DOCUMENT RESUME

ED 118 154 IR 003 074

AUTHOR Hornbeck, Frederick W., Brock, Lynn
TITLE The Graphics Terminal Display System; a Powerful General-Purpose CAI Package.
INSTITUTION San Diego State Univ., Calif.
SPONS AGENCY Navy Personnel Research and Development Center, San Diego, Calif.
REPORT NO NPRDC-TR-76-25
PUB DATE Dec 75
NOTE 65p.; Technical Report, June 1974-April 1975

EDRS PRICE MF-$0.83 HC-$3.50 Plus Postage
DESCRIPTORS Computer Assisted Instruction; *Computer Graphics; *Computer Programs; Computers; Instructional Technology
IDENTIFIERS GRAIL; Graphic Display Systems; Navy

ABSTRACT

The Graphic Terminal Display System (GTDS) was created to support research and development in computer-assisted instruction (CAI). The system uses an IBM 360/50 computer and interfaces with a large-screen graphics display terminal, a random-access slide projector, and a speech synthesizer. An authoring language, GRAIL, was developed for CAI, and it is transportable to other computers. Comparisons are made between this system and others as PLATO, PLANIT, and TICCIT. (Author/CH)

I.R

ED118074

NAVY PERSONNEL RESEARCH AND DEVELOPMENT CENTER SAN DIEGO, CALIFORNIA 92152

NPRDC TR 76-25 DECEMBER 1975

THE GRAPHICS TERMINAL DISPLAY SYSTEM A POWERFUL, GENERAL-PURPOSE CAI PACKAGE

Frederick Wm. Hornbeck
Lynn Brock

APPROVED FOR PUBLIC RELEASE;
DISTRIBUTION UNLIMITED.

NPRDC TR 76-25 December 1975

THE GRAPHICS TERMINAL DISPLAY SYSTEM
A POWERFUL, GENERAL-PURPOSE CAI PACKAGE

Frederick Wm. Hornbeck

and

Lynn Brock

San Diego State University

This research was supported in part by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Navy Personnel Research and Development Center under contract No. N61339-73-C-0184.

Prepared for

Navy Personnel Research and Development Center
San Diego, California 92152

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NPRDC TR 76-25	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) THE GRAPHICS TERMINAL DISPLAY SYSTEM: A POWERFUL GENERAL-PURPOSE CAI PACKAGE		5. TYPE OF REPORT & PERIOD COVERED Technical Report June 1974 - April 1975 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) F. W. Hornbeck L. Brock		8. CONTRACT OR GRANT NUMBER(s) N61339-73-R-184
9. PERFORMING ORGANIZATION NAME AND ADDRESS San Diego State University San Diego, California 92182		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 62763N PF55.522.002.01.60
11. CONTROLLING OFFICE NAME AND ADDRESS Navy Personnel Research and Development Center San Diego, California 92152		12. REPORT DATE December 1975 13. NUMBER OF PAGES 72
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office) Navy Personnel Research and Development Center San DIego, California 92152		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Prepared in cooperation with Navy Personnel Research and Development Center.

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computer-assisted instruction, graphic presentation methods, instructional technology, CAI systems, CAI authoring

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The report describes a system developed to support research and development in computer-based instruction. A powerful and versitile CAI language was developed which allows authors to present materials on a graphic display, on slides, or by means of voice synthesizer. The language was developed on an IBM 360/50 computer and is transportable to other similar machines. Comparisons are made between this system and others, such as PLANIT, PLATO, and TICCIT.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

FOREWORD

This work was supported through a contract with the San Diego State University under Exploratory Development Task Area PF55.522.002 (Methodology for Developing/Evaluating Navy Training Program), Work Unit Number PF55.522.002.01.60 (Advanced Computer Based Research). This work unit is jointly guided by the Advanced Research Projects Agency (Account Symbol 9740400.1311) and the Navy Personnel Research and Development Center (NAVPERSRANDCEN). The project was initiated in response to the requirement for "improvements in training methodologies, measurement techniques, management and administration, including decision criteria required for their rapid implementation" contained in General Operational Requirement 43, Revised 10/71.

Dr. William E. Montague was the technical monitor for NAVPERSRANDCEN in this effort. The work was conceptualized originally in a proposal submitted by Dr. F. W. Hornbeck and Dr. J. R. Levine during the spring of 1973. The developmental effort was the responsibility of Mr. L. Brock assisted by Ms. P. Lamb.

J. J. CLARKIN
Commanding Officer

SUMMARY

Problem

A need existed for a computer-assisted instruction (CAI) system capable of supporting varied research needs. Such a system is especially needed for investigations of the value of graphics displays and voice synthesization. The system would utilize widely available computer services, and standard terminal hardware so that the CAI system would be readily transportable.

Purpose

The purpose of this report is to describe the Graphic Terminal Display System (GTDS), which was designed expressly to meet the R&D requirements of the Navy Personnel Research and Development Center (NAVPERSRANDCEN).

Approach

A Graphics Terminal Display System (GTDS) was developed by the San Diego State University under contract with NAVPERSRANDCEN. The system utilizes an IBM 360/50 computer and interfaces with a large-screen graphics display terminal, a random-access slide projector, and a speech synthesizer. An authoring language, Graphics Assisted Instructional Language (GRAIL), was developed to allow lessonware production for the GTDS. Therefore, the author can present information to a student by means of a visual alphanumeric display, computer graphics, photographs, or synthesized speech. Student responses are transmitted from crosshair cursors on the display or from the keyboard of the terminal. GRAIL has an instruction set making it a versatile CAI author language. Control statements, a FORTRAN subroutine capability, and specific instructions for the flexible utilization of all student station devices combine into a complete closed system.

Findings and Conclusions

The GTDS provides a powerful capability for research support. It compares favorably with other CAI systems such as PLATO, PLANIT and TICCIT, and is transportable to virtually any IBM 360 or 370 computer system. The widespread availability of such computer systems could make GTDS an attractive alternative for CAI applications.

Recommendations

It is recommended that the GTDS be further enhanced so that it can be used for a broad range of Navy training and R&D applications.

CONTENTS

INTRODUCTION

Problem

A need existed for a computer-assisted instruction (CAI) system capable of supporting varied research needs. Such a system is especially needed for investigations of the value of graphics displays and voice synthesization. The system would utilize widely available computer services, and standard terminal hardware so that the CAI system would be readily transportable.

Purpose

The Graphics Terminal Display System (GTDS) was designed under contract with the San Diego State University expressly to support research and development (R&D) activities of the Navy Personnel Research and Development Center (NAVPERSRANDCEN) in advanced computer-assisted instruction (CAI). It is, however, a system of broad applicability and can be relatively easily transported to any installation having access to IBM System 360 or 370 hardware and software support. This report contains a discussion of the functional characteristics of the author language and its software support, a detailed description of the present hardware implementation at the San Diego State University, and comments on the continued development of the system. A complete specification of the Graphics Assisted Instructional Language (GRAIL), the author language, and other external documentation are included in the appendix.

Major requirements of NAVPERSRANDCEN researchers reflected in the design of the GTDS are sophisticated graphic display capabilities, vocal output, and photographic slide projection. Specific commands for the flexible utilization of the devices which provide these capabilities have been incorporated in the author language, GRAIL, a powerful programming language designed expressly for the preparation of computer-assisted instructional material.

SYSTEM DESCRIPTION

The Author Language and Software Support

All GTDS software is carefully structured, with each functional unit isolated in its own explicit module. This, along with extensive documentation in the form of comment statements in programs and adequate system descriptions, will guarantee easy maintenance, modification, and transportation of the package. GTDS is written primarily in IBM System 360 Assembly Language, with some routines written in PL/I.

Major GTDS facilities are the compiler for GRAIL, the author language and Extract and Collect Logically Indicated User Records (EXCALIBUR), a subsystem for the collection of data required for the analysis of student performance. The GRAIL author language includes the following features:

. Structured programming support, which allows the author of CAI materials to check his materials during development, rather than having to wait until they are tested or in general use to see whether they are correct.

. Reentrant code, which allows several students to be "taught" using the same machine instructions, thereby saving storage.

. Expressions (character and arithmetic), which are allowed virtually anywhere a variable can occur.

. Extensible language facility. The macro definition language (MDL) is an integral part of GRAIL. It allows the author to add operations and constructs to the language, and gives him virtually complete freedom in selecting basic operations to be used in teaching a course.

. Dynamic storage allocation. Main (core) storage is allocated only when the structure requiring it is entered, thus reducing main storage use.

. Portability. All code dependent on the operating system is isolated in one area, thus allowing easy conversion to any operating system that runs on the IBM 360-370 series of computers.

. Generation of "machine code" (computer instructions). All machine code is generated by GRAIL, which is typically five to ten times faster in execution than "interpretive" systems.

EXCALIBUR allows author-controlled recording at any time, in addition to providing automatic recording of student log-on and log-off times. Since records are compatible with such languages as FORTRAN, PL/1, COBOL, and ASSEMBLER, data can be analyzed on any machine supporting magnetic tape and these languages. Finally, records can be easily selected for further analysis through very simple PL/1 programs.

Other components of the total system are routines for run-time support of GRAIL courseware and the Votrax speech synthesizer dictionary file (VDF) which contains the Votrax phonemic encoding for the dictionary entries.

One of the principal requirements has been for GRAIL to support high-quality graphics for CAI research. The GTDS has a sophisticated graphics capability which provides the user great flexibility in formatting and selecting sizes for graphics displays with little programming effort. A detailed description of this capability is contained in the Appendix, Section 5 on the Terminal Control System (TCS). The TCS allows the user to modify the scale of a displayed figure. A dimension of the display can be magnified or shrunk using a simple command. In addition, a feature called "clipping" allows the user to designate which part of a figure should actually be drawn on the screen. Figures can be rotated as needed. Another feature called "windowing", allows the user to display any part of a figure on the entire screen, or display a figure on any part of the screen, thereby leaving room for presenting other information. These capabilities are found in no other CAI system. Extensive additional flexibility exists for the GRAIL author through the use of the FORTRAN subroutines described in the TCS documentation. The GRAIL instruction 'FCALL' permits the courseware author to use these and any other FORTRAN subroutines known to the system. This ability permits the author access not only to routines written specifically for this system but to many existing graphing and plotting packages as well.

As the language evolved, many other capabilities were incorporated into what is now a very powerful, general-purpose CAI language. The number of concepts (instructions) in GRAIL is around thirty and there are about the same number of GRAIL functions. They are explicitly defined in Section 2 on language specifications and Section 3 on GRAIL functions in the appendix. Much of the power of the language is provided by the ability to call FORTRAN subroutines and by the inclusion of the Macro Definition Language (MDL) which is described in Section 6 of the appendix. MDL allows the GRAIL author to create new concepts or instructions on an ad lib basis. For instance, the MDL could be used to add a desk calculator mode which a student could use to calculate an answer even though the author did not anticipate that he would need to do the calculation. Recursion is permitted in MDL; the definition of a macro may reference itself.

The highly sophisticated, ALGOL-like control features of GRAIL provide a richness of program flow options not found in other CAI languages. The IF-THEN-ELSE, DO-WHILE/UNTIL, DOINC/DODEC, and CASES constructs (see Section 2.2 of the appendix) provide the courseware author unlimited freedom. The ESCAPE and LIMIT constructs provide safeguards for the wary.

These characteristics make GRAIL ideally suited to R&D activities in CAI. Not only is there abundant flexibility for courseware development, but there is also the capability to emulate or simulate other author languages (using MDL) should that be desirable.

As discussed in Section 1 on concepts and facilities in the appendix, GTDS is designed to run GRAIL courses with a high degree of machine efficiency in terms of both storage and time. Those features which contribute most to this are the fact that GRAIL courses are compiled rather than interpreted, the machine code produced by the compiler is re-entrant, storage allocation is dynamic, and GRAIL course sections are limited to 4K (4096) bytes of compiled code.

This last limitation (on the size of a section) does not limit the size of a course or lesson because a course section may invoke any number of other (non-course) sections. A course (or lesson) consists of one course section which remains loaded throughout the session in which it is being used and any number of other sections which are in core only during execution by the course section of one or more students. Data needed in more than one section of a course are always available if defined at the course level. This organization has the desirable side effect of forcing course authors to employ structured programming strategies. An important benefit of this is the facilitation of section sharing across courses. Once a section describing the terminal keyboard for the student has been written, for instance, it will be available for inclusion in the introductory lesson of any other course.

To facilitate the use of GTDS, a GRAIL course to teach GRAIL programming is now being prepared. This package will contain an instructional portion as well as a reference component. The reference material will consist of the formal definitions of GRAIL instructions as in the appendix of this report but will also have parallel descriptions at a lower level of abstraction and examples of the application of each instruction, function, and other construct.

Present Hardware Implementation

All of the GTDS hardware except the Student Station Interface are 'off the shelf' items that are readily available and in relatively wide use, though not necessarily in CAI applications. The only hardware innovation was the combination of these devices into one system to provide a single, integrated student station.

Whenever possible in the development of software or selection of hardware options, established standards (such as the ASCII character set) and industry conventions (such as the RS-232-C communications hardware interface) have been employed.

The Student Station

The primary component of the student station is a Tektronix 4014 Graphics Display Terminal. This device provides visual alphanumeric and graphic program output to the student. Vendor-supplied and contractor-developed software support four different character font sizes for alphanumeric display on the 11" x 15" direct-view storage tube display screen. The full ASCII upper and lower case character set is supported. Substantial support for sophisticated two-dimensional graphic output is provided. The terminal keyboard and thumb-wheel-controlled cross-hair cursors constitute the mechanisms for student input to the system. A Tektronix 4610 hard copy unit allows for the generation of copy upon command from the display terminal, the hard copy unit, or the computer.

Additional visual display capability is provided by the inclusion of a Kodak RA-960 random-access slide projector in the student station. The projector may be powered on or off, and any one of the 80 slides may be randomly selected for projection under program control by the GTDS hardware and software interface.

Audio response capability is supplied by a Votrax model 6 voice synthesizer. This device generates a set of 63 American English male phonemes with appropriate interphonemic transitions from digital input. GTDS support allows the GRAIL author to generate speech output using phonemic literals, post-compilation but pre-execution dictionary lookup of previously encoded words, or longer, previously encoded passages. System hardware and software components bring speech rate, pitch, and volume under GRAIL author control.

The Student Station Interface (SSI) was designed and fabricated by Sensors, Data, Decisions, Inc. (SDD) of San Diego. The SSI is built on one circuit board mounted in the pedestal of the Tektronix terminal and plugged into the accessory mother board of that device and one circuit board installed in the Votrax speech synthesizer. It contains all the logic for device selection for the computer output data stream, control of the slide projector, and control of the rate, pitch, and speed of the vocal output, and provides device status checking capabilities. It responds to communication rate selection via the external switches of the Tektronix terminal (110, 150, 300, 600, 1200, 1800, 2400, 4800, and 9600 baud). Its installation required minimal alteration of the three basic student station devices. The SSI may be completely removed from the circuit via one of the option switches of the Tektronix terminal keyboard.

Communications

Basic system design requirements call for the student stations to be remote from the computer on which the software resides. This condition is almost always apt to obtain when applications require support of graphics but the number of terminals and level of usage do not justify dedication of a computer to the CAI system which is sufficient for graphics. It will also be so whenever terminals of the same system must be located at two or more distinct locations. In the case of the present implementation, the computer is on the campus of the San Diego State University; terminals are likely to be located at the University, the Naval Training Center (NTC), San Diego, and other sites.

Considering the small number of terminals currently required, major expenditures for high-speed communications channels cannot be justified. Consequently, the communications system has been designed to utilize voice grade (i.e., unconditioned) 4-wire leased lines. These lines are currently $14.00 per month in the San Diego area, which, combined with their low error rate, makes them highly competitive with ordinary, 2-wire dial lines for this application. These lines have an 1800 baud, full duplex capability, although they are currently running at 1200 baud due to equipment limitations at the computer end. This rate is at or near the minimum for any appreciable use of graphics but is satisfactory to support all other aspects of the GTDS. With the exception of the modems and data access arrangements to be described immediately below, all existing hardware in the GTDS can operate at rates up to 9600 baud when used in remote locations. The Tektronix terminal can be driven at much higher rates--about 50K baud--when wired directly to a computer.

Maximum portability of the student stations would be obtained if accoustical couplers sufficient to the 1200 baud, full duplex, data rate were obtainable. If they were, a student station could be conveniently installed at any location where 110 volt a.c. power and an ordinary telephone hand set were available. Although there have been allusions to 1200 baud acoustical couplers in trade journals, we are not aware of the availability of one of demonstrated reliability and reasonable cost.

Consequently, given that leased lines of the direct-dial network are to be used, the only remaining communications interfaces available are Bell System Data Sets or commercially supplied modems used in conjunction with Bell System data access arrangements (DAAs). Tariffs for Bell System 1200 baud data sets are quite high (about $45.00 per month for the CDT set required at the terminal end and $75.00 per month for the CBT set required at the computer end). Hence, we decided to employ purchased modems. The ones selected are General DataCom 202-5A sets which can be used at both ends and cost $361.00 apiece.

The University's Data General Nova 1220 computer serves as a hardware and software I/O interface for the GTDS System. The Nova is attached to the main computer's (an IBM System 360/50) multiplexor channel. System software support in the Nova provides for the intraline editing of textual material as well as good, relatively inexpensive I/O interfacing to the 360. A schematic diagram of the present GTDS implementation is presented in Figure 1.

The Computer

The computer presently serving as host for the GTDS is the IBM System 360/50 at the San Diego State University. This machine currently has 384K bytes of core memory. The operating system is IBM's Disk Operating System. The normal operating configuration provides three partitions for multitask servicing of user programs. The background partition is the largest and handles the bulk of student and faculty jobs. The smaller foreground partitions are used primarily for administrative applications programs (F2) and for POWER (F1), which drives the Remote Job Entry/Remote Job Output terminals on campus. When GTDS is loaded, it resides in 64K bytes in F1 along with POWER. There is only one other on-line system run at this installation so that it is not inconvenient to provide a very high interrupt priority to GTDS.

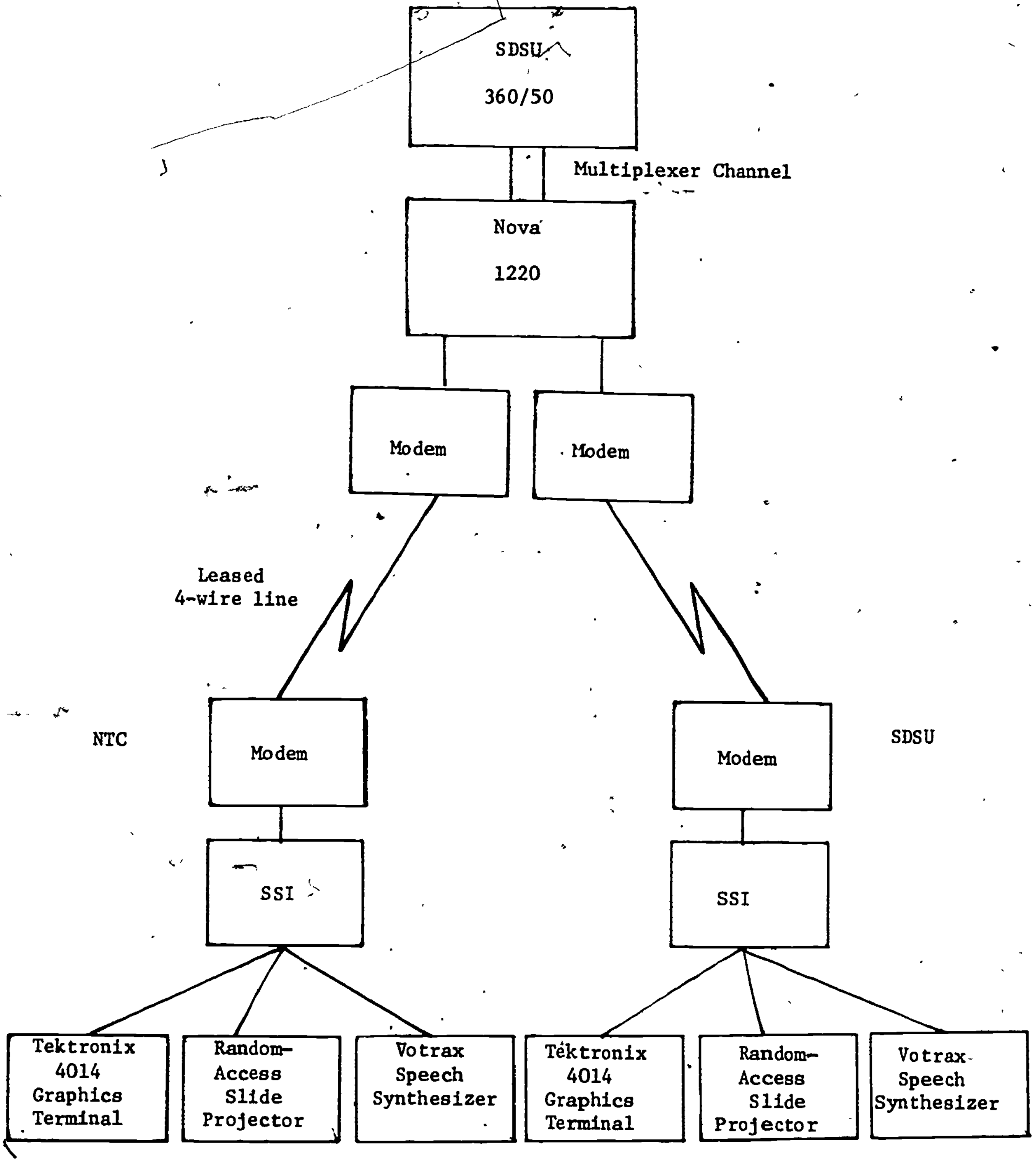
SDSU
360/50
Multiplexer Channel
Nova
1220
Modem
Modem
Leased
4-wire line
NTC
Modem
SDSU
Modem
SSI
SSI
Tektronix 4014 Graphics Terminal
Random-Access Slide Projector
Votrax Speech Synthesizer
Tektronix 4014 Graphics Terminal
Random-Access Slide Projector
Votrax Speech Synthesizer

Figure 1. Schematic diagram of the present implementation of GTDS at San Diego State University.

FINDINGS AND CONCLUSIONS

Unlike "off-the-shelf" hardware components, GRAIL and its supporting software facilities in the GTDS present an entirely new CAI author language. Of course, newness if of no utility in and of itself--particularly as far as software systems are concerned. Evaluating a new system it usually means several months of exterminating problem areas, but GRAIL has many desirable characteristics and should be worth the effort. The greatest need now is to have the system used in as demanding a way as possible. It is anticipated that the NAVPERSRANDCEN projects in which it is scheduled for immediate application will provide such use. The primary goal of providing a good graphics capability for NAVPERSRANDCEN research activities should be met by the existing implementation. The speech synthesizer and random-access slide projector have each figured prominently in the planning of at least one research project to be undertaken at NTC, San Diego.

Comparison With Other Systems

Among the many CAI systems avialable, there are three which have received considerable attention and which collectively reflect most of the options available to the CAI system designer. All three have received substantial backing from the National Science Foundation. They are PLANIT (Bennick and Frye, 1970), PLATO (Programmed Logics for Automated Teaching Operations) (Bitzer and Johnson, 1971), and TICCIT, (Time-Shared Interactive Computer Controlled Instructional Television) (Bunderson, 1972). In the following paragraphs, some features of these systems are noted and compared with those of the GTDS.

Portability and Implementation Options

Two of the systems, PLATO and TICCIT, are very machine dependent and not at all transportable across computers. PLANIT, however, is now written in a subset of FORTRAN IV which is nearly universal across large, multipurpose machines and is supposed to be highly portable. In comparison, GTDS contains some software which is directed at particular student-station hardware components such as the Votrax speech synthesizer and Tektronix graphics display. However, the system will run with little or no modification using other terminals if limited to alphanumeric I/O (which is all that PLANIT supports). Because its software has been developed in assembly language and PL/I, GTDS cannot be transported to machines other than IBM 360 and 370 series, these machines comprise a large subset of the world's computers.

Portability is affected not only by the language(s) employed in system development but also by the overall dependency between software and hardware insofar as core and other machine requirements are concerned. TICCIT is configured for implementation on dedicated minicomputers. No other implementation is possible. PLATO, on the other hand, demands all the resources of a very large computer. It, too, requires a dedicated system, but, in this case, an extremely large and expensive one. PLANIT, presumably, can be run on any computer which supports FORTRAN IV and interactive, on-line utilization.

GTDS can be implemented on virtually any IBM 360 or 370. As discussed below, it would find the environment of a large time-sharing installation very hospitable. On the other hand, it can support a limited number of terminals on a middle-size machine such as the 360/50 at San Diego State without degrading overall system performance. A third alternative would be to use a small 370 in a dedicated capacity. It is quite likely that such a GTDS configuration would be cost effective vis a vis various minibased systems for supporting 20 to 40 terminals. Modular expansion of such a system would be possible through duplication but expansion through moving to a larger central processor would probably be more desirable.

Thus, while GTDC does not exhibit the high degree of transportability claimed for PLANIT, it provides a much greater range of implementation options than either PLATO or TICCIT. It is the only one of the systems under discussion to offer any options as far as size and dedication of central processor are concerned, while still providing significant innovative features such as graphics, slide projection, and audio output.

Courseware Development

Two of the systems, PLATO and PLANIT, are designed for on-line authoring of courseware by content area specialists, i.e., instructors. TICCIT--by virtue of its use of primarily television technology--requires the services of a team of specialists for courseware development. GRAIL is designed for authoring either by individual researchers or instructors, or else by programmers or coders supervised by such content area specialists. GRAIL is somewhat more complicated than PLANIT by virtue of the greater richness of the language but no more difficult to use effectively than TUTOR, the author language of the PLATO System. PLANIT is extremely limiting in terms of courseware organization because of its orientation to frames and limitation to only four frame types.

The GTDS, in its current implementation at San Diego State University, does not support on-line courseware development, but it must be emphasized that this is a limitation of this particular implementation and not a limitation of the basic system design. There is no reason as far as GTDS software is concerned why courseware cannot be written on-line and submitted for compilation by conversational remote job entry (CRJE). Any on-line, general-purpose text editor is adequate for course authoring and any means of delivering the machine-readable code to the compiler is acceptable. Neither on-line authoring and editing nor CRJE involve more than trivial modifications to GTDS. They depend on system software external to GTDS. Even on-line compilation, requires little or no change in GTDS. However, it does require either a much larger partition of dedicated core than is now available to GTDS at San Diego State or implementation on a large IBM 360 or 370 which supports one or the other of IBM's timesharing systems. In such an environment, the GRAIL author would be able to write and debug courseware interactively much as can be done with PLATO or PLANIT. The major differences in course development would result from both the enforced modularity of GRAIL courseware and the fact that actual machine code is generated. The former means that recompilation of a small number of sections (probably one) would be required to correct a programming error and the latter implies that those debugging features common in interpretive systems (i.e., single statement recompilation, arbitrary execution time tracing, etc.) are absent in GRAIL. This is a small price to pay for the efficiencies of the modular, reentrant, machine code produced by the GRAIL compiler.

Cost

Cost, per se, was not a major consideration in the determination of GTDS hardware selection. The primary thrust has been to provide the graphics and other functional capabilities required for R&D activities and not to design a terminal capable of delivering CAI at a minimal cost. Having developed a good CAI package, however, one may speculate about how it might economically be delivered in quantity.

The most expensive component in the GTDS student station, by far, is the Tektronix Graphic Display Terminal. This is a fine device, but the plasma display panel used in the PLATO terminal may ultimately be cheaper in quantity. However, current costs estimates for PLATO student terminals are quite comparable in cost. The cost per unit of Votrax synthesizer should also be less in large quantity. A cheaper GTDS terminal, then, might be designed much like the PLATO terminal but would include an additional component—the speech synthesizer. At present, however, there is no inexpensive hard-copy device to accompany the plasma display panel.

All in all, given the freedom of choice with regard to size of computer main frame and selection of operating system (within the IBM 360 or 370 series), its unique graphics and speech capabilities, and the very modest cost of development—relative to the others, the GTDS System is a bargain in comparison to PLANIT, PLATO, or TICCIT and well suited to R&D applications. If high-quality graphics are required, it should even be competitive for operational CAI activities.

Potential Growth and Development

As discussed in the section on the GRAIL author language (pp. 1-2), even though the language contains constructs oriented to the particular needs at NAVPERSRANDCEN (graphics, speech, etc.), it is still very general. The control statements, FORTRAN subroutine capability, and macro definition capability—in particular—make it both powerful and flexible.

In the absence of graphics requirements, GTDS could support a substantial number of terminals very efficiently on a small IBM System 370 machine. With an intermediate machine such as the 360/50 in the San Diego State University implementation, predefined graphics displays are possible. On a very large machine, on-line graphics could be incorporated.

There are very few restrictions on the options for expansion or change of the system, in fact, because of adherence to the principles of structured programming in the development of the software package. Modularity and the isolation of operating-system-dependent modules, in particular, will facilitate desired adaptations. The fact that the GRAIL compiler is written in assembly language does constrain utilization of the system to IBM 360 and 370 series machines but this is the least restriction possible in writing a compiler to produce machine-code programs. Run-time (in)efficiency is a major problem with many CAI systems and is best avoided by the use of reentrant, compiled code as in the GTDS.

Commitments to the particular hardware of the present student-station configuration are built into GTDS, but these commitments are not irrevocable. Software support of the Tektronix terminal, for instance, is specific to that device to some extent, but the graphics support is independent of the rest of the system.

RECOMMENDATIONS

Although current plans do not include any substantial enhancements of the GTDS, there is still room for improvement. For example, little has been done to accommodate various kinds of data structures (e.g., matrices and lists) directly in GRAIL, and there is virtually no list-processing capability either in GRAIL or in FORTRAN. Development and inclusion of dynamic data definition support and list-processing instructions would greatly enhance the ability of CAI researchers to draw on the contributions of those in the artificial intelligence (AI) community who are addressing relevant problems. Recent advances in natural-language processing, for instance, tend to be manifest in programs developed in LISP or closely related list-processing languages. The potential for substantial and worthwhile improvements in CAI technology are latent in much contemporary AI research, but it will require some effort from those directly concerned with CAI R&D to capitalize on them. The availability of a CAI language which can accommodate the manipulation and evaluation of LISP-like symbolic expressions would certainly help.

In view of the above, it is recommended that GTDS be enhanced so that it can be used for a broad range of Navy training and R&D applications.

REFERENCES

Bennick, F. D. & Frye, C. H. PLANIT Language Reference Manual. Control Data Corporation, No. X0010422, System Development Corporation, 1970.

Bitzer, D. L. & Johnson, R. L. PLATO: A computer-based system used in the engineering of education. Reprint from the Proceedings of the IEEE, 59 (6), 1971.

Bunderson, C. V. Team production of learner-controlled courseware: A progress report. Institute for Computer Uses in Education, Brigham Young University, Provo, Utah, November, 1972.

APPENDIX

GRAPHIC TERMINAL DISPLAY
SYSTEM EXTERNAL
DOCUMENTATION

EXTERNAL DOCUMENTATION CONTENTS

1. GTDS CONCEPTS AND FACILITIES

GTDS - CONCEPTS AND FACILITIES

GTDS HAS THE GENERAL DESIGN GOAL OF SUPPORTING RESEARCH IN COMPUTER ASSISTED INSTRUCTION.

THE FOLLOWING SUBSYSTEMS PROVIDE THIS SUPPORT:

GRAIL GRAPHIC ASSISTED INSTRUCTIONAL LANGUAGE

GRAIL IS THE 'AUTHOR LANGUAGE' IN THE GTDS SYSTEM AND PROVIDES THE FOLLOWING FEATURES:

- DISPLAY OF ALPHANUMERIC AND/OR GRAPHIC INFORMATION. UPPER AND LOWER CASE ALPHANUMERICS MAY BE MIXED (ALONG THE GRAPHIC INFORMATION) IN THE SAME DISPLAY.

- SELECTION AND DISPLAY OF ANY ONE OF EIGHTY SLIDES LOCATED IN THE RANDOM ACCESS SLIDE PROJECTOR.

- CONTROL OF THE SEQUENCE OF PRESENTATION AND ANALYSIS, BOTH ON LOGICAL CONDITION TESTING AND INTERATIVELY. LOGICAL COMBINATIONS (AND,OR) OF RELATIONAL (EQUAL TO, NOT EQUAL, LESS THAN, ETC.) EXPRESSIONS, WITH GROUPING (INDICATED VIA PARENTHESIS) IS ALLOWED.

- INPUT FROM THE TERMINAL INCLUDES UPPER AND LOWER CASE ALPHANUMERIC AS WELL AS THE ABILITY TO DETERMINE THE CURRENT POSITION OF THE CROSSHAIR CURSOR.

- A SIGNIFICANT LEVEL OF 'COMPILE TIME' CHECKING INCLUDING
  - CHECKS FOR PROPER 'NESTING' OF CONSTRUCTS, I.E., IF ONE CONSTRUCT MUST BE WHOLLY CONTAINED WITHIN ANOTHER, THEN AN ERROR MESSAGE RESULTS IF THE AUTHOR VIOLATES THE RESTRICTION.

  - CHECKS FOR PROPER DATA TYPES, FOR EXAMPLE: A CHARACTER STRING VARIABLE USED IN AN ARITHMETIC EXPRESSION IS INVALID.

ALTHOUGH THE ABOVE FEATURES ARE ALMOST ALWAYS ENCOUNTERED IN CAI AUTHOR LANGUAGES, THE FOLLOWING ARE RELATIVELY UNIQUE:

- REENTRANT CODE, WHICH ALLOWS SEVERAL STUDENTS TO BE 'TAUGHT' USING THE SAME MACHINE INSTRUCTIONS, THEREBY SAVING STORAGE.

EXTERNAL DOCUMENTATION

- EXPRESSIONS (CHARACTER AND ARITHMETIC) ARE ALLOWED VIRTUALLY ANYWHERE A VARIABLE CAN OCCUR.

- EXTENSIBLE LANGUAGE FACILITY. THE MACRO DEFINITION LANGUAGE (MDL) IS AN INTEGRAL PART OF GRAIL AND ALLOWS ADDITIONAL OPERATIONS AND CONSTRUCTS TO BE ADDED TO THE LANGUAGE BY THE AUTHOR. THE MDL ALLOWS THE AUTHOR TO HAVE VIRTUALLY COMPLETE FREEDOM IN THE SELECTION OF BASIC OPERATIONS TO BE USED IN TEACHING A COURSE.

- DYNAMIC STORAGE ALLOCATION. MAIN (CORE) STORAGE IS ALLOCATED ONLY WHEN THE STRUCTURE REQUIRING IT IS ENTERED, THUS REDUCING MAIN STORAGE USE.

- PORTABILITY. ALL OPERATING SYSTEM DEPENDENT CODE IS ISOLATED INTO ONE AREA, ALLOWING EASY CONVERSION TO ANY OPERATING SYSTEM WHICH RUNS ON THE IBM 360-370 SERIES OF COMPUTERS.

- GENERATION OF MACHINE CODE
  ACTUAL 'MACHINE CODE' (COMPUTER INSTRUCTIONS) ARE GENERATED BY GRAIL. THIS IS TYPICALLY 5 TO 10 TIMES FASTER IN EXECUTION THAN 'INTERPRETIVE' SYSTEMS.

- SUPPORT OF "STRUCTURED" PROGRAMMING CONCEPTS
  - ENFORCED MODULARITY OF PROGRAMS - THE COMPILER WILL NOT ACCEPT A SOURCE PROGRAM OF MORE THAN 200 SOURCE STATEMENTS.
  - THE GRAIL LANGUAGE DOES NOT CONTAIN A "GOTO" STATEMENT, ALTHOUGH PROVISION IS MADE FOR EARLY TERMINATION OF LOOPING CONSTRUCTS.
  - TRANSPARENT OVERLAYING OF SUBROUTINES IN SUCH A WAY THAT THE AUTHOR NEED NOT WORRY ABOUT STORAGE MANAGEMENT AT ANY TIME.

EXCALIBUR EXTRACT AND COLLECT LOGICALLY INDICATED USER RECORDS

EXCALIBUR PROVIDES FOR THE COLLECTION, MAINTENANCE, AND SELECTION OF AUTHOR SELECTED INFORMATION WITH THE FOLLOWING FEATURES:

- AUTHOR CONTROLLED RECORDING AT ANY TIME, IN ADDITION TO RECORDING STUDENT LOGON, LOGOFF TIMES.

- FORTRAN (AS WELL AS PL/I, COBOL, AND ASSEMBLER) COMPATIBLE RECORDS, ALLOWING DATA ANALYSIS TO BE DONE ON ANY MACHINE SUPPORTING MAGNETIC TAPE AND FORTRAN.

- EASY SELECTION OF RECORDS FOR FURTHER ANALYSIS THROUGH VERY SIMPLE PL/I PROGRAMS.

2. GRAIL LANGUAGE SPECIFICATIONS

2.1. NOTATION

SYNTACTIC VARIABLES:
ANYTHING BEGINNING WITH THE CHARACTER '@' IS REPLACED BY THE FORM INDICATED FOR THE VARIABLE UNDER 'DEFINITIONS'.

METASYMBOLS:
THE CHARACTERS ' (APOSTROPHE), '<', '>', AND '/' ARE TO BE TREATED AS METASYMBOLS UNLESS THEY ARE ENCLOSED IN APOSTROPHES. THEY HAVE THE FOLLOWING MEANINGS:

<,> ARE USED TO INDICATE THAT THE ENCLOSED FORM IS OPTIONAL AND MAY BE OMITTED AT THE USER'S DISCRETION.

/ INDICATES THAT A CHOICE MUST BE MADE AMONG THE ALTERNATIVE FORMS WHICH IT SEPARATES; IF A DEFAULT IS ALLOWED, IT PRECEEDS THE FIRST /, UNLESS OTHERWISE INDICATED BY THE TEXT.

THE FORM ',...' INDICATES THAT THE PRECEDING FORM MAY BE REPEATED A NUMBER OF TIMES (AS SPECIFIED IN THE TEXT) WITH EACH FORM SEPARATED BY COMMAS.

THE FORM '...' INDICATES THAT THE PRECEDING SYNTACTIC VARIABLE MAY BE REPEATED A NUMBER OF TIMES.

PARENTHESIS '(' AND ')' ALWAYS INDICATE THAT ACTUAL PARENTHESES OCCUR, AND IN THE FORM <(> FORM <)>, EITHER BOTH PARENTHESES MUST BE PRESENT OR BOTH OMITTED.

APOSTROPHES INDICATE THAT THE ENCLOSED CHARACTER OR CHARACTERS ARE TO BE TREATED AS THEMSELVES AND NOT AS METASYMBOLS; THUS, THE FORM ''' INDICATES AN ACTUAL *SINGLE* APOSTROPHE - ' AND '/' AN ACTUAL SLASH, ETC.

2.2. SYNTAX

```
@OLABLE   TEXT   (<'@TEXTSTRING' / @CEXP >,...),
                 <POSIT=(<@FEXP-1<>,@FEXP-2<)<,
                 >,CRLT=YES / NO>
```

CAUSES THE INDICATED STRINGS OF CHARACTERS TO BE DISPLAYED. IF NEITHER @FEXP-1 OR @FEXP-2 IS SPECIFIED THE STRING WILL BE DISPLAYED AT THE CURRENT ALPHA-CURSOR POSITION.
IF THE POSIT PARAMETER IS SPECIFIED, @FEXPR-1 INDICATES THE LINE WHERE THE DISPLAY IS TO BEGIN AND @FEXPR-2 THE POSITION WITHIN THE LINE WHERE THE FIRST CHARACTER IS TO BE DISPLAYED. IF @FEXPR-1 IS OMITTED, THE CURRENT LINE IS ASSUMED AND IF @FEXPR-2 IS OMITTED IT IS ASSUMED TO BE 1.
IF THE VALUES SPECIFIED EXCEED THEIR RESPECTIVE MAXIMUM VALUES THEN THE MAXIMUM VALUE IS USED INSTEAD. THESE MAXIMUMS ARE SPECIFIED BY THE EXECUTION VARIABLES $FMLIN AND $FMPOS FOR @FEXP-1 AND @FEXP-2 RESPECTIVELY.

IF THE VALUE OF EITHER OF THE POSIT= PARAMETERS IS EITHER ZERO OR NEGATIVE THEN THAT PARAMETER IS TREATED EXACTLY AS IF IT WERE NOT SPECIFIED.
REGARDLESS OF THE POSIT= PARAMETER (OR IN IT'S ABSENCE) THE ALPHA-CURSOR IS MOVED TO THE FIRST CHARACTER OF THE NEXT LINE AFTER THE CHARACTERS ARE DISPLAYED, UNLESS THE CRLF=NO PARAMTER IS USED, IN WHICH CASE THE ALPHA-CURSOR WILL BE LEFT AT THE POSITION IMMEDIATELY FOLLOWING THE LAST CHARACTER DISPLAYED.

@OLABLE TALK (@VOCAL-FORM,...)

THE @VOCAL-FORMS ARE VOCALIZED IN THE ORDER CODED. THE VARIOUS @VOCAL-FORMS ARE INTERPRETED AS FOLLOWS-

@PHON-LIT (PHONEMIC LITERAL)

THE @PLIT'S AND @UFLIT ARE CONVERTED AT *COMPILE* TIME TO THE 'INTERNAL' PHONEME FORM AND ARE VOCALIZED EXACTLY AS CODED DURING EXECUTION-*NO* PAUSES ARE INSERTED, AND THE SPECIFIED INFLECTIONS ARE NOT MODIFIED.
THE @PHON-LIT MUST FOLLOW THE FORMAT DETAILED UNDER 'PHONETIC INPUT DATA FORMAT'.

@ WORD-FORM

EACH @CLIT IS 'LOOKED UP' IN THE VOTRAX DICTIONARY FILE (VDF) AND CONVERTED TO ITS 'INTERNAL' PHONEME FORM AT *COMPILE* TIME AND A WORD PAUSE (PAI) INSERTED *AFTER* EACH WORD.

IF THE SPECIAL CHARACTERS ',' ,';' OR '.' OCCUR AS THE *SOLE* CHARACTER IN A @CLIT THEN A PA1, PA2, AND PA3 ARE GENERATED, RESPECTIVELY.
IF '?' OCCURS AS THE *LAST* @CLIT THEN THE INFLECTIONS OF THE PRECEDING WORD ARE MODIFIED TO GENERATE A 'RISING INFLECTION' IF POSSIBLE. IF THE '?' IS NOT THE LAST @CLIT IN THE STATEMENT, OR IF IT IS THE *ONLY* @CLIT THEN AN ERROR MESSAGE RESULTS.
THE FOLLOWING RESTRICTIONS ON THIS FORM SHOULD BE NOTED-

THE @CLIT'S MUST BE SPECIFIED AS LOWER CASE- IE THE '@' OR '@@' NOTATION NORMALLY AVAILABLE IN @CLIT'S MAY *NOT* BE USED.
EACH @CLIT MUST BE A SINGLE ENGLISH WORD WHICH IS PRESENT AT *COMPILE* TIME IN THE VDF.

@TEXT-FORM

@CEXP IS EVALUATED DURING *EXECUTION* AND THE FIRST SIX CHARACTERS SHOULD NAME A 'VOCAL TEXT'. IF PRESENT, @FEXP IS EVALUATED AND USED TO SELECT A 'SEGMENT' OF THE VOCAL-TEXT. IF @FEXP IS OMITTED OR ZERO THEN THE ENTIRE VOCAL-TEXT IS VOCALIZED; OTHERWISE ONLY THE INDICATED VOCAL-TEXT SEGMENT IS VOCALIZED.

@PHON-VAR (PHONEMIC VARIABLE)

THE RESULT OF EVALUATING @CEXP IS ASSUMED TO BE A STRING IN 'INTERNAL' PHONEME FORM AND IS THEREFORE VOCALIZED *EXACTLY* AS PRESENTED.
IF CHARACTERS ~~OF~~ THE BIT CONFIGURATION 'XX111111' ARE PRESENT IN THE STRING, THEY ARE CONVERTED TO PA1'S TO AVOID TIMING PROBLEMS.

THE 'INTERNAL' PHONEME CODES RESULTING FROM EVALUATING THE FORMS PRESENT IN THE TALK STATEMENT ARE CONCATENATED AND PRESENTED TO THE AUDIO UNIT IN 63 CHARACTER 'CHUNKS' (OR LESS IF THE TOTAL IS FEWER THAN 63) FOLLOWED BY THE 'SPEAK' CODE. LOADING OF THE NEXT 'CHUNK' OCCURS WHILE THE FIRST 'CHUNK' IS BEING VOCALIZED, SO THAT (WITH PROPER PLANNING) CONTINUOUS SPEECH IS ACHIEVED.

@OLABEL SHOW (@FEXP)
THE SLIDE SPECIFIED BY @FEXP IS SELECTED AND DISPLAYED ON THE RANDOM ACCESS SLIDE PROJECTOR. IF @FEXP IS GREATER THAN THE MAXIMUM VALID SLIDE NUMBER FOR THE TERMINAL THEN THE COMMAND IS IGNORED. IF THE TERMINAL DOES NOT HAVE A SLIDE PROJECTOR (AS INDICATED BY THE EXECUTION VARIABLE $CRASP) THEN THE COMMAND IS IGNORED.

EXTERNAL DOCUMENTATION

```
@OLABEL   HDCPY
                   A PRINTED COPY OF THE CURRENT SCREEN IS MADE.  IF
                   THE STUDENT STATION IS NOT EQUIPPED WITH A HARD
                   COPY UNIT THE STATEMENT IS IGNORED.

@OLABEL   FCALL    @FORNAME, (@EXP-1,EXP-2,EXP-3,...)
                   THE FORTRAN SUBROUTINE IDENTIFIED BY @FORNAME IS CALLED
                   WITH THE ARGUMENTS SPECIFIED BY @EXP-1,@EXP-2...
                   THE SUBROUTINE CALLED MUST BE KNOWN TO THE SYSTEM OR
                   AN ERROR MESSAGE WILL RESULT DURING COMPILATION.
                   IT IS THE USER'S RESPONSIBILITY TO ENSURE THAT THE
                   TYPES OF THE ARGUMENTS PASSED AGREE WITH THE TYPES
                   EXPECTED BY THE SUBROUTINE.

@OBABEL   EXEC     (@CEXP)<,(@EXP-1,@EXP-2,...)>
                   THE SECTION IDENTIFIED BY THE FIRST 6 CHARACTERS
                   OF THE RESULT OF EVALUATING @CEXP IS EXECUTED ,
                   AFTER WHICH CONTROL RETURNS TO THE NEXT STATEMENT
                   IF THE RESULT IS LESS THAN SIX CHARACTERS , THEN
                   IT IS PADDED WITH TRAILING BLANKS.  THE @EXP'S ARE
                   EVALUATED AND PASSED TO THE INVOKED SECTION AS
                   ARGUMENTS.  SEE THE DESCRIPTION OF 'SECTION' FOR
                   ADDITIONAL CONSIDERATIONS.

@LABEL    SECTION <@VAR-1< @UFLIT><,@VAR-2< @UFLIT>>...<@VAR-10
                    < @UFLIT>>>
                   DEFINES THE BEGINNING OF A SECTION AND DEFINES AND
                   IDENTIFIES THE VARIABLES USED TO CONTAIN ARGUMENTS.
                   WHEN CONTROL ENTERS A 'SECTION' STATEMENT THE FOLLOWING
                   ACTIONS OCCUR:
                      1.  ALL VARIABLES DEFINED BY VDEF STATEMENTS ARE
                          INITIALIZED TO NULL OR ZERO.
                      2.  THE VALUES DEFINED IN THE 'EXEC' STATEMENT
                          ARE MOVED TO THE CORRESPONDING (BY POSITION)
                          VARIABLE IN THE 'SECTION' STATEMENT.  IN THE
                          CASE OF A CHARACTER VARIABLE WHERE THE
                          RECEIVING FIELD IS SHORTER THAN THE SENDING
                          FIELD, THE EXCESS CHARACTERS ARE LOST.
                      3.  CONTROL PASSES TO THE NEXT STATEMENT.

                   THE FOLLOWING RESTRICTIONS SHOULD BE OBSERVED:
                      1.  THE NUMBER OF ARGUMENTS SPECIFIED IN THE 'EXEC'
                          STATEMENT WHICH INVOKES A SECTION MUST BE
                          EXACTLY EQUAL TO THE NUMBER OF ARGUMENTS
                          SPECIFIED IN THE 'SECTION' STATEMENT.
```

2. THE TYPE (C, E, OR F) OF EACH ARGUMENT PASSED MUST AGREE WITH THE TYPE OF THE VARIABLE IN THE CORRESPONDING POSITION IN THE 'SECTION' STATEMENT.
3. THE ARGUMENTS PASSED MAY INCLUDE LITERALS, EXECUTION VARIABLES, CONSTRANTS DEFINED BY A 'CDEF' STATEMENT AND/OR NON-ATOMIC EXPRESSIONS - BUT THE VALUE OF THE CORRESPONDING VARIABLE IN THE 'SECTION' STATEMENT MUST NOT BE CHANGED WITHIN THE SECTION.

```
      SEND @MLABEL
```

CONTROL RETURNS TO THE UNIT WHICH INVOKED THE SECTION BEING ENDED. THE VALUES OF THE ARGUMENTS WHICH WERE PASSED TO THE SECTION ARE UPDATED (CHARACTER STRING VALUES ARE TRUNCATED IF NECESSARY).

```
@LABEL . COURSE
```

DEFINES A COURSE WITH THE INDICATED NAME , CAUSES STORAGE TO BE ALLOCATED FOR VARIABLES DEFINED IN THE COURSE AND THE SCREEN TO BE ERASED. $CRSE IS SET TO @LABEL AND $CSECT IS SET TO NULL.

```
        CRSEND @MLABEL
```

DEFINES THE END OF A COURSE AND CAUSES ALL STORAGE ALLOCATED FOR THE COURSE TO BE RELEASED.

```
@LABEL  IF    (@CON-CLS)
        THEN  <NULL>
        ELSE  <NULL>
        IEND  @MLABEL
```

THESE FOUR OPERATIONS ALLOW FOR THE CONDITIONAL EXECUTION OF SUBSEQUENT STATEMENTS AS FOLLOWS -

1. IF @CON-CLS IS TRUE , THE STATEMENTS BETWEEN THE THEN AND ELSE OPERATIONS ARE EXECUTED , AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND

2. IF @COS-CLS IS FALSE , THE STATEMENTS BETWEEN THE ELSE AND IEND ARE EXECUTED , AFTER WHICH CONTROL PASSES TO THE STATEMENT FOLLOWING THE IEND.

3. NO STATEMENTS MAY OCCUR BETWEEN THE IF AND THEN. OPERATIONS.

4. IF THE 'NULL' OPTION IS SPECIFIED FOR A THEN OR ELSE OPERATION , AND IF CONTROL IS PASSED TO IT BY THE IF OPERATION , THE CONTROL IMMEDIATELY PASSES TO THE STATEMENT FOLLOWING THE IEND.

5. IF CONSTRUCTS MAY BE 'NESTED' , THAT IS - AN IF CONSTRUCT MAY OCCUR BETWEEN THE THEN AND ELSE , OR ELSE AND IEND , OPERATIONS OF ANOTHER IF CONSTRUCT.

6. ALL FOUR OPERATIONS COMPRISING THE IF CONSTRUCT ARE REQUIRED WHEN THE CONSTRUCT IS USED.

```
@LABEL   DO    WHILE / UNTIL,(@CON-CLS)
         < STATEMENTS >
         DEND  @MLABEL
```

THESE TWO OPERATIONS ALLOW FOR THE REPETITIVE EXECUTION OF A GROUP OF STATEMENTS AS FOLLOWS -

WHEN CONTROL ENTERS THE DO STATEMENT, THE VALUE OF @CON-CLS IS TESTED. IF IT IS TRUE (FALSE) AND THE WHILE (UNTIL) FORM WAS USED, THEN THE STATEMENTS BETWEEN THE DO AND DEND ARE EXECUTED, AFTER WHICH CONTROL AGAIN ENTERS THE DO STATEMENT.
WHENEVER CONTROL ENTERS THE DO STATEMENT AND @CON-CLS IS FALSE (TRUE) AND THE WHILE (UNTIL) FORM WAS USED, THEN CONTROL PASSES TO THE STATEMENT FOLLOWING THE DEND.

```
@LABEL   DOINC @FVAR,(@FEXP-1),TO,(@FEXP-2)<,BY,(@FEXP-3>)
         DODEC
         < STATEMENTS >
         DEND  @MLABEL
```

WHEN CONTROL ENTERS THE DOINC (DODEC) STATEMENT
@FEXP-1 IS EVALUATED AND ASSOGNED TO @FVAR;
@FEXP-2 IS EVALUATED AND SAVED;
@FEXP-3 IS EVALUATED AND SAVED (IF @FEXP-3 IS OMITTED IT IS ASSUMED EQUAL TO ONE).
@FVAR IS NOW COMPARED TO @FEXP-2, AND IF @FVAR IS GREATER THAN (LESS THAN) @FEXP-2 THEN CONTROL PASSES TO THE STATEMENT FOLLOWING THE DEND. IF @FVAR IS LESS THAN OR EQUAL (GREATER THAN OR EQUAL) TO @FEXP-2 THEN THE STATEMENTS BETWEEN THE DOINC (DODEC) AND THE DEND STATEMENT ARE EXECUTED. WHEN CONTROL ENTERS THE DEND STATEMENT THE SAVED EVALUATION OF @FEXP-3 IS ADDED TO (SUBTRACTED FROM) @FVAR. CONTROL IS THEN TRANSFERRED TO THE POINT IN THE DOINC (DODEC) WHERE @FVAR IS COMPARED TO @FEXP-2 AND THE ABOVE PROCESS IS REPEATED.

```
@LABEL    CASES @FLIT<,EXEC=FIRST / ALL>
          CASE  (@CON-CLS-1)
          < STATEMENTS >
          CASE (@CON-CLS-2)
          < STATEMENTS >
          CASE
          .
          .
          .
      CASE  (@CON-CLS-@FLIT)
          < STATEMENTS >
          CEND  @MLABEL
```

THE CASES CONSTRUCT ALLOWS SELECTION OF GROUPS
OF STATEMENTS TO BE EXECUTED BASED UPON THE
LOGICAL VALUES OF CONDITIONAL CLAUSES. WHEN THE
CASES STATEMENT IS ENTERED THE @CON-CLS OF EACH
CASE STATEMENT IN THE CONSTRUCT IS EVALUATED,
BEGINNING WITH THE FIRST. IF THE @CON-CLS OF
ANY PARTICULAR CASE STATEMENT IS TRUE THEN THE
STATEMENTS BETWEEN THAT CASE STATEMENT AND THE
NEXT CASE OR CEND STATEMENT OF THE CONSTRUCT
ARE EXECUTED. IF THE EXEC=ALL OPTION IS SPECIFIED
THEN THE EVALUATION OF EACH @CON-CLS CONTINUES;
IF THE EXEC EXEC=FIRST OPTION IS SPECIFIED THEN
ONLY THE STATEMENTS BETWEEN THE FIRST CASE
STATEMENT WHICH EVALUATED TO TRUE AND THE NEXT
CASE OR CEND STATEMENT WILL BE EXECUTED.
WITH EITHER EXEC OPTION AT LEAST ONE @CON-CLS
MUST BE TRUE OR EXECUTION OF THE COURSE WILL
BE TERMINATED WITH AN ERROR.
@FLIT IS USED TO SPECIFY THE NUMBER OF CASE
STATEMENTS WITHIN THE CURRENT CASES CONSTRUCT AND
MUST BE EXACTLY EQUAL TO THE NUMBER OF CASE
STATEMENTS. @FLIT MUST BE LESS THAN 100.

```
@LABEL    READ  <@CVAR><,POSIT=(<@FEXP-1><,@FEXP-2>)>
```

CAUSES CHARACTERS TO BE READ FROM THE TERMINAL INTO
@CVAR. IF MORE CHARACTERS ARE ENTERED THAN @CVAR CAN
HOLD, THEY ARE LOST. IF @CBAR IS OMITTED THEN THE
EXECUTION VARIABLE $CANS IS USED AND HAS A MAXIMUM
LENGTH OF 72. IF THE POSIT= OPTION IS SPECIFIED THEN
THE ALPHA-CURSOR IS MOVED TO THE INDICATED POSITION
ON THE SCREEN BEFORE INPUT IS ACCEPTED (SEE 'TEXT').

EXTERNAL DOCUMENTATION

```
@OLABEL   CALC  @VAR,(@EXP)
                @EXP IS EVALUATED AND ASSIGNED TO @VAR.  @EXP AND @VAR
                MUST BE ON THE SAME TYPE.

@LABEL    READCC <@FVAR-1><,@FVAR-2><,@CVAR>
                THE CROSSHAIR CURSOR IS ILLUMINATED AND ITS X-Y
                COORDINATES ARE READ WHEN THE NEXT CHARACTER IS TYPED
                BY THE USER.  THE X VALUE , THE Y VALUE AND THE
                CHARACTER TYPED ARE READ INTO @FVAR-1,@FVAR-2 AND
                @CVAR RESPECTIVELY.  IF ANY (OR ALL) OF THE OPERANDS
                ARE OMITTED , THE VALUES ARE READ INTO THE EXECUTION
                VARIABLES $FCCX , $FCCY AND $CCHR RESPECTIVELY.
                NOTE THAT THE X AND Y VALUES READ ARE IN SCREEN
                COORDINATES AND THAT THE RESULTING LENGTH OF @CVAR
                (OR $CCHR) WILL ALWAYS BE ONE.

@LABEL    RECORD (@CEXP),TYPE=(@CEXP)
                @CEXP-1,@CEXP-2,... ARE CONCATENATED AND WRITTEN
                ON THE RECORDER FILE.  THE RECORD IS IDENTIFIED BY THE
                FIRST TWO CHARACTERS OF THE CHARACTER STRING
                RESULTING FROM EVALUATION OF THE @CEXP SPECIFIED
                FOR TYPE.  IT IS THE USER'S RESPONSIBILITY TO ENSURE
                THAT THE RESULTING RECORD HAS THE CORRECT FORMAT FOR
                ITS TYPE, AND THAT THE TYPE ITSELF IS VALID.

@OBALED    DELAY (@FEXP)
                EXECUTION OF SUBSEQUENT OPERATIONS IS DELAYED BY THE
                SPECIFIED NUMBER OF SECONDS,

@LABEL    ESCAPE @MLABEL
                THIS COMMAND ALLOWS EXIT FROM WITHIN A NEST OF 'IF'
                , 'DO' , 'DODEC' , 'DOINT' , 'COURSE' , 'SECTION'
                AND/OR 'CASES' CONSTRUCTS.  WHEN CONTROL ENTERS THE
                ESCAPE STATEMENT THE DEND, IEND, CRSEND, SEND OR CEND
                HAVING THE MATCHING @MLABEL WILL BE THE NEXT STATE-
                MENT EXECUTED.
                NOTE THAT THE STATEMENT HAVING THE MATCHING @MLABEL
                *MUST* BE WITHIN THE ACTIVE NEST AT THE
                TIME THE ESCAPE STATEMENT IS EXECUTED.

          LIMIT @FLIT
                THE LIMIT STATEMENT IS USED TO SPECIFY AN UPPER
                BOUND ON HOW MANY TIMES THE STATEMENTS WITHIN A
                'DO' , 'DODEC' OR 'DOINC' MAY BE EXECUTED.
```

IF NO LIMIT STATEMENT IS PRESENT THEN EACH SUCH
CONSTRUCT IS LIMITED TO 100 REPETITIONS. IF A LIMIT
STATEMENT IS PRESENT, THEN THE NEXT (AND *ONLY* THE
NEXT) 'DO', 'DOINC' OR 'DODEC'
CONSTRUCT IN THE SOURCE PROGRAM WILL BE LIMITED TO
@FLIT REPETITIONS. @FLIT MAY BE LESS THAN 100
BUT MUST BE GREATER THAN ZERO.

@OLABEL ERASE

THE SCREEN IS ERASED. IF THE TERMINAL IS NOT CAPABLE
OF BEING ERASED THEN THE STATEMENT HAS NO EFFECT.

VTYPE C(@L-1,@U-1),F(@L-2,@U-2),E(@L-3,@U-3)
THIS OPERATION IS USED TO INDICATE THAT ANY VARIABLE
OR FUNCTION REFERENCE IN SUBSEQUENT STATEMENTS WILL
HAVE IT'S TYPE DETERMINED AS FOLLOWS-
IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION
IS IN THE RANGE @L-N,@U-N THEN THE TYPE IS ASSUMED
TO BE INDICATED BY THE CHARACTER PRECEDING THE
PARENTHESIS IN WHICH THE RANGE WAS ENCLOSED.
IF THE FIRST CHARACTER OF THE VARIABLE OR FUNCTION
DOES NOT FALL IN ANY OF THE RANGES, THEN THE TYPE
IS ASSUMED TO BE CHARACTER.
AT THE TIME THE VTYPE STATEMENT IS ENCOUNTERD TH
FOLLOWING ITEMS ARE CHECKED-
1. THE RANGES MUST BE DISJOINT IE. THE RANGES MUST
NOT OVERLAP.
2. @L-N MUST BE LESS THAN OR EQUAL TO @U-N FOR
EACH RANGE
3. ONLY ONE VTYPE STATEMENT IS ALLOWED PER COMPILE
UNIT, AND IT MUST IMMEDIATELY PRECEDE
THE COURSE OR SECTION STATEMENT FOR THE COMPILE UNIT.

VDEF @VAR-1< @UFLIT>,@VAR-2< @UFLIT>...
@VAR-1,@VAR-2... ARE DEFINED AND RESERVED STORAGE
BY THEIR APPEARANCE IN THE VDEF STATEMENT. THE
TYPE (C,F OR E) OF EACH VARIABLE IS DETERMINED
BY THE FIRST CHARACTER OF IT'S NAME IN
CONJUNCTION WITH THE VTYPE STATEMENT CURRENTLY IN
FORCE.
A NAME MAY APPEAR IN ONLY ONE VDEF OR CDEF
STATEMENT WITHIN THE COMPILE UNIT.

CDEF @VAR-1 (@LIT), @VAR-2 (@LIT)...
@VAR-1, @VAR-2,...ARE DEFINED, RESERVED STORAGE AND SET TO THE VALUE INDICATED BY THE ASSOCIATED @LIT. QUANTITIES DEFINED IN THIS WAY MUST NOT HAVE THEIR VALUE CHANGED DURING EXECUTION. MORE EFFICIENT EXECUTION WILL RESULT IF THE NUMBER OF CDEF STATEMENTS IS KEPT TO A MINIMUM.
ALL OTHER RULES WHICH APPLY TO VARIABLES DEFINED BY THE VDEF STATEMENT APPLY TO THE CDEF STATEMENT, WITH THE EXCEPTION THAT THE LENGTH OF CHARACTER VARIABLES IS DETERMINED BY THE LENGTH OF THE ASSOCIATED @LIT.

CHARACTER VARIABLES ARE LIMITED TO 158 CHARACTERS, INCLUDING ALL '@' CHARACTERS. FIXED AND FLOATING POINT VARIABLES MUST BE SPECIFIED IN 32 OR FEWER CHARACTERS.

@LABEL CHKPT

INFORMATION IS SAVED WHICH ALLOWS THE USER TO RESTART AT THE STATEMENT FOLLOWING THE CHKPT STATEMENT IF HE IS 'LOGGED OFF' AS A RESULT OF A MACHINE MALFUNCTION, LOGOF STATEMENT OR 'LOGOFF WITH CONTINUATION' COMMAND.

@OLABEL FIND <@FVAR-1, @FVAR-2>
THE TERMINAL IS INTERROGATED TO DETERMINE THE CURRENT CURSOR POSITION. IF SPECIFIED THEN @FVAR-1 WILL CONTAIN THE X (HORIZONTAL) COORDINATE AND @FVAR-2 WILL CONTAIN THE Y (VERTICAL) COORDINATE.
IN ADDITION THE EXECUTION VARIABLES $FLIN AND $FPOS ARE SET TO INDICATE THE CHARACTER WHOSE CENTER IS NEAREST TO THE CURRENT CURSOR POSITION WITH THE CHARACTER SIZE DETERMINED BY THE CURRENT VALUE OF $FCSIZ.

@LABLE LOGOF

THE USER IS IMMEDIATELY 'LOGGED OFF' FROM CAMELOT. THIS IS ACCOMPLISHED IN A MANNER WHICH ALLOWS THE USER TO RESTART AT THE STATEMENT FOLLOWING THE LAST CHKPT STATEMENT EXECUTED WHEN SHE NEXT CONNECTS TO CAMELOT.

2.3. SYNTACTIC VARIABLES

```
@CON-CLS  := @REXP / <(> @REXP:@LOP:@REXP <)>
@ROP    :=    EQ / NE / GT / LT / GE / LE / NL / NG
@LOP    :=    OR / AND
@AEXP   :=    @FEXP / @EEXP
@FEXP   :=  ( @FVAR / @FREF / @FLIT / <(> @FEXP@AOP@FEXP <)>  )
@EEXP   :=  ( @EVAR / @FREF / @ELIT / <(> @EEXP@AOP@EEXP <)>  )
@AOP    :=    + / - / * / '//'  (// IS THE REMAINDER OPERATOR)
@CEXP   :=  ( @VAR / @CLIT / @FREF / @CEXP@COP@CEXP  )
@COP    :=    ;
@FREF   :=  @FNAME ( @EXP,@EXP,... )
@VAR    :=  @CVAR / @FVAR / #EVAR
@CVAR   ,  @FVAR , @EVAR := 1 TO 6 ALPHANUMERIC CHARACTERS ,
       THE FIRST OF WHICH MUST BE ALPHABETIC.
@LIT  := @FLIT / @ELIT / @CLIT
@CLIT  := '''@STRING'''
@ELIT  := @S@UFLIT
@UFLIT  := @D@UFLIT
@ELIT  := @FLIT@P<@UFLIT><E@FLIT>
@S :=  '' / + / -
@D :=  0 / 1 / 2 / 3 / 4 / 5 / 6 / 7 / 8 / 9
@P :=  .
@STRING := @C / @STRING@C
@C :=  A / B / C /.../ Z / @D /
       @AOP / @COP / ',' / ''' / & / ? / $ /
         / ( / ) / </> / @P / : / =

@LABEL  := @A / @A@LABEL / @A@D
       REQUIRED LABLE , WHICH MUST BE 6 OR FEWER
       CHARACTERS LONG.

@TEXT-STRING := @STRING
      CERTAIN CHARACTERS WILL CAUSE THE EFFECTS NOTED
      NOTED BELOW IF PRESENT IN STRING -
      @ - AT SIGN
          INDICATORS THAT THE SINGLE LETTER IMMEDIATELY
          FOLLOWING IT IS TO BE DISPLAYED AS UPPER CASE.
      @@- TWO AT SIGNS
          INDICATES THAT ALL LETTERS BETWEEN THE @'S
          AND THE NEXT @ ARE TO BE DISPLAYED AS UPPER CASE.
      - - UNDERSCORE
          INDICATES THAT THE TEXT BETWEEN IT AND THE NEXT
          UNDERSCORE IS TO BE UNDERLINED.
```

ANY APOSTROPHES (') OR AMPERSANDS (&) WHICH ARE
TO BE DISPLAYED MUST BE PRESENT TWICE.

```
@VOCAL-FORM := @PHONEMS-FORM
               @WORD-FORM
               @TEXT-FORM
               @PHON-VAR

@PHON-LIT   := '<'@PLIT<,@FLIT>,...'>'
@WORD-FORM  := @CLIT,...
@TEXT-FORM  := (@CEXP<,@FEXP>,...
@PHON-VAR   := '>'@CEXP,...'<'
@PLIT       := COPY FROM VOTRAX LITERATURE
```

$EFRMF(@FEXP)

THE INTEGER CONTAINED IN @FEXP IS CONVERTED TO AN E-TYPE NUMBER. NOTE THAT THE CONVERSION IS EXACT IF AND ONLY IF @FEXP IS LESS VHAN (APPROXIMATELY) 8 DECIMAL DIGITS.

$FFRME(@EEXP)

THE FLOATING POINT NUMBER CONTAINED IN @EEXP IS CONVERTED TO AN INTEGER. THIS CONVERSION IS ACCOMPLISHED BY IGNORING ANY FRACTIONAL PART. IF THE RESULTING INTEGER IS TOO LARGE IN VALUE (> 2&&31-1 OR < -2**31) THEN THE RESULT IS THE LARGEST INTEGER OF APPROPRIATE SIGN, AND THE EXECUTION VARIABLE $FFEI IS SET TO 1. IF THE CONVERSION IS ACCOMPLISHED WITHOUT ERROR $FFEI IS SET TO ZERO.

$CFRMF(@FEXP)

THE F-TYPE NUMBER IDENTIFIED BY @FEXP IS CONVERTED TO A CHARACTER STRING CONTAINING A DECIMAL REPRESENTATION OF THE NUMBER. THE FORM OF THE RESULT IS THE NUMBER, PRECEDED BY A '-' IF NEGATIVE, WITH LEADING ZEROS SUPPRESSED. A ZERO VALUE IS REPRESENTED BY A SINGLE ZERO WITH NO SIGN. A 'NEGATIVE ZERO' IS NOT POSSIBLE.

$FFRMC(@CEXP)

THE STRING CONTAINED IN @CEXP IS CONVERTED TO AN F-TYPE NUMBER. @CEXP MAY CONTAIN A STRING OF THE FORMAT 'B...SD,..B...' WHERE THE B'S INDICATE OPTIONAL SPACES, THE S INDICATES AN OPTIONAL SIGN (+ OR -) AND THE D INDICATES DECIMAL DIGITS. IF THE VALUE REPRESENTED BY THE STRING IS OF TOO LARGE A MAGNITUDE TO BE REPRESENTED BY AN F-TYPE NUMBER, THEN THE RESULT IS THE LARGEST NUMBER OF APPROPRIATE SIGN AND $FFCI IS SET TO -1 OR -2 DEPENDING ON WHETHER THE SIGN OF THE RESULT IS POSITIVE OR NEGATIVE, RESPECTIVELY. IF THE STRING IS NOT OF THE CORRECT FORM, THEN A RESULT OF ZERO IS RETURNED AND THE EXECUTION VARIABLE $FFCI IS SET TO THE POSITION OF THE CHARACTER IN @CEXP WHERE THE ERROR WAS DETECTED.
IF NO ERRORS OCCUR, $FFCI IS SET TO ZERO.

```
EXTERNAL DOCUMENTATION

$CFRME(@EEXP<,@FEXP>)
        @EEXP IS CONVERTED TO A CHARACTER STRING ACCORDING TO
        THE FOLLOWING RULES-
        IF @FEXP IS NOT SPECIFIED THEN-
           IF THE ABSOLUTE VALUE OF @EEXP IS LESS THAN OR
           EQUAL TO 9999.9999 AND GREATER THAN OR EQUAL TO
           .0001 THEN THE RESULTING CHARACTER STRING HAS THE
           FORM 'ZZZZ.9999' WHERE THE Z'S INDICATE ZERO SUPPRESSED
        P  POSITIONS (WHICH ARE NOT RETURNED IF THEY CONTAIN SPACES),

           AND THE 9'S INDICATE POSITIONS WHICH WILL BE ZERO
           FILLED UNTIL THE LAST NON-ZERO DIGIT IS ENCOUNTERED.
           IF THE NUMBER IS NEGATIVE, THEN A '-' IS INSERTED PRIOR
           TO THE FIRST CHARACTER OF THE RESULTING STRING.

           IF THE ABOVE CONDITION IS NOT MET THEN THE RESULTING
           STRING WILL HAVE THE FORM 'ZZZZZZZ.9999999ESXX' WHERE
           THE Z'S AND 9'S ARE HANDLED AS ABOVE, AS IS THE SIGN.
           THE 'E' IS AN ACTUAL CHARACTER IN THE RESULTING STRING
           AND THE 'S' IS THE SIGN (+ OR -) OF THE
           EXPONENT (WHICH IS REPRESENTED BY THE 'XX').  THE
           MEANING OF THIS FORM IS THAT THE NUMBER PRECEEDING THE 'E'

           SHOULD BE MULTIPLIED BY TEN RAISED TO THE XX'TH POWER TO
           PRIVIDE THE CORRECT VALUE.  THUS THE FORM IS VERY SIMILAR
           TO STANDARD SCIENTIFIC NOTATION.

        IF @FEXP IS SPECIFIED THEN CONVERSION IS AS ABOVE EXCEPT
        THAT THE NUMBER OF DIGITS PRINTED AFTER THE DECIMAL POINT
        IS EQUAL TO @FEXP.  HOWEVER, IN NO CASE WILL MORE THAN 7
        SIGNIFICANT DIGITS BE GENERATED, SINCE THIS IS THE
        PRECISION LIMIT OF THE COMPUTER.

        IF THE RESULTING STRING IS LONGER THAN THE OUTPUT
        CHARACTER STRING INTO WHICH IT IS BEING STORED,
        THEN $FCFI IS SET TO -1.

$EFRMC(@CEXP)
        THE CHARACTER STRING IN @CEXP IS CONVERTED TO AN E-TYPE
        NUMBER ACCORDING TO THE FOLLOWING RULES-
        THE STRING MUST BE OF THE FORM 'B...SD...PD...B...ESX...B...'
        WHERE THE B'S INDICATE OPTIONAL BLANKS,
        THE FIRST 'S' INDICATES AN OPTIONAL SIGN (+ OR -),
        THE D'S INDICATE DECIMAL DIGITS TO BE INTERPRETED AS THE
        SIGNIFICANT DIGITS OF THE NUMBER,
        THE 'P' INDICATES AN OPTIONAL DECIMAL POINT,
        THE SECOND 'S' INDICATES THE SIGN OF THE EXPONENT@
```

EXTERNAL DOCUMENTATION

#L2DH IS OPTIONAL, S
THE 'E' INDICATES THE LETTER E, WHICH IS ALSO OPTIONAL.

THE FOLLOWING ADDITIONAL RULES APPLY -

1. IF @CEXP IS A NULL STRING, THEN $FECI IS SET TO -1.
2. IF THE STRING REPRESENTS A NUMBER TOO LARGE IN MAGNITUDE TO BE STORED IN AN E-TYPE NUMBER, THEN $FECI IS SET TO -2 AND THE RESULT IS SET EQUAL TO THE LARGEST NUMBER OF APPROPRIATE SIGN.
3. IF THE STRING REPRESENTS A NUMBER TOO SMALL TO BE STORE IN AN E-TYPE NUMBER, THEN $FECI IS SET TO -3 AND THE RESULT IS THE SMALLEST NON-ZERO NUMBER OF APPROPRIATE SIGN.
4. IF ANY OTHER FORMAT ERROR IS ENCOUNTERED, THEN $FECI IS SET TO THE POSITION IN @CEXP WHERE THE ERROR WAS DISCOVERED AND THE RESULT IS ZERO.
5. IF NO ERRORS OCCUR, THEN $FEFCI IS SET TO ZERO.

$INSTR(@FEXP,@CEXP-1,@CEXP-2,...)

$CEXP-1 IS SEARCHED FOR 'OCCURRANCES' OF @CEXP-2, @CEXP-3,... AND A VALUE OF TRUE IS RETURNED IF @CEXP-2, @DEXP-3,... ALL 'OCCUR' IN @CEXP-1. OTHERWISE, A VALUE OF FALSE IS RETURNED. @CEXP-N IS SAID TO 'OCCUR' IN @CEXP-1 IF AT LEAST ($FLEN(@CEXP-N)*@FEXP)/100 CHARACTERS OF @CEXP-N OCCUR CONTINUOUSLY ANYWHERE IN @CEXP-1.

$INSTRO(@FEXPI,@CEXP-1,@CEXP-2,...)

THIS FUNCTION BEHAVES LIKE $INSTR WITH THE EXCEPTION THAT ANY 'OCCURRANCE' OF @CEXP-N MUST FOLLOW THE 'OCCURANCE' OF @CEXP-(N-1) IN @CEXP-1.

$CSBSTR(@CEXP,@FEXP-1,@FEXP-2)

A STRING OF CHARACTERS FROM @CEXP, BEGINNING WITH THE '@FEXP-1'TH AND CONTINUING FOR @FEXP-2 CHARACTERS IS RETURNED. IF @FEXP-1+@FEXP-2 EXCEEDS $FLEN(@CEXP) THE ONLY THOSE CHARACTERS ACTUALLY IN @CEXP ARE RETURNED. IF $FEXP IS ZERO OR NEGATIVE, OR @FEXP-2 IS NEGATIVE THEN THE EXECUTION VARIABLE $FSBSI IS SET TO -1, AND A NULL STRING IS RETURNED. IF @FEXP-2 IS ZERO THEN A NULL STRING IS RETURNED AND $FSBSI IS SET TO ZERO.

$CNXTWD(@CEXP-1,@FEXP<,@FVAR<,@CEXP-2>>)

THE NEXT 'WORD' IN @CEXP-1 IS RETURNED. @CEXP-1 IS SCANNED BEGINNING WITH THE CHARACTER AT POSITION @FEXP UNTIL THE FIRST CHARACTER WHICH IS *NOT* A DELIMITER IS FOUND.

SUBSEQUENT CHARACTERS FORM THE RESULT UNTIL THE NEXT DELIMITER IS FOUND, AT WHICH TIME @FVAR IS SET (IF SPECIFIED) TO THE POSITION IN @CEXP-1 OF THE CHARACTER WHICH CAUSED THE SCAN TO END. IF THE END OF @CEXP-1 IS ENCOUNTERED DURING THE SCAN, THEN THE CHARACTERS PRIOR TO THE END OF THE STRING ARE RETURNED AND @FVAR IS SET TO ZERO IF IT WAS SPECIFIED.

IF @CEXP-2 IS SPECIFIED THEN EACH CHARACTER IN @CEXP-2 IS USED AS A DELIMITER; OTHERWISE, THE FOLLOWING CHARACTERS CONSTITUTE THE 'DEFAULT' DELIMITERS.

BLANKS	.	:
?	=	! (EXCLAMATION MARK)
+	,	-
;	*	/

$CBEL (@FEXP)
RETURNS @FEXP 'BELL' CHARACTERS.

$CBSP (@FEXP)
RETURNS @FEXP 'BACKSPACE' CHARACTERS.

$CLF (@FEXP)
RETURNS FEXP 'LINE FEED' CHARACTERS.

SCSP (@FEXP)
RETURNS @FEXP SPACES.

$CCR (@FEXP)
RETURNS @FEXP 'CARRIAGE RETURN' CHARACTERS.

$CRPT (@CEXP,@FEXP)
THE CHARACTER STRING INDICATED BY @CEXP IS DUPLICATED @FEXP TIMES AND THE RESULTING VALUE RETURNED.
EXAMPLE, $CRPT ('ABC',3) RESULTS IN 'ABCABCABC'.

$CRLF (@FEXP)
RETURNS @FEXP 'REVERSE LINE FEED' CHARACTERS. IF OUTPUT VIA A 'TEXT' COMMAND, EACH RLF CHARACTER CAUSES THE CURSOR TO MOVE *UP* ONE LINE.

IN FOLLOWING THREE ROUTINES THE ARGUMENT IDENTIFIED AS @FVAR MAY NAME A NEGATIVE VALUE AS FOLLOWS

-1 - INDICATES A *PERMANENT* DISK ERROR WHICH IS NORMALLY A HARDWARE ERROR. NO ADDITIONAL RECOVERY SHOULD BE ATTEMPTED.

-2 - INDICATES A WORD, OR PHONEME FIELD PASSED AS INPUT EXCEEDED 32 CHARACTERS.
ADDITIONALLY THE VARIABLE @CVAR-W IS USED AS A TEMPORARY WORK AREA AND MUST BE AT LEAST   BYTES LONG.

$FRDWD (@CEXP, @CVAR, @CVAR-W)
THE 'WORD' INDICATED BY @CEXP IS 'LOOKED UP' IN THE VDF AND THE RESULTING PHONEMES ARE RETURNED IN 'INTERNAL' FORM IN @CVAR-1. THE RETURNED VALUE IS SET TO ZERO IF THE 'LOOK UP' IS SUCCESSFUL AND TO 1 IF THE VALUE OF @CEXP IS NOT FOUND.

$FDLWD (@CEXP, @CVAR-W)
THE 'WORD' INDICATED BY @CEXP IS REMOVED FROM THE VDF IF PRESENT, AND A VALUE OF ZERO RETURNED.
A VALUE OF 1 IS RETURNED IF THE WORD WAS NOT PRESENT IN THE VDE.

$FADWD (@CEXP-1, @CEXP-2, @CVAR-W)
THE 'WORD' INDICATED BY @CEXP-1 IS ADDED TO THE VDF WITH THE PHONEME INDICATED BY @CEXP-2 AND A VALUE OF ZERO IS RETURNED.
THE FOLLOWING EXCEPTIONS MAY OCCUR AND RESULT IN THE RETURNED VALUE SHOWN.
1 - 'WORD' (@CEXP-1) ADDED IN OVERFLOW AREA - NOT IN ERROR.
2 - 'WORD' (@CEXP-1) ALREADY PRESENT IN VDF.
2 - 'WORD' WAS *NOT* ADDED BECAUSE OF INSUFFICIENT SPACE ON THE DISK FILE.

$FDCOD (@CEXP, @CVAR)
THE RESULT OF EVALUATING @CEXP IS INTERPRETED AS PHONEMES IN 'INTERNAL' FORM, WHICH ARE CONVERTED TO THE 'EXTERNAL' FORM DESCRIBED IN 'PHONETIC INPUT DATA FORMAT' AND PLACED IN @CVAR.
A VALUE OF ZERO IS RETURNED OF NO ERRORS ARE ENCOUNTERED.
A VALUE OF -1 IS RETURNED IF THE INPUT STRING (@CEXP) WAS NULL AND @CVAR IS SET TO NULL
A VALUE OF -2 IS RETURNED OF @CEXP CONTAINED A CHARACTER COMPRISED OF 'XX111111' WHERE THE X'S MAY BE 0 OR 1.
THE OUTPUT STRING (@CVAR) WILL CONTAIN '???' IN THE CORRESPONDING POSITION.
A VALUE OF 1 TO 253 IS RETURNED IF THERE WAS INSUFFICIENT ROOM IN @CVAR TO CONTAIN THE CONVERTED STRING. THE ACTUAL VALUE RETURNED IS ONE *GREATER THAN* THE LAST PHONEME SUCCESSFULLY CONVERTED.

EXTERNAL DOCUMENTATION

$FRAND (@FVAR)
A RANDOM NON-ZERO POSITIVE INTEGER IS RETURNED AND THE VALUE OF @FVAR IS CHANGED. DIFFERENT CALLS TO $FRAN WITH IDENTICAL VALUES OF @FVAR WILL PRODUCE THE SAME VALUE AS AS RESULT.

$CTOD (1)
THE CURRENT TIME OF DAY IS RETURNED AS 8 CHARACTERS IN THE FORM 'HH.MM.SS'. THE TIME IS MAINTAINED AS A 24 HOUR CLOCK WHERE 00.00.00 IS MIDNIGHT.

$FTOD (1)
THE CURRENT TIME OF DAY IS RETURNED AS AN INTEGER INDICATING THE NUMBER OF SECONDS SINCE MIDNIGHT.

$CTUC (@CEXP)
THE RESULT OF EVALUATING @CEXP IS EXAMINED FOR LOWER CASE ALPHABETIC CHARACTERS, WHICH ARE TRANSFORMED INTO THE CORRESPONDING UPPER CASE CHARACTER AND THE RESULTING STRING IS RETURNED.

$CTLC (@CEXP)
ANY UPPER CASE CHARACTERS IN @CEXP ARE TRANSFORMED TO THE CORRESPONDING LOWER CASE CHARACTERS AND THE RESULTING STRING IS RETURNED.

$FMLEN (@CVAR)
RETURNS THE MAXIMUM LENGTH OF @CVAR - I.E., THE LENGTH WHICH APPEARS IN THE VDEF OR CDEF STATEMENT FOR THE VARIABLE.

$FLEN (@CEXP)
RETURNS THE CURRENT LENGTH OF THE RESULT OF EVALUATING @CEXP.

$FNSEG (@CEXP)
RETURNS THE NUMBER OF SEGMENTS IN THE VOCAL TEXT IDENTIFIED BY THE FIRST 6 CHARACTERS RESULTING FROM THE EVALUATION OF @CEXP.

$FLOC (@CEXP-1, @CEXP-2)
THE STRING RESULTING FROM EVALUATING @CEXP-1 IS SEARCHED FOR THE STRING RESULTING FROM @CEXP-2. IF A MATCH IS FOUND THE POSITION IN @CEXP-1 OF THE FIRST CHARACTER OF THE MATCHING STRING IS RETURNED, OTHERWISE ZERO IS RETURNED.
EXAMPLE:
```
  $FLOC ('ABCD', 'BC')
  WOULD RETURN 2

  $FLOC ('ABCD', 'CDE')
  WOULD RETURN ZERO
```

4. EXECUTION VARIABLES

$FLINE- INTEGER
GIVES THE CURRENT LINE NUMBER OF THE ALPHA-CURSOR.

$FPOS- INTEGER
GIVES THE CURRENT POSITION, WITHIN THE LINE, OF THE ALPH-CURSOR.

$CDATE- 8 CHARACTER VARIABLE
CONTAINS THE CURRENT DATE IN THE FORM MM/DD/YY.

$CDOW- 6 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT DAY OF THE WEEK IE, -MONDAY, TUESDAY,...

$CMNTH- 3 TO 9 CHARACTER VARIABLE CONTAINING THE CURRENT MONTH IE. JANUARY, MAY,...

$CFNAM- 0 TO 40 CHARACTER VARIABLE
CONTAINS THE FIRST NAME OF THE STUDENT CURRENTLY RUNNING.

$CLNAM- 0 TO 40 CHARACTER VARIABLE
CONTAINS THE LAST NAME OF THE STUDENT CURRENTLY RUNNING.

$FLAT- POSITIVE OR ZERO INTEGER
'LATENCY' IN SECONDS OF THE MOST RECENT INPUT I.E. - THE ELAPSED TIME FROM WHEN THE LAST 'READ' OR 'READCC' WAS ISSUED TO WHEN THE LAST CHARACTER OF INPUT WAS RECEIVED.

$CSECT- 0 TO 6 CHARACTERS
CONTAINS THE NAME OF THE SECTON CURRENTLY EXECUTING. NULL IF REFERENCED AT THE COURSE LEVEL.

$FID- 9 CHARACTER VARIABLE
CONTAINS THE ID NUMBER OF THE STUDENT CURRENTLY RUNNING.

$CRSE - 1 TO 6 CHARACTER VARIABLE
CONTAINS THE NAME OF THE COURSE CURRENTLY RUNNING.

$CLBL - 1 TO 6 CHARACTERS
CONTAINS THE LABEL OF THE LAST LABELED STATEMENT EXECUTED.

EXTERNAL DOCUMENTATION

$FMLIN- INTEGER
GIVES THE MAXIMUM LINE NUMBER ALLOWABLE ON THE TERMINAL IN USE.

$FMPOS- INTEGER
GIVES THE MAXIMUM POSITION WITHIN A LINE ALLOWABLE ON THE TERMINAL IN USE.

5. GRAPHICS: TERMINAL CONTROL SYSTEM

5.1 VIRTUAL GRAPHICS

5.1.1. THE VIRTUAL DISPLAY

THE VIRTUAL DISPLAY IS AN IMAGINARY TWO-DIMENSIONAL SURFACE WITH A RANGE IN BOTH THE X AND Y DIRECTIONS EQUAL TO THE RANGE OF A SINGLE PRECISION FLOATING POINT NUMBER. USING THE VIRTUAL DISPLAY THE USER MAY CONSTRUCT DRAWINGS, PICTURES, AND GRAPHS OF EXTREME COMPLEXITY AND DETAIL.

SINCE THE UNIT OF MEASUREMENT OF THE VIRTUAL DISPLAY IS ARBITRARY, IT MAY BE ASSUMED TO BE REPRESENTATIVE OF ANY MEASUREMENT UNIT FROM MICRONS TO LIGHT-YEARS, WITH ALL MEASUREMENTS TRANSLATED TO THE ASSUMED UNIT FOR THE GIVEN DRAWING. FOR EXAMPLE, THE USER DECIDES THAT THE BASIC UNIT OF THE VIRTUAL DISPLAY WILL REPRESENT INCHES. THEN THE VIRTUAL COORDINATE (2.,0.5) REPRESENTS A POINT TWO INCHES TO THE RIGHT OF THE ORIGIN ON THE X-AXIS AND ONE HALF INCH UP ON THE Y-AXIS. TO INDICATE THE POINT ONE MILE (63,360 INCHES) TO THE LEFT OF THE ORIGIN ALONG THE X-AXIS, THE VIRTUAL COORDINATE (-63360.0, 0.0) WOULD BE USED.

THE VIRTUAL DISPLAY IS SIMILAR TO NORMAL DISPLAYS AND PLOTTING DEVICES IN THAT THERE IS A MOVABLE POINT WHICH MAY BE THOUGHT OF AS THE WRITING CURSOR ON THE VIRTUAL DISPLAY. THIS POINT IS CALLED THE IMAGINARY BEAM, AND ITS POSITION IS THE VIRTUAL COORDINATE WHICH REPRESENTS THE LOCATION OF THE WRITING CURSOR AS IF THE VIRTUAL DISPLAY WERE AN ACTUAL DEVICE.

SINCE ONLY THE PORTIONS OF VECTORS AND THE POINTS WHICH LIE WITHIN THE CURRENT WINDOW ARE DISPLAYED, THE IMAGINARY BEAM POSITION DOES NOT ALWAYS REPRESENT THE ACTUAL STORAGE BEAM POSITION. THE ACTUAL BEAM IS REPRESENTED ON THE VIRTUAL DISPLAY BY THE REAL BEAM, WHICH IS UPDATED TO REFLECT THE ACTUAL OUTPUT TO THE TERMINAL. WHEN ENTERING VIRTUAL GRAPHICS OR WHENEVER THE WINDOW IS REDEFINED, BOTH THE IMAGINARY BEAM AND THE REAL BEAM ARE SET AT THE VIRTUAL COORDINATE REPRESENTATION (ACCORDING TO THE LATEST WINDOW DEFINITION) OF THE ACTUAL BEAM POSITION.

5.1.2 WINDOWING

ALL OR ANY PORTION OF THE VIRTUAL DISPLAY MAY BE VIEWED AT ANY TIME THROUGH THE TECHNIQUE OF WINDOWING. THE PROTION OF THE

VIRTUAL DISPLAY TO BE SHOWN IS DEFINED BY RECTANGULAR BOUNDARIES. THIS RECTANGLE IS CALLED THE VIRTUAL WINDOW, AND ONLY THOSE VECTORS WHICH PASS THROUGH THE VIRTUAL WINDOW WILL BE DISPLAYED.

IT IS NOT NECESSARY TO USE ALL OF THE SCREEN FOR DISPLAY OF THE VIRTUAL WINDOW. THE USER MAY DEFINE A RECTANGULAR SECTION OF ANY SIZE AND LOCATION ON THE SCREEN AS THE AREA IN WHICH THE WINDOW WILL APPEAR. THIS RECTANGLE IS CALLED THE SCREEN WINDOW AND TOGETHER WITH THE VIRTUAL WINDOW DEFINES THE TRANSFORMATION BETWEEN THE VIRTUAL DISPLAY AND THE SCREEN.

ELIMINATION OF VECTORS AND PORTIONS OF VECTORS WHICH LIE OUTSIDE OF THE WINDOW WILL BE DONE AUTOMATICALLY BY THE VIRTUAL GRAPHIC ROUTINES AS WELL AS THE SCALING AND CONVERSION OF THESE VECTORS THAT ARE CONTAINED IN OR PASS THROUGH THE WINDOW.

IT SHOULD BE NOTED HERE THAT THE SCALING IS NOT RELATED TO THE SIZE OF THE VIRTUAL DISPLAY OR THE SCREEN, BUT IS DETERMINED SOLELY BY THE WINDOW DEFINITION. ALSO, SINCE THE X AND Y EXTENTS OF THE WINDOW MAY BE SEPARATELY DEFINED, THE X AND Y SCALING ARE INDEPENDENT. THIS ALLOWS FOR THE EMPHASIS OF EITHER X OR Y DATA VALUES. CARE MUST BE TAKEN THAT UNWANTED DISTORTION IS NOT INTRODUCED BY ERRONEOUS WINDOW DEFINITIONS. THE INITIAL WINDOW DEFINITION IS SET SO THAT THE PORTION OF THE VIRTUAL DISPLAY WITH COORDINATES EQUIVALENT TO THE SCREEN WILL BE DISPLAYED:

VIRTUAL WINDOW INITIAL VALUES:

X MINIMUM - O., X EXTENT - 1023
Y MINIMUM - O., Y EXTENT - 780

SCREEN WINDOW INITIAL VALUES:

X MINIMUM - O, X EXTENT - 1023
Y MINIMUM - O, Y EXTENT - 780

THE USER UTILIZES THE VIRTUAL DISPLAY BY FIRST DEFINING HIS WINDOW AND THEN CONSTRUCTING HIS DRAWING, PICTURE, OR GRAPH WITH THE USE OF THE VIRTUAL GRAPHIC ROUTINES. THE USER MAY DISPLAY SEVERAL PORTIONS OF THE VIRTUAL DISPLAY AT ONE TIME BY REDEFINING THE WINDOW AND REPROCESSING THE VIRTUAL DISPLAY FOR EACH OR MAY SUPERIMPOSE DATA FROM "SEVERAL" VIRTUAL DISPLAYS BY USING A COMMON SCREEN WINDOW. ALL TRANSFORMATIONS BETWEEN THE VIRTUAL DISPLAY AND THE SCREEN WILL BE BASED UPON THE LATEST WINDOW DEFINITIONS.

SETTING THE VIRTUAL WINDOW

THE PORTION OF THE VIRTUAL DISPLAY TO BE VIEWED IS DETERMINED BY THE VIRTUAL WINDOW. THE VIRTUAL WINDOW IS DEFINED BY A POINT WHICH REPRESENTS ITS LOWER LEFT CORNER AND THE EXTENT OF THE WINDOW IN THE X AND Y DIRECTIONS.

CALLING SEQUENCE:

```
    FCALL VWINDO,(@EEXP-X,@EEXP-XL,@EEXP-Y,@EEXP-&L)
WHERE:    @EEXP-X - MINIMUM X-COORDINATE OF THE VIRTUAL WINDOW.

         @EEXP-XL - EXTENT OF THE VIRTUAL WINDOW IN THE
              X-DIRECTION.
          @EEXP-Y - MINIMUM Y-COORDINATE OF THE VIRTUAL WINDOW.

         @EEXP-YL - EXTEND OF THE VIRTUAL WINDOW IN THE
              Y-DIRECTION.
```

SETTING THE SCREEN WINDOW

THE SCREEN WINDOW DEFINES THE SECTION OF THE SCREEN INTO WHICH THE VIRTUAL WINDOW WILL BE TRANSFORMED. ITS DEFINITION IS SIMILAR TO THAT OF THE VIRTUAL WINDOW.

CALLING SEQUENCE:

```
    FCALL SWINDO,(@FEXP-IX,@FEXP-LX,@FEXP-IY,@FEXP-LY)

WHERE:    @FEXP-IX - MINIMUM SCREEN X-COORDINATE OF THE SCREEN
               WINDOW.
          @FEXP-LX - EXTENT OF THE SCREEN WINDOW IN THE
               X-DIRECTION.
          @FEXP-IY - MINIMUM SCREEN Y-COORDINATE OF THE SCREEN
               WINDOW.
          @FEXP-LY - EXTENT OF THE SCREEN WINDOW IN THE
               Y-DIRECTION.
```

5.1.3. ABSOLUTE VECTORS

VIRTUAL GRAPHICS ALLOW THE USER TO DRAW, MOVE, OR POINT PLOT TO ANY PARTICULAR POINT ON THE VIRTUAL DISPLAY WITH AN ABSOLUTE VECTOR. AN ABSOLUTE VECTOR EXTENDS FROM THE CURRENT IMAGINARY BEAM POSITION TO THE LOCATION SPECIFIED BY THE GIVEN VIRTUAL COORDINATES, (X,Y). MODE ENTRY AND TRANSFORMATION TO SCREEN VECTORS, INCLUDING WINDOWING AND CLIPPING, IS AUTOMATIC.

DRAW

A VECTOR MAY BE DRAWN FROM THE LAST POINT ON THE VIRTUAL DISPLAY AT WHICH THE IMAGINARY BEAM WAS POSITIONED TO A SPECIFIED POINT WITH DRAWA. ONLY THAT PORTION, IF ANY, OF THE VECTOR WHICH PASSES THROUGH THE VIRTUAL WINDOW WILL BE VISIBLE. ON RETURN FROM THIS ROUTINE, THE IMAGINARY BEAM WILL BE POSITIONED AT THE GIVEN VIRTUAL COORDINATES.

CALLING SEQUENCE:

```
    FCALL DRAWA,(@EEXP-X,@EEXP-Y)

WHERE:      @EEXP-X - VIRTUAL X-COORDINATE OF THE POINT.
            @EEXP-Y - VIRTUAL Y-COORDINATE OF THE POINT.
```

MOVE

A MOVE (AN INVISIBLE VECTOR) TO ANY PARTICULAR POINT ON THE VIRTUAL DISPLAY MAY BE MADE BY CALLING MOVEA. ON RETURN FROM THIS ROUTINE, THE IMAGINARY BEAM WILL BE POSITIONED AT THE GIVEN VIRTUAL COORDINATES.

CALLING SEQUENCE:

```
    FCALL MOVEA,(@EEXP-X,@EEXP-Y)

WHERE:      @EEXP-X - VIRTUAL X-COORDINATE OF THE POINT.
            @EEXP-Y - VIRTUAL Y-COORDINATE OF THE POINT.
```

POINT PLOT

A POINT MAY BE PLOTTED AT ANY LOCATION ON THE VIRTUAL DISPLAY WITH POINTA. ONLY IF THE GIVEN VIRTUAL COORDINATES ARE WITHIN THE VIRTUAL WINDOW WILL A POINT ACTUALLY BE DISPLAYED. ON RETURN FROM THIS ROUTINE, THE IMAGINARY BEAM WILL BE POSITIONED AT THE GIVEN VIRTUAL COORDINATES.

CALLING SEQUENCE:

```
    FCALL POINTA,(@EEXP-X,@EEXP-Y)

WHERE:     @EEXP-X - VIRTUAL X-COORDINATE OF THE POINT.
           @EEXP-Y - VIRTUAL Y-COORDINATE OF THE POINT.
```

DASH

A DASHED LINE MAY BE DRAWN FROM THE LAST POINT AT WHICH THE IMAGINARY BEAM WAS POSITIONED TO A SPECIFIED POINT WITH DASHA. ONLY THAT PORTION, IF ANY, WHICH PASSES THROUGH THE VIRTUAL WINDOW WILL BE VISIBLE. ON RETURN FROM THIS ROUTINE, THE IMAGINARY BEAM WILL BE POSITIONED AT THE GIVEN VIRTUAL COORDINATE.

CALLING SEQUENCE:

```
    FCALL DASHA,(@EEXP-X,@EEXP-Y,@EEXP-L)
```

```
WHERE:     @EEXP-Y - VIRTUAL X-COORDINATE OF THE POINT.
           @EEXP-Y - VIRTUAL Y-COORDINATE OF THE POINT.
           @FEXP-L - DASHED LINE SPECIFICATION.
                  A DASHED LINE IS SPECIFIED BY CON-
                  CATENATING INTEGERS DESCRIBING THE
                  LINE SEGMENT LENGTH AND VISIBILITY.
                  ALL CODES EXCEPT 9 SHOULD HAVE 2 OR
                  MORE INTEGERS.
                  1     5 RASTER UNITS, VISIBLE.
                  2     5 RASTER UNITS, INVISIBLE.
                  3    10 RASTER UNITS, VISIBLE.
                  4    10 RASTER UNITS, INVISIBLE.
                  5    25 RASTER UNITS, VISIBLE.
                  6    25 RASTER UNITS, INVISIBLE.
                  7    50 RASTER UNITS, VISIBLE.
                  8    50 RASTER UNITS, INVISIBLE.
                  9    ALTERNATE BRIGHT AND DARK
                       BETWEEN POINTS.

           NOTE:  SCREEN DEFINITION DOES NOT AFFECT
                  DASH SIZE.
```

5.1.4. RELATIVE VECTORS

VIRTUAL GRAPHICS ALSO ALLOW THE USER TO DEFINE A DISPLACEMENT OF GIVEN LENGTH AND DIRECTION ON THE VIRTUAL DISPLAY THROUGH THE USE OF RELATIVE VECTORS. RELATIVE VECTORS OFFER THE ABILITY TO CREATE SIMILAR STRUCTURES AT DIFFERENT POSITIONS ON THE VIRTUAL DISPLAY WITH ONE SET OF DISPLAY COMMANDS. CONVERSION OF THE RELATIVE VECTOR TO AN ABSOLUTE VECTOR, MODE ENTRY, AND TRANSFORMATION TO SCREEN VECTORS, INCLUDING WINDOWING AND CLIPPING, IS AUTOMATIC. ON RETURN FROM A RELATIVE VECTOR ROUTINE, THE IMAGINARY BEAM WILL BE LOCATED AT THE POINT DEFINED BY ITS INITIAL POSITION PLUS THE DISPLACEMENT VALUE.

EXTERNAL DOCUMENTATION

DRAW

A RELATIVE VECTOR MAY BE DRAWN ON THE VIRTUAL DISPLAY FROM THE CURRENT IMAGINARY BEAM LOCATION WITH DRAWR. THE X AND Y DISPLACEMENT VALUES WHICH DEFINE THE LENGTH AND DIRECTION OF THE RELATIVE VECTOR ARE INPUT ARGUMENTS TO DRAWR. ONLY THAT PORTION, IF ANY, OF THE RESULTANT VECTOR WHICH PASSES THROUGH THE VIRTUAL WINDOW WILL BE DISPLAYED.

CALLING SEQUENCE:

```
    FCALL DRAWR,(@EEXP-X,@EEXP-Y)

WHERE:      @EEXP-X - X-VALUE OF THE DISPLACEMENT.
            @EEXP-Y - Y-VALUE OF THE DISPLACEMENT.
```

MOVE

A RELATIVE MOVE ON THE VIRTUAL DISPLAY MAY BE GENERATED BY CALLING MOVER WITH THE X AND Y DISPLACEMENTS AS ARGUMENTS.

CALLING SEQUENCE:

```
    FCALL MOVER,(@EEXP-X,@EEXP-Y)

WHERE:      @EEXP-X - X-VALUE OF THE DISPLACEMENT.
            @EEXP-Y - Y-VALUE OF THE DISPLACEMENT.
```

POINT PLOT

POINTS MAY ALSO BE PLOTTED RELATIVE TO THE CURRENT IMAGINARY BEAM LOCATION ON THE VIRTUAL DISPLAY. IF THE RESULTANT POINT IS NOT WITHIN THE VIRTUAL WINDOW IT WILL NOT BE DISPLAYED.

CALLING SEQUENCE:

```
    FCALL POINTR,(@EEXP-X,@EEXP-Y)

WHERE:      @EEXP-X - X-VALUE OF THE DISPLACEMENT.
            @EEXP-Y - Y-VALUE OF THE DISPLACEMENT.
```

DASH

A DASHED LINE MAY BE DRAWN ON THE VIRTUAL DISPLAY FROM THE CURRENT IMAGINARY BEAM LOCATION TO A POINT DISPLACED BY X AND Y WITH DASHR. ONLY THAT PORTION, IF ANY, OF THE LINE WHICH PASSES THROUGH THE VIRTUAL WINDOW WILL BE DISPLAYED.

CALLING SEQUENCE:

```
    FCALL DASHR,(@EEXP-X,@EEXP-Y,@FEXP-L)

WHERE       @EEXP-X - X-VALUE OF THE DISPLACEMENT.
            @EEXP-Y - Y-VALUE OF THE DISPLACEMENT.
            @FEXP-L - DASHED LINE SPECIFICATION.
                   A DASHED LINE IS SPECIFIED BY CON-
                   CATENATING INTEGERS DESCRIBING THE
                   LINE SEGMENT LENGTH AND VISIBILITY.
                   ALL CODES EXCEPT 9 SHOULD HAVE 2 OR
                   MORE INTEGERS.
                   1     5 RASTER UNITS, VISIBLE.
                   2     5 RASTER UNITS, INVISIBLE.
                   3    10 RASTER UNITS, VISIBLE.
                   4    10 RASTER UNITS, INVISIBLE.
                   5    25 RASTER UNITS, VISIBLE.
                   6    25 RASTER UNITS, INVISIBLE.
                   7    50 RASTER UNITS, VISIBLE.
                   8    50 RASTER UNITS, INVISIBLE.
                   9    ALTERNATE BRIGHT AND DARK
                        BETWEEN POINTS.

            NOTE:  SCREEN DEFINITION DOES NOT AFFECT DASH
                   SIZE.
```

5.1.5. SCALING AND ROTATING

RELATIVE VECTORS ARE USED PRIMARILY TO CONSTRUCT OBJECTS OR ENTITIES WHICH MUST BE DISPLAYED AT A NUMBER OF DIFFERENT LOCATIONS ON THE VIRTUAL DISPLAY. HOWEVER, THE SIZE AND ORIENTATION OF THESE OBJECTS IS NOT ALWAYS THE SAME. FOR THIS REASON, RELATIVE VECTORS ARE AUTOMATICALLY SCALED AND ROTATED BY THE RELATIVE VECTOR ROUTINES ACCORDING TO THE SCALING FACTOR, TRSCAL, AND THE ROTATION FACTORS, TRCOSF AND TRSINF. TRSCAL, TRCOSF, AND TRSINF ARE ALL TERMINAL STATUS AREA VARIABLES. ALL INPUT ARGUMENTS TO THE RELATIVE VECTOR ROUTINES ARE UNSCALED AND UNROTATED. THE INPUT ARGUMENTS DEFINE THE NORMAL SIZE AND ORIENTATION FOR A RELATIVE VECTOR. SCALING AND ROTATION WILL NOT EFFECT ABSOLUTE VECTORS.

SETTING THE SCALE

CALLING THE ROUTINE 'SCALE' CAN BE USED TO ALTER THE LENGTH OF A RELATIVE VECTOR. ALL RELATIVE VECTORS ARE SCALED ACCORDING TO THE CURRENT VALUE. FOR EXAMPLE, IF A SECTION OF RELATIVE VECTOR CODING WILL CONSTRUCT A GIVEN OBJECT AND YOU REQUIRE THE OBJECT TO BE CONSTRUCTED AGAIN AT TWICE THE NORMAL SIZE, THEN CALL SCALE WITH AN ARGUMENT OF 2.0 AND RE-EXECUTE THE CODE WHICH WILL CONSTRUCT THE OBJECT. THE INITIAL SCALE FACTOR IS 1.0

SETTING THE ROTATION

RELATIVE VECTORS MAY ALSO HAVE THEIR DIRECTION ALTERED THROUGH THE RELATIVE VECTOR ROTATION ROUTINE 'ROTAT' -

CALLING SEQUENCE:

```
   FCALL ROTAT,(@EEXP)
```

WHERE:@EEXP - THE ROTATION ANGLE , IN DEGREES.

NOTE: @EEXP MAY BE NEGATIVE , BUT MUST BE IN THE RANGE -XXXXXXXXXX TO +XXXXXXXXXX.

ALL RELATIVE VECTORS ARE ROTATED ACCORDING TO THE VALUE OF THE CURRENT ROTATION FACTOR. IF THE USER WISHES TO CONSTRUCT AN OBJECT DEFINED BY RELATIVE VECTORS AT AN ANGLE DIFFERENT FROM THE NORMAL ORIENTATION, HE CALLS 'ROTAT' WITH THE DESIRED ANGLE OF ROTATION AND EXECUTES THE CODE FOR THE OBJECT.

5.2. DIRECT GRAPHICS

5.2.1 THE SCREEN

THE TERMINAL SCREEN IS A TWO-DIMENSIONAL SURFACE CONSISTING OF A DISCRETE 1024 X 1024 MATRIX OF ADDRESSABLE POINTS, OF WHICH 1024 X 781 OF THESE POINTS LIE IN THE VIEWABLE AREA (VECTORS JUST ABOVE 780 ON THE Y-AXIS MAY BE VISIBLE BUT MARGINAL IN QUALITY. FOR THE PURPOSES OF THIS MANUAL SUCH VECTORS ARE CONSIDERED PART OF THE UNVIEWABLE AREA) OF THE TERMINAL SCREEN. THE ORIGIN OF THE SCREEN LIES AT THE EXTREME LOWER LEFT CORNER.

OPERATIONS ON THE SCREEN ARE CALLED DIRECT GRAPHICS, AND ALLOW THE USER TO RELATE DIRECTLY WITH THE VISIBLE SURFACE OF THE TERMINAL. DIRECT GRAPHICS ALLOW THE USER TO WORK AT A BASIC GRAPHIC LEVEL AND AVOID THE OVERHEAD OF THE VIRTUAL CLIPPING AND TRANSFORMATION ROUTINES. THE USER HAS THE RESPONSIBILITY OF REMAINING ON SCREEN AS ALL COORDINATE INPUT TO DIRECT GRAPHIC ROUTINES ARE INTERPRETED AS MOD 1024.

DIRECT GRAPHICS ARE PRIMARILY USED WITH ALPHANUMERIC OUTPUT AND FOR DISPLAY LAYOUT. THE USER MAY FREELY ALTERNATE BETWEEN DIRECT AND VIRTUAL GRAPHICS. (NOTE: WHEN USING A VIRTUAL GRAPHIC ROUTINE AFTER USE OF DIRECT GRAPHICS OR ALPHANUMERIC OUTPUT, THE IMAGINARY BEAM IS CONSIDERED TO BE POSITIONED AT THE VIRTUAL COORDINATE THAT IS EQUIVALENT TO THE SCREEN COORDINATE OF THE BEAM POSITION UNDER THE CURRENT WINDOW TRANSFORMATION.)

5.2.2. ABSOLUTE VECTORS

AN ABSOLUTE VECTOR IN DIRECT GRAPHICS IS A DRAW, MOVE, OR POINT PLOT FROM THE CURRENT BEAM POSITION TO A SPECIFIED SCREEN COORDINATE. NO WINDOWING OR CLIPPING IS PERFORMED. MORE ENTRY AND APPROPRIATE OUTPUT HANDLING IS AUTOMATIC.

DRAW

A LINE MAY BE DRAWN FROM THE CURRENT BEAM POSITION TO ANY POINT ON THE SCREEN WITH DRWABS. ON RETURN FROM THIS ROUTINE, THE BEAM POSITION IS AT THE GIVEN SCREEN COORDINATE.

CALLING SEQUENCE:

```
    FCALL DRWABS,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:      @FEXP-IX - SCREEN X-COORDINATE OF THE GIVEN POINT.
            @FEXP-IY - SCREEN 7-COORDINATE OF THE GIVEN POINT.
```

MOVE

THE BEAM MAY BE MOVED TO ANY POINT ON THE SCREEN WITH MOVABS.

CALLING SEQUENCE:

```
    FCALL MOVABS,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:      @FEXP-IX - SCREEN X-COORDINATE OF THE GIVEN POINT.
            @FEXP-IY - SCREEN Y-COORDINATE OF THE GIVEN POINT.
```

EXTERNAL DOCUMENTATION

POINT PLOT

A POINT MAY BE PLOTTED AT ANY LOCATION ON THE SCREEN WITH PNTABS. ON RETURN, THE BEAM POSITION IS AT THE GIVEN SCREEN COORDINATES.

CALLING SEQUENCE:

```
    FCALL PNTABS,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:     @FEXP-IX - SCREEN X-COORDINATE OF THE GIVEN POINT.
           @FEXP-IY - SCREEN Y-COORDINATE OF THE GIVEN POINT.
```

DASH

A DASHED LINE MAY BE DRAWN FROM THE CURRENT BEAM POSITION TO ANY POINT ON THE SCREEN WITH DSHABS. ON RETURN FROM THIS ROUTINE, THE BEAM POSITION IS AT THE GIVEN SCREEN COORDINATE.

CALLING SEQUENCE:

```
    FCALL DSHABS,(@FEXP-IX,@FEXP-IY,@FEXP-L)
```

```
WHERE:     @FEXP-IX - SCREEN X-COORDINATE OF THE GIVEN POINT.
           @FEXP-IY - SCREEN Y-COORDINATE OF THE GIVEN POINT.
           @FEXP-L  - DASHED LINE SPECIFICATION.

                      A DASHED LINE IS SPECIFIED BY CON-
                      CATENATING INTEGERS DESCRIBING THE
                      LINE SEGMENT LENGTH AND VISIBILITY.
                      ALL CODES EXCEPT 9 SHOULD HAVE 2 OR
                      MORE INTEGERS.
                      1    5 RASTER UNITS, VISIBLE.
                      2    5 RASTER UNITS, INVISIBLE.
                      3   10 RASTER UNITS, VISIBLE.
                      4   10 RASTER UNITS, INVISIBLE.
                      5   25 RASTER UNITS, VISIBLE.
                      6   25 RASTER UNITS, INVISIBLE.
                      7   50 RASTER UNITS, VISIBLE.
                      8   50 RASTER UNITS, INVISIBLE.
                      9   ALTERNATE BRIGHT AND DARK BE-
                          TWEEN POINTS.
```

5.2.3. RELATIVE VECTORS

RELATIVE VECTORS MAY ALSO BE DRAWN ON THE SCREEN. HOWEVER, NO SCALING OR ROTATIONAL TRANSFORMATIONS ARE APPLIED TO THESE. MODE ENTRY AND APPROPRIATE OUTPUT HANDLING IS AUTOMATIC. DIRECT GRAPHIC RELATIVE VECTORS WILL CAUSE THE BEAM TO MOVE FROM ITS PRESENT POSITION TO THE POINT SPECIFIED BY THE DIRECT DISPLACEMENT.

THE USER AGAIN HAS THE RESPONSIBILITY OF REMAINING ON THE SCREEN. ALL RESULTANT VECTORS WILL HAVE THEIR COORDINATES INTERPRETED AS MOD 1024.

DRAW

A RELATIVE LINE MAY BE DRAWN ON THE SCREEN FROM THE CURRENT BEAM POSITION ACCORDING TO A GIVEN X AND Y DISPLACEMENT WITH DRWREL.

CALLING SEQUENCE:

```
    FCALL DRWREL,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:      @FEXP-IX - X-DISPLACEMENT IN SCREEN COORDINATES.
            @FEXP-IY - Y-DISPLACEMENT IN SCREEN COORDINATES.
```

MOVE

A RELATIVE MOVE MAY BE GENERATED BY MOVREL.

CALLING SEQUENCE:

```
    FCALL MOVREL,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:      @FEXP-IX - X-DISPLACEMENT IN SCREEN COORDINATES.
            @FEXP-IY - Y-DISPLACEMENT IN SCREEN COORDINATES.
```

POINT PLOT

A POINT MAY BE PLOTTED RELATIVE TO THE CURRENT BEAM POSITION WITH PNTREL.

CALLING SEQUENCE:

```
    FCALL PNTREL,(@FEXP-IX,@FEXP-IY)
```

```
WHERE:      @FEXP-IX - X-DISPLACEMENT IN SCREEN COORDINATES.
            @FEXP-IY - Y-DISPLACEMENT IN SCREEN COORDINATES.
```

DASH

A DASHED LINE MAY BE DRAWN ON THE SCREEN RELATIVE TO THE CURRENT BEAM POSITION ACCORDING TO A GIVEN X AND Y DISPLACEMENT WITH A DSHREL.

CALLING SEQUENCE:

```
    FCALL DSHREL,(@FEXP-IX,@FEXP-IY,@FEXP-L)
```

WHERE: @FEXP-IX - X-DISPLACEMENT IN SCREEN COORDINATES.
@FEXP-IY - Y-DISPLACEMENT IN SCREEN COORDINATES.
@FEXP-L - DASHED LINE SPECIFICATION.
A DASHED LINE IS SPECIFIED BY CONCATENATING INTEGERS DESCRIBING THE LINE SEGMENT LENGTH AND VISIBILITY. ALL CODES EXCEPT 9 SHOULD HAVE 2 OR MORE INTEGERS.

1	5 RASTER UNITS, VISIBLE.
2	5 RASTER UNITS, INVISIBLE.
3	10 RASTER UNITS, VISIBLE.
4	10 RASTER UNITS, INVISIBLE.
5	25 RASTER UNITS, VISIBLE.
6	25 RASTER UNITS, INVISIBLE.
7	50 RASTER UNITS, VISIBLE.
8	50 RASTER UNITS, INVISIBLE.
9	ALTERNATE BRIGHT AND DARK BETEEN POINTS.

2. MACRO DEFINITION LANGUAGE

6.1 STATEMENTS

LCLA	DEFINE A LOCAL ARITHMETIC VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS THAT ARE TO BE USED AS SET SYMBOLS, SEPARATED BY COMMAS; SET SYMBOLS MAY BE DEFINED AS SUBSCRIPTED SET SYMBOLS
LCLB	DEFINE A LOCAL BOOLEAN VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS THAT ARE TO BE USED AS SET SYMBOLS, SEPARATED BY COMMAS; SET SYMBOLS MAY BE DEFINED AS SUBSCRPTED SET SYMBOLS
LCLC	DEFINE A LOCAL CHARACTER VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS SEPARATED BY COMMAS; SET SYMBOLS MAY BE DEFINED AS SUBSCRIPTED SET SYMBOLS
GBLA	DEFINE A GLOBAL ARITHMETIC VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS THAT ARE TO BE USED AS SET SYMBOLS, SEPARATED BY COMMAS
GBLB	DEFINE A GLOBAL BOOLEAN VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS THAT ARE TO BE USED AS SET SYMBOLS, SEPARATED BY COMMAS
GBLC	DEFINE A GLOBAL CHARACTER VARIABLE NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - ONE OR MORE VARIABLE SYMBOLS THAT ARE TO BE USED AS SET SYMBOLS, SEPARATED BY COMMAS;SET SYMBOLS MAY BE DEFINED AS SUBSCRIPTED SET SYMBOLS
SETA	SET THE VALUE OF AN ARITHMETIC VARIABLE NAME ENTRY - SETA SYMBOL OPERAND ENTRY - AN ARITHMETIC EXPRESSION
SETB	SET THE VALUE OF A BOOLEAN VARIABLE NAME ENTRY - A SETB SYMBOL OPERAND ENTRY - A 0 OR A 1, OR LOGICAL EXPRESSION ENCOLSED IN PARANTHESES

EXTERNAL DOCUMENTATION

SETC	SET THE VALUE OF A CHARACTER VARIABLE NAME ENTRY - A SETC SYMBOL OPERAND ENTRY - A TYPE ATTRIBUTE, A CHARACTER EXPRESSION, A SUBSTRING NOTATION, OR A CONCATENATION OF CHARACTER EXPRESSIONS AND SUBSTRING NOTATIONS
AGO	TRANSFER CONTROL TO A SPECIFIED STATEMENT NAME ENTRY - A SEQUENCE SYMBOL OR NOT PRESENT OPERANT ENTRY - A SEQUENCE SYMBOL
AIF	CONDITIONALLY TRANSFER CONTROL TO A SPECIFIED STATEMENT NAME ENTRY - A SEQUENCE SYMBOL OR NOT PRESENT OPERAND ENTRY - A LOGICAL EXPRESSION ENCLOSED IN PARANTHESES, IMMEDIATELY FOLLOWED BY A SEQUENCE SYMBOL
MEXIT	STOP EXECUTING STATEMENTS IN THE CURRENT NAME ENTRY - A SEQUENCE SYMBOL OR NOT PRESENT OPERANT ENTRY - NOT USED, MUST NOT BE PRESENT; MAY ONLY BE USED AS PART OF A MACRO DEFINITION
ANOP	NULL STATEMENT;USED TO PROVIDE A REQUIRED LABEL NAME ENTRY - A SEQUENCE SYMBOL OPERAND ENTRY - NOT USED, MUST NOT BE PRESENT
ACTR	SPECIFY A LOOPING LIMIT NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - AN ARITHMETIC SETA EXPRESSION
MNOTE	PRINT AN ERROR MESSAGE NAME ENTRY - A SEQUENCE SYMBOL, A VARIABLE SYMBOL OR NOT PRESENT OPERAND ENTRY - A SEVERITY CODE, FOLLOWED BY A COMMA, FOLLOWED BY ANY COMBINATION OF CHARACTERS ENCLOSED IN APOSTROPHES; MAY ONLY BE USED AS PART OF A MACRO DEFINITION
MACRO	NAME ENTRY - NOT USED, MUST NOT BE PRESENT OPERAND ENTRY - NOT USED, SHOULD NOT BE PRESENT; MAY ONLY BE USED AS PART OF A MACRO DEFINTION
MEND	NAME ENTRY - A SEQUENCE SYMBOL OR NOT PRESENT OPERAND ENTRY - NOT USED, MUST NOT BE PRESENT; MAY ONLY BE USED AS A PART OF A MACRO DEFINITION

6.2. SYSTEM VARIABLE SYMBOLS

&SYSLIST(N)	REFERS TO THE N'TH POSITIONAL-OPERAND OF THE CURRENT MACRO
&SYSINDX	CONTAINS A FOUR CHARACTER FIELD GIVING A UNIQUE NUMBER FOR EACH MACRO CALL

6.3. FEATURES

6.3.1. SUBLIST NOTATION

IF A POSITIONAL OR KEYWORD OPERAND IS CODED WITH SURROUNDING PARENTHESIS THEN ITEMS WITHIN THE PARENTHESIS WHICH ARE SEPARATED BY COMMAS MAY BE REFERRED TO BY 'SUBLIST NOTATION'-
IF THE OPERAND WERE CODED AS 'A=(CAT,DOG,HOUSE)'
THEN &A(2) WOULD REFER TO 'DOG'

6.3.2. SUBSTRING NOTATION

A SUBSET OF THE CHARACTERS COMPRISING THE VALUE OF A SYMBOLIC PARAMETER OR ANY CHARACTER VARIABLE SYMBOL MAY BE REFERRED TO BYENCLOSING THE VARIABLE IN APOSTROPHES AND FOLLOWING THAT BY ASTARTING POSITION AND LENGTH ENCLOSED IN PARENTHESIS-
IF &A HAS THE VALUE 'ABCDEF' THEN '&A'(2,3) WILL YIELD THE VALUE 'BCD'

6.4 ATTRIBUTES

NUMBER	N'	GIVES THE NUMBER OF OPERANDS IN A SUBLIST IF USED WITH A SYMBOLIC PARAMETER; GIVES THE NUMBER OF POSITIONAL PARAMETERS IF USED AS N'&SYSLIST; GIVES THE NUMBER OF OPERANDS IN THE SUBLIST OF THE N'TH POSITIONAL OPERAND IF USED AS N'&SYSLIST(N)
COUNT	K'	GIVES THE NUMBER OF CHARACTERS IN A SYMBOLIC PARAMETER

7. PHONETIC INPUT DATA FORMAT

PHONEMES ARE IDENTIFIED BY THE ONE TO THREE CHARACTER CODE USED IN THE VOTRAX LITERATURE, AND ARE SEPARATED FROM THE INFLECTION BY ONE OR MORE COMMA'S OR BLANKS. THE INFLECTION IS INDICATED BY A SINGLE DIGIT - 1 FOR IN1, 2 FOR IN2, 3 FOR IN3, AND 4 FOR IN4. AN OMITTED INFLECTION IS ASSUMED TO BE 2.
EXAMPLE:

```
S, 1, AH1, IY, T, 3, R, UH3, 4, AH1, N, 3, IH, 1, K,
1, S, 1
```

IS ENCODED AS HEX

```
X'9F.D5.C9.2A.EB.64.D5.CD.85.99.9F'
```

7.1 VOTRAX PHONETIC CODES

PHONEME	HEX	EXAMPLES
PAO	03	
PA1	3E	
A	20	INITI(A)TED
A1	06	
A2	05	
AE	2E	H(A)T, (A)T, TR(A)CK; K(A)NSAS
AE1	2F	(A)LTITUDE, (A)CTUAL
AH	24	H(O)T, BL(O)CK, F(A)R, J(O)B, W(A)TCH
AH1	15	DEP(A)RTURE, F(O)XTR(O)T, UP(O)N
AH2	08	
AW	3D	(AW)FUL, (C(A)LL, C(O)ST, L(O)GGED, (O)FF
AW1	13	
AW2	30	
AY	21	
B	OE	(B)RAKES, OR(B)IT
CH	10	
D	IE	(D)AY, INITIATE(D), CO(D)ES
DT	04	BU(TT)ER
E	2C	K(EE)PER, SH(EE)T, EXC(EE)DS
E1	3C	K(I)LO, Z(E)RO, (E)MERGENCY
EH	3B	TH(E)RE, ST(EA)DY, S(E)NSE, QU(E)ST
EH1	02	M(E)SSAGE, L(E)VEL, QU(E)STION, D(E)DICATED
EH2	01	INTERFER(E)NCE, ID(E)NTIFY, MIN(U)S, (E)XECUTE
EH3	00	NEG(A)TIVE, DEDICAT(E)D, IDENT(I)FY
ER	3A	H(ER), OBS(ER)VE, B(IR)D, KEEP(ER)
F	ID	(F)IRE, INTER(F)ERENCE

G	1C	(G)ET, NE(G)ATIVE
H	1B	(H)AY, A(H)EAD
I	27	SH(I)P, WITH(I)N, (I)S, M(I)SSED
I1	OB	INTERF(E)RENCE
I2	OA	ALT(I)TUDE, (I)NTERFERENCE, W(I)THIN, (I)N(I)TIATED
I3	09	DED(I)CATED, EX(E)CUTE
IU	36	
J	1A	
K	19	(K)EY, SI(CK), (C)AR
L	18	(L)IGHT, (L)EVE(L), WE(LL)
M	OC	(M)Y, UNIFOR(M)
N	OD	(N)I(N)E
NG	14	BRI(NG)
O	26	F(O)R, UNIF(O)RM
O1	35	N(O)RMAL, KIL(O), H(O)LD
O2	34	
OO	17	F(OO)T, B(U)SH
OO1	16	NORM(A)L, ERR(O)R
P	25	(P)OT
R	2B	A(R)EA
S	1F	(S)EA
SH	11	(SH)Y
T	2A	(T)EA
TH	39	(TH)REE
THV	38	(TH)EN
U	28	ASS(U)ME
U1	37	EXEC(U)TE
UH	33	PL(U)S, (U)P, C(O)ME, T(OU)CH
UH1	32	(O)BSERVE, (U)NABLE
UH2	31	(A)CCOUNT, (U)PON, (A)CCOUNT
UH3	23	LEV(E)L
V	OF	SE(V)EN
W	2D	(W)ON
Y	29	MAR(Y), GALL(EY), D(E)PARTURE
Y1	22	(Y)ES
Z	12	(Z)ERO
ZH	07	A(Z)URE, MEA(S)URE

8. GRAIL CODING CONVENTIONS

FORMAT:

LABLE OP OPERANDS COMMENTS

OR

*COMMENTS CARD

THE 'LABEL' (IF PRESENT) *MUST* START IN CC1 AND MUST BE LESS THAN 7 CHARACTERS LONG.

'OP' STARTS IN ANY COLUMN BEYOND CC9, AND THE 'OP' OF ALL STATEMENTS BETWEEN DO AND DEND, DOINC OR DODEC AND DEND, THEN AND ELSE, ELSE AND IEND, CASE AND THE NEXT CASE AT THE SAME LEVEL, AND CASE AND THE NEXT CEND AT THE SAME LEVEL ARE INDENTED TWO COLUMNS WITH RESPECT TO THE ENCLOSING STATEMENTS.

'OPERANDS' ARE SEPARATED FROM THE 'OP' BY ONE BLANK, AND MAY CONTAIN IMBEDDED BLANKS *ONLY* IF THEY ARE ENCLOSED BETWEEN APOSTROPHES.

'COMMENTS' ARE SEPARATED FROM 'OPERANDS' BY AT LEAST ONE BLANK, AND SHOULD START IN CC36 WHENEVER POSSIBLE.

A 'COMMENT CARD' HAS AN ASTERISK IN CC1, NAD THE CONTENTS OF THE REST OF THE CARD APPEAR ON THE SOURCE LISTING, BUT ARE OTHERWISE IGNORED.

CONTINUATION CARDS -

IF THE 'OPERANDS' WILL NOT FIT ON A SINGLE CARD, THEN THEY MAY BE CONTINUED ON A SECOND CARD BY PUNCHING AN 'X' IN CC72 AND CONTINUING THE OPERANDS IN CC16 OF THE NEXT CARD.

A SINGLE OPERAND *MUST* BE CONTINUED WITHIN ONE CARD.

9. COMPILER ERROR MESSAGES

$BGN0001 -	THE DO , DOINC, DODEC , IF OR CASES STATEMENT OCCURS PRIOR TO ANY COURSE OR SECTION STATEMENT.
$BGN0002 -	THE NESTING LIMIT HAS BEEN EXCEEDED.
$CEV0001 -	EXECUTION VARIABLE NOT PERMITTED.
$CKL0001 -	REQUIRED LABEL OMITTED
$CKL0002 -	LABLE SPECIFIED, BUT NOT PERMITTED.
$CKL0003 -	LABEL LONGER THAN SIX CHARACTERS
$DGC0001 -	CHARACTER VALUE MISSING.
$DGC0002 -	CHARACTER VALUE.
$IFC0001 -	IMPROPER NESTING WITHIN AN 'IF' CONSTRUCT.
$IFC0002 -	IMPROPER ORDER OF 'ELSE' , 'THEN' AND/OR 'IEND' STATEMENTS ; OR MULTIPLE OCCURANCES OF SAME WITHIN A SINGLE CONSTRUCT.
$IFC0003 -	THE OPERAND OF A 'THEN' OR 'ELSE' STATEMENT IS NOT OMITTED OR 'NULL'.
$LNK0001 -	MORE THAN TWO OPERANDS.
$LNK0002 -	ARGUMENT SUBOPERAND IS NULL.
$LNK0003 -	INVALID FORMAT FOR ARGUMENT SUBOPERAND.
$TOK0001 -	EXPRESSION IS INVALID. ERROR DETECTED AT INDICATED POINT.
$VLR0001 -	INVALID SYNTAX OR VARIABLE TYPE IN ASSIGNMENT EXPRESSION.
$XND0001 -	THE MATCHING LABEL ON A CRSEND,CEND,DEND,IEND, FEND OR SEND DOES NOT MATCH THE LABEL ON THE STATEMENT BEGINNING THE CONSTRUCT.
$XND0002 -	THE WRONG TYPE OF 'END' STATEMENT HAS BEEN SPECIFIED- AN IEND FOR A DO CONSTRUCT , ETC.
$XND0003 -	AN 'END' HAS BEEN SPECIFIED OUTSIDE OF *ANY* ACTIVE CONSTRUCT.
CALC0001 -	TWO OPERANDS NOT PROVIDED.
CALC0002 -	TYPE OF VARIABLE AND EXPRESSION DIFFER.
CALC0003 -	EXECUTION VARIABLE USED AS RECEIVING VARIABLE.

EXTERNAL DOCUMENTATION

CASE0001 -	IMPROPER NESTING OF CASE STATEMENT.
CASE0002 -	MORE CASE STATEMENTS FOUND THAN EXPECTED.
CASE0003 -	LOGICAL EXPRESSION NOT SPECIFIED.
CASS0001 -	THE OPERAND OF THE 'CASES' STATEMENT IS OMITTED OR LESS THAN ONE.
CASS0002 -	CASES CONSTRUCT NESTING EXCEEDS 10 LEVELS; THE CURRENT CONSTRUCT WILL BE CHECKED FOR A PROPER CASE COUNT.
CASS0003 -	EXEC PARAMETER NOT 'ALL' OR 'FIRST'.
CASS0004 -	NUMBER OF CASE STATEMENTS EXCEEDS 99.
CDEF0001 -	INVALID FORMAT.
CDEF0002 -	TYPE OF VARIABLE CONFLICTS WITH SPECIFIED VALUE.
CEND0001 -	FEWER CASE STATEMENTS WERE FOUND THAN INDICATED ON THE CASES STATEMENT.
CEND0002 -	MORE CASE STATEMENTS WERE FOUND THAN INDICATED ON THE CASES STATEMENT.
DELY0002 -	OPERAND NOT INTEGER.
DEND0001 -	MORE THAN ONE OPERAND.
ERAS0001 -	OPERAND SUPPLIED, NONE ALLOWED.
ESCP0001 -	THE SPECIFIED LABEL IS NOT WITHIN THE CURRENT
EXEC0001 -	SECTION NAME OMITTED.
EXEC0002 -	MORE THAN TWO OPERANDS.
EXEC0004 -	INVALID EXPRESSION FOR ARGUMENT.
FCLL0001 -	FORTRAN SUBROUTINE NAME MORE THAN SIX CHARACTERS.
FCLL0002 -	FORTRAN SUBROUTINE NAME OMITTED.
FEND0001 -	OTHER THAN 'NO' OR 'YES' SPECIFIED FOR ERASE =.
FIND0001 -	WRONG NUMBER OF OPERANDS.
IFXX0001 -	CONDITIONAL CLAUSE MISSING.
LIMT0001 -	OTHER THAN ONE OPERAND SPECIFIED.
RDCC0001 -	MORE THAN THREE OPERANDS SPECIFIED.
RDCC0002 -	EXECUTION VARIABLE USED AS RECEIVING VARIABLE.
RDCC0003 -	SECOND OPERAND NOT AN INTEGER VARIABLE.
RDCC0004 -	THIRD OPERAND NOT A CHARACTER VARIABLE.

EXTERNAL DOCUMENTATION

READ0001 -	MORE THAN TWO OPERANDS FOR POSIT= PARAMETER.
READ0002 -	MORE THAN ONE OPERAND SPECIFIED.
READ0004 -	EXECUTION VARIABLE USED AS RECEIVING VARIABLE.
RECD0001 -	TYPE= PARAMETER OMITTED.
RECD0002 -	MORE THAN ONE OPERAND FOR TYPE= PARAMETER.
RECD0003 -	OPERAND MISSING.
RECD0004 -	OPERAND NOT CHARACTER STRING.
RECD0005 -	'TYPE' OPERAND NOT CHARACTER STRING.
SECT0001 -	MORE THAN 10 ARGUMENTS.
SHOW0001 -	NO OPERAND.
SHOW0002 -	MORE THAN ONE OPERAND.
TALK0001 -	UNKNOWN VOCAL-FORM.
TEXT0001 -	A VALUE OTHER THAN 'YES' OR 'NO' WAS SPECIFIED FOR THE CRLF= OPERAND.
TEXT0002 -	MORE THAN TWO OPERANDS ARE PRESENT FOR THE "POSIT=" OPERAND.
TEXT0003 -	FIRST POSIT= VALUE NOT INTEGER
TEXT0004 -	SECOND POSIT= VALUE NOT INTEGER
VDEF0001 -	UNKNOWN TYPE OF VARIABLE.
VDEF0002 -	VARIABLE NAME LONGER THAN SIX CHARACTERS.
VDEF0003 -	LENGTH SPECIFICATION MISSING FOR CHARACTER VARIABLES.
UTYP0001 -	THERE ARE NOT EXACTLY THREE OPERANDS.
VTYP0002 -	MORE THAN ONE VTYPE STATEMENT SUPPLIED.
VTYP0003 -	TYPE CHARACTER NOT C, E OR F.
VTYP0004 -	FORMAT OF AN OPERAND INCORRECT.
VTYP0005 -	END OF RANGE LESS THAN BEGINNING OF RANGE.
VTYP0006 -	MULTIPLE RANGES SPECIFIED FOR A SINGLE TYPE.
VTYP0007 -	RANGE CHARACTER NOT ALPHABETIC.
VTYP0008 -	RANGES OVERLAP.

DISTRIBUTION LIST

Assistant Secretary of the Navy (Manpower and Reserve Affairs)
Chief of Naval Operations (OP-96)
Chief of Naval Operations (OP-914)
Chief of Naval Operations (OP-964)
Chief of Naval Operations (OP-987P10)
Chief of Naval Operations (OP-103B)
Chief of Naval Personnel (Pers-10c)
Chief of Naval Education and Training (OOA)
Chief of Naval Education and Training (N-2)
Chief of Naval Education and Training (N-5)
Chief of Naval Education and Training (N-7)
Chief of Naval Technical Training
Chief of Naval Technical Training (015)
Chief of Naval Technical Training (016)
Chief of Naval Technical Training (N-3)
Chief of Naval Technical Training (N-4)
Chief of Naval Material (NMAT 0344)
Chief of Naval Material (NMAT 035)
Chief of Naval Education and Training Support
Chief of Naval Education and Training Support (N-21)
Chief of Naval Research (Code 450) (4)
Chief of Naval Research (Code 458) (2)
Commander Training Command, U. S. Pacific Fleet
Commander Training Command, U. S. Atlantic Fleet (Code N3A)
Commanding Officer, Fleet Combat Direction Systems Training
  Center, Pacific (Code 00E)
Commanding Officer, Fleet Training Center, San Diego
Commanding Officer, Naval Training Equipment Center
Commanding Officer, Naval Damage Control Training Center
Commanding Officer, Naval Aerospace Medical Institute
Commanding Officer, Naval Education and Training Program Development Center
Commanding Officer, Service School Command, San Diego
Commanding Officer, Naval Education and Training Support Center, Pacific
Commanding Officer, Naval Development and Training Center (Code 0120)
Officer in Charge, Naval Education and Training Information Systems
  Activity, Memphis Detachment
Director, Training Analysis and Evaluation Group (TAEG)
Superintendent, Naval Academy
Superintendent, Naval Postgraduate School
Superintendent, U. S. Military Academy
Superintendent, U. S. Air Force Academy
Superintendent, U. S. Coast Guard Academy
Assistant Director, Life Sciences, Air Force Office of Scientific Research
Army Research Institute for Behavioral Sciences
Personnel Research Division, Air Force Human Resources Laboratory (AFSC),
  Lackland Air Force Base
Occupational and Manpower Research Division, Air Force Human Resources
  Laboratory (AFSC), Lackland Air Force Base

Technical Training Division, Air Force Human Resources Laboratory,
  Lowry Air Force Base
Flying Training Division, Air Force Human Resources Laboratory,
  Williams Air Force Base
Advanced Systems Division, Air Force Human Resources Laboratory,
  Wright-Patterson Air Force Base
Secretary Treasurer, U. S. Naval Institute
Technical Library, Air Force Human Resources Laboratory,
  Lackland Air Force Base
National Research Council
National Science Foundation
Science and Technology Division, Library of Congress
Defense Documentation Center (12)