE
DELETE
HELP


Select Field:
Description
Uplink
UIC
WC
JSN
APL
EIC
NOUN NAME
ID SER
SH
DESIG
WD DATE
DEF DATE
HRS EXP
HRS REM
DEAD DATE
CONTACTS
PRI
T A

SCRIPTS FOR STACK: CSMP

```
** STACK SCRIPT ************************************

on closestack
  if the freesize of this stack > 0.15 * the size of this stack then
    doMenu "Compact Stack"
  end if
end closestack

on gohome
  play "BYE"
end gohome

function CLICKLINE
   return trunc(((scroll of the target)¬
  + (item 2 of the clickloc) - (item 2 of the rect of the target¬
  )) div the textheight of the target) + 1
end CLICKLINE

function JULIANDATE
  -- returns julian date
  put the date into CURRENT
```

```
   convert CURRENT to seconds
   put "1/1/" into FIRSTDATE
   put char 4 to 5 of third item of the long date after FIRSTDATE
   convert FIRSTDATE to seconds
   put char 5 of third item of the long date into YEAR
  return YEAR & (CURRENT - FIRSTDATE) div 86400 + 1
end JULIANDATE


** BKGND #1, FIELD #1: Description
**************************************
on  mouseup
   show card picture
   set visible of field "Description" to false
   repeat with i=1 to the number of buttons
     show button i
   end repeat
    show background button "sorry"
end  mouseup

** BKGND #1, FIELD #29: FIELD CHOICE
**************************************
on  mouseup
   put line clickline() of me into fieldname
  hide me
```

```
   ask"Please enter Search String."
  put "find" && quote & it & quote && "in field" && ¬
   quote & fieldname & quote into msg
   hide msg
    send returnkey to hypercard
end mouseup

** BKGND #1, BUTTON #1: HELP ************************************
on mouseUp
 PLAY "HELP"
  push this card
 go to stack "ARGOS HELP"
end mouseUp

** BKGND #1, BUTTON #2: Find ************************************
on mouseUp
  put empty into fld "FIELD CHOICE"
   repeat with count = 1 to number of flds
      put the short name of fld count & return after fld "field choice"
   end repeat
 PLAY "SEARCH"
   show fld "field choice"
    answer "Please select search field"
end mouseUp
```

```
** BKGND #1, BUTTON #3: LIBRARY
************************************
on  mouseUp
  PLAY "LIBRARY"
   push card
  go to card library OF STACK "ARGOS"
end  mouseUp

** BKGND #1, BUTTON #4: EXIT ***********************************
on  mouseUp
   gohome
  go home
end  mouseUp

** BKGND #1, BUTTON #5: PRINT ***********************************
on  mouseUp
   set visible of background btn "COVER" to true
   doMenu Print Card
   set visible of background btn "COVER" to false
end  mouseUp

** BKGND #1, BUTTON #6: Return
************************************
on  mouseUp
  play "RETURN"
```

```
  pop card
  play "Navigate Mode"
end mouseUp

** BKGND #1, BUTTON #7: COVER ***********************************
on mousewithin
 SET THE VISIBLE OF me TO FALSE
end mousewithin
on mouseenter
 SET THE VISIBLE OF me TO FALSE
end mouseenter

** BKGND #1, BUTTON #8: DELETE
*************************************
on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
  repeat
    domenu delete card
    go next card
    if number of this card = 1 then
     exit repeat
   end if
  end repeat
end mouseUp
```

```
** BKGND #1, BUTTON #9: LOCK/UNLOCK
************************************
on  mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
 END REPEAT
end  mouseUp

** BKGND #1, BUTTON #10: WORK PKG
************************************
on  mouseUp
   go to card id 6397
end  mouseUp

** BKGND #1, BUTTON #11: Prev ************************************

on  mouseStillDown
   visual effect scroll left
  go to prev card of BACKGROUND "CSMP"
end  mouseStillDown

** BKGND #1, BUTTON #12: Next ************************************
```

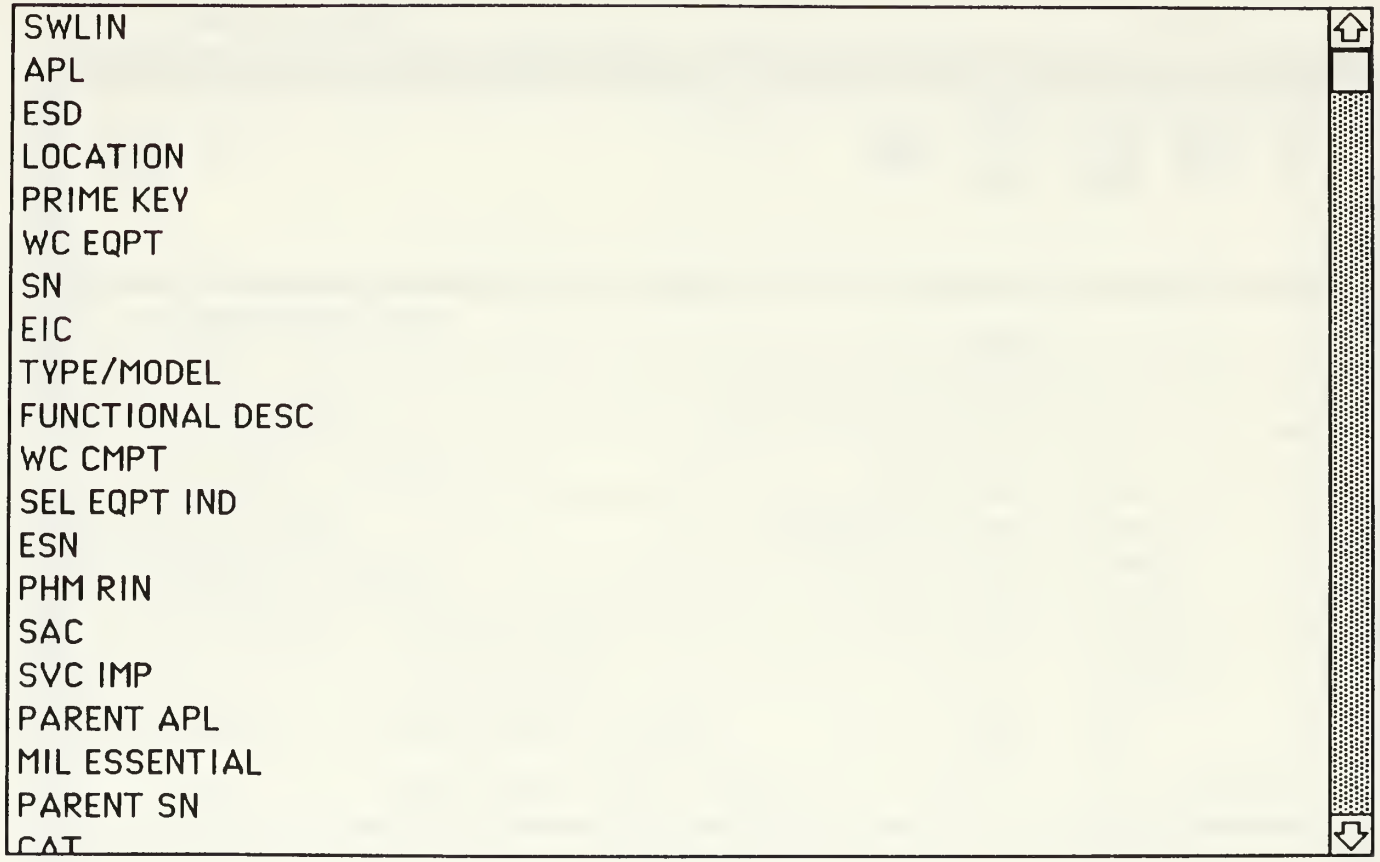
```
on  mouseStillDown
   visual  effect  scroll  right
  go to next card of BACKGROUND "CSMP"
end  mouseStillDown

** BKGND #1, BUTTON #13: ADD AWR
**************************************
on  mouseUp
  PLAY "2K"
   push  card
   lock  screen
  go to stack EQUIPMENT
  put field "UIC" into UIC
  go to first card of stack "CSMP"
   domenu  new  card
  put JULIANDATE() into field "WD DATE"
  put UIC into field "UIC"
   unlock  screen
  select text of field "WC"
end  mouseUp

** BKGND #2, FIELD #1: PRINT CHOICE
**************************************
on  mouseup
  put line clickline() of me into SORTNAME
```

```
    ask "What is the Print Criteria?"
   go to first card
   lock screen
   repeat for (the number of cards)
     if (the short id of this card <> 6397) then
       put fld sortname into SORTVAL
      if SORTVAL = it then
          unlock screen
         print card
         lock screen
      end if
    end if
    go next
   end repeat
end  mouseup
```

APPENDIX D

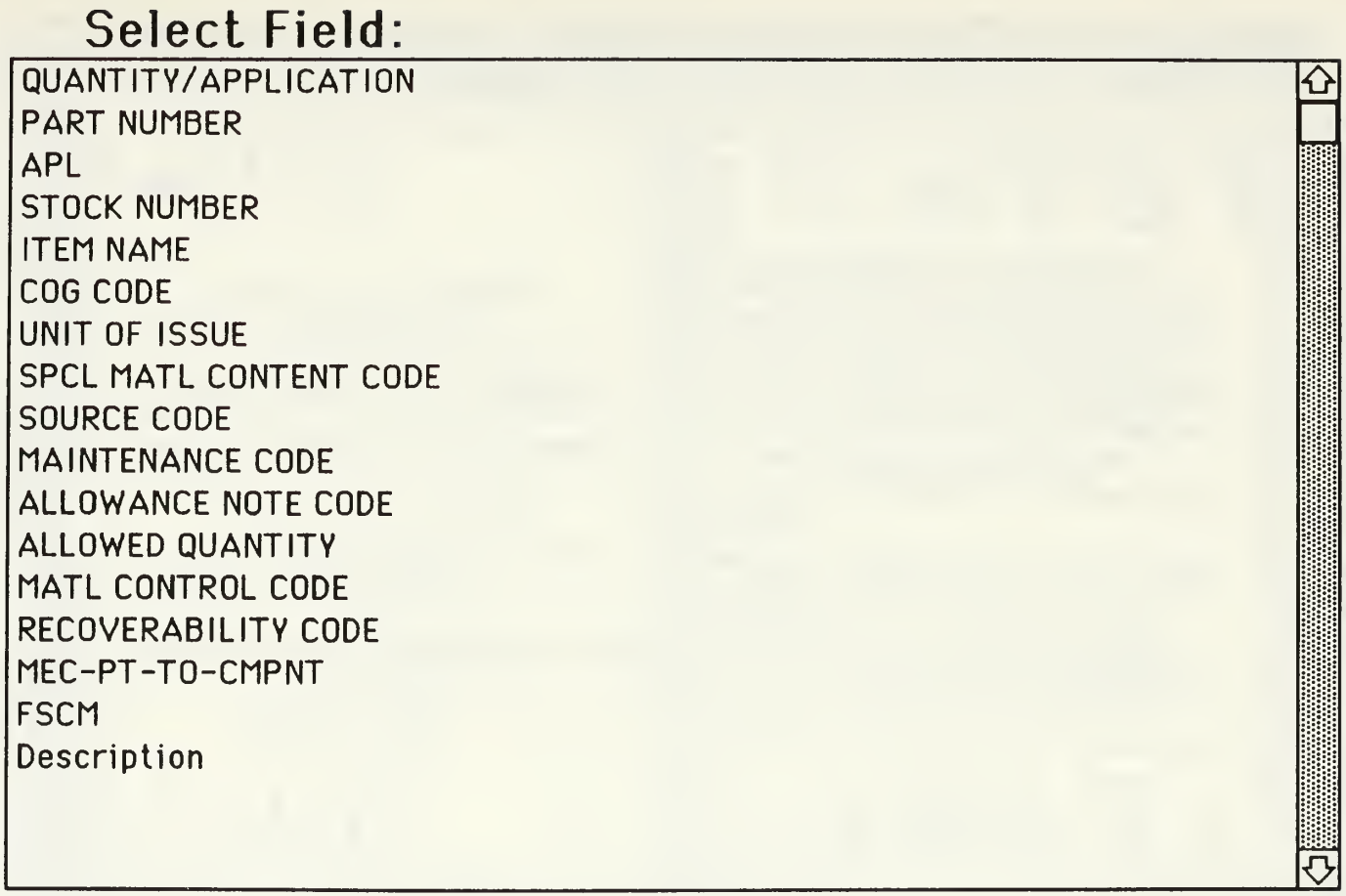
EQUIPMENT STACK

Select Field:



APPENDIX E

COSAL STACK


Select Field:
QUANTITY/APPLICATION
PART NUMBER
APL
STOCK NUMBER
ITEM NAME
COG CODE
UNIT OF ISSUE
SPCL MATL CONTENT CODE
SOURCE CODE
MAINTENANCE CODE
ALLOWANCE NOTE CODE
ALLOWED QUANTITY
MATL CONTROL CODE
RECOVERABILITY CODE
MEC-PT-TO-CMPNT
FSCM
Description

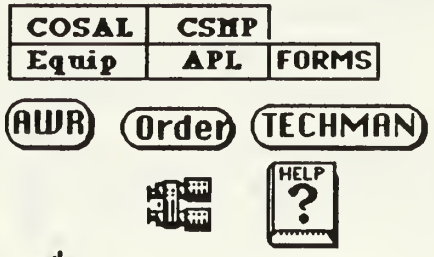
APPENDIX F

HELP STACK

Help

Buttons

Buttons are the main command mechanism which makes Argos function. There are two basic types of buttons: transparent and visible. Visible buttons have a button like appearance such as the following:



Help

The square buttons without ICONs are used to transfer the user to other data stacks. The round buttons such as AWR, Order etc. are used to perform a specific action. The buttons which are ICONS only are used for the following functions:

SEARCH for a string

Help - will bring user to this stack

Return will return the user back to thew stack of origin.

Help

Lighthouse - will take user back to the top level of the system.

Library - will take the user to the library of tech pubs, etc.

next - will take the user to the next card or record.

prev - will take the user to the previous card or record.

buoy - will show credits of Argos

printer - will print current screen.

Help

The current version of argos has two modes: Order Mode and Navigate Mode. The user can determine which mode they are in by looking at the top right of the screen where it will be visible. Whenever the mode changes the user will be notified by a human voice which will declare the new mode.

The computer voice is used to declare the title of each new card which is viewed. This voice may be activated and deactivated by the user by selecting/ deselecting the voice button at the upper right hand side of the screen. MacinTalk

Help

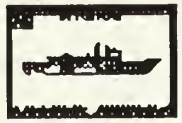
must be present in the System folder in for this function to operate.

If the mouse button is depressed, when the pointing finger icon is on top of a button, some action will take place. The action which takes place is determined by the button which is depressed and the mode that Argos is in at the time it is depressed. Some buttons will change the mode of Argos when depressed, such as the Order button.

Some buttons will navigate the user to unique functional areas or to unique stacks. Stacks are the discrete programs of the Argos system which buttons will link

Help

Transparent buttons are generally overlaid on top of a graphic such as the following:



Generally transparent buttons will be found somewhere on almost any graphic, such as an illustrated parts breakdown.

SCRIPTS FOR STACK: ARGOS HELP

```
** STACK SCRIPT ************************************
on openStack
  hide menuBAR
  hide message box
end openStack


** BKGND #1, BUTTON #1: Next ***********************************
on mouseUp
  visual effect wipe left
  go to next card of background "HELP"
end mouseUp

** BKGND #1, BUTTON #2: RETURN
**************************************
on mouseUp
 POP CARD
end mouseUp

** BKGND #1, BUTTON #3: Prev ***********************************
on mouseUp
  visual effect wipe right
```

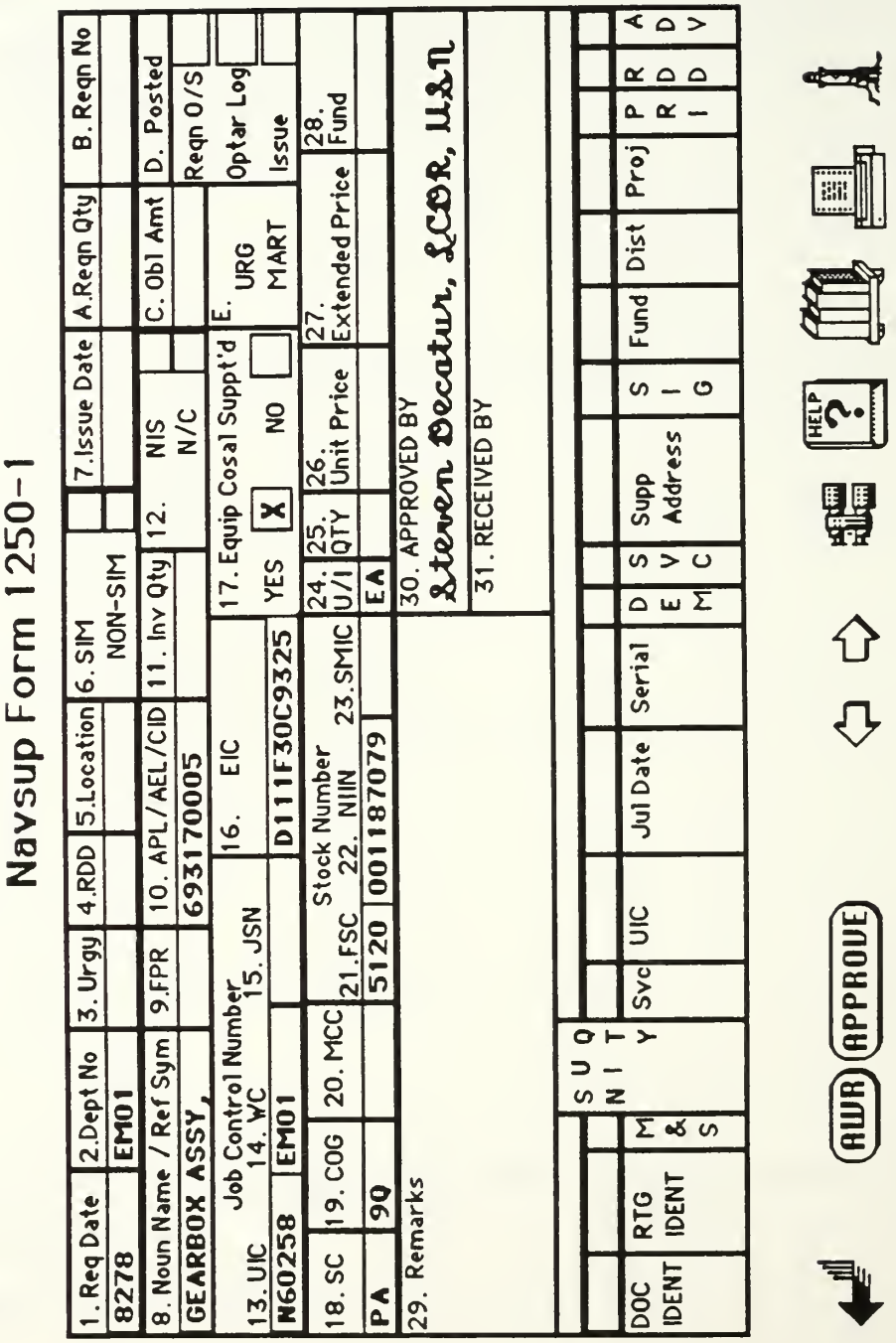
```
  go to previous card of BACKGROUND "HELP"
end  mouseUp

** BKGND #1, BUTTON #4: READ ************************************
on  mouseUp
   OPEN FILE "NPSCS:hypercardhelp"
 REPEAT

    repeat with count = 1 to 16
     READ FROM FILE "NPSCS:hypercardhelp" UNTIL RETURN
    IF IT IS EMPTY THEN EXIT MOUSEUP
     PUT IT AFTER background FIELD "TEXT1"
    end repeat

    repeat with count = 1 to 16
     READ FROM FILE "NPSCS:hypercardhelp" UNTIL RETURN
    IF IT IS EMPTY THEN EXIT MOUSEUP
     PUT IT AFTER background FIELD "TEXT2"
    end repeat
    domenu New Card

 END REPEAT
   CLOSE FILE "NPSCS:hypercardhelp"
end  mouseUp
```

```
** BKGND #1, BUTTON #5: Find ************************************
on mouseUp
   ask "Input the search string: "
  find it
  hide msg
  put "find" && it into msg
  hide msg
end mouseUp

** BKGND #1, BUTTON #6: PRINT ************************************
on mouseUp
   doMenu Print Card
end mouseUp

** BKGND #1, BUTTON #7: LIBRARY
****************************************
on mouseUp
  PLAY "LIBRARY"
   push card
  go to card library OF STACK "ARGOS"
end mouseUp

** BKGND #1, BUTTON #8: EXIT *************************************
on mouseUp
   gohome
```

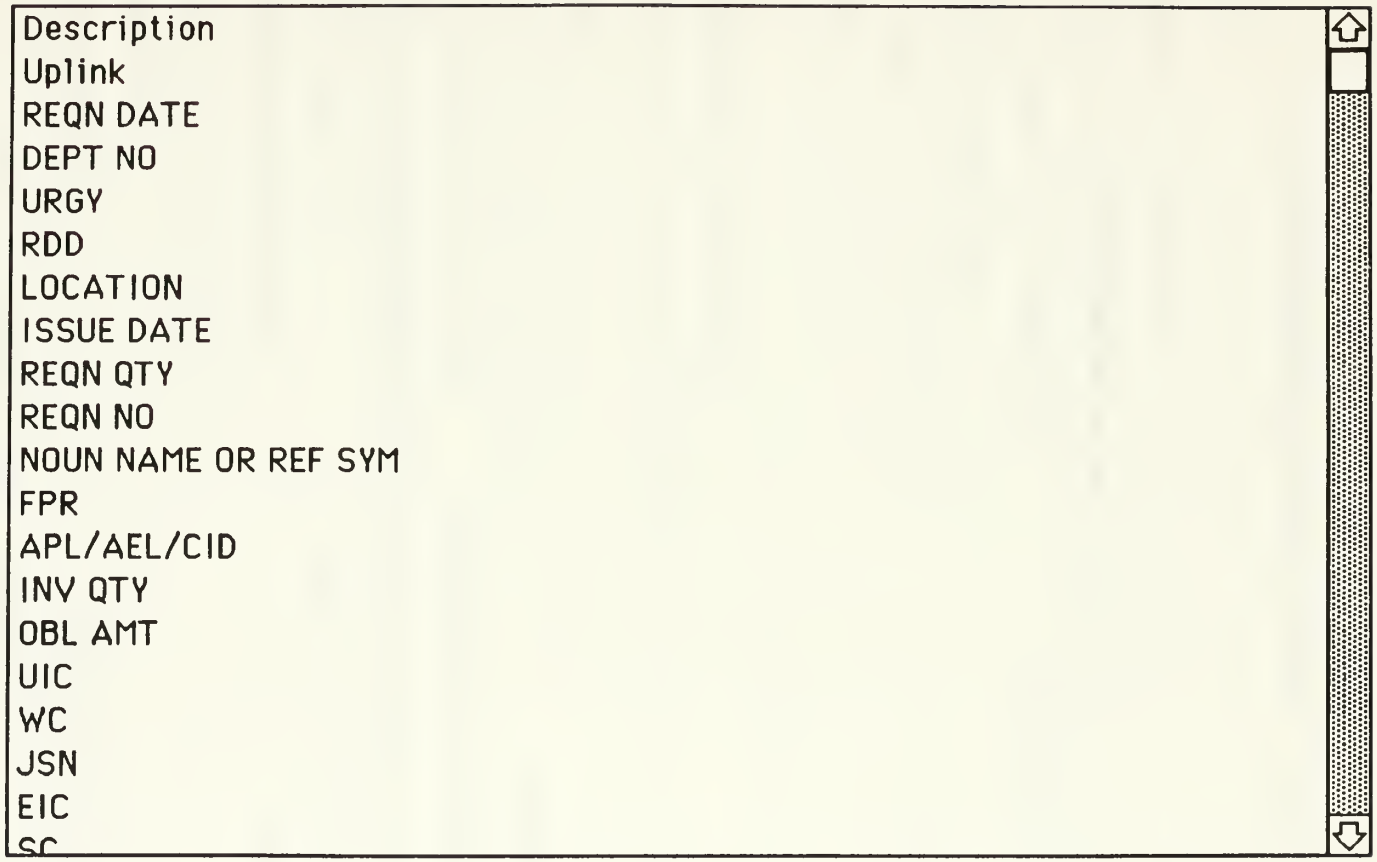
```
  go home
end  mouseUp
```

APPENDIX G

FORMS STACK



Select Field:



SCRIPTS FOR STACK: FORMS

```
** STACK SCRIPT ************************************

on closestack
  if the freesize of this stack > 0.15 * the size of this stack then
    doMenu "Compact Stack"
  end if
end closestack

on gohome
  play "BYE"
end gohome

function CLICKLINE
   return trunc(((scroll of the target)¬
  + (item 2 of the clickloc) - (item 2 of the rect of the target¬
  )) div the textheight of the target) + 1
end CLICKLINE

on ZAK
  put empty into fld "APPROVED"
  set the locktext of fld "APPROVED" to false
```

```
  click at the location of fld "APPROVED"
  type "Steven Decatur, LCDR, USN"
  set the locktext of fld "APPROVED" to true
end ZAK

on PIERRE
  put empty into fld "APPROVED"
  set the locktext of fld "APPROVED" to false
  click at the location of fld "APPROVED"
  type "J.P.Jones, LCDR, USN"
  set the locktext of fld "APPROVED" to true
end PIERRE

** BACKGROUND #1: NAVSUP 1250
************************************
on  returnkey
  if (char 1 to 11 of msg) = "find string" then
    put the id of this card into tempid
    if visible of field "Description" then
      set lockscreen to true
      send returnKey to Hypercard
    if tempid <> id of this card then
      go recent card
      hide card picture
      set visible of field "Description" to true
```

```
        repeat with i=1 to the number of buttons
        hide button i
       end repeat
         hide background button "sorry"
     end if
      set lockscreen to false
   else
       send returnKey to Hypercard
   end if
  else
     send returnKey to Hypercard
  end if
end  returnKey

** BKGND #1, FIELD #9: REQN QTY
*************************************
on  closefield
  if field "UNIT PRICE" is not empty then
   put field "UNIT PRICE" times field "REQN QTY" into field "OBL AMT"
  end if
end  closefield

** BKGND #1, FIELD #27: QTY **********************************
on  closefield
  if field "UNIT PRICE" is not empty then
```

```
    put field "UNIT PRICE" times field "QTY" into field "EXTENDED PRICE"
  end if
end  closefield

** BKGND #1, FIELD #28: UNIT PRICE
*************************************
on  closefield
   set numberFormat to ".00"
  if field "REQN QTY" is not empty then
    put (field "UNIT PRICE" * field "REQN QTY") into field "OBL AMT"
  end if
  if field "QTY" is not empty then
    put (field "UNIT PRICE" * field "QTY") into field "EXTENDED PRICE"
  end if
end  closefield

** BKGND #1, BUTTON #1: HELP ************************************
on  mouseUp
 PLAY "HELP"
   push this card
  go to stack "ARGOS HELP"
end  mouseUp

** BKGND #1, BUTTON #2: Find ************************************
on  mouseUp
```

```
 PUSH CARD
 PLAY "SEARCH"
 GO TO CARD ID 4322
end  mouseUp

** BKGND #1, BUTTON #3: LIBRARY
**************************************
on  mouseUp
  PLAY "LIBRARY"
   push  card
  go to card library OF STACK "ARGOS"
end  mouseUp

** BKGND #1, BUTTON #4: EXIT ************************************
on  mouseUp
   gohome
  go home
end  mouseUp

** BKGND #1, BUTTON #5: PRINT ************************************
on  mouseUp
   set visible of background btn "COVER" to true
   doMenu Print Card
   set visible of background btn "COVER" to false
end  mouseUp
```

```
** BKGND #1, BUTTON #6: Return
************************************
on  mouseUp
  play "RETURN"
   pop card
   play "Navigate Mode"
end  mouseUp

** BKGND #1, BUTTON #7: AUTOTAB
**************************************
on  mouseUp
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE TEXTSTYLE OF FIELD COUNT TO BOLD
 END REPEAT
end  mouseUp

** BKGND #1, BUTTON #8: APPROVE
************************************
on  mouseUp
  play "PASSWORD PLEASE"
end  mouseUp

** BKGND #1, BUTTON #9: COVER ************************************
on  mousewithin
```

```
 SET THE VISIBLE OF me TO FALSE
end  mousewithin
on  mouseenter
 SET THE VISIBLE OF me TO FALSE
end  mouseenter

** BKGND #1, BUTTON #10: AWR ************************************
on mouseUp
  set LockScreen to TRUE
  put field "NOUN NAME OR REF SYM" into NOUNNAME
  put field "WC" into WC
  put field "APL/AEL/CID" into APL
  put field "RDD" into DEADDATE
  put field "UIC" into UIC
  put field "JSN" into JSN
  put field "EIC" into EIC
  put field "REQN DATE" into REQNDATE
   go to stack equipment
   find APL in field "APL"
  put field "LOCATION" into LOCATION
   go to stack csmp
   go to first card of background "CSMP"
   domenu new card
  put DEADDATE into field "DEAD DATE"
  put REQNDATE into field "WD DATE"
```

```
  put REQNDATE into field "DEF DATE"
  put NOUNNAME into field "NOUN NAME"
   put APL into field "APL"
   put UIC into field "UIC"
  put WC into field "WC"
   put EIC into field "EIC"
   put JSN into field "JSN"
  put LOCATION into field "LOCATION"
   set LockScreen to FALSE
   select text of field "ID SER"
end  mouseUp

** BKGND #1, BUTTON #11: DELETE
************************************
on  mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
   repeat
     domenu delete card
    go next card
     if  number of this card = 1 then
      exit repeat
    end if
   end repeat
end  mouseUp
```

```
** BKGND #1, BUTTON #12: Prev ************************************

on mouseStillDown
  visual effect scroll left
  go to prev card of background "NAVSUP 1250"
end mouseStillDown

** BKGND #1, BUTTON #13: Next ************************************
on mouseStillDown
  visual effect scroll right
  go to next card of background "NAVSUP 1250"
end mouseStillDown

** BKGND #2, FIELD #1: FIELD CHOICE
************************************
on mouseup
  put line clickline() of me into FIELDNAME
  ask "Search Criteria?"
 POP CARD
  put "find" && quote & it & quote && "in field" && ¬
  quote & FIELDNAME & quote into msg
  hide msg
  send returnkey to HyperCard
end mouseup
```

APPENDIX H
DEVELOPERS SCRIPTS

The following scripts were used in the development of Argos. They are all contained in either hidden buttons or hidden fields in the Argos stacks. The first script is not contained in any stack, but may be used in a new button to show all hidden buttons.

** script to show ALL hidden background buttons **

```
on mouseUp
  repeat with count = 1 to number of background buttons
    show background button count
  end repeat
end mouseUp
```

```
**********************************************************************
Argos Stack Developers Scripts
**********************************************************************

** BKGND #1, FIELD #4: BUTTONS ***********************************
**  used to auto insert cardlink of an item into the field "DATA" **

on mouseup
 GLOBAL CARDID
  put CARDID into SECOND ITEM OF line¬
   (clickline()) of field "DATA"
 SET VISIBLE OF FIELD "BUTTONS" TO FALSE
   show card picture
 REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
    set visible of button COUNT to true
 END REPEAT
end mouseup

** BKGND #1, BUTTON #15: GRAPHICS REWRITE *************************
** This script rewites the script of all the graphic card buttons on a card **

on mouseup
 REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
    set the script of button COUNT to¬
     "-- Graphic Handler may be found in this cards background"¬
```

```
    & return & "On MouseUp" & return &¬
    "GRAPHIC (number of me)" & return & "end MouseUp"
  end repeat
end  mouseup

** BKGND #1, BUTTON #19: INSERT PARTNUMBER
**************************************
** This script rewites the script of card buttons on a card **
** The functionality of this script was not used in the final version of Argos
* *
on  mouseUp
 GLOBAL BUTTONNAME
 GLOBAL CARDID
 PUT EMPTY INTO BUTTONNAME
 PUSH CARD
 ASK "INPUT PARTNUMBER"
 GO TO STACK COSAL
 FIND IT IN FIELD "PART NUMBER"
 PUT SHORT ID OF THIS CARD INTO CARDID
 POP CARD
  hide  card  picture
 REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
    set visible of button COUNT to false
 END REPEAT
 IF FIELD "BUTTONS" IS EMPTY THEN
```

```
  REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
    PUT ((short name of CARD BUTTON COUNT) & "," & COUNT¬
   & RETURN) AFTER FIELD "BUTTONS"
  END REPEAT
 END IF
 ANSWER "PLEASE SELECT BUTTON NAME"
 SET VISIBLE OF FIELD "BUTTONS" TO TRUE
end  mouseUp

** BKGND #1, BUTTON #20: NONE,NONE
**************************************
** This script places NONE, NONE on all lines of the field DATA for intermediate
* *
**  development.  **

on  mouseUp
 ANSWER "ARE YOU SURE"
 IF IT <> "OK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
  PUT "NONE, NONE" INTO LINE COUNT OF FIELD "DATA"
 END REPEAT
end  mouseUp

** BKGND #1, BUTTON #21: SOMETHING,NONE
**************************************
```

```
** This script places NONE as the second element on all lines of the field DATA
* *
**  for  intermediate  development.  **

on  mouseUp
 ANSWER "ARE YOU SURE"
 IF IT <> "OK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF CARD BUTTONS
  PUT (CHAR 17 TO 25 OF LINE 4 OF THE SCRIPT OF BUTTON COUNT)¬
  & ", NONE" INTO¬
  LINE COUNT OF FIELD "DATA"
 END REPEAT
end  mouseUp

**  CARD  #4,  BUTTON  #1:  other  ************************************
**  This  script  was  used  to  demonstrate  graphics  **
**  for  intermediate  development.  **

on  mouseUp
   put  the  userLevel  into  saveLevel
   if  the  userLevel  <  3  then  set  userLevel  to  3  --  "Painting"
   if  the  userLevel  <  3  then  exit  mouseUp
   choose  lasso  tool
   get  the  loc  of  me
   put  first  item  of  it  into  Size1
```

```
  put second item of it  into second item of size1
  click at size1 with commandKey
  DoMenu copy picture
  set dragSpeed to 150
  drag from size1 to 238,130
  choose browse tool
  set userLevel to saveLevel
  pass mouseup
end  mouseUp

** CARD #4, BUTTON #2: SHIP ************************************

** This script was used to demonstrate graphics **
** for intermediate development. **

on  mouseUp
   put the userLevel into saveLevel
   if the userLevel < 3 then set userLevel to 3 -- "Painting"
   if the userLevel < 3 then exit mouseUp
   put the loc of button "missile" into misstart
  -- 394,246
   put first item of misstart into missileSize
   put second item of misstart into second item of missileSize
  repeat with i = 1 to 50
     set loc of button missile to 394,246-i
```

```
    end repeat
    set loc of button missile to missileSize
    choose lasso tool
    get the loc of me
    put first item of it into Size1
    put second item of it into second item of size1
    click at size1 with commandKey
    DoMenu copy picture
    set dragSpeed to 150
    drag from size1 to 75,248
    domenu flip horizontal
    drag from 75,248 to size1
    domenu flip horizontal
    choose browse tool
    set userLevel to saveLevel
    pass mouseup
end mouseUp
```

```
**********************************************************************
APL Stack Developers Scripts
**********************************************************************

** script to delete all but one card in APL  **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
   go first card of background apl
   repeat
     domenu delete card
     go next card of background apl
     if (short id of first card of background apl) =¬
     (short id of last card of background apl) then
      exit repeat
    end if
   end repeat end mouseUp

** script to read data from a text file into the APL stack **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
```

```
PUT "NPS-CS:THESIS:RAW DATA:APL" INTO FILENAME
 open file FILENAME
 repeat 236 times
   read from file FILENAME for 1107
 end repeat
 repeat 2379 times
   read from file FILENAME for 1107
    put (char (1) to (11) of it) into field "APL"
    put (char (12) to (59) of it) into field "NOMENCLATURE"
    put (char (60) to (68) of it) into field "NIIN"
    put (char (69) to (1054) of it) into field "CHARACTERISTICS"
    put (char (1055) to (1059) of it) after field "FSCM"
    put (char (1060) to (1061) of it) into field "AINAC"
    put (char (1062) to (1063) of it) into field "LSSC"
    put (char (1064) of it) into field "COG"
    put (char (1065) to (1066) of it) into field "ID"
   put (char (1067) of it) into field "AEL COL NUM"
    put (char (1068) of it) into field "FLAG"
    put (char (1069) to (1073) of it) into field "DATE"
    put (char (1074) to (1076) of it) after field "USER"
    put (char (1077) to (1080) of it) into field "EQU CNT"
    put (char (1081) to (1086) of it) into field "POPULATION"
    put (char (1087) of it) into field "SECTION"
    domenu New Card
 end repeat
```

```
  close file FILENAME end mouseUp

** script to toggle between locking and unlocking text fields in the APL stack
* *

on  mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
  END REPEAT end mouseUp
```

```
**********************************************************************
COSAL Stack Developers Scripts
**********************************************************************

** script to read data from a text file into the Cosal stack **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 put "NPS-CS:THESIS:RAW DATA:COSAL" into FILENAME
  open file FILENAME
  repeat 16429 times
    read from file FILENAME for 99
  end repeat
  repeat 5 times
    read from file FILENAME for 99
    if it is empty then
      exit repeat
   end if
    put (char (1) to (11) of it) into field "APL"
    put (char (12) to (41) of it) into field "PART NUMBER"
    put (char (42) to (46) of it) into field "FSCM"
    put (char (47) to (61) of it) into field "STOCK NUMBER"
    put (char (62) to (80) of it) into field "ITEM NAME"
```

```
    put (char (81) to (82) of it) into field "COG CODE"
    put (char (83) to (84) of it) into field "UNIT OF ISSUE"
    put (char (85) of it) into field "MATL CONTROL CODE"
    put (char (86) of it) into field "SPCL MATL CONTENT CODE"
    put (char (87) to (89) of it) into field "QUANTITY/APPLICATION"
    put (char (90) to (91) of it) into field "SOURCE CODE"
    put (char (92) to (93) of it) into field "MAINTENANCE CODE"
    put (char (94) of it) into field "RECOVERABILITY CODE"
    put (char (95) of it) into field "ALLOWANCE NOTE CODE"
    put (char (96) to (98) of it) into field "ALLOWED QUANTITY"
    put (char (99) of it) into field "MEC-PT-TO-CMPNT"
    domenu New Card
  end repeat
  close file FILENAME end mouseUp

** script to delete all but one card in COSAL **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
  repeat
    domenu delete card
    go next card
    if number of this card = 1 then
     exit repeat
```

```
    end if
   end repeat end mouseUp


** script to toggle between locking and unlocking text fields in the COSAL
stack **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
  END REPEAT end mouseUp

** script to delete all but one card in COSAL **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
   repeat
     domenu delete card
    go next card
    if number of this card = 1 then
     exit repeat
    end if
```

```
  end repeat end mouseUp
```

** script to toggle between locking and unlocking text fields in the COSAL stack **

```
on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
  END REPEAT end mouseUp
```

** script to change all fields to textstyle bold in the COSAL stack **

```
on mouseUp
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE TEXTSTYLE OF FIELD COUNT TO BOLD
  END REPEAT end mouseUp
```

```
**********************************************************************
CSMP Stack Developers Scripts
**********************************************************************

** script to delete all but one card in CSMP stack  **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
  repeat
    domenu delete card
    go next card
    if  number of this card = 1 then
     exit repeat
   end if
  end repeat end mouseUp

** script to toggle between locking and unlocking text fields in the CSMP stack
* *

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
```

```
SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
END REPEAT end mouseUp
```

```
**********************************************************************
EQUIPMENT Stack Developers Scripts
**********************************************************************

** script to delete all but one card in EQUIPMENT stack **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
  go first card of background apl
  repeat
    domenu delete card
    go next card of background apl
    if (short id of first card of background apl) =¬
    (short id of last card of background apl) then
     exit repeat
   end if
  end repeat end mouseUp

** script to toggle between locking and unlocking text fields in the equipment
stack **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
```

```
 IF IT <> "ZAK" THEN EXIT MOUSEUP
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE LOCKTEXT OF FIELD COUNT TO NOT LOCKTEXT OF FIELD COUNT
  END REPEAT end mouseUp

** script to read data from a text file into the EQUIPMENT stack **

on  mouseUp
  ASK "WHAT IS THE PASSWORD? "
  IF IT <> "ZAK" THEN EXIT MOUSEUP
  PUT "NPS-CS:THESIS:RAW DATA:equipment" INTO FILENAME
  open file FILENAME
   repeat 7420 times
     read from file FILENAME for 303
   end repeat
   repeat 10 times
     read from file FILENAME for 303
     put (char (1) to (10) of it) into field "SWLIN"
     put (char (11) to (21) of it) into field "APL"
     put (char (22) to (26) of it) into field "PRIME KEY"
     put (char (27) to (46) of it) into field "ESD"
     put (char (47) to (66) of it) after field "ESN"
     put (char (67) to (78) of it) into field "EIC"
     put (char (79) to (93) of it) into field "SN"
     put (char (94) to (105) of it) into field "LOCATION"
```

```
put (char (106) to (109) of it) into field "WC EQPT"
put (char (110) to (113) of it) into field "WC CMPT"
put (char (114) to (139) of it) after field "TYPE/MODEL"
put (char (140) to (187) of it) into field "FUNCTIONAL DESC"
put (char (188) to (192) of it) into field "PHM RIN"
put (char (193) to (193) of it) into field "SEL EQPT IND"
put (char (194) to (203) of it) into field "SAC"
put (char (204) to (204) of it) into field "SVC IMP"
put (char (205) to (205) of it) after field "MIL ESSENTIAL"
put (char (206) to (216) of it) into field "PARENT APL"
put (char (217) to (231) of it) into field "PARENT SN"
put (char (232) to (232) of it) into field "CAT"
put (char (233) to (239) of it) into field "SCAT"
put (char (240) to (245) of it) into field "QTY APPL"
put (char (246) to (247) of it) after field "DATA ORIG"
put (char (248) to (248) of it) into field "INSTAL STATUS"
put (char (249) to (250) of it) into field "VAL SPC ACT"
put (char (251) to (251) of it) into field "RNV"
put (char (252) to (258) of it) into field "AILSIN"
put (char (259) to (266) of it) into field "UPDATE JCN"
put (char (267) to (271) of it) after field "LAST UPDATE"
put (char (272) to (274) of it) into field "LAST UPDATE USER"
put (char (275) to (283) of it) into field "UIC"
put (char (284) to (284) of it) into field "TYPE #"
put (char (285) to (295) of it) into field "FLAGS"
```

```
    put (char (296) to (300) of it) into field "FSCM"
    put (char (301) to (301) of it) after field "AEL COL #"
    put (char (302) to (302) of it) into field "COS TYPE"
    put (char (303) to (303) of it) into field "ALT FLAG"
    domenu New Card
  end repeat
  close file FILENAME end mouseUp

** script to change all fields to textstyle bold in the EQUIPMENT stack **

on mouseUp
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE TEXTSTYLE OF FIELD COUNT TO BOLD
 END REPEAT end mouseUp

** script to delete all but one card in FORMS stack  **

on mouseUp
 ASK "WHAT IS THE PASSWORD? "
 IF IT <> "ZAK" THEN EXIT MOUSEUP
  repeat
    domenu delete card
    go next card
    if  number of this card = 1 then
     exit repeat
```

```
 end if
end repeat end mouseUp
```

```
**********************************************************************
FORMS Stack Developers Scripts
**********************************************************************

** script to change all fields to textstyle bold in the FORMS stack **

on mouseUp
 REPEAT WITH COUNT = 1 TO NUMBER OF FIELDS
  SET THE TEXTSTYLE OF FIELD COUNT TO BOLD
  END REPEAT end mouseUp
```

BIBLIOGRAPHY

Akscyn, R. M., McCracken, D. L., Yoder, E., "KMS: A Distributed Hypermedia System for Managing Knowledge in Organizations," *Communications of the ACM*, v. 31 no. 7, July 1988.

Anderson, J., Fishman, B., "The Smalltalk Programming Language: An Introduction to Object-Oriented Programming," *Byte*, August 1986.

Apple Computer, Inc., *HyperCard™ User's Guide*, 1987.

Apple Professional Developers Association (APDA), *HyperCard™ Script Language Guide, The HyperTalk™ Language*, Draft, 11 August 1987.

Brodie, M. L., "On the Development of Data Models," In: Brodie, M. L. Mylopoulos, J., Schmidt, J. W., editors. *On Conceptual Modelling*. Springer-Verlag, New York, 19-47, 1984.

Chickering, J. E., "The Advent of the Paperless Ship," *Naval Engineers Journal*, May 1988.

Conklin, J., "Hypertext: An Introduction and Survey," *IEEE*, September 1987.

Conklin, J., "Hypertext: An introduction and Survey," *IEEE Computer*, September 1987.

Dadam, P., and others, "A DBMS Prototype to Support Extended NF2 Relations: An Integrated View on Flat Tables and Hierarchies," 1986 ACM 0-89791-191.

Goldberg, A., and Robson, D., *Smalltalk-80: The Language and its Implementation*, Reading, MA, Addison-Wesley, 1983.

Goldberg, A., Robson, D., "Smalltalk Programming Language," *Software World*, pp. 2-10, 1983.

Goodman, D., *The Complete HyperCard™ Handbook*, Bantam Computer Books, 1987.

Gruendig, L., Pistor, P.: *Landinformationssysteme und ihre* Anforderungen an Datenbank-schnittstellen.

Haerder, T., Reuter, A., *Database Sytems for Nonstandard Applications Proc.* Int. Computing Symposium (H.J. Schneider, ed.), Erlangen, West Germany, March 1983, Teubner-Verlag, Stuttgart, pp. 452-466.

Halasz, F. G., "Reflections on NoteCards: Seven Issues for the Next Generation of Hypermedia Systems," Communications of the ACM, v. 31, no. 7, July 1988.

Harvey, G., *Understanding HyperCard for Version 1.1*, SYBEX, 1988.

Kaehler, T., Patterson, D., "Small Taste of Smalltalk," *Byte*, August 1986.

Kent, William, "Limitations of Record-Based Information Models," *ACM Transactions on Database Systems*, v. 4, no. 1, March 1979.

Korth, H. F., Silberschatz, A., *Database System Concepts*, McGraw-Hill, 1986.

Korth, H. F., Silberschatz, A., *Database System Concepts* , Mcgraw-Hill Book Company, 1986.

Lorie, R. A., "Issues In Databases for Design Applications," *File Structures and Data Bases for CAD*, North-Holland Publishing Company, IFIP, 1982.

Lorie, R., and others, *User Interface and Access Techniques for Engineering Databases*, IBM Research Laboratory, San Jose, CA 95193, 1984.

MacLennan, B. J., *Principles of Programming Languages*, New York, Holt, Rinehart and Winston, 1987.

McCracken D. L., Akscyn, R. M., "Experience with ZOG Human-Computer Interface System," *Int J. Man-Machine Studies* , v. 21, 293-310, 1984.

Perkins, R. C., "Data Analysis The Key to Data Base Design," *QED* Information Sciences, Inc. Wellesley, MA, 1984.

Ruff, D., LCdr, USN, from: "The Advent of the Paperless Ship," *Naval Engineers Journal*, July 1988, pp 157-159.

Schek, H. J., Pistor, P., "Data Structures for an Integrated DataBase Management and Information retrieval System," *Proceedings of the Eighth International Conference on Very Large Data Bases*, Mexico City, September, 1982.

Schumaker, K. J., Fishman, B., "Object Oriented Language for the Macintosh: An Overview of the Languages and Their Capabilities," *Byte*, August 1986.

Shafer, D., *HyperTalk ™ Programming*, first edition, 1988.

Suzuki, N., "Smalltalk-80: An Object-Oriented Language," *Systems and Control*, January 1985.

Wu, C. T. , "An Effect of Set Type To Query Formulation in Relational DataBase Systems," Unpublished Manuscript, Naval Postgraduate School, Monterey, CA 93943.

Wu, C.T., "An Effect of Set Type To Query Formulation in Relational DataBase Systems," Unpublished Manuscript, Naval Postgraduate School, Monterey, CA 93943.

INITIAL DISTRIBUTION LIST

1.	Defense Technical Information Center Cameron Station Alexandria, VA 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, CA 93943-5002	2
3.	Office of Naval Research Office of the Chief of Naval Research Attn: CDR Michael Gehl, Code 1224 800 N. Quincy Street Arlington, Virginia 22217-5000	1
4.	Space and Naval Warfare Systems Command Attn: LCDR Topperoff Nation Center 1, Room 11N08 2511 Jefferson Davis Hwy Washington, D.C. 20363-5100	1
5.	Office of the Chief of Naval Operations Attn: Capt Don Rhodes,USN Code OP-403 Washington, D.C. 20350-2000	1
6.	Department of the Navy Naval Sea Systems Command Attn: Mr. Clifford Gieger Code: Cheng L Washington, D.C. 20362-5101	1
7.	Office of the Secretary of Defense Attn: CDR Barber,USN STARS Program Office Washington, D.C. 20301	1
8.	Office of the Secretary of Defense Attn: Mr. Joel Trimble STARS Program Office Washington, D.C. 20301	1

9. Commanding Officer — 1
Naval Research Laboratory
Code 5150
Attn: Dr. Elizabeth Wald
Washington, D.C. 20375-5000

10. Navy Ocean System Center — 1
Attn: Linwood Sutton, Code 423
San Diego, California 92152-5000

11. National Science Foundation — 1
Division of Computer and Computation Research
Washington, D.C. 20550

12. Department of the Navy — 1
Naval Sea Systems Command
Attn: Mr. Phil Styles Code: CEL-TD1
Washington, D.C. 20362-5101

13. Department of the Navy — 1
Naval Sea Systems Command
Attn: Mr. Mike Mehalic Code: CEL-PAB
Washington, D.C. 20362-5101

14. Office of Naval Research — 1
Computer Science Division, Code 1133
Attn: Dr. Van Tilburg
800 N. Quincy Street
Arlington, Virginia 22217-5000

15. David W. Taylor Naval Ship R&D Center — 1
Attn: Mr J. Hawkins Code: 1740.2
Bethesda, Maryland 20084-5000

16. Navy Management Systems Support Office — 1
Detachment Pacific
Attn: Mr. Lyle Rich Code: 311
Naval Station Box 217
San Diego, California 92136-5217

17. Commander Naval Security Group Command — 2
Code: G-30
3801 Nebraska Ave. NW
Washington, D.C. 20390-5211

18. Commanding Officer 1
U.S. Naval Security Group Activity
Misawa, Japan
APO San Francisco, Ca. 96519-0006

19. Commanding Officer 2
USS Jarrett (FFG-33)
Attn: CDR B.B. Giannotti, USN
FPO San Francisco, California 96519

20. U.S. Naval Security Group Activity 2
Attn: Lt. Kevin F. Duffy, USN
Misawa, Japan
APO San Francisco, Ca. 96519-0006