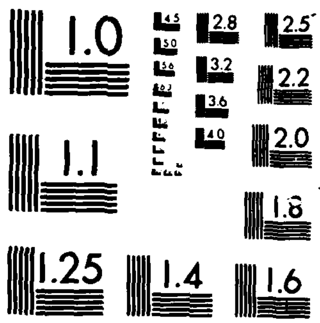
AD-A163 952 INTERACTIVE COMPUTER GRAPHICS FOR ANALYSIS AND DESIGN 1/3
OF CONTROL SYSTEMS(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED J R BULLARD DEC 85 AFIT/GE/EE/85D-5 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEPARTMENT OF THE AIR FORCE

AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

INTERACTIVE COMPUTER GRAPHICS FOR ANALYSIS AND DESIGN OF CONTROL SYSTEMS

THESIS

AFIT/GE/EE/85D-5 John R. Bullard
Capt. USAF

AFIT/GE/EE/85D-5

INTERACTIVE COMPUTER GRAPHICS FOR ANALYSIS AND DESIGN OF CONTROL SYSTEMS

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Accesion For	
NTIS CRA&I	☑
DTIC TAB	☐
Unannounced	☐
Justification	
By	
Distribution /	
Availability Codes	
Dist	Avail and / or Special
A-1	

John R. Bullard, B.S.E.E.

Captain, USAF

December 1985

Approved for public release; distribution unlimited.

Preface

This investigation is a continuation of the Interactive Control Engineering Computer Analysis Package (ICECAP) which began with Captain Glen T. Logan and has been continued by Major Charles J. Gembarowski in 1982, Captain Robert E. Wilson and Captain Mark A. Travis in 1983, Captain Abraham T. Armold and Lt Chiewcharn Narathong in 1984. The purpose of this package, which is hosted on a VAX 11/780 computer, is to aid the control engineer in the design and analysis of continous and discrete control systems.

I wish to thank Robert L. Ewing for this invaluable assistance on the use of the VAX 11/780 and on the use of GWCORE. I wish to thank Dr. Gary B. Lamont for his guidance and encouragement throughout this thesis effort. Also, I would like to thank Dr. Robert E. Fontana, Captain Gary C. Tarczyski and Captain Susan Mashiko for their suggestions and encouragement during the development and design of the interactive graphics design.

Finally, I would like to thank my wife, Linda, and family for their support and encouragement and for the many sacrifices they made during my graduate study.

Table of Contents

INTERACTIVE COMPUTER
GRAPHICS FOR ANALYSIS AND
DESIGN OF CONTROL SYSTEMS

THESIS

AFIT/GE/EE/85D-5 John R. Bullard
Capt. USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

LIST OF FIGURES

AFIT/GE/EE/85D-5

ABSTRACT

This thesis reports on the on-going effort to design and implement an Interactive Control Engineering Computer Analysis Package (ICECAP). When fully implemented this package will allow control engineers to design and analyze continuous and discrete control problems. This project was started by Captain Glen T. Logan's implementation of TOTAL. Captain Charles J. Gembarowski continued Logan's effort and began implementation of the "user friendly" command structure. Captain Mark Travis modified the graphical output and incorporated the use of a graphical package called GWCORE. Captain Robert E. Wilson implemented the help/teach modules and completed the continuous time functions started by Gembarowski. Captain Abraham Armold provided the discrete command structure so that discrete analysis and design could be performed by ICECAP.

The main emphasis of the thesis investigation was to implement an interactive graphical input routine which would complement the DEFINE commmand which exists in ICECAP.

Chapter I

INTRODUCTION

INTRODUCTION

(Interactive Control Engineering Computer Analysis Package).

The purpose of this thesis effort is to continue the work of improving and adding to the interactive graphics controls package called ICECAP. The main emphasis will be on providing a graphical way of entering the desired system's characteristics into ICECAP versus entering the system's open-looped transfer function (OLTF) through the keyboard. This will allow, for example the control engineer the ability to graphically place the desired system's poles and zeroes through the use of the cursor control keys of the keyboard. In addition, the option of displaying the locus solution, for the entered OLTF, will be available to the user, with the added capability to add and delete new poles and zeros without deleting the previous computed locus solution.

In control engineering, one of the engineer's goals is to control a system. A system is an interconnection of components forming a system configuration which will provide a desired system response (5). The system to be controlled for example may be chemical, electrical, or mechanical. But in order to accurately determine the resulting output of a control system, it requires that the control engineer model this system in the most accurate way possible.

Since the "dawning" of digital computers, the control engineer tried to harness the mathematical power of computers to model and design systems better. At first, computers were used mainly for "number crunching". As computer hardware became cheaper, newer and faster methods were developed which allowed the control engineer to aacomplish many functions within the same program. Recently, graphics have come into the design process. Graphics gives the designer more flexibility in displaying the open-loop transfer function and its solution and provides to the user a more accurate picture of desired system.

BACKGROUND

The Air Force Institue of Technology (AFIT), under the direction of Dr. Gary Lamont, has supported several efforts to develop an integrated computer-aided design tool for the analysis of control systems. The initial baseline was developed in a 1978, by Stanely J. Larimer (1), in which a computer program was developed called TOTAL (An Interactive Computer-Aided Design Program for Digital and Continuous System Analysis and Synthesis). Since the introduction of TOTAL, several thesis efforts have added and modified TOTAL, such as the provision to perform matrix operations and improved graphics.

TOTAL is an interactive controls package which is currently hosted on the CDC Cyber at Wright-Patterson AFB, Ohio (1). TOTAL was rehosted by Glen Logan to run on a VAX 11/780 (2). TOTAL prompts the user with an OPTION> statement which the user must respond with a particular option code, based on the desired action. However, the prompt does not describe nor inform the user which code should be entered and it became very tedious either to remember these codes or to have a user's guide next to the terminal in order to make use of the control package. In an effort to make the VAX version of TOTAL more friendlier, Charles Gembarowski wrote a Pascal umbrella package which provided the user with menus in which the available commands were shown to allow the user to step through the package without having to remember code option numbers or have a user's guide next to the terminal (2,3). After these and many other improvements, the VAX TOTAL was renamed ICECAP (Interactive Control Engineering Computer Analysis Package).

ICECAP is a menu driven TOTAL, which results in a friendlier human interface. It also has a help command to provide to the inexperienced user information about the commands provided in the menus. Once the user understands the commands and the command structure, the user can then abbreviate and combine commands to reduce time of working through the menus. ICECAP provides the user with a list of possible commands before each prompt. The user then selects the desired command. As each command is selected, it activates another command menu until all the information needed is provided. The user then has the choice of displaying the output on the screen or may send the data to a file for printing.

ICECAP has just recently been made available to the control engineering students. ICECAP is currently hosted on the VAX/VMS computer system. ICECAP is a menu driven program with all of the functions of the TOTAL controls packages plus many other improvements such as discrete system analysis and matrix operations (25,28).

The output from ICECAP is a graphical description of any system entered in by way of a system transfer function. Several graphical plots are available, such as the root locus plot, bode plot, and overall system response. A user may view on the screen or print to a local file the root locus, the system response to various inputs, a Bode diagram, showing both a phase and magnitude plot, and a description of the system's figures of merit. Initially all of the plots used text characters such as stars (*) and dashes (-). Mark Travis, in a follow-on effort, rewrote all of the screen graphics using a graphics package called GWCORE (4). By doing this, all of the screen displayed plots could be drawn using lines for the grids and plots. This provided to the user a clearer picture of the system's response. In addition, Captain Travis provided the capability to display four plots (root locus, system response, magnitude and phase Bode plots) on the same screen, simultaneously. This greatly enhanced the pictorial description of the entered system. This allows the user to see all of graphical outputs on the same screen and giving the user a more definitive description of the system characteristics.

PROBLEM STATEMENT

To enter the system description into TOTAL or ICECAP, the user must enter the computed transfer function either in polynominal or factored form. This requires the user to do advance computations prior to entering this into the controls package. The objective of this thesis is to modify the ICECAP program to allow the user to input the system characteristics through graphical means. Also, the iterative process of design and computation will be reduced by having the controls package compute and display several root locus designs to the user.

SCOPE

This investigation is primarily concerned with continuous time systems but since this thesis is primarily concerned with the graphical description of a system, this may be used in the discrete systems as well. The baseline for this thesis investigation is the result of Mark Travis's thesis (4), ICECAP-II, as well as the present version of ICECAP, titled just ICECAP. Also, since the graphic outputs are currently displayed on the Tektronicx 4014 terminal, this terminal will be used initially, until a driver is written which will allow the graphics to be displayed on a VT 240 terminal or the VT 125. GWCORE will be used to display and store the various computed root locus designs.

APPROACH

The first step will involve a complete familiarization of ICECAP so that all functions and routines are well known. Since the ICECAP program is hosted on the VAX/VMS, a knowledge of the VAX/VMS operating system is also required. Once this is completed, then various ways of interacting graphically with the VAX/VMS will be investigated. At present, the cursor controls for example may be used or a computer "mouse" may be incorporated. After familiarization, a set of requirements for the graphics module will be identified and agreed upon before the design stage starts. A human interface analysis will be conducted to determine the best mode of graphical entry is available and which grid would provide the most accurate input but also making it a pleasure to use.

During the design phase various screen displays will be examined to see which display would provide the "best" information to the user. Then ways of improving the display will be investigated to see how color will affect the display. A means of storing data will need to be investigated which will allow a comparison analysis to be done of two different systems, either graphically or through system characteristics, to determine if the user's requirements have been met. A scheme to store data is needed because the main program, ICECAP, only allows the storage of one transfer function at a time. In order to compare more than one transfer function on the same screen, the data must be stored and available for interactive use.

Once the manner in which the graphical inputs is chosen, then a driver program will be designed and coded which will test out various points and grids for accuracy. This main program will call several subroutines and will be used to get familiar with the various graphical routines. Once the graphics program is completed, the routines will be integrated into ICECAP.

Once integrated, the final step will be to test out the modified ICECAP package with various students and instructors, if possible. A series of test cases will be tested out to verify the various capabilities of the new graphics module. Throughout this thesis investigation, documentation will be maintained to provide a "living document" which will provide a stepping stone map of the steps taken to accomplish this thesis investigation.

THESIS OVERVIEW

Chapter two discusses in detail the requirements of the interactive graphics program. Specific tools will be examined including topics on the human interface, computer graphics, and software engineering methods for software design. Chapter three examines twelve guidelines relating to interactive design which should be considered. Chapter four lays out in detail the design of this thesis investigation. The interactive grid for the two design planes will be discussed along with an indepth discussion of all of the commands and the result of each.

Chapter five presents the results of testing. Chapter six concludes with discussion of specific accomplishments and recommendations for further efforts. There are eight appendixes which help to provide a "living document" for future investigations. Appendix A describes all of the main modules contained in the umbrella Pascal program. Appendix B is a comprehensive list describing all of the other fortran modules of ICECAP. Appendix C is the first flow chart of ICECAP. This appendix contains a complete flow chart of every command within ICECAP. The author wishes to thank Cynthia F. Marshall and Derrick Riddle for their time and efforts in making this structured chart. Appendix D is a "living document" of computer-aided control packages, listed in alphabetical order. Appendix E describes all of the commands of ICECAP along with the valid abbreviations. Appendix F provides data flow diagrams of ICECAP. Appendix G gives specific instructions on where the source code for the results of this investigation will be kept. Appendix H describes to future thesis students intructions on how to invoke and add new modules to ICECAP.

Chapter II

REQUIREMENTS DEFINITON

INTRODUCTION

Past efforts have modified, improved and added capabilities to the computer-aided design tool called TOTAL, and has evolved to what is known today as ICECAP. Initially TOTAL was used as a simple control analysis tool, but with the addition of math functions, discrete time analysis, and other functions, ICECAP has become an invaluable aid to control analysis and sythesis. But with all the modifications and additions, the input mode for TOTAL and ICECAP has remained the same, the keyboard. This effort will expand the input mode for ICECAP and allow the user to input system characteristics graphically as well as allow the user the options of displaying several root locus design on the same grid.

This chapter discusses some the tools that are considered in this effort. First the basic structure of a typical control system is described. Next a human interface section discusses some of the techniques used to convey information from machine to the user. Computer graphics is discussed next in which various physical devices are described. Next software engineering topics are discussed which describe different approaches to breaking a complex program into several simple routines which are linked into one main program.

Lastly, a description of the constraints and functional requirements are given which supports the identication of the main objectives of this investigation.

System Analysis

Typically, a control engineer sets up the control system similar to the block diagram shown in figure II-1 (29:146).

```
                 +-------+
-----&gt;O-----&gt;|  GTF  |--+----&gt;
      ^          +-------+  |
      |          +-------+  |
      +-------|  HTF  |--+
                 +-------+
```

where: GTF - is the forward tranfer function
HTF - is the feedback tranfer function
OLTF - is defined as GTF*HTF
CLTF - is defined as $\frac{GTF}{1+GTF*HTF}$

Figure II-1. Typical Control Block Diagram

There are two common methods of specifying transfer functions for input into ICECAP: ratio of polynominals and polynominal roots (factors). The control engineer (user) may specify the open-loop transfer function (OLTF), or the closed-loop transfer function (CLTF) directly into ICECAP, or may specify the forward transfer function (GTF) and the feedback transfer function (HTF) and have ICECAP combine the GTF and HTF to compute the required OLTF and CLTF. If the OLTF is entered into ICECAP, in factored form, the root locus can be plotted directly. But if the transfer function is entered in as a ratio of polynominals, then these polynominals must be factored to determine the factors needed for the root locus plot.

ICECAP can provide the following graphical outputs: Bode plot (magnitude and phase), time response plot, and a root locus plot. When ICECAP is run on a graphics terminal, e.g. tektronix 4014, the graphical output to the screen are line plots. When ICECAP is run on a non-graphical terminal, e.g. VT 100, the graphical plots are drawn with characters. Either terminal output is sufficient for the control engineer to do control engineering analysis, because tablular data can be provided which gives exact values at crucial points of interest.

Human Interface

The interface between the computer and the user is probably one of the most important areas when it comes to interactive graphics. If the interface isn't designed well, it could lead to degraded user productivity, frustration and confusion. It has been demonstrated that the human mind is visually oriented and information can be acquired significantly faster "by discovering graphical relationships in complex pictures than by reading text" (30:12). Text is a one-dimensional sequential string of information, while graphic pictures provide up to three dimensions (maybe more) of information which to a trained user can indicate the overall performance or crucial points of a systems response.

The human interface is composed of two languages: the language between the user and the computer and the language between the computer and the user (7). In order for the user to interact with the CAD package, the user must know what commands or actions are required and what the affect of the command or action will be. Normally, a computer-aided design (CAD) package will present a prompt to the user, indicating that the system is on and waiting for user inputs. The prompts given to the user can be as simple as a blinking cursor or as complicated as multi-dimensioned windows. Within ICECAP, menus are provided which display to the user specific command words which are valid for that level of interaction. This has two advantages: 1) the user does not need to remember specific commands or actions in order to accomplish a desired task, and 2) the novice as well as the experienced user can use the CAD package with very little training.

To reduce confusion and increase productivity, the keywords used in the menus must have some meaning to the user. In ICECAP, menus are provided at each level of the process until all the information is provided. For example, the following commands are used to input a forward transfer function:

DEFINE - this indicates the input mode

GTF - indicates that the forward transfer function will be provided

POLY - indicates that the transfer function will be provided in polynominal form

Within ICECAP help is available for the inexperienced user which explains the valid commands of ICECAP. This will be continued in this effort so that the user can effectively use and understand the various aspects of the graphics package. For the experienced user of ICECAP, commands may be catenated together, and/or abbreviated which helps to reduce interaction time and provide the desired activity sooner.

A means for error checking should also be included as part of a good CAD package so that the user is assured that the entered command is valid. This error checking should screen all commands and respond with an appropriate message if an error occurs while entering commands. ICECAP uses case statements to compare entered commands valid ICECAP commands. If the user enters a command that is not valid, ICECAP repeats the entered command back to the user along with a statement that the command is not a valid ICECAP word.

The language used by the computer to communicate with the user is in two modes: textual and graphical. The textual language must be clear and concise. It must be understood by the user in order to have any meaning. The menus of ICECAP are meant to present to the user a description of some desired action which the user may want to accomplish. Error messages must also convey some meaning to the user or the user will become frustrated. The graphics "language" must be in a from that is understood by the user. If the graphical picture is not displayed correctly, misunderstanding can occur. Graphical outputs of ICECAP include the Bode plot (phase and magnitude), time response, and root locus plots.

Computer Graphics

Computer graphics plays a great role in this effort. Previous efforts have designed graphical outputs, but this effort will be the first attempt to provide a graphical input. To simplify the task of providing graphical tools in which to draw graphs and plots, a general purpose graphics package will be used. This not only simplifies the designers tasks but also allows the program to be flexible and portable to other computer systems (12).

Since the physical world is larger than the computer graphics terminal, a conversion must be provided to either magnify or shrink the desired area so that it is of use to the user. This is known as windowing (12:53). The graphics screen may divided into several windows such as: a menu window, a command window or a graphics window. The menu window would consist of a number of functions which the user may select during the interactive graphics mode. The graphic window would serve as the central area of drawing/designing. The command window would be used to enter commands or particular parameters as needed (13). A message window may also be incorporated, which may be used to show the status of the system along with any error messages.

There are a number of ways that the user may graphically interact with a computer-aided design (CAD) package: mouse, joystick, cursor controls, trackball, lightpen. To have a control package which is flexible as well as portable, several modes of interactive input should be available for the user. The mouse, joystick, and trackball are all devices which when interfaced with a CAD package, indicate position and/or orientation.

A mouse is a hand-held device which moves across a flat surface moving two potentiometers. The output of these potentiometers are converted to digital values and are used to determine the direction and magnitude of the mouse's movement. Buttons are provided to select or de-select commands and/or information off the CAD screen (8).

The joystick and trackball are other input devices which are used to indicate position and/or orientation. While the joystick enables the user to indicate direction and speed, the trackball allows the user to specify the distance of the movement. The joystick has a greater speed in moving the cursor across the screen, but it suffers in accuracy. The trackball has a set of potentiometers which rotate as the ball is turned. The resolution is usually finer and the accuracy higher than that of the joystick (9).

The lightpen is a pointing input device. A lightpen is composed of a photocell and an amplifier which is able to distinguish light and dark screen areas. The light pen is great for selecting, but is very limited in its ability to draw (10).

In 1967, the Stanford Research Institute conducted an experiment with a mouse, a set of cursor controls, light pens and a joystick to determine which one was the easiest to use, most accurate and "the comparative ease with which the untrained user could become reasonably proficient using the various devices" (11). The results of their experiments showed that the mouse was both faster an more accurate than the joystick, light pen or the cursor controls for the experienced user. However, for the inexperienced user, the light pen was faster and more accurate. The joystick rated very low, partly due to the scale which was applied: 4:1 for a normal finger position of the stick verses 2:1 for the mouse potentiometers.

Software Engineering

Structured software design involves the process where a large complex system is broken down into independent modules which, at their lowest level, can be coded. Structured design also requires specific well-defined statements about the inputs and outputs of the system, as well as the data processing done within the system itself (13).

There are several ways to break up a large complex task into smaller more managable tasks. Three approaches are presented: top-down, bottom-up, and structured design. The top-down approach begins with the most general description of the solution and progresses to the most detailed, which can be coded into a subroutine or procedure, see figure II-2.

Top level	Most general solution
-	
-	
-	
n-intervening levels	n- depends on the problem complexity
-	
-	
-	
Bottom level	most detailed solution

Figure II-2. Top-down approach (14:18)

A top-down design approach imposes a hierachical structure upon the software program. There are several advantages to the top-down approach. First the problem starts with the most general and progresses to the specific. This ensures that all of the interfaces between the system are identified and tested early in the design phase (13:176). The top-down approach helps to ensure structured communication between modules and allows several modules to be designed, coded and tested independently of the overall system (15:45). For this thesis investigation, the top-down approach will be used.

The bottom-up approach, on the other hand, starts at the lowest program module and progresses to the highest. This approach produces a great deal of code in the early stages of the project development, but the progress is short lived. The greatest problem of the bottom-up approach is the integration and debugging. When a problem is discovered, it is very complicated to determine which module is at fault. Since the lower modules are coded first, drivers are required to test out the modules prior to integration (15).

One approach which has been developed by Stevens, Myers and Constantine, to structure the interrelationships between modules, is the structured designed approach. The structured design approach is composed of three basic tools: data flow diagrams, structured charts, and data dictionaries. Figure II-3 shows the basic elements of the data flow diagram.

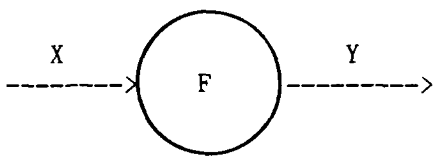


Figure II-3. Basic Elements of a Data Flow Diagram.

Referring to figure II-3, X is the data input to transformation block F, which transforms the data X into Y. The activity described in block F should represent the action on the input data X. The next step is the generation of the structured chart, which describes the relationships between different modules in a hierachical structure. The last step is the development of the data dictionary. The data dictionary describes the data and the activities associated with the system.

Constraints

Several constraints are involved in this effort. The hardware constraints are that the graphics input mode must be compatible with the current terminals available in the digital laboratory, e.g. VT 100, VT 125, VT 200, VT 250, Tektronix 4014, Evans/Sutherland. This range of terminals provides a number of ways to input graphical data, however, this effort will initially restrict the development on the VT-125. The VT-125 terminal does have cursor controls, however, at this time other modes of graphical entry have not been identified. Since this investigation will use color as much as possible, a color montior is needed to display and test out various designs.

There are several software constraints. The graphical library currently in use in ICECAP is the GWCORE graphics library written by the George Washington University (21). This library has recenty been upgraded to provide some of the input routines that the first release did not contain. With the use of GWCORE, several modes of input are available such as a joystick, lightpen, trackball, and cursor controls. Also, the graphical input will be used as a further addition to the capabilities of the ICECAP package. The numberical data, menus, command words, and error recovery must be compatible with the existing ICECAP program.

Summary of Requirements

A complete list of requirements for this effort is summarized in figure II-4. Each entry is prioritized. Those functions having a priority of one will be considered a minimum in order to provide an interactive graphics input to the ICECAP package. Those functions with a priority of two will be designed and implemented if time allows. The interactive graphics package will be a separate entity and will be accessed by a selection of GRAPHICS by the user, within the existing ICECAP program.

Interactive Graphics Package

1. Root Locus Input (1)
2. Incorporation of color (1)
3. Display of the root locus solution (1)
4. Provide the capability to expand, shrink, and zoom the displayed grid (2)
5. Provide user aids such as a unit circle and cross hairs designating major intersections (2)
6. Allow for the entry of text onto the displayed grid (2)
7. Allow the option of inserting and deleting poles and zeros into the design (2)
8. Provide on-line assistance for menu commands (2)

Figure II-4. Summary of Requirements

Summary

This chapter has defined and discussed in detail all of the requirements considered essential for a interactive computer graphics package. These areas include: System Analysis, Human Interface, Computer Graphics, Software Engineering, Constraints and Summary of Requirements. With these requirements, an interactive computer-aided design tool should be easy to learn, modify (for future efforts) and maintain.

Chapter III

SYSTEM DESIGN

INTRODUCTION

This chapter will discuss some of the issues which must be considered in the design of a interactive computer graphics package. First, twelve guidelines relating to interactive graphic design will be examined and specific decisions will be made on how they will be integrated into the system design. Specific requirements will then be discussed, along with draft designs of the root locus graphics.

HUMAN INTERFACE

In a graphical design, the communication link between the user and the computer is one of the most important consideration of the design. Without a good user interface, the user will become frustrated, bored or angry with the communication and ultimately cease all further contact with the computer program. The "designer" of any computer system, hardware of software, must consider several important areas. Steven Woffinden, in his master's thesis brought to light twelve general design principles that should be considered during the design phase. These twelve guidelines are presented in Figure III-1.

1. Determine the purpose of the system
2. Know the user
3. Identify the resources available
4. Identify the human factors of the design
5. Determine the interface language
6. Consider the operation language
7. Design for Evolution
8. Optimize Training
9. Accomodate different levels of experience
10. Compare input selections
11. Be consistent
12. Accomodate Errors

Figure III-1. General Design Principles (17)

Each of these guidelines will be addressed in the next few paragraphs.

1. Determine the purpose of the system

ICECAP's purpose is to help in the design and analysis of control systems. This initially started with program called TOTAL (1). Later, as more functions and capabilities were added it became apparent that the user friendliness needed to be improved. ICECAP was designed to meet this need by providing a friendly interface through the use of menus (25). This effort continues the goal of providing a friendly interface by providing a graphical means of entering the characteristics of system to be designed and analyzed. The goal is to simulate the process of the controls design (analysis and synthesis), but to have the computer provide the "book keeping" necessary so that the designer of a system need not be hampered with long or time-consuming calculations.

2. Know the User

The user of this system will be a person with a need to design or analyze a control system. This need can arise from a variety of sources, from academic school work to the most complex control system industry has to offer. Certain aspects of what is needed has already been incorporated into ICECAP's design. This includes the menu-driven displays, matrix manipulation, frequency- as well as time-domain analysis tools, along with continuous and discrete time analysis.

The user must always be kept in mind in any additions or modifications of ICECAP since the user is the ultimate reason for ICECAP's existence as other design guidelines will indicate.

3. Identify the Resources Available

The resources available for design are broken up into hardware and software. There is a definite relationship between computer hardware and the associated software that runs within it. But in order to make the software as portable as possible on different hardware systems, the software must be designed with portablility in mind. If this is not taken into account, then the system becomes device dependent and thus it restricts the usefulness of the software. For example, ICECAP-II was designed and implemented on the Tektronix 4014 graphics terminal. This ICECAP program was able to make use of the excellent graphics, but in doing so it became device dependent and was not able to be transported to other terminals. Only recently have drivers been written which allow the excellent graphics to be displayed on the other computer terminals.

Another aspect of the hardware/software interface which must be addressed is the availablility of input devices. To design the software to run only with a mouse or light pen may provide for a better human factors design, but if other input devices are not provided for then this too would restrict the portablility and versitility of the "system". To this end, this effort will make use of cursor control keys which are available on almost all terminals. Other input devices such as the mouse, joystick, light pen, etc, will be considered as time permits so that all of these input devices can be used.

With the hardware and hardware/software issues addressed, the final consideration is the software. Two areas need to be addressed: ICECAP and the graphic programs. ICECAP as it stands today is a very powerful tool. The user can input the systems characteristics either in polynominal, factored, or matrix form. Once entered, the system characteristics can be subjected to various inputs such as an impulse, a step, ramp or parabolic with user defined amplitude. It is menu-driven and provides tabular as well as graphical output. This software package is powerful, but at the same time complex. Care must exercised when modifications or additions are made to ICECAP so that the program does not lose any of its capablities or functions. The user friendliness will be continued in this effort and will be addressed further in guideline five, determine the interface language.

Another aspect to be considered is the availablility of graphic programs. Since this effort is primarily concerned with interactive graphic displays, the choice of a graphics package is extremely important. The choice will determine the versatility of the graphics displayed as well as the portabilility of the resultant computer program. In AFIT's Information Science Laboratory (ISL), there are four graphic packages which are available: Plot-10, GraLib, GWCORE, GKS. Plot-10 is a Tektronix commerical product developed for use on the Tektronix 4014 family of graphics terminals. (18) Since Plot-10 was designed for the Tektronix terminal, it is a not device independent graphics package, thus it will not be used in this effort.

GraLib is graphics package developed by the Lawrence Livermore Laboratory. (19) This package was designed to be a device independent graphics package but it was unsuccessful. Several of the routines were transferred by Philip Tarbell in 1981, but the conclusion reached by

Tarbell was that it be used for graphics research and development (20). Tarbell's recommendation was to use GWCORE.

GWCORE is a graphics package developed by George Washington University (21). It has been used successfully in at least two master's theses (4,32). GWCORE was developed on a VAX and can be used on various computer terminals as drivers are written.

GKS is a Digital Graphics package that has just become available within the ISL (31). The graphics calls are very similar to GWCORE, but the best attribute of GKS is the portability. GKS comes with drivers for the VT-125 monochrome, VT-125 color, VT-240 monochrome, VT-240 color, and the Tektronix 4014. This graphics package would have been used if it was available sooner. GWCORE will be used for this effort.

4. Identify the Human Factors of the Design

Henry Simpson, in his article "A Human-Factors Style Guide for Program Design" discussed six design principles that should be considered relating to human factors. These factors are shown in figure III-2 (22).

A. Provide Feedback

B. Be Consistent

C. Minimize Human Memory Demands

D. Keep the Program Simple

E. Match the program to the Operators Skill Level

F. Sustain Operator Orientation

Figure III-2. General Human-Factors (22)

A. Feedback

Feedback is necessary in all forms of communication. When humans communicate, their facial expressions along with body movements such as eye contact and nodding of the head, provide feedback to other humans which nonverbally communicate interest and understanding. Feedback from the computer on the other hand takes deliberate effort on the part of the designer. It must be "designed in ". ICECAP currently provides a very user friendly feedback, that of echoing back the user's input and responding with an appropriate message when the input is not consistent with command words that ICECAP "understands". This will be continued with the addition of the interactive graphics package. In the graphics part of ICECAP, the commands are echoed back and if the user enters a command which is not valid, the program will response with an error message. Also, as the user places the poles and zeros on the root locus grid, the location of the pole or zero entered in will be echoed back to the user to insure that the value is correct.

B. Consistency

Being consistent means that as the user progresses through the program, the menus and error messages are displayed in a certain place on the screen and that the commands used mean the same thing whenever they are used. For example, when entering in an open-loop transfer function (OLTF), the user would enter

DEFINE OLTF

At this point the user has the choice of entering it in polynominal or factored form. Selecting POLY to enter OLTF will respond in the same way as would defining a CLTF POLY, e.g. the response is consistent.

C. Minimize Human Memory Demands

Through extensive studies, it has been shown that the human brain can not be expected to accurately remember no more than seven values of a given parameter at one time (23). To help minimize the memory demands, menus have been extensively used in ICECAP to provide to the user commands which are valid, and hopefully have meaning to the user. Care must be exercised when selecting the appropriate word (command) to be shown on the menu. The word must communicate some desired action, otherwise confusion enters the mind of the user causing frustration. Help files should be provided, and are provided in ICECAP, to help explain the words listed in the command menu. This goes back to "knowing the user" and the command words and the explaination should have a common baseline with the user.

D. Keep the Program Simple

Keeping the program simple refers to the interface between the user and the computer program. The designer should build the software package in a way such that the use of the package is simple and easy to use. It should be as natural as possible to the way the user would accomplish the task without a computer. Keeping track of data should be kept to a minimum, as mentioned in the previous paragraph and the resultant output should be in a form that is understandable and useful to the user.

E. Match the Program to the Operators Skill Level

This facet relates back to knowing the user. The designer of any computer program must take into account the users skill level and accomodate for changes in the user's skill level. Initially, the novice user will need prompting for each input to the program and will rely on the help files a great deal. But once the user gains experience and knowledge of the computer program, the user may want to by-pass the various command menus. This feature has already been incorporated into ICECAP through the use of abbreviated command strings and the option to turn off the menu prompting. This feature will be continued in this effort, to the extent possible.

F. Sustain Operator Orientation

This last area of human factors refers to keeping the user aware of where he is in the program and allowing him the option of backing out. This is done by the use of the main command menu which acts as a "homebase" for the user. Whenever the user becomes disoriented in ICECAP, the user may enter a "$" which will immediately jump the user back to the main ICECAP menu.

These six human factor guidlines reduce to one central idea: know your user.

5. Determine the Interface Language

The interface language deals with the communication between the user and the computer program. ICECAP uses a series of menus which provide the user options from which to choose. At each level, the selection further defines the particular action until enough information is supplied such that the action can be accomplished.

Feedback is provided by echoing back to the screen the command chosen. If a command is chosen that is not part of ICECAP's vocabulary, an error message is provided. This will be continued in this effort and new commands entered will be added to the vocabulary of ICECAP.

6. Consider the Operation Environment

This topic deals with the location of the system and how the user fits into this environment. The goal of any system is to be accessable to the user as much as is needed and to be operational at all times. The goal of this system, ICECAP, is to be a student's aid in the design and analysis of control systems. To accomplish this goal, this program must be transportable, to the largest extent possible, to as many computer terminals as possible. Since GWCORE is being used as the graphics package, only the VT-125, VT-240 and Tektronix terminals can be used. VT-100 terminals must be modified with a graphics board in order to make use of the graphical displays.

7. Design for Evolution

This is but a start in the graphical interaction area. The design however, will allow for modifications and improvements. This will be done by documenting the design through structured charts and module descriptions, location in the appendix of this thesis. As each step is taken, documentation will be provided so that those who follow on in this area will be able to start with the tools developed here and progress without retracing the steps already taken.

8. Optimize Training

With the design of a new system, there arises a need to train the novice user as to the procedures for entry, use and finally output of the program. With some systems, a user's manual and/or on-the-job training (OJT) is necessary. Within ICECAP, on-line assistance is provided through the use of HELP command. The user can access information about the system and other commands as needed to accomplish a particular task. The design of ICECAP is to provide detailed menus to the user so that a user's manual is unnecessary. An alternative to this approach would be to provide a "demo" of the system, allowing the user to watch, or participate. In this way the user can watch as the system demonstrates the various commands and capabilities.

9. Accommodate Levels of Expertise

The goal of this system is to provide self-explainatory menus, which will guide the novice user through the system and capablities but at the same time provide a way for the experienced user to glide through to the necessary parts of the program in the least amount of time. This is accomplished already within ICECAP through two ways.

First, ICECAP allows the user to abbreviate commands. ICECAP interprets shorted these commands in the same way as the long way. The second way is that command strings can be strung together, thus avoiding the unnecessary step of displaying the intermediate menus. This allows the experienced user the ability to zero in to the particular area of interest.

10. Compare Input Selections

There are several ways to communicate with a computer system. One way is provide a prompt for the user, and the user is expected to know the correct command in order to accomplish a particular task. Within ICECAP, however, this communication has been accomplished through the use of menus. This has a couple of advantages. First all the valid commands are displayed to the user, which in turn helps alleviate the need for a user's manual. There are however, several types of menus: static, dynamic but visible, and dynamic but invisible. See figure III-3 for a complete description of each.

STATIC - user types in desired command through the use of the keyboard. Possible command options are provided.

DYNAMIC AND CONTINOUSLY VISIBLE - the user may select a command from the menu by moving the cursor over the desired command through the use of a graphic input device. Once the command has been selected, the command is highlighted.

DYNAMIC BUT INVISIBLE - the user selects a command from a menu as described above, but once the command is selected, the menu removed to allow more drawing area. With this design, the top line of the screen would be used to provide feedback as to the action underway. Also a symbol of some sort or a special character string would be needed to allow the user to bring back the menu and select another command.

Figure III-3. Menu Types and Description

Currently, ICECAP uses the static menu system. All possible commands are displayed to the user. The user enters the desired action by use of the keyboard and either the action is accomplished or another static menu is displayed. An alternative would be the dynamic visible or dynamic invisible menu. Eliminating the menus is of great value when the graphics become crowded by the menus being displayed. If the menus could be moved during the interactive graphics mode and returned when the user desired them, this would enhance the graphical entry as well as the display as a whole.

11. Be Consistent

Being consistent was covered under the human factors discussion. However, one final note regarding the interactive graphics. When plotting a root locus of an OLTF, symbols are used to indicate the roots of the transfer function, e.g. an "X" for a pole and an "O" for a zero. This has been incorporated into this graphics package.

12. Anticipate Errors

The designer must recognize that humans do make mistakes, either in entering the commands, the data, or in the evaluation of the system. The computer program must be able to recognize this error and be able to respond to the human user with an appropriate message that is clear enough so the user can understand and react accordingly. Error recovery is not always easy to do at times because the designer must be able to anticipate all errors that the user may enter and be able to recover gracefully, i.e. without "bombing" the system. For example, in TOTAL a carriage return, in response to the prompt, returned the user to the host computer system. This sometimes was very frustrating because some of the data entered would have to re-entered. ICECAP is a little more friendly, in that it provides an error message when the wrong command is entered in or if the data hasn't been entered in yet. This concept of error recovery will be used in this effort and will the user to recover due to an improper input.

SYSTEM DESIGN

This section outlines the design specifications for the interactive graphics routine. Also, an initial design will be presented, which will act as a possible design for this effort. The graphics routine will be a separate routine and will serve as an alternative to defining an Open-Looped Transfer Function (OLTF) through the use of the DEFINE command. Listed below are the design consideration for this thesis effort.

1. The graphics package will be user friendly. The commands will be simple but at the same time convey meaning as to the desired action. Abbreviations will be allowed and will be unique.

2. Feedback will be provided to the user wherever and whenever possible to keep the user informed as to the exact status and working level of the system.

3. Error messages will be used to inform the user of wrong or invalid commands.

4. This routine will be modular in structure and linked to the current version of ICECAP. Also, to the extent possible, this design will have minimal dependency on other routines.

5. The time required to draw or display text will be kept to a minimum.

6. Aids will be provided which will help the user in determining the exact location of the cursor. This is dependent on the functions of the graphics package, GWCORE.

INITIAL DESIGN

The initial design is the first attempt at describing the screen layout of this design. Referring to Figure III-4, the grid is layed out so that the grid uses the maximum amount of screen area. This has two advantages. First, this increases the size of the characters and of the root locus design. Secondly, the accuracy of the cursor is enhanced. This is done by a comparing the number of unit steps needed to move the cursor across the screen, and the number of oixels available on the screen. Since the number of pixels is set, then a way to increase the cursor value is to make the grid as big as possible. This will be feature will be available through the grid commands.

The screen is broken up into three major areas: design, menu, and command. The design area is place where the poles, zeros, text, and root locus will be displayed. The main parameter will be the grid which will be used to scale the poles, zeros and locus. The menu area is where the menus will be displayed. This area will display all of the valid commands and error messages. The command area is where the user will enter the desired command. The command will be echoed back at the top of the screen, so that the user will always know what function the graphics routine is performing.

SUMMARY

This chapter has discussed the guidelines which should be addressed in any man-machine interface. As each guideline was discussed, specific application was made to this design and ICECAP. A list of design requirements was described next, along with an initial design. Each area of the screen was identified, plus possible menu commands. The next chapter will discuss in detail the specific design implemented.

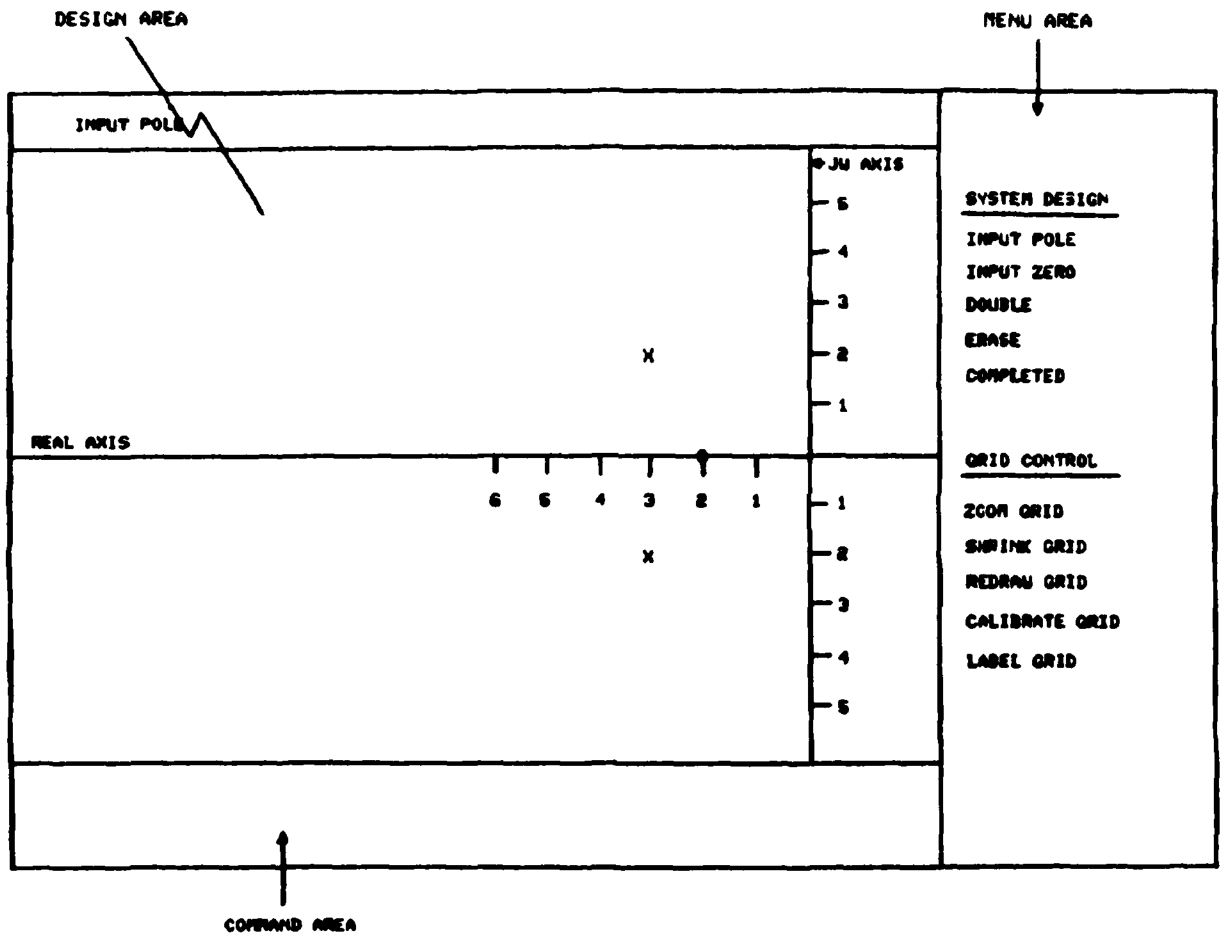


FIGURE III-4. INITIAL DESIGN

CHAPTER IV

DETAILED DESIGN AND SYSTEM IMPLEMENTATION

INTRODUCTION

This chapter discusses the details of the implemented design. The implementation strategy is discussed describing various methodolgies used in a software design along with a discussion of the the method chosen. Following this is a discussion of how this effort was transferred to the baseline, ICECAP. Then the details of this design are discussed, providing insights and rationale for the design decisions made throughout this project. Lastly is a discussion of the various commands available for the interactive graphics design.

DETAILED DESIGN STRATEGY

The selection of an development methodology has a significant impact on both the coding and testing of the final product. The most popular method is the top-down approach (26:340) where the designer is forced to consider the major functions of the system first and the less important ones last. Programmers, however, prefer the bottom-up approach, where the individual independent modules are written first and the higher level modules are written to accomodate the needs of the lower levels. In the top-down approach, the main program is written, coded, and tested first, with all of the sub-modules written as stubs. As each sub-module is written, it is integrated and tested with the other sub-modules, until the total program is completed. In the bottom-up approach, the lowest modules are written first with drivers written to "drive" the sub-modules. As each module is written, coded, and integrated, drivers are replaced until the total system is completed.

Another approach that is sometimes used is called the incremental approach. This strategy consists of three steps: 1) code and test a single module, 2) integrate another tested module to the first and test the combination, 3) repeat until the entire system is completed (27). With this approach if an error occurs, then debugging starts with the last added module.

In reality, it is often necessary to combine several strategies in order to solve a particular problem in a given enviornment. Although the top-down approach is being maintained throughout the ICECAP development, a complete top-down implementation is not practical since a large amount of ICECAP's modules have already been developed, tested and debugged. Thus, this effort will use the incremental approach. Each module developed is written and tested external to the main ICECAP program and integrated only when the modules are validated. In this way, fewer errors are expected to be included in the system.

ICECAP-II TRANSFER

The baseline for this effort, from now on called Graphical ICECAP, was ICECAP-II developed in part by Captain Mark Travis. ICECAP-II was designed to work on a Tektronix 4014 graphics terminal and to take advantage of the excellent graphics capability of the Tektronix. Taking advantage of the graphics capability, however prevented the program from being transportable to other graphic terminals due to the special terminal calls. In order to make Graphical ICECAP as portable as possible (as stated in the design requirements), ICECAP-II's special calls were modified to be capatible with the VT-125 terminal. This was done be utilizing the VT-125 driver of GWCORE. However, once transferred, two major problems were very apparent that were not problems with the Tektronix 4014.

The first problem involved the VT-125 driver itself. Within the VT-125 driver, minimum and maximum values for the screen are set. These maximum and minimum values correspond to the number of visible pixels available for the VT-125 terminal. However, the driver was designed to be imitate the Tektronix 4014 terminal as much as possible. In doing this, the minimum horizontal screen value (the left-most point where GWCORE allowed the designer to draw) was set to 144 instead of 0. This restricted the drawing capability of the graphics Graphical ICECAP. In order to make as much use of the screen as possible, this was changed to 0 (zero). This allowed for complete access to the total screen and allowed the displayed graphics to be larger and more readable. Once the minimum X value of the VT-125 driver was changed, the graphics within the baseline, ICECAP-II, no longer displayed the graphics correctly on the screen. Thus each graphical module developed by Mark Travis has to be revised to accomodate the change in screen coordinates.

The second major problem encountered with the transfer of ICECAP-II to the VT-125 terminal was the two modes of the VT-125 terminal. The Tektronix displays text and graphics without changing modes. However, the VT-125 terminal has two modes: text and graphics. It is possible to display text in the graphics mode, but reading text in graphics mode is very difficult. Thus to display text commands and input text commands involves switching from graphics to text mode within the program. This again involved modifying existing code and testing out those modifications prior to integration of the new interactive graphic modules. Both of these problems also had to be dealt with during the development of the interactive graphics modules. It was an easier task, however to accomodate these restrictions into the new design than to modify the existing code.

INTERACTIVE GRAPHICS OF ICECAP

The approach used to develop the interactive capability was done incrementally. Since documentation for GWCORE was not very understandable, the graphics calls were experimented with until the desired results were obtained. Initially, there was only one level of commands. However, as the design process went along, it became obvious that a second level was needed, due in part to the restricted menu area. Several grid designs were drawn, in which the space alloted was compared to the accuracy required. Once the grid and the command structure was finalized, several techniques were tried to place X's and O's on the screen. Too much time was taken in the drawing of circles for O's so the decision was made to use text O's. This proved to be a very good choice. After all of the graphics were completed, then this routine was integrated with ICECAP. The approach was to make the interactive graphics routine as modular as possible so that if changes were required in the graphics routine, only the graphics routine need be affected.

Incorporating interactive graphics into ICECAP was accomplished in two phases. The first phase involved the development of the interactive modules that would display a blank grid to the user, along with menu options, a cursor controlled marker, and the storage of user selected points in the appropriate variables for transfer to ICECAP. The second phase involved the integration of this interactive module with ICECAP, transferring the user defined characteristics of the system to the appropriate variables and allowing the user to display the root locus on the displayed grid.

PHASE 1

The grid chosen for this effort is a blank root locus grid. This was chosen because one of the plots used by the control engineers to determine the stability of a system is the root locus of the system. Two grids were designed, one for the s-plane and the other for the z-plane. The z-plane grid is shown in figure VI-1, the s-plane grid is shown in figure VI-2. In order to allow for maximum drawing area, the

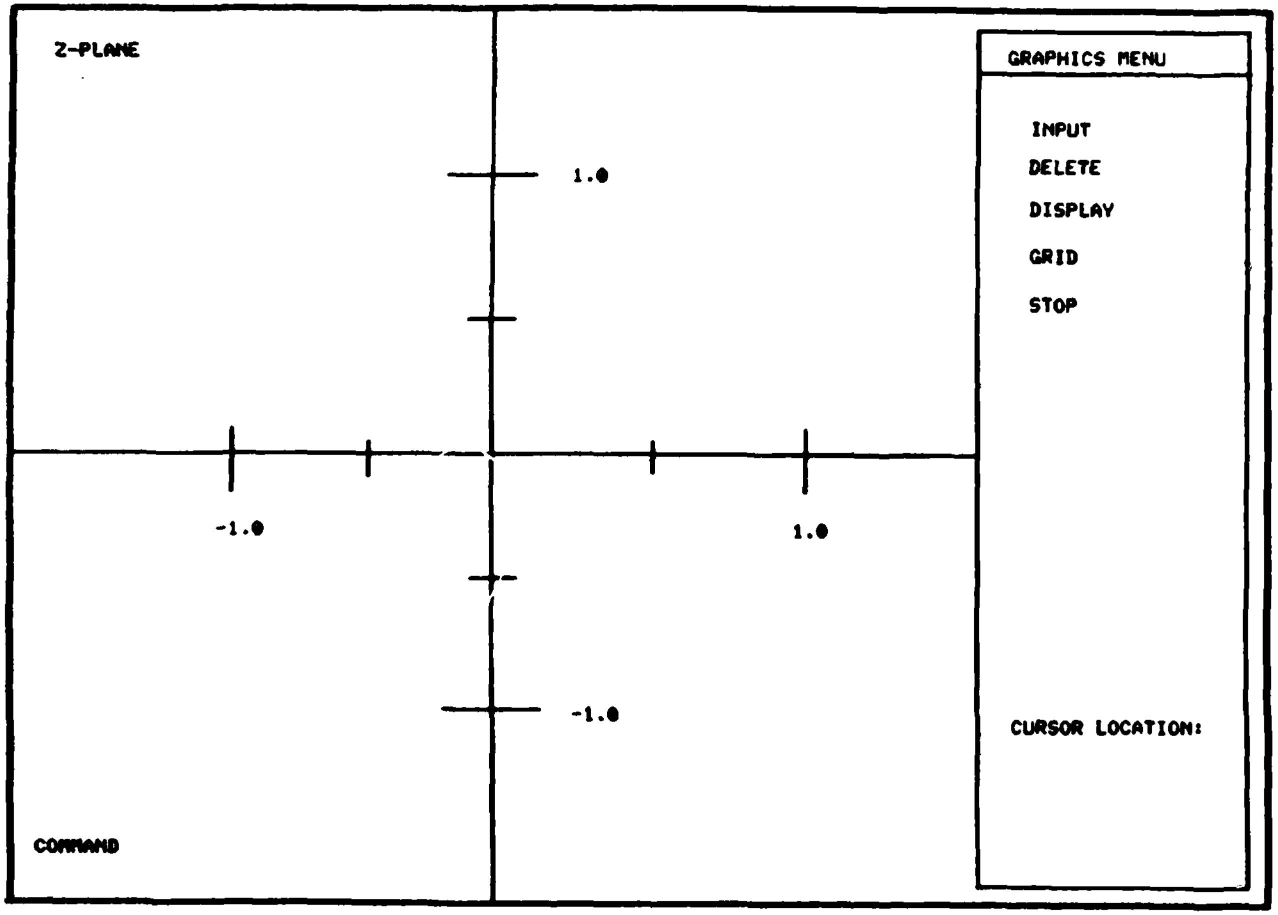


FIGURE VI-1. Z-PLANE GRID

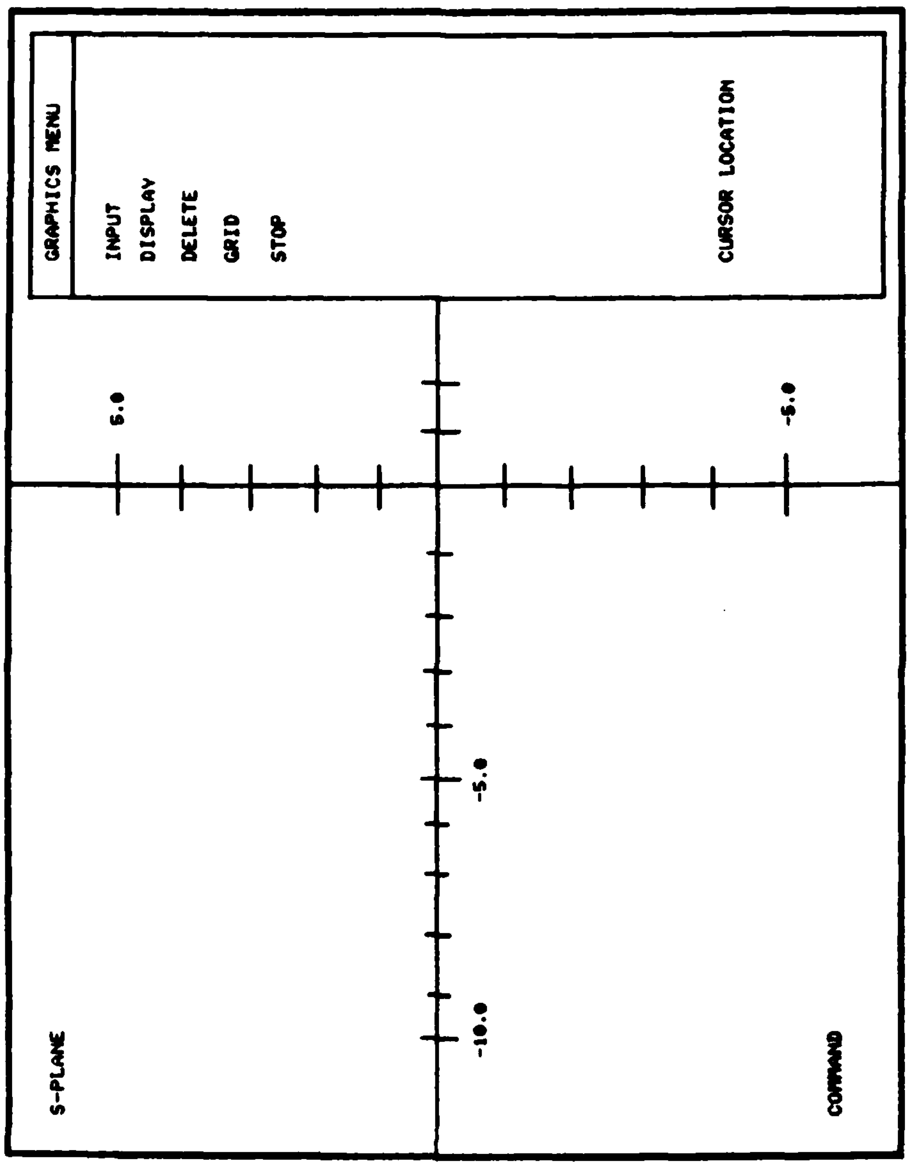


FIGURE VI-2. S-PLANE GRID

"X" axis divides the drawing screen in half horizontally, and the "Y" axis divides the drawing screen in half vertically for the z-plane grid and slightly to the right of center for the s-plane grid. The menus, message area, and cursor location feedback is located in a box on the right side of the screen. This was done so that the user only has to look in one area for all of the feedback and available commands. As specified by the design requirements, a command line is provided in the bottom left corner of the screen which echos back the command selected by the user. Also, the plane the user has selected is displayed in the upper left of the screen.

COLOR

Colors are an integral part of this design. The four colors available for drawing lines, text, or fill-ins are listed in figure VI-3, along with the level of intensity of white displayed on a monochrome screen. As shown in Figure VI-3, there is a difference in the monochrome for the various colors used. This difference in the shades of white were used to make important data stand out. For example, the brightest white was used to display the menu commands and the labels on the grid, and the lighter shades were used to provide aids for the user, such as the unit circle and cross marks.

COLOR	MONOCHROME
Green	White
Red	Greyish white
Blue	Grey
Black	Black

Figure VI-3. Color Comparison

Within GWCORE, there are only two ways to delete a line or a string of text from the screen, either by deleting the segment number given to the particular line or string of text, or by deleting every segment and re-drawing the lines and text. This proved to be very cumbersome, especially when trying to delete menus and X-Y cursor locations. Thus, colors were used to display the text and to erase the text. The menus and cursor coordinates are written in green and when the next level of menus is needed, or when the coordinates need to be erased, they are simply written again using the color black. To the user the menus and coordinates were erased from the screen. Within the program, the commands are executed and the coordinates stored.

TEXT

GWCORE specifies three ways to display text in graphics mode: string precision, character precision, and stroke precision (21).

String precision text is the fastest of the three modes, but there is little or no control over the size or orientation of the string of text. The display speed of string precision text on the VT-125 and the Tektronix 4014 are about the same. In the character precision mode, GWCORE writes each character separately, which means it takes a longer to draw up a string of text. But, the designer has more control over the orientation and spacing of the characters within the string text. The font types are limited, however to one. The slowest method of displaying text is the stroke precision. Each character is drawn on the graphics screen. Even though this is the slowest method, the characters can be oriented in any position necessary with any desired size, with any of four different font types. In order to display the information as quickly as possible, the string precision is used exclusively for displaying characters for the menus and for the labelling on the grids.

CURSOR

The only cursor marker available is a cursor shaped as a diamond, with a cross in the center. The cursor is controlled by the arrow buttons of the keyboard. Other input devices are permitted by GWCORE, such as a light pen, a pick device, etc. but these input routines are still not error free. The arrow buttons may be held down for continuous movement of the cursor across the screen or stroked independently for small movement. The cursor position is not available for display continuously by GWCORE, however, the grid position of the cursor is displayed once the user has designated by hitting the carriage return.

The location of the cursor is available within GWCORE as world coordinates. These coordinates correspond to minimum and maximum screen values initialized in the VT-125 driver. To designate a point on the grid, the user moves the cursor to the desired point on the grid and then depresses the return key. Once designated, a scaling is done on the world coordinates of the X-Y cursor location designated by the user. The accuracy of the designated point depends upon the grid size designed. The resolution of the grid design for this effort are: for the s-plane is 0.025 for each depression of the cursor keys and 0.005 for the z-plane. This resolution can be modified by re-scaling the grid through the use of the grid control commands.

COMMAND MENUS

The command menus are made up of two levels, as shown in Figure VI-4. Two levels are used due to restricted size of the menu area. The first menu consists of specfic verbs that the user wishes to accomplish such as INPUT, DELETE, DISPLAY, GRID, or STOP. The commands of both levels are listed according to the order of user by the user.

Level 1:	INPUT	DELETE	DISPLAY	GRID	STOP
Level 2:	Pole	Pole	Pole	Expand	
	Zero	Zero	Zero	Shrink	
	Text	Cross or Circle	Cross or Circle	Zoom	
	Exit	Exit	Locus	Exit	
			Exit		

Figure VI-4. Command Structure

INPUT

Poles, zeros and text are entered in by selecting INPUT in the first level. Selecting INPUT erases the top level menu and produces the second level menu. Poles and zeros are entered in by entering either POLE or ZERO. Once the selection has been made, a cursor appears at the origin of the grid. At this point, the user moves the cursor to the desired location with the arrow buttons. Once the cursor is placed at the desired position, the user designates it by depressing the carriage return. At this point, the cursor location is displayed to the user and a message appears informing the user to depress the carriage return when the user has read and understood the location of the pole or zero. If the user had selected POLE, a text "X" is displayed on the screen, at the cursor position. If the user had selected ZERO, a text "O" is displayed on the screen if a zero was selected on the input menu. If the pole or zero entered in is above the real axis, i.e. a complex one, the complex conjunate is automatically displayed on the grid and position is stored.

A time comparison was done on the drawing of an X and a circle on the grid and the placing of a text "X" and "O". The speed difference was so great that the text X and O are used for echoing the position of poles and zeros. Since the location of the text X and O are offset from the exact position required, an X-Y offset is used to place the center of the X and O at the exact location desired.

When TEXT is selected, a message appears requiring the user to move the cursor to the designated the location of where the text will be placed. Then the text information is entered in. Once the user has depressed the carriage return, the text is displayed on the screen at the designated location with a specfied character size.

DELETE

The DELETE command allows the user to delete an entered pole or zero, or to delete a displayed root locus graph. This allows the user the option of deleting undesired poles or zeros of a particular system that have been entered in by the user. The user may also delete the unit circle or cross (+) marks if so desired. This is done by drawing the by drawing the circle or crosses again using the color black.

DISPLAY

The DISPLAY command allows the user to display the entered poles or zeros in numberical form, to display the locus of the entered system, or to display the cross (+) marks (in the s-plane) or the unit circle (in the z-plane). Displaying the numberical values of the poles and zeros ensures that the entered values are the values intended by the user and also confirms to the user the values which will be transferred to ICECAP for the open-loop transfer function (OLTF). This meets the design requirement of feeding back to the user information about the entered data, and providing aids which help the user to locate and plot desired poles and zeros.

Displaying the root locus of the entered OLTF is the main objective of this investigation. Once the user has entered in the desired poles and zeros of the system, the root locus is displayed by entering in LOCUS when in the DISPLAY menu. If the user desires to modify the open-loop transfer function, OLTF, this can be done by exiting out of this menu and entering the INPUT mode again. Once the new OLTF is entered, the new root locus can be displayed by the same procedure described above. However, the new root locus plot is drawn in a different color, so that the user can distinguish the old root from the new. This design could be used as a teaching aid in graphically describing the effect of pole and zeros of a plotted root locus design.

Since the cursor position cannot be displayed continously, aids were design in to help the user identify major points on the root locus grid. A unit circle is available for the Z-plane grid and cross (+) markers are available for the S-plane grid.

GRID

The GRID commands are not included in this effort due to the lack of time and the complexity of the task. The intended design of the grid menu is to allow the user the option of expanding, shrinking the displayed grid or zooming in on a particular location. The expanding and shrinking of the grid would allow for more accurate input of poles and zeros plus allow the user to view the system locus by shrinking or expanding. By expanding the grid size, a different scale value can be used to designate the intervals on the grid. For example, the maximum negative value for the s-plane is -10.0. This corresponds to a scale factor of 0.025. By expanding the grid so that the maximum negative value is -1.0, the scale factor would be 0.0025, assuming that the grid remains the same. Zooming would allow the user to enlarge a specified area or examine a particular part of the displayed root locus of system.

STOP

The interactive secession is ended when the user enters STOP. Once entered, the numberical values of the poles and zeros, along with the computed polynominal are transferred into ICECAP through the use of common statements. The flag is set which indicates that a OLTF has been entered and is stored in the common variables. Once STOP is entered, control is transferred to the main menu of ICECAP.

Phase 2

This section describes the particular "hooks" with the main program called ICECAP. This graphics subroutine was designed so that there are minimal "hooks" to the main program, ICECAP. This graphical program was initially done as a stand alone module so that the design could be solidified and tested prior to integration with ICECAP. Once the design was completed, the main driver of ICECAP, ICER.PAS, was modified to allow access to the graphical subroutine. (The location of ICER.PAS file is described in appendix G.) The main menu was modified to display the word "Graphics", the dictionary was modified to allow for the entry of the word graphics, and a module was developed to call the graphics FORTRAN subroutine and to clear the screen once the graphics were completed.

Once control is transferred to the graphics subroutine, the user does not have any access to the rest of ICECAP. Control remains there until STOP is entered in by the user. The display and storage of the poles and zeros of the user defined system is within the graphics subroutine. If the user selects DISPLAY LOCUS, then the poles and zeros are transferred to ICECAP through the common statements and computation is done on the entered OLTF. Within ICECAP, the values for displaying the root locus are stored in a local files, opened and filled by the subroutines modified by ICECAP-II. Once the local file is filled, the graphics program translates to data and displays the root locus solution on the displayed grid. If the user selects to

STOP, then the OLTF is transferred, through the common statements and is available for computation to the rest of the ICECAP program. The OLTF transferred to ICECAP is the last OLTF defined by the user.

DOCUMENTATION

To continue the "living document" concept, all of the documentation for the interactive graphics routine as well as all of the modules of ICECAP are provided in the appendixes. Appendix A describes all of the ICER.PAS modules. Appendix B describes all of the FORTRAN subroutines. Appendix C contains a structured flow chart of this design effort plus every module within ICECAP. Appendix E contains all of the valid commands for ICECAP along with valid abbreviations. Appendix F contains all of the data flow diagrams for Graphical ICECAP. Appendix G describes the location of the source of this thesis effort plus the modified ICECAP modules.

SUMMARY

This chapter discussed in detail the implemented design and how the graphics package was integrated to the main program ICECAP. The commands for the menus was discussed along with a figure which shows the Z-plane as well as the S-plane grids. Throughout this design, the user has been kept in mind. The displays were kept simple, the drawing of the grids by the program were kept in mind so that minimal time was spent in drawing, and colors were used whenever possible to highlight significant areas of interest. This effort has demonstrated the feasability of interactive graphics. Follow-on efforts can build upon this effort and enhance and provide other interactive graphic areas within ICECAP.

CHAPTER V

TESTING

INTRODUCTION

This chapter discusses some of the various methods used in software testing. Top-down as well as bottom-up testing is examined, stating the advantages and disadvantages of each. Incremental testing is also discussed and how this approach was used to test out the various parts of this investigation. Finally, this chapter ends with a discussion of the test results obtained from testing out the end product of this investigation.

TESTING METHODS

When a software package is ready for testing, it is assumed to be completed, i.e. the process of debugging has ended. Debugging is the process of eliminating known errors, while testing is an attempt to show that more software bugs still exist (33). It is difficult, if not impossible to demonstrate that a program is completely error-free. Testing can verify that a certain subset of inputs produce a predictable set of outputs, but this in no way ensures that the program is error-free. Two test strategies will be discussed, top-down and bottom-up.

TOP-DOWN TESTING

Top-down testing is where the top most modules of the program are tested first, with stubs written for the routines not yet completed. The stubs usually are "dummy" subroutines which return either a set value or display a message indicating that a the routine was called and returned. As the development continues, each of the "stubs" are exchanged for the "real" subroutines. There are several advantages with this type of testing, specifically:

- the user can see a preliminary design of the program early in the development stage and can evaluate and provide feed-back to the designer as to whether the requirements of the program are what the user intended.

- serious design problems are detected early and interface incompatibilities can be resolved before major routines are written.

- debugging is accomplished easier, because any errors dis-covered after a new module has been added is narrowed down to the last added module. This helps diminish time necess-ary to track and locate the source of the error.

BOTTOM-UP TESTING

In bottom-up testing, the lowest level modules are tested first. The next higher-level modules are added next and are integrated into a system that is tested. The integration continues until all the modules have been added and integrated. In bottom-up testing, drivers are required which simulates that action of the higher modules. This is one of the major drawbacks of this testing method. The drivers themselves may involve as much design and testing as the module under test.

In practice, the combination of top-down and bottom-up are used in the testing of modules and programs throughout the program development. Since there are constraints on time, cost and resources, exhaustive testing is not possible. Therefore, it is necessary to design test cases that will detect the most amount of errors. Random-input testing is generally regarded as the poorest method of detecting the most errors, so an organized approach is used. The two approaches are known as functional testing and structural testing.

FUNCTIONAL TESTING

Functional testing involves testing the program as a "black box". Testing a program using the black box concept is an exhaustive input technique where the user tries to "break" the program by using every possible input condition as a test condition (33). Inputs are provided to the system and the outputs are observed and compared with the design requirement to see if the requirements have been met. This type of testing inherently takes the user's point of view (34). The user enters commands into the program and waits for some type of feedback. The program can accept the input string and accomplish the desired task, or the system can reject stream and produce an error message. If all of the erroneous inputs are trapped, i.e. only valid commands are allowed to process beyond the initial command stage, then the only commands left to be tested are the valid commands. At this stage, the valid commands are tested and the output from the system is compared against the user defined requirements. If the requirements are not met then the system will need to be modified or the system requirements will need to be modified to comform to the capabilities of the system.

STRUCTURAL TESTING

Structural testing on the other hand examines the implementation details, such as progamming style, control methods, source language, and data base design (34). The goal in structural testing is to test out every possible path through the system at least once to verify the correctness of the program. Structural testing, also known as "white box" testing, is based on the development of valid test cases which will exercise every path within the program at least once to insure that all of the "bugs" have been revealed and corrected. For a large program, the task of testing every path is countably infinite "because each loop multiplies the path count by the number of times through the loop. A small routine can have millions or billions of paths" (34).

GRAPHICAL ICECAP TESTING

The main objective of this investigation was to allow the user to define the open-loop transfer function (OLTF) through graphical means. The method chosen to test out this capability is incremental functional testing. This was chosen because of two major factors. First, exhaustive testing is not possible due to limited time and resources. Secondly, since the majority of ICECAP has already been written and tested as a package, it was simpler to code, test, debug a separate module, than to integrate every subroutine before testing each module independently. The testing of Graphical ICECAP is divided into two phases: graphic performance testing and general functional testing.

GRAPHIC PERFORMANCE TESTING

The primary purpose for graphics performace testing is to examine the graphical aspects of the design and identify errors with the displayed graphics. There are three segments to the graphics: the initial setup, the interactive input, and the displayed output.

INITIAL SETUP

As discussed in chapter IV, the implemented design, the user is asked, prior to any drawing, which plane he chooses to design in. After selecting either the s-plane or z-plane, the appropriate grid is displayed. The s-plane grid is layed out so that there are exactly 40 strokes of the cursor between the major divisions, i.e. between 0 and 1, 1 and 2, etc. This defines the lowest number and the lowest factor of numbers that can be entered in through the cursor, e.g. 0.025. This was tested and verified through multiple inputs to the system.

INTERACTIVE INPUT

The resolution of the cursor is very clear moving horizontally, meaning that there is only on point that corresponds horizontal cross-hairs of the cursor. However, moving vertically, there are two positions that are highlighted when the cursor is moved over a single point. This error has to do with the drawing capabilities of the graphics package, GWCORE. The thickness of the horizontal line is thinner than the thickness of a vertical line. This cannot be changed, but the top-most position of the line corresponds to the major divisions of the grid, i.e. to position the cursor at 0,j2, the cursor would be moved vertically until the vertical line of the cross hair is on the top most part of the j2 line.

Testing the outer-most boundary of the grid was also tested to verify that the values were valid and true. No errors were detected. The cursor moved freely to all parts of the displayed grid, and clipping is done to prevent the user from moving and designating points outside of the displayed screen.

DISPLAYED OUTPUT

There are two forms of displayed output, helpful aids and numerical data/locus plots. For each plane there is one aid for the user. In the s-plane, cross (+) marks are available to help the user locate major divisions of the grid. For the z-plane, an unit circle is available to help locate complex value that reside within the unit circle. Both aids were tested as to the accuracy of the displayed aid. Each aid was displayed correctly and in the correct position, so that if the user selected a point either on the cross marks or on the unit circle, the value stored was correct value.

The numerical data and the displayed locus are available to the user when selected. The numerical values of the entered poles and zeros are stored in a file called ROOTS.DAT and are displayed in the message area when selected by the user. The values are accurate to 7 decimal places, the numerical accuracy of REAL*4 variables. This is consistent with the main program, ICECAP, so it was deemed correct. The other displayed output is the display of the X's (for the poles), the O's (for the zeros), and the locus of points between these poles and zeros. Since the text characters X and O were used to echo the placement of the poles and zeros of the OLTF, a scaling was done prior to the display of the X's and O's so that the placement was correct, i.e. the center of the X and the O was at the point desired. The computation necessary for determining the locus of points of the entered OLTF is done by the main program ICECAP and the values are stored in a file called LOCPLT.DAT. Once the numerical values of the OLTF have been transferred to ICECAP and the LOCPLT.DAT file filled, then the file is read and the values scaled for display on the selected grid. Several root locus plots were done and compared to the graphical output of ICECAP-II and no differences found.

GENERAL FUNCTIONAL TESTING

The remainder of the testing involved the verification of the commands of the menus. Two levels of menus are available within Graphical ICECAP. Each of the commands were entered in to verify that the program recognized and the appropriate action was taken. Error trapping is done and when the user enters an invalid command, the program responds with: INVALID COMMAND, HIT <CR>. At this point, the user hits should hit the return key and the program awaits a valid command. There are however some inputs which will cause the program not to respond. That is when the user tries to move the cursor when a command from the keyboard is needed. As explain in chapter IV, the VT-125 has two operating modes: graphical and textural. Within the interactive part of the program, the modes are constantly be changed between the two modes.

SUMMARY

Software testing aimed at trying to find errors with the input or ouputs of modules is very important to the success of the program as well as providing the user with a useful tool. Functional or structural testing can be used, based on the objective of the program and the environment. Graphical tests were done and described in this chapter, which helped find and correct errors in the overall graphics package.

CHAPTER VI

CONCLUSIONS AND RECOMMENDATIONS

INTRODUCTION

This chapter discusses the conclusions and recommendations of the thesis investigation. Specific conclusions about the Graphical ICECAP are given, along with detailed recommendations concerning follow-on efforts and areas of improvement.

CONCLUSIONS

The successful design and implementation of this thesis effort demonstrates the capability of interactive graphics for the design and analysis of control systems. The product of this investigation now allows control engineers to design control systems similar to the way control courses are taught, i.e. by placing poles and zeros on a blank grid and computing the locus of points for the roots of the system. But the computation and drawing tasks are now given to the computer and the control engineer can now just concentrate on the results of the roots of the entered system versus taking the time to compute and draw each root locus design by hand.

This effort also demonstrated the use of color. This not only helps the user distinguish different segments of the screen but also highlights various root locus designs. Color was also in invaluable aid in the displaying and erasing of menus used throughout this project.

The "living" document concept enables ICECAP to be baselined. This also provides a self-contained document that describes all of the features and capabilities of the current version of ICECAP. This thesis effort endeavored to provide complete documentation of this thesis effort along with past descriptions so that all of the information that describes ICECAP is in one document.

RECOMMENDATIONS

The following recommendations are made for additional efforts and improvements to the Graphical ICECAP design.

1. Continuation of the Graphical ICECAP design

This thesis effort needs to continued. This thesis effort "broke the ice" so to speak on truly using interactive graphics within ICECAP. Further efforts could expand the interactive graphics to the system response, Bode plot, or Nyquist plot. Captain Mark Travis demonstrated the capability of drawing four graphs on one screen, thus allowing the user to see the system response, bode plots, and root locus all at the same time. This would be a great improvement to allow the user to interact with the root locus and be able to see the change to the other three graphs.

2. GKS Graphics Package

This thesis effort used the graphical standard called GWCORE, from George Washington University. The graphics package has great versitility in drawing and displaying two or three dimensional drawing. But the documention is extremely poor and not well maintained. Also, to date only two driver are available, the VT-125 and the Tektronix 4014. This severely resticts the portability of the final design. The AFIT Digital Laboratory recently received Digital's graphics package called GKS. It came too late to totally rewrite all of the modules developed under this effort, but new efforts should use the new standard in an effort to make Graphical ICECAP more portable.

3. Documentation of ICECAP

ICECAP has been designed and improved by many AFIT students. Throughout ICECAP history, however the documentation has not quite kept up with the improvements. Some students have tried to document previous efforts, but failed to thoroughly follow through in the source code. Some students only documented their specific improvements, which means time needed to spent researching several theses before the total picture of ICECAP can be achieved. When this thesis effort started, it was very difficult to determine not only where the source code was, but also which version was the correct version for the object code. Significant time was lost at the beginning of this effort due to poor documentation alone. It is recommended that future students document not only what they modify or improve but also clearly identify where the source and object code are located.

SUMMARY

This chapter has described several conclusions and recommendations for the improvement and modification of ICECAP. This thesis effort has been challenging and rewarding. Further efforts in interactive graphics will provide students as well as design engineers with a valuable tool that has not been available in the past.

APPENDIX A

ICER MODULE DESCRIPTIONS

A.1 Introduction

The purpose of this appendix is to describe the function of every procedure contained in ICER.PAS which is the Pascal program portion of ICECAP. Every VARIABLE in a Pascal program has an associated TYPE. The TYPE determines both the values that the VARIABLE can assume, and the operations that may be performed upon it. A list of all VARIABLE declarations used in ICER (the main driver program used in ICECAP) is presented in section A.2 of this appendix. The ICER module descriptions that appear in section A.3 make reference to these VARIABLE declarations.

A.2 Variable Declaration Used in ICECAP

This section contains a listing of all the VARIABLE declarations used in the program ICECAP. The format is of the form VARIABLE_NAM : TYPE. The format definitions of basic Pascal variable TYPEs can be found in any introductory book on the Pascal language. Three TYPEs declared in ICER are non-standard and are listed below:

- o BIGSTRING = PACKED ARRAY[1..COMMANDSIZE] OF CHAR
- o STRING = PACKED ARRAY[1..WORDSIZE] OF CHAR
- o BUFFER = ARRAY[1..BUFFERSIZE] OF STRING

Table A-1. Variable Declarations Used in ICECAP

```
ABORT : BOOLEAN
ALLMATCH : BOOLEAN
ANSWERFLAG : BOOLEAN
BUFFERPOINTER : INTEGER
CLC : INTEGER
COMMAND:ARRAY[1..COMMANDSIZE] OF CHAR
COMMANDBUFFER : BUFFER
COMMANDWORD : STRING
HEADER : BOOLEAN
I : INTEGER
ICOMMAND : BIGSTRING
J : INTEGER
LETTER : CHAR
LINE : BIGSTRING
MESSAGE : BIGSTRING
MESSAGEBUFFER : BUFFER
OPTIONNUMBER : INTEGER
OUTCOMMAND : BIGSTRING
OVERFLOW : BOOLEAN
PRINTFLAG : BOOLEAN
RESOLVED : BOOLEAN
STATUS : INTEGER
TCOMMANDWORD : STRING
TCOMMAND1 : STRING
TCOMMAND2 : STRING
TOTALFLAG : BOOLEAN
UCOMMAND : BIGSTRING
WHERE : STRING
WLC : INTEGER
WORD : ARRAY[1..WORDSIZE] OF CHAR
```

A.3 Module Descriptions

The following module descriptions serve to describe completely the function and structure of each module in ICER, the main driver program used in ICECAP. The reader should be aware that reference [30] presentes this information in a structured chart format. This investigation follows the module description set up by [61] which is repeated with the insertion of this effort's modules. A complete listing of all modules described is presented in Table A-2.

The modules are presented in alphabetical order. If a module is equipment dependent, the application is cited. The TYPE declaration associated with each input/output parameter is noted alongside the parameter. Table A-1 contains a listing of all the VARIABLE declarations used in ICER. To conserve space multiple descriptions may appear on a single page. However, module descriptions are not "split" between pages. Unless otherwise noted, all modules are written in VAX/VMS Pascal.

The symbol "var" is used to denote those calling parameters which are "passed by name" (i.e. The address of the variable is made known to the called procedure). The order of listing the parameters follows the order in which they must be cited in the calling sequence.

Table A-2. ICER Program Modules

BOXIT	HELP_TURN
CHANGE	HIGHLIGHT
CHANGE_PROMPT	ICER
CLEAR	INSERT
COPY	INSERT_RT
CURSORRC	INTERPRET
DEFINE	LOCUS
DEFINE_GAIN	LOCUS_AUTOSCALE
DEFINE_PROMPT	LOCUS_GRAPHICS
DEFINE_TF	LOCUS_MAGNIFY
DEFINE_TF_PLANE	LOCUS_SHRINK
DELETE	LOCUS_ZOOM
DELETE_RT	NOGRAPHICS
DICTIONARY	PACK_BUFFER
DISCRETE	PAUSE
DISPLAY_OR_PRIN	PAUSE1
DISPLAY_TF	PRINT_BUFFER
FIND_BORDERS	READCOM
FORM	STR$UPCASE
GRAPHICS	TFORM
HELP	TFORM_PROMPT
HELP_CHANGE	TFORM_TF
HELP_COPY	TFORM_TF_TYP
HELP_DEFINE	TITLE_SLIDE
HELP_DELETE	TRIM
HELP_DISPLAY	TURN
HELP_FORM	TURN_PROMPT
HELP_INITIAL	TURN_X
HELP_INSERT	
HELP_PRINT	
HELP_PROMPT	
HELP_SYSTEM	
HELP_TEACH	
HELP_TFORM	

****

PROCEDURE NAME : BOXIT

FUNCTION : Draws a box given the coordinates of the upper left hand corner of the box and the width (number of columns) and the depth (number of rows) of the box.

APPLICATION: VT100 Terminal

PROCEDURE(S) CALLED : CURSORRC
GRAPHICS
NOGRAPHICS

CALLING PARAMETER(S) : ROW : INTEGER
COL : INTEGER
WIDTH : INTEGER
HEIGHT : INTEGER

COMMENTS : None

****

PROCEDURE NAME : CHANGE

FUNCTION : Looks for legal object of the command word CHANGE. The CHANGE command provides the user the option to change the value of TSAMP, the numerator or denominator constant of a transfer function and the option to change planes.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : PLANCHG
SAMPCHG
CHGCONS
CHANGE_PROMPT
PAUSE
TRIM

CALLING PARAMETER(S) : var COMMANDBUFFER : BUFFER
var BUFFERPOINTER : INTEGER
var RESOLVED : BOOLEAN

COMMENTS : The modules named PLANCHG, SAMPCHG, and CHGCONS are written in FORTRAN and calls other FORTRAN modules.

****

****

PROCEDURE NAME : CHANGE_PROMPT

FUNCTION : Instructs the user on the use the command word CHANGE.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) :

COMMENTS : None

****

PROCEDURE NAME : CLEAR

FUNCTION : Clears the screen and places the cursor in the home position.

APPLICATION : VT100 Terminal

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE NAME : COPY

FUNCTION : Copies a Transfer Function to another Transfer Function.

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PACK_BUFFER
PAUSE
TOTICE

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****
PROCEDURE NAME : CURSORRC

FUNCTION : Places cursor at a certain position on the screen (ROW, COLUMN).

APPLICATION: VT100 Terminal

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : ROW : INTEGER
                        COL : INTEGER

COMMENTS : None

****
PROCEDURE NAME : DEFINE

FUNCTION : Looks for a legal object of the command word DEFINE and takes appropriate action.

PROCEDURE(S) CALLED : DEFINE_PROMPT
                       DEFINE_TF
                       PAUSE
                       TOTICE
                       TRIM

CALLING PARAMETER(S) :   VAR COMMANDBUFFER : BUFFER
                         VAR BUFFERPOINTER : INTEGER
                         VAR RESOLVED : BOOLEAN

COMMENTS : None
****
PROCEDURE NAME : DEFINE_GAIN

FUNCTION :   Allows the user to set the GAIN of the   system to any desired value.

PROCEDURE(S) CALLED : CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PACK_BUFFER
                       PAUSE
                       PRINT_BUFFER
                       TOTICE
                       TRIM

CALLLING PARAMETER(S) : None

COMMENTS : None
****

****

PROCEDURE NAME : DEFINE_PROMPT

FUNCTION : Provides information on the legal object of the command word DEFINE and waits for user response.

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN)

COMMENTS : None

****

PROCEDURE NAME : DEFINE_TF

FUNCTION : Looks for a legal object of the command string DEFINE [transfer function] and takes appropriate action.

PROCEDURE(S) CALLED : CLEAR
DEF_TF_PLANE
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER
TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

PROCEDURE NAME : DEF_TF_PLANE

FUNCTION : Processes the object of DEFINE (transfer function) (POLY or FACT) to determine which PLANE is desired.

PROCEDURE(S) CALLED : PAUSE
TOTICE

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

```
***********************************************************
PROCEDURE NAME :  DELETE

FUNCTION  :   Looks  for a legal object of the command  word
DELETE and takes appropriate action.

PROCEDURE(S) CALLED :  DELETE_RT
                       PAUSE
                       PRINT_BUFFER
                       TRIM

CALLING PARAMETER(S) :  VAR COMMANDBUFFER : BUFFER
                        VAR BUFFERPOINTER : INTEGER
                        VAR RESOLVED : BOOLEAN

COMMENTS  :   This is the beginning of the delete a pole  or
zero of a transfer function command string.
***********************************************************
PROCEDURE NAME :  DELETE_RT

FUNCTION :  Looks for a legal object of the command  string
DELETE   (transfer  function)  (POLE  or  ZERO)  and   takes
appropriate action.

PROCEDURE(S) CALLED :  DELETER
                       PAUSE
                       PRINT_BUFFER
                       TRIM

CALLING PARAMETER(S) :  VAR COMMANDBUFFER : BUFFER
                        VAR BUFFERPOINTER : INTEGER
                        VAR RESOLVED : BOOLEAN

COMMENTS  :   None
***********************************************************
PROCEDURE NAME :  DICTIONARY

FUNCTION : Checks each word in a command string against all
legal  ICECAP KEYWORDs.  It notifies the user of all illegal
command  words used in the command string.  If it finds  any
illegal command words it returns ALLMATCH = FALSE.

PROCEDURE(S) CALLED :  PAUSE
                       TRIM

CALLING PARAMETER(S) :  VAR ALLMATCH : BOOLEAN
                        VAR COMMANDBUFFER : BUFFER
                        BUFFERPOINTER : INTEGER

COMMENTS : None
***********************************************************
```

****
PROCEDURE NAME : DISCRETE

FUNCTION : Processes the objects of TFORM_TF_TYP to determine which method is used.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : BAKDIF, CTFGTF, CTFHTF, CTFOLT, GTFCTF, HIGHLIGHT, IMPUL, INVDIF, INVIMP, INVTUS, NOHIGHLIGHT, OLTCTF, STOW, STOWP, TUSTIN, WPTOS, WPTOZ, WTOS, WTOZ, ZTOW, ZTOWP

CALLING PARAMETER(S) : None

COMMENTS : With exception to HIGHLIGHT and NOHIGHLIGHT, the rest of the modules called are written in FORTRAN.

****

PROCEDURE NAME : DISPLAY_OR_PRIN

FUNCTION : Displays item on screen or prints item in ANSWER File.

PROCEDURE(S) CALLED : CLEAR, HIGHLIGHT, NOHIGHLIGHT, LOCUS, PACK_BUFFER, PAUSE, PRINT_BUFFER, TOTICE, TRIM

CALLING PARAMETER(S) : COMMANDBUFFER : BUFFER
BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMMENTS : None
****

****

PROCEDURE NAME : DISPLAY_TF

FUNCTION : Displays the roots of polynomials of a transfer function to greater decimal digit accuracy.

PROCEDURE(S) CALLED : DISPLY

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : DISPLY is a FORTRAN module.

****

PROCEDURE NAME : FIND_BORDERS

FUNCTION : Finds the four borders for the Root Locus plot (known in TOTAL as AA, BB, CC, DD) based on the poles and zeros of OLTF.

PROCEDURE(S) CALLED : None

CALLING PROCEDURE(S) : None

COMMENTS : This routine is written in VAX/VMS FORTRAN and is referenced by the filename FINDBORD.ICE.

****

PROCEDURE NAME : FORM

FUNCTION : Forms OLTF or CLTF depending upon user's choice.

PROCEDURE(S) CALLED : BOXIT
CLEAR
CURSORRC
GRAPHICS
HIGHLIGHT
NOGRAPHICS
NOHIGHLIGHT
PAUSE
TOTICE

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

****
PROCEDURE NAME : GRAPHICS

FUNCTION : Places the terminal in the graphics mode so that the Special Graphics Characters in Table 3-9 off the VT100 User Guide can be used.

APPLICATION: VT100 Terminal

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : None

COMMENTS : None

****
PROCEDURE NAME : HELP

FUNCATION : Looks for a legal object of the command word HELP and takes appropriate action.

PROCEDURE(S) CALLED : HELP_COPY
HELP_INITIAL
HELP_SYSTEM
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER
TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None
****
PROCEDURE NAME : HELP_CHANGE

FUNCTION : Explains how to use the CHANGE command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE

CALLING PARAMETER(S) : None

COMMENTS : None

****

```
*************************************************************
PROCEDURE NAME :  HELP_COPY

FUNCTION :  Explains how to use the COPY command.

APPLICATION:  VT100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PAUSE
                       PRINT_BUFFER

CALLING PARAMETER(S) :  None

COMMENTS :  None
*************************************************************
PROCEDURE NAME :  HELP_DEFINE

FUNCTION  :   Instructs the user in the use of  the  DEFINE
command.

APPLICATION :  VT-100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PAUSE
                       PAUSE1
                       PRINT_BUFFER

CALLING PARAMETER(S) :  None

COMMENTS :  None
*************************************************************
PROCEDURE NAME :  HELP_DELETE

FUNCTION  :   Explains the use of the  DELETE command.

APPLICATION :  VT-100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PAUSE
                       PRINT_BUFFER

CALLING PARAMETER(S) :  None

COMMENTS :  None
*************************************************************
```

****
PROCEDURE NAME : HELP_DISPLAY

FUNCTION : Explains the use of the DISPLAY command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None
****
PROCEDURE NAME : HELP_FORM

FUNCTION : Instructs the user in the use of the FORM command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR, HIGHLIGHT, NOHIGHLIGHT, PAUSE, PAUSE1, PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None
****
PROCEDURE NAME : HELP_INITIAL

FUNCTION : Displays all valid command words that can be used to start a command string. Valid abbreviations are shown in upper case.

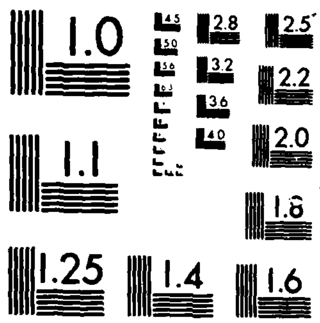
APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None
****

AD-A163 952 INTERACTIVE COMPUTER GRAPHICS FOR ANALYSIS AND DESIGN 2/3
OF CONTROL SYSTEMS(U) AIR FORCE INST OF TECH
WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING
UNCLASSIFIED J R BULLARD DEC 85 AFIT/GE/EE/85D-5 F/G 9/2 NL



MICROCOPY RESOLUTION TEST CHART

NATIONAL BUREAU OF STANDARDS 1963-A

****

PROCEDURE NAME : HELP_INSERT

FUNCTION : Explains the use of the INSERT command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE NAME : HELP_PRINT

FUNCTION : Instructs the user in the use of the PRINT command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE: HELP_PROMPT

FUNCTION : Displays all valid command words that can be used to start a command string. In this display the command words appear in all capital letters.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : None

COMMENTS : This display can be turned off by the user at any time by issuing the command TURN MAINMENU OFF.

****

```
***************************************************************
PROCEDURE NAME :  HELP_SYSTEM

FUNCTION  :  Displays  general information about the use  of
the ICECAP system program.  Also displays all valid  ICECAP
initial commmand words and their use.

APPLICATION :  VT-100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PAUSE
                       PRINT_BUFFER

CALLING PARAMETER(S) :  None

COMMENTS :  None
***************************************************************
PROCEDURE NAME :  HELP_TEACH

FUNCTION  :   Instructs  the user through a sample problem.

APPLICATION :  VT-100 Terminal

PROCEDURE(S) CALLED :  None

CALLING PARAMETER(S) :  None

COMMENTS :  None
***************************************************************
PROCEDURE NAME :  HELP_TFORM

FUNCTION  :  Instructs  the user in the use of the  discrete
transformation command, TFORM.

APPLICATION :  VT-100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       NOHIGHLIGHT
                       PAUSE
                       PRINT_BUFFER

CALLING PARAMETER(S) :  None

COMMENTS :  None

***************************************************************
```

****
PROCEDURE NAME : HELP_TURN

FUNCTION : Instructs the user in the use of the TURN command.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PAUSE
PRINT_BUFFER

CALLING PARAMETER(S) : None

COMMENTS : None
****
PROCEDURE NAME : HIGHLIGHT

FUNCTION : Puts user display into reverse video mode.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : None

COMMENTS : None
****
PROGRAM NAME : ICER

FUNCTION : This is the main driver routine for the program ICECAP (Interactive Control Engineering Computer Analysis Package).

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HELP_PROMPT
HIGHLIGHT
INTERPRET
NOHIGHLIGHT
READCOM
TITLE_SLIDE
TOTINI

CALLING PARAMETER(S) : None

COMMENTS : ICER is written in two languages-- VAX/VMS Pascal and VAX/VMS FORTRAN. The source code for ICER can be found under the filename ICER.PAS.
****

****
PROCEDURE NAME : INSERT

FUNCTION : Looks for a legal object of the command word INSERT and takes appropriate action.

PROCEDURE(S) CALLED : INSERT_RT
PAUSE
PRINT_BUFFER
TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : This is the beginning of the insert a pole or zero of transfer function command string.

****

PROCEDURE NAME : INSERT_RT

FUNCTION : Looks for a legal object of the command string INSERT (transfer function) (POLE or ZERO) and takes the appropriate action.

PROCEDURE(S) CALLED : ALTER
PAUSE
PRINT_BUFFER
TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

```
****************************************************************
PROCEDURE NAME :  INTERPRET

FUNCTION : Reads command words out of the command buffer and
calls appropriate procedures for action.

PROCEDURE(S) CALLED :  COPY
                       DEFINE
                       DICTIONARY
                       DISPLAY_OR_PRIN
                       FORM
                       HELP
                       PACK_BUFFER
                       PAUSE
                       READCOM
                       TOTICE
                       TRIM
                       TURM

CALLING PARAMETER(S) :  VAR COMMANDBUFFER : BUFFER
                        VAR BUFFERPOINTER : INTEGER
                        VAR RESOLVED : BOOLEAN

COMMENTS :  None

****************************************************************

PROCEDURE NAME :  LOCUS

FUNCTION  :  Displays and/or prints the Root Locus for  the
OLTF which must be already defined.

APPLICATION:  VT-100 Terminal

PROCEDURE(S) CALLED :  CLEAR
                       HIGHLIGHT
                       LOCUS_AUTOSCALE
                       LOCUS_MAGNIFY
                       LOCUS_SHRINK
                       LOCUS_ZOOM
                       NOHIGHLIGHT
                       PAUSE
                       TRIM

CALLING PARAMETER(S) :   VAR COMMANDBUFFER : BUFFER
                         BUFFERPOINTER : INTEGER
                         VAR RESOLVED : BOOLEAN

COMMENTS :  None
***********************************************************
```

****
PROCEDURE NAME : LOCUS_AUTOSCALE

FUNCTION : Displays and/or prints the Root Locus for the OLTF which must be already defined. Chooses the borders based on the locations of poles and zeros of the OLTF.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
FIND_BORDERS
TOTICE

CALLLING PARAMETER(S) : None

COMMENTS : None

****
PROCEDURE NAME: LOCUS_GRAPHICS

FUNCTION : This procedure calls the main driver of the interactive graphics package. After completing the graphics CLEAR is called, which clears the screen, and the terminal is returned to the text mode.

APPLICATION: VT-125 Graphics terminal

PROCEDURE(S) CALLLED : GRAPH
CLEAR

CALLING PARAMETERS : none

COMMENTS : none

****
PROCEDURE NAME : LOCUS_MAGNIFY

FUNCTION : Displays and/or prints the Root Locus for the OLTF which must be already defined. Doubles the size of the locus from the last time it was shown.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : MAGNIFY
TOTICE

CALLING PARAMETER(S) : None

COMMENTS : None

****

****
PROCEDURE NAME : LOCUS_SHRINK

FUNCTION : Displays and/or prints the Root Locus for the OLTF which must be already defined. Shrinks the size of the locus by a factor of two.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : MAGNIFY
TOTICE

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE NAME : LOCUS_ZOOM

FUNCTION : Displays and/or prints the Root Locus for the OLTF which must be already defined. User chooses the center point and the horizontal distance to the rightmost border.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : TOTICE
ZOOM_DATA

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE NAME : NOGRAPHICS

FUNCTION : Takes the user display out of the graphics mode. Restores the lowercase character set.

APPLICATION: VT-100 Terminal

PROCDURE(S) CALLED : None

CALLING PARAMETER(S) : None

COMMENTS : None

****

****

PROCEDURE NAME : NOHIGHLIGHT

FUNCTION : Puts user display into normal video.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : None

COMMENTS : None

****

PROCEDURE NAME : PACK_BUFFER

FUNCTION : Takes the contents of the command buffer and packs it into string OUTCOMMAND which is then sent to the module TOTICE.

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : COMMANDBUFFER : BUFFER
VAR OUTCOMMAND : BIGSTRING
COMMANDLEVEL : INTEGER

COMMENTS : None

****

PROCEDURE NAME : PAUSE

FUNCTION : Causes program to halt until (RETURN) is pressed, thereby allowing the user time to read the screen.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : None

COMMENTS : None

****

****
PROCEDURE NAME : PAUSE1

FUNCTION : Causes the program to halt until two (RETURN)s are entered which means continue or a $ and (RETEUN) are entered which means abort. Only used by Help/Teach modules.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : None

COMMENTS : None

****
PROCEDURE NAME : PRINT_BUFFER

FUNCTION : Prints out entire command buffer with no leading blanks, one trailing blank, no abbreviations, and with one space between words. Words are in upppercase.

PROCEDURE(S) CALLED : TRIM

CALLING PARAMETER(S) : COMMANDBUFFER: BUFFER
BUFFERPOINTER : INTEGER

COMMENTS : None

****

PROCEDURE NAME : READCOM

FUNCTION : Reads in ICOMMAND (until [CR]), changes it to uppercase (UCOMMAND), breaks it into command words, and puts the command words into COMMANDBUFFER.

PROCEDURE(S) CALLED : TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER

COMMENTS : None

****

****

PROCEDURE NAME : STR$UPCASE

FUNCTION : Puts all letters of the input string into uppercase.

APPLICATION : VAX/VMS Library Function

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : %STDESCR UCOMMAND : BIGSTRING
%STDESCR ICOMMAND : BIGSTRING

COMMENTS : The results of this function must be assigned to an INTEGER variable. ICECAP uses the integer variable STATUS for this purpose.

****

PROCEDURE NAME : TFORM

FUNCTION : TFORM is the beginning word of the discrete command string. This module looks for the second word or the command string, which is the name of the type of transfer function i.e. OLTF,CLTF,GTF, or HTF.

PROCEDURE(S) CALLED : PAUSE
TFORM_PROMPT
TFORM_TF
TRIM

CALLING PARAMETER(S) : var COMMANDBUFFER : BUFFER
var BUFFERPOINTER : INTEGER
var RESOLVED : BOOLEAN

COMMENTS : None

****
PROCEDURE NAME : TFORM_PROMPT

FUNCTION : Provides information on the legal object of the command word TFORM and waits for user response.

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : None

COMMENTS : None

****

****

PROCEDURE NAME : TFORM_TF

FUNCTION : This module looks for the third word of the command string TFORM_TF, which represents the "from " plane to the "to" plane.

PROCEDURE(S) CALLED : HIGHLIGHT
NOHIGHLIGHT
PAUSE
TFORM_TF_TYP
TRIM

CALLING PARAMETER(S) : var COMMANDBUFFER : BUFFER
var BUFFERPOINTER : INTEGER
var RESOLVED : BOOLEAN

COMMENTS : None

****

PROCEDURE NAME : TFORM_TF_TYP

FUNCTION : This module looks for the fourth word of the command string TFORM_TF_TYP, which is the name of a method to perform a transformation.

PROCEDURE(S) CALLED : CLEAR
DISCRETE
HIGHLIGHT
NOHIGHLIGHT
PAUSE
TRIM

CALLING PARAMETER(S) : var COMMANDBUFFER : BUFFER
var BUFFERPOINTER : INTEGER
var RESOLVED : BOOLEAN

COMMENTS : None

****

****
PROCEDURE NAME: TITLE_SLIDE

FUNCTION : Displays initial screen showing ICECAP in large letters and copyright information.

APPLICATION: VT-100 Terminal

PROCEDURE(S) CALLED : BOXIT
CLEAR
CURSORRC
GRAPHICS
HIGHLIGHT
NOGRAPHICS
NOHIGHLIGHT

CALLING PARAMETER(S) : None

COMMENTS : None

****
PROCEDURE NAME : TRIM

FUNCTION : Trims the trailing blanks off of the SOURCE string and places the stripped version into the DESTINATION string.

PROCEDURE(S) CALLED : None

CALLING PARAMETER(S) : VAR SOURCE : STRING
VAR DESTINATION : STRING

COMMENTS : None

****
PROCEDURE NAME : TURN

FUNCTION : Used to turn the various control switches ON and OFF.

PROCEDURE(S) CALLED : PAUSE
TRIM
TURN_PROMPT
TURN_X

CALLING PARAMETERS : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

****

PROCEDURE NAME : TURN_PROMPT

FUNCTION : In the absence of an object for TURN, it prompts for one.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN

COMMENTS : None

****

PROCEDURE NAME : TURN_X

FUNCTION : Processes the object of TURN by looking for OFF or ON and sets the switch accordingly.

APPLICATION : VT-100 Terminal

PROCEDURE(S) CALLED : CLEAR
HIGHLIGHT
NOHIGHLIGHT
PACK_BUFFER
PAUSE
TOTICE
TRIM

CALLING PARAMETER(S) : VAR COMMANDBUFFER : BUFFER
VAR BUFFERPOINTER : INTEGER
VAR RESOLVED : BOOLEAN
VAR HEADER : BOOLEAN

COMMENTS : None

****

A.4 Summary

The module descriptions for the ICECAP modules (main program and subprograms) are listed in this appendix in alphabetical order for ease of reference. As new modules are added to the program they can be easily documented in this appendix by including them in alphebetical order and by following the established standard format.

APPENDIX B

FORTRAN Module Descriptions

B.1 Introduction

This appendix gives descriptions of the FORTRAN modules used in ICECAP and descriptions of the Graphical FORTRAN modules that were written to permit the graphics to be interactive. A complete source listing of these modules is maintained in the AFIT Digital Equipment Laboratory.

B.2 Description of New FORTRAN Modules

The following FORTRAN modules annotated without an asterisk were previously developed as a result of (2,4, 24, 26) . Those modules with an asterisk were developed and implemented by this thesis investigation. These modules are coded in FORTRAN as they are more closely related to the FORTRAN portion of ICECAP than they are to the Pascal portion. TOTICE is the interface module between the Pascal portion of ICECAP and the FORTRAN modules. Since most of its code was derived from VAXTOTAL's mainline FORTRAN program, the decision was made [4] to code TOTICE in FORTRAN.

B.3 Description of all the FORTRAN Subroutines used in Graphical ICECAP.

MODULE NAME: ADD

DESCRIPTION: Adds polynomials C1 and C2 together and places the sum in array C3. The polynomials are first placed sequentially in array C3 and then SIMPLE is called to add coefficients of like powers of S. The dimension of array C3 must equal the dimensions of C1 and C2.

CALLING SEQUENCE: CALL ADD(C1, NT1, C2, NT2, C3, NT3, M)

- C1 - A double precision array containing polynomial coefficients and corresponding powers S in the format specified by M.
- NT1 - Number of occupied elements in array C1.
- C2 - A double precision array containing a second polynomial in the format specified by M.
- NT2 - Number of occupied elements in array C2.
- C3 - A double precision array containing a polynomial which is the sum of the polynomials C1 and C2.
- NT3 - Number of occupied elements in array C2
- M - An integer constant specifying which the format of the polynomial coefficients.

CALLS: SIMPLE

MODULE NAME: ADAPT

DESCRIPTION: Interfaces routine READS with the root locus programs. ADAPT asks the user for all information for the specific option number, prints out prompts, calls READS to receive user input, and stores input in the data base.

CALLS: READS

MODULE NAME: ADVANZ

DESCRIPTION: Computes the Z-transform from the LaPlace Transform of a transfer function for sample-data analyses.

CALLS: READS, PARTFR, WPLN

MODULE NAME: ALPHA

DESCRIPTION: Contains a table of 84 keys, including the numbers 0-50 for order of polynomial, polynomial names, and polynomial calculator operations.

CALLS: none

MODULE NAME: ALTER

DESCRIPTION: This module determines which part of a transfer function is going to have a root inserted, i.e. a pole or zero.

CALLS: READS, INSERT

MODULE NAME: ANG1

DESCRIPTION: Computes magnitude and phase angle.

CALLS: BANG, BOX

MODULE NAME: ARROW

DESCRIPTION: This module uses the GWCORE to draw arrowheads for the block diagrams.

CALLS: LINA2, MOVR2

MODULE NAME: AW

DESCRIPTION: Computes the phase angle of the discrete or continuous open or closed-loop transfer function frequency response.

CALLS: none

MODULE NAME: AZCALC

DESCRIPTION: Processes letters A through Z in calculator mode.

CALLS: CALVAR, CALKEY, CALABB

MODULE NAME: AZCOMP

DESCRIPTION: Processes letters A through Z in the compiler mode.

CALLS: COMCOM, COMVAR, COMABB

MODULE NAME: BAKDIF

DESCRIPTION: This module is the main module called by ICECAP to begin calculation on the backward difference equation.

CALLS: CLEAR1, READS, BIFORM

MODULE NAME: BANG

DESCRIPTION: Computes angle of a vector from its rectangular components (A,B) by finding ARCTAN(B/A).

CALLS: none

MODULE NAME: BIFORM

DESCRIPTION: Substitutes S = ALPHA * (Z + SIGMA1) / (Z + SIGMA2) into CLTF(S) or Z = ALPHA * (S +SIGMA1) / (S + SIGMA2) into CLTF(Z).

CALLS: BITERM, FACTO

MODULE NAME: BIFORM2

DESCRIPTION: This module is the same as BIFORM except that it makes the following substitution:

Z = ALPHA * 1/(S + SIGMA2)

This module is used to perform the inverse backward difference algorithm.

CALLS: BITERM1, FACTO

MODULE NAME: BILIN

DESCRIPTION: Performs the general bilinear transformation from the Z to the W or W' planes or vice versa.

CALLS: TERM, ROTPOLY, SZWROOT

MODULE NAME: BITERM

DESCRIPTION: Returns POLYQ = (ALPHA**I) * ((Z + SIGMA1)**I) * ((Z + SIGMA2)**(NP - I)).

CALLS: EXPAND, DBLMULT

MODULE NAME: BITERM1

DESCRIPTION: This module returns the following:

PLOYQ = [(Z + SIGMA2)**J](1/[ALPHA**J])

It is used only for the inverse backward difference algorithm.

CALLS: EXPAND

MODULE NAME: BLEND

DESCRIPTION: Orders roots prior to plotting root locus.

CALLS: none

MODULE NAME: BLOCKER

DESCRIPTION: Performs block diagram manipulations using polynomial routines.

CALLS: POLYMLT, FACTO, CANCEL, POLYADD, POLYSUB

** MODULE NAME: BORDER

DESCRIPTION: This module draws the border for the graphics package.

CALLS: STNDX, SLNDX, SCHSIZ, CRRSEG, MOVA2, PLINA2, TEXT, CLRSEG

MODULE NAME: BOX

DESCRIPTION: Reduces an angle to its primary value between +/- PI, i.e., removes multiples of PI.

CALLS: none

MODULE NAME: BOXER

DESCRIPTION: Reduces an angle to its primary value between +/- PI.

CALLS: none

MODULE NAME: BREAK

DESCRIPTION: "Zeros in" on breakpoints for root locus.

CALLS: none

MODULE NAME: CADJB

DESCRIPTION: Computes the polynomial G(S) from the matrix product (C)T(adj[SI-A])B, where the adjoint matrix is supplied as ADJ(10,10,10).

CALLS: none

MODULE NAME: CALABB

DESCRIPTION: Processes abbreviation in calculator mode.

CALLS: none

MODULE NAME: CALCK

DESCRIPTION: Processes "+", "-", "*", or "/" found in calculator mode.

CALLS: CALCTR, CALNUM

MODULE NAME: CALCTR

DESCRIPTION: Simulates HP-45 calculator.

CALLS: RDRNUM, FORMT, STORE, RECALL

MODULE NAME: CALKEY

DESCRIPTION: Processes a calculator key found in the calculator mode.

CALLS: CALCTR

MODULE NAME: CALNUM

DESCRIPTION: Processes a number in the calculator mode.

CALLS: RDRNUM

MODULE NAME: CALVAR

DESCRIPTION: Processes a variable in the calculator mode.

CALLS: LISTER

MODULE NAME: CANCEL

DESCRIPTION: Cancels pole/zero pair within a specified tolerance.

CALLS: EXPAND

MODULE NAME: CANROOT

DESCRIPTION: Cancels equal zeros and poles according to a specified tolerance. Separate tolerances are provided for the real part of each root and for the imaginary part of each root.

CALLS: none

MODULE NAME: CDEXP

DESCRIPTION: Calculates the exponential function of a complex number in double precision.

CALLS: none

MODULE NAME: CHALK

DESCRIPTION: Performs a chalk pitch axis H.Q. (?) criterion analysis.

CALLS: READS, AW, FW

MODULE NAME: CHGCONS

DESCRIPTION: This module is used to modify the numerator or denominator constant (gain) of a transfer function.

CALLS: READS, PARTS, EXPAND

MODULE NAME: CIRCLE

DESCRIPTION: Uses the GWCORE to draw a circle given the coordinates of the center and radius.(Developed by Kevin Rose)

CALLS: LINA2, MOVA2, STEPLN

MODULE NAME: CLAUSE

FILE NAME: CLAUSE.FOR

DESCRIPTION: Clause serves as a recursive routine. It handles FOR, NEXT, IF..THEN..ELSE..END, and WHILE features.

CALLS: GETSYM, ERROR, PUTID, WPOLY, STACKP, EXPR, PARSE

MODULE NAME: CMULT

FILE NAME: QCMULT.FOR

DESCRIPTION: CMULT multiplies two complex numbers.

CALLS: none

MODULE NAME: CNTER

DESCRIPTION: Computes the number of characters in the integer part of CBZ and returns the answer in CBZ.

CALLS: none

MODULE NAME: COEFF

DESCRIPTION: Adds the missing power terms in a polynomial by inserting a zero coefficient with the appropriate power and moving the original terms to make room for this missing term.

CALLS: SIMPLE, ORDER3

MODULE NAME: COMABB

DESCRIPTION: Processes abbreviations in the compiler mode.

CALLS: none

MODULE NAME: COMAND

FILE NAME: COMAND.FOR

DESCRIPTION: Sets up the MATLAB's command table. It also verifies whether the command is valid or not. If the command is valid, COMAND will process that command.

CALLS: ERROR, GETSYM, STACKP, FILES, PRNTID, FUNS

MODULE NAME: COMCOM

DESCRIPTION: Processes commands in the compiler mode.

CALLS: none

MODULE NAME: COMEOL

DESCRIPTION: Processes end-of-line in compiler mode.

CALLS: none

MODULE NAME: COMNUM

DESCRIPTION: Processes a digit in compiler mode.

CALLS: RDRNUM

MODULE NAME: COMOP

DESCRIPTION: Processes indices within open parenthesis.

CALLS: none

MODULE NAME: COMPOLY

DESCRIPTION: Forms the polynomial from a set of roots. Both real and imaginary polynomial coefficients are calculated.

CALLS: none

MODULE NAME: COMVAR

DESCRIPTION: Processes variables in the compiler mode.

CALLS: COMEOL, COMOP

MODULE NAME: CONVERT

DESCRIPTION: Called when an equal sign (MCOMM(MPT)=5) is encountered. It replaces all variables to the right of the equal sign with their corresponding data values until all variables have been converted.

CALLS: LISTER

MODULE NAME: CONVRT

DESCRIPTION: Changes the format of an array with real polynomial coefficients and corresponding powers of S to allow a place for the imaginary part of the coefficient.

CALLING SEQUENCE: CALL CONVRT(A,NA,PN,NPN)

A = A double precision array with a two-place format such that each real coefficient of a polynomial is immediately followed with its corresponding power of S.

NA = Number of occupied locations in array A.

PN = A double precision array with a three-place format such that each real coefficient is followed by a zero in the next location and the corresponding power of S is the third location.

NPN = The number of occupied locations that are used in PN(3*NA/2).

CALLS: none

MODULE NAME: COPYIER

DESCRIPTION: Implements the COPY command to transfer variables from one location in the data base to another.

CALLS: TFECHO, MIX, MATMIX, WRITMS, READMS, MATECHO

MODULE NAME: CORINI

DESCRIPTION: Initializes the GWCORE and the VT-125 device driver.

CALLS: INIT, NITSRF, SCLIPW, SCORTP, SELSRF

MODULE NAME: CPLXV

DESCRIPTION: undetermined

CALLS: RTECHO, CONVERT, EXPAND, DELETE

MODULE NAME: CPOLY

FILE NAME: QCPOLY.FOR

DESCRIPTION: Forms a polynominal of order n with n complex eigenvalues.

CALLS: CMULT, WPOLY

MODULE NAME: CROSS

FILE NAME: QCROSS.FOR

DESCRIPTION: This subroutine is used in polynominal multiplication algorithm.

CALLS: none

** MODULE NAME: CROSS

DESCRIPTION: This subroutine displays "+" signs on the screen to help the user to identify major cross points on the graphics grid.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG

MODULE NAME: CUT

FILE NAME: QCUT.FOR

DESCRIPTION: Partition a 2nx2n matrix as

$$Z = \begin{matrix} U_{nxn} & M_{nxn} \\ V_{nxn} & N_{nxn} \end{matrix}$$

CALLS: STACKG

MODULE NAME: CXPAND

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: CTFGTF

DESCRIPTION: This module is used to transfer CLTF into GTF.

CALLS: XFER

MODULE NAME: CTFHTF

DESCRIPTION: This module is used to transfer CLTF into HTF.

CALLS: XFER

MODULE NAME: CTFOLT

DESCRIPTION: This module is used to transfer CLTF into OLTF.

CALLS: XFER

MODULE NAME: DASHER

DESCRIPTION: Draws horizontal and vertical dashed lines.

CALLS: PLOT

MODULE NAME: DATFILL

DESCRIPTION: Fills array HISTORY(802) with NSAMP samples of the time history YVECT(NY) spaced every NSKIP state transitions.

CALLS: UVECTOR, XVECTOR, YVECTOR

MODULE NAME: DBLMULT

DESCRIPTION: Performs double precision polynomial multiplication (Armold).

CALLS: none

MODULE NAME: DECODER

DESCRIPTION: Decodes the information stored in the MCOMM and DATM arrays into one of six entities: COMMAND, VARIABLE, NUMERIC DATA, OPEN PARENTHESIS, EQUAL SIGN, or OPTION NUMBER.

CALLS: PLOT, MODIFY

MODULE NAME: DEFU

DESCRIPTION: Collects an input description from the user and sets up a list of points out on TAPE11 - VECTOR.

CALLS: READS

MODULE NAME: DELETE

DESCRIPTION: Deletes a pole or zero of a transfer function (Armold).

CALLS: EXPAND

MODULE NAME: DELETER

DESCRIPTION: This module determines which function's pole or zero is deleted.

CALLS: READS, DELETE

** MODULE NAME: DELMENU

DESCRIPTION: This subroutine displays the delete menu and takes the appropriate action.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG

MODULE NAME: DERIV3

DESCRIPTION: Takes the derivative of a polynomial.

CALLS: none

MODULE NAME: DESTOY

FILE NAME: DESTROY.FOR

DESCRIPTION: Erases the variables in the storage.

CALLS: STACKG, STACKP

MODULE NAME: DET

DESCRIPTION: Finds the determinant of a matrix using a diagonalizing procedure.

CALLS: none

MODULE NAME: DIAGON

FILE NAME: QDIAGON.FOR

DESCRIPTION: Forms a diagonal matrix (nxn) with a given vector (nx1).

CALLS: none

MODULE NAME: DIGITR

DESCRIPTION: Computes the discrete time response using recursive difference equations.

CALLS: NUMBER, PLOT, PROPGAT, READS, SPAXIS, SYMBOL

MODULE NAME: DIRIV

DESCRIPTION: Takes the derivative of a transfer function with the numerator polynomial located in PN and the denominator polynomial located in PD. It then stores the numerator of the derivative in PN and the denominator in PD.

CALLING SEQUENCE: CALL DIRIV(PN,NPN,PD,NPD)

PN = A double precision array containing the numerator polynomial in the format: real part, imaginary part, and the order of S stored in back-to-back locations (input). Numerator polynomial of the derivative function (output).

NPN = Number of occupied elements in array PN.

PD = A double precision array containing the denominator polynomial in the same format as the numerator polynomial (input). Denominator polynomial of the 0` derivative function (output).

NPD = number of occupied elements in array PD.

CALLS: MULTIP, ADD, MLTPL

MODULE NAME: DISMENU

DESCRIPTION: This routine displays the display menu for the user in the interactive graphics package.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG

MODULE NAME: DISPLY

DESCRIPTION: This module displays a transfer function's roots to a selected number of digits.

CALLS: XFER, CLEAR1

MODULE NAME: DISPLY2

DESCRIPTION: This module displays a transfer function's polynominals to a selected number of digits.

CALLS: XFER, CLEAR1

MODULE NAME: DISPLAY_ALL

FILE NAME: ALL.FOR

AUTHOR: Mark Travis

DESCRIPTION: Basic control routine for the display option which plots all four responses simultaneously.

CALLS: DELALL, FIND_BORDERS, NEWFRM, READS, TOTICE

MODULE NAME: DISPLAY_BODE

FILE NAME: BODE.FOR

AUTHOR: Mark Travis

DESCRIPTION: Plots the magnitude and phase simultaneously.

CALLS: DELALL, NEWFRM, READS, TOTICE

MODULE NAME: DIVI

DESCRIPTION: Divides a double precision complex number by double precision complex number.

CALLS: none

MODULE NAME: DMULR

DESCRIPTION: A polynomial factoring routine.

CALLS: none

MODULE NAME: DNTER

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: DOLOOP

DESCRIPTION: Implements a standard DO LOOP to transfer one array into another array.

C = input array
D = output array
NC = number of occupied elements in C
ND = NC

CALLS: none

** MODULE NAME: DRWCIR

DESCRIPTION: This subroutine draws up a unit circle on the Z-plane interactive graphics grid.

CALLS: SLNDX, CRRSEG, MOVA2, SSTEPLN, LINA2, CLRSEG

** MODULE NAME: DRWTEXT

DESCRIPTION: This routine prompts the user for any text the user desires to put on the interactive root locus plot. The text is saved internally by GWCORE, and can be recalled later.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG, WBLOC2, MNTOW2 SFONT, SCHSIZ, WKEYBD

MODULE NAME: DS

DESCRIPTION: Processes a "$" in compiler and calculator mode and "?" in compiler mode.

CALLS: none

MODULE NAME: ECHOS

DESCRIPTION: An output routine which is used to print polynomial coefficients and roots in a compact table.

CALLS: none

** MODULE NAME: ERASE

DESCRIPTION: This module is responsible for erasing the previous menu. The color is changed to black, and the menu is written again, deleting the menu from the user's view.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG

MODULE NAME: EQID

FILE NAME: LIB.FOR

DESCRIPTION: EQID is used to check whether two given strings are the same or not.

CALLS: none

MODULE NAME: ERROR

FILE NAME: ERROR.FOR

DESCRIPTION: Prints error messages.

CALLS: none

MODULE NAME: EVALU3

DESCRIPTION: Evaluates a polynomial for a complex value (Z = x + jY). It can evaluate both real coefficient and coefficient polynomials.

CALLS: ORDER3

MODULE NAME: EVALU8

DESCRIPTION: Evaluates a polynomial at the pole for which the partial fraction expansion coefficient is being sought.

CALLS: ORDER3

MODULE NAME: EXPAND

DESCRIPTION: Expands the roots of a polynomial into a corresponding set of polynomial coefficients.

CALLS: POLYMLT

MODULE NAME: EXPR

FILE NAME: EXPR.FOR

DESCRIPTION: EXPR processes MATLAB's expression according to the rialroad diagram for EXPR in Narathong's Appendix A.

CALLS: PUTID, GETSYM, ERROR, TERM, STACK1, STACK2

MODULE NAME: FACT

DESCRIPTION: A function subroutine to calculate n-factorial (n!).

CALLS: none

MODULE NAME: FACTO

DESCRIPTION: A setup routine which calls subroutine DMULR to factor a polynomial.

CALLS: DMULR, ROOT, ROOT2

MODULE NAME: FACTOR

FILE NAME: FACTOR.FOR

DESCRIPTION: FACTOR processes MATLAB's factor according to the railroad diagrams for FACTOR in Narathong's Appendix A.

CALLS: ERROR, GETSYM, GETCH, EXPR, STACK1, PUTID, FUNS, STACKG, MATFN'S, STACK2

MODULE NAME: FILES

FILE NAME: FILES.FOR

DESCRIPTION: FILES is a system dependent routine to allocate files.

CALLS: none

MODULE NAME: FINDBORD

DESCRIPTION: This module is used to establish the borders for the plot of the root locus. The location of the borders is calculated based on the location of the poles and zeros.

CALLS: none

MODULE NAME: FLOP

FILE NAME: FLOP.FOR

DESCRIPTION: FLOP is a system dependent double precision function. It counts and possibly chops each floating point operation.

CALLS: none

MODULE NAME: FORM

DESCRIPTION: Forms the denominator polynomial that will be used to evaluate the partial fraction expansion coefficient corresponding to a pole of the plant. The routine multiplies all of the poles together, excluding the pole (and its conjugate if the pole has an imaginary part) for which the partial fraction expansion coefficient is being sought.

CALLS: GETPOL

MODULE NAME: FORMER

FILE NAME: FORMER.FOR

AUTHOR: Mark Travis

DESCRIPTION: Draws ICECAP's valid block diagrams in response to an invalid form command.

CALLS: ARROW, CIRCLE, CLTSEG, CRTSEG, LINA2, MOVA2, MOVR2, NEWFRM, SCHPRE, SCHSIZ, SFONT, SQUARE, SWINDO, TEXT

MODULE NAME: FORMR

FILE NAME: FORMR.FOR

DESCRIPTION: FORMR is a machine dependent routine which prints outputs with a Z format.

CALLS: none

MODULE NAME: FORMT

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: FRACTOR

DESCRIPTION: Calls DMULR to factor polynomials. Real and imaginary roots are stored as separate double precision arrays in COMMON.

CALLS: DMULR

MODULE NAME: FREQR

DESCRIPTION: Responsible for the frequency response analysis.

CALLS: FREQOUT, FREPLOT, NICHOLS, PLOTSET, TEKFREQ, CHALK

MODULE NAME: FREQOUT

DESCRIPTION: Performs continuous and discrete frequency response analyses. FUNCTIONs FW(W) and AW(W) are called to compute magnitude and phase angle versus frequency.

CALLS: READS, PLOT, SPAXIS, BOXER

MODULE NAME: FREPLOT

DESCRIPTION: Draws frequency response plots on a printer or terminal.

CALLS: none

MODULE NAME: FT

DESCRIPTION: (Function) obtains the value of a continuous time response function for a given value of T using the function FTT and performs a superposition on two responses in the case of a pulse input.

CALLING
SEQUENCE: X = F(T)

T = Time at which a response is to be evaluated
INPUTR = 4, input is a pulse
[] 4, input is an impulse, step, ramp, or sinusoid
RWIDTH = width of input pulse in seconds

NOTES: If INPUTR [] 4, FT calls FTT once and simply returns the results to the calling program. If INPUTR = 4, FTT is called twice for values of time T and T-RWIDTH and the results subtracted using superposition to obtain the pulse response.

CALLS: FTT

MODULE NAME: FTOR

DESCRIPTION: Appears to be routine to generate factorials, (N-1)!

CALLS: none

MODULE NAME: FTT

DESCRIPTION: (Function) calculates the value of a continuous time response function for a given value of T using information placed in COMMON by the subroutine TIMER.

CALLING SEQUENCE: X = FTT(T)

T = Time in seconds at which function is to be evaluated. The variables in COMMON form the coefficients of the function to be evaluated which has the form:

```
F1(I) = ZZZ(I) * EXP(XR*T)
F2(I) = YYY(I) * EXP(W(I)*T)
F#(I) = CR(I) * EXP(EC(I)*T) * SIN(OM(I)*T + FE(I)
```

CALLS: none

MODULE NAME: FUNS

FILE NAME: FUNS.FOR

DESCRIPTION: FUNS sets up the MATLAB's functiona; table. It also verifies whether the function is valid or not. If the function is valid, FUNS sets the two control variables namely, FIN, and FUN which will be used to signal MATLAB subroutine to call functional routines, MATFN1 thru MATFN6.

CALLS: PRNTID

MODULE NAME: FW

DESCRIPTION: Calculates the discrete or continuous open or closed loop frequency response magnitude in linear magnitude or decibels for a given frequency in hertz or radians per second.

CALLS: none

MODULE NAME: GAIN

FILE NAME: GAIN.FOR

AUTHOR: Mark Travis

DESCRIPTION: Prompts the user for a value of gain and sends the value to TOTICE.

CALLS: TOTICE

MODULE NAME: GANG1

DESCRIPTION: Computes phase angle of test point for root locus. If GANG = 0, it is a locus point for GA < 0. If GANG = 180, it is a locus point for GA > 0.

CALLS: BANG, BOX

MODULE NAME: GENMMPY

DESCRIPTION: Post-multiplies AMAT by BMAT and stores the result in CMAT. If AMAT and BMAT do not conform, the routine aborts.

CALLS: none

MODULE NAME: GETCH

FILE NAME: GETCH.FOR

DESCRIPTION: GETCH gets the next character from the buffer.

CALLS: none

MODULE NAME: GETLIN

FILE NAME: GETLIN.FOR

DESCRIPTION: GETLIN reads in the input and puts it in the buffer 80 characters at a time.

CALLS: XCHAR, GETCH, PUTID, FILES, EDIT

MODULE NAME: GETPOL

DESCRIPTION: Takes a set of roots and multiplies them to form a polynomial.

CALLS: MULTIP, MLTPL

MODULE NAME: GETSYM

FILE NAME: GETSYM.FOR

DESCRIPTION: GETSYM primarily verifies each character in the buffer which contains 80 characters. This buffer was read previously in by subroutine GETLIN.

CALLS: GETCH, GETVAL, PRNTID

MODULE NAME: GETVAL

FILE NAME: GETVAL.FOR

DESCRIPTION: GETVAL forms a numerical value of each character in the buffer.

CALLS: GETCH

MODULE NAME: GONOGO

DESCRIPTION: Controls printout of user information bulletins.

CALLS: none

---

* MODULE NAME: GRAPH

DESCRIPTION: This is the main driver of the interactive computer graphics program. All of the menus, grids, and data points are controlled from this module. This module uses the Graphics package GWCORE to display all of its graphics.

CALLS: INIT, NITSRF,SELSRF,SCLIPW,SCORTP,SWINDO,SVPRT2,BORDER VTLSIZ, HEADER, GRID,MENU,CRTSEG,WKEYBD,CLTSEG,ERASE,INPMENU, PPOLE,DRWTEXT,DISMENU,CROSS,DRWCIR,STOP,TOTICE,GRALOC,DELMENU DERSEG,GRIDMENU,DELALL,VTALPH

** MODULE NAME: GRALOC

DESCRIPTION: This module displays the computed locus on the selected grid. Every time this module is called, the line color is changed so that the user can differenitate between the various root locus plots.

CALLS: SLNDX, CRRSEG, MOVA2, TEXT, CLTSEG

** MODULE NAME: GRID

DESCRIPTION: This module is responsible for drawing up the specified grid that the user selected, e.g. S-plane, or Z-plane.

CALLS: SLNDX,CRRSEG,MOVA2,LINA2,STNDX,CLRSEG,SCHSIZ,CRTSEG TEXT,CLTSEG

** MODULE NAME: GRIDMENU

DESCRIPTION: This module is responsible for displaying the grid menu selections. Currently, the user does not have any control over the grid.

CALLS: STNDX,CRRSEG,MOVA2,TEXT,CLRSEG

MODULE NAME: GROUP

DESCRIPTION: Used in determining the size of the transfer function for plotting.

CALLS: none

MODULE NAME: GTFHTF

DESCRIPTION: undetermined

CALLS: PHOFS, CADJB, FACTO, CANCEL

MODULE NAME: GTFCTF

DESCRIPTION: This module is used to transfer GTF into CLTF.

CALLS: XFER

** MODULE NAME: HEADER

DESCRIPTION: This module draws a box around the graphics menu area AND displays the words GRAPHICS MENU and CURSOR LOCATION:.

CALLS: SFONT, STNDX, SLNDX, SCHSIZ, CRRSEG, MOVA2, LINA2 TEXT, CLRSEG

MODULE NAME: HELP

DESCRIPTION: Consists of a series of PRINT statements and provides several levels of user assistance.

CALLS: none

MODULE NAME: HTFCTF

DESCRIPTION: This module is used to transfer HTF into CLTF.

CALLS: XFER

MODULE NAME: IDENITY

DESCRIPTION: Sets up identity matrix by zeroing it out and then filling in the diagonal with 1's.

CALLS: MATZERO

MODULE NAME: IMPUL

DESCRIPTION: This is the main module that calculates the transformation of a function using the impluse method. The user is prompt with the option to include a zero order hold or a Padea approximation function.

CALLS:

** MODULE NAME: INPMENU

DESCRIPTION: This subroutine displays the input menu in the interactive graphics program.

CALLS: STNDX, CRTSEG, MOVA2, CLTSEG

MODULE NAME: INSERT

DESCRIPTION: This module overwrites a pole or zero depending on the location parameter K.

CALLS: INSERT_RT, PAUSE, TRIM, PRINT_BUFFER

MODULE NAME: INVDIF

DESCRIPTION: This is the main module that calculates the inverse of the backward difference equation.

CALLS: CLEAR1, READS, BIFORM2

MODULE NAME: INVIMP

DESCRIPTION: This is the main module that calculates the inverse of the impluse equation.

CALLS: READS, POLAR, UNPARTL, FACTO, EXPAND

MODULE NAME: INVTUS

DESCRIPTION: When called by DISCRETE, this module is the main module that calculates the inverse TUSTIN equation.

CALLS: READS, BIFORM

MODULE NAME: LISTER

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: LOCUS_PLOT

FILE NAME: LOCPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: Plots the root locus which is calculated by ROOT12.

CALLS: CIRCLE, CLRSEG, CRRSEG, DELALL, LINA2, LINR2, MOVA2, MOVR2, NEWFRM, PLINA2, ROUNDSTEP, SCGJST, SCHPLA, SCHPRE, SCHSIZ, SCHUP2, SFONT, SVPTR2, TEXT, ZOOM_DATA

MODULE NAME: MADD

DESCRIPTION: Adds or subtracts matrices, CMAT = AMAT + BMAT or CMAT = AMAT - BMAT. If AMAT and BMAT do not have the same dimensions, the routine aborts.

CALLS: none

MODULE NAME: MAG_LABEL

FILE NAME: FRPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: This module determines the label and grid interval for the frequency response magnitude plot.

CALLS: ROUNDSTEP

MODULE NAME: MAGNIFY

DESCRIPTION: This module is used to double the size of the root locus as it appears on the plot. This is done by dividing the location of each boundary by two.

CALLS:

MODULE NAME: MAGNITUDE_PLOT

FILE NAME: FRPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: This module uses the GWCORE to plot the frequency response magnitude.

CALLS: CLRSEG, CRRSEG, DELALL, LINA2, LINR2, MAG_LABEL, MOVA2, MOVR2, NEWFRM, PLINA2, SCHJST, SCHPLA, SCHPRE, SCHSIZ, SCHUP2, SFONT, SVPRT2, TEXT

MODULE NAME: MATECHO

DESCRIPTION: An output routine designed to print the elements of a matrix of arbitrary dimensions.

CALLS: none

MODULE NAME: MATFN1

FILE NAME: MATFN1.FOR

DESCRIPTION: MATFN! evaluates functions involved in Gaussian elimination.

CALLS: ERROR, WGECO, WGESL, RSET, WCOPY, WGEDI, WGEFA, WSWAP, HILBER, WSCAL

MODULE NAME: MATFN2

FILE NAME: MATFN2.FOR

DESCRIPTION: MATFN2 evaluates elementary functions and functions involved in eigenvalues and eigenvectors.

CALLS: ERROR, WCOPY, WSET, HTRIDI, IMTQL2, HTRINK, CORTH, COMQR3, WLOQ, QMUL, QATAN, WSQRT, WSCAL, WAXPY, WDIV

MODULE NAME: MATFN3

FILE NAME: MATFN3.FOR

DESCRIPTION: MATFN3 evaluates functions involved in singular valus decomposition.

CALLS: ERROR, WSVDC, WCOPY, WRSCAL

MODULE NAME: MATFN4

FILE NAME: MATFN4.FOR

DESCRIPTION: MATFN4 evaluates functions involved in QR decomposition in least squares sense.

CALLS: ERROR, STACK1, WCOPY, WSET, WQRDC, WQRSL, WSWQP

MODULE NAME: MATFN5

FILE NAME: MATFN5.FOR

DESCRIPTION: MATFN5 performs file handling and other I/O operations.

CALLS: ERROR, FILES, PRINT, PUTID, SAVLOD, STACKP, RSET, RAT, BASE, WCOPY, STACK1, PLOT

MODULE NAME: MATFN6

FILE NAME: MATFN6.FOR

DESACRIPTION: MATFN6 evaluates utility functions such as MAGIC, KRONECKER, PRODUCT, SIZE, EYE , RAND, etc.

CALLS: ERROR, WCOPY, WMUL, WDIV, USER, RSET, MAGIC, WSET

MODULE NAME: MATIN

DESCRIPTION: An input routine which interactively requests the input of a matrix one row (or column) at a time.

CALLS: READS, MATECHO

MODULE NAME: MATLAB

FILE NAME: MATLAB.FOR

DESCRIPTION: MATLAB is used to initialize all necessary control variables and flags.

CALLS: FILES, WSET, PUTID, PARSE, MATFN1 thru MATFN6

MODULE NAME: MATMIX

DESCRIPTION: undetermined

CALLS: MATRAN

MODULE NAME: MATRAN

DESCRIPTION: undetermined

CALLS: XMAT

MODULE NAME: MATRIX

DESCRIPTION: Responsible for the operation of the matrix manipulation subroutines: MATS, MATOPR and MOREMAT.

CALLS: MATOPR, MATS, MOREMAT

MODULE NAME: MATOPR

DESCRIPTION: Performs all matrix operations such as add, multiply, matrix inverse, and transpose.

CALLS: PHOFS, FACTO, ECHOS, MADD, MATECHO, GENMMPY, MINV, TRANPOS, GTFHTF

MODULE NAME: MATS

DESCRIPTION: Obtains input for all matrix operations.

CALLS: MATIN, READS, MATZERO, IDENITY, TEST

** MODULE NAME: MENU

DESCRIPTION: This module displays the main menu commands to the user. When a command is selected, the appropriate menu is then displayed.

CALLS: STNDX, CRTSEG, MOVA2, TEXT, CLTSEG

MODULE NAME: MATZERO

DESCRIPTION: Generates a "zero" matrix.

CALLS: none

MODULE NAME: MINV

DESCRIPTION: Performs the inversion of a square non-singular matrix of maximum size 10 X 10. If the matrix is not square, singular, or too large, the routine aborts.

CALLS: none

MODULE NAME: MISCELL

DESCRIPTION: Performs four miscellaneous options: erase Tektronix screen, make hardcopy (Tektronix), list mode control switch settings, and compute the integral of (CLTF)2/2PI.

CALLS: ERASE, HDCOPY, MSQINT

MODULE NAME: MIX

DESCRIPTION: undetermined

CALLS: TRANFER

MODULE NAME: MIXPOL

FILE NAME: MIXPOL.FOR

DESCRIPTION: Forms a polynominal of order n with given real and complex eigenvalues.

CALLS: WPOLY, CPOLY, RPOLY, QROOT

MODULE NAME: MLTPL

DESCRIPTION: Multiplies the real polynomial coefficients by a scale factor and stores the resultant polynomial in a new array.

CALLS: none

MODULE NAME: MMPY

DESCRIPTION: A special matrix multiply routine used by PHOFS and EXPAND. It multiplies AMAT and BMAT and stores the product in CMAT.

CALLS: none

MODULE NAME: MODERN

FILE NAME: MODERN.FOR

DESCRIPTION: Performs a state variable feedback design using phase variable representation using the algorithm presented in section 3.4 of Narathong's thesis.

CALLS: TFORM, STACKG, QSAVE, NUM

MODULE NAME: MODIFY

DESCRIPTION: undetermined

CALLS: CONVERT, UP, NODV, FACTOR, NODI, ONEDV, CPLXV, TWODV

MODULE NAME: MOREMAT

DESCRIPTION: Performs matrix augmentation operations.

CALLS: GENMMPY, MADD, MATZERO

MODULE NAME: MPOLY

FILE NAME: MPOLY.FOR

DESCRIPTION: Performs polynominal multiplication using the algorithm presented in section 3.3 of Narathong's thesis.

CALLS: WPOLY

MODULE NAME: MRIC

FILE NAME: RICCATI.FOR

DESCRIPTION: MRIC solves the continuous time algebraic matrix Riccati equation, using the eigen number approach.

CALLS: FORMR, SORT, RPOLY, CPOLY, MIXPOLY, NEST, CUT, ANSWER1, ANSWER2, QSAVE, DESTOY, MATFN2

MODULE NAME: MSQINT

DESCRIPTION: undetermined

CALLS: DET

MODULE NAME: MULT

DESCRIPTION: Multiplies two double precision complex numbers.

CALLS: none

MODULE NAME: MULTIP

DESCRIPTION: Multiplies two polynomials and then calls SIMPLE to combine the coefficients with like powers of S.

CALLS: SIMPLE

MODULE NAME: NEST

FILE NAME: QNEST.FOR

DESCRIPTION: Multiplies a matrix polynominal using a nest multiply algorithm.

CALLS: STACK2, STACKG

MODULE NAME: NICHOLS

DESCRIPTION: Draws a log-magnitude/angle plot (Nichols chart).

CALLS: PLOT, DASHER, SPAXIS, TITLES, SYMBOL, NUMBER

MODULE NAME: NODI

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: NODV

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: NOTICE

DESCRIPTION: Prints out user information bulletins.

CALLS: GONOGO

MODULE NAME: NUM

FILE NAME: NUM.FOR

DESCRIPTION: Computes coefficients of a numerator of a transfer function using the algorithm in section 3.2 of Marathong's thesis.

CALLS: MATMUL, MATVEC, VECPRO

MODULE NAME: OLTCTF

DESCRIPTION: This module is used to transfer OLTF into CLTF.

CALLS: XFER

MODULE NAME: ONEDV

DESCRIPTION: undetermined

CALLS: POLECHO, CONVERT, FACTO

MODULE NAME: OPTIMAL

FILE NAME: OPTIMAL.FOR

DESCRIPTION: Computes a feedback matrix gain K using a positive definite solution to Riccati equation and a closed loop matrix.

CALLS: STACK2, STACKG, STACKP

MODULE NAME: ORDER3

DESCRIPTION: Orders polynomial coefficients in descending powers.

CALLS: none

MODULE NAME: ORDPOLE

DESCRIPTION: Checks for multiple poles and stores the multiplicity in the array MULPOLE. The extra multiple poles are deleted and only a single copy of each pole is stored in the array DBPOLE.

CALLS: none

MODULE NAME: PARSE

FILE NAME: PARSE.FOR

DESCRIPTION: PARSE controls the interpretation of each statement. It calls subroutines that process the various syntatic quantities such as command, expression, term, and factor.

CALLS: FILES, PROMPT, GETLIN, PUTID, GETSYM, COMAND, FUNS, ERROR, STACKP, CLAUSE, EXPR, TERM, FACTOR

MODULE NAME: PARTFR

DESCRIPTION: Performs the partial fraction e[illegible]nsion required by ADVANZ.

CALLS: GETPOL, FORM, CONVRT, MULTIP, MLTPL, DIRIV, [illegible]ALU8

MODULE NAME: PARTL

DESCRIPTION: Performs a Heaviside partial fraction expansion and computes the time function from the inverse LaPlace Transform of the partial fraction terms.

CALLS: FT, FTOR, NUMBER, PLOT, POLAR, READS, SPAXIS, SPECS, SYMBOL

MODULE NAME: PCHAR

DESCRIPTION: Checks for legal P-value (called by READS).

CALLS: none

MODULE NAME: PEAK

DESCRIPTION: Finds the first value of T after T = TMIN where the slope is zero, using an iterative search technique.

CALLS: FT

MODULE NAME: PHASE_LABEL

FILE NAME: PHPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: Determines the labeling interval for the phase shift plot. The label values are written to disk file "PHLABEL.DAT".

CALLS: ROUNDSTEP

MODULE NAME: PHASE_PLOT

FILE NAME: PHPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: Plots the phase shift generated by FREQOUT.

CALLS: CLRSEG,CRRSEG, DELALL, LINA2, LINR2, MOVA2, NEWFRM, PHASE_LABEL, PLINA2, SCHJST, SCHPLA, SCHPRE, SCHSIZ, SCHUP2, SFONT, SVPRT2, TEXT

MODULE NAME: PHOFS

DESCRIPTION: Uses Leverrier's Algorithm to compute adj[SI-A] and det[SI-A]. If the input matrix AMAT is not square, the routine is aborted.

CALLS: MMPY

MODULE NAME: PLANCHG

DESCRIPTION: This module provides the user with the option to change planes, i.e. S-plane, Z-plane, W-plane, and W'-plane (designated as WP).

CALLS: CLEAR1

MODULE NAME: PLOT

FILE NAME: PLOT.FOR

DESCRIPTION: PLOT is used to X versus Y on specified unit number.

CALLS: none

MODULE NAME: PLOTFIN

DESCRIPTION: Finishes CALCOMP plots for PARTL and DIGITR by drawing boxes, grids, axes, tic marks, and titles.

CALLS: ANMODE, BELL, CHRSIZ, DASHER, DRAWA, INITT, MOVEA, MOVABS, PLOT, SPAXIS, SWINDO, SYMBOL, TITLES, VWINDO

MODULE NAME: PLOTSET

DESCRIPTION: Finishes the CALCOMP plots for frequency responses by drawing boxes, axes, titles, labels, and grid lines.

CALLS: PLOT, DASHER, TITLES, SYMBOL, NUMBER, SPAXIS

MODULE NAME: POLAR

DESCRIPTION: Converts a set of Cartesian coordinates AC and BD to polar form as a magnitude (FACT) and an angle (FACTR).

CALLS: none

** MODULE NAME: PPOLE

DESCRIPTION: This is a major routine within the interactive graphics program which displays the cursor, draws the X or O for the poles and zeros, scales the position of the cursor and displays the values to the screen, stores the values in the appropriate variables and erases the values from the screen.

CALLS: STNDX, WBLOC2, MNTOW2, SCHSIZ, CRRSEG, MOVA2, TEXT, CLRSEG, SFONT, CRTSEG, CLTSEG, WBUTN, VTALPH

MODULE NAME: POLE

DESCRIPTION: Calculates a low-rate pole in the Z-plane from a given high-rate pole in the Z-plane.

CALLS: none

MODULE NAME: POLECHO

DESCRIPTION: An output routine which tabulates polynomial coefficients to ten decimal places with corresponding index numbers.

CALLS: none

MODULE NAME: POLY

DESCRIPTION: Performs transfer function input and polynomial operations. The polynomial roots are passed back through COMMON and program control is returned to POLY.

CALLS: ALPHA, ECHOS, EXPAND, POLYADD, POLYMLT, POLYSUB, READS, SWAP, SWAPER, XFER

MODULE NAME: POLYADD

DESCRIPTION: Performs polynomial addition, POLYC = POLYA + POLYB.

CALLS: none

MODULE NAME: POLYCO

DESCRIPTION: Forms a polynomial from a set of roots. Both real and imaginary coefficients are calculated.

CALLS: none

MODULE NAME: POLYM

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: POLYMLT

DESCRIPTION: Performs polynomial multiplication, POLYC = POLYA * POLYB. If the order of the resulting POLYC is greater than 50, the routine aborts.

CALLS: none

MODULE NAME: POLYSUB

DESCRIPTION: Performs polynomial subtraction, POLYC = POLYA - POLYB.

CALLS: POLYADD

MODULE NAME: PRINT

FILE NAME: PRINT.FOR

DESCRIPTION: PRINT serves as a primary output routine.

CALLS: FILES, PRNTID

MODULE NAME: PRNTID

FILE NAME: PRNTID.FOR

DESCRIPTION: PRNTID prints the variable names.

CALLS: none

MODULE NAME: PROMPT

FILE NAME: PROMPT.FOR

DESCRIPTION: PROMPT is used to issue MATLAB prompt with an optional pause.

CALLS: none

MODULE NAME: PROPGAT

DESCRIPTION: Iterates a recursive difference equation of the form:

$$c(k) = a(1)r(k) + a(2)r(k-1) + \ldots + a(n)r(k-N+1) - b(2)c(k-1) - \ldots - b(N)c(k-N+1)$$

to obtain the current value of c(k) given the past N inputs (r) and outputs (c).

CALLING SEQUENCE: CALL PROPGAT(A,B,R,C,N,K)

A = Vector array of coefficients of R terms
B = Vector array of coefficients of C terms
R = Vector array of past N inputs (including present)
C = Vector array of past N outputs (including present)
N = Order of difference equation
K = Current index value

CALLS: RIN

MODULE NAME: PUTID

FILE NAME: LIB.FOR

DESCRIPTION: PUTID is used to store the variable name into the storage.

CALLS: none

MODULE NAME: QROOT

FILE NAME: QROOT.FOR

DESCRIPTION: QROOT selects n positive real parts of eigen-values.

CALLS: none

MODULE NAME: QSAVE

FILE NAME: QSAVE.FOR

DESCRIPTION: Saves an output with a given name.

CALLS: GETLIN, GETSYM, STACKP

MODULE NAME: RDRNUM

DESCRIPTION: Processes a digit for READER.

CALLS: none

MODULE NAME: READER

DESCRIPTION: Replaces the FORTRAN READ function. READER has two modes: compiler and calculator. In the compiler mode, READER processes user input one character at a time and stores valid commands and data in MCOMM and DATM, respectively. The calculator mode accepts user input for the simulated Hewlett-Packard (HP-45) calculator.

CALLS: RINIT, FORMT, AZCALC, CALNUM, CALCK, DS, COMEOL, AZCOMP, COMNUM

MODULE NAME: READS

DESCRIPTION: Interactive input routine which provides error protection and recovery and allows the user to retain control of the calling program, even when it is requesting some numerical data input. It is designed to be used in place of the standard FORTRAN READ statement whenever any input is needed.

CALLS: TOTNUM, READER, HELP, RSCHAR, PCHAR

MODULE NAME: RECALL

DESCRIPTION: Performs calculator RECALL command.

CALLS: RDRNUM

MODULE NAME: RES1

DESCRIPTION: Forms the overall low-rate transfer function from residues for simple poles.

CALLS: MULTIP, ADD, ORDER3, SIMPLE, ROTPOLY, COMPLOY

MODULE NAME: RES2

DESCRIPTION: Forms the overall low-rate transfer function from residues for multiple poles (multiplicity = two).

CALLS: MULT, MULTIP, ADD, ORDER3, SIMPLE, ROTPOLY

MODULE NAME: RES3

DESCRIPTION: Forms the overall low-rate transfer function from residues for multiple poles (multiplicity = three).

CALLS: MULT, MULTIP, ADD, ORDER3, SIMPLE, ROTPOLY

MODULE NAME: RIN

DESCRIPTION: Computes the current input r(k) for use by PROPGAT. It is capable of generating an impulse, step, ramp, pulse, or sinusoid depending upon the value of INPUTR.

CALLS: none

MODULE NAME: RINIT

DESCRIPTION: Initializes READER variables

CALLS: none

MODULE NAME: ROOT10

DESCRIPTION: The main module responsible for controlling the required function needed to perform root locus analysis.

CALLS: ROOT11, ROOTS, ROOT12, ADAPT, TITLES, SMULR, BLEND, PLOT

MODULE NAME: ROOT11

DESCRIPTION: Draws boxes, titles, axes, labels, poles, and zeros (everything except actual root locus branches) on the CALCOMP plot. ROOT11 also searches for a specified Zeta, and draws the unit circle (in the Z-plane), the Zeta-line is the S- or W-plane, and the Zeta-curve in the Z-plane.

CALLS: SYMBOL, PLOT, NUMBER, DNTER, CNTER, GANG1, ANG1, GROUP, UCIRC

MODULE NAME: ROOT12

DESCRIPTION: Computes points on each branch of the locus within the boundaries of calculation using an iterative search technique.

CALLS: ANG1, BANG, BOX, GANG1, SYMBOL, NUMBER, PLOT, BREAK, SEEK

MODULE NAME: ROOTS

DESCRIPTION: Calculates the closed loop transfer function in polynomial form for a given value of gain. ROOTS also factors the denominator polynomial and prints out closed loop poles at the gain of interest.

CALLS: SMULR, SYMBOL

MODULE NAME: ROOT2

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: RPOLY

FILE NAME: QRPOLY.FOR

DESCRIPTION: RPOLY froms a polynominal of order n with n real eigenvalues.

CALLS: DIAGON, MATFN2

MODULE NAME: ROTPOLY

DESCRIPTION: Finds roots of a polynomial.

CALLS: none

MODULE NAME: ROUNDSTEP

FILE NAME: ROUNDSTEP.FOR

AUTHOR: Mark Travis

DESCRIPTION: Performs rounding on the number to be used for the grid scale.

CALLS: none

MODULE NAME: RSCHAR

DESCRIPTION: Checks for legal R or S register (called by READS).

CALLS: none

MODULE NAME: RTECHO

DESCRIPTION: An output routine designed to print out the real and imaginary parts of an array of polynomial roots to ten decimal places. The routine also prints an index number for each root.

CALLS: none

MODULE NAME: SAMPCHG

DESCRIPTION: This module enables the parameters TSAMP to be changed and displayed for conformation.

CALLS: READS

MODULE NAME: SAVLOD

FILE NAME: SAVLOD.FOR

DESCRIPTION: SAVLOD is used for save and load data to and from the user disk.

CALLS: none

MODULE NAME: SCALER

DESCRIPTION: Takes minimum and maximum values of a function and ...

CALLS: none

MODULE NAME: SEEK

DESCRIPTION: Searches boundaries for any locus branches which re-enter the region of calculation after leaving it and for any locus branches which start outside the region of calculation and then enter it.

CALLS: GANG1, ANG1, BANG, BOX

MODULE NAME: SHRINK

DESCRIPTION: This module is used to shrink the size of the root locus as it appears on the plot by a factor of two. This is done by multiplying the location of each boundary by two.

CALLS:

MODULE NAME: SIMPLE

DESCRIPTION: Simplifies a polynomial by adding coefficients of like powers.

CALLS: none

MODULE NAME: SIMULAT

DESCRIPTION: Performs state transition simulation operations.

CALLS: READS, XVECTOR, YVECTOR, UVECTOR, PLOT, DATFILL, SCALER, SPAXIS, SYMBOL, DEFU, NUMBER

MODULE NAME: SLIDE

FILE NAME: SLIDE.FOR

AUTHOR: Mark Travis

DESCRIPTION: Draws the title slide for Graphical ICECAP.

CALLS: CLTSEG, CRTSEG, LINA2, MOVA2, MPICC, SCHJST, SCHPRE, SCHSIZ, SCHSPA, SFONT, SVPRT2, TEXT

MODULE NAME: SMULR

DESCRIPTION: Used in factoring input numerator and denominator polynomials when the input transfer function is in factored form.

CALLS: none

MODULE NAME: SORT

FILE NAME: SORT.FOR

DESCRIPTION: SORT rearranges the eigenvalues. It puts the largest eigenvalue in the top of the stack and the smallest in the bottom.

CALLS: none

MODULE NAME: SPECS

DESCRIPTION: Finds the continuous time response figures of merit: rise time, duplication time, peak time, settling time, peak value, and final value. It also writes these values to an output device.

CALLING
SEQUENCE: CALL SPECS

```
    NGO = 6, Output is written to file ANSWER
        = 7, Output is written to user's terminal
 FINVAL = Final value of response computed by subroutine
          TIMER
    DEL = Iteration step size
```

CALLS: ZEROIN, PEAK, FT

MODULE NAME: SQUARE

FILE NAME: SQUARE.FOR

AUTHOR: Mark Travis

DESCRIPTION: This module draws a rectangle for making block diagrams.

CALLS: LINR2, MOVR2

MODULE NAME: STACK1

FILE NAME: STACK1.FOR

DESCRIPTION: STACK1 performs unary operations and a transpose of a matrix since these operations are very simple. For a serious matrix computation, the LINPACK and EISPACK is used.

CALLS: WRSCAL, ERROR, WCOPY

MODULE NAME: STACK2

FILE NAME: STACK2.FOR

DESCRIPTION: STACK2 performs binary and ternary operations such as addition, substraction, multiplication, etc.

CALLS: ERROR WAXPY, WCOPY, WSCAL, WDIV, WMUL

MODULE NAME: STACKG

FILE NAME: STACKG.FOR

DESCRIPTION: STACKG is used to load data from the bottom of the stack to the top of the stack. This data will then be used in the actual computations.

CALLS: PUTID, ERROR, WCOPY

MODULE NAME: STACKP

FILE NAME: STACKP.FOR

DESCRIPTION: STACKP is used to put variables into stacks.

CALLS: ERROR, FUNS, PUTID, WCOPY, WSET, PRINT

** MODULE NAME: STOP

DESCRIPTION: This routine transfers all of the stored poles and zeros, the OLTF polynominal, the number of poles and zeros to ICECAP and returns the user to ICECAP.

CALLS: EXPAND

MODULE NAME: STORE

DESCRIPTION: Performs calculator STORE command.

CALLS: RDRNUM

MODULE NAME: STOW

DESCRIPTION: This module allows the user to perform a transformation from the S-domain to the W-domain. It uses the impluse method and the bilinear method, which is the joining of the modules IMPUL and ZTOW.

CALLS: READS, ZOH, ZOH1, POLAR, UNPARTL, FACTO, EXPAND, ZOHD BIFORM

MODULE NAME: STOWP

DESCRIPTION: This module performs the transformation from the S-domain to the W'-domain. It is the joining of the two modules IMPUL and ZTOWP.

CALLS:

MODULE NAME: SWAP

DESCRIPTION: undetermined

CALLS: READS, XFER, EXPAND

MODULE NAME: SWAPER

DESCRIPTION: undetermined

CALLS: XFER, ALPHA, SWAP, ECHOS

MODULE NAME: SZWROOT

DESCRIPTION: Performs the root conversion between S, Z, W and W' planes.

CALLS: none

MODULE NAME: TEKFREQ

DESCRIPTION: Plots frequency response on Tektronix Model 4010 and 4014 Terminals.

CALLS: INITT, VWINDO, SWINDO, MOVABS, DRWABS, MOVEA, CHRSIZ, ANMODE, BELL

MODULE NAME: TERM

DESCRIPTION: Calculates the individual terms used in the bilinear transformation mechanized in Subroutine BILIN.

CALLS: COMPOLY, MULTIP, COEFF

MODULE NAME: TEST

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: TFECHO

DESCRIPTION: An output routine which tabulates the numerator and denominator polynomial coefficients and roots of a transfer function in a compact form.

CALLS: none

MODULE NAME: TFORM

FILE NAME: TRANSFORM.FOR

DESCRIPTION: Computes a transformation matrix which transforms a state equation from a physical variable from to a phase variable form.

CALLS: STACKG, STACK2

MODULE NAME: TIME_LABEL

FILE NAME: TIMPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: Determines the appropriate grid line spacing for the time response plot.

CALLS: ROUNDSTEP

MODULE NAME: TIME_PLOT

FILE NAME: TIMPLT.FOR

AUTHOR: Mark Travis

DESCRIPTION: Plots the continuous time response using data calculated by subroutine PARTL.

CALLS: CLRSEG, CRRSEG, DELALL, LINA2, LINR2, MOVA2, MOVR2, NEWFRM, PLINA2, SCHJST, SCHPLA, SCHPRE, SCHSIZ, SCHUP2, SFONT, SVPRT2, TEXT, TIME_LABEL

MODULE NAME: TIMER

DESCRIPTION: Performs the continuous and discrete time response analyses through three secondary overlays.

CALLS: PARTL, DIGITR, PLOTFIN

MODULE NAME: TITLES

DESCRIPTION: Processes input for CALCOMP and TEKPLOT titles.

CALLS: none

MODULE NAME: TOTICE

DESCRIPTION: This module is the main interface between the new ICECAP modules and the old VAXTOTAL modules. ICECAP takes commands that have been formu;ated by the user and translates them to option numbers and commands that VAXTOTAL normally processes. TOTICE passes these option numbers and commands to the VAXTOTAL modules for action and then returns control back to ICECAP.

CALLS:

MODULE NAME: TOTINI

DESCRIPTION: This module is the initialization module for the VAXTOTAL modules that are used in ICECAP. Statements were added to set the default domain to the S-plane.

CALLS: WRITMS

MODULE NAME: TOTNUM

DESCRIPTION: Processes a digit.

CALLS: none

MODULE NAME: TRANFER

DESCRIPTION: undetermined

CALLS: XFER

MODULE NAME: TRANPOS

DESCRIPTION: Transposes (exchanges rows and columns) AMAT to form CMAT.

CALLS: none

MODULE NAME: TRFF

FILE NAME: TRANSFER.FOR

DESCRIPTION: Performs a C(SI -A)*-1B function.

CALLS: STACKG, ERROR, MATFN2, NUM, WCOPY, MCOPY, COPYIER

MODULE NAME: TTYPLOT

DESCRIPTION: Generates a printer plot of the root locus. Each point is stored as an X-Y coordinate in the local file DOODLE, resulting in a 61 X 71 matrix of points and grids. Locus points are rounded to the nearest discrete value on the plot.

CALLS: none

MODULE NAME: TUSTIN

DESCRIPTION: This is the main module that calculates the TUSTIN transformation.

CALLS: READS, BIFORM

MODULE NAME: TWODV

DESCRIPTION: undetermined

CALLS: MATECHO, CONVERT

MODULE NAME: TXCONV

DESCRIPTION: Calculates a low-rate discrete transform from a given high-rate discrete transform.

CALLS: READS, BILIN, ORDER3, MULTIP, ORDPOLE, ZMULT1, RES1, RES2, RES3, WMULT1, WMULT2, WMULT3, DOLOOP, ADD, COEFF, ROTPOLY, CANROOT, COMPOLY, WZBILIN

MODULE NAME: UCIRC

DESCRIPTION: Draws the unit circle for Z-plane loci.

CALLS: PLOT

MODULE NAME: UNPARTL

DESCRIPTION: undetermined

CALLS: CXPAND

MODULE NAME: UPDATE

DESCRIPTION: Transfers (reads/writes) information between TOTAL's COMMON and the local file MEMORY.

CALLS: CLOSMS, PLOTE

MODULE NAME: UPDA

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: UVECTOR

DESCRIPTION: Gets the next value of the U vector.

CALLS: none

MODULE NAME: WMULT1

DESCRIPTION: Calculates residues for simple W-plane poles.

CALLS: WPOLE, EVALU3, DERIV3, DIVI

MODULE NAME: WMULT2

DESCRIPTION: Calculates residues for W-plane poles with multiplicity of two.

CALLS: WPOLE, EVALU3, DERIV3, DIVI, MULT

MODULE NAME: WMULT3

DESCRIPTION: Calculates residues for W-plane poles with multiplicity of three.

CALLS: WPOLE, EVALU3, DERIV3, DIVI, MULT

MODULE NAME: WPLN

DESCRIPTION: Performs W-plane and W'-plane manipulations.

CALLS: CDEXP, MULT, DIVI, ROOT, POLYCO

MODULE NAME: WPOLE

DESCRIPTION: Calculates a low-rate pole in the W- or W'-plane from a given high-rate pole in the W- or W'- plane.

CALLS: DIVI, MULT

MODULE NAME: WPOLY

FILE NAME: WPOLY.FOR

DESCRIPTION: WPOLY multiplies a column vector of order (nx1) and a row vector (1xn) together.

CALLS: CROSS

MODULE NAME: WPTOS

DESCRIPTION: This module performs the multiple transformation first from W' to Z via bilinear method, and then from Z to S via inverse impluse method.

CALLS:

MODULE NAME: WPTOZ

DESCRITPION: This module performs the bilinear transformation from W' to the Z.

CALLS:

MODULE NAME: WTOS

DESCRIPTION: This module performs the multiple transformation first from W to Z via bilinear method, and then from Z to S via inverse impluse method.

CALLS: READS, BIFORM, POLAR, UNPARTL, FACTO, EXPAND

MODULE NAME: WTOZ

DESCRIPTION: This module performs the bilinear transformation from W to Z.

CALLS: BIFORM

MODULE NAME: WZBILIN

DESCRIPTION: Initiates the specific bilinear transformation from the W'-plane to the Z-plane. The actual transformation is performed by BILIN.

CALLS: ORDER3, COEFF, BILIN

MODULE NAME: XCHAR

FILE NAME: XCHAR.FOR

DESCRIPTION: XCHAR is a system dependent routine to handle special characters.

CALLS: none

MODULE NAME: XFER

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: XMAT

DESCRIPTION: undetermined

CALLS: none

MODULE NAME: XVECTOR

DESCRIPTION: Propagates X vector one iteration.

CALLS: UVECTOR, MMPY

MODULE NAME: YVECTOR

DESCRIPTION: Computes Y = [CMAT] X + [DMAT] U.

CALLS: MMPY

MODULE NAME: ZEROIN

DESCRIPTION: Finds the first value of T after T = TMIN, where the function FT(T) equals some specified value GOAL using an iterative search procedure.

CALLS: FT

MODULE NAME: ZFORMS

DESCRIPTION: Computes Z-transforms and inverse Z-transforms using impulse invariance, first difference approximation, and Tustin approximation techniques.

CALLS: READS, POLAR, UNPARTL, FACTO, EXPAND, BIFORM, MPY

MODULE NAME: ZMULT1

DESCRIPTION: Calculates residues for simple Z-plane poles.

CALLS: POLE, EVALU3, DERIV3, DIVI

MODULE NAME: ZOH

DESCRIPTION: This module multiplies a transfer function by the Padea approximation, i.e. 2/(S+[2/TSAMP]).

CALLS:

MODULE NAME: ZOHD

DESCRIPTION: This module multiplies a transfer function by (Z-1)/Z.

CALLS: DBMULT, FACTO

MODULE NAME: ZOH1

DESCRIPTION: This module multiplies a transfer function by 1/S so that once transformed to the Z-domain it can be multiplied by (Z-1)/Z.

CALLS:

MODULE NAME: ZOOM_FLAG

FILE NAME: ZOOMFLAG.FOR

AUTHOR: Mark Travis

DESCRIPTION: Sets the logical variable zoom to true for control of the viewport in LOCUS_PLOT.

CALLS: none

MODULE NAME: ZTOW

DESCRIPTION: This is the main module that performs the bilinear transformation from the Z-domain to the W-domain.

CALLS:

MODULE NAME: ZTOWP

DESCRIPTION: This is the main module that performs the transformation from the Z-domain to the W'-domain (desginated as WP in the program).

CALLS: READS, BIFORM

B.4 Summary

This appendix has provided descriptions of the new FORTRAN modules being used in ICECAP and descriptions of the VAXTOTAL FORTRAN modules that were revised so they could be used in ICECAP. Furthermore, a description of all the VAXTOTAL modules prior to modification is included in this appendix. A complete source listing of all these modules is maintained in the AFIT Digital Equipment Laboratory.

APPENDIX C

FLOW CHART OF ICECAP

C.1 INTRODUCTION

This appendix contains a structure chart of ICECAP. Every subroutine that is called is identified. Each level is denoted by six dots which are used to separate the different levels. A longer structure chart is available on the VAX 11/780, DUA3: [DERRICK.DIR]. The longer structured chart contains every call that is made within ICECAP,in the order that the call is made. This list is a subset of that longer list,where only the first call is recorded, not multiple calls.

C.2 STRUCTURED CHART

The structured chart is a flow chart of the 3.1 version of ICECAP stored in DUA1: [ICECAP1.MODULES]. All of the Pascal procedures, and FORTRAN functions and subroutines are included. The description of each of these routines is described in appendix B. Each of these names is the title of the subroutine or procedure. The file name may be different than the subroutine, and this has been integrated into the module description whenever a difference exists. It is hoped that future efforts will modify this structured chart, thus providing a structured document which will aid in the design and integration of new capabilities for ICECAP.

C.3 ICECAP STRUCTURED CHART

```
ICER
|...... CLEAR
|...... TITLE_SLIDE
|............. CLEAR
|............. GRAPHICS
|............. HIGHLIGHT
|............. NOHIGHLIGHT
|............. BOXIT
|.................... GRAPHICS
|.................... CURSORRC
|.................... NOGRAPHICS
|............. CURSORRC
|............. READLN
|...... TOTINI
|............. WRITMS
|...... MATINI
|............. FILES
|............. WSET
|............. PUTID
|............. SETUP
|.................... UNTRAP
|........................... LIB$INSV
|.................... LIB$ESTABLISH
|.................... UNTRAP
|.................... LIB$INSV
|...... HELP_PROMPT
|............. CLEAR
|............. HIGHLIGHT
|............. NOHIGHLIGHT
|...... READCOM
|............. UNPACK
|............. PACK
|...... INTERPRET
|............. DICTIONARY
|.................... TRIM
|........................... UNPACK
|........................... PACK
|.................... PAUSE
|........................... HIGHLIGHT
|........................... NOHIGHLIGHT
|........................... READLN
|........................... CLEAR
|............. PAUSE
|............. READCOM
|............. PACK_BUFFER
|.................... UNPACK
|.................... PACK
```

```
|.............. TOTICE
|.................... CLEAR1
|.................... READER
|........................... RINIT
|........................... FORMT
|........................... AZCALC
|.................................. CALVAR
|........................................ LISTER
|............................................... GOTOER
|.................................. CALKEY
|........................................ CALCTR
|............................................... RDRNUM
|............................................... FORMT
|............................................... STORE
|...................................................... RDRNUM
|............................................... RECALL
|...................................................... RDRNUM
|.................................. CALABB
|........................... CALNUM
|.................................. RDRNUM
|........................... CALCK
|.................................. CALCTR
|.................................. CALNUM
|........................... DS
|........................... COMEOL
|........................... AZCOMP
|.................................. COMCOM
|.................................. COMVAR
|........................................ COMEOL
|........................................ COMOP
|.................................. COMABB
|........................... COMNUM
|.................................. RDRNUM
|.................... GOTOER
|.................... PLOTST
|.................... PLOT
|.................... GOTOER
|.................... UPDATE
|........................... PLOTND
|........................... CLOSMS
|.................... TIMER
|........................... PARTL
|.................................. GOTOER
|.................................. POLAR
|.................................. FTOR
|.................................. READS
|........................................ TOTNUM
|........................................ READER
|........................................ HELP
|........................................ RSCHAR
```

```
|........................................ PCHAR
|................................ FT
|........................................ FTT
|................................ READS
|................................ SPECS
|........................................ ZEROIN
|................................................ FT
|........................................ PEAK
|................................................ FT
|........................................ FT
|................................ PLOT
|................................ READS
|................................ FT
|................................ GOTOER
|................................ AXIS
|................................ SYMBOL
|................................ NUMBER
|........................ DIGITR
|................................ GOTOER
|................................ READS
|................................ PROPGAT
|........................................ RIN
|................................................ GOTOER
|................................ PLOT
|................................ READS
|................................ AXIS
|................................ SYMBOL
|................................ NUMBER
|........................ PLOTFIN
|................................ PLOT
|................................ DASHER
|................................ TITLES
|................................ SYMBOL
|................................ AXIS
|................................ INITT
|................................ VWINDO
|................................ SWINDO
|................................ MOVEA
|........................................ STACKG
|................................................ PUTID
|................................................ EQID
|................................................ ERROR
|................................................ WCOPY
|........................................ STACKP
|................................................ ERROR
|................................................ FUNS
|........................................................ PRNTID
|........................................................ EQID
|................................................ PUTID
|................................................ EQID
```

```
|............................................... WCOPY
|............................................... WSET
|............................................... PRINT
|...................................................... ROUND
|...................................................... FILES
|...................................................... PRNTID
|...................................................... FORMZ
|................................. DRAWA
|................................. MOVABS
|................................. CHRSIZ
|................................. ANMODE
|................................. BELL
|................... POLY
|.......................... GOTOER
|.......................... ALPHA
|.......................... SWAP
|................................. GOTOER
|................................. READS
|................................. XFER
|................................. EXPAND
|........................................ POLYMLT
|.......................... ECHOS
|.......................... SWAPER
|................................. GOTOER
|................................. XFER
|................................. ALPHA
|................................. SWAP
|................................. ECHOS
|.......................... POLYADD
|.......................... POLYSUB
|................................. POLYADD
|.......................... POLYMLT
|.......................... READS
|.......................... SWAP
|.......................... EXPAND
|.......................... XFER
|................... ROOT10
|.......................... ADAPT
|................................. GOTOER
|................................. READS
|.......................... TITLES
|.......................... ROOT
|................................. ZRPOLY
|.......................... BLEND
|.......................... PLOT
|.......................... ROOT11
|................................. SYMBOL
|................................. PLOT
|................................. NUMBER
|................................. DNTER
```

```
|................................. CNTER
|................................. GANG1
|........................................ BANG
|........................................ BOX
|................................. ANG1
|........................................ BANG
|........................................ BOX
|................................. GROUP
|................................. UCIRC
|........................................ PLOT
|.......................... ROOTS
|................................. SMULR
|................................. SYMBOL
|.......................... ROOT12
|................................. GOTOER
|................................. ANG1
|................................. BANG
|................................. BOX
|................................. GANG1
|................................. SYMBOL
|................................. NUMBER
|................................. PLOT
|................................. BREAK
|................................. SEEK
|........................................ GANG1
|........................................ GOTOER
|........................................ ANG1
|........................................ BANG
|........................................ BOX
|.................... FREQR
|.......................... FREQOUT
|................................. GOTOER
|................................. READS
|................................. FW
|................................. AW
|................................. PLOT
|................................. AXIS
|................................. PLOT
|................................. BOXER
|.......................... PLOTSET
|................................. PLOT
|................................. DASHER
|........................................ PLOT
|................................. TITLES
|................................. SYMBOL
|................................. NUMBER
|................................. AXIS
|.......................... FREPLOT
|.......................... NICHOLS
|................................. PLOT
```

```
|.................................. DASHER
|.................................. AXIS
|.................................. TITLES
|.................................. SYMBOL
|.................................. NUMBER
|........................... TEKFREQ
|.................................. INITT
|.................................. VWINDO
|.................................. SWINDO
|.................................. MOVABS
|.................................. DRWABS
|.................................. MOVEA
|.................................. DRAWA
|.................................. CHRSIZ
|.................................. ANMODE
|.................................. BELL
|........................... CHALK
|.................................. READS
|.................................. AW
|.................................. FW
|.................... READER
|.................... DECODER
|........................... GOTOER
|........................... PLOTND
|........................... CLOSMS
|........................... PLOT
|........................... MODIFY
|.................................. CONVERT
|......................................... LISTER
|.................................. GOTOER
|.................................. UPDA
|.................................. NODV
|.................................. FACTOR
|......................................... ERROR
|................................................ BELL
|......................................... GETSYM
|................................................ GETCH
|................................................ GETVAL
|....................................................... GETCH
|................................................ FLOP
|................................................ PRNTID
|....................................................... NOREVE
|......................................... GETCH
|......................................... STACK1
|................................................ WRSCAL
|....................................................... FLOP
|................................................ ERROR
|................................................ WCOPY
|......................................... GETLIN
|................................................ XCHAR
```

```
|................................................ GETCH
|................................................ PUTID
|................................................ FILES
|................................................ EDIT
|......................................... PUTID
|......................................... FUNS
|......................................... STACKG
|......................................... STACK2
|................................................ ERROR
|................................................ WAXPY
|....................................................... FLOP
|................................................ MPOLY
|....................................................... WPOLY
|................................................ WCOPY
|................................................ WDOTUR
|....................................................... FLOP
|................................................ WDOTUI
|....................................................... FLOP
|................................................ WSCAL
|....................................................... WMUL
|.............................................................. FLOP
|................................................ WDIV
|....................................................... ERROR
|....................................................... FLOP
|................................................ WRSCAL
|................................................ WMUL
|.................................. CONVERT
|.................................. NODI
|.................................. ONEDV
|......................................... POLECHO
|......................................... CONVERT
|......................................... FACTO
|................................................ ROOT
|.................................. CPLXV
|......................................... RTECHO
|......................................... CONVERT
|......................................... EXPAND
|......................................... DELETE
|................................................ EXPAND
|.................................. TWODV
|......................................... MATECHO
|......................................... CONVERT
|..................... READER
|..................... HELP
|............................ GOTOER
|..................... FRACTOR
|............................ FACTO
|..................... TTYPLOT
|..................... MISCELL
|............................ ERASE
```

```
|........................... HDCOPY
|........................... MSQINT
|.................................. DET
|.................... MATRIX
|........................... MATS
|.................................. GOTOER
|.................................. MATIN
|......................................... GOTOER
|......................................... READS
|......................................... MATECHO
|.................................. READS
|.................................. MATZERO
|.................................. IDENITY
|......................................... MATZERO
|.................................. TEST
|........................... MATOPR
|.................................. GOTOER
|.................................. PHOFS
|......................................... MMPY
|.................................. FACTO
|.................................. ECHOS
|.................................. MADD
|.................................. MATECHO
|.................................. GENMMPY
|.................................. MINV
|.................................. TRANPOS
|.................................. GTFHTF
|......................................... PHOFS
|......................................... CADJB
|......................................... FACTO
|......................................... CANCEL
|........................... MOREMAT
|.................................. GOTOER
|.................................. GENMMPY
|.................................. MADD
|.................................. MATZERO
|.................... COPYIER
|........................... TFECHO
|.................................. XFER
|.................................. CLEAR1
|........................... MIX
|.................................. GOTOER
|.................................. TRANFER
|......................................... GOTOER
|......................................... XFER
|........................... MATMIX
|.................................. GOTOER
|.................................. MATRAN
|......................................... GOTOER
|......................................... XMAT
```

```
|........................... WRITMS
|........................... READMS
|........................... MATECHO
|.................... ZFORMS
|........................... GOTOER
|........................... READS
|........................... POLAR
|........................... UNPARTL
|.................................. CXPAND
|........................... FACTO
|........................... EXPAND
|........................... BIFORM
|.................................. BITERM
|......................................... EXPAND
|......................................... DBLMULT
|.................................. FACTO
|........................... MMPY
|.................... BLOCKER
|........................... GOTOER
|........................... POLYMLT
|........................... FACTO
|........................... CANCEL
|.................................. EXPAND
|........................... POLYADD
|........................... POLYSUB
|.................... NOTICE
|........................... GONOGO
|.................... SIMULAT
|........................... GOTOER
|........................... READS
|........................... UVECTOR
|........................... YVECTOR
|.................................. MMPY
|........................... XVECTOR
|.................................. UVECTOR
|.................................. MMPY
|........................... PLOT
|........................... DATFILL
|.................................. UVECTOR
|.................................. YVECTOR
|.................................. XVECTOR
|........................... SCALER
|.................................. GOTOER
|........................... AXIS
|........................... SYMBOL
|........................... NUMBER
|........................... PLOT
|........................... DEFU
|.................................. GOTOER
|.................................. READS
```

```
|...................... ADVANZ
|.............................. GOTOER
|.............................. READS
|...................... TXCONV
|.............................. GOTOER
|.............................. READS
|.............................. COMPOLY
|.............................. BILIN
|...................................... TERM1
|.............................................. COMPOLY
|.............................................. MULTIP
|.............................................. COEFF
|...................................................... SIMPLE
|...................................................... ORDER3
|...................................... ROTPOLY
|.............................................. GOTOER
|...................................... SZWROOT
|.............................................. GOTOER
|.............................. ORDER3
|.............................. MULTIP
|...................................... SIMPLE
|.............................. ORDPOLE
|.............................. ZMULT1
|...................................... POLE
|...................................... EVALU3
|.............................................. ORDER3
|...................................... DERIV3
|...................................... DIVI
|.............................. WMULT1
|...................................... WPOLE
|.............................................. DIVI
|.............................................. MULT
|...................................... EVALU3
|...................................... DERIV3
|...................................... DIVI
|.............................. WMULT2
|...................................... WPOLE
|...................................... EVALU3
|...................................... DERIV3
|...................................... DIVI
|...................................... MULT
|.............................. WMULT3
|...................................... WPOLE
|...................................... EVALU3
|...................................... DERIV3
|...................................... DIVI
|...................................... MULT
|.............................. RES1
|...................................... MULTIP
|...................................... ADD
```

```
|.................................... SIMPLE
|............................. ORDER3
|............................. SIMPLE
|............................. ROTPOLY
|............................. COMPOLY
|...................... WZBILIN
|............................. ORDER3
|............................. COEFF
|............................. BILIN
|...................... RES2
|............................. MULT
|............................. MULTIP
|............................. ADD
|............................. ORDER3
|............................. SIMPLE
|............................. ROTPOLY
|...................... RES3
|............................. MULT
|............................. MULTIP
|............................. ADD
|............................. ORDER3
|............................. SIMPLE
|............................. ORDER3
|............................. ROTPOLY
|...................... MULTIP
|...................... DOLOOP
|...................... ADD
|...................... ORDER3
|...................... COEFF
|...................... ROTPOLY
|...................... CANROOT
|...................... COMPOLY
|........... PAUSE
|........... COPY
|................. CLEAR
|................. PACK_BUFFER
|................. TOTICE
|................. PAUSE
|................. HIGHLIGHT
|................. NOHIGHLIGHT
|........... DEFINE
|................. DEFINE_TF
|...................... CLEAR
|...................... HIGHTLIGHT
|...................... NOHIGHLIGHT
|...................... DEF_TF_PLANE
|...................... PAUSE
|...................... TRIM
|...................... PRINT_BUFFER
|............................. TRIM
```

```
|..................... DEFINE_GAIN
|............................ READLN
|............................ PACK_BUFFER
|............................ TOTICE
|............................ PAUSE
|..................... CLEAR
|..................... SETDEF
|..................... MATICE
|............................ PARSE
|................................... FILES
|................................... PROMPT
|.......................................... NOREVE
|................................... GETLIN
|................................... PUTID
|................................... GETSYM
|................................... COMAND
|.......................................... EQID
|.......................................... ERROR
|.......................................... GETSYM
|.......................................... STACKP
|.......................................... FILES
|.......................................... PRNTID
|.......................................... REVERS
|.......................................... FUNS
|.......................................... URAND
|.......................................... CLEAR1
|.......................................... BELL
|.......................................... GETLIN
|................................... FUNS
|................................... ERROR
|................................... PUTID
|................................... EQID
|................................... STACKP
|................................... CLAUSE
|.......................................... GETSYM
|.......................................... ERROR
|.......................................... PUTID
|.......................................... WCOPY
|.......................................... STACKP
|.......................................... EQID
|................................... EXPR
|.......................................... PUTID
|.......................................... ERROR
|.......................................... STACK1
|.......................................... GETSYM
|.......................................... STACK2
|.......................................... GETSYM
|................................... TERM
|.......................................... GETSYM
|.......................................... STACK2
```

```
|......................................... ERROR
|.................................. FACTOR
|........................... MATFN1
|.................................. ERROR
|.................................. WGECO
|......................................... WGEFA
|................................................ IWAMAX
|................................................ WDIV
|................................................ WSCAL
|................................................ WAXPY
|......................................... WSIGN
|................................................ PYTHAG
|................................................ WMUL
|......................................... WRSCAL
|......................................... WDIV
|......................................... WMUL
|......................................... FLOP
|......................................... WAXPY
|......................................... WASUM
|................................................ FLOP
|......................................... WDOTCR
|................................................ FLOP
|......................................... WDOTCI
|................................................ FLOP
|.................................. WGESL
|......................................... WAXPY
|......................................... WDIV
|......................................... WDOTCR
|......................................... WDOTCI
|.................................. WASUM
|.................................. RSET
|.................................. WCOPY
|.................................. WGEDI
|......................................... WMUL
|......................................... WDIV
|......................................... WSCAL
|......................................... WAXPY
|......................................... WSWAP
|.................................. WSCAL
|.................................. WGEFA
|.................................. WSWAP
|.................................. HILBER
|.................................. WPOFA
|......................................... WDOTCR
|......................................... WDOTCI
|......................................... WDIV
|.................................. WSET
|.................................. RREF
|......................................... WASUM
|......................................... IWAMAX
```

```
|....................................... WSET
|....................................... WSWAP
|....................................... WDIV
|....................................... WSCAL
|....................................... WAXPY
|.......................... MATFN2
|................................. ERROR
|................................. WCOPY
|................................. WSET
|................................. HTRIDI
|....................................... FLOP
|....................................... PYTHAG
|................................. IMTQL2
|....................................... FLOP
|................................. HTRIBK
|....................................... FLOP
|................................. CORTH
|....................................... FLOP
|....................................... PYTHAG
|................................. COMQR3
|....................................... FLOP
|....................................... PYTHAG
|....................................... WSQRT
|............................................. PYTHAG
|............................................. FLOP
|....................................... WDIV
|................................. WLOG
|....................................... PYTHAG
|....................................... ERROR
|................................. WMUL
|................................. WATAN
|....................................... ERROR
|....................................... WDIV
|....................................... WLOG
|................................. WSQRT
|................................. WLOG
|................................. PYTHAG
|................................. ROUND
|................................. FLOP
|................................. WSCAL
|................................. WAXPY
|................................. WDIV
|................................. COMQR3
|.......................... MATFN3
|................................. ERROR
|................................. WSVDC
|....................................... WNRM2
|............................................. PYTHAG
|....................................... WSIGN
|....................................... WDIV
```

```
|...................................... WSCAL
|...................................... FLOP
|...................................... WDTOCR
|...................................... WDOTCI
|...................................... WAXPY
|...................................... WRSCAL
|...................................... PYTHAG
|...................................... WMUL
|...................................... RROTG
|............................................ FLOP
|............................................ PYTHAG
|...................................... RROT
|............................................ FLOP
|...................................... WSWAP
|................................ FLOP
|................................ IWAMAX
|................................ PYTHAG
|................................ WASUM
|................................ WNRM2
|................................ WSVDC
|................................ WCOPY
|................................ WRSCAL
|................................ WDOTCR
|................................ WDOTCI
|.......................... MATFN4
|................................ ERROR
|................................ STACK1
|...................................... WRSCAL
|...................................... ERROR
|...................................... WCOPY
|................................ WCOPY
|................................ WSET
|................................ WQRDC
|...................................... WSWAP
|...................................... WNRM2
|...................................... WSWAP
|...................................... WSIGN
|...................................... WDIV
|...................................... WSCAL
|...................................... FLOP
|...................................... WDOTCR
|...................................... WDOTCI
|...................................... WAXPY
|...................................... PYTHAG
|................................ FLOP
|................................ WQRSL
|...................................... WDIV
|...................................... WCOPY
|...................................... WDOTCR
|...................................... WDOTCI
```

```
|.......................................... WAXPY
|.................................. WSET
|........................... MATFN5
|.................................. ERROR
|.................................. FILES
|.................................. PRINT
|.................................. PUTID
|.................................. WASUM
|.................................. SAVLOD
|.................................. RSET
|.................................. STACKP
|.................................. RAT
|.......................................... ROUND
|.................................. FLOP
|.................................. BASE
|.................................. WCOPY
|.................................. STACK1
|.................................. PLOT
|........................... MATFN6
|.................................. ERROR
|.................................. WCOPY
|.................................. WMUL
|.................................. WDIV
|.................................. FLOP
|.................................. USER
|.................................. RSET
|.................................. MAGIC
|.......................................... RSWAP
|.................................. EQID
|.................................. URAND
|........................... NOREVE
|.................... RESETD
|.................... DEFINE_TF
|.................... PAUSE
|.................... DEFINE_PROMPT
|........................... CLEAR
|........................... HIGHLIGHT
|........................... NOHIGHLIGHT
|.................... TRIM
|.................... PAUSE
|.................... TOTICE
|.............. DISPLAY_OR_PRIN
|.................... CLEAR
|.................... PACK_BUFFER
|.................... TOTICE
|.................... PAUSE
|.................... LOCUS
|........................... LOCUS_AUTOSCALE
|.................................. FIND_BORDERS
|.................................. CLEAR
```

```
|................................... TOTICE
|............................ LOCUS_MAGNIFY
|................................... MAGNIFY
|................................... CLEAR
|............................ LOCUS_SHRINK
|................................... SHRINK
|................................... CLEAR
|............................ LOCUS_ZOOM
|................................... TOTICE
|................................... ZOOM_DATA
|.......................................... READS
|................................... TOTICE
|............................ CLEAR
|............................ HIGHLIGHT
|............................ NOHIGHLIGHT
|............................ TRIM
|............................ PAUSE
|..................... DISPLAY_TF
|............................ DISPLY2
|................................... XFER
|................................... CLEAR1
|............................ DISPLY
|................................... XFER
|................................... CLEAR1
|..................... SETDIS
|..................... MATICE
|..................... RESDIS
|..................... TRFF
|............................ GETLIN
|............................ GETSYM
|............................ PUTID
|............................ STACKG
|............................ ERROR
|............................ MCOPY
|............................ MATFN2
|............................ DCOPY
|............................ VCOPY
|............................ NUM
|............................ CLEAR1
|............................ COPYIER
|..................... CLEAR
|..................... MRIC
|............................ GETLIN
|............................ GETSYM
|............................ PUTID
|............................ STACKG
|............................ ERROR
|............................ MATFN1
|............................ STACK2
|............................ STACK1
```

```
|.............................. STACKP
|.............................. CLEAR1
|.............................. FORMR
|.................................... STACKG
|.................................... STACK1
|.................................... PUTID
|.................................... STACKP
|.............................. MATFN2
|.............................. SORT
|.............................. STACKP
|.............................. RPOLY
|.................................... QROOT
|.................................... DIAGON
|.................................... STACK1
|.................................... MATFN2
|.................................... PUTID
|.................................... STACKP
|.............................. CPOLY
|.................................... QROOT
|.................................... CMULT
|.................................... PRINT
|.................................... WPOLY
|.......................................... CROSS
|.................................... PUTID
|.................................... STACKP
|.............................. MIXPOL
|.................................... CPOLY
|.................................... STACKG
|.................................... RPOLY
|.................................... WPOLY
|.................................... PUTID
|.................................... STACKP
|.............................. NEST
|.................................... STACKG
|.................................... STACK2
|.............................. CUT
|.................................... PUTID
|.................................... STACKP
|.............................. ANSWER1
|.................................... STACKG
|.................................... MATFN1
|.................................... STACK2
|.............................. ANSWER2
|.................................... STACKG
|.................................... MATFN1
|.................................... STACK2
|.............................. PRINT
|.............................. QSAVE
|.................................... GETLIN
|.................................... GETSYM
```

```
|................................... PUTID
|................................... STACKG
|................................... STACKP
|............................ POSITIVE
|................................... STACKG
|................................... MATFN2
|............................ OPTIMAL
|................................... STACKG
|................................... MATFN1
|................................... STACK1
|................................... STACK2
|................................... PUTID
|................................... STACKP
|................................... PRINT
|............................ DESTOY
|................................... STACKP
|..................... CLEAR
|..................... MODERN
|............................ GETLIN
|............................ GETSYM
|............................ PUTID
|............................ STACKG
|............................ ERROR
|............................ MATFN2
|............................ DCOPY
|............................ VCOPY
|............................ TRFORM
|................................... STACKG
|................................... PUTID
|................................... STACKP
|................................... STACK2
|................................... STACKP
|............................ VCOPY
|............................ NUM
|................................... MATMUL
|................................... MATVEC
|................................... VECPRD
|............................ CLEAR1
|............................ GETLIN
|............................ GETSYM
|............................ PUTID
|............................ ERROR
|............................ STACKG
|............................ MATFN1
|............................ STACK2
|............................ PRINT
|............................ STACKP
|............................ QSAVE
|..................... CLEAR
|..................... HIGHLIGHT
```

```
|..................... NOHIGHLIGHT
|..................... PAUSE
|..................... TOTICE
|..................... PRINT_BUFFER
|.............. DFORM
|..................... TOTICE
|..................... PAUSE
|..................... CLEAR
|..................... HIGHLIGHT
|..................... NOHIGHLIGHT
|..................... BOXIT
|..................... GRAPHICS
|..................... CURSORRC
|..................... NOGRAPHICS
|..................... PRINT_BUFFER
|.............. DHELP
|..................... CLEAR
|..................... SETHEP
|..................... HELMAT
|............................ REVERS
|............................ FUNS
|............................ PRNTID
|............................ NOREVE
|............................ GETLIN
|............................ GETSYM
|............................ FILES
|............................ BELL
|..................... RESHEP
|..................... PAUSE
|..................... HIGHLIGHT
|..................... PRINT_BUFFER
|..................... NOHIGHLIGHT
|..................... TRIM
|.............. DISPLAY_PRIN
|.............. TOTICE
|.............. PAUSE
|.............. TFORM
|..................... TFORM_TF
|............................ TFORM_TF_TYP
|................................... CLEAR
|................................... HIGHLIGHT
|................................... NOHIGHLIGHT
|................................... DISCRETE
|.......................................... GTFCTF
|................................................. XFER
|.......................................... IMPUL
|................................................. READS
|................................................. ZOH
|........................................................ DBLMULT
|........................................................ FACTO
```

```
|............................................... ZOH1
|...................................................... DBLMULT
|...................................................... FACTO
|............................................... POLAR
|............................................... UNPARTL
|............................................... FACTO
|............................................... EXPAND
|............................................... ZOHD
|...................................................... DBLMULT
|...................................................... FACTO
|........................................ CTFGTF
|............................................... XFER
|........................................ PAUSE
|........................................ OLTCTF
|............................................... XFER
|........................................ CTFOLT
|............................................... XFER
|........................................ HTFCTF
|............................................... XFER
|........................................ CTFHTF
|...............................................XFER
|........................................ TUSTIN
|............................................... READS
|............................................... BIFORM
|........................................ BAKDIF
|............................................... CLEAR1
|............................................... READS
|............................................... BIFORM
|........................................ INVIMP
|............................................... READS
|............................................... POLAR
|............................................... UNPARTL
|............................................... FACTO
|............................................... EXPAND
|........................................ INVTUS
|............................................... READS
|............................................... BIFORM
|........................................ INVDIF
|............................................... CLEAR1
|............................................... READS
|............................................... BIFORM2
|...................................................... BITERM1
|............................................................. EXPAND
|...................................................... FACTO
|........................................ STOWP
|............................................... READS
|............................................... ZOH
|............................................... ZOH1
|............................................... POLAR
|............................................... UNPARTL
```

```
|....................................................... FACTO
|....................................................... EXPAND
|....................................................... ZOHD
|....................................................... BIFORM
|............................................... WPTOS
|....................................................... READS
|....................................................... BIFORM
|....................................................... POLAR
|....................................................... UNPARTL
|....................................................... FACTO
|....................................................... EXPAND
|............................................... STOW
|....................................................... READS
|....................................................... ZOH
|....................................................... ZOH1
|....................................................... POLAR
|....................................................... UNPARTL
|....................................................... FACTO
|....................................................... EXPAND
|....................................................... ZOHD
|....................................................... BIFORM
|............................................... WTOS
|....................................................... READS
|....................................................... BIFORM
|....................................................... POLAR
|....................................................... UNPARTL
|....................................................... FACTO
|....................................................... EXPAND
|............................................... WPTOZ
|....................................................... READS
|....................................................... BIFORM
|............................................... ZTOWP
|....................................................... READS
|....................................................... BIFORM
|............................................... WTOZ
|....................................................... BIFORM
|............................................... ZTOW
|....................................................... BIFORM
|............................................... HIGHLIGHT
|............................................... NOHIGHLIGHT
|....................................... PAUSE
|....................................... TRIM
|....................................... PRINT_BUFFER
|....................................... PAUSE
|............................... TFORM_TF_TYP
|............................... PAUSE
|............................... CLEAR
|............................... HIGHLIGHT
|............................... NOHIGHLIGHT
|............................... TRIM
```

```
|...................... TFORM_TF
|...................... PAUSE
|...................... TFORM_PROMPT
|............................. CLEAR
|............................. HIGHLIGHT
|............................. NOHIGHLIGHT
|...................... TRIM
|...................... PAUSE
|.............. CHANGE
|...................... PLANCHG
|............................. CLEAR1
|...................... PAUSE
|...................... SAMPCHG
|............................. READS
|...................... PAUSE
|...................... CHGCONS
|............................. READS
|............................. PARTS
|............................. EXPAND
|...................... CHANGE_PROMP
|............................. CLEAR
|............................. HIGHLIGHT
|............................. NOHIGHLIGHT
|............................. RETURN
|...................... TRIM
|.............. INSERT
|...................... INSERT_RT
|............................. ALTER
|.................................... READS
|.................................... INSERT
|............................. PAUSE
|............................. CLEAR
|............................. HIGHLIGHT
|............................. NOHIGHLIGHT
|............................. TRIM
|............................. PRINT_BUFFER
|...................... PAUSE
|...................... TRIM
|...................... PRINT_BUFFER
|.............. DELETE
|...................... DELETE_RT
|............................. DELETER(ABE)
|.................................... READS
|.................................... DELETE
|............................. PAUSE
|............................. CLEAR
|............................. HIGHLIGHT
|............................. NOHIGHLIGHT
|............................. TRIM
|............................. PRINT_BUFFER
```

```
|............................ PAUSE
|..................... PAUSE
|..................... TRIM
|..................... PRINT_BUFFER
|.............. PACK_BUFFER
|.............. TOTICE
|.............. TURN
|..................... TURN_X
|............................ CLEAR
|............................ PACK_BUFFER
|............................ TOTICE
|............................ PAUSE
|............................ HIGHLIGHT
|............................ NOHIGHLIGHT
|............................ TRIM
|..................... PAUSE
|..................... TURN_PROMPT
|............................ CLEAR
|............................ HIGHLIGHT
|............................ NOHIGHLIGHT
|..................... TRIM
|.............. TOTICE
|.............. PAUSE
|.............. HELP_SYSTEM
|..................... CLEAR
|..................... HIGHLIGHT
|..................... PAUSE1
|............................ HIGHLIGHT
|............................ NOHIGHLIGHT
|............................ RESET
|............................ READLN
|............................ CLEAR
|..................... NOHIGHLIGHT
|.............. TRIM
|.............. PAUSE
|...... HIGHLIGHT
|...... NOHIGHLIGHT
```

C.4 SUMMARY

The structure charts for ICECAP have been presented. It is hoped that follow-on efforts will add to and update this appendix so that a current structure chart is available for designers as well as customers of ICECAP.

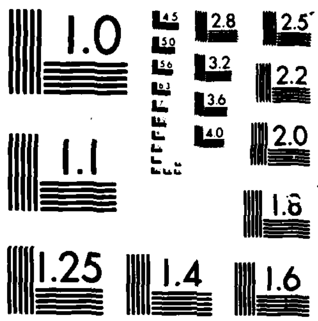
AD-A163 952 INTERACTIVE COMPUTER GRAPHICS FOR ANALYSIS AND DESIGN OF CONTROL SYSTEMS(U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING J R BULLARD DEC 85 AFIT/GE/EE/85D-5 3/3

UNCLASSIFIED F/G 9/2 NL

END

FILMED

DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

APPENDIX D

COMPUTER AIDED DESIGN PACKAGES for CONTROL

D.1 Introduction

This appendix provides a synopsis of representative computer-aided design programs studied during the ICECAP project. Programs cited in prior efforts (2, 4, 24) are listed for completeness.

D.2. List of Programs Studied

The following programs were studied during prior efforts (references appear to the right of each entry).

ADAPT (2,4)
BLZ (2,4)
BUCOP (6)
BYPASS (2,4)
CACHE
CADS (2,4)
CALICO (2,4)
CC (33)
CESA (2,4)
CTRL-C (24)
DELIGHT-MIMO (4)
DIGIKON (4)
FORTRAC (4)
FREDOM and TIMDOM (24)
HONEY-X (4)
INTERAC (2)
I-G SPICE (4)
KEDDC (24)
LPASS (2,4)
LSAP (2,4)
LSD (24)
MATRIXx (4)
SOFE (4)
SUPER-SCEPTRE (4)
TOTAL (2,4)
VAXTOTAL (4)
WINDOW (2)

PROGRAM NAME: ADAPT -- Recursive Digital Filters (Kalman Filters)

DESCRIPTION: Reads desired filter parameters (SIGMA, M and Q), generates an initial S (covariance) matrix, T matrix, and W (weight) matrix. The S and W matrices are used to initialize the S and W matrices, respectively. Next, 101 sample points provided by the user are read as input to the Kalman filter. ADAPT tabulates the sample number filter input, filter output, error signal, and Kalman gain.

LANGUAGE: DEC FORTRAN

HOST COMPUTER: PDP-11/20

REFERENCE:

Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

PROGRAM NAME: BLZ -- Bilinear Z-transform

DESCRIPTION: An interactive digital filter design program that calculates digital transfer function coefficients and magnitude function, applies a bilinear transformation with pre-warping to obtain realizable stable digital filters. Consists of a main program and 4 subroutines for pre-warping.

LANGUAGE: DEC FORTRAN

HOST COMPUTER: PDP-11/20

REFERENCE:

Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

PROGRAM NAME: BPASS -- Band Pass Filter Design

DESCRIPTION: Designs maximally flat Butterworth or Chebychev filter with equal ripple in pass band (band pass or band stop). Generates digital filter coefficients for up to six second order sections in cascade (12th order). Must be called as a subroutine from the main program.

LANGUAGE: FORTRAN IV

HOST COMPUTER: PDP-11/20

REFERENCE:

Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

PROGRAM NAME: BUCOP -- Bucknell University Control Package

DESCRIPTION: BUCOP is a menu driven computer-aided design package and is capable of performing several linear control systems operations which include:

1. State variable representation and transfer function transformation
2. Linear system simulation (time response)
3. Frequency response
4. Polynominal root finding
5. Root locus
6. Provides printouts and plots of the above

BUCOP stores both the user input as well as the results of the computation into files. There are four files : state variable file, transfer file, frequency file, and a root locus file. The user supplies the names of these files. This allows the user to store values of the transfer function into different files so that multiple files can be evaluated and compared. The use of files is also useful since BUCOP is designed to operate on a multi-window computer system such as the Apollo.

LANGUAGE: FORTRAN 77

HOST COMPUTER:

REFERENCE:

Aburdene, Maurice F. and Sheri Surchek BUCOP: A Computer-Aided Design Control Systems Package, Interactive Computer-Aided Circuit Design, IEEE 1984, pages 29-36.

PROGRAM NAME: CACHE

DESCRIPTION: CACHE is a computer-aided analysis tool that was developed at Carnegie-Mellon University (CMU) to help solidify the concepts discussed and developed in the control classes. CACHE is customed designed on a Hewlett-Packard personal computer. The menu-driven program allows the users to perform the following: input transfer functions either in polynominal or factored form or in matrix form, block diagram manipulation, pole-zero diagrams, root-locus plots, frequency response, phase-magnitude plots on a Nichols chart, time response plots, linear state-variable feedback, and linear-quadratic optimal regulator design.

LANUAGE: PASCAL

HOST COMPUTER: Helett-Packard HP9836 Personal CAD Stations

REFERENCE

Mason, Mark J., Charles P. Neuman, Bruce H. Krogh. "CACHE: An Interactive Control System Analysis and Design Package", IEEE Transactions on Education, Vol. E-28, No. 3, Aug 1985.

PROGRAM NAME: CADS -- Computer Automated Design of Systems

DESCRIPTION: Simulates and optimizes control systems and circuits. Control system is defined in block diagram form (transfer functions). The transfer functions are reduced to first order differential equations. The unknown or adjustable parameters are set by a minimization routine to acheive the desired reponse. Batch (cards) input.

LANGUAGE: FORTRAN IV

HOST COMPUTER:

REFERENCE:

Vines, Larry Paul. Computer Aided Design of Systems. Monterey, California: Naval Postgraduate School, June 1975.

PROGRAM NAME: CALICO -- Computer Aided Linear Time-Invariant Compensator Optimization Program

DESCRIPTION: For design of compensators to acheive desired response in accordance with selected cost function. Batch (cards) input. Four major parts including subroutines -- 180 K words (210 K with plotting routines)

LANGUAGE: FORTRAN IV

HOST COMPUTER: IBM 360-67

REFERENCE:

Mancini, Anthony J. Computer Aided Control System Design Using Frequency Domain Specifications. Monterey, California: Naval Postgraduate School, June 1976.

PROGRAM NAME: CC -- Classical Control

DESCRIPTION: CC is a menu-driven computer-aided design contr[Col system design and analysis package for classical, sampled-data, state space and optimal control systems. CC was developed by the California Institute of Technology for use in system and control classes, both in under graduate and graduate work. CC provides the following plots: Bode, Nyquist, inverse Nyquist, Nichols, Time, and root locus. Hardcopies of the plots are available through the screen dump programs to a dot matrix printer.

LANGUAGE: Micro-soft Basic

HOST COMPUTER: IBM-PC, Zenith-100 PC

REFERENCE:

Thompson, Peter M. User's Guide to Program CC, Version 3 Systems Technology, Inc., March 1985.

PROGRAM NAME: CESA -- Complete Eigenstructure Assignment Program

DESCRIPTION: An interactive program to design a state space control law for multi-input, multi-output systems. Includes regulator, disturbance rejector, and tracker design capabilities.

LANGUAGE: FORTRAN IV

HOST COMPUTER: CDC CYBER

REFERENCES:

Kennedy, Thomas A. The Design of Digital Controllers for the C-141 Aircraft Using Entire Eigenstructure Assignment and the Development of an Inter-Active Computer Design Program. MS Thesis. Wright-Patterson Air Force Base,Ohio: Air Force Institute of Technology, March 1979.

PROGRAM NAME: CTRL-C

DESCRIPTION: CTRL-C provides the control systems engineer with a conversational environmen for the analysis and design of multivariable systems. CTRL-C is an interpretive language that has evolved from MATLAB. Powerful matrix handling abilities normally encountered only in APl are provided without cumbersome syntax. UNIX-like file handling commands provide access to data files. Graphics are created with the same natural commands used to manipulate matrices. The standard matrix primitive like QR, QZ, SVD, and inversion, are augmented with matrix primitives for control systems engineering. These include the solution of LQG problems, computation of multivariable zeros, system decomposition, transfer function determination, and eigenstructure assignment. Vectors are used to represent arbitrary sampled-data signals. A family of primitives calculate the multivariable time and frequency domain measures required for control analysis. Primitives for filtering, FFT, analysis, and other digital signal processing calculations become very natural using complex vector manipulation concepts. Custom control design techniques are created by writing short procedures in CTRL-C language and defining them as new primitives.

REFERENCE: Abbas Emani-Naeini
John N. Little
Steve N. Bangert
Systems Control Technology, Inc.
Advance Technology Division
1801 Page Mill Road
Palo Alto, California 94303

PROGRAM NAME: DELIGHT-MIMo (in development)

DESCRIPTION: A highly interactive system for optimization based design of multivariable control systems. Uses color graphics and graphics tablet system interconnections. Employs highly sophisticated semi-infinite optimization algortihms.

LANGUAGE: FORTRAN 77

HOST COMPUTER: VAX 11/780

REFERENCE:

Polak, E. "Interactive Software for Computer-Aided-Design of Control Systems via Optimization," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

PROGRAM NAME: DIGIKON

DESCRIPTION: Batch and interactive packages for analysis of single-input/single-output control systems. Intended mainly for industrial use. Used for multi-rate digital design. Does root locus, eigenvalue and eigenvector analyses. Packages are designed for both continuous and discrete systems.

LANGUAGE: FORTRAN IV

HOST COMPUTER: IBM 370/370, CDC 6600, Honeywell 66

REFERENCE:

Harvey, C. A. and J. E. Wall. "Phases in the Development of Control System Design Software," Proceedings of the 20th IEEE Conference on Decision and Control, December 1981.

PROGRAM NAME: FORTRAC

DESCRIPTION: For the design of multivariable digital control systems. Can design a discrete control law, can design an observer, and can run a simulation of the system for the resulting controller. Takes the continuous time description of a linear system and synthesizes a control law for discrete-time-optimal regulators, disturbance rejectors, and trackers.

LANGUAGE: FORTRAN

HOST COMPUTER: CDC 6600/CYBER-74

REFERENCE:

Colgate, James A. INTERAC - An Interactive Software Package for Direct Digital Control Design. MS Thesis. Wright-Patterson Air Force Base, Ohio; Air Force Institute of Technology, December 1977.

PROGRAM NAME: FREDOM and TIMDOM

DESCRIPTION: FREDOM and TIMDOM are two linked interactive computer packages written to aid in the numerical analysis and design of linear control systems. FREDOM deals with the classical frequency-domain techniques while TIMDOM handles the modern time-domain techniques. The packages are linked together to provide the user with the ability to switch domains as necessary. The packages are written in the extended BASIC used by the HP9845 series desktop computers. Some capabilities available in FREDOM include root locus plotting, feedback compensation, and parameter optimization of SISO system design. For analysis purposes, root locus as well as Routh-Hurwitz and Jury-Blanchard stability criteria are available with graphic output. Bode Nyquist diagrams can be plotted. Model Reduction techniques for SISO and MIMO systems are available. In TIMDOM, major topics of existing CAd programs are pole placement, model reduction (exact and modal aggregation, and minimum realization), filtering, estimation and numerical solutions of two point boundary-value problems. The techniques will handle continuous-time systems and are applicable in many cases to discrete-time systems. In addition, a group of input-output and support routines are available. These handle file-manipulation, plotting and other graphics output, linear algebra operations, and Laplace and Z-transform inversion.

REFERENCE : R. Morel and M. Jamshidi
Department of Electrical
and Computer Engineering
University of New Mexico
Albuquerque, New Mexico 87131

PROGRAM NAME: HONEY-X

DESCRIPTION: Interactive package for control system analysis and design intended for research and development applications. Handles multiple-input/multiple-output systems. Does matrix manipulation and Nichols and Nyquist analyses. Finds the time history response of a control system. Handles Kalman filtering and optimal control.

LANGUAGE; FORTRAN 77

HOST COMPUTER: Honeywell 66 (under MULTICS)

REFERENCE:

Harvey, C. A. and J. E. Wall. "Phases in the Development of Control System Design Software," Proceedings of the 20th IEEE Conference of Decision and Control, December 1981.

PROGRAM NAME: I-G SPICE

DESCRIPTION: Interactive graphics version of the SPICE2 program. SPICE2 is a circuit analysis program featuring AC analysis, transient analysis, DC, noise, sensitivity, driving point impedance, Fourier, temperature, distortion, transfer characteristics, and transmission analysis.

LANGUAGE: FORTRAN

HOST COMPUTERT VAX, PRIME, IBM maxi's, CDC maxi's

REFERENCE: AB Associates Announcement

PROGRAM NAME: INTERAC -- An Interactive Software Package for Direct Digital Control Design

DESCRIPTION: Synthesizes a discrete multi-variable feedback gain matrix to control a multi-input, multi-output continuous control system. Three types of design problems are solved: regulator, disturbance rejector, and tracker.

LANGUAGE: FORTRAN IV

HOST COMPUTER: CDC CYBER

REFERENCE;

Colgate, James A. INTERAC - An Interactive Software Package for Direct Digital Control Design. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, December 1977.

PROGRAM NAME: KEDDC

DESCRIPTION: KEDDC is a comprehensive software package especially designed to cover a wide range of control engineering tasks. At present it contains about 700 modules for process identification, controller design, and testing using a broad variety of modern as well as classical methods. Command-driven dialog, graphics output, a unified software interface to hardware and operating system, numerical efficiency, flexibility of use, transparency and bersatility in the presentation of results are the main features of this portable CAD system.

REFERENCE: Dr. Ing. Chr. Schmid
Department of Electrical Engineering
Ruhr-University Bochum
IC 3/141
P.O.B. 102 148
D-4630 Bochum 1, F.R.G.

PROGRAM NAME: LPASS -- Low Pass Filter Design

DESCRIPTION: Designs maximally flat Butterworth or equiripple Chebychev low pass filter. First analog filter is specified, then transformed with bilinear Z-transform to yield the equivalent digital filter. Interactive or batch. Must be called as a subroutine from main program.

LANGUAGE: FORTRAN IV

HOST COMPUTER: PDP-11/20

REFERENCE:

Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

PROGRAM NAME: LSAP -- Linear Systems Analysis Program

DESCRIPTION: An interactive program with graphics capability used for analysis and design of linear control systems. Classical design tools: transfer function manipulation, root locus analysis, frequency response, time response. Analyzes both continuous and sampled data systems. 32K overlay structure.

LANGUAGE: Pascal

HOST COMPUTER: PDP-11/45, RSX-11M operating system

REFERENCE:

Herget, C. J. and Thomas P. Weis. Linear Systems Analysis Program User's Manual. UCID-30184. Livermore, California: Lawrence Livermore Laboratory, October 1980.

PROGRAM NAME: LSD

DESCRIPTION: LSD is an interactive program that performs many of the calculations necessary for the design of linear gaussian quadratic regulators and related calculations. A user "friendly" program, LSD has been used by students and engineers with little knowledge of computer programming or the operating system on which the program runs. The program works with 15 key matrices which the user can save on a file for reuse at a later session.

The LSD program, developed by Singer-Kearfott, is written entirely in FORTRAN and has a highly modular structure (over 50 subroutines) which permits easy modification to enhance its capabilities or to use new algorithms that become available. Versions of the LSD program have run on the following computers: UNIVAC 1108 under Infonet, IBM 3033 under TSO, PDP-11/03 under RT-11. (LSD has been broken into four subprograms that transfer files to each other for PDP-11 operation).

A version of LSD with a slightly different user dialog has been developed with the assistance of members of the Polytechnic Institute of New York to run under the UNIX operating system. This version has on-line graphics capabilities for display of Bode plots and transient response simulations. Other versions of LSD prepare plot files which are inputs to other graphics programs.

A project is now in progress to prepare a version of LSD to run on an IBM Personal Computer with 256K of user memory.

REFERENCE: Bernard Friedland
The Singer Company, Kearfott Division and
Polytechnic Institute of New York
Systems Research Department
1150 McBride Avenue
Little Falls, New Jersey 07424

PROGRAM NAME: MATRIXx

DESCRIPTION: A data analysis, systems identification, control design and simulation package. It is an interactive software system for computer-aided design and analysis of control systems for dynamic plants. Handles multiple-input/multiple-output systems. Has command interpreter. Solves Riccati equations. Uses state-of-the-art algorithms for linear system analysis, differential equation solution and Fourier transformation.

LANGUAGE: ANSI FORTRAN 77

HOST COMPUTER: VAX 11/780, planned for IBM 3033, CDC

REFERENCE:

Walker, Robert, Charles Gregory, Jr., and Sunil Shah. "MATRIXx: A Data Analysis, System Identification Control Design and Simulation Program," Abstract of paper awaiting publication.

PROGRAM NAME: SOFE -- A Generalized Digital Simulation for Optimal Filter Evaluation

DESCRIPTION: Helps to design and evaluate Kalman filters for integrated systems. SOFE is a Monte Carlo simulation that can be used for system performance analysis once the Kalman filter is designed and verified. A companion post-processor program, SOFEPL, is used for doing ensemble averaging across runs and for making pen plots. Uses batch.

LANGUAGE: '66 ANSI FORTRAN

HOST COMPUTER: CDC CYBER-74

REFERENCE:

Musick, Stanton H. SOFE: A Generalized Digital Simulation for Optimal Filter Evaluation User's Manual, AFWAL-TR-80-1108. Wright-Patterson Air Force Base, Ohio: Avionics Laboratory, October 1980.

PROGRAM NAME: SUPER-SCEPTRE

DESCRIPTION: Analyzes electronic circuits, mechanical systems, logic, transfer functions, and guidance and control systems.

LANGUAGE: FORTRAN

HOST COMPUTER: VAX, PRIME, IBM maxi's, CDC maxi's

REFERENCE: AB Associates announcement

PROGRAM NAME: TOTAL -- Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis

DESCRIPTION: An interactive computer aided design program for continuous and discrete control systems. Classical tools: Block diagram manipulation, root locus analysis, frequency response, time response. Modern Techniques: Matrix manipulation and state-space analysis. Continuous to discrete transformations: impluse invariance, Tustin approximation, first difference approximation. 65K over-lay structure (1 main, 19 primary, 25 secondary) -- total of 600,000 (octal).

LANGUAGE: FORTRAN IV / FORTRAN-77

HOST COMPUTER: CDC CYBER

REFERENCE:

Larimer, Stanley J. TOTAL User's Manual (CAD). Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, June 1981.

PROGRAM NAME: VAXTOTAL

DESCRIPTION: The implementation of TOTAL (cf.) on the VAX 11/780. Interactive mode of operation at 9600 baud.

LANGUAGE: DEC FORTRAN-77

HOST COMPUTER: VAX-11/780

REFERENCE:

Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. Wright-Patterson Air Force Base, Ohio: Air Force Institute of Technology, March 1982.

PROGRAM NAME: WINDOW -- Non recursive Digital Filter Program

DESCRIPTION: Generates coefficients for non-recursive digital filters, produces CRT or hardcopy plots of filter response.

LANGUAGE: DEC FORTRAN

HOST COMPUTER: PDP-11/20

REFERENCE:

Brubaker, Thomas A. Development of Improved Design Methods for Digital Filtering Systems. AFAL-TR-77-207. Fort Collins, Colorado: Colorado State University, 1 November 1977.

D.3 Summary

Several computer-aided control system design packages have been synopsized. A brief description is given for each along with an indication of the language used and of the type of computers that host the various packages.

APPENDIX E

COMMAND LANGUAGE DEFINITION

E.1 Introduction

This appendix presents the ICECAP command language definitions in flow chart form. These definitions unambiguously define the ICECAP command language. Definitions are provided in alphabetical order. The standards used to develop the command language diagrams are also provided. Finally, an exhausted listing of every legal ICECAP command is presented along with the allowable abbreviation in each case. The reader unfamiliar with the mnemonics used in these definitions is referred to Appendix I (25).

E.2 List of Command Language Definitions

A list of the described command language definitions appears below:

- CHANGE
- COPY
- DEFINE
- DELETE
- DISPLAY
- FORM
- GRAPHICS
- HELP
- INSERT
- PRINT
- TFORM
- TURN

E.3 Command Language Definition Standards

For the sake of clarity and uniformity the following standards (developed by Gembarowski (4)) are used in developing the diagrams that portray the command language definitions:

E.3.1 All diagrams are to be read from left to right.

E.3.2 Bracketed terms indicate choices. Only one choice per bracket is allowed.

E.3.3 A lower case command word indicates that the feature has not yet been implemented in the language.

E.3.4 The full spelling of each command word is used in each case. It is understood that the abbreviations described in section E.4 are also valid.

E.3.5 Also, the carriage return and the dollar sign are valid choices at any point in the diagram. The carriage return causes the system to prompt the user regarding the choices for the next allowable word. A dollar sign aborts the present commmand string.

E.3.6 In addition at least one blank must separate the words in the command string.

E.3.7 The blanks in some of the brackets are there only to give the diagram balance.

E.3.8 Words that need no object in order to be complete commands are not shown. To date, this includes the commands STOP, UPDATE, and RECOVER.

E.4 ICECAP Commands

The following figures describe the sequence of command words that form an ICECAP command.

```
                        |-        -|
                        | CLDK     |
                        | CLNK     |
                        | GDK      |
                        | GNK      |
         |-        -|   | HDK      |
         | CHANGE   |   |          |
         |_        _|   | HNK      |
                        | OLDK     |
                        | OLNK     |
                        | PLANE    |
                        | TSAMP    |
                        |_        _|
```

Figure E-1. Command Language Definition for CHANGE

```
                  _    _      _      _
                 |      |    |        |
                 |      |    | CLTF   |    | CLTF   |
     _      _    |      |    | GTF    |    | GTF    |
    |        |   |      |    |        |    |        |
    |  COPY  |   |      |    | HTF    |    | HTF    |
    |_      _|   |      |    | OLTF   |    | OLTF   |
                 |_    _|    |_      _|
```

Figure E-2. Command Language Definition for COPY

```
     _        _     _          _
    |          |   |   GAIN     |
    |  DEFINE  |   |            |
    |_        _|   |   MATRIX   |
                   |_          _|

                    _      _
     _        _    | CLTF   |      _      _
    |          |   | GTF    |     | FACT   |
    |  DEFINE  |   |        |     |        |
    |_        _|   | HTF    |     | POLY   |
                   | OLTF   |     |_      _|
                   |_      _|
```

Figure E-3. Command Language Definition for DEFINE

```
                    _      _
     _        _    | CLTF   |      _      _
    |          |   | GTF    |     | ZERO   |
    |  DELETE  |   |        |     |        |
    |_        _|   | HTF    |     | POLE   |
                   | OLTF   |     |_      _|
                   |_      _|
```

Figure E-4. Command Language Definition for DELETE

```
                         CLTF
                         EQUATION
                         GAIN
                         GTF
                         HTF
                         LISTING/F
                         LISTING/T
                         LOCUS/BRANCH
                         LOCUS/GAIN
                         LOCUS/ZETA
                         MATRIX
        DISPLAY
                         MODERN
                         OLTF
                         PFE
                         RESPONSE/F
                         RESPONSE/T
                         RICCATI
                         SCAN/MAG
                         SCAN/PHASE
                         SPECS
                         SWITCHES
                         TRANSFER/F

                                        AUTOSCALE
        DISPLAY          LOCUS          MAGNIFY

                                        SHRINK
                                        ZOOM

                                        CLTF
                                        OLTF
        DISPLAY          ROOT
                         POLY           GTF
                                        HTF
```

Figure E-5. Command Language Definition for DISPLAY

```
            OLTF
FORM        CLTF USING GTF AND HTF
            CLTF USING OLTF
```

Figure E-6. Command Language Definition for FORM

```
             INPUT        POLE
                          ZERO
                          TEXT
                          EXIT

             DISPLAY      POLE
                          ZERO
                          LOCUS
                          CROSS
                          CIRCLE
                          EXIT

GRAPHICS     DELETE       POLE
                          ZERO
                          TEXT
                          CROSS
                          CIRCLE
                          EXIT

             GRID         EXPAND
                          SHRINK
                          ZOOM
                          EXIT

             STOP
```

Figure E-7. Command Language Definition for GRAPHICS

```
                        CHANGE
                        COPY
                        DEFINE
                        DELETE
                        DISPLAY
                        FORM
       HELP             INITIAL
                        INSERT
                        MATRIX
                        PRINT
                        SYSTEM
                        TEACH
                        TFORM
                        TURN
```

Figure E-8. Command Language Definition for HELP

```
                  CLTF
     INSERT       GTF          ZERO

                  HTF          POLE
                  OLTF
```

Figure E-9. Command Language Definition for INSERT

```
                      |¯            ¯|
                      | CLTF         |
                      | EQUATION     |
                      | GAIN         |
                      | GTF          |
                      | HTF          |
                      | LISTING/F    |
                      | LISTING/T    |
   |¯       ¯|        | LOCUS/BRANCH |
   | PRINT   |        | LOCUS/GAIN   |
   |_       _|        | LOCUS/ZETA   |
                      | OLTF         |
                      | PFE          |
                      | RESPONSE/F   |
                      | RESPONSE/T   |
                      | SCAN/MAG     |
                      | SCAN/PHASE   |
                      | SPECS        |
                      |_            _|

                                 |¯         ¯|
                                 | AUTOSCALE |
 |¯       ¯|   |¯       ¯|       | MAGNIFY   |
 | PRINT   |   | LOCUS   |       |           |
 |_       _|   |_       _|       | SHRINK    |
                                 | ZOOM      |
                                 |_         _|

                                      |¯    ¯|
                                      | CLTF |
    |¯        ¯|    |¯      ¯|        | OLTF |
    |  PRINT   |    | ROOT   |        |      |
    |_        _|    | POLY   |        | GTF  |
                    |_      _|        | HTF  |
                                      |_    _|
```

Figure E-10. Command Language Definition for PRINT

```
                           SW
              CLTF         SWP         BAKDIF
              GTF          SZ          BILINEAR
   TFORM                   WPS         IMPULSE
              HTF          WPZ         TUSTIN
              OLTF         WS          MULTIPLE
                           WZ
                           ZW
                           ZWP
```

Figure E-11. Command Language Definition for TFORM

```
                   ANSWER
                   CANCEL
                   CLOSED
   TURN                           ON
                   DECIBELS
                                  OFF
                   GRID
                   HERTZ
                   MAINMENU
```

Figure E-12. Command Language Definition for TURN

E.4 ICECAP Abbreviated Commands

The following list contains every valid ICECAP command defined as of the conclusion of this investigation. The accepted abbreviation in listed alongside the command. Mnemonic definitions may be found in Appendix I. Program execution upon receipt of these commands is the subject of (25). Further details may be found by matching the contents of (25) against the procedure DISPLAY_OR_PRIN in the Pascal source listing of ICECAP located in the AFIT Digital Equipment Laboratory.

COMMAND	ABBREVIATION
CHANGE (numerator or denominator gain)	
CHANGE CLNK	CHA CLN
CHANGE PLANE	CHA PLA
CHANGE TSAMP	CHA TSA
COPY (source) (destinstion)	
COPY CLTF OLTF	COP CLT OLT
COPY GTF HTF	COP GTF HTF
DEFINE GAIN	DEF GAI
DEFINE INPUT	DEF INP
DEFINE (function) (fact/poly)	
DEFINE CLTF POLY	DEF CLT POL
DEFINE OLTF FACT	DEF OLT FAC
DEFINE MATRIX	DEF MAT
DELETE (function) (POLE or ZERO)	
DELETE HTF ZERO	DEL HTF ZER
DISPLAY (function)	
DISPLAY OLTF	DIS OLT

COMMAND	ABBREVIATION
DISPLAY EQUATION	DIS EQU
DISPLAY GAIN	DIA GAI
DISPLAY LISTING/F	DIS L/F
DISPLAY LISTING/T	DIS L/T
DISPLAY LOCUS AUTOSCALE	DIS LOC AUT
DISPLAY LOCUS MAGNIFY	DIS LOC MAG
DISPLAY LOCUS SHRINK	DIS LOC SHR
DISPLAY LOCUS ZOOM	DIS LOC ZOO
DISPLAY LOCUS/BRANCH	DIS L/B
DISPLAY LOCUS/GAIN	DIS L/G
DISPLAY LOCUS/ZETA	DIS L/Z
DISPLAY MATRIX	DIS MAT
DISPLAY MODERN	DIS MOD
DISPLAY PFE	DIS PFE
DISPLAY POLY (function)	
DISPLAY POLY GTF	DIS POLY GTF
DISPLAY RESPONSE/F	DIS R/F
DISPLAY RESPONSE/T	DIS R/T
DISPLAY ROOT (function)	
DISPLAY ROOT CLTF	DIS ROO CLT
DISPLAY SCAN/MAG	DIS S/M
DISPLAY SCAN/PHASE	DIS S/P
DISPLAY SPECS	DIS SPE
DISPLAY SWITCHES	DIS SWI
DISPLAY TRANSFER/F	DIS T/F
FORM OLTF	FOR OLT
FORM CLTF USING GTF AND HTF	FOR CLT USI GTF AND HTF
FORM CLTF USING OLTF	FOR CLT USI OLT
GRAPHICS	GRA
HELP CHANGE	HEL CHA
HELP COPY	HEL COP
HELP DEFINE	HEL DEF
HELP DELETE	HEL DEL
HELP DISPLAY	HEL DIS
HELP FORM	HEL FOR
HELP INITIAL	HEL INI
HELP INSERT	HEL INS
HELP MATRIX	HEL MAT
HELP PRINT	HEL PRI
HELP SYSTEM	HEL SYSTEM
HELP TEACH	HEL TEA
HELP TFORM	HEL TFO
HELP TURN	HEL TUR

COMMAND	ABBREVIATION
INSERT (function) (POLE or ZERO)	
INSERT HTF ZERO	INS HTF ZER
PRINT EQUATION	PRI EQU
PRINT GAIN	PRI GAI
PRINT LISTING/F	PRI L/F
PRINT LISTING/T	PRI L/T
PRINT LOCUS AUTOSCALE	PRI LOC AUT
PRINT LOCUS MAGNIFY	PRI LOC MAG
PRINT LOCUS SHRINK	PRI LOC SHR
PRINT LOCUS ZOOM	PRI LOC ZOO
PRINT LOCUS/BRANCH	PRI L/B
PRINT LOCUS/GAIN	PRI L/G
PRINT LOCUS/ZETA	PRI L/Z
PRINT PFE	PRI PFE
PRINT RESPONSE/F	PRI R/F
PRINT RESPONSE/T	PRI R/T
PRINT ROOT (function)	
PRINT ROOT CLTF	DIS ROO CLT
PRINT SCAN/MAG	PRI S/M
PRINT SCAN/PHASE	PRI S/P
PRINT SPECS	PRI SPE
RECOVER	REC
STOP	STO
TFORM (function) ("from"plane"to"plane) (method)	
TFORM CLTF SZ TUSTIN	TFO CLT SZ TUS
TURN ANSWER (ON/OFF)	TUR ANS (ON/OFF)
TURN CANCEL (ON/OFF)	TUR CAN (ON/OFF)
TURN CLOSED (ON/OFF)	TUR CLO (ON/OFF)
TURN DECIBELS (ON/OFF)	TUR DEC (ON/OFF)
TURN GRID (ON/OFF)	TUR GRI (ON/OFF)
TURN HERTZ (ON/OFF)	TUR HER (ON/OFF)
TURN MAINMENU (ON/OFF)	TUR MAI (ON/OFF)
UPDATE	UPD

E.5 Summary

This appendix has provided an unambiguous definition of the ICECAP command language in diagram form. An exhaustive listing of all ICECAP commands and their legal abbreviations is included. The standards which were used in developing the diagrams have been provided to help the reader understand the diagrams and to serve as a guideline for others who will be extending the language to add more commands.

APPENDIX F

Graphical ICECAP Data Flow Diagrams

F.1 INTRODUCTION

This appendix is intended to be used for those who will continue in the development of new routines for Graphical ICECAP. Although most of the information contained in this appendix is aimed specifically at Graphical ICECAP,the data flow diagrams are general enough to be used as the basis for the development of a entirely new system. This appendix is a "living document" and is repeated from Appendix A of Mark Travis's thesis. It is presented here for completeness and for those who will follow in this effort.

F.2 Description of Data Flow Diagrams

The data flow diagrams on the following pages represent the initial design of Graphical ICECAP. Each diagram represents, in a hierarchical fashion the major processes data elements which make up the system.

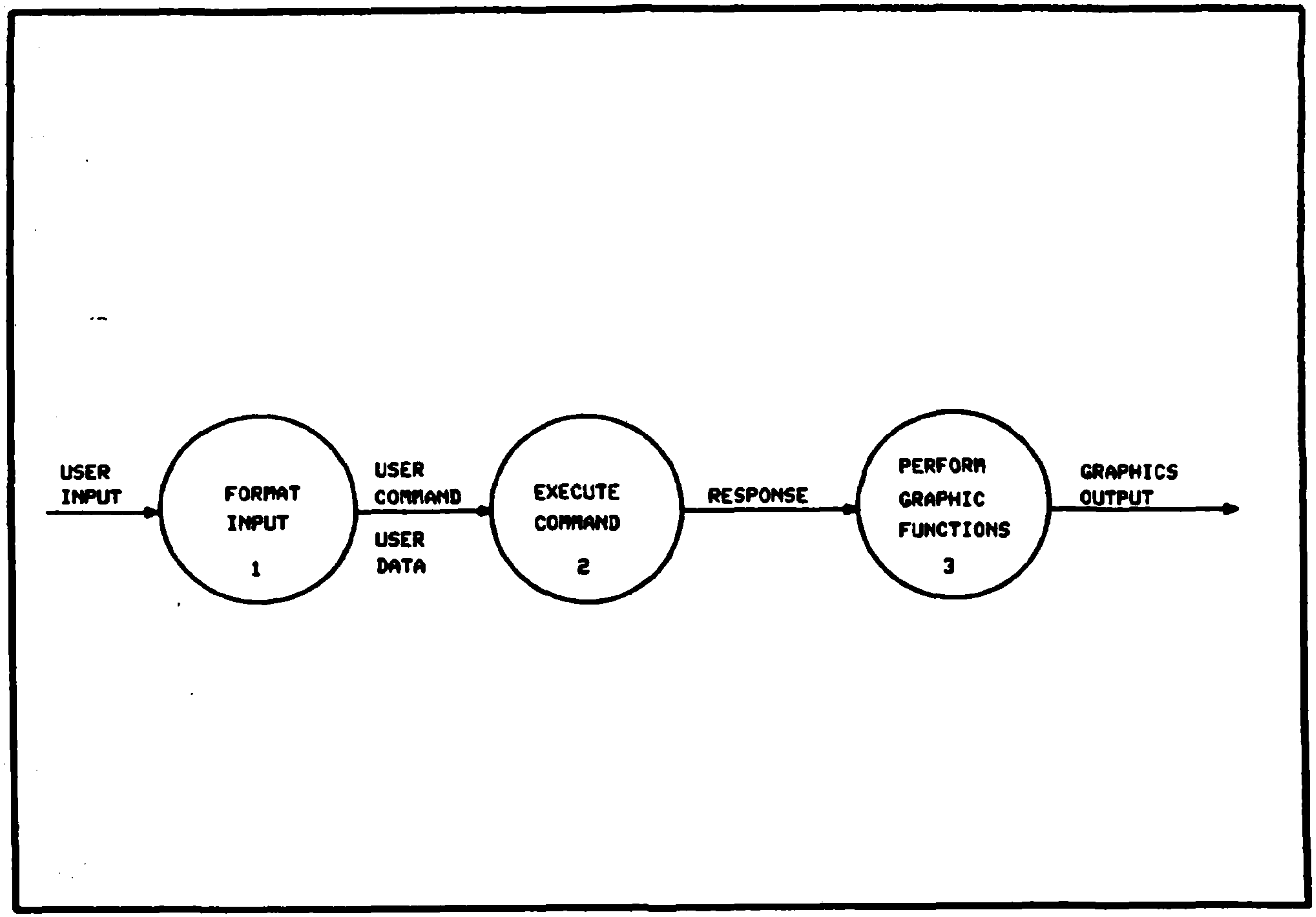


FIGURE F-1. OVERALL SYSTEM DIAGRAM

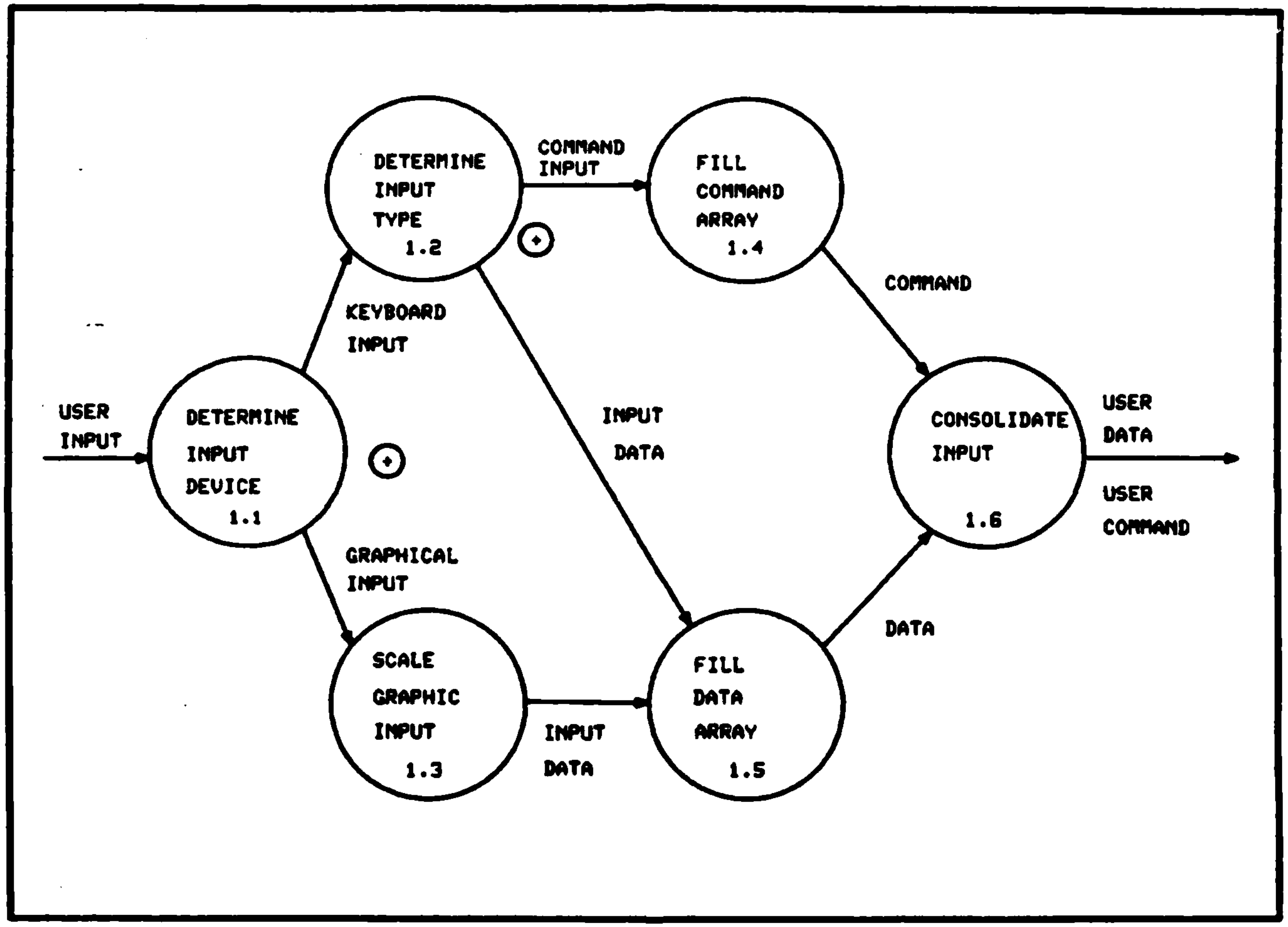


FIGURE F-2. FORMAT INPUT (NODE 1)

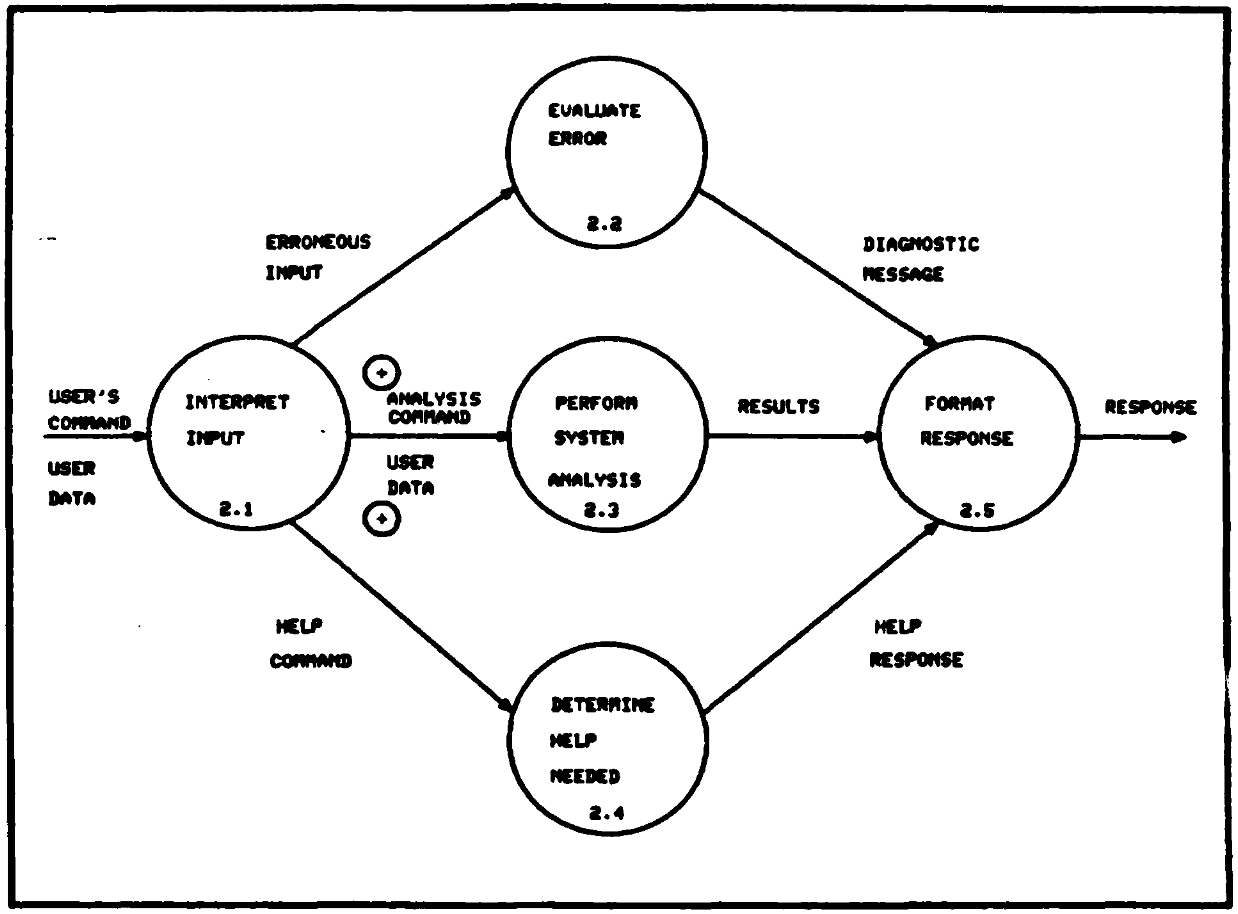


FIGURE F-3. EXECUTE COMMAND (NODE 2)

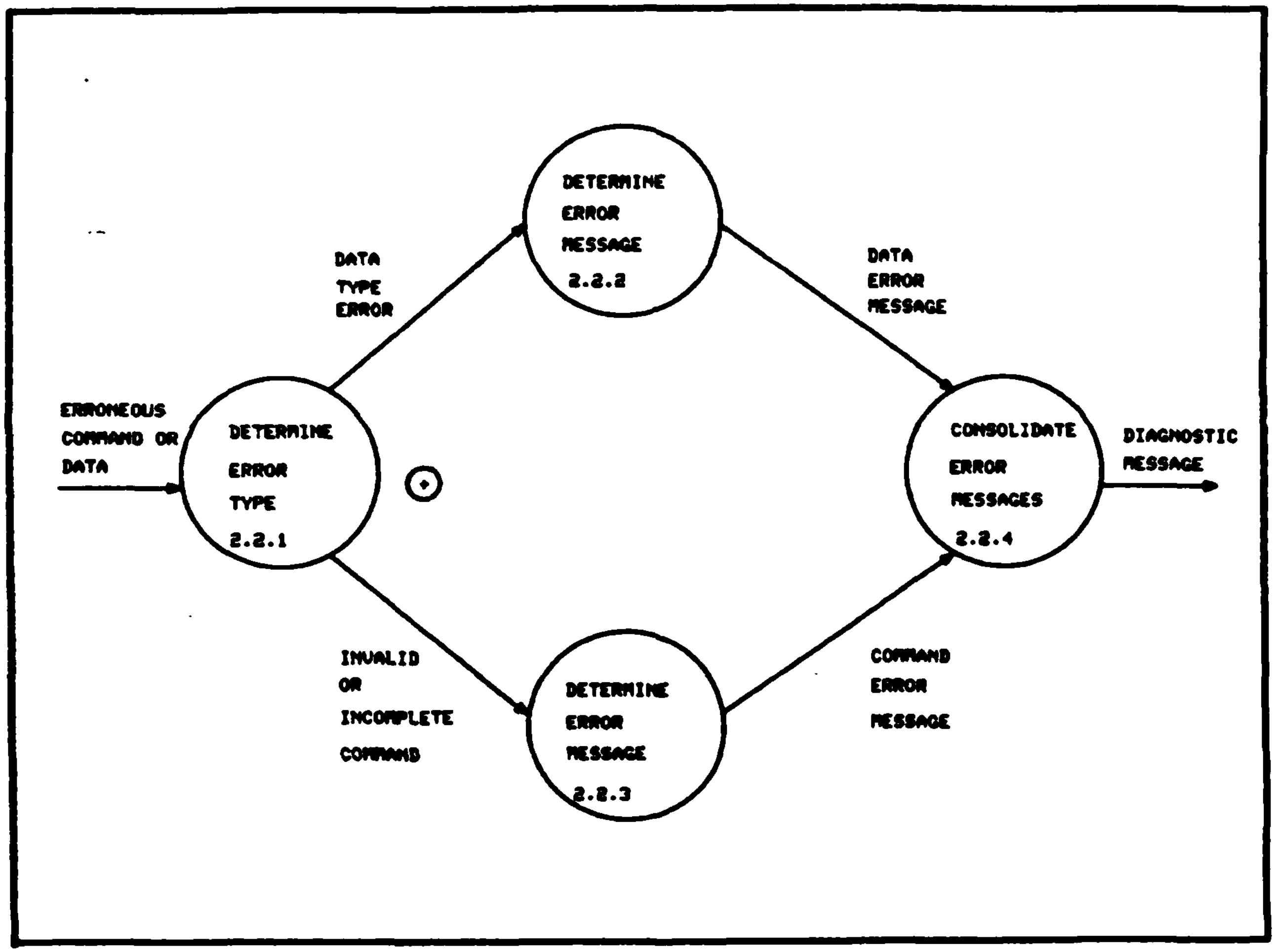


FIGURE F-4. EVALUATE ERROR (NODE 2.2)

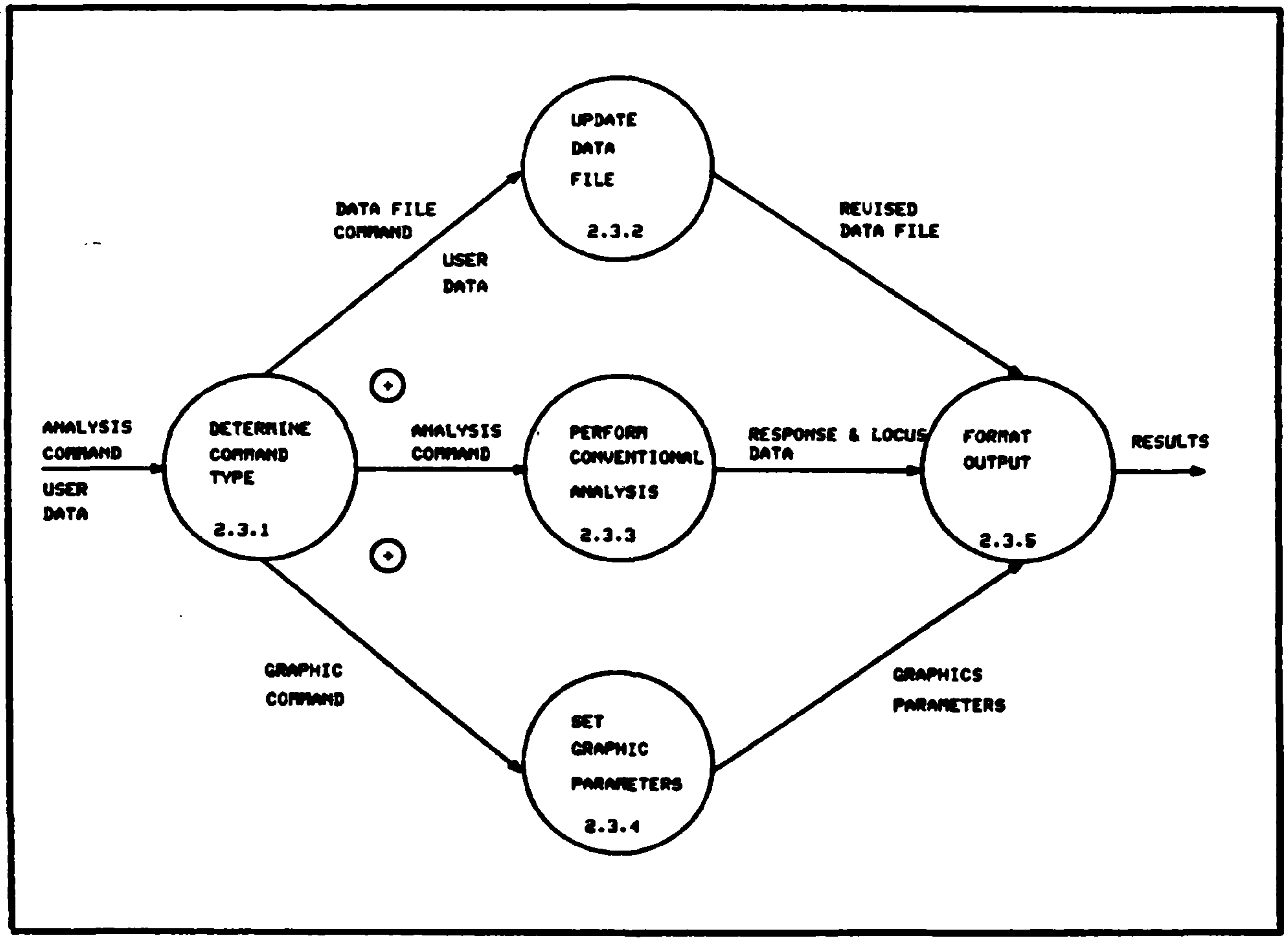


FIGURE F-5. PERFORM SYSTEM ANALYSIS (NODE 2.3)

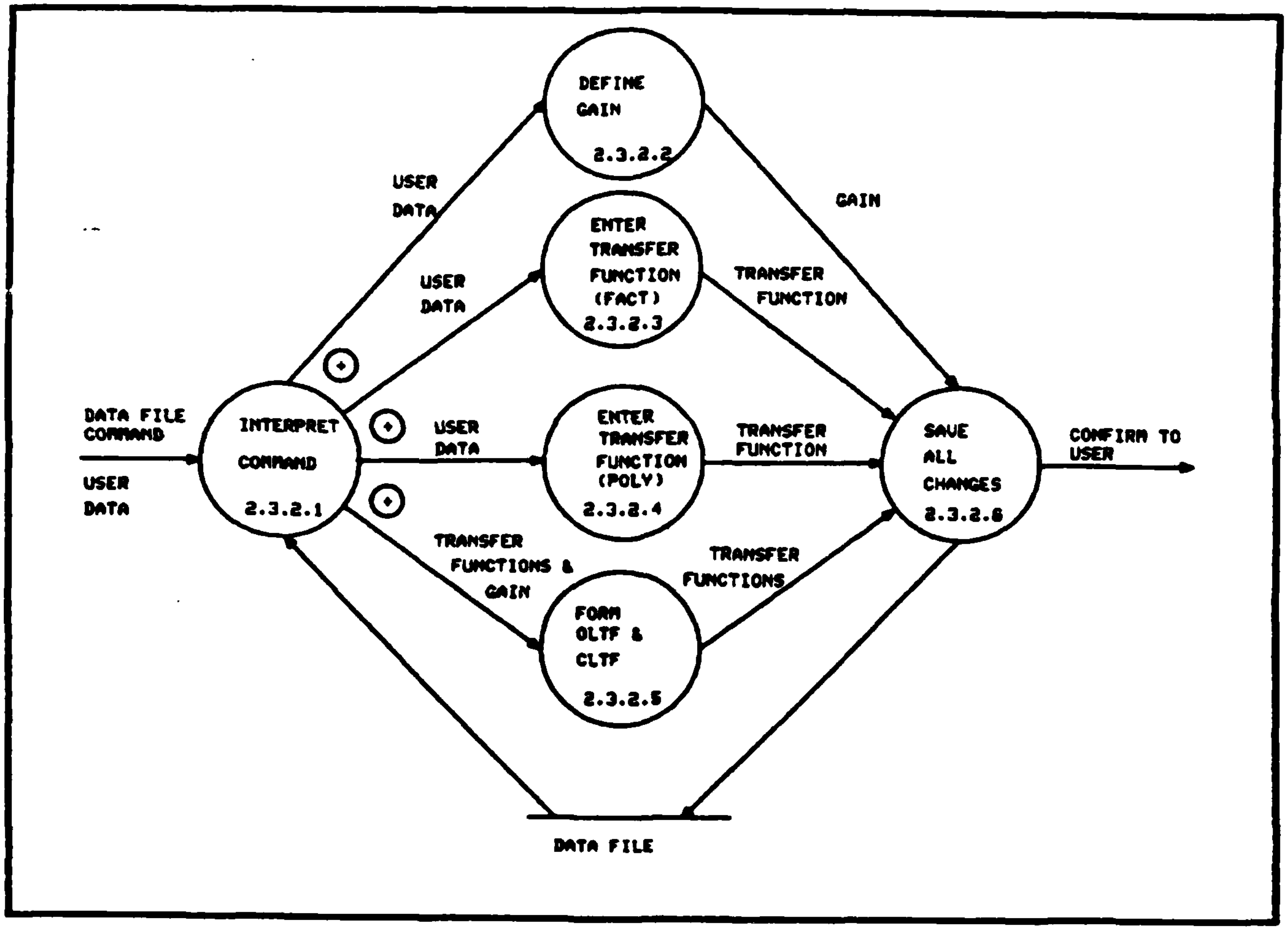


FIGURE F-6. UPDATE DATA FILE (NODE 2.3.2)

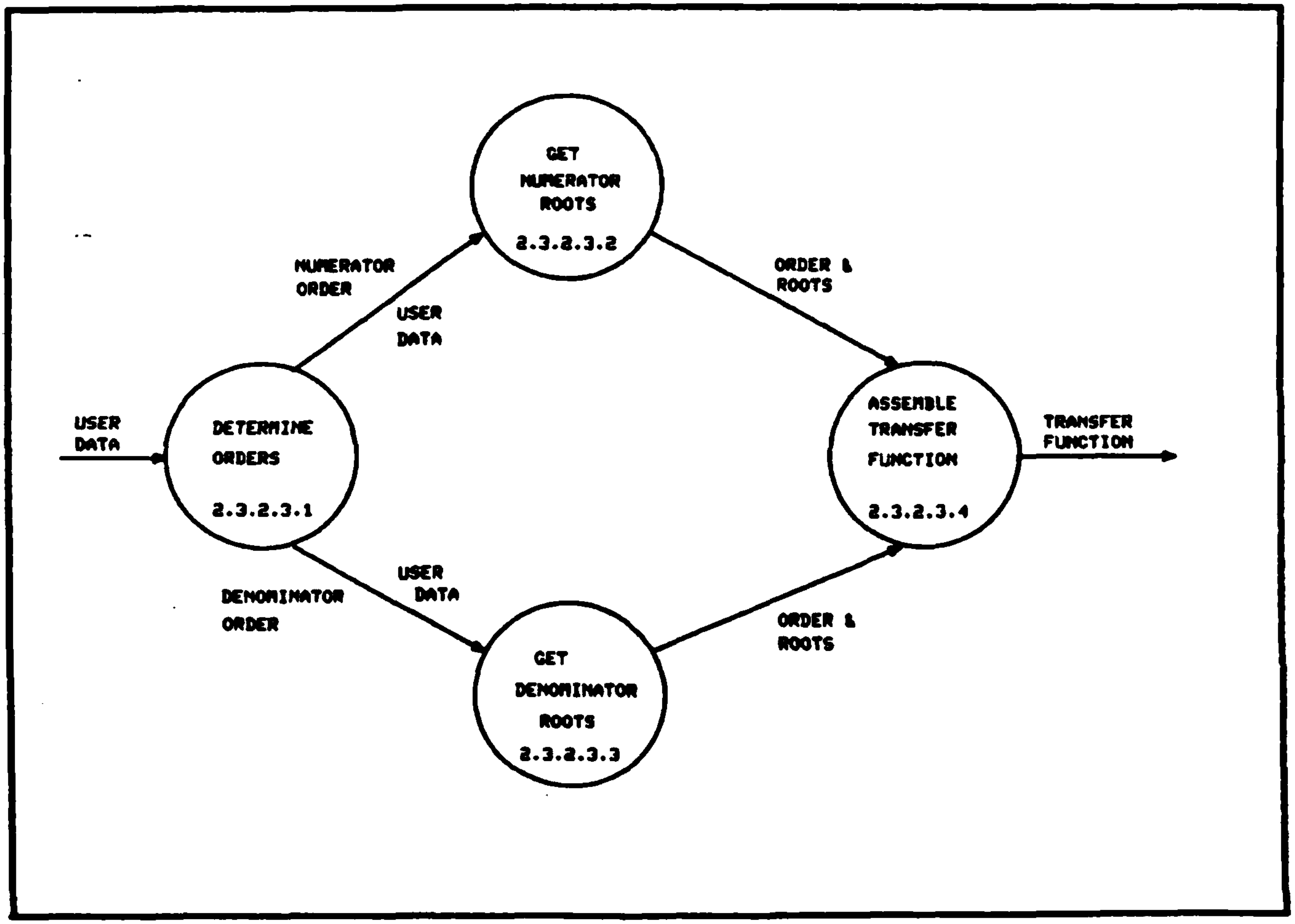


FIGURE F-7. ENTER TRANSFER FUNCTION (ROOTS) (NODE 2.3.2.3)

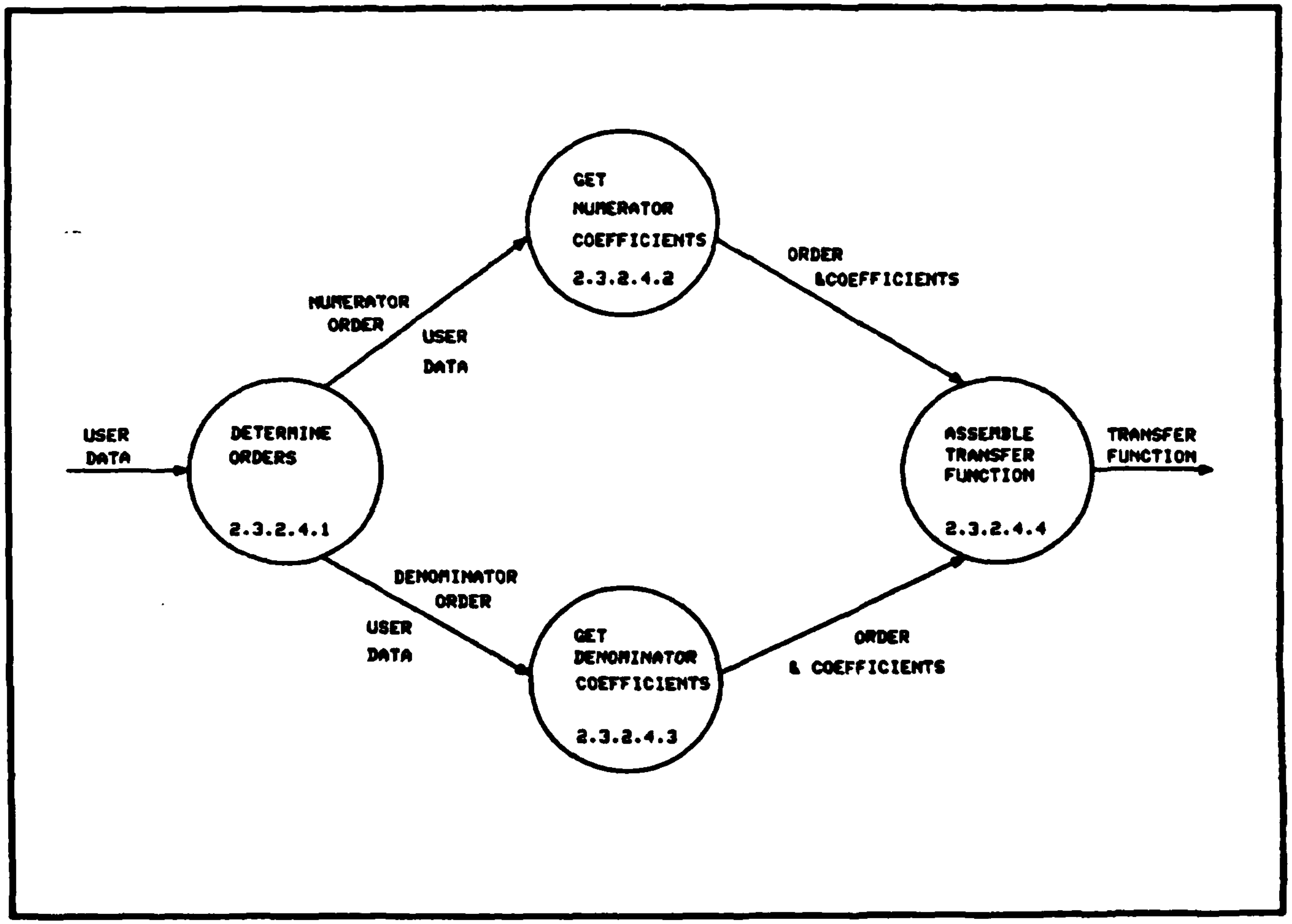


FIGURE F-8. ENTER TRANSFER FUNCTION (POLYNOMINAL) (NODE 2.3.2.4)

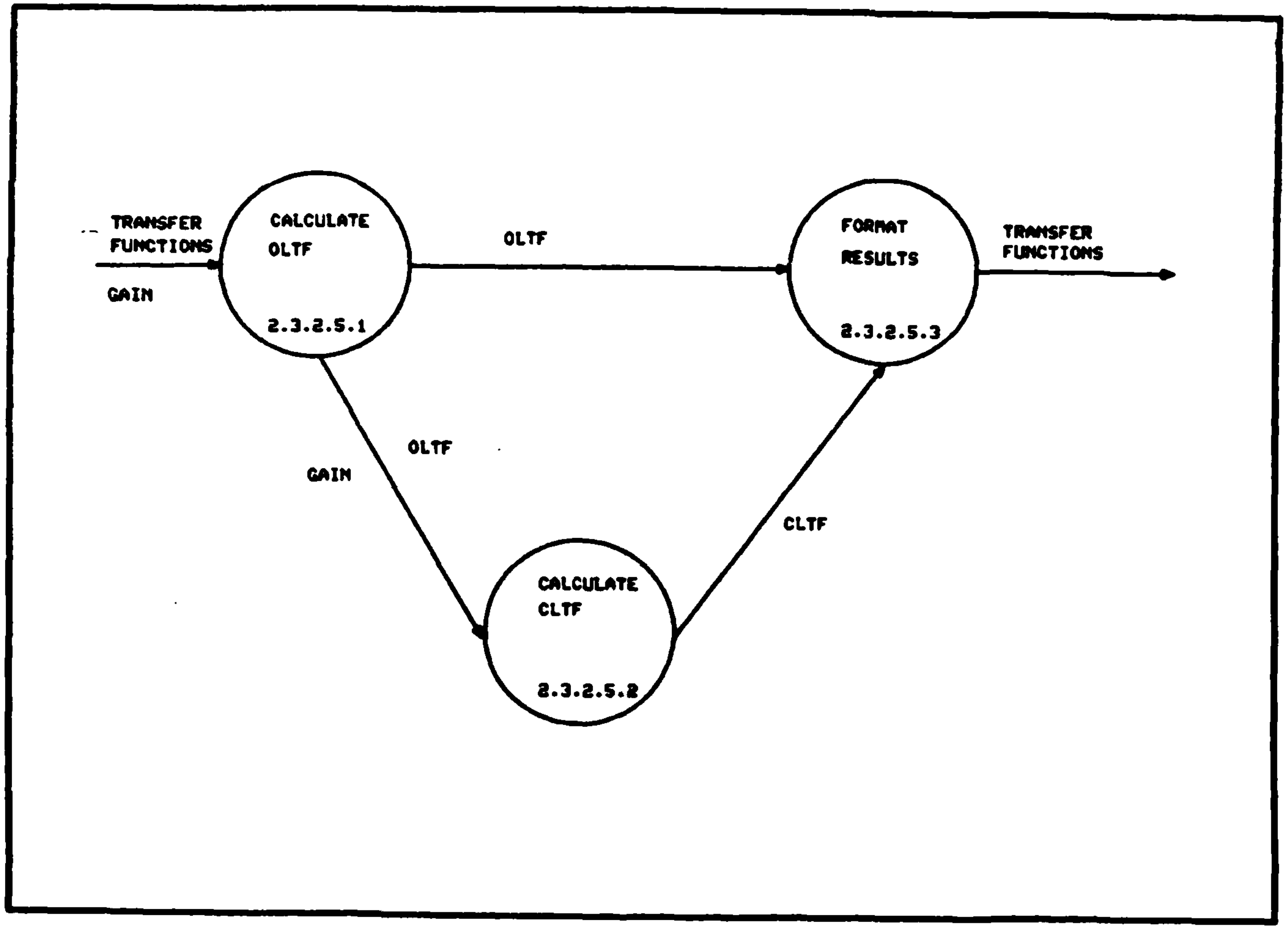


FIGURE F-9. FORM OLTF & CLTF (NODE 2.3.2.5)

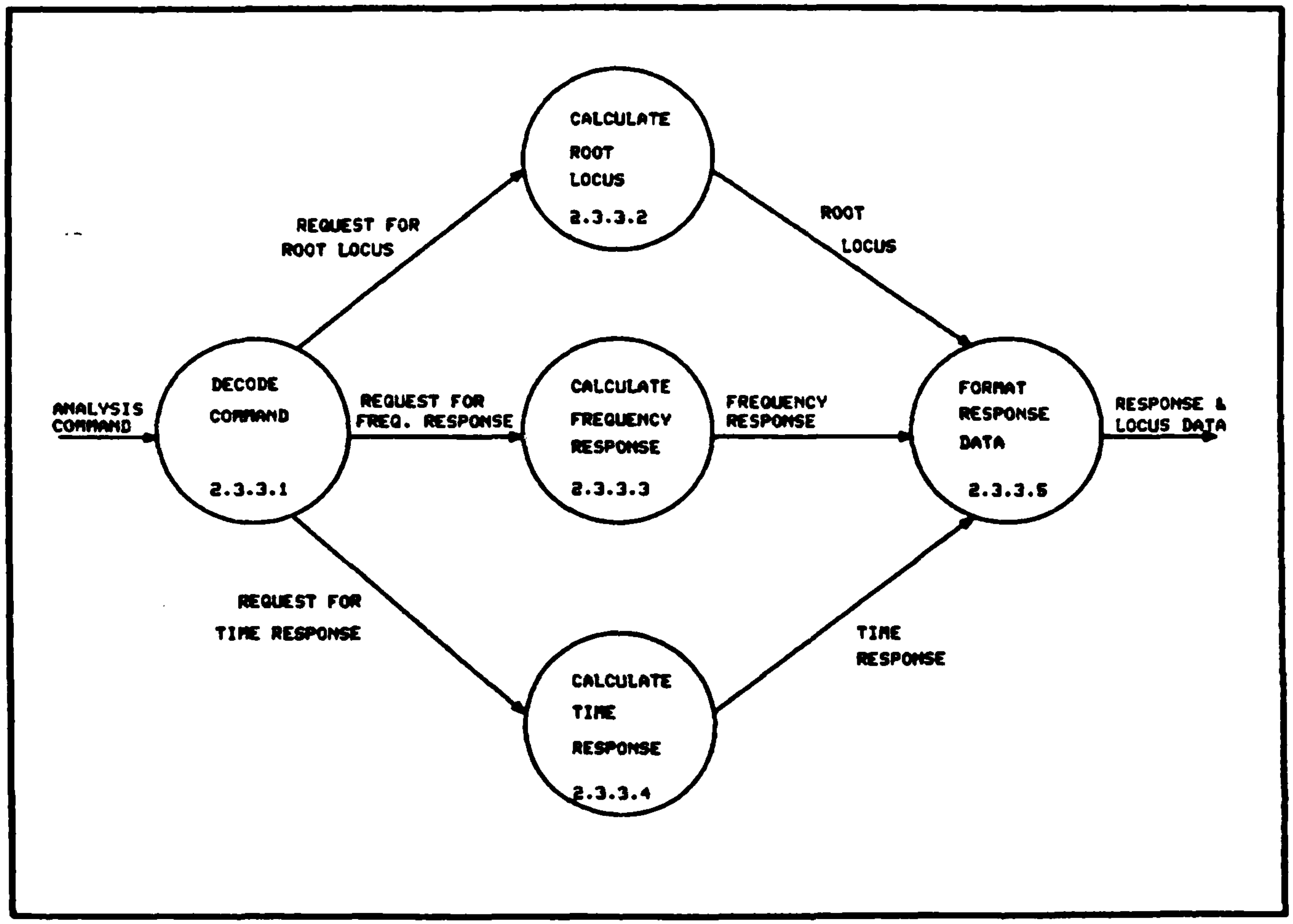


FIGURE F-10. PERFORM CONVENTIONAL ANALYSIS (NODE 2.3.3)

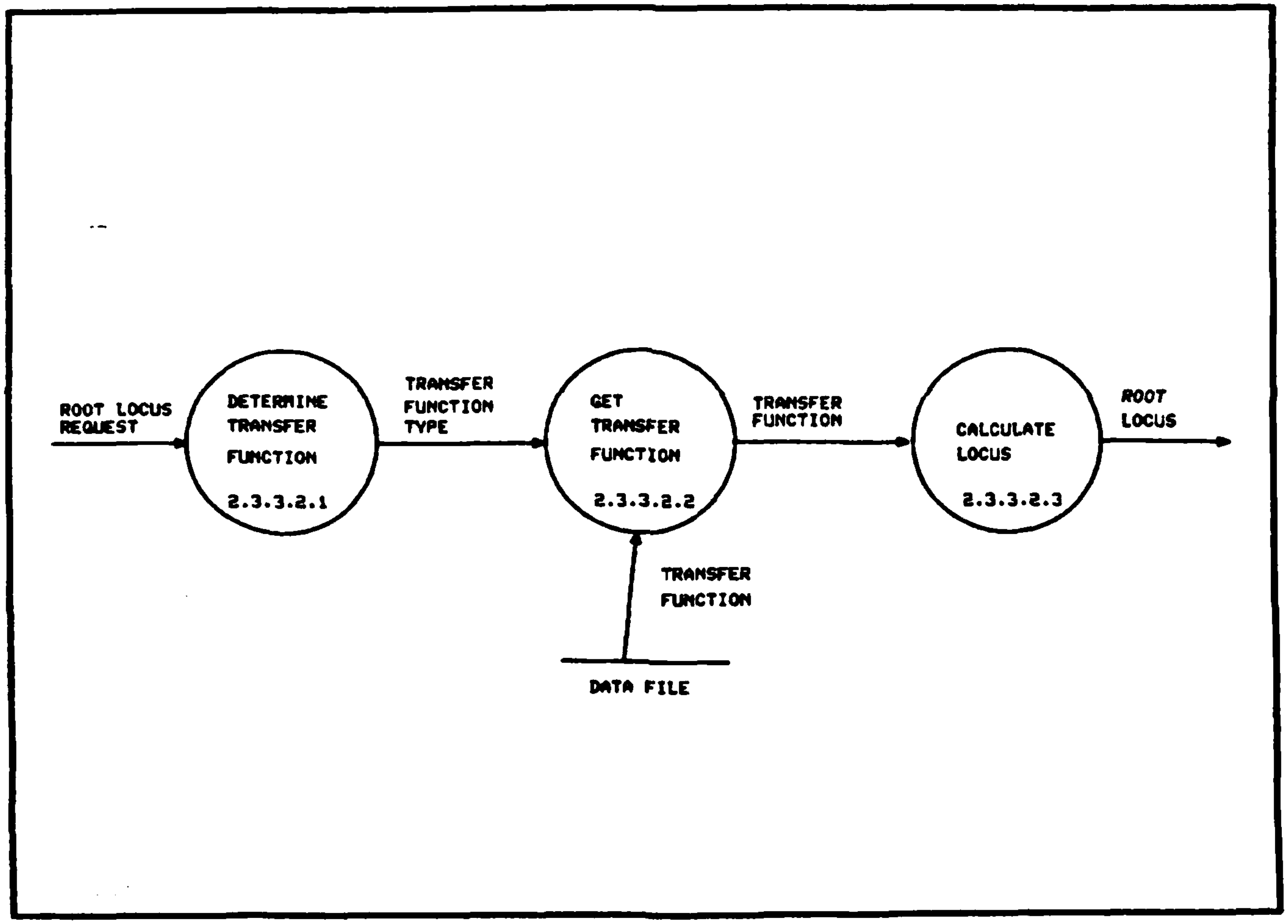


FIGURE F-11. CALCULATE ROOT LOCUS (NODE 2.3.3.2)

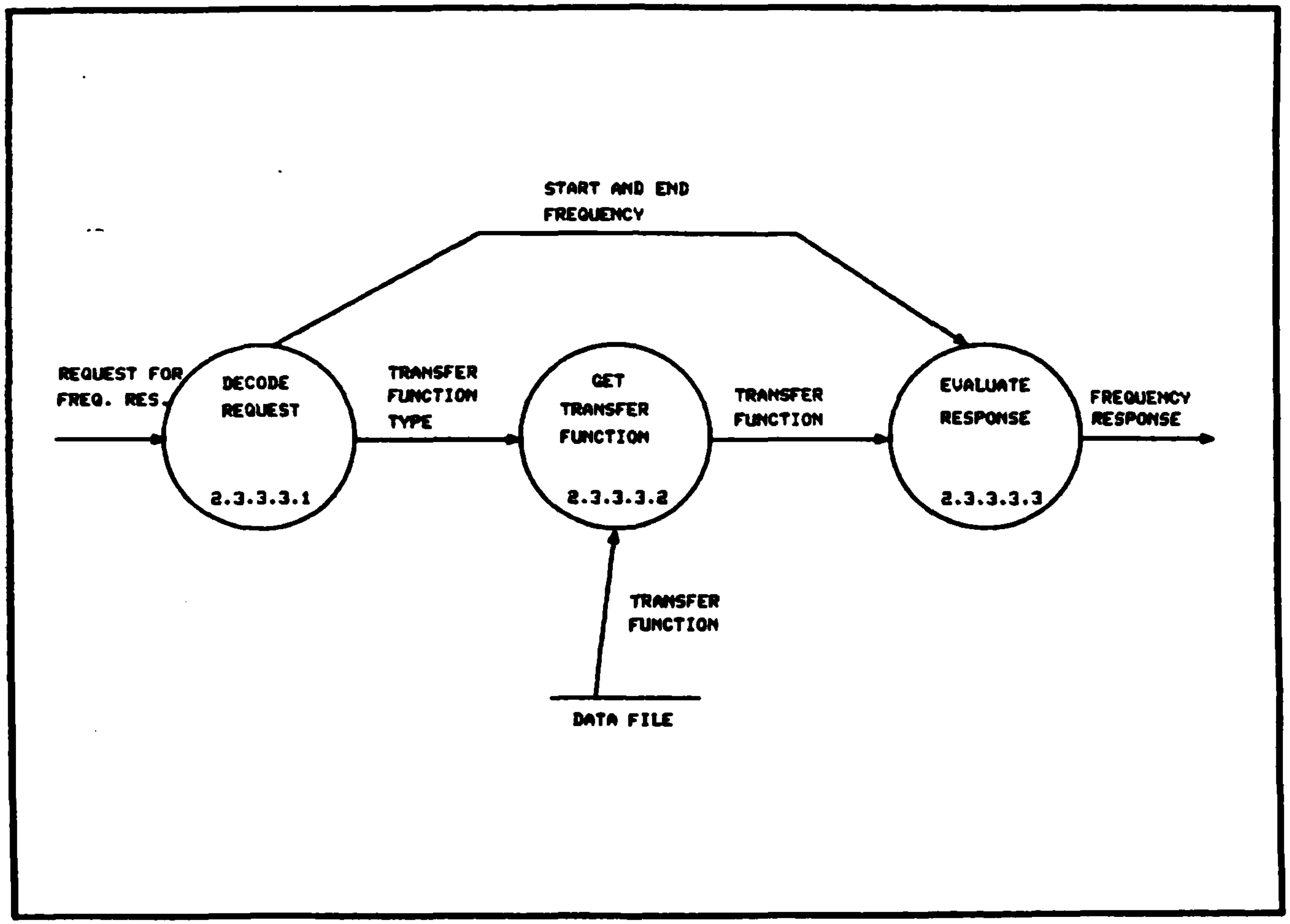


FIGURE F-12. CALCULATE FREQUENCY RESPONSE (NODE 2.3.3.3)

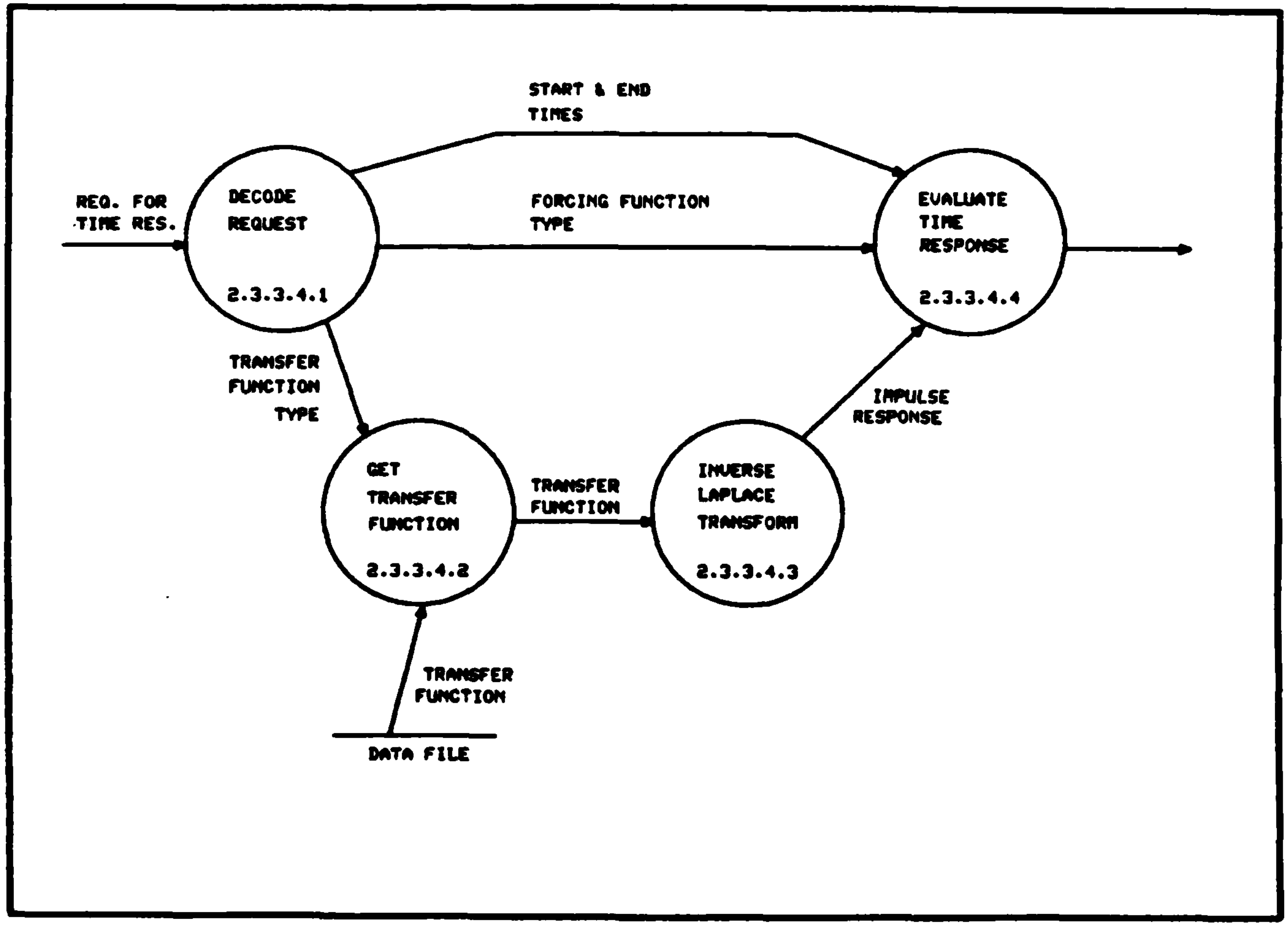


FIGURE F-13. CALCULATE TIME RESPONSE (NODE 2.3.3.4)

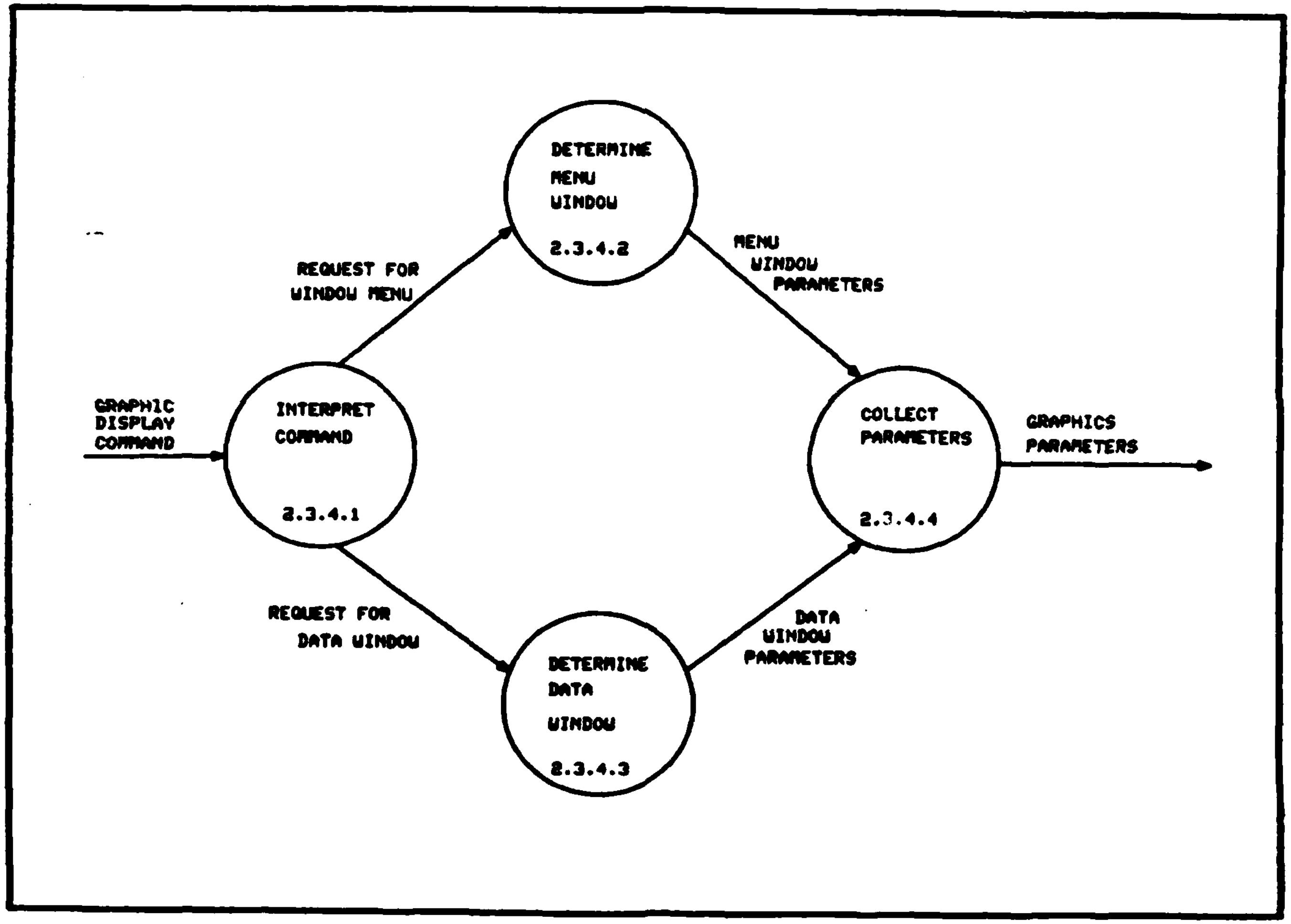


FIGURE F-14. SET GRAPHICS PARAMETERS (NODE 2.3.4)

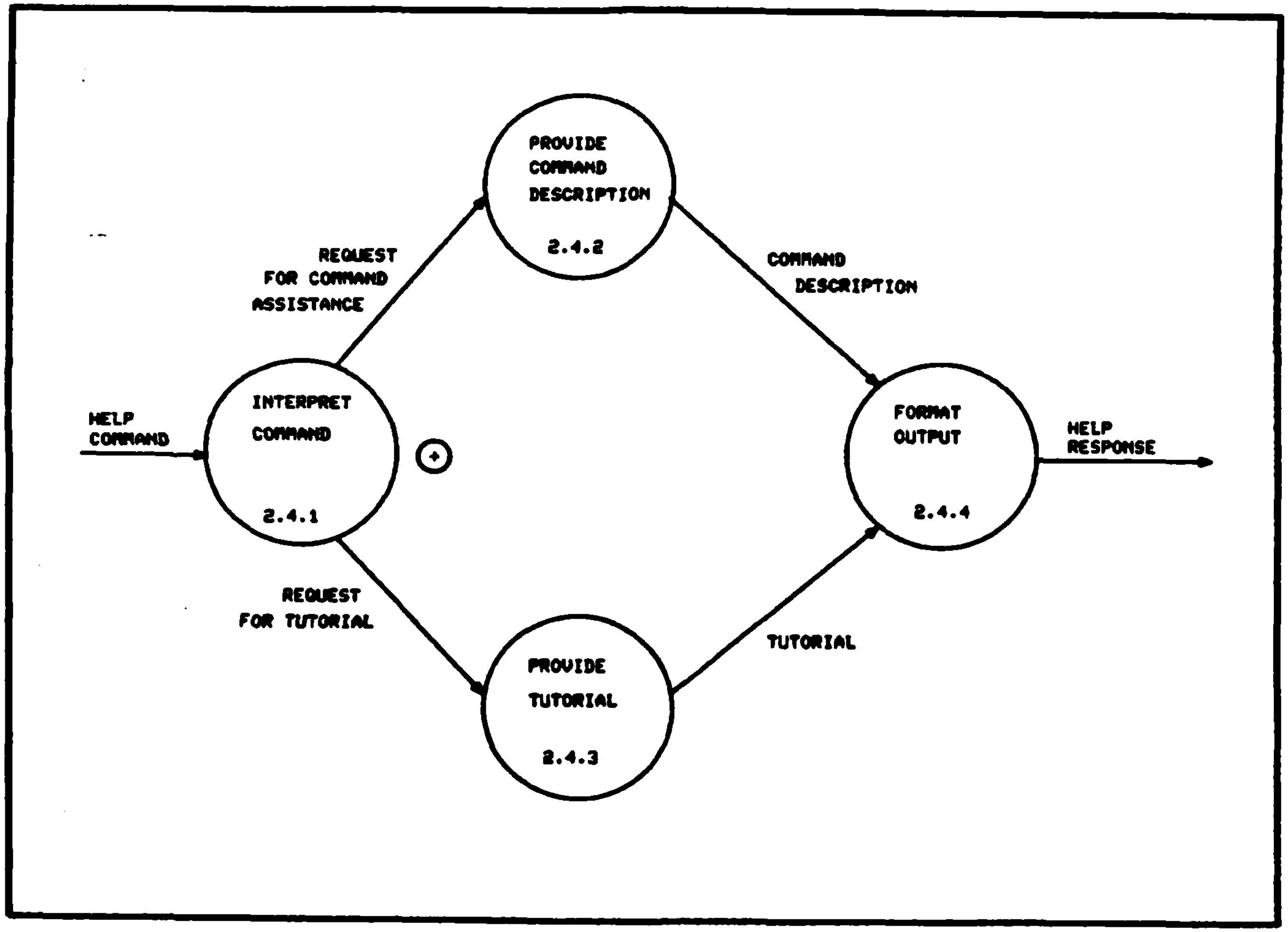


FIGURE F-15. DETERMINE HELP NEEDED (NODE 2.4)

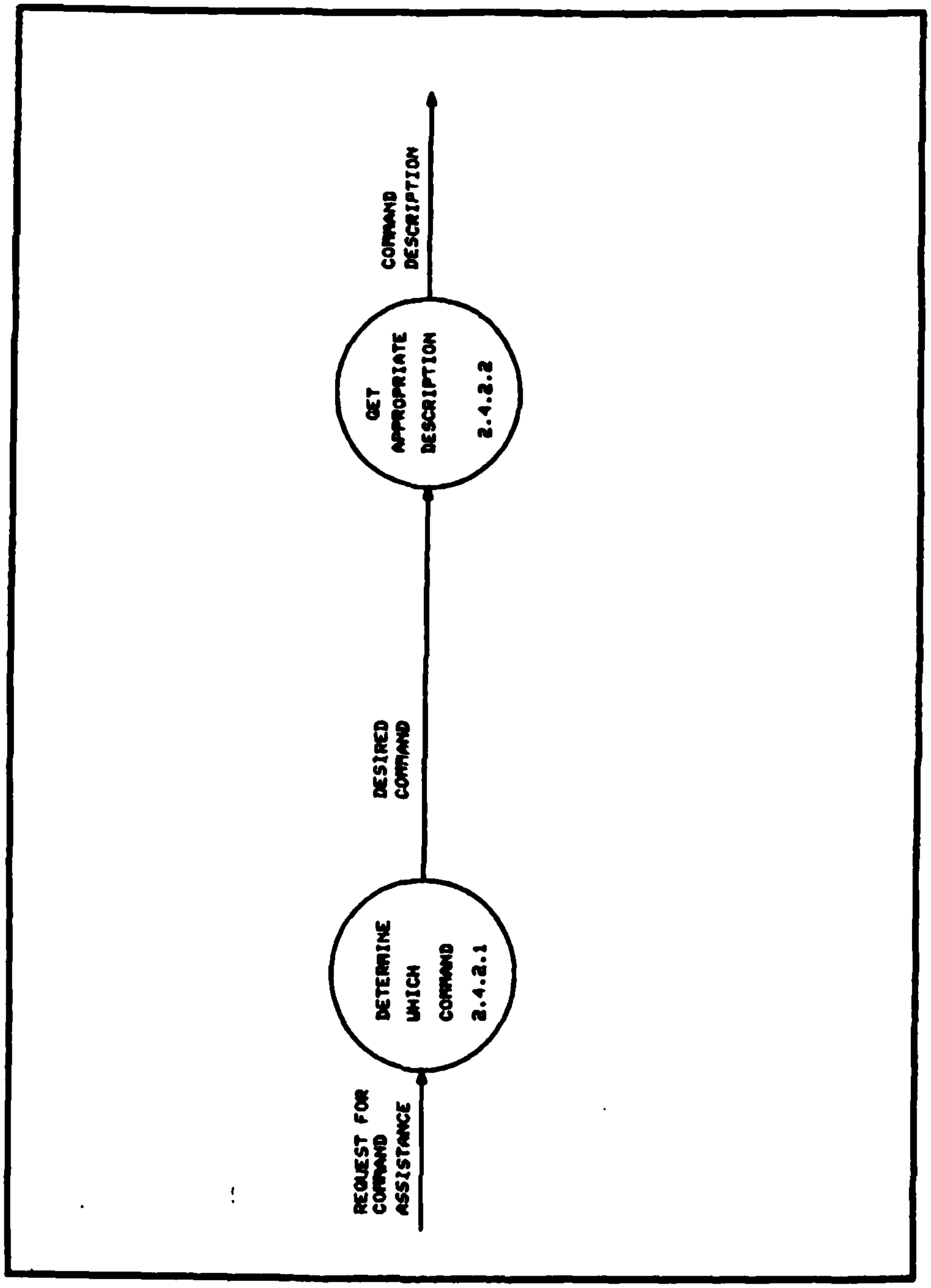


FIGURE F-16. PROVIDE COMMAND DESCRIPTION (NODE 2.4.2)

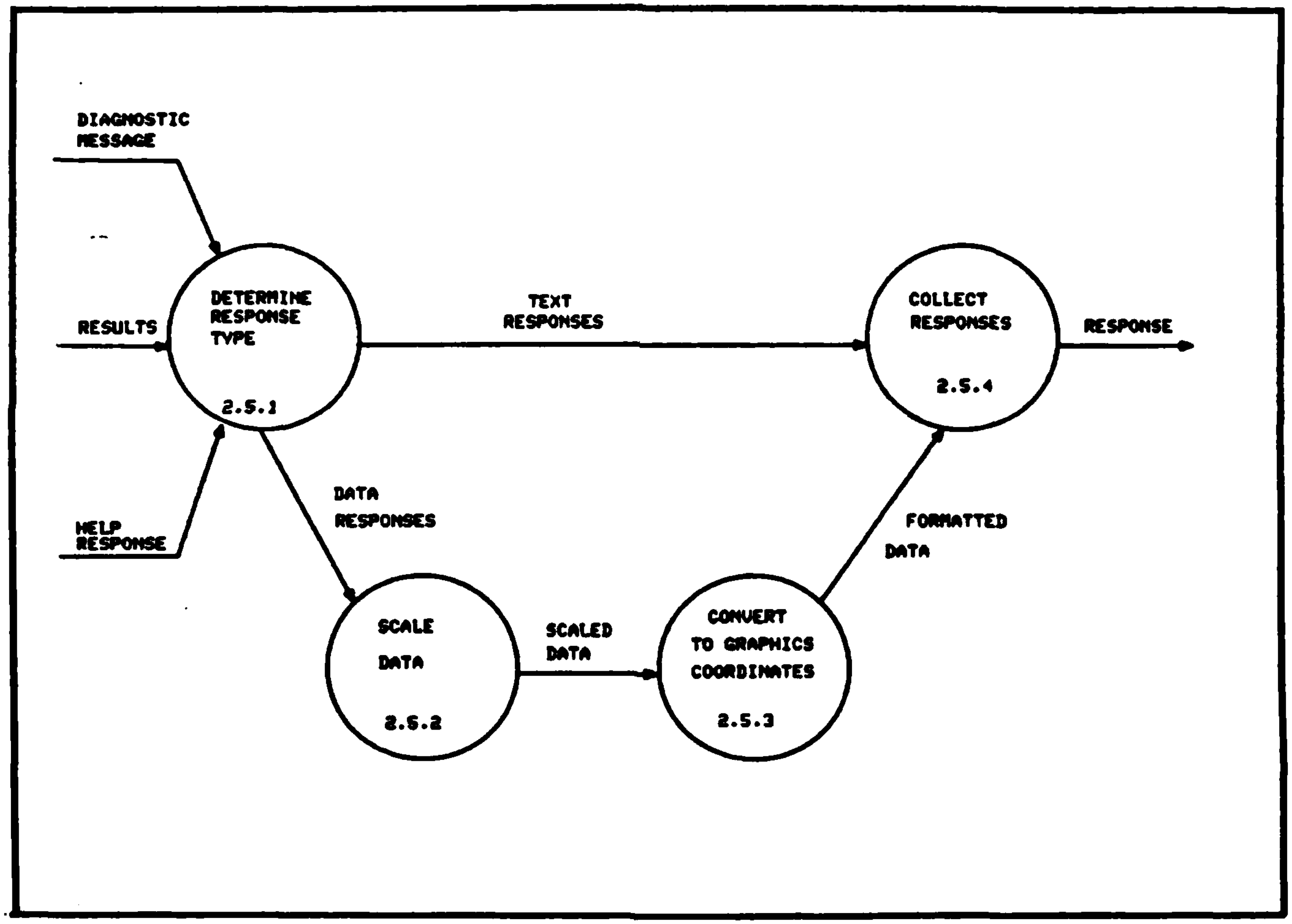


FIGURE F-17. FORMAT RESPONSE (NODE 2.5)

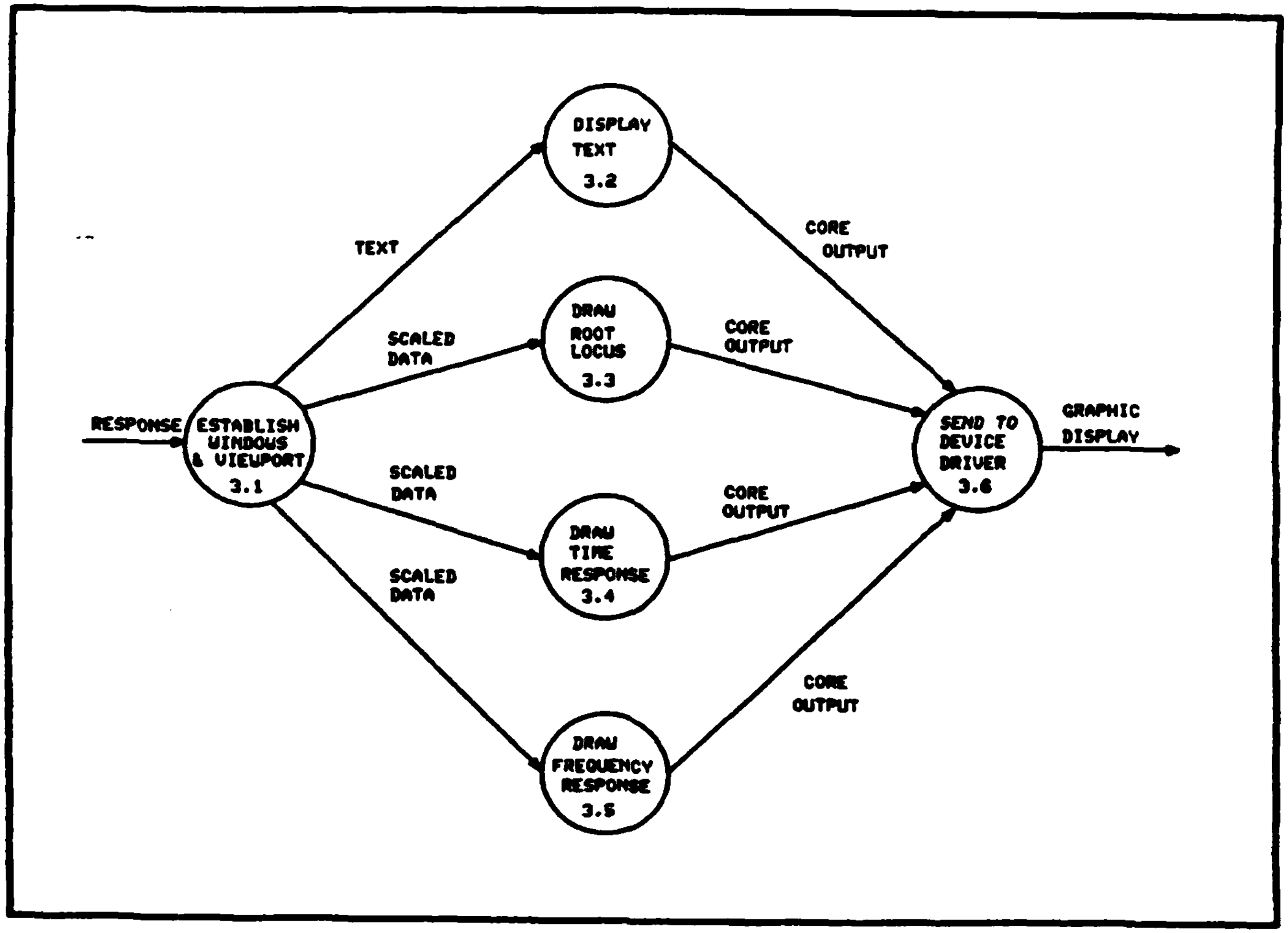


FIGURE F-18. PERFORM GRAPHIC FUNCTIONS (NODE 3)

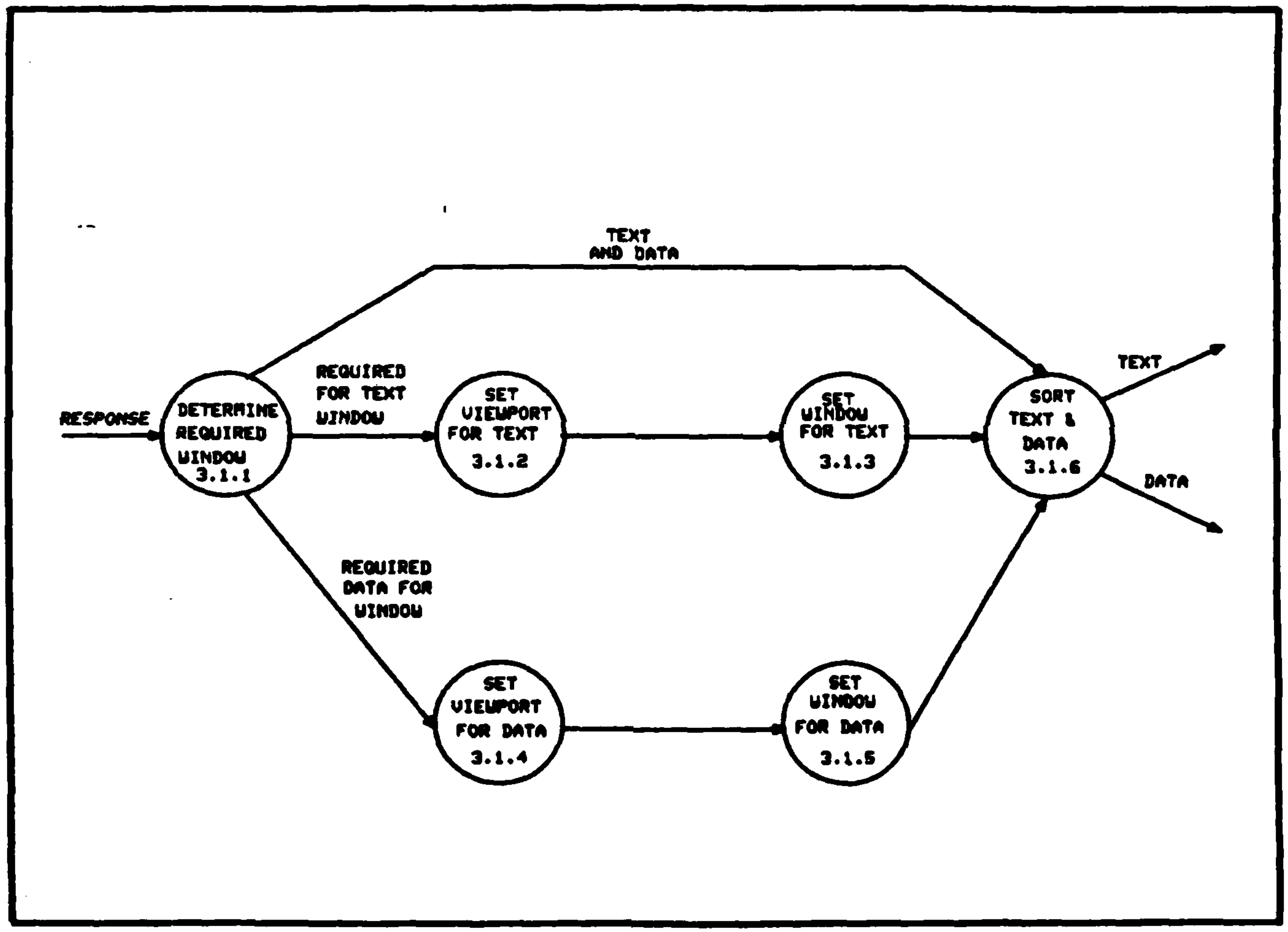


FIGURE F-19. ESTABLISH WINDOWS AND VIEWPORTS (NODE 3.1)

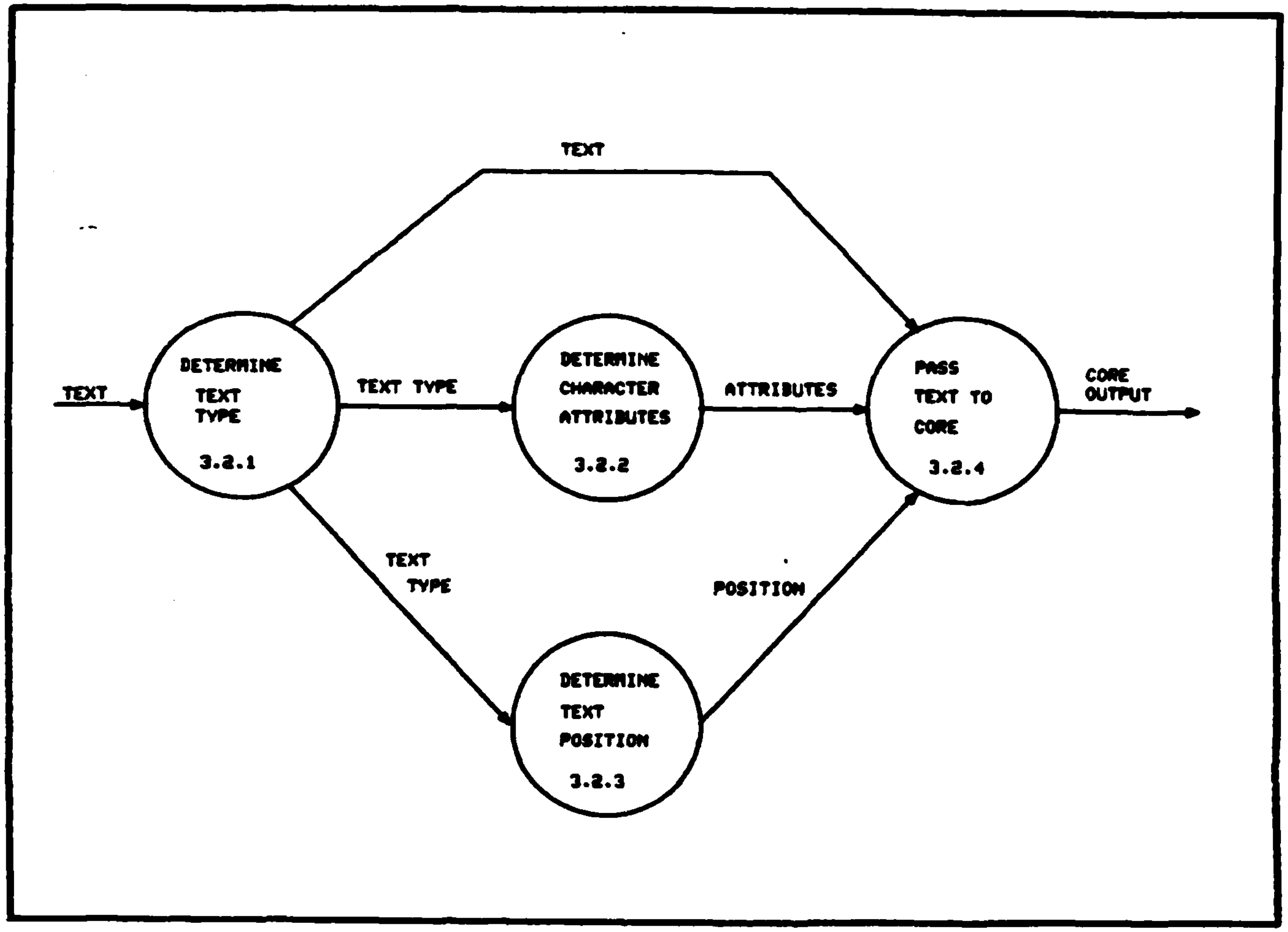


FIGURE F-20. DISPLAY TEXT (NODE 3.2)

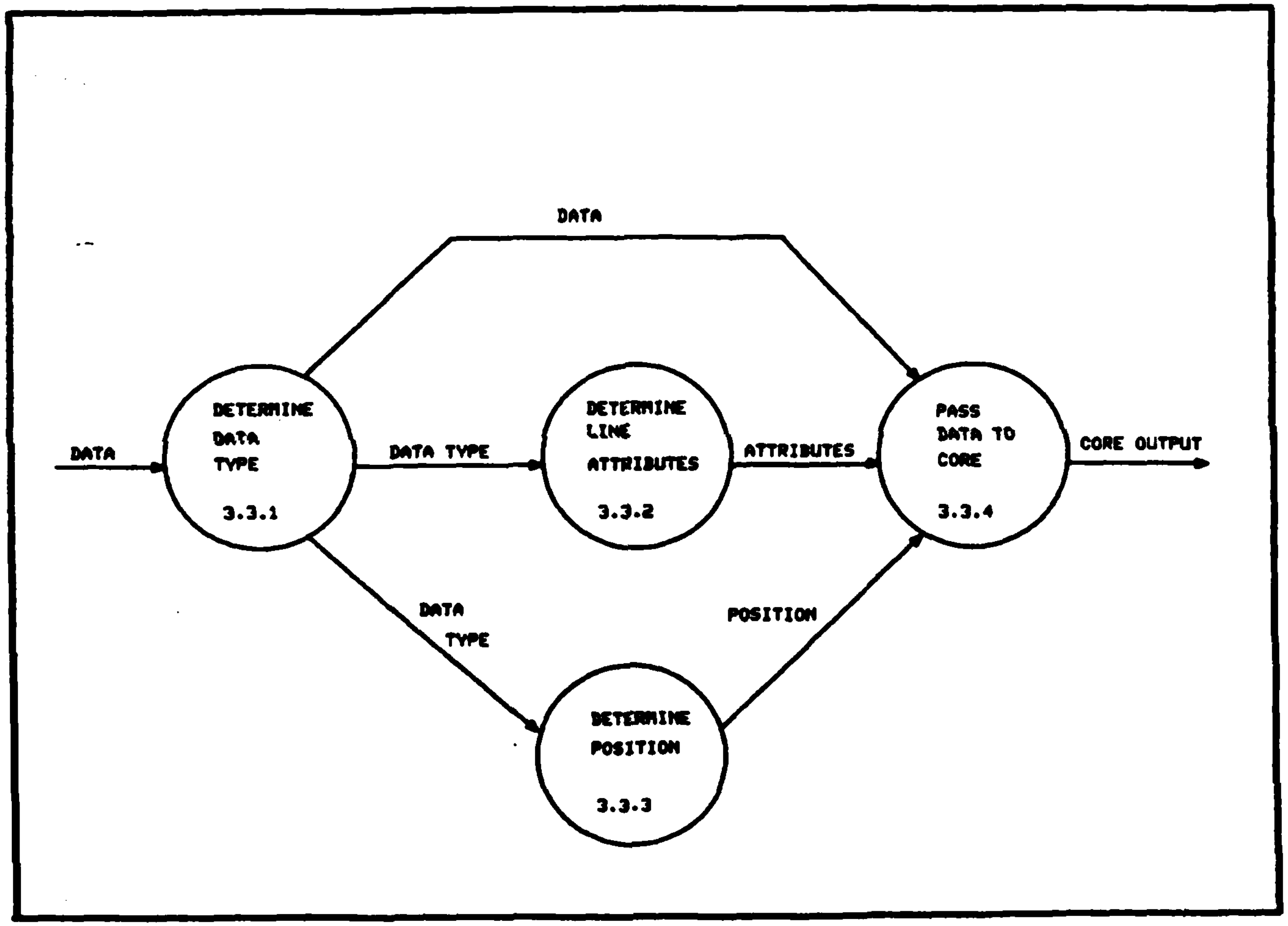


FIGURE F-21. DISPLAY DATA (NODES 3.3 - 3.5)

F.3 Summary

This appendix has presented important design docum[illegible]t-ation for Graphical ICECAP. This data flow diag[illegible]ms document this thesis's effort and is a starting point [illegible]or future efforts.Specific module descriptions are provided in appendixes A, B, and C.

APPENDIX G

ICECAP Source Code Location

G.1 Introduction

ICECAP contains two program listings in which one is written in Pascal and the other is FORTRAN. Together these two programs make-up ICECAP. Both of these listings are too large to be included in this document. This appendix describes the location of each listing and provides a point of contact (24).

G.2 Location of Pascal Code

ICECAP's Pascal source code is contained in a single file named ICER.PAS. This file is located under the directory entitled [ICECAP1.MODULES]. The user directory [ICECAP1] is located on winchester drive DUA1 and contains all the files related to ICECAP. The contents of this directory have been stored on magnetic tape (under the file name AFITUSER.BAK) in the AFIT Digital Equipment Laboratory on the VAX 11/780 system.

G.3 Location of FORTRAN Code

Included in the directory, [ICECAP1.MODULES], is ICECAP's FORTRAN source code. The FORTRAN code has several extensions. The original code, which came form VAXTOTAL, has the ".DEK" extension. The modules which have been modified by reference (24) contain the extension ".ICE" and new modules of restructured modules contain the extension ".ABE". The last extension ".stu" is given to source files that serve no purpose exececpt to act as a "stub" to satisfy a subroutine call (24).

G.4 Point of Contact

A complete listing of both source codes is on file in the AFIT Digital Equipment Laboratory under the name ICECAP. Points of contact are Dr. Gary B. Lamont (AV 785-2024 Commerical: (513) 255-2024) and Mr. Robert L. Ewing.

BIBLIOGRAPHY

1. Larimer, Stanley J. An Interactive Computer Aided Design Program for Digital and Continous Control System Analysis and Synthesis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, Mar 1978 (AD-A055 418).

2. Logan, Glen T. Development of an Interactive Computer Aided Design Program for Digital and Continuous Control System Analysis and Synthesis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1982 (AD-A118 042).

3. Gembarowski, Charles J. Development of an Interactive Control Engineering Computer Analysis Package for Discrete and Continuous Systems. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1982 (AD-A124 707).

4. Travis, Mark A. Interactive Computer Graphics for System Analysis. MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, 1983 (AD-A138 025).

5. Dorf, Richard C. Modern Control Systems (Second Edition). Reading, Massachusetts: Addison-Wesley Publishing Company, 1974.

6. Aburdene, Maurice F. and Sheri Surchek. "BUCOP: A Computer-Aided Design Control Systems Package," CAD/CAM IEEE, 7 :29-36 (1984).

7. Foley, James D. and Victor L. Wallace, "The Human Factors of Computer Graphics Interaction Techniques," IEEE Transactions on Computer Graphics, Nov 1984.

8. Foley, James D. and A. Van Dam Fundmentals of Interactive Computer Graphics, Addison-Wesley Publishing Co., 1982.

9. Scott, J. D Introduction to Computer Graphics, Addison-Wesley, 1982.

10. Artwich, Bruce A. Applied Concepts in Microcomputing Graphics, Prentice-Hall, 1984.

11. Taylor, Robert W. "Display-Selection Techniques for Text Manipulation," IEEE Transactions on Human Factors in Electronics, HFE-8 :5-15 (1967).

12. Newman, William M. and Robert F. Sproul <u>Principles of Interactive Graphics,</u> McGraw-Hill, 1979.

13. Weinberg, Victor <u>Structured Analysis,</u> Yourdon Press, 1979.

14. Cassel, Don <u>The Structured Alternative: Program Design, Style, and Debugging,</u> Prentice-Hall, 1983.

15. McGowan, Clement L. and John R. Kelly <u>Top-Down Structured Programming Techniques,</u> Mason/Charter, 1976.

16. Martin ,D. and G. Estin, "Models of Computation and Systems-Evaluation of Vertex Probabilities in Graph Models of Computation," <u>Journal of the ACM, 14</u> :281-299, 1967.

17. Woffinden, Dourd S. <u>An Interactive Environment for a Computer-Aided System</u> . MS Thesis. Naval Postgraduate School, Monterey, California, June 1984.

18. PLOT-10 User's Manual. Beaverton, OR: Tektronix Inc., 1971.

19. Keller, Pete, et al. GRAFLIB Reference Manual. Livermore, CA: Lawrence Livermore Laboratory, 1980.

20. Tarbell, Phillip B. <u>Continued Development and Implementation of a Standard Graphics Package for the AFIT VAX 11/780.</u> MS Thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1981.

21. Reference Manual for GWCORE, George Washington University, Washington D. C., 1981.

22. Simpson, Henry "A Human-Factors Style Guide for Program Design", <u>Byte Publications Inc.,</u> April 1982.

23. Miller, George A. "The Magic Number Seven, Plus or Minus Two: Some Limits on out Capacity for Processing Information", <u>Psychology Review 63</u> (2): 81-97, March 1956.

24. Wilson, Robert E. <u>Continued Developement of an Interactive Control Engineering Computer Analysis Package (ICECAP) for Discrete and Continous Systems.</u> MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1983.

25. Armold, Abraham T. <u>Further Development of an Interactive Control Engineering Computer Analysis Package (ICECAP) for Discrete and Continus Systems.</u> MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1984 (AD-A151 783).

26. Yourdon, Edward and Larry L. Constantine. Structured Design, 2nd ed. New York: Yourdon Press, 1978.

27. Thompson, Peter M. User's Guide to Program CC, Version 3. Aeronautical Systems Division, Air Force Systems Command. Contract F33657-83-C-0242 with Systems Technology, Inc. Wright-Patterson AFB, OH, March 1985.

28. Narathong, Chiewcharn A Modern Control Theory Enhancement Tool to an Interactive Control Engineering Computer Analysis Package (ICECAP). MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson ARB, OH, December 1984.

29. D'Azzo, John J. and Constantine H. Houpis. Linear Control Systems Analysis and Design (Second Edition). New York: McGraw-Hill Book Company, 1981.

30. Raeder, Georg "A Survey of Current Graphical Programming Techniques," lEEE Computer, 23 :11-25 (August 1985).

31. Digital Equipment Corporation. GKS User's Manual. Maynard, Massachusetts: Digital Equipment Corporation, June 1985.

32. Rose, Kevin W. Development of an Interactive Computer Graphics System Library and Graphics Tools. MS thesis. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB, OH, December 1982.

VITA

John R. Bullard was born on October 25, 1952 in Oswego, New York. In 1970, he graduated from G. Ray Bodley High School, located in Fulton, New York. He entered the Air Force in November 1971, as an airman basic. After military basic training, he stayed at Lackland AFB for electronic training before he was sent to Griffiss AFB, New York. He left Griffiss AFB in May 1975, and returned to Lackland AFB to perform instructor duty at the Electronic Technical School. During his assignment there, he attended San Antonio College, part-time. After completing 35 semester hours of credit, he applied for schooling through the Airman Education Commissioning Program (AECP). He was accepted in October 1977, and started school in January 1978 at Texas A&M University. After Graduation, in May 1980, he was assigned to ASD/ENA, Wright-Patterson AFB. After working in the engineering branch of the F-16 Program Office for three years, he was re-assigned to the Air Force Institute of Technology (AFIT). He is married to Linda Ann Beauch Bullard, and has two daughters Deborah and Elizabeth.

Permanent address: 7534 Westshire Drive
San Antonio, Texas 78227

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/EE/85D-5	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering — 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Air Force Institute of Technology Wright-Patterson, AFB, Ohio 45433	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION — 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO. / WORK UNIT NO.
11. TITLE (Include Security Classification) See Box 19	

12. PERSONAL AUTHOR(S)
John R. Bullard, B.S.E.E., Capt, USAF

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT (Yr., Mo., Day)	15. PAGE COUNT
MS Thesis	FROM ______ TO ______	1985 December	246

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB. GR.	ICECAP, COMPUTER GRAPHICS, INTERACTIVE GRAPHICS CONTROL SYSTEMS, CORE STANDARD, SYSTEM ANALYSIS
09	02		

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

Title: Interactive Computer Graphics for Analysis and Design of Control Systems

Thesis Chairman: Dr. Gary B. Lamont

Approved for public release: IAW AFR 190-1.
16 JAN 86
LYNN E. WOLAVER
Dean for Research and Professional Development
Air Force Institute of Technology (AEC)
Wright-Patterson AFB OH 45433

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED XX SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL Dr. Gary B. Lamont	22b. TELEPHONE NUMBER (Include Area Code) 513-255-3576 — 22c. OFFICE SYMBOL AFIT/ENG

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE

19. ABSTRACT

This thesis reports on the on-going effort to design and implement an Interactive Control Engineering Computer Analysis Package (ICECAP). When fully implemented this package will allow control engineers to design and analyze continuous and discrete control problems. This project was started by Captain Glen T. Logan's implementation of TOTAL. Captain Charles J. Gembarowski continued Logan's effort and began implementation of the "user friendly" command structure. Captain Mark Travis modified the graphical output and incorporated the use of a graphical package called GWCORE. Captain Robert E. Wilson implemented the help/teach modules and completed the continuous time functions started by Gembarowski. Captain Abraham Armold provided the discrete command structure so that discrete analysis and design could be performed by ICECAP.

The main emphasis of the thesis investigation was to implement an interactive graphical input routine which would complement the DEFINE commmand which exists in ICECAP.

END

FILMED

3-86

DTIC