IMPLEMENTATION AND EVALUATION OF A CORE GRAPHICS SYSTEM ON A VAX 11/780 C. (U) AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGI. J W TAYLOR UNCLASSIFIED DEC 83 AFIT/GCS/MA/83D-8
Implementation and Evaluation of a Core Graphics System on a VAX 11/780
with a UNIX Operating System

Thesis

AFIT/MA/606/83D-8 John W. Taylor
Capt USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY (ATC)
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio
Implementation and Evaluation of a
Core Graphics System on a VAX 11/780
with a UNIX Operating System

Thesis

AFIT/MA/668/83D-8 John W. Taylor
Capt USAF

Approved for public release; distribution unlimited
Implementation and Evaluation of a Core Graphics System on a VAX 11/780 Computer with a UNIX Operating System

Thesis

Presented to the Faculty of the School of Mathematics of the Air Force Institute of Technology Air University in Partial Fulfillment of the Requirements for the Degree of Master of Science

by

John W. Taylor
CAPT USAF
Graduate Computer Science
December 1983

Approved for public release; distribution unlimited.
Preface

This report describes the implementation and evaluation of the University of Pennsylvania Pascal Core System graphics package for use on a VAX 11/780 computer with the UNIX operating system. Device drivers for the Tektronics 4014, Visual 550, and Hewlett Packard 7220A plotter have been developed and implemented for this Core package.


I would like to thank my thesis advisor, Professor Charles Richard, for his guidance and help on this project. Lt. Col Hal Carter, Dr. Gary Lamont, and Mr. Joe Hamlin were very helpful with their comments and criticisms of this project.
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preface</td>
<td>ii</td>
</tr>
<tr>
<td>List of Figures</td>
<td>vi</td>
</tr>
<tr>
<td>List of Tables</td>
<td>vii</td>
</tr>
<tr>
<td>Abstract</td>
<td>viii</td>
</tr>
<tr>
<td>I. Introduction</td>
<td>I-1</td>
</tr>
<tr>
<td>Background</td>
<td>-1</td>
</tr>
<tr>
<td>Problem Definition</td>
<td>-1</td>
</tr>
<tr>
<td>Review of Literature</td>
<td>-2</td>
</tr>
<tr>
<td>Objective of Thesis</td>
<td>-3</td>
</tr>
<tr>
<td>Scope</td>
<td>-3</td>
</tr>
<tr>
<td>Approach</td>
<td>-5</td>
</tr>
<tr>
<td>Past Development Efforts</td>
<td>-6</td>
</tr>
<tr>
<td>Overview of Thesis</td>
<td>-6</td>
</tr>
<tr>
<td>II. System Design</td>
<td>II-1</td>
</tr>
<tr>
<td>Core System</td>
<td>-1</td>
</tr>
<tr>
<td>Pascal Core System</td>
<td>-1</td>
</tr>
<tr>
<td>GKS System</td>
<td>-2</td>
</tr>
<tr>
<td>Computer Hardware</td>
<td>-2</td>
</tr>
<tr>
<td>Graphic Devices</td>
<td>-2</td>
</tr>
<tr>
<td>Hewlett Packard 7220A 4 Color Plotter</td>
<td>-3</td>
</tr>
<tr>
<td>Visual 550 Raster Graphics Terminal</td>
<td>-4</td>
</tr>
<tr>
<td>Tektronics 4014 Vector Graphics Terminal</td>
<td>-4</td>
</tr>
<tr>
<td>Computer Languages Interfacing</td>
<td>-5</td>
</tr>
<tr>
<td>III. Detailed Design and Implementation</td>
<td>III-1</td>
</tr>
<tr>
<td>Pascal Core System</td>
<td>-1</td>
</tr>
<tr>
<td>Conversion from VAX VMS to VAX UNIX</td>
<td>-1</td>
</tr>
<tr>
<td>Operating System</td>
<td>-2</td>
</tr>
<tr>
<td>Pascal Code Changes</td>
<td>-2</td>
</tr>
<tr>
<td>Pascal Code Additions</td>
<td>-3</td>
</tr>
<tr>
<td>Library Files</td>
<td>-4</td>
</tr>
<tr>
<td>Compile Files</td>
<td>-5</td>
</tr>
<tr>
<td>Include Files</td>
<td>-5</td>
</tr>
<tr>
<td>Subroutine Dependencies</td>
<td>-7</td>
</tr>
<tr>
<td>Duplicated Variables</td>
<td>-9</td>
</tr>
<tr>
<td>Graphic Devices Used</td>
<td>-9</td>
</tr>
<tr>
<td>Hewlett Packard 7220A 4 Color plotter</td>
<td>-10</td>
</tr>
</tbody>
</table>
Appendix D: Pascal Core System Additions ........ D-1
Appendix E: User Documentation for Pascal Core System based on SIGGRAPH August 1979 Core Standards ........ E-1
Appendix F: Core Routines Calling/Called By Other Core Routines ........ F-1
Appendix G: Device Driver Source Code ........ G-1
Appendix H: VAX 11/780 and CDC 6600 HP 7220A Calcomp Users Manual Supplement ........ H-1
Appendix I: Pascal, C, and FORTRAN Program Compatibility/Incompatibility Examples . I-1
Appendix J: Duplicated Variables Created when Unique Underscore Identifier is Removed from the University of Pennsylvania Core Variables ........ J-1

VITA ............................................. 1
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I-1</td>
<td>Levels of Implementation</td>
<td>I-4</td>
</tr>
<tr>
<td>II-1</td>
<td>Compatible and Incompatible Programming Languages</td>
<td>II-8</td>
</tr>
<tr>
<td>IV-1</td>
<td>Two and Three Dimensional CPU Timing Test, System Saturated</td>
<td>IV-3</td>
</tr>
<tr>
<td>IV-2</td>
<td>Two and Three Dimensional CPU Timing Test, Five Users or Less on System</td>
<td>IV-4</td>
</tr>
<tr>
<td>IV-3</td>
<td>Two and Three Dimensional Clock Timing Test, System Saturated</td>
<td>IV-5</td>
</tr>
<tr>
<td>IV-4</td>
<td>Two and Three Dimensional Clock Timing Test, Five Users or Less on System</td>
<td>IV-6</td>
</tr>
</tbody>
</table>
List of Tables

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>II-1</td>
<td>Data Type Compatibility Among Pascal, FORTRAN, and C.</td>
<td>II-9</td>
</tr>
</tbody>
</table>
Abstract

A Pascal Core System using the VAX VMS 11/780 operating system from the University of Pennsylvania was converted to run on the Air Force Institute of Technology (AFIT) VAX 11/780 UNIX operating system. Major problems in converting from the VMS operating system to the UNIX operating system were encountered, but they were solved and documented for others to use and avoid. The size of the Pascal Core System and its lengthy compile times were a major system limitation. Device drivers were written in C for the Visual 550, the Tektronics 4014, and the Hewlett Packard 7220A 4 color plotter and can be applied to other projects. The Core package that was converted follows the SIGGRAPH 1979 standard. An application program, called intcore.run, is available that demonstrates the Pascal Core routines that are supported by this system.
I. Introduction

Background

Currently, no comprehensive graphics software capability exists on the Air Force Institute of Technology's (AFIT) VAX 11/780 computer with the UNIX operating system located in the School of Engineering (EN) computer room. Limited graphics capability, such as the Tektronics Plot 10 and the S package, does currently exist on the system, but this capability was not initially delivered with the new computer system. The newness of this VAX 11/780 system is one reason for the limited graphics capability.

More comprehensive graphics packages and device drivers do exist, but they were developed to run on the CDC 6600 computer, which has been in place for several years. But as more and more graduate students learn to use the VAX 11/780 computer system, they also need a similar graphics capability on the VAX.

Problem Definition

Since no comprehensive graphics software capability does currently exist on the VAX UNIX system, this thesis effort will be to implement and evaluate the University of
Pennsylvania's Pascal graphics software package for possible use on this system. The graphics package that will be evaluated is based on the Core System standard of 1979 [1].

Also, only a very limited number of graphics devices currently exist. This thesis effort will also attempt to make more graphics devices usable on the VAX system.

Review of Literature

The Pascal Core System graphics package has been obtained from the University of Pennsylvania for evaluation on the VAX. This package has been chosen because it is the most complete Pascal Core system available at present.

Other graphics packages are:

a. Tektronics Plot-10 package for possible use on the Visual 550 and the Tektronics 4014.

b. Movie BYU for possible use on the Visual 550 and the Tektronics 4014.

c. Simple Graphics Package (SGP) for possible use on the Visual 550 and the Tektronics 4014.

d. Hewlett Packard Calcomp Package for possible use on the Hewlett Packard 7220A 4 color plotter.

e. Vpr utility package for possible use on the Versatec plotter.

f. Graph(), and Plot() utility packages for possible use on the Visual 550 and the Tektronics 4014.

g. S package written in Ratfor for possible use by the Hewlett Packard plotter and the Tektronics 4014. When in the Tektronics 4014 emulation mode, the Visual 550
Objective of Thesis

The objective of this thesis is to:

i) Convert the University of Pennsylvania's (UP) existing device independent package based on the proposed U.S. Standard Core System in Pascal to execute on a UNIX VAX.

ii) Design and implement device drivers for the three graphic devices available in the EN computer facility to interface with the Core packages.

iii) Implement and evaluate this package for performance and use of the varied features.

iv) Design and test a typical application to integrate and demonstrate the compared features.

Scope

This report presents the rehosting of the UP Core System graphics package on the VAX 11/780 computer using the UNIX operating system.

This project develops a graphics system that can be used by other application programs at AFIT to manipulate graphics. Existing software is used as much as possible.

Basically, there are three levels in graphics software (Figure I-1). Level 1, the highest level, is the applications program, and is machine independent.
Figure I-1. Levels of Implementation
Level 2, the interface level, is the Core System package. It controls the graphics commands from the Level 1 applications programs to the device dependent drivers at Level 3. Level 2 is also machine independent. Level 3 is the lowest level. It contains machine dependent routines that translate graphics commands from Level 2 into commands for the actual graphics devices.

Level 1

One or more applications programs will be designed in Pascal to demonstrate and evaluate the Core System package once the Level 2 and Level 3 implementations are operational.

Level 2

This is where the Core System will be implemented. The Pascal package will be developed to interface between the applications programs and the graphics drivers. This package will be transportable to other computers.

Level 3

The graphics interface will be done here. This project will interface with three graphic devices. The three devices are, Visual 550 raster graphics terminal, Tektronics 4014 vector graphics terminal, and the Hewlett Packard 7220A 4 color plotter.

Approach
I will follow these steps to solve this problem:

1. Literature search to locate an existing Core System package in Pascal, and any proposed international standard GKS systems that are currently available.

2. Convert the selected Pascal Core System package to run on the VAX 11/780 computer with its UNIX operating system.

3. Implement software drivers for the three graphic devices.

4. Develop the means for FORTRAN 77, Pascal, and C software to communicate between any combination of these languages.

5. Write application programs to demonstrate these Core System features.

6. Evaluate the performance of this core system on the VAX 11/780 using the UNIX operating system.

Past Development Efforts

Past development efforts at AFIT were done by Harold L. Curling (Dec 1980), Philip Boman Tarbell (Dec 1981), and Capt Rose (Dec 1982). These projects were done on a VAX 11/780 computer using the VMS operating system.

Overview of Thesis

The next chapter describes the Core System packages that are proposed, along with the possible graphic device drivers. In this chapter, possible ways to implement this graphics system are explained. The advantages and the
disadvantages of each are presented. The third chapter specifies the system configuration that has been chosen. The fourth chapter provides the performance and functional analysis to evaluate the Pascal Core System package that has been implemented. Chapter five presents the results of the analysis from the previous chapter, conclusions reached, and recommendations for future work.
II. System Design

The proposed Core System package used in this project is presented. Also, the GKS System (Graphic Kernal Standard, the European Graphics Standard) is discussed briefly. The possible computer hardware to be used for this Core System package, along with possible graphics device drivers, are discussed.

Core System

The Core System, with a Pascal package, is used for this thesis effort. This Core System was chosen over the GKS System because of the more developed Core packages available.

Pascal Core System

From literature searches and past development efforts at AFIT, the University of Pennsylvania Pascal Core System package has been chosen for this thesis development effort. It is a complete 3-D Core package written on a VAX 11/780 computer using a VMS operating system. The Apple Computer Core package, and the European Core package from the University of Glasgow were not chosen because they were not as complete as the University of Pennsylvania Core package [9,10,12,20,22,34].

Problems of rehosting this Pascal Core package occurred. One such problem was converting the magnetic tape containing this package over to AFIT's UNIX operating system.
this was done, then the Pascal programs were corrected to run on the UNIX Pascal system. Another problem was linking graphic device driver software to the Pascal Core system.

GKS System

The GKS System (Graphic Kernal Standard, European graphic standard) was considered early in the planning stages of this project. However, because no GKS system was readily available at the time this project was planned, it was decided to use the Core System packages available to us. At the most recent meeting of the GKS Committee, the final standards for the GKS System had not been agreed on [11,12,14,15,27].

Computer Hardware

Currently, the EN Computer facility has access to several computer systems. The following main computer systems are:

1. VAX 11/780
2. CDC 6600
3. DEC PDP 11/60
4. Harris

The VAX 11/780 computer will be used for this thesis effort.

Graphic Devices

There are several terminals, printers, and plotters available in the EN computer facility at AFIT. For this thesis effort, not only will graphics devices have to be
chosen, but also graphic software to run the chosen devices.

The available graphic devices are:

1. Visual 550 raster graphic terminal
2. Tektronics 4054 vector graphic terminal
3. Versatec raster plotter
4. Hewlett Packard 7220A 4 color plotter
5. Tektronics 4014 vector graphic terminal

The graphic devices to be used in this effort are:

1. Hewlett Packard 7220A 4 color plotter
2. Visual 550 Raster Graphics Terminal
3. Tektronics 4014 Vector Graphics Terminal

These graphic devices have been chosen because they are available to students at any time, and they are the most used. See Figure I-1 for more information on where these devices will fit into this project.

**HP 7220A 4 Color Plotter**

The HP plotter will be used interactively. One of the device drivers that might be used for this plotter is a Calcomp package that was originally used on the CDC 6600 computer. It is written in FORTRAN IV and has been converted to FORTRAN 77 to run on the VAX. See Appendix H for further information on this Calcomp package.

The following graphics capabilities for this plotter are being developed:

1. Draw a line (pen down position)
2. Select one of four possible pens (4 colors)
3. Move the pen (pen up position)
4. Write a string of text
5. Cross hatching
6. Read pen position (x,y coordinates)
7. Set the character size
8. Initialize the plotter
9. Put the plotter online or offline
10. Place the pen in the up or down position

**Visual 550 Raster Graphics Terminal**

The Visual 550 terminal will be used interactively.

The following graphics capabilities are being developed for the Visual 550 terminal:

1. Write a string of text
2. Graphics/Text screen
3. Clear the screen
4. Draw a line (point A to point B)
5. Fill a closed polygon
6. Move the cursor
7. Read the cross hair position (x,y coordinates)
8. Initialize the terminal
9. Mode selection (pixels on or off)
10. Set the character size

**Tektronics 4014 Vector Graphics Terminal**

The Tektronics 4014 terminal will be used interactively.

The following graphics capabilities are being developed for the Tektronics 4014 terminal:
1. Text/Graphics screen
2. Clear the screen
3. Draw a line (point A to point B)
4. Move the cursor
5. Write a string of text
6. Initialize the terminal
7. Read the cross hairs (x,y coordinates)
8. Set the character size

Computer Languages Interfacing

A critical part of this thesis design effort is to be able to interface the Pascal Core System programs with the device dependent graphic software programs. If the Pascal Core System programs are incompatible with the several existing graphic software programs, then a way to communicate with these graphics packages has to be found. Currently, the graphic software programs are written in either FORTRAN 77 or in C.

A Core System main program written in FORTRAN 77 can interface with the FORTRAN 77 graphic software packages without the need for any special links. For the graphics packages written in C, then a C subroutine must be written to pass data arguments on to the C graphics package.

A Pascal Core System main program can interface with a Pascal software package without any special modifications. Also, for graphics packages written in C, a C subroutine can pass the data arguments on to the C graphics package.

The AFIT VAX 11/780 computer with its UNIX operating
system cannot pass I/O data to and from FORTRAN 77 subroutines, using either C or Pascal main programs, unless major system modifications are made.

The FORTRAN 77 system automatically places an underscore "_" character after program, subroutine, or function names. So for a FORTRAN 77 program to communicate with either a Pascal or C routine, then an underscore character must be inserted by the user after the Pascal or C routine name. This works well for a C routine, but the Pascal system will not allow or recognize underscore characters. So for a FORTRAN 77 program to communicate with a Pascal routine, then a C interface program must be written. FORTRAN 77 can communicate with a Pascal routine in this manner, however, a Pascal program cannot communicate with a FORTRAN 77 routine in this manner. When a C interface routine tries to call a FORTRAN 77 routine, then a "bus error" occurs when FORTRAN I/O is attempted.

The current way of passing data arguments to and from a C interface program is by reference. This allows the FORTRAN 77 and Pascal programs to pass data arguments to and from the C interface routines using pointers or addresses.

Appendix I shows how to set up data arguments for a FORTRAN 77 to C to Pascal routine, and for a Pascal to C routine. Currently, no way exists to pass data from a C routine to a FORTRAN 77 routine and back.

Figure II-1 outlines the compatible and incompatible programming languages.

Table II-1 shows the data type relationships between
Pascal, FORTRAN 77, and C.
Figure II-1. Compatible and Incompatible Programming Languages
<table>
<thead>
<tr>
<th>PASCAL DATA TYPES</th>
<th>FORTRAN DATA TYPES</th>
<th>C DATA TYPES</th>
</tr>
</thead>
<tbody>
<tr>
<td>REAL</td>
<td>REAL</td>
<td>FLOAT</td>
</tr>
<tr>
<td>REAL</td>
<td>DOUBLE PRECISION</td>
<td>DOUBLE</td>
</tr>
<tr>
<td>CHAR</td>
<td>CHARACTER</td>
<td>CHAR</td>
</tr>
<tr>
<td>--------</td>
<td>INTEGER*2</td>
<td>SHORT INT</td>
</tr>
<tr>
<td>INTEGER</td>
<td>INTEGER</td>
<td>LONG INT OR INT</td>
</tr>
<tr>
<td>BOOLEAN</td>
<td>LOGICAL</td>
<td>LONG INT OR INT</td>
</tr>
</tbody>
</table>

Table II-1. Data Type Compatibility Among Pascal, FORTRAN, and C
III. Detailed Design and Implementation

This chapter will describe in detail the conversion and implementation of the VAX 11/780 Pascal Core System, designed for the VMS operating system from the University of Pennsylvania, to the Air Force Institute of Technology (AFIT) VAX with its UNIX operating system.

Pascal Core System

The conversion of the University of Pennsylvania Core package was complex because of the many conflicts between the VMS Pascal and the UNIX Pascal. Using the AFIT VAX C shell and text editor, all of the approximately 300 pascal subroutines had to be modified to be usable on the UNIX VAX system.

Conversion from VAX VMS to VAX UNIX Operating System

The major portion of the conversion required changing the VMS Pascal to run on the UNIX Pascal. Changes to the variables, file names, and system routines not supported on the UNIX VAX, were some of the many changes that had to be made.

Routines had to be added to support multiple graphics devices that were not implemented by the University of Pennsylvania Pascal Core System.

Library files and their dependencies had to be researched and restructured due to the requirements of the UNIX VAX Pascal, which was not a problem under the VMS VAX Pascal.
system. This structure of dependencies is critical to the UNIX VAX Pascal because of the one-pass loader.

The Include files also had to be structured due to the Pascal language restrictions of subroutines only knowing the existence of other routines above them.

Duplicated variables were a problem because the VMS VAX Pascal accepts underscore characters, while the UNIX VAX Pascal does not. Removing the underscore characters sometimes caused duplications with other variables originally created without the underscore character. These duplications had to be renamed to eliminate multiple defined errors.

Graphic devices chosen for this project were selected from the ones available in the AFIT Computer room. The two selected were chosen for their availability and familiarity to students, and research to develop these graphics drivers was more complete.

The graphic device drivers were researched and written specifically for the Core System. Other graphics packages were analyzed, but due to the size constraint of the Pascal Core System on the UNIX VAX, these graphics packages were too large. To save disk space, it was decided to develop smaller, more specific device drivers.

Pascal Code Changes

The changes made to convert the VMS VAX Core System to run on the UNIX VAX Core System is detailed in Appendix C.
Most of these changes were made using the UNIX C shell. This system shell allows a user to create an executable file that will automatically make changes to the selected text files. In this way, with the files in one directory, files can be updated automatically without having to use a text editor to make changes for each specific file. Some of the more specific changes did have to be made on an individual basis, but these were kept to a minimum.

**Pascal Code Additions**

Appendix D contains a list and description of the Core routines added to this system to run on the UNIX operating system.

The device dependent routines were added because the UNIX Pascal code cannot access the FORTRAN 77 device drivers written for the VMS system.

Since the University of Pennsylvania Core System only supported one graphics device, routines were added to support multiple devices. But due to the size problem encountered, the system was modified to support multiple devices, but only have one device active at a time. This is a compromise, but this will save space and allow the use of more than one device while the Core program is active. Currently the AFIT VAX system only allows one device per user while the program is running, but the changes in the program will allow for more than one device at a future date.
Library Files

The UNIX Pascal library file is listed in Appendix B. It shows the order of dependencies that must be followed when building this library file. The most dependent modules are at the bottom while the least dependent modules are at the top of the file.

The UNIX libraries use a one pass loader. The UNIX system scans a library from top to bottom in one pass, so the routines that are called by other routines must follow them. If not, then an undefined error may occur when a Pascal program is compiled. The order of the current library ensures that all the dependent routines are organized to follow the calling routines.

The userext.h files are independent of each other and can be in any order in the library. The rest of the files must be in the stated order or an undefined error may occur.

The UNIX system has a routine that randomizes the library so that the order of the modules in the library does not matter. It creates an index at the beginning of the file to list all the modules in the particular library. When a program is loaded using this library, it then searches this index for the required module. However, there is a size limit to this index. If a library contains more than approximately 100 modules, then this index space may be exceeded. The Core System uses about 300 modules, so this randomizer cannot be used. When compiling without this index, a warning message appears on the screen stating the
library date is not correct. This is only a warning and will occur every time a compile is run using this library file.

**Compile Files**

Appendix A contains a list of all the UNIX files that need to be compiled. This file automatically compiles the files that have been changed. Only the files that need to be updated are compiled.

This C shell file checks the time and date to see if any changes have been made to the text file or if an object file exists. If any changes have been made or an object file no longer exists, then the file is compiled. If only one of the files has been changed, then this C shell compile file is run and only the text file that has been changed will be recompiled, the others will be checked and ignored.

**Include Files**

Include files are needed in Pascal to define the variables and procedures called by that Pascal routine. Instead of typing in all necessary variables and procedures, the Include files allow a user to type in a single line to define several other variables or procedures. This allows different programs and routines to access other routines.

The following .h Include files are ordered in the Pascal procedure orientation, where procedures and functions only recognize other routines that are above them. This order allows the user to easily include these files into their
programs with the dependencies in the correct order. The top most Include file is the most dependent, while the bottom file is the least dependent. The modules listed in the userext.h file are independent of each other and can appear in any order within this file. However, the other .h files are very dependent, and their ranking is vital. The following order is:

1. error.h
2. driverext.h
3. ddutilext.h
4. ctlOext.h
5. ctl1ext.h
6. util2ext.h
7. utilext.h
8. common.h
9. userext.h

Include files were built using .h instead of .i file specifications. The .i file specification is treated as text, but requires too much space. So the .h file specification is used since it requires less space.

The original VMS Pascal Include files were not structured as shown in this appendix. Error.h and Common.h include files were added to the already existing ones. This was done to ensure the proper dependencies were maintained. Also, the order had to be established to ensure that the most dependent routines were above their calling routines.

The routines within each Include file had to be
restructured to also maintain the correct calling sequence. This was not done for the VMS Pascal Core System.

The Core routines only have the necessary Include files in their code in order to save space in the program and on the disk. The application programs will have all of these Include files in their program to ensure that any Core routine that is called is defined. This is a tradeoff between saving space on the system and easier usage for a user writing an application program. This way a user does not have to determine what Core routines he will be using, the routines will already be defined for him.

The other Include files to be used define constants (defconst.h), types (deftype.h), and variables (extvar.h or gblvar.h). These are global data types and must be in all routines and application programs.

Subroutine Dependencies

To develop this Core package for the UNIX operating system the structure of the Core routines had to be determined. A way to determine what routine called what other routine or routines had to be determined in order to convert from the VMS operating system to the UNIX operating system.

Structured Analysis and Design Technique (SADT) diagrams were first considered to try to show the dependencies of all the routines and their interaction. But this was found early on to be too confusing. Ultimately, SADT diagrams were used to show the overall Core system at a very high

III-7
level, but to go to any lower levels would have been too cumbersome. A combination of SADT diagrams and Structure charts were considered but dropped for the same reasons.

Structure charts were tried but, with approximately 300 subroutines to make charts and establish the dependencies, this would have taken too many charts. It was tried, but a rough outline of these routines in structure chart form were too complex and cumbersome to prove very useful. It would have taken too much effort to trace the calling steps of one routine from the independent core side to the dependent device driver following the structure charts.

Appendix F shows how the final interdependencies were shown. This appendix shows the connections of the desired routines in an ordered structure in only a few pages. It shows each routine, what that routine does, who it calls, and who it is called by. It is structured by Include file, with the most dependent files at the beginning and the least dependent files at the end of each Include file. This way the routines are organized so someone can easily trace a routine and its dependencies from start to finish. The routines are organized so that to locate the "called" routines you only have to look up, and to locate "called by" routines you only have to look down. This way a user only has to look in one direction to follow any string of dependencies. This has been one of the most useful appendices for converting from the VMS to the UNIX operating system.

III-8
Duplicated Variables

As shown in Appendix C, the underscore character is accepted in the VMS Pascal system, but not in the UNIX Pascal system. This required the removal of the underscore character from all routines and variables. Programs at first did not work as they should have because of this.

A way to isolate and change duplicate names had to be found. A database of all VMS variables and a database of all variables with the underscore character removed were compared for duplicates. Once these duplicates were found their usage had to be determined and which variable names to be changed had to be determined. A word processor on a home system was used to cross check all these variables using a "find" function on the word processor.

Appendix J shows the variables that had potential to cause problems, and the ones that have been isolated and corrected.

Graphic Devices Used

As stated in chapter 2, there were 3 graphics devices to choose from. It was hoped at first to integrate all 3 of these graphics devices into the UNIX Core system. But time constraints forced the selection of only 2 of the 4 graphics devices. The 2 graphics devices chosen were selected because they had the most research done on them, could be converted within the time constraints, and were the most readily available ones to other students.

A lack of technical manuals for each of these graphics
devices was a major obstacle.

Except for the Hewlett Packard plotter, there are currently no other color graphics devices available.

The Tektronics 4014 Vector Graphics Terminal was not implemented because it is not yet connected to the VAX to support graphics, though the device driver software for it has been developed (see Appendix G and K).

The two graphics devices chosen were the Visual 550 Graphics Terminal (Tektronics 4014 emulation mode), and the Hewlett Packard 7220A 4 Color Plotter.

**Hewlett Packard 7220A 4 Color Plotter**

This graphics device was chosen because there are two of them connected to Visual 100 terminals, and considerable research was done on its graphics output. Another key reason is that this is currently the only working color graphics device connected to the VAX system.

At first the CDC Calcomp program was converted over from FORTRAN IV on the CDC computer to FORTRAN 77 on the VAX to be used on the plotter. The program now runs on the VAX and Appendix H explains how. But the Calcomp package was converted over in FORTRAN 77, and the incompatibility would not allow any of these routines to be used as potential graphics drivers for the Pascal Core System.

Other graphics packages were examined, but the size limitation forced the creation of specific device drivers.

**Visual 550 Raster Graphics Terminal**
The Visual 550 terminal was chosen to be used in the Tektronics emulation mode because more research has been done for the Tektronics than the Visual 550 Raster terminal. There are currently 4 of these Visual 550 terminals, plus in the future there are 2 Tektronics 4014 terminals that will be connected to the VAX. This will create 6 usable 4014 terminals. Again, the time constraint played a major role in selecting what terminals to develop.

**Graphic Device Drivers**

Device driver package sizes and programming language interfacing were the key elements in selecting the graphics drivers for the 2 selected graphics devices.

Device driver packages such as the Tektronics Plot-10, Movie BYU, or the S package were studied. But their large size was the main reason they were not selected.

Also, interfacing the FORTRAN 77 graphics drivers with Pascal has been determined not to work (See Appendix I).

So it was decided to develop specific graphics drivers for the Pascal Core System to save space and avoid programming language problems.

The input for these graphics devices is simulated through the keyboard because no joysticks or graphics tablets, or other such input devices are as yet installed.

**Language Chosen for Device Drivers**

The C language was chosen for the graphics device

III-11
drivers. C, FORTRAN 77, and Pascal programming languages were considered. FORTRAN 77 was ruled out because of the incompatibility with Pascal. Pascal was also ruled out because if future Core packages are developed on the UNIX in FORTRAN 77 and the device drivers are written in Pascal, then new drivers would have to be developed. C was chosen because it is compatible with both Pascal and FORTRAN 77. Also, if new programming languages, such as ADA, are used on a UNIX system, C will hopefully interface with it. There would be a chance that Pascal or FORTRAN 77 would not interface with a new programming language.

Drivers Created for Visual 550
(Tektronics 4014 Emulation Mode)

Appendices G and K show the device drivers developed for the Visual 550 in the Tektronics 4014 emulation mode.

These drivers were written to support the Pascal Core graphics dependent routines. They replace the VMS graphics routines written in FORTRAN 77 for the Grinnel graphics device. Only the routines needed by the Tektronics 4014 were developed. The color variables have been ignored for this device. But they could be used at a future date.

Tran14, the procedure that translates x,y coordinates into low/high x,y bytes, is very efficient. It translates all incoming x,y coordinates into the low/high x,y bytes, but only sends the bytes that are required for a particular graphics command.

Cross14, which reads the cross hair position, works as it
should, but there is a problem with the EOF returned by the system. The Tektronics returns 5 bytes when reading the cross hair. Byte 1 is the keyboard character depressed, bytes 2 and 3 are the X low and high bytes, and bytes 4 and 5 are the Y low and high bytes. These are returned and read correctly, but an EOF is also returned causing system problems. The only way to flush out the system buffer is to do another read. This solves the problem, but it is still not fully understood why the EOF is returned.

For routines such as line14 which draws a line, and point14 which draws a point, a move must first be made to the start of the line or point. This occurs even if drawing consecutive lines connected together. This is wasteful, but must be done to correctly support this Core system. To complicate matters, the current screen position (represented by xcop ad ycop variables) can move at any time, but the actual cursor can remain in the older location, so a move to the start of the line or point must be made.

A fill routine has been added to this graphics driver. It is part of the polygon fill code that came with the VMS system. It fills by using a "line scan" method. This "line scan" method draws horizontal lines starting at the bottom of a polygon, and works upward.

Drivers Created for Hewlett Packard 7220A 4 Color Plotter

Appendices G and K show the device drivers created for the Hewlett Packard plotter.

The drivers written for the plotter are designed to run
at 2400 baud, which is the maximum rate that the plotter can operate at.

The way the Core system moves the current screen position without updating the cursor (plotter pen in this case) forces the plotter pen to be in the up position for its default value. This causes a lot of "clicking" as the pen is moved to the correct position pen lowered, line drawn, and the pen raised. For several lines or points being drawn this causes a lot of noise and unnecessary movement.

Some background characters are sent back to the Visual 100 screen as the plotter is taken on and offline. They are only "(" and ")" characters sent one at a time, and though present, do not interfere with the application program.

A pause in the graphics routines when the plotter is initialized, or a new graphics frame is to be created, prompts the user to change paper in the plotter if he wants to.

Cross hatching has not as yet been implemented for the plotter due to the time constraint.
IV. Analysis/Test

This chapter analyzes the UP Pascal Core System as implemented on the VAX 11/780 with a UNIX operating system. The Core programs, portability, VAX UNIX system support, and the user environment will be considered in this analysis.

To analyze this system, data and notes kept over a several month period will be used to help make observations in addition to the tests made on the functional Core system.

Much of the analysis and observations made in this chapter will be judgements made over an extended time on the VAX computer system. But these views will be kept in line with the functional and performance analysis. The views will be as objective as possible.

Timing Analysis

This analysis was made to determine the speed of the Core package when run on the AFIT VAX UNIX system. The CPU and clock timing analysis was done on the system when it was saturated, and with five users or less. Several runs were made, and the results averaged.

A program, written in Pascal, allows a variable to be entered to select the number of lines to be drawn for either a 2 dimensional or a 3 dimensional system. The program draws groups of 200 lines, each line being 5.8 cm long. The only potential skewing of results may occur as the variable to select the number of lines to be drawn in either 2 dimensional or 3 dimensional form is selected.
Timing of the results was done by a system timer. The results were for program usage, system usage (these times combined for total cpu time), and clock time for how long the job took to run. The cpu time will be to the nearest tenth of a second, while the clock time is recorded in minutes and seconds.

Two Dimensional and Three Dimensional Test

The CPU and clock timing test was run for line intervals of 200, 1000, 2000, 3000, 4000, and 5000 lines. Figures IV-1, IV-2, IV-3, and IV-4 show the results for five users or less and for a saturated system.

The times for both a 2 dimensional and a 3 dimensional system were the same when averaged out. This is consistent because the Core system multiplies all values of its matrices. Single run times were different, but only by fractions of a second.

For a saturated system the CPU times for either the 2 dimensional system or a 3 dimensional system do not follow a linear path. The CPU time to draw 1000 lines for a saturated system was 15 seconds, while the CPU time to draw 5000 lines was 185 seconds. It should have taken only 75 seconds to draw 5000 lines based on the linear extrapolation of the 1000 line estimate.

Even the clock times show that a linear relationship is not followed.
Figure IV-2. Two and Three Dimensional CPU Timing Test, Five Users or Less on System
Figure IV-3. Two and Three Dimensional Clock Timing Test, System Saturated
Figure IV-4. Two and Three Dimensional Clock Timing Test. Five Users or Less on System.
For these timing tests, the 9600 baud serial transfer rate was very noticeable. The system sends 905 bytes in one block transfer. This was causing the lines to be drawn intermittently, where some lines would be drawn, then a slight pause before more lines would be drawn.

The operating system allowed more CPU time per user when there were fewer users on the system. This allowed for the slightly faster CPU times, but much faster clock times. However, the linear path when comparing CPU time was not followed in either case.

The more blocks of graphic data that has to be transferred to the graphics terminal over the serial line, after being processed by the operating system, causes a progressively larger time to draw greater numbers of lines.

Space Analysis

The space analysis is grouped into either computer memory usage, program usage, or disk storage usage.

Computer Memory Usage

The VAX memory contains 4 million bytes of storage. This memory is made up of pages (1 page equals 1024 bytes). To load an executable program of say 153,600 bytes would take 150 pages. There are 3906 pages available on the VAX at any given time.

The simplest Core program draws a diagonal line from one corner of the screen to the opposite corner. It takes 159 pages to run this program. So 3906 pages divided by 159
pages lets only 25 users access the VAX memory at one time, ignoring the dynamic operating system.

The other extreme is the intcore.run program. It takes 1322 pages to run this program. So 3906 pages divided by 1322 pages lets only 3 users access the VAX memory at one time, again ignoring the dynamic operating system.

Though this does not take into account the dynamic operating system on the VAX, it does give an indication of how large the Core system is compared to the memory available on the VAX.

**Program Usage**

The main emphasis here is to focus attention on how the data, especially the data for lines, is stored.

Presently, data for the x,y,z coordinates is stored for both starting and ending points of lines. These are dynamic storage locations that can be added to the program or deleted. To save space, only the points for lines that are not consecutive need both starting and ending points. Lines that are consecutive only need the continuation point. If all the lines were continuations, then half the program space would be saved. The Core system currently only manipulates line segments with both a starting and ending point.

It requires 156 bytes for the creation of 1 retained segment using the UP Core package. It also takes 368 bytes to store 1 line primitive within a given retained segment. So to store 1000 line primitives within a single retained
segment would require 368,156 bytes.

But most of this program storage is taken by the $x, y, z$ coordinates for a polygon, each using an array of 12 real numbers. These arrays are used for a possible 12 sided polygon, but are wasted for line primitives because each coordinate of a line primitive only uses one of these real numbers. 246 bytes of program storage is wasted for each line primitive, and 246,000 bytes are wasted for 1000 line primitives.

Disk Storage Usage

Currently to hold the entire Core system routines, both text and object files, requires about 12,275 blocks (one block equals 512 bytes) of disk space.

Since this amount of storage is large, and the storage available on disks is limited, this has required that the entire Core system be stored on magnetic tape, only bringing in the files as necessary. The entire Core system is brought onto disk at times, but for only limited amounts of time.

Only the current demonstration program, intcore.run (see Appendix E, part 4), containing 2664 blocks, is kept on disk for any extended time.

Single Screen Graphics Versus Dual Screen Graphics

This is a comparison using the Core System on a single screen that contains both the text and the graphics, or
using a dual screen that contains one screen for text, and
the other for graphics.

The main comparison will be between the Visual 550
(Tektronics 4014 emulation mode) single screen system, and
the Hewlett Packard Plotter with the Visual 100 as its text
screen. The Core System that was written for the VMS
operating system used a dual screen setup.

For a program that is all graphics or all text, a single
screen is no problem. The problem arises when the
application program attempts to mix Core output with
non-Core output.

One solution, to support a single screen concept, is to
prompt the user to continue displaying graphics or go back
to a text screen for more information. This causes a
breaking of the smooth flow of a program. This was the
method used when converting the intcore.run program to work
with the Visual 550 terminal. The Core escape function
allows for this non-Core input/output. Escape(12) puts the
application into the Core input/output mode by first
clearing the screen. While Escape(13) puts the application
back into the standard input/output mode, again clearing the
screen. Escape(14) allows the user to select one of four
available pens on the Hewlett Packard plotter.

Another solution is to break the screen into sections,
one section for text only, while the other section is for
graphics only. This is a compromise that has to be
determined when writing an application program.

The dual screen concept allows for a smoother flow of
information for both text and graphics. There is no need to determine when to switch from text to graphics or back.

**Portability of Pascal Core System**

The Pascal Core System is not portable. It takes time and effort to determine how to convert it from one machine to another. It is more machine dependent than independent.

The main problem is in the interpretation of Pascal itself. It all depends on how a particular computer system has implemented the Pascal. An example is that the VMS Pascal allows for variable length strings, while the UNIX Pascal only allows for fixed length strings. Or the VMS Pascal allows for upper and lower case letters, while the UNIX Pascal only recognizes lower case letters. Another example is where the VMS Pascal allows the underscore character for key words, but the UNIX Pascal flags this as an error.

Each of these differences has to be determined and corrections made to all 300 routines, if necessary, before the Core System can be converted correctly.

This doesn't mean that the Pascal Core System cannot be converted, but it is very time consuming. Now that the Pascal Core System has been converted over to run on the VAX UNIX system, any updates will be fairly easy to make on the UNIX system. But to convert this same system over to another Pascal system, besides the VMS or UNIX systems, will probably require the same tedious effort.
Usability of Pascal Core System for Education

Education is this UNIX Pascal Core System's intended use. It can help develop a student's understanding of graphics and the goal of portability of graphics between computer systems.

A student can study how a Core System is supposed to work, then using this Pascal Core System, try out various graphic routines using the intcore.run program. In this way a student can further understand how the Core is supposed to function.

Application programs can also be supported by this graphics package. Not only can this Core System be used for education, but development of applications in support of other objectives.

Very Large Scale Integration (VLSI) can be developed using this Core package. Currently the real constraint is the speed of drawing lines and the resolution of the output device. The Visual 550 (Tektronics 4014 emulation mode) has a 1023 by 778 pixel resolution. This is sufficient for designs if they are created and viewed in sections. The Hewlett Packard plotter also has a resolution problem. It can draw the VLSI designs on 16 inch by 12 inch paper, but the pen widths are a limiting factor. A VLSI design would have to be drawn in parts, then the plotter pages taped together to form the overall design. This is currently done for a graphics package supporting the Versatec raster plotter, but the resolution of the Versatec plotter is much greater than the Hewlett Packard plotter.
SADT diagrams can be created on this system. A limiting factor in this system is that there is currently no way to permanently store images out to disk. Once the Core system is exited, all the data and designs created are lost and would have to be reentered.

Core System Limitations

There are currently VAX system limitations that restrict a student's ability to complete an assigned task whether it is text editing, programming for a class assignment, or a graphics project. The main system limitation is the available disk space.

Disk Space

There are six disk systems currently available for use. Each user is assigned to one of these six disk systems. The available storage capacity of each disk system is therefore the limitation for each user. If a user runs out of disk space on his disk system, the operating system does not look for more room on the other disk systems. The user has to delete unnecessary disk files in order to create more room on the disk system.

The disk systems are grouped as shown below:

<table>
<thead>
<tr>
<th>system</th>
<th>k bytes</th>
<th>used</th>
<th>free</th>
<th>% used</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>7623</td>
<td>3791</td>
<td>3832</td>
<td>50 %</td>
</tr>
<tr>
<td>usr</td>
<td>141545</td>
<td>117201</td>
<td>24344</td>
<td>83 %</td>
</tr>
<tr>
<td>ls</td>
<td>76123</td>
<td>53284</td>
<td>22839</td>
<td>70 %</td>
</tr>
<tr>
<td>en</td>
<td>141578</td>
<td>126303</td>
<td>15275</td>
<td>89 %</td>
</tr>
<tr>
<td>ui</td>
<td>26848</td>
<td>22133</td>
<td>4715</td>
<td>82 %</td>
</tr>
<tr>
<td>tmp</td>
<td>7317</td>
<td>414</td>
<td>6903</td>
<td>6 %</td>
</tr>
</tbody>
</table>

IV-13
This is not necessarily the most up to date information of the usage of the disk systems. But over several months the disk system has been consistently saturated as shown above. The usr, en, and ls disk systems are the ones used most by the students.

Disk system ul has been used to develop the Pascal Core package. Currently the entire Core system is stored out on magnetic tape. The intcore.run is the only large disk file currently stored out on this particular disk system. It takes up approximately 1363 k bytes, or 5% of the total disk space.

When the entire Core package is transferred from magnetic tape to disk, it takes approximately 12,275 blocks of disk storage, or about 23% of the entire disk space. This is the main reason that the Core package is stored out on magnetic tape. This allows more room to be available on the disk system. But at times this Core package has to be loaded into disk for development. When this happens, the disk usage runs up to approximately 92%. Great care then has to be taken so as not to run out of available disk space. At 97% usage, the disk system automatically shuts off input into it.

Core Program Limitations

The Core System is limited by the time it takes to compile a program, the structure of the include files, and
the routine dependencies.

**Compile Times**

The time it takes to compile a Core program depends on how saturated the VAX system is, the size of the Core program, and the size of the library.

Two comparisons will be made. One comparison will be made on a simple program that draws a single line from one corner of the screen to the opposite corner. The other comparison will be made on the Core program that tests all the separate Core routines.

The simple Core program generally takes less than 35 seconds CPU time, or 5 minutes clock time to compile when the system is busy. Busy is meant to be more than half the available terminals and modem lines are in use. When the system is not busy, it takes less than 5 minutes clock time for the program to compile.

Whereas for the more complex Core program, the times differ dramatically. Trying to compile this program when the system is busy takes 4 hours plus. The plus means this large program has never compiled when the system was busy. Only at night, after 1900 has this program compiled, taking about 8 CPU minutes and over 2 clock hours. With just a single user on the system it takes about 4 CPU minutes and 30 clock minutes to compile.

Since the removal of the word processing capability on the VAX, the compile times have decreased from over 2 hours in the evening to less than 1 hour. But for a single user
the time still takes about 30 minutes.

Include Files

The size of the Pascal Core programs is dependent mainly on the number of include files used. The more include files used, the more overhead the UNIX system requires.

A test was run to compare several Pascal Core routines with their include files, and then compared when these include files were removed. Cltseg.p and cltseg.o files were two of the files used in the test runs. With the include files the cltseg.p and cltseg.o files required 1339 and 6696 bytes respectively. Without the include files they only required 998 and 2230 bytes.

It is estimated that the intcore.run file, which currently uses 1,354,151 bytes could be reduced to about 500,000 bytes with the include files removed.

However, to eliminate these include files would make the Pascal Core System unusable. If these include files were to be removed, then the user would have to know every dependency that exists within the Core System. The user would have to know every variable and parameter used. If any of the variables or parameters were left out, then the program would not compile.

Include files have to be used in the Pascal Core system regardless of the overhead involved.

Routine Dependencies

The routine dependencies have been determined and must be
followed. The UNIX Pascal requires that the dependent routines be above the calling routines or undefined errors will occur.

**Functional Completeness of Pascal Core Package**

The intcore.run program was used to test the Pascal Core System to determine what works and what does not work.

Appendix E, part 1, shows what is supported by the Pascal Core System based on the SIGGRAPH 1979 Standard as published in the August 1979 Computer Graphics Journal. Of those routines listed, I will explain which ones do not work correctly.

The Core 3D perspective projection does not work. This system has not allowed for a fourth homogeneous coordinate.

The routine ddemptseg, which returns dynamic memory to the VAX operating system, does not work correctly. When it tries to dispose of the specified segment a "value out of range error" occurs. The reason for this has not yet been found. To bypass this problem, the dispose has been commented out. This allows the Core program to continue, but the dynamic memory is not returned to the VAX operating system.

The termcore procedure also tries to dispose of the unused space in the program. It gets a value out of range error. This dispose statement has been commented out so the program will continue to function correctly.

When entering numeric data into a Pascal program, outside of the Core system, through a readln statement, if a non
numeric entry is made, then the entire program will crash. This is a system constraint and there is no way around it except to be more careful.

Since there are currently no color terminals available, no color is supported in this system. Color capability is allowed for in the Pascal Core System, but it is untested.

**Timesharing versus Dedicated System**

The current VAX system is set up in a timesharing mode. For the current usage of the Core system, the present system setup is satisfactory. But this setup only has one user and limited graphics display. Also, the graphics devices available are limited and in competition for by other students.

To support a class of 20 students using this Pascal Core package would require a disk system with 10,000 k bytes for each student. This would require a disk system of 200,000 k bytes, when for the entire school there is only 400,000 k bytes available.

Graphics terminals would be needed to support these 20 students. To support these students would require the 4 Visual 550's and the 2 Hewlett Packard Plotters with their 2 supporting Visual 100 terminals.

With this many computer resources needed, then something has to give. One alternative is to use the Pascal Core system as just a demonstration to familiarize students on how the Core system works. This would not tie up much of the currently available resources, yet would allow students
to further understand the Core system.

Another way is to acquire more disk systems and graphics terminals and dedicate them, still on a timesharing mode, to the Core system only.

The PDP 11/60 is another alternative. It could be set up with enough disk space to support the Core system on a dedicated basis.

To dramatically reduce the time to draw lines on the VAX system would require a dedicated system with parallel data transfer. The system currently draws 67 lines a second, but would require a dedicated system to draw 30 frames a second.

Updating the Core Package

Updates to the Core system will have to be made in the best way possible because the system is so large. The UNIX C shell command allows the user to make wholesale changes to all files located within one directory. This way most of the changes to the Core package can be made automatically.

It would be easiest to convert the entire new Core system than try to add only the new changes to the older version of the Core system. This way all the variables and procedures will be consistent. Following the changes made in Appendix C will simplify the conversion process.

There is no easy way to convert the Core package on the VAX. The VMS system is so different from the UNIX system that no standard way to convert from one to the other using a completely automated system exists. Most of the changes can be automated, but some of the changes must be made on an
individual basis.
Conclusions and Recommendations

Conclusions

The Pascal Core System, for the most part, works at AFIT as it is supposed to. This is a good tool to aid students in understanding how the Core System is supposed to work. It helps because a student can get a "hands on" feeling to how the Core system works. He no longer has to guess how the Core system works.

The problems encountered using the UNIX operating system will be an aid to others trying to convert other systems to the UNIX system. If the other projects do not involve the Core system, the lessons learned developing the Core system will still help them.

The use of Pascal limits the portability of this system, even though this system is very modular. The different interpretations of Pascal on different systems limits its transportability. Too many changes have to be made to enable the Pascal to run correctly on another system. The use of Pascal makes the Core system more dependent on the particular system it is designed on.

Most of the time spent developing the Core system was spent in converting the Pascal procedures from the VMS system over to the UNIX system.

This project showed that a Core system on a VMS system can be converted over to run on a UNIX system, but with great difficulty. Many programming and UNIX system constraints had to be resolved at each step of the
Interfacing of computer programming languages such as Pascal, C, and FORTRAN 77 was a problem. The only common language among the three was found to be C.

Recommendations

A FORTRAN 77 Core system should be converted and tested on the UNIX system to determine what problems arise. Then this FORTRAN 77 system should be compared against the Pascal system now implemented. It should be tested for size, compile time, and speed of drawing lines. The current graphics drivers should be used for this system if possible. Portability of this system should be considered in the evaluation.

The Pascal Core system should be updated to the latest VMS Pascal Core package that has been sent to AFIT from the University of Pennsylvania. This will help determine how easily updates can be made to this system.

If possible, add a graphics tablet and lightpen to the Core system to add more flexibility to the system. This will allow the simulation of most of the input side to be changed over to actual input device commands, instead of being simulated by the keyboard.

Bring in and develop a color terminal to demonstrate the color capability of the Pascal Core System. Though the color capability is supported by this system, it has not yet been tested to see if it actually works.

Multiple device capability should be added to the VAX
system to allow the core program to switch back and forth between different devices while the program is running. Currently, the program must be stopped so the user can move to another graphics terminal, losing what the user had already done on the previous Core program.

The ability to permanently store images out to disk for later use would be a good addition to this package. The Graphics Standard Planning Committee (GPSC) Metafile proposal takes this into consideration [1].
Bibliography


BIB-1


28. Comi, Patrick M. *Implementation of the VGM Graphics*


Appendix A

Pascal Core System and Device Driver Compile Files

******************************************************************************
#
# date: 12 oct 83
# version: 1.0
# name: compile
# description: this system program automatically compiles the *.c and *.p
# files listed below creating *.o files
# operating system: UNIX
# language: c shell
# inputs: n/a
# outputs: n/a
# global variables used: n/a
# global variables changed: n/a
# global tables used: n/a
# library routines: n/a
# files read: *.c and *.p files listed below
# files written: *.o files listed below
# modules called: n/a
# calling modules: n/a
# author: Capt John W. Taylor
#
******************************************************************************
#to run: >make -f compile
#
******************************************************************************
*
* the "/" slashes after the *.o files should be backwards
* slashes
*
******************************************************************************
all: one two three four five six seven eight nine ten eleven twelve thirteen
fourteen fifteen sixteen seventeen eighteen nineteen
******************************************************************************
one: awaitbut.o awaitkey.o awaitpick.o awaitstr2.o awaitstr3.o/
   clrseg.o cltseg.o crrseg.o crrseg.o dclndc.o dclndx.o/
   ddrntpdf.o ddprntseg.o delall.o derseg.o
#
awaitbut.o: awaitbut.p
   pc -c -w -o awaitbut.p
   rm awaitbut.o
#
awaitkey.o: awaitkey.p
   pc -c -w -o awaitkey.p
   rm awaitkey.o
#
awaitpick.o: awaitpick.p
   pc -c -w -o awaitpick.p
# rm awaitpick.o
#
awaitstr2.o : awaitstr2.p
   pc -c -w -o awaitstr2.p
# rm awaitstr2.o
#
awaitstr3.o : awaitstr3.p
   pc -c -w -o awaitstr3.p
# rm awaitstr3.o
#
clrseg.o : clrseg.p
   pc -c -w -o clrseg.p
# rm clrseg.o
#
cltseg.o : cltseg.p
   pc -c -w -o cltseg.p
# rm cltseg.o
#
crrseg.o : crrseg.p
   pc -c -w -o crrseg.p
# rm crrseg.o
#
crtseg.o : crtseg.p
   pc -c -w -o crtseg.p
# rm crtseg.o
#
dclndc.o : dclndc.p
   pc -c -w -o dclndc.p
# rm dclndc.o
#
dclndx.o : dclndx.p
   pc -c -w -o dclndx.p
# rm dclndx.o
#
ddprntpdf.o : ddprntpdf.p
   pc -c -w -o ddprntpdf.p
# rm ddprntpdf.o
#
ddprntseg.o : ddprntseg.p
   pc -c -w -o ddprntseg.p
# rm ddprntseg.o
#
dela11.o : dela11.p
   pc -c -w -o dela11.p
# rm dela11.o
#
derseg.o : derseg.p
   pc -c -w -o derseg.p
# rm derseg.o
#*****************************************************************************
two : dstdcl.o environ.o escape.o ibgndx.o iclndc.o iclndx.o/
     iconst.o idtect.o iflndx.o ihilit.o iitn2.o iitr2.o/
     iitr3.o ilndx.o ilstyl.o
#
dstdcl.o : dstdcl.p

A-2
pc -c -w -o dstdcl.p
# rm dstdcl.o
#
environ.o : environ.p
  pc -c -w -o environ.p
# rm environ.o
#
escape.o : escape.p
  pc -c -w -o escape.p
# rm escape.o
#
ibgndx.o : ibgndx.p
  pc -c -w -o ibgndx.p
# rm ibgndx.o
#
iclndc.o : iclndc.p
  pc -c -w -o iclndc.p
# rm ibgndx.o
#
iclndx.o : iclndx.p
  pc -c -w -o iclndx.p
# rm iclndx.o
#
iconst.o : iconst.p
  pc -c -w -o iconst.p
# rm iconst.o
#
idtect.o : idtect.p
  pc -c -w -o idtect.p
# rm idtect.o
#
iflndx.o : iflndx.p
  pc -c -w -o iflndx.p
# rm iflndx.o
#
ihilit.o : ihilit.p
  pc -c -w -o ihilit.p
# rm ihilit.o
#
iitn2.o : iitn2.p
  pc -c -w -o iitn2.p
# rm iitn2.o
#
iitr2.o : iitr2.p
  pc -c -w -o iitr2.p
# rm iitr2.o
#
iitr3.o : iitr3.p
  pc -c -w -o iitr3.p
# rm iitr3.o
#
ilndx.o : ilndx.p
  pc -c -w -o ilndx.p
# rm ilndx.o
ilstyl.o : ilstyl.p
    pc -c -w -o ilstyl.p
# rm ilstyl.o

******************************************************************************
thre : ilwid.o imksym.o indcs2.o initcore.o initde.o/
       initgr.o initvs.o inqbut.o inqdevst.o inqecho.o/
       inqcap.o inqkey.o inqloc2.o inqloc3.o inqlocdi.o
#
    ilwid.o : ilwid.p
    pc -c -w -o ilwid.p
# rm ilwid.o
# imksym.o : imksym.p
    pc -c -w -o imksym.p
# rm imksym.o
#
    indcs2.o : indcs2.p
    pc -c -w -o indcs2.p
# rm indcs2.o
#
    indcs3.o : indcs3.p
    pc -c -w -o indcs3.p
# rm indcs3.o
#
    initcore.o : initcore.p
    pc -c -w -o initcore.p
# rm initcore.o
#
    initde.o : initde.p
    pc -c -w -o initde.p
# rm initde.o
#
    initgr.o : initgr.p
    pc -c -w -o initgr.p
# rm initgr.o
#
    initvs.o : initvs.p
    pc -c -w -o initvs.p
# rm initvs.o
#
    inqbut.o : inqbut.p
    pc -c -w -o inqbut.p
# rm inqbut.o
#
    inqdevst.o : inqdevst.p
    pc -c -w -o inqdevst.p
# rm inqdevst.o
#
    inqecho.o : inqecho.p
    pc -c -w -o inqecho.p
# rm inqecho.o
#
    inqcap.o : inqcap.p
    pc -c -w -o inqcap.p
# rm inqcap.o

A-4
# inqkey.o : inqkey.p
  pc -c -w -o inqkey.p
# rm inqkey.o
#
inqloc2.o : inqloc2.p
  pc -c -w -o inqloc2.p
# rm inqloc2.o
#
inqloc3.o : inqloc3.p
  pc -c -w -o inqloc3.p
# rm inqloc3.o
#
inqlocdi.o : inqlocdi.p
  pc -c -w -o inqlocdi.p
# rm inqlocdi.o
******************************************************************************
four : inqlocp2.o inqlocp3.o inqpic.o inqstrdi.o inqstroke.o/
  inqval.o ioseg.o iotseg.o ipen.o ipesty.o ipid.o iproj.o/
  ispn2.o ispn3.o irsnam.o
#
inqlocp2.o : inqlocp2.p
  pc -c -w -o inqlocp2.p
# rm inqlocp2.o
#
inqlocp3.o : inqlocp3.p
  pc -c -w -o inqlocp3.p
# rm inqlocp3.o
#
inpic.o : inpic.p
  pc -c -w -o inpic.p
# rm inpic.o
#
inqstrdi.o : inqstrdi.p
  pc -c -w -o inqstrdi.p
# rm inqstrdi.o
#
inqstroke.o : inqstroke.p
  pc -c -w -o inqstroke.p
# rm inqstroke.o
#
inqval.o : inqval.p
  pc -c -w -o inqval.p
# rm inqval.o
#
ioseg.o : ioseg.p
  pc -c -w -o ioseg.p
# rm ioseg.o
#
riottseg.o : iotseg.p
  pc -c -w -o iotseg.p
# rm iotseg.o
#
ipen.o : ipen.p
  pc -c -w -o ipen.p
rm ipen.o

ipesty.o : ipesty.p
          pc -c -w -o ipesty.p
rm ipesty.o

ipid.o : ipid.p
          pc -c -w -o ipid.p
rm ipid.o

iproj.o : iproj.p
          pc -c -w -o iproj.p
rm iproj.o

ipsn2.o : ipsn2.p
          pc -c -w -o ipsn2.p
rm ipsn2.o

ipsn3.o : ipsn3.p
          pc -c -w -o ipsn3.p
rm ipsn3.o

irsnam.o : irsnam.p
          pc -c -w -o irsnam.p
rm irsnam.o

isdet.o : isdet.p
          pc -c -w -o isdet.p
rm isdet.o

ishilt.o : ishilt.p
          pc -c -w -o ishilt.p
rm ishilt.o

isitn2.o : isitn2.p
          pc -c -w -o isitn2.p
rm isitn2.o

isitr2.o : isitr2.p
          pc -c -w -o isitr2.p
rm isitr2.o

isitr3.o : isitr3.p
          pc -c -w -o isitr3.p
rm isitr3.o

istdcl.o : istdcl.p
          pc -c -w -o istdcl.p
rm istdcl.o

isttyp.o : isttyp.p
pc -c -w -o isttyp.p
# rm isttyp.o
#
isvis.o : isvis.p
  pc -c -w -o isvis.p
# rm isvis.o
#
itrtp.o : itrtp.p
  pc -c -w -o itrtp.p
# rm itrtp.o
#
itmndx.o : itmndx.p
  pc -c -w -o itmndx.p
# rm itmndx.o
#
ivcpar.o : ivcpar.p
  pc -c -w -o ivcpar.p
# rm ivcpar.o
#
ivdpth.o : ivdpth.p
  pc -c -w -o ivdpth.p
# rm ivdpth.o
#
ivisib.o : ivisib.p
  pc -c -w -o ivisib.p
# rm ivisib.o
#
ivpdis.o : ivpdis.p
  pc -c -w -o ivpdis.p
# rm ivpdis.o
#
ivnpor.o : ivnpor.p
  pc -c -w -o ivnpor.p
# rm ivnpor.o
#
#******************************************************************************
six : ivprt2.o ivprt3.o ivrfpt.o ivup2.o ivup3.o iwindo.o/
  lina2.o lina3.o linr2.o linr3.o marka2.o marka3.o/
  markr2.o markr3.o mkpiccur.o
#
ivprt2.o : ivprt2.p
  pc -c -w -o ivprt2.p
# rm ivprt2.o
#
ivprt3.o : ivprt3.p
  pc -c -w -o ivprt3.p
# rm ivprt3.o
#
ivrfpt.o : ivrfpt.p
  pc -c -w -o ivrfpt.p
# rm ivrfpt.o
#
ivup2.o : ivup2.p
  pc -c -w -o ivup2.p
# rm ivup2.o
#
ivup3.o : ivup3.p
    pc -c -w -o ivup3.p
    # rm ivup3.o

iwindo.o : iwindo.p
    pc -c -w -o iwindo.p
    # rm iwindo.o

lina2.o : lina2.p
    pc -c -w -o lina2.p
    # rm lina2.o

lina3.o : lina3.p
    pc -c -w -o lina3.p
    # rm lina3.o

linr2.o : linr2.p
    pc -c -w -o linr2.p
    # rm linr2.o

linr3.o : linr3.p
    pc -c -w -o linr3.p
    # rm linr3.o

marka2.o : marka2.p
    pc -c -w -o marka2.p
    # rm marka2.o

marka3.o : marka3.p
    pc -c -w -o marka3.p
    # rm marka3.o

markr2.o : markr2.p
    pc -c -w -o markr2.p
    # rm markr2.o

markr3.o : markr3.p
    pc -c -w -o markr3.p
    # rm markr3.o

mkpiccur.o : mkpiccur.p
    pc -c -w -o mkpiccur.p
    # rm mkpiccur.o

#***************************************************************************
sevn : mova2.o mova3.o movr2.o movr3.o ndcwcs2.o newframe.o/
    newline.o pickxy.o plina2.o plina3.o plinr2.o plinr3.o/
    pmrka2.o pmrka3.o pmrkr2.o
    #

mova2.o : mova2.p
    pc -c -w -o mova2.p
    # rm mova2.o

mova3.o : mova3.p
    pc -c -w -o mova3.p
    # rm mova3.o
# movr2.o : movr2.p
  pc -c -w -o movr2.p  # rm movr2.o
# movr3.o : movr3.p
  pc -c -w -o movr3.p  # rm movr3.o
# ndcwcs2.o : ndcwcs2.p
  pc -c -w -o ndcwcs2.p  # rm ndcwcs2.o
# newframe.o : newframe.p
  pc -c -w -o newframe.p  # rm newframe.o
# newline.o : newline.p
  pc -c -w -o newline.p  # rm newline.o
# pickxy.o : pickxy.p
  pc -c -w -o pickxy.p  # rm pickxy.o
# plina2.o : plina2.p
  pc -c -w -o plina2.p  # rm plina2.o
# plina3.o : plina3.p
  pc -c -w -o plina3.p  # rm plina3.o
# plinr2.o : plinr2.p
  pc -c -w -o plinr2.p  # rm plinr2.o
# plinr3.o : plinr3.p
  pc -c -w -o plinr3.p  # rm plinr3.o
# pmrka2.o : pmrka2.p
  pc -c -w -o pmrka2.p  # rm pmrka2.o
# pmrka3.o : pmrka3.p
  pc -c -w -o pmrka3.p  # rm pmrka3.o
# pmrkr2.o : pmrkr2.p
  pc -c -w -o pmrkr2.p  # rm pmrkr2.o
******************************************************************************
@ght : pmrkr3.o polloc2.o polloc3.o polval1.o polya2.o polya3.o/ polyr2.o polyr3.o printer.o renseg.o sbgndx.o sbupdt.o/
sc1pW.o sclpbp.o sclpfp.o
#
pmrkr3.o : pmrkr3.p
   pc -c -w -o pmrkr3.p
   rm pmrkr3.o
#
polloc2.o : polloc2.p
   pc -c -w -o polloc2.p
   rm polloc2.o
#
polloc3.o : polloc3.p
   pc -c -w -o polloc3.p
   rm polloc3.o
#
polval.o : polval.p
   pc -c -w -o polval.p
   rm polval.o
#
polya2.o : polya2.p
   pc -c -w -o polya2.p
   rm polya2.o
#
polya3.o : polya3.p
   pc -c -w -o polya3.p
   rm polya3.o
#
polyr2.o : polyr2.p
   pc -c -w -o polyr2.p
   rm polyr2.o
#
polyr3.o : polyr3.p
   pc -c -w -o polyr3.p
   rm polyr3.o
#
ro : printer.p
   pc -c -w -o printer.p
   rm printer.o
#
renseg.o : renseg.p
   pc -c -w -o renseg.p
   rm renseg.o
#
sbgndx.o : sbgndx.p
   pc -c -w -o sbgndx.p
   rm sbgndx.o
#
sbupd1.o : sbupd1.p
   pc -c -w -o sbupd1.p
   rm sbupd1.o
#
sc1pW.o : sc1pW.p
   pc -c -w -o sc1pW.p
   rm sc1pW.o
#
sclpbp.o : sclpbp.p
pc -c -w -o sclpfp.p
# rm sclpfp.o
sclpfp.o : sclpfp.p
   pc -c -w -o sclpfp.p
# rm sclpfp.o

*******************************************************************************
nine : scortp.o sdtect.o setallbut.o setbut.o setechod.o/
       setechog.o setkey.o setloc2.o setloc3.o setlocp2.o/
       setlocp3.o setmargin.o setpic.o setstroke.o setval.o
#
scortp.o : scortp.p
   pc -c -w -o scortp.p
# rm scortp.o
#
sdtect.o : sdtect.p
   pc -c -w -o sdtect.p
# rm sdtect.o
#
setallbut.o : setallbut.p
   pc -c -w -o setallbut.p
# rm setallbut.o
#
setbut.o : setbut.p
   pc -c -w -o setbut.p
# rm setbut.o
#
setechod.o : setechod.p
   pc -c -w -o setechod.p
# rm setechod.o
#
setechog.o : setechog.p
   pc -c -w -o setechog.p
# rm setechog.o
#
setkey.o : setkey.p
   pc -c -w -o setkey.p
# rm setkey.o
#
setloc2.o : setloc2.p
   pc -c -w -o setloc2.p
# rm setloc2.o
#
setloc3.o : setloc3.p
   pc -c -w -o setloc3.p
# rm setloc3.o
#
setlocp2.o : setlocp2.p
   pc -c -w -o setlocp2.p
# rm setlocp2.o
#
setlocp3.o : setlocp3.p
   pc -c -w -o setlocp3.p
# rm setlocp3.o
#
setmargin.o : setmargin.p
  pc -c -w -o setmargin.p
 #  rm setmargin.o
#
setpic.o : setpic.p
  pc -c -w -o setpic.p
 #  rm setpic.o
#
setstroke.o : setstroke.p
  pc -c -w -o setstroke.p
 #  rm setstroke.o
#
setval.o : setval.p
  pc -c -w -o setval.p
 #  rm setval.o

#*******************************************************************************
ten : sfldndx.o shilit.o simmed.o sitn2.o sitr2.o sitr3.o/
  slndx.o slstyl.o slwid.o smksym.o sndcs2.o sndcs3.o/
  spen.o spesty.o spid.o
#
sfldndx.o : sfldndx.p
  pc -c -w -o sfldndx.p
 #  rm sfldndx.o
#
shilit.o : shilit.p
  pc -c -w -o shilit.p
 #  rm shilit.o
#
simmed.o : simmed.p
  pc -c -w -o simmed.p
 #  rm simmed.o
#
sitn2.o : sitn2.p
  pc -c -w -o sitn2.p
 #  rm sitn2.o
#
sitr2.o : sitr2.p
  pc -c -w -o sitr2.p
 #  rm sitr2.o
#
sitr3.o : sitr3.p
  pc -c -w -o sitr3.p
 #  rm sitr3.o
#
slndx.o : slndx.p
  pc -c -w -o slndx.p
 #  rm slndx.o
#
slstyl.o : slstyl.p
  pc -c -w -o slstyl.p
 #  rm slstyl.o
#
slwid.o : slwid.p
  pc -c -w -o slwid.p
 #  rm slwid.o
# smksym.o : smksym.p
  pc -c -w -o smksym.p
# rm smksym.o
#
sndcs2.o : sndcs2.p
  pc -c -w -o sndcs2.p
# rm sndcs2.o
#
sndcs3.o : sndcs3.p
  pc -c -w -o sndcs3.p
# rm sndcs3.o
#
spen.o : spen.p
  pc -c -w -o spen.p
# rm spen.o
#
spesty.o : spesty.p
  pc -c -w -o spesty.p
# rm spesty.o
#
spid.o : spid.p
  pc -c -w -o spid.p
# rm spid.o

******************************************************************************
elev : sproj.o ssdet.o sshilt.o ssitn2.o ssitr2.o ssitr3.o/
      ssvis.o strtyp.o stxndx.o svdpth.o svisib.o svpdis.o/
      svpnor.o svp2t2.o svp3t3.o
#
sproj.o : sproj.p
  pc -c -w -o sproj.p
# rm sproj.o
#
ssdet.o : ssdet.p
  pc -c -w -o ssdet.p
# rm ssdet.o
#
sshilt.o : sshilt.p
  pc -c -w -o sshilt.p
# rm sshilt.o
#
ssitn2.o : ssitn2.p
  pc -c -w -o ssitn2.p
# rm ssitn2.o
#
ssitr2.o : ssitr2.p
  pc -c -w -o ssitr2.p
# rm ssitr2.o
#
ssitr3.o : ssitr3.p
  pc -c -w -o ssitr3.p
# rm ssitr3.o
#
ssvis.o : ssvis.p
  pc -c -w -o ssvis.p
# rm ssvis.o
# sttyp.o: sttyp.p
c -c w -o sttyp.p
# rm sttyp.o
# stxndx.o: stxndx.p
c -c w -o stxndx.p
# rm stxndx.o
# svdpth.o: svdpth.p
c -c w -o svdpth.p
# rm svdpth.o
# svisib.o: svisib.p
c -c w -o svisib.p
# rm svisib.o
# svpdis.o: svpdis.p
c -c w -o svpdis.p
# rm svpdis.o
# svpnor.o: svpnor.p
c -c w -o svpnor.p
# rm svpnor.o
# svpr2.o: svpr2.p
c -c w -o svpr2.p
# rm svpr2.o
# svpr3.o: svpr3.p
c -c w -o svpr3.p
# rm svpr3.o
#**************************************************************************
twel: svrfpt.o svup2.o svup3.o swindo.o termcore.o termde.o/
termgr.o termvs.o termgr.o wcsndc2.o
# svrfpt.o: svrfpt.p
c -c w -o svrfpt.p
# rm svrfpt.o
# svup2.o: svup2.p
c -c w -o svup2.p
# rm svup2.o
# svup3.o: svup3.p
c -c w -o svup3.p
# rm svup3.o
# swindo.o: swindo.p
c -c w -o swindo.p
# rm swindo.o
# termcore.o: termcore.p
c -c w -o termcore.p
# rm termcore.o
# termde.o : termde.p
# pc -c -w -o termde.p
# rm termde.o
# termgr.o : termgr.p
# pc -c -w -o termgr.p
# rm termgr.o
# termvs.o : termvs.p
# pc -c -w -o termvs.p
# rm termvs.o
# textgr.o : textgr.p
# pc -c -w -o textgr.p
# rm textgr.o
# wcsndc2.o : wcsndc2.p
# pc -c -w -o wcsndc2.p
# rm wcsndc2.o

# thir : clipper.o concol.o daddprim.o dclrsseg.o dcrsprod.o/
# ddotprod.o ddrawseg.o demptseg.o dfindseg.o/
# diniprim.o dininiti.o dnewfrm.o dnewline.o ddprevseg.o
# clipper.o : clipper.p
# pc -c -w -o clipper.p
# rm clipper.o
# concol.o : concol.p
# pc -c -w -o concol.p
# rm concol.o
# daddprim.o : daddprim.p
# pc -c -w -o daddprim.p
# rm daddprim.o
# dclrsseg.o : dclrsseg.p
# pc -c -w -o dclrsseg.p
# rm dclrsseg.o
# dcrsprod.o : dcrsprod.p
# pc -c -w -o dcrsprod.p
# rm dcrsprod.o
# ddotprod.o : ddotprod.p
# pc -c -w -o ddotprod.p
# rm ddotprod.o
# ddrawseg.o : ddrawseg.p
# pc -c -w -o ddrawseg.p
# rm ddrawseg.o
# demptseg.o : demptseg.p


```
  pc -c -w -o ddemptseg.p
  #  rm ddemptseg.o
  #
  ddfindseg.o : ddfindseg.p
  pc -c -w -o ddfindseg.p
  #  rm ddfindseg.o
  #
  ddiniprim.o : ddiniprim.p
  pc -c -w -o ddiniprim.p
  #  rm ddiniprim.o
  #
  ddinitdi.o : ddinitdi.p
  pc -c -w -o ddinitdi.p
  #  rm ddinitdi.o
  #
  ddnewfrm.o : ddnewfrm.p
  pc -c -w -o ddnewfrm.p
  #  rm ddnewfrm.o
  #
  ddnewline.o : ddnewline.p
  pc -c -w -o ddnewline.p
  #  rm ddnewline.o
  #
  ddprevseg.o : ddprevseg.p
  pc -c -w -o ddprevseg.p
  #  rm ddprevseg.o

frtn : ddrealequ.o ddunitvec.o ddusrtrns.o ddvtran.o/
     ddfindseg.o ddfinddi.o
#  frtn : ddrealequ.o ddunitvec.o ddusrtrns.o ddvtran.o/

  #  ddrealequ.o : ddrealequ.p
  pc -c -w -o ddrealequ.p
  #  rm ddrealequ.o
  #
  ddunitvec.o : ddunitvec.p
  pc -c -w -o ddunitvec.p
  #  rm ddunitvec.o
  #
  ddusrtrns.o : ddusrtrns.p
  pc -c -w -o ddusrtrns.p
  #  rm ddusrtrns.o
  #
  ddvtran.o : ddvtran.p
  pc -c -w -o ddvtran.p
  #  rm ddvtran.o
  #
  ddfinddi.o : ddfinddi.p
  pc -c -w -o ddfinddi.p
  #  rm ddfinddi.o
  #
  ddfindseg.o : ddfindseg.p
  pc -c -w -o ddfindseg.p
  #  rm ddfindseg.o
  #
  ddfinddi.o : ddfinddi.p
  pc -c -w -o ddfinddi.p
  #  rm ddfinddi.o
  #
  ddnewfrm.o : ddnewfrm.p
  pc -c -w -o ddnewfrm.p
  #  rm ddnewfrm.o
  #
  ddnewline.o : ddnewline.p
  pc -c -w -o ddnewline.p
  #  rm ddnewline.o
  #
  ddprevseg.o : ddprevseg.p
  pc -c -w -o ddprevseg.p
  #  rm ddprevseg.o

```

A-16
error.o : error.p
    pc -c -w -o error.p
#    rm error.o
#
e scapedd.o : escapedd.p
    pc -c -w -o escapedd.p
#    rm escapedd.o
#
funnel.o : funnel.p
    pc -c -w -o funnel.p
#    rm funnel.o
#
imgtrans.o : imgtrans.p
    pc -c -w -o imgtrans.p
#    rm imgtrans.o
#
#******************************************************************************
fitn : initdd.o initinput.o invmat.o locrel.o makemat.o/
     makinvmat.o mpiccur.o ndcx.o ndcy.o newfrmdd.o pline.o
#
initdd.o : initdd.p
    pc -c -w -o initdd.p
#    rm initdd.o
#
initinput.o : initinput.p
    pc -c -w -o initinput.p
#    rm initinput.o
#
invmat.o : invmat.p
    pc -c -w -o invmat.p
#    rm invmat.o
#
locrel.o : locrel.p
    pc -c -w -o locrel.p
#    rm locrel.o
#
makemat.o : makemat.p
    pc -c -w -o makemat.p
#    rm makemat.o
#
makinvmat.o : makinvmat.p
    pc -c -w -o makinvmat.p
#    rm makinvmat.o
#
mpiccur.o : mpiccur.p
    pc -c -w -o mpiccur.p
#    rm mpiccur.o
#
ndcx.o : ndcx.p
    pc -c -w -o ndcx.p
#    rm ndcx.o
#
ndcy.o : ndcy.p
    pc -c -w -o ndcy.p
#    rm ndcy.o
newfrmdd.o : newfrmdd.p
  pc -c -w -o newfrmdd.p
# rm newfrmdd.o

plline.o : plline.p
  pc -c -w -o plline.p
# rm plline.o

***************************************************************
sxtn : plmark.o plpgon.o pltext.o perdiv.o pgetbut.o pgetcur.o/
pgetkey.o pgetloc2.o pgetloc3.o pgetstr2.o pgetstr3.o/
pgetval.o pputcur.o primxtent.o
#
plmark.o : plmark.p
  pc -c -w -o plmark.p
# rm plmark.o
#
plpgon.o : plpgon.p
  pc -c -w -o plpgon.p
# rm plpgon.o
#
pltext.o : pltext.p
  pc -c -w -o pltext.p
# rm pltext.o
#
perdiv.o : perdiv.p
  pc -c -w -o perdiv.p
# rm perdiv.o
#
pgetbut.o : pgetbut.p
  pc -c -w -o pgetbut.p
# rm pgetbut.o
#
pgetcur.o : pgetcur.p
  pc -c -w -o pgetcur.p
# rm pgetcur.o
#
pgetkey.o : pgetkey.p
  pc -c -w -o pgetkey.p
# rm pgetkey.o
#
pgetloc2.o : pgetloc2.p
  pc -c -w -o pgetloc2.p
# rm pgetloc2.o
#
pgetloc3.o : pgetloc3.p
  pc -c -w -o pgetloc3.p
# rm pgetloc3.o
#
pgetstr2.o : pgetstr2.p
  pc -c -w -o pgetstr2.p
# rm pgetstr2.o
#
pgetstr3.o : pgetstr3.p
  pc -c -w -o pgetstr3.p
# rm pgetstr3.o
#
pgetval.o : pgetval.p
   pc -c -w -o pgetval.p
# rm pgetval.o
#
pputcur.o : pputcur.p
   pc -c -w -o pputcur.p
# rm pputcur.o
#
primxten.o : primxten.p
   pc -c -w -o primxten.p
# rm primxten.o

svtn : ptrack.o pterm.o scx.o scy.o txtcoord.o winxmap.o/
stubs.o pplace.o pcread.o pchar.o ppoly.o pdrpab.o/
   pterm.o perase.o pinit.o pdrlin.o pmovab.o
#
pplace.o : pplace.p
   pc -c -w -o pplace.p
# rm pplace.o
#
pcread.o : pcread.p
   pc -c -w -o pcread.p
# rm pcread.o
#
pchar.o : pchar.p
   pc -c -w -o pchar.p
# rm pchar.o
#
ppoly.o : ppoly.p
   pc -c -w -o ppoly.p
# rm ppoly.o
#
pdrpab.o : pdrpab.p
   pc -c -w -o pdrpab.p
# rm pdrpab.o
#
pterm.o : pterm.p
   pc -c -w -o pterm.p
# rm pterm.o

ase.o : perase.p
   pc -c -w -o perase.p
# rm perase.o
#
pinit.o : pinit.p
   pc -c -w -o pinit.p
# rm pinit.o
#
pdrlin.o : pdrlin.p
   pc -c -w -o pdrlin.p
# rm pdrlin.o
#
pmovab.o : pmovab.p
   pc -c -w -o pmovab.p
# rm pmovab.o
#
#----------------------------------------------------------------------
eqhtn : charhp.o movehp.o linehp.o termhp.o inithp.o/
offhp.o onhp.o selectpenhp.o char14.o move14.o/
tran14.o cross14.o out14.o outhp.o crosshp.o/
penhp.o sizehp.o size14.o
#
charhp.o : charhp.c
 cc -c -w -o charhp.c
#
movehp.o : movehp.c
 cc -c -w -o movehp.c
#
linehp.o : linehp.c
 cc -c -w -o linehp.c
#
termhp.o : termhp.c
 cc -c -w -o termhp.c
#
inithp.o : inithp.c
 cc -c -w -o inithp.c
#
offhp.o : offhp.c
 cc -c -w -o offhp.c
#
onhp.o : onhp.c
 cc -c -w -o onhp.c
#
selectpenhp.o : selectpenhp.c
 cc -c -w -o selectpenhp.c
#
char14.o : char14.c
 cc -c -w -o char14.c
#
move14.o : move14.c
 cc -c -w -o move14.c
#
line14.o : line14.c
 cc -c -w -o line14.c
#
alpha14.o : alpha14.c
 cc -c -w -o alpha14.c
# rm alpha14.o
A-20

```bash
# graphl4.o : raphl4.c
  cc -c -w -o graphl4.c
# rm graphl4.o
#
term14.o : term14.c
  cc -c -w -o term14.c
# rm term14.o
#
initl4.o : init14.c
  cc -c -w -o init14.c
# rm init14.o
#
clr14.o : clr14.c
  cc -c -w -o clr14.c
# rm clr14.o
#
tranl4.o : tran14.c
  cc -c -w -o tran14.c
# rm tran14.o
#
crossl4.o : cross14.c
  cc -c -w -o crossl4.c
# rm crossl4.o
#
outl4.o : out14.c
  cc -c -w -o out14.c
# rm outl4.o
#
outhp.o : outhp.c
  cc -c -w -o outhp.c
# rm outhp.o
#
crosshp.o : crosshp.c
  cc -c -w -o crosshp.c
# rm crosshp.o
#
penhp.o : penhp.c
  cc -c -w -o penhp.c
# rm penhp.o
#
sizehp.o : sizehp.c
  cc -c -w -o sizehp.c
# rm sizehp.o
#
sizel4.o : sizel4.c
  cc -c -w -o sizel4.c
# rm sizel4.o
#
**********************************************************************
nintn : alpha550.o char550.o clr550.o cross550.o graph550.o/
      init550.o line550.o mode550.o out550.o move550.o/
      term550.o tran550.o size550.o
#
alpha550.o : alpha550.c
```
cc -c -w -o alpha550.c
  # rm alpha550.o
  
char550.o : char550.c
  cc -c -w -o char550.c
  # rm char550.o
  
clr550.o : clr550.c
  cc -c -w -o clr550.c
  # rm clr550.o
  
cross550.o : cross550.c
  cc -c -w -o cross550.c
  # rm cross550.o
  
graph550.o : graph550.c
  cc -c -w -o graph550.c
  # rm graph550.o
  
init550.o : init550.c
  cc -c -w -o init550.c
  # rm init550.o
  
line550.o : line550.c
  cc -c -w -o line550.c
  # rm line550.o
  
mode550.o : mode550.c
  cc -c -w -o mode550.c
  # rm mode550.o
  
out550.o : out550.c
  cc -c -w -o out550.c
  # rm out550.o
  
move550.o : move550.c
  cc -c -w -o move550.c
  # rm move550.o
  
term550.o : term550.c
  cc -c -w -o term550.c
  # rm term550.o
  
tran550.o : tran550.c
  cc -c -w -o tran550.c
  # rm tran550.o
  
size550.o : size550.c
  cc -c -w -o size550.c
  # rm size550.o
# date: 12 oct 83
# version: 1.0
# name: compile1
# description: this system program automatically compiles the *
# .c and .p files listed below creating .o files *
# operating system: UNIX
# language: c shell
# inputs: n/a
# outputs: n/a
# global variables used: n/a
# global variables changed: n/a
# global tables used: n/a
# library routines: n/a
# files read: .c and .p files listed below
# modules written: .o files listed below
# modules called: n/a
# calling modules: n/a
# author: Capt John W. Taylor
#
# to run: >make -f compile1
#
# the "/" slashes after the .o files should be backwards
# slashes
#
# all: one two
#
# one: charhp.o movehp.o linehp.o termhp.o inithp.o/
# offhp.o onhp.o selectpenhp.o char14.o move14.o/
# tran14.o cross14.o out14.o outhp.o crosshp.o/
# penhp.o sizehp.o size14.o
#
# charhp.o : charhp.c
#     cc -c -w -o charhp.c
#     rm charhp.o
#
# movehp.o : movehp.c
#     cc -c -w -o movehp.c
#     rm movehp.o
#
# linehp.o : linehp.c
#     cc -c -w -o linehp.c
#     rm linehp.o
#
# termhp.o : termhp.c

A-23
cc -c -w -o termhp.c
# rm termhp.o

init hp.o : init hp.c
cc -c -w -o init hp.c
# rm init hp.o

off hp.o : off hp.c
cc -c -w -o off hp.c
# rm off hp.o

on hp.o : on hp.c
cc -c -w -o on hp.c
# rm on hp.o

select pen hp.o : select pen hp.c
cc -c -w -o select pen hp.c
# rm select pen hp.o

char 14.o : char 14.c
cc -c -w -o char 14.c
# rm char 14.o

move 14.o : move 14.c
cc -c -w -o move 14.c
# rm move 14.o

line 14.o : line 14.c
cc -c -w -o line 14.c
# rm line 14.o

alpha 14.o : alpha 14.c
cc -c -w -o alpha 14.c
# rm alpha 14.o

graph 14.o : graph 14.c
cc -c -w -o graph 14.c
# rm graph 14.o

term 14.o : term 14.c
cc -c -w -o term 14.c
# rm term 14.o

init 14.o : init 14.c
cc -c -w -o init 14.c
# rm init 14.o

clr 14.o : clr 14.c
cc -c -w -o clr 14.c
# rm clr 14.o

tran 14.o : tran 14.c
cc -c -w -o tran 14.c
# rm tran 14.o
cross14.o : cross14.c
    cc -c -w -o cross14.c
#    rm cross14.o
#
out14.o : out14.c
    cc -c -w -o out14.c
#    rm outhp.o
#
outhp.o : outhp.c
    cc -c -w -o outhp.c
#    rm outhp.o
#
crosshp.o : crosshp.c
    cc -c -w -o crosshp.c
#    rm crosshp.o
#
penhp.o : penhp.c
    cc -c -w -o penhp.c
#    rm penhp.o
#
sizehp.o : sizehp.c
    cc -c -w -o sizehp.c
#    rm sizehp.o
#
sizel4.o : sizel4.c
    cc -c -w -o sizel4.c
#    rm sizel4.o
#
******************************************************************************
two : alpha550.o char550.o clr550.o cross550.o graph550.o/
    init550.o line550.o mode550.o out550.o move550.o/
    term550.o tran550.o size550.o
#
alpha550.o : alpha550.c
    cc -c -w -o alpha550.c
#    rm alpha550.o
#
char550.o : char550.c
    cc -c -w -o char550.c
#    rm char550.o
#
clr550.o : clr550.c
    cc -c -w -o clr550.c
#    rm clr550.o
#
cross550.o : cross550.c
    cc -c -w -o cross550.c
#    rm cross550.o
#
graph550.o : graph550.c
    cc -c -w -o graph550.c
#    rm graph550.o
#
init550.o : init550.c
    cc -c -w -o init550.c
# rm init550.o
#
line550.o : line550.c
   cc -c -w -o line550.c
# rm line550.o
#
mode550.o : mode550.c
   cc -c -w -o mode550.c
# rm mode550.o
#
out550.o : out550.c
   cc -c -w -o out550.c
# rm out550.o
#
move550.o : move550.c
   cc -c -w -o move550.c
# rm move550.o
#
term550.o : term550.c
   cc -c -w -o term550.c
# rm term550.o
#
tran550.o : tran550.c
   cc -c -w -o tran550.c
# rm tran550.o
#
size550.o : size550.c
   cc -c -w -o size550.c
# rm size550.o
Appendix B

Pascal Core System and Device Driver Library Files

******************************************************************************
# date: 12 oct 83
# version: 1.0
# name: archive
# description: a system routine to automatically build the
#              upcore library for use in compiling core
#              application programs
# operating system: UNIX
# language: c shell
# inputs: n/a
# outputs: n/a
# global variables used: n/a
# global variables changed: n/a
# global tables used: n/a
# library routines: n/a
# files read: .c, .p files listed below
# files written: lib1
# modules called: n/a
# calling modules: n/a
# author: Capt John W. Taylor
#
******************************************************************************
#
# to run >make -f archive
#
******************************************************************************
#
* after the ".o/" specifications, the "/" slashes should
* be backwards slashes
*
******************************************************************************
all: f1 f2 f3 f4 f5 f6 f7 f8 f9 f10 f11 f12 f13
#
# userext.h
f1:
    ar q lib1 awaitbut.o/
    awaitkey.o/
    awaitpick.o/
    awaitstr2.o/
    awaitstr3.o/
    clrseg.o/
    c1tseg.o/
    crrseg.o/
    c1tseg.o/
    dclndc.o/
    dclndx.o/
    dprntpdf.o/
    dprntseg.o/
    delall.o/

# userext.h cont

f2:

ar q lib1

inqdevst.o/
inqecho.o/
inqicap.o/
inqkey.o/
inqloc2.o/
inqloc3.o/
inqlocdi.o/
inqlocp2.o/
inqlocp3.o/
inpic.o/
inptid.o/
inqt3.o/
inquant.o/
inqv1.o/
inseg.o/
inseg.o/
ipen.o/
ipesty.o/
ipurp.o/
iproj.o/
ipsn2.o/
ipsn3.o/
irnamb.o/
isdet.o/
ishtlt.o/
isitn2.o/
isitr2.o/
isitr3.o/
istdc1.o/
isttyp.o/

B-2
isvis.o/
itrtyp.o/
itxndx.o/
ivcpar.o/
ivdpth.o/
ivisib.o/
ivpdis.o/
ivpnor.o/
ivprt2.o/
ivprt3.o/
ivrfpt.o/
ivup2.o/
ivup3.o/
iwindo.o/
li2.a/
li3.o/
lin2.o/
lin3.o/
mark2.a/
marka3.o/
markr2.o/
markr3.o/
mkpiccur.o

# userext.h cont
f3:
    ar q lib1 mova2.o/
mova3.o/
movr2.o/
movr3.o/
newframe.o/
newline.o/
pickxy.o/
plina2.o/
plina3.o/
plinr2.o/
plinr3.o/
prkka2.o/
prkka3.o/
prkkr2.o/
prkkr3.o/
polloc2.o/
polloc3.o/
polval.o/
polya2.o/
polya3.o/
polyr2.o/
polyr3.o/
printer.o/
renseg.o/
sbgndx.o/
sbupdt.o/
sclipw.o/
sclpbpp.o/
sclpf.o/
scopt.o/

B-3
sdetect.o/
setalibout.o/
setbut.o/
setechod.o/
setechog.o/
setkey.o/
setloc2.o/
setloc3.o/
setlocp2.o/
setlocp3.o/
setmargin.o/
setpic.o/
setstroke.o/
setval.o/
shlndx.o/
shilit.o/
sitn2.o/
sitr2.o/
sitr3.o/
slndx.o/
slstyl.o/
slwid.o/
smksym.o/
sndcs2.o/
sndcs3.o

# userext.h cont
f4 :
ar q lib1 spen.o/
spesty.o/
spid.o/
sproj.o/
ssdet.o/
sshilt.o/
ssitn2.o/
ssitr2.o/
ssitr3.o/
ssvis.o/
strtyp.o/
stxndx.o/
svdpth.o/
svvisib.o/
svpdis.o/
svpnor.o/
svpr2.o/
svpr3.o/
svrfrt.o/
svup2.o/
svup3.o/
swindo.o/
termcore.o/
termde.o/
termgr.o/
termvs.o/
textgr.o/
mpiccur.o/
escape.o

#include "common.h"

f5 :
ar q lib1 txtcoor.o/
siemed.o/
ddnewline.o/
wcsndc2.o/
ndcwcs2.o

#include "util1ext.h"
f6 :
ar q lib1 makinvmat.o/
locrel.o/
ddinitdi.o/
ddclrseg.o/
ddnewfrm.o/
ddvtran.o/
ddprevseg.o/
ddfndsegg.o/
ddrawseg.o/
demptseg.o/
ddaddprim.o/
ddinprim.o/
eqnames.o

#include "util2ext.h"
f7 :
ar q lib1 invmat.o/
primxtent.o/
winvovmap.o/
perdiv.o/
clipper.o/
ddcrsprod.o/
nddotprod.o/
ndreal equ.o/
ndunitvec.o/
ndwrlntns.o/
ndusrtrns.o

#include "ctl1ext.h"
f8 :
ar q lib1 pgetcuro/
pgetstr3.o/
pgetstr2.o/
pgetloc3.o/
pgetloc2.o/
pgetval.o/
pgetkey.o/
pgetbut.o/
initinput.o/
escapedd.o/
newfrmd.d.o/
funnel.o/
initidd.o

#include "ctl0ext.h"
f9 :
ar q lib1 pputcur.o/
ptrack.o/
plmark.o/
plpolygon.o/
pltext.o/
pline.o

# ddutilext.h
f10:
ar q lib1 makemat.o/
imtrans.o/
ndcy.o/
ndcx.o/
sy.o/
sx.o/
concol.o

# driverext.h
f11:
ar q lib1 ptterm.o/
stubs.o/
pplace.o/
pread.o/
pchar.o/
pdrpab.o/
pterm.o/
perase.o/
pinit.o/
pdrlin.o/
pmovab.o/
ppoly.o/
charhp.o/
crosshp.o/
inithp.o/
linehp.o/
movehp.o/
offhp.o/
onhp.o/
penhp.o/
selectpenhp.o/
termhp.o/
outhp.o/
alphan14.o/
char14.o/
clr14.o/
cross14.o/
init14.o/
line14.o/
move14.o/
graph14.o/
term14.o/
tran14.o/
out14.o

# driverexh.h cont
ar q lib1 alphal550.o/
char550.o/
clr550.o/
cross550.o/
init550.o/
line550.o/
mode550.o/
move550.o/
graph550.o/
term550.o/
tran550.o/
out550.o

#error.h
f13:
    ar q lib1 error.o
**archivel**

*date: 12 oct 83*

*version: 1.0*

*name: archivel*

*description: a system routine to automatically build the graphics library for use in compiling application programs*

*graphics device drivers written for:*

1. tektronics 4014
2. hewlett packard 7220a plotter
3. visual 550

*operating system: UNIX*

*language: c shell*

*inputs: n/a*

*outputs: n/a*

*global variables used: n/a*

*global variables changed: n/a*

*global tables used: n/a*

*library routines: n/a*

*files read: .c, .p files listed below*

*files written: lib1*

*modules called: n/a*

*calling modules: n/a*

*author: Capt John W. Taylor*

**to run >make -f archivel**

*after the ".o/" specifications, the "/" slashes should be backwards slashes*

**tektronics 4014 graphics drivers**

**f1:**

```
ar q lib2 alpha14.o/
   char14.o/
   clr14.o/
   cross14.o/
   init14.o/
   line14.o/
   move14.o/
   graph14.o/
   size14.o/
   term14.o/
   tran14.o/
   out14.o
```
# hewlett packard 7220a graphics drivers
#

f2 :  
   ar q lib2 charhp.o/  
       crosshp.o/  
       inithp.o/  
       linehp.o/  
       movehp.o/  
       offhp.o/  
       onhp.o/  
       penhp.o/  
       selectpenhp.o/  
       sizehp.o/  
       termhp.o/  
       outhp.o

#  
# visual 550 graphics drivers  
#

f3 :  
   ar q lib2 alpha550.o/  
       char550.o/  
       clr550.o/  
       cross550.o/  
       init550.o/  
       line550.o/  
       mode550.o/  
       move550.o/  
       graph550.o/  
       size550.o/  
       term550.o/  
       tran550.o/  
       out550.o

#

f4 :  
   ranlib lib2
University of Pennsylvania (UP) Pascal Core System Changes From VMS To UNIX Operating System

298 files total

I. LIST OF CHANGES MADE TO UP PASCAL CORE FILES
II. USER CALLABLE PROCEDURES
III. INCLUDE FILES
IV. BATCH PROGRAMS TO TEST CORE PACKAGE
V. FORTRAN SUBROUTINES TO DRIVE THE GRINNELL GRAPHICS DEVICE
VI. DOCUMENTATION FILES
VII. COMPILATION FILES - FOR VMS VAX
VIII. PROCEDURES TRANSPARENT TO USER
IX. GRINNELL DEVICE DEPENDENT ROUTINES
X. ROUTINES NOT USED, OR DUPLICATES

**************************************************************************
I. LIST OF CHANGES MADE TO UP PASCAL CORE FILES

(numbers) are the cross reference to the UPCORE.xxx files before the conversion to the VAX 11/780 UNIX usable files

THE FOLLOWING CHANGES WERE MADE TO ALL THE FILES LISTED BELOW, IF ANY ADDITIONAL CHANGES FOR A FILE WAS MADE, THEN IT WILL BE EXPLAINED ON A FILE BY FILE BASIS.

1. ALL CAPITAL LETTERS CONVERTED TO LOWER CASE LETTERS
2. ALL UPCORE.xxx FILE NAMES CHANGED TO ORIGINAL VAX 11/780 VMS OPERATING SYSTEM NAMES, WITH A .p, .h, .doc, .com, or .f FILE SPECIFICATION MADE FOR USE BY THE VAX 11/780 UNIX COMPUTER
3. "Module filename" HEADERS COMMENTED OUT IN .p FILES
4. "Xinclude 'incl:xxxxxxx.pas" INCLUDE FILES IN .p FILES CHANGED TO "#include 'xxxxx.h"
5. [global] and [external] LABELS REMOVED
6. "end." REMOVED FROM .p FILES, SINCE THESE ARE PROCEDURES AND THEY MUST TERMINATE WITH AN "end;"" 
7. ALL UNDERSCORES "_" REMOVED SINCE THE UNIX PASCAL SYSTEM WILL NOT ACCEPT UNDERSCORE CHARACTERS
8. CHANGED "EXTERN" TO "EXTERNAL" IN .h FILES
9. IF THE PASCAL PROCEDURE FILE NAME HAS NOT BEEN RENAMED OFF TO THE RIGHT OF THE .p FILE, THEN THIS FILE NAME IS ALSO THE PROCEDURE CALLABLE NAME. IT WILL BE ENCLOSED IN PARENTHESES.
10. (##- or + 1*) REMOVED FROM .h FILES
11. "#" REMOVED FROM .h FILES
12. NEWFRAME AND CLRSEG PROCEDURES CHANGED TO DDNEWFRAME AND DDCLRSEG PROCEDURES TO DISTINGUISH BETWEEN THE NEWFRAME.P AND CLRSEG.P FILES
13. -w OPTION ADDED TO THE COMPILIE FILE TO ELIMINATE THE WARNING MESSAGES IN THE .p FILE COMPILES THAT FLAGGED THE VARIABLES THAT ARE UNUSED
14. INCLUDE .h FILES ADDED TO THE .p FILES AND THE PROCEDURE PARAMETER LIST REMOVED TO AVOID DUPLICATION ERRORS. THIS ALLOWS THE PROCEDURES TO BE REFERENCED BY THE .h FILES
15. VARIABLE STRING ARRAYS ON THE VAX 11/780 WITH VMS OPERATING SYSTEM USING LENGTH(GETS THE LENGTH OF A STRING) AND PAD(PADS THE STRING WITH SPACES) IS DELETED. THIS VAX WAS ALSO ABLE TO CHANGE THE USABLE LENGTH OF A STRING. HOWEVER, THE UNIX PASCAL OPERATING SYSTEM CANNOT MANIPULATE VARIABLE STRING LENGTHS, SO THE UPCORE SYSTEM HAS BEEN ALTERED TO USE FIXED STRING LENGTHS OF 12 AND
80 CHARACTERS. THE UNIX SYSTEM AUTOMATICALLY SPACE FILLS ANY REMAINING UNUSED STRING SPACE.

18. THE .h FILES ARE NOT IN THE CORRECT PASCAL PROCEDURE ORDER. PROCEDURES THAT REFERENCE OTHER ROUTINES MUST BE BELOW THEM IN THE .h FILE, OR AN 'UNDEFINED' ERROR WILL OCCUR. THE CALLING PROCEDURE MUST BE ABLE TO 'LOOK UP' TO FIND THE NECESSARY ROUTINE. THE .h FILES HAVE BEEN MODIFIED AND STRUCTURED TO RUN ON THE UNIX OPERATING SYSTEM. THE FOLLOWING .h FILES ARE IN THEIR CORRECT ORDER:

```
error.h    --- added 23 Jun 83
driverext.h
ddtypeext.h
c10ext.h
c11ext.h
util2ext.h
utilext.h
common.h    --- added 23 Jun 83
userext.h
```

19. VISINIT VARIABLE ADDED TO THE FOLLOWING FILES TO DETERMINE IF THE GRAPHICS DEVICE HAS BEEN INITIALIZED:
```
lin2.p, lina3.p, linr2.p, linr3.p, marka2.p,
marka3.p, markr2.p, markr3.p, plina2.p, plina3.p,
plinr2.p, plinr3.p, pmk2.p, pmk3.p, pmrk2.p, pmrk3.p,
polya2.p, polya3.p, polyr2.p, polyr3.p, textgr.p,
simmed.p, sbudtp.p, ssvis.p, delall.p, derseg.p,
newframe.p, sshilt.p, ssitn2.p, ssitr2.p, ssitr3.p
```

II. USER CALLABLE PROCEDURES

(001) awaitbut.p
(002) awaitkey.p
(003) awaitpick.p    --- (034) ddname.p is old, not used
                     warning message variables debugging,
                     newz, and i never used
                     (corrected)
(004) awaitstr2.p
(005) awaitstr3.p
(011) clrseg.p
(012) c1tseg.p
(019) crrseg.p    --- errors "otherwise" in case statement lines
                     91 and 99: deleted identifier line 159
                     lastsegement / inserted
                     (corrected) complete variable set inserted
                     in case, and an 'else' was missing from
                     line 159
(020) c1tseg.p
(023) dclndc.p
(024) dclndx.p
(038) dprintpdf.p (printpdf)

C-3
--- error line 89, replaced field id with record id
  "writeln(------char.length:1)
  (corrected) function length was added to
  procedure

(039) ddprntseg.p (printseg)

--- error line 79 char.length, replaced
  field id with record id
  (corrected) function length was added to
  procedure

(050) delall.p
(051) derseg.p
(053) dstdcl.p
(055) environ.p (saveenv and restoreenv)
(059) escape.p
(067) ibgndx.p
(068) icindc.p
(069) icindx.p
(070) iconst.p
(071) idetect.p
(072) iflndx.p
(073) ihilit.p

(074) iitn2.p  --- error line 46, "otherwise" in case statement
  (corrected) complete variable set added to
  case

(075) iitr2.p  --- error line 46, "otherwise" in case statement
  (corrected) complete variable set added to
  case

(076) iitr3.p
(077) ilndx.p
(078) ilstyl.p
(079) ilwid.p
(081) imksym.p
(083) indcs2.p
(084) indcs3.p
(085) initcore.p  --- (163) oldinitco.p is the old initcore
(087) initde.p
(088) initgr.p
(090) inqbut.p
(091) inqdevst.p (inqdevstat)
(092)inqecho.p
(093) inqicap.p
(094) inqkey.p
(095) inqloc2.p
(096) inqloc3.p
(097) inqlocdi.p
(098) inqlocp2.p
(099) inqlocp3.p
(100) inqpic.p
(101) inqstrdi.p
(102) inqstroke.p
(103) inqval.p
(107) ioseg.p
(108) iotsseg.p
(109) ipen.p
ipesty.p
ipid.p
iproj.p
ipsn2.p
ipsn3.p
irsnam.p
isdet.p
ishilt.p
isitn2.p
isitr2.p
isitr3.p
istdc1.p
isttyp.p
isvis.p
itrtyper.p
itxndx.p
ivcpar.p
ivdpth.p
ivisib.p
ivpdis.p
ivpnor.p
ivprt2.p
ivprt3.p
ivrfpt.p
ivup2.p
ivup3.p
iwindo.p
lin2.p
lin3.p
marka2.p
marka3.p
markr2.p
markr3.p
mppiccur.p ---- does not appear in the compile file

mova2.p
mova3.p
movr2.p
movr3.p
mppiccur.p (makepiccurrent)
dcwcs2.p
newframe.p
newline.p
pickxy.p --- warning messages, variables debugging, newz, and i never used
(corrected)

plina2.p
plina3.p
plinr2.p
plinr3.p
parka2.p
parka3.p
parkr2.p
(209) parkr3.p
(210) polloc2.p
(211) polloc3.p
(212) polyval.p
(213) polya2.p
(214) polya3.p
(215) polyr2.p
(216) polyr3.p
(221) printer.p
(230) renseg.p
(231) sbgndx.p
(232) sbupdt.p (beginupdt and endupdt)
(233) sclipw.p
(234) sclvp.p
(235) sclvp.p
(236) scortp.p
(239) sdtect.p
(240) setallbut.p
(241) setbut.p
(242) setechod.p
(243) setechod.p
(244) setkey.p
(245) setloc2.p
(246) setloc3.p
(247) setlocp2.p
(248) setlocp3.p
(249) setmargin.p
(250) setpic.p
(251) setstroke.p
(252) setval.p
(253) sfldx.p
(254) shilit.p
(255) simmed.p
(256) sitn2.p
(257) sitr2.p
(258) sitr3.p
(259) sfldx.p
(260) simstyl.p
(261) slwid.p
(262) saksym.p
(263) sndcs2.p
(264) sndcs3.p
(265) spen.p
(266) spesty.p
(267) spd.p
(268) spproj.p
(269) ssdet.p
(270) sshilt.p
(271) ssitn2.p
(272) ssitr2.p
(273) ssitr3.p
(274) ssvis.p
(275) strtyp.p
(277) svdpth.p
(278) ssvisib.p
(279) svpdis.p
(280) svpnor.p
(281) svprt2.p
(282) svprt3.p
(283) svrftp.p
(284) svup2.p
(285) svup3.p
(286) swindo.p
(287) termcore.p --- dispose(temp) commented out because of pointer value error. Reason for error not yet found.

visinit set to false to deactivate device

(288) termde.p
(289) termgr.p
(290) textgr.p --- "*" added before "with p^ do" line, now compiles correctly (corrected)

(297) wcsndc2.p

************************ 174 FILES

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

III. INCLUDE FILES

(021) cttl0ext.h --- defines external procedures
(022) cttl1ext.h --- defines external procedures
(043) ddutilext.h --- defines external procedures
(047) defconst.h --- defines external constants
penmax variable reset to 4 to support 4 pens

(049) deftype.h --- defines external types
changed "nametype = varying[12] of char;" to "nametype = packed array [1..12] of char;"
changed "charray = varying [charmax] of char;" to "charray = packed array [1..charmax] of char;"
changed "charline = varying [linelength] of char;" to "charline = packed array [1..linelength] of char;"
(corrected)

(052) driverext.h --- defines external procedures
(056) envtype.h --- defines external types
(061) extvar.h --- defines external variables
variables terror(determines if graphics device on or off line), and surfacename (determines what graphics device is in use) added

(063) gblvar.h --- defines external variables
(292) unspecvar.h --- defines external variables
(293) userext.h --- defines external procedures
this include file references all user callable procedures listed in part I.

(295) util2ext.h --- defines external procedures
IV. BATCH PROGRAMS TO TEST CORE PACKAGE

(006) batch1.p --- batch

demonstrates operation of 6 logical input
devices warning, many fields not used
(corrected)

(007) batch2.p --- batch

draws a diagonal line on screen, very simple
program warning, many fields not used
(corrected)

(105) intcore.p --- 55 pages of code, extensive core test

**** many errors ****
(corrected) if statements replace 'otherwise'
statements in case; read12, read80, and
readlet procedures added to intcore to read
input from crt into strings(packed array);
name1[1] = 'y' compares changed to name1 =
'y  
modified and commented to run on UNIX Pascal
system. Minor changes.

*********** 3 FILES

*********** 13 FILES

V. FORTRAN SUBROUTINES TO DRIVE THE GRINNELL GRAPHICS DEVICE

FORTRAN SUBROUTINE NAMES THAT ARE NOT FAMILIAR TO
THE UNIX SYSTEM THAT ARE NOT DEFINED IN ANY OF
THE FORTRAN SUBROUTINES:

grfcd  grsbf1  grqons  grqlc
grqrcds grfvc  grcbck  grfar
grfer  grwdb  grsbf4  grqon
grqbld  grqlcs  grnin  grnbys
grczcl  grcdwd  grcdhg  grcrep

These routines are not used in the UNIX Core System.

(064) gcom.f --- common declaration, not used
(065) gpc3.f --- no good, not used
(170) pchar.f
(171) pcread.f
(172) pdcirc.f
(173) pdrdsk.f
(174) pdrlab.f
(175) pdrlin.f
(176) pdrpab.f
(177) pdrrec.f
(178) perase.f
(179) perchr.f
(180) percir.f
(182) perdisk.f
(183) perlab.f
(184) perlin.f
(185) perpab.f
(186) perrec.f
(187) persec.f
(193) pgetpo.f
(198) pinit.f
(203) plower.f
(204) pmoveab.f
(205) pmovef.f
(217) pplace.f
(218) ppoly.f
(223) psend.f ---- not used
(224) psetco.f
(225) psetgr.f
(226) pterm.f
(229) pvport.f

************************ 31 FILES

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

VI. DOCUMENTATION FILES

(009) changes.doc --- changes to the pascal core system
(015) core.doc --- brief summary of core system
(017) coreext.doc --- documentation on any additional routines
(018) coreuse.doc --- using the core system
(048) defs.doc --- definitions of various parameters and attributes
(104) intcore.doc --- shows all command menus
(162) notes.doc --- miscellaneous notes
(294) userincl.doc

************************* 8 FILES

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

VII. COMPILATION FILES - FOR VMS VAX

(008) batcomp.com --- batch compile of all core routines
(013) complib.com --- compile all high level dd routines
(016) coreall.com --- compile entire core di system
(139) linkint.com

************************* 4 FILES

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

VIII. PROCEDURES TRANSPARENT TO USER
(010) clipper.p --- warning, line 102 (************)
    line 195 newpt(p,k...) not defined
    contains outcode procedure, so (165)
    outcode.p not needed
    (corrected)

(025) ddaddprim.p (addprim)
    --- warning, variables prim and i never used
    (corrected)

(026) ddclrseg.p
(027) ddcrsprod.p (function) (crossproduct)
(028) dddotprod.p (dotproduct)
(029) dddrawseg.p (drawseg)
(030) ddemptseg.p (emptyseg)
    --- dispose(temp) commented out
    because it doesn't work

(031) ddfindseg.p (function) (findseg)
(032) ddiniprim.p (initprim)
    --- symbol redone to be set to mrkr instead of
    defaulting to 1 as before

(033) ddinitdi.p (initdi)
(035) ddnewfma.p (ddnewframe)
(037) ddpresseg.p (function) (prevseg)
(040) ddrealsqu.p (function) (realequal)
(041) ddunitvec.p (function) (unitvector)
(042) ddusrtrans.p (usertrans)
(044) ddvtran.p (function) (vtran)
    --- replaces (045) ddvtrnorg.p
    error, line 201 "otherwise" case statement
    (corrected) if statement replaces
    otherwise statement

(046) ddnldtns.p (worldtrans)
(057) eqnames.p (function)
    --- lines 20, 21, 22, 23 "length" is an
    undefined function
    lines 21, 23 "pad" is an
    undefined function
    (corrected) since 'length' is not used,
    these lines are commented out

(058) error.p --- error, line 289 "otherwise" case statement
    (corrected) if statement replaces otherwise
    statement

(106) invmat.p
(142) locrel.p (function) (locrec)
    --- error line 50 iclass improperly used
    lines 50, 51 length undefined
    (corrected) changed to now only use 12
    character strings

(144) makinvmat.p --- documentation says makinvmat.p replaced by
    (143) makemat.p, but (003) awaitpick.p
    uses it

(181) perdiv.p
(220) primxtent.p
(276) stxndx.p
(298) winvewmap.p
IX. GRINNELL DEVICE DEPENDENT ROUTINES

(014) concol.p
(036) ddnewline.p --- does not appear in the (013) complib.com file
(060) escapedd.p --- three routines added:
escl2 - put terminal on line
escl3 - put terminal off line
esc14 - select a pen

variable terror (checks if graphics devices on or off line) added

(062) funnel.p --- interface between dd and di routines
(080) imgtrans.p
(086) initdd.p --- rewritten to initialize device
(089) initinput.p
(143) makemat.p --- documentation says makemat.p replaces
(144) makinvmat.p, but (144) is used by
(003) awaitpick.p this appears in both the device independent compile file (016)
coreall.com, and the device dependent compile file (013) complib.com

(156) ndcx.p (function)
(157) ndcy.p (function)
(159) newframedd.p (newframedd)
(166) plline.p --- warning, variables co2, co3, col not used
line 84 "*" added
(corrected)

(167) plmark.p --- warning, variables co1, co2, co3 not used
error, lines 102, 103, 104, 105, 106 "char"
incompatible with string
(corrected) changed 'char' in driverext.h
file to 'ddchar' in plmark, plower, and
perchr; changed single "." compares to
'.' and 1 to 80 in pchar calls!

(168) plpgon.p --- warning, variables co1, co2, co3 not used
line 180 "*" added
(corrected)

(169) pltext.p --- error line 45 length and line 72 cstring
allowed only on records, not on strings
warning, co1, co2, co3 not used
(corrected) function length added; changed
for loops to loop 80 times; pchar changed
to use 80;

(188) pgetbut.p
(189) pgetcur.p (pgetcursor)
--- part of device driver
parameter type not identical to type of var
parameter istat of ptrack
this file appears in both the dependent
(190) pgetkey.p --- warning, variable keynum not used  
    error, line 49 string.length improperly used  
    (corrected) mostly commented out now,  
    changed to read input from crt into 80  
    character string  
    Prompt message added to code
(191) pgetloc2.p --- large section of code commented out, so it  
    will run on UNIX
(192) pgetloc3.p --- warning, variables locnum, butnum, l1, l1,  
    l2, not used  
    (corrected)
(194) pgetstr2.p --- warning, variable strokenum not used  
    (corrected)  
    Procedure rewritten to run on UNIX.
(195) pgetstr3.p --- warning, variables arraysiz, xarray,  
    yarray, zarray, snretrn, and strokenum  
    not used (corrected)
(196) pgetval.p --- warning, variable valnum not used  
    (corrected)
(219) pputcur.p
(227) ptrack.p
(228) pttterm.p
(237) scx.p (function)
(238) scy.p (function)
(291) txtcoor.p --- error, line 51 7*class replaced field id  
    with a record id  
    (corrected) function length added;

**************************************************************************  29 FILES
**************************************************************************

X. FILES NOT USED, OR DUPLICATES

(034) ddname.p (oldawaitpick)  
    --- not used, use (003) awaitpick.p instead  
    error line 375, end; expected  
    warning, kvariable newz not used  
    --- does not appear in the compile file (016)  
    coreall.com  
    --- changed procedure name from awaitpick to  
    oldawaitpick
(045) ddvtrnorg.p (oldvtran)  
    --- replaced by (044) ddvtran.p  
    error, line 187 "otherwise" case statement  
    --- does not appear in the compile file (016)  
    coreall.com  
    --- changed procedure name from vtran to  
    oldvtran
(054) end.p --- not used at all  
    does not appear in the compile file (016)  
    coreall.com
(066) header.p — not used at all
does not appear in the compile file (016)
coreall.com

(082) incl.p — undefines in include files?????
does not appear in the compile file (016)
coreall.com

(161) newpt.p (oldnewpt)
--- a newpt procedure exists within (010)
clipper.p
--- changed procedure name from newpt to oldnewpt

(163) oldinitco.p (oldinitcore)
--- this is the old initcore, use (085)
initcore.p does not appear in the compile file (016) coreall.com
--- changed procedure name from initcore to oldinitcore

(164) oldmakmat.p (oldmakemat)
--- this is the old makemat, use
(143) makemat.p
many errors!!!!!
does not appear in the compile file (016) coreall.com
--- changed procedure name from makemat to oldmakemat

(165) outcode.p (oldoutcode)
--- zain undefined, this subroutine is not used
in (010) clipper.p, (010) uses its own
outcode procedure does not appear in the compile file
(016) coreall.com
--- changed procedure name from outcode to oldoutcode

(222) ptrmat.p --- does not appear in the compile file (016)
coreall.com
used as a debug aid, prints a 4X4 matrix

****************************************************************************** 10 FILES

****************************************************************************** 298 TOTAL FILES

---
Appendix D

Pascal Core System Additions

The following Pascal files were added to the Core System to replace the device dependent routines written in FORTRAN 77 for the VMS Pascal Core System.

These routine names were kept the same to keep changes to the other Core routines to a minimum. Not all the routines were converted into Pascal. Only the routines necessary to run the Core System on the UNIX Core System were converted.

Device Dependent Routines:

- pmovab.p - moves to the specified screen coordinates
- pdrlin.p - draws a line between 2 given endpoints
- pinit.p - initializes the specified graphics device
- perase.p - erases the specified graphics screen
- pterm.p - terminates the specified graphics device
- pdrpab.p - draws a point on the specified graphics device
- ppoly.p - draws filled polygons using a scan line conversion method
- pchar.p - writes a text string to specified graphics device (maximum 80 characters)
- pcread.p - gets the cross hair position in x,y coordinates on the screen
- pplace.p - moves the cursor to the specified screen coordinates

Also, the University of Pennsylvania Core System was designed to support only one graphics device. The following device independent routines were added to allow for more than one graphics device. The last two were rewritten to support the new graphics devices.

Device Independent Routines:

- initvs.p - initialize the variables for the selected graphics device (supports only one active graphics device at a time)
- teravs.p - terminate the previously selected graphics device
- pgetstr2.p - get stroke device
- initdd.p - initialize device dependent variables
Appendix E

User Documentation for Pascal Core System based on SIGGRAPH August 1979 Core Standards

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

user documentation for the university of pennsylvania pascal core system on the afit vax 11/780 with unix operating system(engineering room 133).

this document explains how to use the pascal core system on the afit vax when writing application programs. it follows the siggraph 79 standard as published in the august 1979 computer graphics journal.

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

table of contents

1. core functions with equivalent pascal procedure names(including parameters)
   * - means the pascal procedure is not implemented

2. table of contents of user level procedures

3. application program example

4. program that demonstrates and tests the core system

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

1. core functions with equivalent pascal procedure names(including parameters)

2 output primitives

2.1 overview

2.2 functional capabilities

2.2.1 current position

2.2.1.1 move

move_abs_2(x,y)
move2(x,y : real)
move_abs_3(x,y,z)
move3(x,y,z : real)
move_rel_2(dx,dy)
movr2(dx,dy : real)
move_rel_3(dx,dy,dz)
movr3(dx,dy,dz : real)

2.2.1.2 inquire current position

inquire_current_position_2(x,y)
ipsn2(var x,y : real)
inquire_current_position_3(x,y,z)
ipsn3(var x,y,z : real)

E-1
2.2.2 line drawing primitives

2.2.2.1 line

\[ \text{line-abs}_2(x,y) \]
\[ \text{line-abs}_3(x,y,z) \]
\[ \text{line-rel}_2(dx,dy) \]
\[ \text{line-rel}_3(dx,dy,dz) \]

2.2.2.2 polyline

\[ \text{polyline-abs}_2(x\_array,y\_array,n) \]
\[ \text{polyline-abs}_3(x\_array,y\_array,z\_array,n) \]
\[ \text{polyline-rel}_2(dx\_array,dy\_array,n) \]
\[ \text{polyline-rel}_3(dx\_array,dy\_array,dz\_array,n) \]

2.2.3 text primitives

2.2.3.1 text

\[ \text{text}(\text{character string}) \]
\[ \text{text}(\text{charst},n) \]
\[ \text{textgr(ch : chararray)} \]

2.2.3.2 inquire text extent

* \[ \text{inquire_text-extent}_2(\text{character string}, \text{surface name}, \text{dx}, \text{dy}) \]
* \[ \text{inquire_text-extent}_3(\text{character string}, \text{surface name}, \text{dx}, \text{dy}, \text{dz}) \]

2.2.4 marker primitives

2.2.4.1 marker

\[ \text{marker-abs}_2(x,y) \]
\[ \text{marker-abs}_3(x,y,z) \]
\[ \text{marker-rel}_2(dx,dy) \]
\[ \text{marker-rel}_3(dx,dy,dz) \]

2.2.4.2 polymarker

\[ \text{polymarker-abs}_2(x\_array,y\_array,n) \]
\[ \text{polymarker-abs}_3(x\_array,y\_array,z\_array,n) \]
\[ \text{polymarker-rel}_2(dx\_array,dy\_array,n) \]
\[ \text{polymarker-rel}_3(dx\_array,dy\_array,dz\_array,n) \]

2.2.5 polygon

\[ \text{polygon-abs}_2(x\_array,y\_array,n) \]
\[ \text{polya2(xarray,yarray : rarray)} \]
\[ \text{polygon-abs}_3(x\_array,y\_array,z\_array,n) \]
\[ \text{polya3(xarray,yarray,zarray : rarray)} \]
\[ \text{polygon-rel}_2(dx\_array,y\_array,n) \]
\[ \text{polyr2(dxarray,dyarray : rarray)} \]
\[ \text{polygon-rel}_3(dx\_array,y\_array,z\_array,n) \]
polyr3(dxarray, dyarray, dzarray : array; n : integer)

2.2.6 primitive attributes

3 picture segmentation and naming
3.1 overview
3.1.1 retained segments
3.1.2 temporary segments
3.2 functional capabilities
3.2.1 retained segments
3.2.1.1 create_retained_segment(segment_name)
crrseg(segnamel : nametype)
3.2.1.2 close_retained_segment()
crlseg;
3.2.1.3 delete_retained_segment(segment_name)
derseg(segnamel : nametype)
3.2.1.4 delete_all_retained_segments()
delall;
3.2.1.5 rename_retained_segment(segment_name, new_name)
renseg(segnamel, newname : nametype)
3.2.1.6 inquire_retained_segment_surfaces(segment_name, array_size, view_surface_array, number_of_surfaces)
3.2.1.7 inquire_retained_segment_names(array_size, segment_name_array, number_of_segments)
irsnam(max : integer; var namearray : array; var num : integer)
3.2.1.8 inquire_open_retained_segment(segment_name)
ioseg(var segname1 : nametype)
3.2.1.9 retained segment attributes
3.2.1.10 naming of primitives for pick input
3.2.2 temporary segments
3.2.2.1 create_temporary_segment()
crtseg;
3.2.2.2 close_temporary_segment()
cltseg;
3.2.2.3 inquire_open_temporary_segment(open)
ioseg(var open : boolean)

4 attributes
4.1 overview
4.2 functional capabilities
4.2.1 static attributes
4.2.1.1 setting primitive static attribute values
set_linestyle(linestyle)
s1styl(linstyl : integer)
set_linewidth(linewidth)
  slwid(width : integer)
set_pen(pen)
  spen(penn : integer)
* set_font(font)
* set_charsize(charwidth, charheight)
* set_charplane(dx_plane, dy_plane, dz_plane)
* set_charup_2(dx_charup, dy_charup)
* set_charup_3(dx_charup, dy_charup, dz_charup)
* set_charpath(charpath)
* set_charspace(charspace)
* set_charjust(charjust)
* set_charprecision(charprecision)
set_marker_symbol(symbol)
  saksym(sym : integer)
set_pick_id(id)
  spid(id : integer)
* set_polygon_interior_style(interior_style)
set_polygon_edge_style(edge_style)
  spesty(edge : integer)
set_line_index(line_index)
  slndx(lindex : integer)
set_fill_index(fill_index)
  sfldnx(flindex : integer)
set_text_index(text_index)
  stxndx(txindex : integer)
* set_line_color(line_color)
* set_fill_color(fill_color)
* set_text_color(text_color)
* set_line_intensity(line_intensity)
* set_fill_intensity(fill_intensity)
* set_text_intensity(text_intensity)
* set_vertex_indices(vertex_indices)
* set_vertex_colors(vertex_colors, n)
* set_vertex_intensities(vertex_intensities)

4.2.1.2
* set_primitive_attributes_2(primitive_attribute_array_2)
* set_primitive_attributes_3(primitive_attribute_array_3)

4.2.1.3 inquiring primitive static attribute values
inquire_linestyle(linestyle)
  ilstyl(var linstyl : integer)
inquire_linewidth(linewidth)
  ilwid(var width : integer)
inquire_pen(pen)
  ipen(var penn : integer)
* inquire_font(font)
* inquire_charsize(charwidth, charheight)
* inquire_charplane(dx_plane, dy_plane, dz_plane)
* inquire_charup_2(dx_charup, dy_charup)
* inquire_charup_3(dx_charup, dy_charup, dz_charup)
* inquire_charpath(charpath)
* inquire_charspace(charspace)
* inquire_charjust(charjust)
* inquire_charprecision(charprecision)
inquire_marker_symbol(symbol)
imksym(var sym : integer)
inquire_pick_id(id)
  ipid(var id : integer)
  * inquire_polygon_interior_style(interior_style)
inquire_polygon_edge_style(edge_style)
  ipesty(var edge : integer)
inquire_line_index(line_index)
  ilndx(var lindex : integer)
inquire_fill_index(fill_index)
  iflndx(var flindex : integer)
inquire_text_index(text_index)
  itxndx(var txindex : integer)
  *
inquire_line_color(line_color)
  * inquire_fill_color(fill_color)
  * inquire_text_color(text_color)
  * inquire_line_intensity(line_intensity)
  * inquire_fill_intensity(fill_intensity)
  * inquire_text_intensity(text_intensity)
  * inquire_vertex_indices(n, vertex_indices, number_of_vertices)
  * inquire_vertex_colors(n, vertex_colors, number_of_vertices)
  * inquire_vertex_intensities(n, vertex_intensities, number_of_vertices)
  *
inquire_primitive_attributes_2 (primitive_attribute_array_2)
  *
inquire_primitive_attributes_3 (primitive_attribute_array_3)

4.2.1.4
* inquire_primitive_attributes_2
  (primitive_attribute_array_2)
* inquire_primitive_attributes_3
  (primitive_attribute_array_3)

4.2.1.5
set_image_transformation_type(type)
  strtyp(ttype : integer);

4.2.1.6
inquire_image_transformation_type(type)
  itrtyp(var ttype : integer)

4.2.1.7
inquire_segment_image_transformation_type
  (segment_name_type)
  isttyp(segmentname: nametype, var ttype : integer)

4.2.2 retained segment dynamic attributes
4.2.2.1 setting retained segment dynamic attribute values
set_visibility(visibility)
  svisib(vis : boolean)
set_highlighting(highlighting)
  shil(t hil : boolean)
set_detectability(detectability)
  sdtact(detect : integer)
set_image_translate_2(tx, ty)
  sitn2(tx, t y : real)
set_image_transformation_2(sx, sy, a, tx, ty)
  sitr2(sx, sy, ax, tx, ty : real)
set_image_transformation_3(sx, sy, sz, ax, ay, az, tx, ty, tz)
  sitr3(sx, sy, sz, ax, ay, az, tx, ty, tz : real)

4.2.2.2 inquiring retained segment dynamic attribute values
inquire_visibility(visibility)
  ivisib(var vis : boolean)
inquire_highlighting(highlighting)
  ihilit(var hilit : boolean)
inquire_detectability(detectability)
  idtect(var dtect : integer)
inquire_image_translate_2(tx,ty)
  iitr2(var tx,ty : real)
inquire_image_transformation_2(mx,sy,a,tx,ty)
  iitr2(var sx,sy,az,tx,ty : real)
inquire_image_transformation_3(sx,sy,sz,ax,ay,az,
  tx,ty,tz)
  iitr3(var sx,sy,sz,ax,ay,az,tx,ty,tz : real)

4.2.2.3 setting a retained segment's dynamic attribute
values
set_segment_visibility(segment_name_visibility)
  ssvis(segment1 : nametype; vis : boolean)
set_segment_highlighting(segment_name,highlighting)
  sshilt(segment1 : nametype; hilit : boolean)
set_segment_detectability(segment_name,detectability)
  ssdet(segment1 : nametype; det : integer)
set_segment_image_translate_2(segment_name,tx,ty)
  ssitn2(segment1 : nametype; tx,ty : real)
set_segment_image_transformation_2(segment_name,sx,
  sy,az,tx,ty)
  ssitr2(segment1 : nametype; sx,sy,az,tx,ty : real)
set_segment_image_transformation_3(segment_name,sx,
  sy,sz,ax,ay,az,tx,ty,tz)
  ssitr3(segment1 : nametype; sx,sy,sz,ax,ay,az,
  tx,ty,tz : real)

4.2.2.4 inquiring a retained segment's dynamic attribute
values
inquire_segment_visibility(segment_name,visibility)
  isvis(segment1 : nametype; var vis : boolean)
inquire_segment_highlighting(segment_name,highlighting)
  ishilt(segment1 : nametype; var hilt : boolean)
inquire_segment_detectability(segment_name,
  detectability)
  isdet(segment1 : nametype; var det : integer)
inquire_segment_image_translate_2(segment_name,tx,ty)
  isitn2(segment1 : nametype; var tx,ty : real)
inquire_segment_image_transformation_2(segment_name,
  sx,sy,az,tx,ty)
  isitr2(segment1 : nametype; var sx,sy,az,tx,ty : real)
inquire_segment_image_transformation_3(segment_name,
  sx,sy,sz,ax,ay,az,tx,ty,tz)
  isitr3(segment1 : nametype; var sx,sy,sz,ax,ay,az,
  tx,ty,tz : real)

4.2.3 attribute value ranges
  4.2.3.1 primitive static attributes
  4.2.3.2 retained segment static attributes
  4.2.3.2 retained segment dynamic attributes

5 viewing transformations
  5.1 overview

E-6
5.1.1 2d versus 3d
  5.1.1.1 2d viewing operation
  5.1.1.2 3d viewing operation
5.1.2 view plane
5.1.3 projection
5.1.4 windows, view volumes, and clipping
5.1.5 normalized device coordinate space, viewsurfaces, and viewports
5.1.6 world coordinate transformations
5.1.7 image transformation
5.1.8 sample scenario for defining the viewing transformation

5.2 functional capabilities
5.2.1 2d viewing operation
  5.2.1.1 set_window(xmin,xmax,ymin,ymax)
    swindo(unimax,vmin,vmax : real)
  5.2.1.2 set_view_up_2(dx_up,dy_up)
    svup2(dxup,dyup : real)
  5.2.1.3 set_ndc_space_2(width,height)
    sndcs2(width,height : real)
  5.2.1.4 set_viewport_2(xmin,xmax,ymin,ymax)
    svprt2(xmin,xmax,ylinymax : real)
  5.2.1.5 inquiry for individual viewing operation parameters
    inquire_window(xmin,xmax,ymin,ymax)
      iwindo(var umanx,ymin,vmax : real)
    inquire_view_up_2(dx_up,dy_up)
      ivup2(var dxup,dyup : real)
    inquire_ndc_space_2(width,height)
      indcs2(var width,height : real)
    inquire_viewport_2(xmin,xmax,ymin,ymax)
      ivprt2(var xmin,xmax,ymax : real)
  5.2.1.6 map_ndc_to_world_2(ndc_x,ndc_y,x,y)
    ndcwcs2(nx,ny : real; var wx,wy : real)
  5.2.1.7 map_world_to_ndc_2(x,y,ndc_x,ndc_y)
    wcsndc2(wx,wy : real; var nx,ny : real)
5.2.2 3d viewing operation
  5.2.2.1 set_view_reference_point(x_ref,y_ref,z_ref)
    svrfpt(xref,yref,zref : real)
  5.2.2.2 set_view_plane_normal(dx_norm,dy_norm,dz_norm)
    svpnor(dxnorm,dynorm,dznorm : real)
  5.2.2.3 set_view_plane_distance(view_distance)
    svpdis(dist : real)
  5.2.2.4 set_view_depth(front_distance,back_distance)
    svdpth(front,back : real)
5.2.2.5
set_projection(projection_type,dx_proj,dy_proj,dz_proj)
sproj(projtype : integer; dxproj,dyproj,dzproj : real)

5.2.2.6
* smwindo(umin,umax,vmin,vmax)

5.2.2.7
set_view_up_3(dx_up,dy_up,dz_up)
svup3(dx-up,dy-up,dz-up : real)

5.2.2.8
set_ndc_space_3(width,height,depth)
sndcs3(width,height,depth : real)

5.2.2.9
set_viewport_3(xmin,xmax,ymin,ymax,zmin,zmax)
svp3(xmin,xmax,ymin,ymax,zmin,zmax : real)

5.2.2.10
* set_viewing_parameters(viewing_parameter_array)

5.2.2.11 inquire for individual viewing operation parameters
inquire_view_reference_point(x_ref,y_ref,z_ref)
inrft(var xref,yref,zref : real)

5.2.2.12
* inquire_viewing_parameters(viewing_parameter_array)

5.2.2.13
* map_ndc_to_world_3(ndc_x,ndc_y,ndc_z,x,y,z)

5.2.2.14
* map_world_to_ndc_3(x,y,z,ndc_x,ndc_y,ndc_z)

5.2.3 viewing control
5.2.3.1
set_window_clipping(on_off)
sclipw(clip : boolean)

5.2.3.2 depth clippint
set_front_plane_clipping(on_off)
sclpfp(clip : boolean)
set_back_plane_clipping(on_off)
sclpbp(clip : boolean)

5.2.3.3
set_coordinate_system_type(type)
socrt(p(handedness : integer)

5.2.3.4
5.2.4 world coordinate transformations
5.2.4.1
* set_world Coordinate_matrix_2(matrix_2)
* set_world Coordinate_matrix_3(matrix_3)
5.2.4.2
* inquire_world Coordinate_matrix_2(matrix_2)
* inquire_world Coordinate_matrix_3(matrix_3)
5.2.5 default values
5.2.6 viewing specification validity

6 input primitives
6.1 overview
6.2 functional capabilities
6.2.1 logical input facilities
6.2.1.1 pick
6.2.1.2 keyboard
6.2.1.3 button
6.2.1.4 stroke
6.2.1.5 locator
6.2.1.6 valuator
6.2.2 minimum set of input devices
6.2.3 input device identification
6.2.4 initializing and enabling input devices
6.2.4.1
initialize_device(device_class,device_num)
initde(iclass : nametype; inum : integer)
6.2.4.2
initialize_group(device_class,device_num_array,n)
initgr(iclass : nametype; devarray : iarray; n : integer)
6.2.4.3
* enable_device(device_class,device_num)
6.2.4.4
* enable_group(device_class,device_num_array,n)
6.2.4.5
* disable_device(device_class,device_num)
6.2.4.6
* disable_group(device_class,device_num_array,n)
6.2.4.7
* disable_all()
6.2.4.8
terminate_device(device_class,device_num)
ternde(iclass : nametype; inum : integer)
6.2.4.9
terminate_group(device_class,device_num_array,n)
termgr(iclass : nametype; devarray : iarray; n : integer)
6.2.5 reading sampled devices
6.2.5.1
* read_locator_2(locator_num,x,y)
* read_locator_3(locator_num,x,y,z)
6.2.5.2
* readvaluator(valuator_num, value)

6.2.6 event handling
6.2.6.1
* await_event(time, event_class, event_num)
6.2.6.2
* flush_device_events(event_class, event_num)
6.2.6.3
* flush_group_events(event_class, event_num_array, n)
6.2.6.4
* flush_all_events()

6.2.7 associating devices
6.2.7.1
* associate(event_class, event_num, sampled_class, sampled_num)
6.2.7.2
* disassociate(event_class, event_num, sampled_class, sampled_num)
6.2.7.3
* disassociate_device(device_class, device_num)
6.2.7.4
* disassociate_group(device_class, device_num_array, n)
6.2.7.5
* disassociate_all()

6.2.8 accessing event report data
6.2.8.1
* get_pick_data(segment_name, pick_id)
6.2.8.2
* get_keyboard_data(input_string, num_input)
6.2.8.3
* get_stroke_data_2(array_size, x_array, y_array, num_positions)
* get_stroke_data_3(array_size, x_array, y_array, z_array, num_positions)
6.2.8.4
* get_locator_data_2(locator_num, x, y)
* get_locator_data_3(locator_num, x, y, z)
6.2.8.5
* get_valuator_data(valuator_num, value)

6.2.9 synchronous input functions
6.2.9.1
* await_any_button(time, button_num)
  awaitbut(var butnum : integer)
6.2.9.2
* await_pick(time, pick_num, segment_name, pick_id)
  awaitpick(picnum : integer; var segname1 : nametype;
           var pickid : integer)
6.2.9.3
* await_keyboard(time, keyboard_num, input_string, length)
  awaitkey(keynum : integer; var string : charline;
            var length : integer)
6.2.9.4
* await_stroke_2(time, stroke_num, array_size, x_array,
                 y_array, num_positions)
  awaitstr2(strokenum : integer; arraysiz : integer)
var xarray, yarray : array;
var numpos : integer)
await_stroke_3(time, stroke_num, array_size, x_array,
y_array, z_array, num_positions)
awaitstr3(strokenum : integer; arraysiz : integer;
var xarray, yarray, zarray : array;
var numpos : integer)

6.2.9.5
await_any_button_get_locator_2(time, locator_num, button_num, x, y)
polloc2(locnum : integer; var butnum : integer;
var lx, ly : real)
await_any_button_get_locator_3(time, locator_num, button_num, x, y, z)
polloc3(locnum : integer; var butnum : integer;
var lx, ly, lz : real)

6.2.9.6
await_any_button_get_valuator(time, valuator_num, button_num, value)
polval(valnum : integer; var butnum : integer;
var rvalue : real)

6.2.10 device echoing
6.2.10.1
set_echo(device_class, device_num, echo_type)
setechod(iclass : nametype; inum : integer;
echo : integer)

6.2.10.2
set_echo_group(device_class, device_num_array, n, echo_type)
setechog(iclass : nametype; devarray : iarray;
n, echo : integer)

6.2.10.3
* set_echo_segment(device_class, device_num, segment_name)

6.2.10.4
* set_echo_surface(device_class, device_num, surface_name)

6.2.10.5
* set_echo_position(device_class, device_num, echo_x, echo_y)

6.2.11 setting input device characteristics
6.2.11.1
set_pick(pick_num, aperture)
setpic(picnum : integer; picapr : aperture)

6.2.11.2
set_keyboard(keyboard_num, buffer_size, initial_string, cursor_start)
setkey(keynum : integer; bufsiz : integer; initstrg : char-line; startpos : integer)

6.2.11.3
set_button(button_num, prompt_switch)
setbut(butnum, prompt : integer)

6.2.11.4
set_all_buttons(prompt_switch)
setallbut(prompt : integer)

6.2.11.5
set_stroke(stroke_num, buffer_size, distance, time)
setstroke(strokenum : integer; bufsiz : integer;
6.2.11.6
set_locator_2(locator_num, loc_x, loc_y)
setloc2(locnum : integer; locx, locy : real)
set_locator_3(locator_num, loc_x, loc_y, loc_z)
setloc3(locnum : integer; locx, locy, locz : real)

6.2.11.7
set_locport_2(locator_num, xmin, xmax, ymin, ymax)
setlocp2(locnum : integer; xmin, xmax, ymin, ymax : real)
set_locport_3(locator_num, xmin, xmax, ymin, ymax, zmin, zmax)
setlocp3(locnum : integer; xmin, xmax, ymin, ymax, zmin, zmax : real)

6.2.11.8
set_valuator(valuator_num, initial_value, low_value, high_value)
setval(valnum : integer; initval, lowval, highval : real)

6.2.12 inquiry
6.2.12.1
inquire_input_capabilities(level, device_counts, timing)
inqicap(var level : integer; var devcnts : array6)

6.2.12.2
* inquire_input_device_characteristics(device_class, device_num, implementation, echo, view_surface, association_size, association_class, association_num, association_count, duplication_size, duplication_class, duplication_num, duplication_count, precision, delay)

6.2.12.3 inquire dimension
inquire_stroke_dimension(stroke_num, dimension)
inqstrdi(strokenum : integer; var dimens : integer)
inquire_locator_dimension(locator_num, dimension)
inqlocdi(locnum : integer; var dimens : integer)

6.2.12.4
inquire_device_status(device_class, device_num, initialized, enabled)
inqdevstat(iclass : nametype; inum : integer; var inited : boolean)

6.2.12.5
* inquire_device_associations(event_class, event_num, array_size, sampled_class_array, sampled_num_array, number_of_associations)

6.2.12.6 inquire input status parameters
inquire_echo(device_class, device_num, echo_type)
inquecho(iclass : nametype; inum : integer; var echo : integer)
* inquire_echo_surface(device_class, device_num, surface_name)
* inquire_echo_position(device_class, device_num, echo_x, echo_y)
inquire_pick(pick_num, aperture)
inqpic(picnum : integer; var picapr : aperture)
inquire_keyboard(keyboard_num, buffer_size, initial_string, cursor_start)
inqkey(keynum : integer; var bufsiz : integer)
initstrg : char-line; var startpos : integer)
inquire_button(button_num, prompt_switch)
inqbut(button_num : integer; var prompt : integer)
inquire_stroke(stroke_num, buffer_size, distance, time)
inqstro(stroke_num : integer; var buffer_size : integer; var distance, time : real)
inquire_locator_2(locator_num, loc_x, loc_y)
inqloc2(locator_num : integer; var loc_x, loc_y : real)
inquire_locator_3(locator_num, loc_x, loc_y, loc_z)
inqloc3(locator_num : integer; var loc_x, loc_y, loc_z : real)
inquire_locport_2(locator_num, xmin, xmax, ymin, ymax)
inqlocp2(locator_num : integer; var xmin, xmax, ymin, ymax : real)
inquire_locport_3(locator_num, xmin, xmax, ymin, ymax, zmin, zmax)
inqlocp3(locator_num : integer; var xmin, xmax, ymin, ymax, zmin, zmax : real)
inquire_valuator(valuator_num, initial_value, low_value, high_value)
inqval( valuator_num : integer; var initial_value, low_value, high_value : real)

6.2.12.7
* inquire_echo_segments(device_class, device_num,
array_size, echo_segment_array, number_echo_segments)

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

7 control

7.1 overview
7.1.1 initialization and termination
7.1.2 view surface control
7.1.3 picture change control
7.1.4 frame control
7.1.5 error handling

7.2 functional capabilities
7.2.1 initialization and termination

7.2.1.1 initialize_core(out_level, in_level, dimension, hidden_surface)

initcore() 

7.2.1.2
terminate_core()

terms()

7.2.2 view surface initialization and selection

7.2.2.1 initialize_view_surface(surface_name, type, mode)
initvs(devices : integer)

7.2.2.2
terminate_view_surface(surface_name)

terms()

7.2.2.3
* inquire_outputCapabilities(surface_name, levels,
physical, sizes, prim_attr, seg_attr, batching)

7.2.2.4
* select_view_surface(surface_name)

E-13
7.2.2.5
* deselect_view_surface(surface_name)

7.2.2.6
* inquire_selected_surfaces(array_size,view_surface_names,number_of_surfaces)

7.2.3 picture change control
7.2.3.1 immediate visibility control
7.2.3.2
set_immediate_visibility(immediacy)
  simed(vis : boolean)
7.2.3.3
* make_picture_current()
7.2.3.4
* batching of updates
7.2.3.5
* begin_batch_of_updates()
7.2.3.6
* end_batch_of_updates()
7.2.3.7
inquire_control_status()
  iconst(var vis,batch : boolean)
7.2.3.8
* set_visibilities(segment_name_array,visibility_array,n)

7.2.4 frame control
7.2.4.1
new_frame()
  newframe()

7.2.5 error handling
7.2.5.1
* report_most_recent_error(error_report)
7.2.5.2
* log_error(error_report)

7.2.6 color models
7.2.6.1
* set_color_model(model)
7.2.6.2
* inquire_color_model(model)

7.2.7 auxiliary mode
7.2.7.1
set_background_index(index)
  sbgndx(index : integer)
7.2.7.2
inquire_background_index(index)
  ibgndx(var index : integer)
7.2.7.3
* set_display_mode(mode)
7.2.7.4
* inquire_display_mode(mode)

7.2.8 pixel array functions
7.2.8.1
* set_pixel_array(index_array,columns,rows)
7.2.8.2
* inquire_pixel_array(columns,rows,index_array,n_columns,n_rows)
7.2.8.3
set_pixel_pattern_origin_2(x_abs, y_abs)

7.2.8.4

inquire_pixel_pattern_origin_2(x_abs, y_abs)

7.2.9 index table functions

7.2.9.1

define_color_index(surface_name, i, c1, c2, c3)

dclndx(i : integer; c1, c2, c3 : real)

7.2.9.2

define_intensity_index(surface_name, i, intensity)

7.2.9.3

inquire_color_index(surface_name, i, c1, c2, c3)

iclndx(i : integer; var c1, c2, c3 : real)

7.2.9.4

inquire_intensity_index(surface_name, i, intensity)

7.2.9.5

define_color_indices(surface_name, i1, i2, c1_array, c2_array, c3_array)

dclndc(i1, i2 : integer; c1array, c2array, c3array : ndxarray)

7.2.9.6

define_intensity_indices(surface_name, i1, i2, i_array)

7.2.9.7

inquire_color_indices(surface_name, i1, i2, c1_array, c2_array, c3_array)

iclndc(i1, i2 : integer; var c1array, c2array, c3array : ndxarray)

7.2.9.8

inquire_intensity_indices(surface_name, i1, i2, i_array)

7.2.10 index tables - standard assignment

7.2.10.1

define_standard_color_indices(surface_name, i1, i2, i3, low1, low2, low3, high1, high2, high3)

dstdcl(i1, i2, i3 : integer; low1, low2, low3, high1, high2, high3 : real)

7.2.10.2

define_standard_intensity_indices(surface_name, i, low, high)

7.2.10.3

inquire_standard_color_indices(surface_name, i1, i2, i3, low1, low2, low3, high1, high2, high3)

istdcl(var i1, i2, i3 : integer; var low1, low2, low3, high1, high2, high3 : real)

7.2.10.4

inquire_standard_intensity_indices(surface_name, i, low, high)

8 special interfaces between the core system and the application program

8.1 overview

8.2 functional capabilities

8.2.1 general escape functions

8.2.1.1

escape(function_name, parameter_count, parameter_list)

escape(rnum : integer)

8.2.1.2
* inquire_escape(function_name, support_indicator)

8.2.2 specific escape functions
8.2.2.1
*

escape(prompt, 4, parameter_list)

9 philosophy of interfacing the core system with its environment
9.1 overview
9.2 impact areas
9.2.1 operating systems
9.2.1.1 core system invocation
9.2.1.2 unescorted text
9.2.2 programming languages

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

2. table of contents of user level procedures

awaitbut 6.2.9.1
awaitkey 6.2.9.3
awaitpick 6.2.9.2
awaitstr2 6.2.9.4
awaitstr3 6.2.9.4
clrseg 7.2.3.5
cltseg 3.2.2.2
crrseg 3.2.1.1
crtseg 3.2.2.1
dclndc 7.2.9.5
dclndx 7.2.9.1
delall 3.2.1.4
derseg 3.2.1.3
dstdcl 7.2.10.1
escape 8.2.1.1
ibgnndx 7.2.7.2
iclndc 7.2.9.7
iclndx 7.2.9.3
iconst 7.2.3.7
idtect 4.2.2.2
iffindx 4.2.1.3
ihilit 4.2.2.2
iitn2 4.2.2.2
iitr2 4.2.2.2
iitr3 4.2.2.2
ilndx 4.2.1.3
ilstyl 4.2.1.3
ilwid 4.2.1.3
imksym 4.2.1.3
indcs2 5.2.1.5
indcs3 5.2.2.11
initcore 7.2.1.1
initde 6.2.4.1
initgr 6.2.4.2
initvs 7.2.2.1
inqbut 6.2.12.6
inqdevstat 6.2.12.4
<table>
<thead>
<tr>
<th>Command</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>plina2</td>
<td>2.2.2.2</td>
</tr>
<tr>
<td>plina3</td>
<td>2.2.2.2</td>
</tr>
<tr>
<td>plinr2</td>
<td>2.2.2.2</td>
</tr>
<tr>
<td>plinr3</td>
<td>2.2.2.2</td>
</tr>
<tr>
<td>pmrka2</td>
<td>2.2.4.2</td>
</tr>
<tr>
<td>pmrka3</td>
<td>2.2.4.2</td>
</tr>
<tr>
<td>pmrkr2</td>
<td>2.2.4.2</td>
</tr>
<tr>
<td>pmrkr3</td>
<td>2.2.4.2</td>
</tr>
<tr>
<td>polloc2</td>
<td>6.2.9.5</td>
</tr>
<tr>
<td>polloc3</td>
<td>6.2.9.5</td>
</tr>
<tr>
<td>polval</td>
<td>6.2.9.6</td>
</tr>
<tr>
<td>polya2</td>
<td>2.2.5</td>
</tr>
<tr>
<td>polya3</td>
<td>2.2.5</td>
</tr>
<tr>
<td>polyr2</td>
<td>2.2.5</td>
</tr>
<tr>
<td>polyr3</td>
<td>2.2.5</td>
</tr>
<tr>
<td>renseg</td>
<td>3.2.1.5</td>
</tr>
<tr>
<td>sbgndx</td>
<td>7.2.7.1</td>
</tr>
<tr>
<td>sclipw</td>
<td>5.2.3.1</td>
</tr>
<tr>
<td>sclpbp</td>
<td>5.2.3.2</td>
</tr>
<tr>
<td>sclpfp</td>
<td>5.2.3.2</td>
</tr>
<tr>
<td>scortp</td>
<td>5.2.3.3</td>
</tr>
<tr>
<td>sdtect</td>
<td>4.2.2.1</td>
</tr>
<tr>
<td>setallbut</td>
<td>6.2.11.4</td>
</tr>
<tr>
<td>setbut</td>
<td>6.2.11.3</td>
</tr>
<tr>
<td>setechod</td>
<td>6.2.10.1</td>
</tr>
<tr>
<td>setechog</td>
<td>6.2.10.2</td>
</tr>
<tr>
<td>setkey</td>
<td>6.2.11.2</td>
</tr>
<tr>
<td>setloc2</td>
<td>6.2.11.6</td>
</tr>
<tr>
<td>setloc3</td>
<td>6.2.11.6</td>
</tr>
<tr>
<td>setlocp2</td>
<td>6.2.11.7</td>
</tr>
<tr>
<td>setlocp3</td>
<td>6.2.11.7</td>
</tr>
<tr>
<td>setpic</td>
<td>6.2.11.1</td>
</tr>
<tr>
<td>setstroke</td>
<td>6.2.11.5</td>
</tr>
<tr>
<td>setval</td>
<td>6.2.11.8</td>
</tr>
<tr>
<td>sfldnx</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>shilit</td>
<td>4.2.2.1</td>
</tr>
<tr>
<td>simed</td>
<td>7.2.3.2</td>
</tr>
<tr>
<td>sitn2</td>
<td>4.2.2.1</td>
</tr>
<tr>
<td>sitr2</td>
<td>4.2.2.1</td>
</tr>
<tr>
<td>sitr3</td>
<td>4.2.2.1</td>
</tr>
<tr>
<td>slndx</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>slstil</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>slwid</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>smksym</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>sndcs2</td>
<td>5.2.1.3</td>
</tr>
<tr>
<td>sndcs3</td>
<td>5.2.2.8</td>
</tr>
<tr>
<td>spen</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>spesty</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>spid</td>
<td>4.2.1.1</td>
</tr>
<tr>
<td>sproj</td>
<td>5.2.2.5</td>
</tr>
<tr>
<td>ssdet</td>
<td>4.2.2.3</td>
</tr>
<tr>
<td>sshilt</td>
<td>4.2.2.3</td>
</tr>
<tr>
<td>ssitn2</td>
<td>4.2.2.3</td>
</tr>
<tr>
<td>ssitr2</td>
<td>4.2.2.3</td>
</tr>
<tr>
<td>ssitr3</td>
<td>4.2.2.3</td>
</tr>
</tbody>
</table>
3. application program example

A sample program is created here to demonstrate how to use the pascal core package. The test program will draw a diagonal line from the lower left corner to the upper right corner of the tk4014 or the hp7220a.

1. First copy "header.i" file to the pascal file you will use to create your program.

        ex  >cp header.i batch2.p

Then type your program name after the comments in the header.i file. Also add the necessary variables to the 'const', 'var', and 'type' locations.

Start your program after the end of the header.i file.

(* *******************************************************  
  * date: 12 oct 83  
  * version: 1.0  
  * name: header  
  * description: a header file that includes all the necessary  
  *   variables and procedure parameters needed to  
  *   build a core application program  
  * operating system: UNIX  
  * language: pascal  
  * inputs: n/a  
  * outputs: n/a  
  * global variables used: defconst.h, deftype.h, extvar.h,  
  *   error.h, driverext.h, ddutilext.h, ctloext.h,  
  *   ct1lext.h, utillext.h, util2ext.h, common.h,  
  *   userext.h  
  * global variables changed: n/a  
  *********************************************************)

E-19
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: n/a
* calling modules: n/a
* author: Capt John W. Taylor

************************************************************
(* this file is to be included at the beginning of every *)
(* main core pascal program. *)
(* this file should be the first text entered into the main *)
(* program that is being built. ex. *)
(* cp header.i filename.p *)
(* this will create the necessary include files needed to *)
(* run the pascal core system. the remaining code is now *)
(* added on. *)
(* *)

(* program name starts here *)
(* ex. *)
(* program test(input,output); *)

program batch2(input,output); (* program name added here *)

const (* global constants used by the core system *)
#include 'defconst.h'

type (* global types used by the core system *)
#include 'deftype.h'

var (* global variables used by the core system *)
#include 'extvar.h'

devices : integer; (* devices is the only variable added *)

(* include files with the procedure declarations called by *)
(* the main pascal program *)
#include 'error.h'
#include 'driverext.h'
#include 'ddutilext.h'
#include 'ct10ext.h'
#include 'ct11ext.h'
#include 'util2ext.h'
#include 'utilext.h'
#include 'common.h'
#include 'userext.h'

(* this is about the simplest core program you can write. it *)
(* simply does what is necessary to draw a diagonal line on the *)
(* screen. *)
begin
  writeln;
  writeln('device 1 = tk4014');
  writeln('device 2 = hp7220a');
  writeln;
  writeln(' device =');
  readln(devices);
  initcore; (* initialize core *)
  (* the system *must* be initialized before anything else
  can be done. therefore initcore must be the first core
  routine called *)
  initvs(devices); (* select either the tk4014 or hp7220a *)
  if (devices = 2) then
    begin
      spen(1);
    end;
  crtsen; (* create temporary segment *)
  (* if anything is to be drawn, it must be a primitive
  within a segment. to keep things simple here, we are
  creating a temporary segment to put our line primitive
  in. *)
  lina2(1,1); (* line absolute 2 *)
  (* draw a line from the current operating point ((0,0) as
  initialized) to the absolute point (1,1). since the
  default window is 0-1 in the x and y directions, this
  will draw a diagonal line across the screen. *)
  termvs; (* take tk4014 or hp7220a off-line *)
  termcore; (* terminate core *)
  (* this routine will close the temporary segment and
  terminate the core system. *)
end.

2. now compile the program to run:

   ex : >pc -w batch2.p lib1 <cr>

       -w : suppresses any warning messages

       lib1 : library containing the object code for
       all the core routines

3. to run the program use either a v100 terminal connected to
a hp7220a or a visual 550 terminal in the tk4014
emulation mode.

   ex : >a.out

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

4. a program called intcore.run is available to demonstrate
how the core system graphics works. it is menu driven
which allows a user to demonstrate many aspects of the
core system on a computer.

the program is self explanatory. to execute the program

> intcore.run

type the following run file:

here is a list of the menu screens that appear in this program:

**** main menu of interactive core routines ****
0 - this menu
2 - output primitives
3 - picture segmentation and naming
4 - primitive & segment attributes
5 - viewing operations and coordinate transformations
6 - input primitives
7 - control
8 - special interfaces
9 - miscellaneous routines

**** output primitives routines menu ****
2 - this menu
200 - mova2 201 - mova3 202 - movr2
203 - movr3 204 - ipsn2 205 - ipsn3
206 - lina2 207 - lina3 208 - linr2
209 - linr3 210 - marka2 211 - marka3
212 - markr2 213 - markr3 214 - plina2
215 - plina3 216 - plinr2 217 - plinr3
218 - polya2 219 - polya3 220 - polyr2
221 - polyr3 222 - pmrka2 223 - pmrka3
224 - pmrkr2 225 - pmrkr3 226 - text
227 - setmargin 228 - newline

**** picture segmentation & naming routines menu ****
3 - this menu
300 - crrseg 301 - clrseg 302 - derseg
303 - delall 304 - renseg 305 - irsnam
306 - ioseg 307 - crtseg 308 - cltseg
309 - iotseg

**** primitive & segment attributes routines menu ****
4 - this menu
400 - slndx 401 - slstyl 402 - slwid
403 - spen 404 - smksym 405 - spesty
406 - spid 407 - ilndx 408 - ilstyl
409 - ilwid 410 - ipen 411 - imksym
412 - ipesty 413 - ipid 414 - strtyp
415 - svisib 416 - shilit 417 - sdetect
418 - sitn2 419 - sitr2 420 - sitr3
421 - itrtyp 422 - ivisib 423 - ihilit
viewing operations and coordinate transformation routines

5 - this menu

500 - swindo 501 - svup2 502 - sndcs2
503 - svprtl2 504 - iwindow 505 - ivup2 506 - indcs2 507 - ivprt2 508 - svrfpt
509 - svpnor 510 - svpdis 511 - svdpth 512 - sproj
513 - svup3 514 - sndcs3 515 - sproj 516 - ivrfpt 517 - ivpnor
518 - ivpdis 519 - ivdpth 520 - iproj 521 - ivup3
522 - indcs3 523 - ivprtl3 524 - sclipw 525 - sclpfps 526 - sclpbp
527 - scortp 528 - ivcpars 529 - ndcwcs2

input routines menu

6 - this menu

601 - initde 602 - initgr 603 - termde 604 - termgr 605 - awaitbut 606 - awaitpick
607 - awaitkey 608 - awaitstr2 609 - awaitstr3 610 - polloc2 611 - polloc3 612 - polval
613 - setechod 614 - setechog 615 - setpic 616 - setkey 617 - setbut 618 - setallbut
619 - setstroke 620 - setloc2 621 - setloc3 622 - setlocp2 623 - setlocp3 624 - setval
625 - inqicap 626 - inqstrdi 627 - inqlocdi 628 - inqdevstat 629 - inqecho 630 - inqpic
631 - inqkey 632 - inqbut 633 - inqstroke 634 - inqloc2 635 - inqloc3 636 - inqlocp2
637 - inqlocp3 638 - inqval

control routines menu

7 - this menu

700 - initcore 701 - newframe 702 - termcore 720 - sbgndx 721 - ibgndx 730 - dclndx
731 - iclndx /31 - iclndx /732 - dclndc 733 - iclndc 734 - dstdc1 735 - stdc1 736 - simmed
737 - iconst 740 - initvs 741 - termvs

special interfaces menu

8 - this menu
801 - test pgm for viewing pipeline 802 - print the contents of the display file
803 - print contents of one segment

******** miscellaneous routines menu ********

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>this menu</td>
</tr>
<tr>
<td>903</td>
<td>lineindex</td>
</tr>
<tr>
<td>904</td>
<td>xcoor</td>
</tr>
<tr>
<td>905</td>
<td>ycoor</td>
</tr>
<tr>
<td>906</td>
<td>zcoor</td>
</tr>
<tr>
<td>907</td>
<td>coorcnt</td>
</tr>
<tr>
<td>909</td>
<td>prims</td>
</tr>
<tr>
<td>912</td>
<td>textindex</td>
</tr>
<tr>
<td>921</td>
<td>chars</td>
</tr>
<tr>
<td>924</td>
<td>fillindex</td>
</tr>
<tr>
<td>926</td>
<td>symbol</td>
</tr>
<tr>
<td>927</td>
<td>segname</td>
</tr>
<tr>
<td>929</td>
<td>highlighted</td>
</tr>
<tr>
<td>931</td>
<td>imagetrans</td>
</tr>
<tr>
<td>935</td>
<td>bgndnx</td>
</tr>
<tr>
<td>944</td>
<td>ndcspc</td>
</tr>
</tbody>
</table>

998 - snapshot
Appendix F

Core Routine Calling/Called By Other Core Routines

THE ROUTINES ARE LISTED IN THEIR INCLUDE FILES IN THE ORDER
OF MOST DEPENDENCE AT THE TOP, TO THE LEAST DEPENDENCE AT THE
BOTTOM.

I. Error.h Files
II. Dri
erext.h Files
III. Ddutil
text.h Files
IV. CtlOext.h Files
V. Ctl1ext.h Files
VI. Util2ext.h Files
VII. Uti
text.h Files
VIII. Common.h Files
IX. Userext.h Files
X. Device Drivers

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

F-1
I.  Error.h

error: prints error messages to screen
    ***** no calls
    ------ called by most others

II. Driverext.h

pmovab: move to specified screen coordinates
    ***** no calls
    ------ called by pmovre, pitext, pimark, escapedd
pdrlin: draw a line between given points
    ***** no calls
    ------ called by pdrlab, piline, plpgon, ppoly
perlin: erases line between given points
    ***** no calls
    ------ called by perlab, ppoly
pinit: initialize display device
    ***** no calls
    ------ called by initdd
perase: erases entire screen
    ***** no calls
    ------ called by newframedd, escapedd
pexec: sends the contents of the command buffer to display device
    ***** no calls
    ------ called by plpgon, funnel, newframedd, escapedd
pterm: deassign display device
    ***** no calls
    ------ called by teracore
pgetpo: return screen limits
    ***** no calls
    ------ not called
pmovre: moves the current screen pixels relative to their current position
    ***** calls pmovab
    ------ not called
pvport: set screen limits
    ***** no calls
    ------ called by funnel
psetco: set background color
    ***** no calls
    ------ called by concol, newframedd, escapedd
psetgr: set current grey scale
    ***** no calls
    ------ called by escapedd
ptterm: gets a flag to determine where output is to be directed
    ***** no calls
    ------ not called
pdrpab: draws a point at specified location
    ***** no calls
called by pdrcir, pimark

perpab: erases a point at a specified location

**** no calls

called by percir

pdrlab: draw a line from current screen position to newly specified position

**** calls pdrlin

---- not called

perlab: erase a line from the current screen position to the newly specified screen position

**** calls perlin

---- not called

pdrrec: draws a rectangle given opposing corners

**** no calls

---- called by pdrdsk, newframedd

perrec: erases a rectangle given opposing corners

**** no calls

---- called by perdsk

ppoly: fill a polygon area on screen

**** calls pdrlin, perlin

---- called by pipgon

pdrcir: draw a circle

**** calls pdrpab

---- called by escapedd

percir: erase a circle

**** calls perpab

---- called by escapedd

pdrdsk: fill a circle

**** calls pdrrec

---- called by escapedd

perdsk: erase a disk

**** calls perrec

---- called by escapedd

pchar: print a string of characters

**** no calls

---- called by plower, pitext, pimark, escapedd

plower: convert upper case characters to lower case

**** calls pchar

---- called by escapedd

perchr: erase characters

**** no calls

---- called by escapedd

pcread: get X,Y cursor position

**** no calls

---- called by ptrack

pplace: place cursor at specified point

**** no calls

---- called by pputcur


III. Ddutiltext.h

concol: convert color index from DI core system to graphics device equivalent

F-3
**** calls psetco
----- called by pline, pitext, plpgon, pimark
scx : converts ndc X coordinates in DI core system to its
equivalent screen coordinates
**** no calls
----- called by pline, pitext, plpgon, pimark, pputcur,
    escaped, txtcoor
scy : converts ndc Y coordinates in DI core system to its
    equivalent screen coordinates
**** no calls
----- called by pline, pitext, plpgon, pimark, pputcur,
    escaped, ddnewline, txtcoor
ndcx : converts X screen coordinates to ndc coordinates
**** no calls
----- called by ptrack, txtcoor
ndcy : converts Y screen coordinates to ndc coordinates
**** no calls
----- called by ptrack, ddnewline, txtcoor
imqtrans : performs image transformations of coordinates passed
to the driver from the DI core system in 3D
**** no calls
----- called by pline, pitext, plpgon, pimark,
    awaitpick
makemat : creates a 4X4 image transformation matrix from a
    1X9 matrix
**** no calls
----- called by pline, pitext, plpgon, pimark,
    awaitpick

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

IV. Ctl0ext.h

pline : DD routine to display a line or polyline on the
    graphics device
    ***** calls error, concol, makemat, imqtrans, pdrlin,
        scx, scy
    ----- called by funnel
pitext : DD routine to display a text string on the graphics
device
    ***** calls error, concol, makemat, imqtrans, pmovab,
        scx, scy, pchar
    ----- called by funnel
plpgon : DD routine to display a filled polygon on the
    graphics device
    ***** calls error, concol, makemat, imqtrans, ppolyl
        pexec, pdrlin, scx, scy
    ----- called by funnel
pimark : DD routine to display a marker symbol or polymarker
    on the graphics device
    ***** calls error, concol, makemat, imqtrans, pdrpab,
        pmovab, pchar, scx, scy
    ----- called by funnel
ptrack : read the position of the cursor on screen device
    and return X,Y coordinates
**** calls pcread, ndcx, ndcy
----- called by pgetloc2, pgetstr2, pgetcursor
putcur: place cursor on the screen device
**** calls pplace, scx, scy
----- called by pgetloc2

V. Ctl1ext.h

initdd: initialize graphic device driver
**** calls pinit
----- called by initcore
funnel: interface between DD and DI routines
**** calls pvport, piline, pitext, pilpolygon, pimark, pexec
----- called by addprim, drawseg, ddnewframe
newframedd: clears the graphic driver screen
**** calls perase, psetco, pdrrec, pexec
----- called by ddnewframe
escapedd: allow user to access non-standard core routines available for graphics device
**** calls pdrcir, pdrdsk, perdisk, pmovea, pchar, 
scx, scy, plower, pexec, perase, psetco, psetgr, 
perchr, percir
----- called by escape
initinput: initialize input variables
**** no calls
----- called by initcore
pgetbut: get button number
**** no calls
----- called by awaitbut, polval
pgetkey: get string of characters
**** no calls
----- called by awaitkey
pgetval: get input value
**** no calls
----- called by polval
pgetloc2: get the locator position on tablet from joystick
**** calls pputcur, ptrack
----- called by polloc2
pgetloc3: 3D cursor location input
**** no calls
----- called by polloc3
pgetstr2: get stroke position from tablet
**** calls ptrack
----- called by awaitstr2
pgetstr3: 3D stroke device input
**** no calls
----- called by awaitstr3
pgetcursor: get cursor position from tablet
**** calls ptrack
----- called by awaitpick, pickxy

********************************************************************
VI. Util2ext.h

usertrans : transforms the points in a primitive via user
    specified modeling matrix
    ***** no calls
        ----- called by addprim
worldtrans : transforms the points in a primitive via world
    coordinate transformation
    ***** no calls
        ----- called by addprim
unitvector : computes normalized version of the first
    coordinate of the vector
    ***** no calls
        ----- called kby sproj, svpnor, svup2, svup3
realequal : compares 2 real numbers for equality
    ***** no calls
        ----- called by vtran, crrseg, iitn2, iiitr2, indcs2,
            ipsn2, ivpitr2, ivup2, svndcs2, svndcs3, sproj,
            ssitn2, ssitr2, ssitr3, svpnor, svup2, svup3
dotproduct : compute the dotproduct of 2 vectors
    ***** no calls
        ----- called by vtran
crossproduct : compute the crossproduct of 2 vectors
    ***** no calls
        ----- called by vtran
clipper : clips a normalized, transformed primitive
    ***** no calls
        ----- called by addprim
perdiv : performs a perspective division of a primitive in a
    truncated right regular pyramid
    ***** no calls
        ----- called by addprim
winvewmap : maps from window coordinates to viewport coordinates
    ***** no calls
        ----- called by addprim
primxtent : calculates the extent of the corners of an enclosing
    box used by the pick operation
    ***** no calls
        ----- called by addprim
invmat : invert a 4X4 matrix
    ***** calls prtmat
        ----- called by ndcwcs2

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

VII. Utilext.h

eqnames : compares 2 strings for equality
    ***** no calls
        ----- called by findseg, prevseg
initprim : initialize attributes of primitives to current
    values of global primitive attributes
    ***** no calls
        ----- called by lina2, lina3, linr2, linr3, marka2,
            marka3, markr2, markr3, plina2, plina3, plinr2,
plmr3, pmrka2, pmrka3, pmrkr2, pmrkr3, polya2, polya3, polyr2, polyr3, textgr

addprim : adds the primitive to the currently open segment if the segment is a retained one
        ***** calls usertrans, worldtrans, clipper, perdiv, winvewmap, funnel, primxent
        ----- called by lina2, lina3, linr2, linr3, marka2, marka3, markr2, markr3, plina2, plina3, plinr2, plinr3, pmrka2, pmrka3, pmrkr2, pmrkr3, polya2, polya3, polyr2, polyr3, textgr

emptyseg : delete all primitives in segment
        ***** no calls
        ----- called by cltseg, delall, derseg, termcore

drawseg : draw all primitives in a segment
        ***** calls funnel
        ----- called by simmed, endbupdt, ssvis

findseg : find the requested segment
        ***** calls eqnames
        ----- called by crrseg, derseg, isdet, ishilt, isitn2, isitr2, isitr3, isttyp, isvis, renseg, ssdet, sshilt, ssitn2, ssitr2, ssitr3, ssvis

prevseg : locate the requested segment, then return a pointer to the previous segment in the list
        ***** calls eqnames
        ----- called by derseg

vtran : calculates the viewing transformation matrix from the current values of the viewing parameters
        ***** calls prtmatl, crossproduct, realequal, dotproduct
        ----- called by ndcwcs2, wcsndc2, crrseg, crrseg

ddnewframe : clear the screen to the background color
        ***** calls newframedd, funnel
        ----- called by simmed, delall, derseg, newframe, endbupdt, sshilt, ssitn2, ssitr2, ssitr3, ssvis

ddclrseg : closes the currently open retained segment
        ***** no calls
        ----- called by clrseg, delall, derseg, termcore

initdi : initializes all device independent parts of core system
        ***** no calls
        ----- called by initcore

locrec : locates the input device record
        ***** calls error
        ----- called by awaitbut, awaitkey, awaitpick, awaitstr2, awaitstr3, initde, initgr, inqdevstat, inqecho, inqkey, inqloc2, inqloc3, inqlocdi, inqlocp2, inqlocp3, inqpic, inqstrdi, inqstroke, inqval, pickxy, polloc2, polloc3, polval, setallbut, setbut, secho, setechog, setkey, setloc2, setloc3, setlocp2, setlocp3, setpic, setstroke, setval, termde, termgr

makinvmat : creates an inverse 4X4 transformation matrix from a 1X9 matrix
        ***** no calls
        ----- not called

******************************************************************************
VIII. Common.h

ndcwcs2: map ndc coordinates to wcs coordinates
        ***** calls error, vtran, invmat
        ------ called by ddnewline, txtcoor
wcsndc2: map wcs to ndc coordinates
        ***** calls error, vtran
        ------ called by ddnewline, txtcoor, setmargin
ddnewline: goes to beginning of new line
        ***** calls ndcwcs2, wcsndc2, ndcy, scy
        ------ called by newline
simmed: sets the value of immediate visibility
        ***** calls error, ddnewframe, drawseg
        ------ called by mkpiccur, makepiccurrent
txtcoor: computes diagonal of a text string in wcs 2 for later
        use in picking string
        ***** calls wcsndc2, ndcwcs2, scx, scy, ndcx, ndcy
        ------ called by textgr

IX. Userext.h Files --- none of these routines are called by
any other routine

awaitbut: wait for button device, get number
        ***** calls error, locrec, pgetbut
awaitkey: wait for keyboard event, get string
        ***** calls error, locrec, pgetkey
awaitpick: get segment and output primitive
        ***** calls makeinvmat, pgetcursor, error, locrec
        contains its own makemat, imagtrans, imagextnt
        ------ called by makeinvmat, pgetcursor, error, locrec
awaitstr2: wait for event, get X,Y stroke data
        ***** calls error, locrec, pgetstr2
awaitstr3: wait for event, get X,Y,Z stroke data
        ***** calls error, locrec, pgetstr3
clrseg: closes the currently open retained segment
        ***** calls error, ddclrseg
cltseg: closes the currently open temporary segment
        ***** calls error, emptyseg
crrseg: create a new, empty retained segment
        ***** calls error, vtran, findseg, realequal
crtseg: creates a new, empty temporary segment
        ***** calls error, vtran
dclndc: define all indices in the color index table
        ***** calls error
dclndx: define specified entry in color index table
        ***** calls error
printpdf: print current display file
        ***** no calls
printseg: print current segment
        ***** no calls
delall: deletes all retained segments
        ***** calls error, ddnewframe, emptyseg, ddclrseg
derseg : deletes retained segment with specified segment name
  ***** calls error, dclirseg, ddnnewframe, emptyseg, findseg, prevseg
dstdcl : provides standard assignment for indices loading into the color table
  ***** calls error
saveenv : ,save current environment on stack
  ***** calls error
restoreenv : restores most recently saved environment
  ***** no calls
escape : access non standard core routines
  ***** calls error, escapedd
ibgnx : copies background index parameter
  ***** calls error
iclndc : copies color values of all indices
  ***** calls error
iclndx : copies color parameters of specific indices
  ***** calls error
iconst : gets values of immediate visibility and batching of updates
  ***** calls error
idtect : gets value of the global segment attribute of detectability
  ***** calls error
iflndx : gets the value of the global primitive attribute of fill index
  ***** calls error
ihilit : gets the value of the global segment attribute of highlighting
  ***** calls error
iitn2 : get value of image translation in X,Y
  ***** calls error, realequal
iitr2 : get value of image transformation in X,Y
  ***** calls error, realequal
iitr3 : gets the value of the global segment attribute of image transformation
  ***** calls error
ilndx : gets the value of the global primitive attributes of line index
  ***** calls error
ilstyl : gets the value of the global primitive attribute of line style
  ***** calls error
ilwid : gets the value of the global primitive attribute of line width
  ***** calls error
imksym : gets the value of the global primitive attribute of marker symbol
  ***** calls error
indcs2 : get limits of ndc space in X,Y
  ***** calls error, realequal
indcs3 : gets the limits of the portion of the display device mapped to by ndc values
  ***** calls error
initcore : initialize core system
calls error, initdd, initdi, initinput

initde: initialize specified device

initgr: initialize device group

initvs: initialize selected graphics device

inqbut: get button value

inqdevstat: get device status

inqecho: get echo of device

inqicap: inquire the input capabilities

inqkey: get keyboard

inqloc2: get locator device coordinates in X,Y

inqloc3: get locator device coordinates in X,Y,Z

inqlocdi: get the dimension of the specified locator device

inqlocp2: get bounds of the specified locator device in X,Y

inqlocp3: get bounds of the specified locator device in X,Y,Z

inqpic: get aperture characteristics of specified pick device

inqstrdi: get the dimension of the specified stroke device

inqstroke: copy current stroke buffer size, distance, and time parameters into storage variables

inqval: copy initial value, low value, and high value of selected valuator

ioseg: returns the name of the currently open retained segment if there is one

iotseg: returns a boolean value specifying if there is a currently open temporary segment

ipen: gets the value of the global primitive attribute of pen

ipesty: gets the value of the global primitive attribute of polygon edge style

ipid: gets the value of the global primitive attribute of pick id

iproj: gets the current value of the projection type for parallel or perspective

ipsn2: get current X,Y position

F-10
ipsn3: returns the value of the current operating point

irsnam: returns up to a max of the names of all currently existing retained segments

isdet: gets segment attribute of detectability

ishilt: gets segment attribute of highlighting

isitn2: gets segment attribute of image translation for X,Y

isitr2: gets segment attribute of image transformation for X,Y

isitr3: gets segment attribute of image transformation for X,Y,Z

istdcl: returns the nine parameters which describe the standard assignment of color index values

isttyp: get segment image transformation type

isvis: get segment visibility

itrtypr: gets the value of the global segment attribute of image transformation

itxndx: gets the value of the global primitive attribute of text index

ivcpar: gets the current values of window clipping

ivdpth: gets the view depth

ivisib: gets the value of the global segment attribute of visibility

ivpdis: gets the current value of the view plane distance

ivpnor: gets the current value of the view plane normal vector

ivprt2: get values of coordinates of lower left and upper right of rectangular viewport in X,Y

ivprt3: gets the minimum and maximum bounds of the viewport

ivrftp: gets the current value of the view reference point

ivup2: get current value of view up vector in X,Y

ivup3: gets the current value of the un-normalized version of the view up vector

iwindo: gets the current values of the lower left and upper
right corners of the window

***** calls error

lina2 : creates a line primitive with X,Y endpoints

***** calls error, initprim, addprim

lina3 : creates a line primitive with X,Y,Z endpoints

***** calls error, initprim, addprim

linr2 : creates a line primitive displayed with dx, dy endpoints

***** calls error, initprim, addprim

linr3 : creates a line primitive displayed with dx, dy, dz endpoints

***** calls error, initprim, addprim

marka2 : create a marker primitive at X,Y

***** calls error, initprim, addprim

marka3 : create a marker primitive at X,Y,Z

***** calls error, initprim, addprim

markr2 : create a marker primitive at location displaced by dx, dy

***** calls error, initprim, addprim

markr3 : create a marker primitive at location displaced by dx, dy, dz

***** calls error, initprim, addprim

mkpiccur : update picture

***** calls error, simmed

mova2 : sets the current operating point to the absolute location X,Y

***** calls error

mova3 : sets the current operating point to the absolute location X,Y,Z

***** calls error

movr2 : sets the current operating point to the relative location X,Y

***** calls error

movr3 : sets the current operating point to the relative location X,Y,Z

***** calls error

makepiccurrent : update picture

***** calls error, simmed

newframe : start a new frame

***** calls error, ddnewframe

newline : start new text string line

***** calls error, ddnewline

pickxy : wait for event, get segment name and picked point

***** calls pgetcursor, error, locrec, makeinvmat

plina2 : create a polyline primitive at position X,Y

***** calls error, initprim, addprim

plina3 : create a polyline primitive at position X,Y

***** calls error, initprim, addprim

plinr2 : create a polyline primitive displaced at dx, dy

***** calls error, initprim, addprim

plinr3 : create a polyline primitive displaced at dx, dy, dz

***** calls error, initprim, addprim

pmrka2 : create a polymarker primitive at position X,Y

***** calls error, initprim, addprim

pmrka3 : create a polymarker primitive at position X,Y,Z

***** calls error, initprim, addprim
pmrkr2: create a polymarker primitive displaced at dx, dy
    ***** calls error, initprim, addprim
pmrkr3: create a polymarker primitive displaced at dx, dy, dz
    ***** calls error, initprim, addprim
polloc2: wait for event, get X,Y location
    ***** calls error, locrec, pgetloc2
polloc3: wait for event, get X,Y,Z location
    ***** calls error, locrec, pgetloc3
polval: wait for event, get the value
    ***** calls error, locrec, pgetbut, pgetval
polya2: create a polygon primitive at X,Y
    ***** calls error, initprim, addprim
polya3: create a polygon primitive at X,Y,Z
    ***** calls error, initprim, addprim
polyr2: create a polygon primitive displaced at dx, dy
    ***** calls error, initprim, addprim
polyr3: create a polygon primitive displaced at dx, dy, dz
    ***** calls error, initprim, addprim
printer: diagnostic routine that displays input structure
    at once
    ***** no calls
renseg: rename retained segment
    ***** calls error, findseg
sbgndx: sets the background index to the given index
    ***** calls error
beginbupdt: sets the value of batch of update flag to true
    ***** calls error
endbupdt: sets the value of batch of update flag to false
    ***** calls error, ddnewframe, drawseg
sclipw: sets the viewing parameter of window clipping
    ***** calls error
sclpbp: sets the viewing parameter of back plane clipping
    ***** calls error
sclpfp: sets the viewing parameter of front plane clipping
    ***** calls error
scortp: sets the world coordinate system handedness to left
    or right handed
    ***** calls error
sdtect: sets the value of the global segment attribute of
detectability
    ***** calls error
setallbut: set prompt switch for all button devices
    ***** calls error, locrec
setbut: set prompt switch for specified button
    ***** calls error, locrec
setechod: set the echo type for specified device
    ***** calls error, locrec
setechog: set the echo type for all devices in the specified
    group
    ***** calls error, locrec
setkey: set current keyboard
    ***** calls error, locrec
setloc2: set locator device position to X,Y coordinate
    ***** calls error, locrec
setloc3: set locator device position to X,Y,Z coordinate
setlocp2 : set bounds of locator device in X,Y
setlocp3 : set bounds of locator device in X,Y,Z
setmargin : resets the lefmargin index to the current ndc X coordinate
setpic : set aperature size in X,Y for specified pick device
setstroke : set current stroke device
setval : set valuator device
sflndx : sets the value of the global primitive attribute of polygon fill index
shilit : sets the value of the global segment attribute of highlighting
sitn2 : sets the value of the global segment attribute of image transformation to the translation in the X,Y plane
sitr2 : sets the value of the global segment attribute of image transformation to 2-D transformation
sitr3 : sets the value of the global segment attribute of image transformation to 3-D transformation
slndx : sets the value of the global primitive attribute of line index
slstyl : sets the value of the global primitive attribute of line style
slwid : sets the value of the global primitive attribute of line width
smksym : sets the value of the global primitive attribute of marker symbol
sndcs2 : defines usable portion of the output device in X,Y
sndcs3 : defines usable portion of the output device in X,Y,Z
spen : sets the value of the global primitive attribute of pen
spesty : sets the value of the global primitive attribute of polygon edge style
spid : sets the value of the global primitive attribute of pickid
sproj : sets projection type to either parallel or perspective
ssdet: set segment detectability
   ***** calls error, findseg
sshilt: sets the segment attribute of highlighting of the segment name
   ***** calls error, findseg, ddnewframe
ssitn2: sets the segment attribute of image transformation of the segment name to the translation in the X,Y plane
   ***** calls error, findseg, ddnewframe, realequal
ssitr2: sets the segment attribute of image transformation of the segment name to the 2-D transformation
   ***** calls error, findseg, ddnewframe, realequal
ssitr3: sets the segment attribute of image transformation of the segment name to the 3-D transformation
   ***** calls error, findseg, ddnewframe, realequal
ssvis: sets the segment attribute of visibility
   ***** calls error, findseg, ddnewframe, drawseg
strtyp: sets the value of the global segment attribute of image transformation style
   ***** calls error
stxndx: sets the value of the global primitive attribute of text index
   ***** calls error
svdpth: sets the view depth
   ***** calls error
svvisib: sets the value of the global segment attribute of visibility
   ***** calls error
svpdis: sets the view plane distance
   ***** calls error
svpnor: sets the view plane normal
   ***** calls error, realequal, unitvector
svpnt2: defines the viewport to be a rectangle
   ***** calls error
svpnt3: defines the viewport in 3-D
   ***** calls error
svrfpt: sets the view reference point
   ***** calls error
svup2: sets the view up vector to a normalized version X,Y
   ***** calls error, realequal, unitvector
svup3: sets the view up vector to a normalized version X,Y,Z
   ***** calls error, realequal, unitvector
swindo: sets the window to be a rectangle
   ***** calls error
termcore: terminate the core system
   ***** calls error, ddclrseg, emptyseg, pterm
termde: terminate specified device
   ***** calls error, locrec
termgr: terminates all initialized devices in specified group
   ***** calls error, locrec
termvs: terminates the selected graphics device
   ***** calls error, pterm, escape
textgr: insert a string into the current segment
   ***** calls error, initprim, txtcoor, addprim
X. Device Drivers

Visual 550

---

alpha550 : places the visual 550 into alpha mode
    ***** calls out550
char550 : writes a string of characters (up to 80) to the
    ***** calls alpha550, out550, graph550
clr550 : clears the visual 550 screen
    ***** calls out550
cross550 : reads the cross hair position in x,y coordinates on
    the visual 550, also returns the character used to
    initiate the read
    ***** calls out550
graph550 : puts the visual 550 in graphics mode
    ***** calls out550
init550 : initializes the visual 550 screen
    ***** calls out550
line550 : draws a line between 2 given endpoints on visual 550
    ***** calls graph550, tran550
mode550 : turns the pixels on or off for the visual 550, with
    the pixels off, lines or points can be selectively
    deleted
    ***** calls out550
move550 : places the visual 550 cursor at a specified location
    ***** calls graph550, tran550
out550 : the central routine that outputs graphics commands
    to the visual 550
    ---- called by alpha550, char550, clr550, cross550,
        graph550, init550, mode550, size550, term550,
        tran550
size550 : set the character size for the visual 550
    ***** calls out550
term550 : terminates the visual 550
    ***** calls out550
tran550 : translates given x,y coordinate values into x,y
    high/low byte character values, then moves to the
    specified location on the visual 550
    ***** calls out550
    ---- called by line550, move550

Tektronics 4014

---

alpha14 : places the tk4014 into alphanumerics mode
    ***** calls out14
    ---- called by char14
char14 : writes a string of characters (up to 80) to the tk4014
    ***** calls out14
    ---- called by pchar
clr14: clears the tk4014 screen
***** calls out14
----- called by perase, escaped

cross14: reads the cross hair position in x,y coordinates on
the tk4014, plus returns the character used to
initiate the read
***** calls out14
----- called by pcread

graph14: puts the tk4014 into graphics mode
***** calls out14
----- called by char14, line14, place14, point14, pdrpab

init14: initializes the tk4014
***** calls out14
----- called by pinit

line14: draws a line between 2 given endpoints on tk4014
***** calls graph14, tran14
----- called by pdrlin, ppoly

move14: places tk4014 cursor at a specified location
***** calls graph14, tran14
----- called by pmovab, pchar, pplace

out14: the central routine that outputs graphics commands
----- called by alpha14, char14, clr14, cross14, graph14,
init14, size14, term14, tran14

size14: set character size on tk4014
***** calls out14

term14: terminates the tk4014
***** calls out14
----- called by pterm

tran14: translates given x,y coordinate values into x,y
high/low byte character values, then moves to the
specified location on the tk4014
***** calls out14
----- called by line14, move14

---

Hewlett Packard 7220A

---

charhp: writes a string of characters (up to 80) on the hp7220a
***** calls outhp
----- called by pchar

crosshp: returns the x,y coordinates of the hp7220a
***** calls outhp
----- called by pcread

inithp: initializes the hp7220a, pen up is the default
***** calls outhp

linehp: draws a line between 2 given endpoints on the hp7220a
***** calls outhp
----- called by pdrlin

movehp: moves the current hp7220a pen to the specified x,y
coordinate with the pen up
***** calls outhp
----- called by pmovab, pchar, pplace

offhp: takes the hp7220a offline
***** calls outhp
----- called by pinit, perase, escaped
onhp : places the hp7220a online
   ***** calls outhp
      ------ called by pinit, perase, escaped
outhp : central output routine for the hp7220a, allowing up to
       a maximum baud rate of 2400
      ------ called by charhp,crosshp,inithp,linehp,movehp,
           offhp,onhp,penhp,selectpenhp,sizehp,termhp
penhp : selects the specified hp7220a pen
   ***** calls outhp
selectpenhp : selects the specified hp7220a pen (1-4 possible)
     0 means return current pen to its bin holder
   ***** calls outhp
      ------ called by pinit, escaped
sizehp : sets the character size for hp7220a in centimeters
   ***** calls outhp
termhp : terminates the hp7220a, pen up is the default
   ***** calls selectpenhp, outhp
      ------ called by pterm
Appendix G

Device Driver Source Code

**********************************************************************************************
*                                                                                             *
* In the "#define" sections of code, the "/'" should be replaced by a backwards slash          *
*                                                                                             *
**********************************************************************************************

/***************************************************************************/
* date: 12 oct 83
* version: 1.0
* name: alpha14(pascal), alp14_(fortran 77)
* description: places the tk4014 into alphanumerics mode
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to put tk4014 into alpha mode
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out14
* calling modules: pchar
* author: Capt John W. Taylor

******************************************************************************/
#define US '/037'
alpha14()
{
  char ary[2]; /* control character array */
  ary[0] = US;
  out14(1,ary); /* US control character */
}

alp14_()
{
  alpha14();
}

/******************************************************************************/
* date: 12 oct 83
* version: 1.0
* name: char14(pascal), chr14_(fortran 77)
* description: writes a string of characters (up to 80) to the tk4014
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a

char14(num, outary)
int *num;
char outary[];
{
    int i; /* integer to pass to output routine */
    i = *num; /* output routine will not accept */
    out14(i, outary);
}

chr14_(num, outary)
int *num;
char outary[];
{
    char14(num, outary);
}

/*
 * date: 12 Oct 83
 * version: 1.0
 * name: clr14(pascal), clr14_(fortran77)
 * description: clears the tk4014 screen
 * operating system: VAX 11/780 UNIX
 * language: c
 * inputs: n/a
 * outputs: graphics command to clear the tk4014 screen
 * global variables used: n/a
 * global variables changed: n/a
 * global tables used: n/a
 * library routines: n/a
 * files read: n/a
 * files written: n/a
 * modules called: out14
 * calling modules: pchar, escaped
 * author: Capt John W. Taylor
 */
#define ESC '033'
#define FF '014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
clr14()
{
    char ary[2]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
ary[1] = FF; /* FF control character */
out14(2,ary);
tmpxyz[0] = ESC; /* clear array to start over */
tmpxyz[1] = ESC;
tmpxyz[2] = ESC;
}

clr14_()
{
clr14();
}

/*****************************************
* date: 12 oct 83
* version: 1.0
* name: cross14(pascal), crs14_(fortran 77)
* description: reads the cross hair position in x,y coordinates on
* the tk4014, plus returns the character used to
* initiate the read
* operating system: VAX 11/780 UNIX
* language: c
* inputs: x,y coordinates from the tk4014, character used to initiate *
* the read
* outputs: n/a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out14
* calling modules: pcread
* author: Capt John W. Taylor
*
/*****************************************
#define ESC '033'
define SUB '032'
cross14(x,y,icnt)
int *x, *y, *icnt;
{
    char ary[5];
    int i;

    /* turn on cross hair */
    ary[0] = ESC; /* ESC control character */
    ary[1] = SUB; /* SUB control character */
    out14(2,ary);

    /* get characters from screen */
i = read(1,ary,5);

    if (i == 1) /* carriage return used here */
    {
        read(1,ary,5); /* another read necessary since carriage
                        return used */

G-3
/* convert screen characters to x,y */
*x = ((ary[0] & 037)*32 + (ary[1] & 037));
*icnt = ' '; */
else /* any other key used here */
{
  /* convert screen characters to x,y */
  *icnt = ary[0];
}

if (i == 1) /* carriage return used here */
  read(1,ary,5);
else /* any other key used */
{
  read(1,ary,5);
  read(1,ary,5);
}
}
crs14(x,y,icnt)
int *x, *y, *icnt;
{
cross4(x,y,icnt);
}

#define GS '/035'
graph14()
{
  char ary[2]; /* array to hold characters */
  ary[0] = GS; /* GS control character */
  out14(1,ary);
}
gph14_()
{
        graph14();
}

init14();

#define ESC '/033'
define FF '/014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
init14()
{
        char ary[2]; /* array to hold characters */
 ary[0] = ESC; /* ESC control character */
 ary[1] = FF; /* FF control character */
 out14(2,ary);
 tmpxyz[0] = ESC; /* clear buffer */
 tmpxyz[1] = ESC;
 tmpxyz[2] = ESC;
}

int14_()
{
        init14();
}

lin14();

#define ESC '/033'
define FF '/014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
lin14()
{
        char ary[2]; /* array to hold characters */
 ary[0] = ESC; /* ESC control character */
 ary[1] = FF; /* FF control character */
 out14(2,ary);
 tmpxyz[0] = ESC; /* clear buffer */
 tmpxyz[1] = ESC;
 tmpxyz[2] = ESC;
}
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: tran14
* calling modules: pdrlin, ppoly
* author: Capt John W. Taylor

*******************************************************************************/
line14(x,y)
int *x, *y;
{
    tran14(x,y); /* translate and move */
}

lin14_(x,y)
int *x, *y;
{
    line14(x,y);
}
*******************************************************************************/

* date: 12 oct 83
* version: 1.0
* name: move14(pascal), mve14_(fortran 77)
* description: moves tk4014 cursor to a specified location
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to place cursor at specified location
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: graph14, tran14
* calling modules: pmovab, pchar, pplace
* author: Capt John W. Taylor

*******************************************************************************/
move14(x,y)
int *x, *y;
{
    graph14(); /* set tk4014 to move */
    tran14(x,y); /* translate and move */
}

mve14_(x,y)
int *x, *y;
{
    move14(x,y);
}
/*----------------------------------------------*/
/* * date: 12 oct 83 */
/* * version: 1.0 */
/* * name: out14 */
/* * description: the central routine that outputs graphics commands */
/* * to the tk4014 */
/* * operating system: VAX 11/780 UNIX */
/* * language: c */
/* * inputs: n/a */
/* * outputs: write command to output characters */
/* * global variables used: n/a */
/* * global variables changed: n/a */
/* * global tables used: n/a */
/* * library routines: n/a */
/* * files read: n/a */
/* * files written: n/a */
/* * modules called: n/a */
/* * calling modules: alpha14, char14, clr14, cross14, graph14, */
/* * init14, line14, move14, size14, term14, tran14 */
/* * author: Capt John W. Taylor */
/* */
/*----------------------------------------------*/
out14(num,outary)
int num; /* number of characters in array */
char outary[]; /* array of output characters */
{
    write(2,outary,num);
}

/***********************************************/
/* * date: 17 oct 83 */
/* * version: 1.0 */
/* * name: size14(pascal), siz14 (fortran) */
/* * description: set the character size(4 sizes possible) */
/* * 1 = 74 characters, 35 lines (default) */
/* * 2 = 81 characters, 38 lines */
/* * 3 = 121 characters, 58 lines */
/* * 4 = 133 characters, 64 lines */
/* * operating system: VAX 11/780 UNIX */
/* * language: c */
/* * inputs: n/a */
/* * outputs: n/a */
/* * global variables used: n/a */
/* * global variables changed: n/a */
/* * global tables used: n/a */
/* * library routines: n/a */
/* * files read: n/a */
/* * files written: n/a */
/* * modules called: out14 */
/* * calling modules: n/a */
/* * author: Capt John W. Taylor */
/* */
/************************************************/
#define ESC '/033'

6-7
size14(num)
int *num;
{
    char ary[2]; /* array to hold characters */
    ary[0] = ESC;
    if (*num == 4) /* 133 characters, 64 lines */
        ary[1] = ';';
    else if (*num == 3) /* 121 characters, 58 lines */
        ary[1] = ':';
    else if (*num == 2) /* 81 characters, 38 lines */
        ary[1] = '9';
    else /* 74 characters, 35 lines */
        ary[1] = '8'; /* default */
    out14(2, ary);
}

term14(pascal), trm14 (fortran 77)
* description: terminates the tk4014
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to terminate the tk4014
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out14
* calling modules: pterm
* author: Capt John W. Taylor
*
#define ESC '033'
#define FF '014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */

term14()
{
    char ary[2]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
    ary[1] = FF; /* FF control character */
    out14(2, ary);
    tmpxyz[0] = ESC; /* clear buffer */
    tmpxyz[1] = ESC;
    tmpxyz[2] = ESC;
tran14() {
    term14();
}

char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */

#define msmask 040
#define x1smask 0100
#define y1smask 0140

int value1, value2, value3, value4;
int ary[4]; /* array to hold characters for actual cursor move */

int flag; /* determine if lo y set, 0=not set, 1=set */
int i; /* counter */

int value1 = *x;
value4 = *y;
value2 = value1;
value1 = value1 & 037;
value2 = value2;
value2 = value2 >> 5;
value2 = value2 & 037;
value2 = value2 | msmask;
value1 = value1 | x1smask;
value3 = value4;
value3 = value3 & 037;
value4 = value4;
value4 = value4 >> 5;
value4 = value4 & 037;
value4 = value4 | msmask;
value3 = value3 + ylsmask;

i = -1; /* need to start at -1, so when incremented will start at 0 */
flag = 0;
if (value4 != tmpxyz[0]) /* value4 = hi y */
{
    i = i + 1;
    ary[i] = value4;
}
if (value3 != tmpxyz[1]) /* value3 = lo y */
{
    i = i + 1;
    ary[i] = value3;
    flag = 1; /* lo y set if flag = 1 */
}
if (value2 != tmpxyz[2]) /* value2 = hi x */
{
    if (flag == 0) /* lo y not already set */
    {
        i = i + 1; /* lo y not already set */
        ary[i] = value3;
    }
    i = i + 1;
    ary[i] = value2;
}
i = i + 1; /* lo x always set */
ary[i] = value1;
tmpxyz[0] = value4; /* keep current bytes for next comparison */
tmpxyz[1] = value3;
tmpxyz[2] = value2;
i = i + 1; /* need to add 1, since arrays and counters are off by 1 */
out14(i,ary);

 /***************************************************************************/
 * * date: 12 oct 83
 * * version: 1.0
 * * name: charhp(pascal), chrhp_(fortran 77)
 * * description: writes a string of characters (up to 80) on the
 * * hp7220a
 * * operating system: VAX 11/780 UNIX
 * * language: c
 * * inputs: n/a
 * * outputs: graphics command to write string
 * * global variables used: n/a
 * * global variables changed: n/a
 * * global tables used: n/a
 * * library routines: n/a
 * * files read: n/a
 * * files written: n/a
 * * modules called: outhp
 * * calling modules: pchar
 * * author: Capt John W. Taylor
 * */

G-10
#define ETX '/003'

ccharhp(num,outary)
int *num;
char outary[];
{
    int i;
    char ary[4]; /* array to hold characters */
    ary[0] = 'L';
    ary[1] = 'B';
    outhp(2,ary); /* set hp to alpha mode */
    i = *num;
    outhp(i,outary); /* write text string */
    ary[0] = ETX; /* ETX control character */
    outhp(1,ary); /* pen up */
}

chrhp_(num,outary)
int *num;
char outary[];
{
    charhp(num,outary);
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
* name: crosshp(pascal), crshp_(fortran 77)
* description: this procedure returns the x,y coordinates of the
* the hp7220a. a delay can be made to read the keyboard
* character(0=no delay, read x,y pen position immediately
* 1=delay, wait for keyboard character, then
* read x,y pen position
* operating system: VAX 11/780 UNIX
* language: c
* inputs: x,y coordinates from hp7220a
* outputs: n/a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp, offhp, onhp
* calling modules: pcread
* author: Capt John W. Taylor
*
******************************************************************************

#include <sys/ioctl.h>
#include <sgtty.h>
#include <stdio.h>
#include <signal.h>


crosshp(x,y,icnt,d)
int *x; /* x coordinate, range 0-16000 */
int *y; /* y coordinate, range 0-11400 */
int *icnt; /* keyboard character */
int *d; /* delay, 0=no delay, 1=delay */
{
    extern float bgrp_[1];
    struct sgttyb iostr;
    int n, oflags, (*istat)();
    char ary[3]; /* array of characters */
    char *buf, *calloc(), 1flag;

    if (*d == 1) /* delay to read keyboard character */
        { 
            offhp(); /* take hp7220a offline for read */
            read(1,icnt,1); /* delay and read keyboard character */
            onhp(); /* put hp7220a back online */
        }
    else *icnt = 0; /* set icnt to 0, no delay used */

    buf = calloc(100,sizeof(*buf));
    istat=signal(SIGINT,SIG_IGN);
    ioctl(0, TIOCGETP, &iostr);
    oflags = iostr.sg_flags;
    iostr.sg_flags &= ~ECHO;
    iostr.sg_flags |= CBREAK;
    ioctl(0, TIOCSETP, &iostr);
    ary[0] = 'O';
    ary[1] = 'A';
    ary[2] = 'I';
    outhp(3,ary); /* get x,y coordinates, and pen status */
    gets(buf); /* 0=pen up, 1=pen down */
    iostr.sg_flags = oflags;
    ioctl(0, TIOCSETP, &iostr);
    n = sscanf(buf, "%ld,%ld", x, y);
    free(buf);
    istat=signal(SIGINT,istat);
}

crshp_(x,y,icnt,d)
int *x,*y,*icnt,*d;
{
    crosshp(x,y,icnt,d);
}

G-12
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: pinit, perase
* author: Capt John W. Taylor
*
*******************************************************************************/
inithp()
{
    char ary[6]; /* array to hold characters */
    ary[0] = 'I'; /* initialize hp */
    ary[1] = 'N';
    ary[2] = ' ';
    ary[3] = 'P'; /* pen up */
    ary[4] = 'U';
    ary[5] = ' ';
    outhp(6, ary);
}

inthp_()
{
    inithp();
}

*******************************************************************************/
*  
*  *  date: 12 oct 83  
*  *  version: 1.0  
*  *  name: linhp(pascal), linhp_(fortran 77)  
*  *  description: draws a line from the current pen position to the  
*  *      specified x,y coordinate on the hewlett packard  
*  *  *  operating system: VAX 11/780 UNIX  
*  *  language: c  
*  *  inputs: n/a  
*  *  outputs: graphics command to draw a line on the hp7220a  
*  *  global variables used: n/a  
*  *  global variables changed: n/a  
*  *  global tables used: n/a  
*  *  library routines: n/a  
*  *  files read: n/a  
*  *  files written: n/a  
*  *  modules called: outhp  
*  *  calling modules: pdrlin  
*  author: Capt John W. Taylor
*
*******************************************************************************/
linhp(x, y),
int *x, *y;
{
    int temp; /* temporary coordinate */
    int i; /* counter */
    char ary[5]; /* character coordinate */
    ary[0] = 'P'; /* move to specified location */
    ary[1] = 'A';
    outhp(2, ary);
for(i=0; i<5; i++)
    ary[i] = ' '

temp = *x;
sprintf(ary, "%d", temp);
outhp(5, ary); /* *x coordinate */
ary[0] = 'P';
ary[1] = 'U';
ary[2] = 'A';
outhp(3, ary);

for(i=0; i<5; i++)
    ary[i] = ' '

temp = *y;
sprintf(ary, "%d", temp);
outhp(5, ary); /* *y coordinate */
ary[0] = 'P';
ary[1] = 'A';

linhp_(x,y)
int *x, *y;
{
    linehp(x,y);
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
* name: movehp(pascal), movehp_(fortran 77)
* description: moves the current hp7220a pen to the specified x,y coordinate with the pen up
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to move hp7220a pen
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: pmovab, pchar, pplace
* author: Capt John W. Taylor
******************************************************************************
movehp(x,y)
int *x, *y;
{
    int temp; /* temporary coordinate */
    int i;
    char ary[5]; /* array to hold coordinate */
    ary[0] = 'P'; /* pen up */
    ary[1] = 'U';
    ary[2] = 'A';
    outhp(3, ary);
    ary[0] = 'P'; /* move to specified location */
    ary[1] = 'A';
outhp(2,ary);
for(i=0; i<5; i++)
    ary[i] = ',';
temp = *x;
sprintf(ary,"%d",temp);
outhp(5,ary); /* *x coordinate */
ary[0] = ',';
outhp(1,ary);
for(i=0; i<5; i++)
    ary[i] = ',';
temp = *y;
sprintf(ary,"%d",temp);
outhp(5,ary); /* *y coordinate */
ary[0] = ' '; 
outhp(1,ary);
}

movehp_(x,y)
int *x, *y;
{
    movehp(x,y);
}

*****************************************************************************
* * date: 12 oct 83
* version: 1.0
* name: offhp(pascal), offhp_(fortran 77)
* description: takes the hp7220a offline
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to take hp7220a offline
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: pinit, perase, escapedd
* author: Capt John W. Taylor
*
*****************************************************************************
#define ESC '/033'
offhp()
{
    char ary[3]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
    ary[1] = '.';
    ary[2] = ' ');
    outhp(3,ary); /* take hp offline */
}

offhp_()
offhp();
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
* name: onhp(pascal), onhp_(fortran 77)
* description: places the hp7220a online
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to place hp7220a online
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: pinit, perase, escaped
* author: Capt John W. Taylor
******************************************************************************
#define ESC '033'
onhp() {
    char ary[3]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
    ary[1] = '.';
    ary[2] = '(';
    outhp(3,ary); /* put hp online */
}
onhp_() {
    onhp();
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
* name: outhp
* description: central output routine for the hp7220a, allowing
* up to a maximum baud rate of 2400
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: writes out commands from other routines
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: n/a

8-16
* calling modules: charhp, crosshp, inithp, linehp, offhp, onhp, movehp, *  
  penhp, selectpenhp, sizehp, termhp  
  * author: Capt John W. Taylor  
  *  
  #include <sgtty.h>  
  #include <sys/ioctl.h>  
  #include <stdio.h>  
  #include <signal.h>  

outhp(n, ichar)  
int n;    /* number of characters in array */  
int ichar[]; /* array of characters */  
{  
    static int buffree = 100; /* some initial space required to turn on */  
    struct sgttyb iostr;  
    char *buf, *calloc(j, inbuf[10]);  
    int i, j, (*istat)(), oflags;  
    char lflag;  
    j = n;  
    buf = calloc(j, sizeof(*buf));  
    for (i = 0; i < j; i++)  
        buf[i] = ichar[i];  
    for (i = 0; buffree < j + 10; i++)  
        {  
            /* wait for enough room in buffer */  
            buffree = 100; /* in case awakened by interrupt */  
            if (i > 0)  
                sleep(30); /* at least 30 seconds of data in buffer */  
            istat = signal(SIGINT, SIG_IGN); /* ignore interrupts */  
            ioctl(0, TIOCGETP, &iostr);  
            oflags = iostr.sg_flags;  
            iostr.sg_flags &= ~ECHO;  
            iostr.sg_flags |= CBREAK;  
            ioctl(0, TIOCSETP, &iostr);  
            printf("/033.B"); /* request buffer space */  
            gets(inbuf);  
            iostr.sg_flags = oflags;  
            ioctl(0, TIOCSETP, &iostr);  
            istat = signal(SIGINT, istat); /* allow interrupts */  
            sscanf(inbuf, "%d", &buffree);  
        }  
    buffree -= j;  
    write(2, ichar, n); /* write data to hp7220a */  
    free(buf);  
}  

/***/

*  
* date: 16 oct 83  
* version: 1.0  
* name: penhp(pascal), penhp_(fortran 77)  
* description: place the hewlett packard pen in the up or down  
* position (0=pen up, 1=pen down)  
* operating system: VAX 11/780 UNIX  
* language: c  
* inputs: n/a  

G-17
* outputs: n/a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: n/a
* author: Capt John W. Taylor

********************************************************************
penhp(num)
int *num; /* 0 = pen up, 1 = pen down */
{
    char ary[3]; /* array to hold characters */
    ary[0] = 'P';
    if (*num == 1) /* pen down */
        ary[1] = 'D';
    else /* pen up, default */
        ary[1] = 'U';
    ary[2] = ';';
    outhp(3,ary);
}

penhp_(num)
int *num;
{
    penhp(num);
}

******************************************************************************
*
* date: 12 oct 83
* version: 1.0
* name: selectpenhp(pascal), spnhp_(fortran 77)
* description: selects the specified hp7220a pen (1-4 possible)
* 0 means return current pen to its bin holder
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to select or return pen
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: outhp
* calling modules: pinit, escapedd, termhp
* author: Capt John W. Taylor
*
******************************************************************************
selectpenhp(num)
int *num;
{
char ary[4]; /* array to hold characters */
int i; /* temporary holder of *hp пен */
char temp[2]; /* character number of *hp пен */

ary[0] = 'S'; /* select 1-4 pens */
ary[1] = 'P';
i = *num;
sprintf(temp, "%d", i); /* convert pen to character */
ary[2] = temp[0];
ary[3] = 'I';
outhp(4, ary);

spnhp_(num)
int *num;
{
    selectpenhp(num);
}

sizehp (width, height)
double *width; /* width of character in centimeters */
double *height; /* height of character in centimeters */
{
    char ary[8]; /* array to hold numbers and characters */
    ary[0] = 'S';
    ary[1] = 'I';
    outhp(2, ary);
    sprintf(ary, "%f", *width);
    outhp(8, ary);
    ary[0] = ',';
    outhp(1, ary);
    sprintf(ary, "%f", *height);
    outhp(8, ary);
}

ary[0] = 'I';
OUTH(1,ary);

SIZHP(width,height)
double *width, *height;
{
    SIZHP(width,height);
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
* name: termhp(pascal), trmhp_(fortran 77)
* description: terminates the hp7220a, pen up is the default
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to terminate hp7220a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: selectpenhp, outhp
* calling modules: pterm
* author: Capt John W. Taylor
******************************************************************************

TERMH P(num)
int *num;
{
    char ary[6]; /* array to hold characters */
    ary[0] = 'I'; /* initialize hp */
    ary[1] = 'N';
    ary[2] = 'I';
    ary[3] = 'P'; /* pen up */
    ary[4] = 'U';
    ary[5] = 'I';
    OUTH(6,ary);
    *num = 0;
    SELECTPENHP(num);
}

TRMHP(num)
int * num;
{
    TERMH P(num);
}

******************************************************************************
* date: 12 oct 83
* version: 1.0
******************************************************************************

6-20
* name: alpha550(pascal), alp50_(fortran 77)
* description: places the visual 550 into alpha mode
* operating system: VAX 11/760 UNIX
* language: c
* inputs: n/a
* outputs: graphics command to put the visual 550 into alpha mode
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out550
* calling modules: n/a
* author: Capt John W. Taylor

#define US '/037'
alpha550()
{
    char ary[2]; /* control character array */
    ary[0] = US;
    out550(1,ary); /* US control character */
}

alp50_()
{
    alpha550();
}

char550(num, outary)
int *num;
char outary[];
{
int i; /* integer to pass to output routine */
i = *num; /* output routine will not accept */
out550(i,outary);
}

chr50_(num,outary)
int *num;
char outary[];
{
    char550(num,outary);  
}

/******************************************************************************
* * date: 12 oct 83
* * version: 1.0
* * name: clr550(pascal), clr50_(fortran 77)
* * description: clears the visual 550 screen
* * operating system: VAX 11/780 UNIX
* * language: c
* * inputs: n/a
* * outputs: graphics command to clear the tk4014 screen
* * global variables used: n/a
* * global variables changed: n/a
* * global tables used: n/a
* * library routines: n/a
* * files read: n/a
* * files written: n/a
* * modules called: out550
* * calling modules: n/a
* * author: Capt John W. Taylor
* */

#define ESC '/033'
#define FF '/014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
clr550()
{
    char ary[2]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
    ary[1] = FF; /* FF control character */
    out550(2,ary);
    tmpxyz[0] = ESC; /* clear buffer */
    tmpxyz[1] = ESC;
    tmpxyz[2] = ESC;
}

clr50_()
{
    clr550();
}

/******************************************************************************
* * date: 12 oct 83
* * version: 1.0
* * name: cross550(pascal), crs50_(fortran 77)
* */

6-22
* description: reads the cross hair position in x,y coordinates on
  the visual 550, plus returns the character used to
  initiate the read
* operating system: VAX 11/780 UNIX
* language: c
* inputs: x,y coordinates from the visual 550, character used to
  initiate the read
* outputs: n/a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out550
* calling modules: n/a
* author: Capt John W. Taylor

#define ESC '/033'
#define SUB '/032'

cross550(x,y,icnt)
int *x, *y, *icnt;
{
  char ary[5];
  int i;

  /* turn on cross hair */
  ary[0] = ESC; /* ESC control character */
  ary[1] = SUB; /* SUB control character */
  out550(2,ary);

  /* get characters from screen */
i = read(1,ary,5);

  if (i == 1) /* carriage return used here */
  {
    read(1,ary,5); /* another read necessary since carriage
                    return used */
    /* convert screen characters to x,y */
    *x = ((ary[0] & 037)*32 + (ary[1] & 037));
    *icnt = ' ';
  }
  else /* any other key used here */
  {
    /* convert screen characters to x,y */
    *icnt = ary[0];
  }

  if (i == 1) /* carriage return used here */
  read(1,ary,5);
  else /* any other key used */


```c
{
    read(1, ary, 5);
    read(1, ary, 5);
}

crs50_(x, y, icnt)
int *x, *y, *icnt;
{
    cross550(x, y, icnt);
}

}  // read(1,ary,5);

#define GS '/0351

graph550()
{
    char ary[2]; /* array to hold characters */
    ary[0] = GS; /* GS control character */
    out550(1, ary);
}

gph50_()
{
    graph550();
}

}  // read(1,ary,5);

#define GS '/0351

init550()
{
    in550_(for...
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */

int500();

}{
char ary[2]; /* array to hold characters */
ary[0] = ESC; /* ESC control character */
ary[1] = FF; /* FF control character */
out550(2, ary);
tmpxyz[0] = ESC; /* clear buffer */
tmpxyz[1] = ESC;
tmpxyz[2] = ESC;
}

int500();

}
lin50_(x,y)
int *x, *y;
{
  line550(x,y);
}

/***************************************************************************/

#define ESC '/033'

mode550(num)

int *num; /* mode of pixels */
{
  char ary[4]; /* array to hold output characters */
  ary[0] = ESC; /* ESC control character */
  ary[1] = '/';
  if (*num == 0)
    ary[2] = '0';
  else
    ary[2] = '1';
  ary[3] = 'd';
  out550(4,ary);
}

mde50_(num)
int *num;
move550(x,y)
int *x, *y;
{
    graph550(); /* set visual 550 to move */
    tran550(x,y); /* translate and move */
}

mve50_(x,y)
int *x, *y;
{
    move550(x,y);
}

move550(num);
}
* author: Capt John W. Taylor
*
out550(num, outary)
int num; /* number of characters in array */
char outary[]; /* array of output characters */
{
    write(2, outary, num);
}
**************************************************************************
* date: 12 oct 83
* version: 1.0
* name: size550 (pascal), siz50_ (fortran 77)
* description: set the character size on the visual 550
* 4 sizes possible:
* 1 = 80 characters by 34 lines (default)
* 2 = 40 characters by 17 lines
* 3 = 26 characters by 11 lines
* 4 = 20 characters by 8 lines
* operating system: VAX 11/780 UNIX
* language: c
* inputs: n/a
* outputs: n/a
* global variables used: n/a
* global variables changed: n/a
* global tables used: n/a
* library routines: n/a
* files read: n/a
* files written: n/a
* modules called: out550
* calling modules: n/a
* author: Capt John W. Taylor
*
**************************************************************************
#define ESC '/033'
size550(num)
int *num;
{
    char ary[2];
    ary[0] = ESC;
    if (*num == 4) /* 20 characters by 8 lines */
        ary[1] = '3';
    else if (*num == 3) /* 26 characters by 11 lines */
        ary[1] = '2';
    else if (*num == 2) /* 40 characters by 17 lines */
        ary[1] = '1';
    else /* 80 characters by 34 lines, default */
        ary[1] = '0';
    out550(2, ary);
}
```c
int *num;
{
    size50(num);
}

/*******************************************************
 * date: 12 oct 83
 * version: 1.0
 * name: term550(pascal), trm50_(fortran 77)
 * description: terminates the visual 550
 * operating system: VAX 11/780 UNIX
 * language: c
 * inputs: n/a
 * outputs: graphics command to terminate the visual 550
 * global variables used: n/a
 * global variables changed: n/a
 * global tables used: n/a
 * library routines: n/a
 * files read: n/a
 * files written: n/a
 * modules called: out550
 * calling modules: n/a
 * author: Capt John W. Taylor

#define ESC '/033'
#define FF '/014'
char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
term550()
{
    char ary[2]; /* array to hold characters */
    ary[0] = ESC; /* ESC control character */
    ary[1] = FF; /* FF control character */
    out550(2,ary);
    tmpxyz[0] = ESC; /* clear buffer */
    tmpxyz[1] = ESC;
    tmpxyz[2] = ESC;
}

trm50_()
{
    term550();
}

/*******************************************************
 * date: 12 oct 83
 * version: 1.0
 * name: tran550
 * description: translates given x,y coordinate values into x,y high/
 *              low byte character values, then moves to the
 *              specified location on the visual 550
 * operating system: VAX 11/780 UNIX
 * language: c
 * inputs: n/a
```
* outputs: graphics command to move visual 550 cursor *
* global variables used: temp *
* global variables changed: temp *
* global tables used: n/a *
* library routines: n/a *
* files read: n/a *
* files written: n/a *
* modules called: out550 *
* calling modules: line550, move550 *
* author: Capt John W. Taylor *

char tmpxyz[3]; /* global array to hold 3 bytes used for comparisons */
tran550(x, y) int *x, *y;

k int valuel, value2, value3, value4;
char ary[4]; /* array to hold characters for actual cursor move */
int flag; /* determine if 10 y set, 0=not set, 1=set */
i; /* counter */
#define msmask 040
#define xlsmask 0100
#define y1smask 0140
valuel = *x;
value4 = *y;
value2 = valuel;
value2 = value2 & 037;
value2 = value2 >> 5;
value2 = value2 & 037;
value2 = value2 & msmask;
valuel = valuel & xlsmask;
value3 = value4;
value3 = value3 & 037;
value4 = value4;
value4 = value4 >> 5;
value4 = value4 & 037;
value4 = value4 & msmask;
value3 = value3 & ylsmask;

i = -1; /* need to start at -1, so when incremented will start at 0 */
flag = 0;
if (value4 != tmpxyz[0]) /* value4 = hi y */
{
    i = i + 1;
    ary[i] = value4;
}
if (value3 != tmpxyz[1]) /* value3 = lo y */
{
    i = i + 1;
    ary[i] = value3;
    flag = 1; /* lo y set if flag = 1 */
}
if (value2 != tmpxyz[2]) /* value2 = hi x */
{

"G-30"
if (flag == 0) /* lo y not already set */
{
    i = i + 1;
    ary[i] = value3;
}
    i = i + 1;
    ary[i] = value2;
}
i = i + 1; /* lo x always set */
ary[i] = value1;
tmpxyz[0] = value4; /* keep current bytes for next comparison */
tmpxyz[1] = value3;
tmpxyz[2] = value2;
i = i + 1; /* need to add 1, since arrays and counters are off by 1 */
out550(i, ary);
Appendix H

VAX 11/780 and CDC 6600 HP7220A Calcomp Users Manual Supplement

Vax 11/780 Execution

Setup Visual 100 Terminal to run HP Plotter Calcomp Package

1. Set “blue box” to “61” - for 9600 baud
2. Flip the “blue box” switch on
3. Logon to Vax
4. Set Visual 100 terminal to 2400 baud
   a. Depress “setup” key
   b. Depress “5” key
   c. T.Speed set to “2400” by depressing “7” key
   d. R.Speed set to “2400” by depressing “8” key
   e. Depress “setup” key

Setup HP 7220A Plotter

1. Set HP plotter to “2400” baud by adjusting its switch in back of the plotter
2. Turn HP plotter on

Compile HP Plotter Package

>f77 "source file" hplot.o p10io.o <cr>

Run HP Plotter Package

>a.out

The hplot.f, hplot.o, p10io.c, and p10io.o source and object files are located on disk drive "U1" under the user id of:

jtaylor/hplotdir/hplot.f
hplot.o
p10io.c
p10io.o
Appendix I

Compatibility/Incompatibility Examples

FORTRAN To C To Pascal Example

FORTRAN MAIN --- > C SUBROUTINE OR --- > PASCAL PROCEDURE OR

<--- FUNCTION <--- FUNCTION

program fortran

integer intary(10)
integer intr
double precision realary(10)
double precision rel
character charary(10) or charary*10
character chr
logical bolary(10)
logical bol

call ctopascal(intary,intr,realary,rel,charary,chr,
bolary,bol)
end

---------
copascal.c - c routine

copascal_(intary,intr,realary,rel,charary,chr,bolary,bol)
int intary []; /* c arrays start at 0 not at 1 */
int *intr;
double realary []; /* c arrays start at 0 not at 1 */
double *rel;
char charary []; /* c arrays start at 0 not at 1 */
char *chr;
int bolary []; /* c arrays start at 0 not at 1 */
int *bol;
{
pascalsub(intary,intr,realary,rel,charary,chr,bolary,bol);

return(intary [], *intr, realary [], *rel, charary [],
       *chr, bolary [], *bol);
/* return only used if the c routine is
called as a function */
}

---------
pascalsub.p - pascal subroutine

type
  intnum = array [1..10] of integer;
  realnum = array [1..10] of real;
  charstr = array [1..10] of char;
  bolstr = array [1..10] of boolean;
procedure pascalsub(var intary : intnum;
  var intr : integer;
  var realary : realnum;
  var rel : real;
  var charary : charstr;
  var chr : char;
  var bolary : bolstr;
  var bol : boolean;
begin
  end;

---------
how to compile and what libraries to use

>cc -c copascal.c --compile the c routine
>pc -c pascalsub.p --compile the pascal routine
>f77 copascal.o pascalsub.o fortranmain.f -lpc -lm -lc
       --compile the main fortran program
>a.out --to execute the program
Pascal To C To FORTRAN Example

(**********************************************************************************************
 *                                                                                         *
 * date:                                                                                      *
 * version:                                                                                    *
 * name:                                                                                      *
 * description:                                                                                *
 * operating system: UNIX                                                                      *
 * language:                                                                                    *
 * inputs:                                                                                    *
 * outputs:                                                                                    *
 * global variables used:                                                                     *
 * global variables changed:                                                                  *
 * global tables used:                                                                        *
 * library routines:                                                                          *
 * files read:                                                                                *
 * files written:                                                                             *
 * modules called:                                                                            *
 * calling modules:                                                                           *
 * author: Capt John W. Taylor                                                                *
 *                                                                                         *
 **********************************************************************************************)

PASCAL MAIN ---> C SUBROUTINE OR (won't work) FORTRAN SUBROUTINE

--- FUNCTION -- OR FUNCTION

pascalmain.p - main pascal program

program pastest(input,output);
(* this program does not work on the present VAX UNIX *)
(* system due to FORTRAN I/O errors occurring when *)
(* a FORTRAN subroutine is called by a C subroutine. *)
(* a major system modification is needed in order for *)
(* this example program to work. *)
(* this program would pass data to a fortran routine by *)
(* interfacing through a c routine first, then *)
(* passing data back from the fortran routine to *)
(* the pascal program once again through the c routine. *)
(* no actual data is used in this demonstration program, *)
(* this program just shows how to set up each program and *)
(* routine to pass variables *)
type
  intnum = array [1..10] of integer;
  realnum = array [1..10] of real;
  charstr = array [1..10] of char;
  bolstr = array [1..10] of boolean;

var
  intary : intnum;
  intr : integer;
  realary : realnum;
  rel : real;
  charary : charstr;
  chr : char;
  bolary : bolstr;

I-3
bol : boolean;
#include 'ctofortran.h'
begin
  ctofortran(intary,intr,realary,rel,charary,chr,bolary,bol);
  (* call c interface to fortran *)
  (* at the end of the c routine, the data values
   will be returned and used if necessary *)
end.

ctofortran.c  - c routine  ( won't work, system I/O error )

ctofortran(intary,intr,realary,rel,charary,chr,bolary,bol)
int intary []; /* c arrays start at 0 not at 1 */
int *intr;
double realary []; /* c arrays start at 0 not at 1 */
double *rel;
char charary []; /* c arrays start at 0 not at 1 */
char *chr;
int bolary []; /* c arrays start at 0 not at 1 */
int *bol;
{
    callfor(); (does not work, system I/O error)
 ***********************
return(intary[],*intr,realary[],*rel,charary[],
       *chr,bolary[],*bol);
    /* return only used if the c routine is
       called as a function */
}

-----------

callfor.f  - (does not work, system I/O error)
  ***********************

subroutine callfor(intary,intr,realary,rel,
                   charary,chr,bolary,bol)
integer intary(10)
integer intr
real realary(10)
real rel
character charary(10) or charary*10
character chr
logical bolary(10)
logical bol
return
stop
end

-----------

ctofortran.h  - the .h include file

procedure ctofortran(var intary : intnum;
    var intr : integer;
    var realary : realnum;
\begin{verbatim}
var rel : real;
var charary : charstr;
var chr : char;
var bolary : bolstr;
var bol : boolean);

how to compile and what libraries to use
>cc -c ctofortran.c --compile the c routine
>f77 -c callfor.f --compile the fortran routine
>pc ctofortran.o callfor.o pascalmain.p -lF77 -lI77 -lc
--compile the main pascal program
>a.out --to execute the program
\end{verbatim}
**Pascal To Pascal Example**

(*------------------------------------------------------------------------*)
* *
* date: *
* version: *
* name: *
* description: *
* operating system: UNIX *
* language: *
* inputs: *
* outputs: *
* global variables used: *
* global variables changed: *
* global tables used: *
* library routines: *
* files read: *
* files written: *
* modules called: *
* calling modules: *
* author: Capt John W. Taylor *
* *
*------------------------------------------------------------------------*)

PASCAL MAIN -------> PASCAL SUBROUTINE OR FUNCTION

pascalmain.p - main pascal program

program pastest(input,output);
(* this pascal program shows how to interface to another *)
(* pascal routine *)
type
    ary = packed array [1..80] of char;
var
    x,y :, integer;
    num : real;
    outary : ary;
#include 'passub.h'
begin
    passub(x,y,num,outary); (* call a pascal routine *)
end.

passub.p - pascal routine
#include 'passub.h'
procedure passub;
begin
    x := 0;
    y := 0;
    num := 100.0;
    outary := 'Sample character string';
end;

passub.h - include .h file

---

I-6
procedure passub(var x, y : integer
                     var num : real;
                     var outary : ary); external;

how to compile pascal programs

>pc -c passub.p      --compile the pascal routine
>pc passub.o pascalmain.p      --compile the pascal program
>a.out      --how to execute the program
Appendix J

Duplicated Variables Created when Unique Underscore Identifier is Removed from the University of Pennsylvania Core Variables

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

University of Pennsylvania variables with underscores to create a unique variable name

These are global variables used by all pascal procedures

1. defconst.h file with "const" pascal variables
   none

2. deftype.h file with "type" pascal variables

   SEG_PTR
   PRIM_PTR
   OWNER_SEG
   NEXT_PRIM
   PICK_ID
   LINE_INDEX
   LINE_WIDTH
   LINE_STYLE
   TEXT_INDEX
   POLY_INTERIOR_STYLE
   POLY_EDGE_STYLE
   FILL_INDEX
   VERTEX_INDICES
   SEG_TYPE
   SEG_NAME
   IMAGE_TRANS
   TRANS_LIMIT
   PRIM_LIST
   NEXT_SEG
   ESC_PTR
   ESC_REC
   N_INTEGER
   N_REAL
   N_CHAR
   A_INTEGER
   A_REAL
   A_CHAR
   PRIM_REC
   SEG_REC
   VIEW_STATE_REC
   WORLD_MAT1
   WORLD_MAT2
3. extvar.h file with "var" pascal variables

PRIM_ATTR
SEG_ATTR
LAST_SEGMENT
LEFT_MARGIN
VIEW_STATE
ENV_SAVE_PTR
COLOR_TABLE
DD_REC
OLD_TRANSFORM
OLD_COLOR
DD_START

Converted pascal variables with the underscore identifier removed that may conflict with other variables already created without an underscore identifier are listed below.

Possible conflicts arise when local variable names are created for use within a specific procedure where the global variable is also used, but because the underscore has been removed from the global variables, a conflict occurs.

The underscore has to be removed because the UNIX Pascal will not accept underscore identifiers.

1. PICK_ID = pickid : conflicts, but corrected by renaming the local variable to "pickidl" in the following procedures

   pickxy
   awaitpick

2. SEG_NAME = segname : conflicts, but corrected by renaming the local variable to "segnamel" in the following procedures

   crrseg
derseg
renseg
printseg
ssvis
sshilt
ssdet
ssitn2
ssitr2
ssitr3
isvis
ishilt
isdet
isitn2
isitr2
isitr3
isttyp
ioseg
pickxy
awaitpick
batch1
Appendix K

# Table of Contents

## I. Tektronics 4014 User Subroutines

(Pascal) / (Fortran 77)

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha14</td>
<td>K-4</td>
</tr>
<tr>
<td>char14</td>
<td>-4</td>
</tr>
<tr>
<td>clr14</td>
<td>-4</td>
</tr>
<tr>
<td>cross14</td>
<td>-5</td>
</tr>
<tr>
<td>graph14</td>
<td>-5</td>
</tr>
<tr>
<td>init14</td>
<td>-5</td>
</tr>
<tr>
<td>line14</td>
<td>-6</td>
</tr>
<tr>
<td>move14</td>
<td>-6</td>
</tr>
<tr>
<td>size14</td>
<td>-7</td>
</tr>
<tr>
<td>term14</td>
<td>-7</td>
</tr>
</tbody>
</table>

## II. Hewlett Packard 7220A Plotter User Subroutines

(Pascal) / (Fortran 77)

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>charhp</td>
<td>K-8</td>
</tr>
<tr>
<td>crosshp</td>
<td>-8</td>
</tr>
<tr>
<td>inithp</td>
<td>-9</td>
</tr>
<tr>
<td>linehp</td>
<td>-9</td>
</tr>
<tr>
<td>movehp</td>
<td>-9</td>
</tr>
<tr>
<td>offhp</td>
<td>-10</td>
</tr>
<tr>
<td>onhp</td>
<td>-10</td>
</tr>
<tr>
<td>penhp</td>
<td>-10</td>
</tr>
<tr>
<td>selectpenhp</td>
<td>-11</td>
</tr>
<tr>
<td>sizehp</td>
<td>-11</td>
</tr>
<tr>
<td>termhp</td>
<td>-11</td>
</tr>
</tbody>
</table>

## III. Visual 550 User Subroutines

(Pascal) / (Fortran 77)

<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpha550</td>
<td>K-13</td>
</tr>
<tr>
<td>char550</td>
<td>-13</td>
</tr>
<tr>
<td>clr550</td>
<td>-13</td>
</tr>
<tr>
<td>cross550</td>
<td>-14</td>
</tr>
<tr>
<td>graph550</td>
<td>-14</td>
</tr>
<tr>
<td>init550</td>
<td>-14</td>
</tr>
<tr>
<td>line550</td>
<td>-15</td>
</tr>
<tr>
<td>mode550</td>
<td>-15</td>
</tr>
<tr>
<td>move550</td>
<td>-16</td>
</tr>
<tr>
<td>size550</td>
<td>-16</td>
</tr>
<tr>
<td>term550</td>
<td>-17</td>
</tr>
</tbody>
</table>
IV. Graphic Terminal/Plotter Usage
1. Tektronics 4014 Vector Terminal ........................................ K-18
2. Hewlett Packard 7220A 4 Color Plotter .......................... -18
3. Visual 550 Raster Terminal ................................................ -18

V. Sample Pascal and Fortran 77 Programs
1. TK4014 in Pascal ................................................................. K-20
2. TK4014 in Fortran 77 ......................................................... -21
3. HP7220A in Pascal ............................................................... -22
4. HP7220A in Fortran 77 ......................................................... -23
5. Visual 550 in Pascal .............................................................. -24
6. Visual 550 in Fortran 77 ....................................................... -25

VI. VAX 11/780 Execution
1. Compile and Run Pascal Programs ......................................... K-27
2. Compile and Run Fortran 77 Programs .................................... -27
1. **alpha14 / alp14**

   **Purpose:** Enter the Tektronics 4014 into alphanumerics mode

   **Calling Sequence:**
   - Pascal: `alpha14;`
   - Fortran: `call alp14`

   **Programming Considerations:** primarily used to set device into alpha mode for writing text

2. **char14 / chr14**

   **Purpose:** Write a string of text to the Tektronics 4014

   **Calling Sequence:**
   - Pascal: `char14(num,outary);`
     - `num`: integer; - number of characters to print
     - `ary = packed array [1..??] of char;` - maximum number in array
     - `outary: ary;` - array of characters to output
   - Fortran 77: `call chr14(num,outary)`
     - `integer num - number of characters to print`
     - `character outary(??) - array of characters to output`
     - `?? - maximum number in array`

   **Programming Considerations:** must be in alpha mode when using these subroutines. Characters will be lost if they are drawn outside the limits of the screen.

3. **clr14 / clr14**

   **Purpose:** Clears the Tektronics 4014 screen, and places it into alpha mode

   **Calling Sequence:**
   - Pascal: `clr14;`
   - Fortran: `call clr14`

   **Programming Considerations:** Any text or graphics on screen will be erased. The Tektronics 4014 will now be in the
alpha mode.  clr14, init14, and term14 do the same thing.

4. cross14 / crs14

Purpose: Reads the cross hair position in x,y coordinates on the Tektronics 4014, plus it returns the character from the keyboard used to initiate the cross hair read.

Calling Sequence:

Pascal: cross14(x,y,icnt);

Fortran: call crs14(x,y,icnt)

Programming Considerations: Must be in graphics mode when using these subroutines. To use, the cross hairs will appear on the screen, move them with the 2 thumbwheels to the desired position, then depress any keyboard character to initiate the read. No delay need be added, the system will automatically wait for a keyboard character before continuing.

5. graph14 / gph14

Purpose: Enter the Tektronics 4014 into graphics mode

Calling Sequence:

Pascal: graph14;

Fortran: call gph14

Programming Considerations: primarily used to set device into graphics mode for reading the cross hairs, drawing lines, and setting the character size.

6. init14 / int14

Purpose: Initializes the Tektronics 4014 screen, and places it into alpha mode

Calling Sequence:
Pascal: init14;

Fortran: call init14

Programming Considerations: Any text or graphics on screen will be erased. The Tektronics 4014 will now be in the alpha mode. clr14, init14, term14 do the same thing.

7. line14 / lin14

Purpose: Draw a line from the current cursor position on the Tektronics 4014 to the specified x,y coordinate

Calling Sequence:

Pascal: line14(x,y);

x : integer; - x coordinate, range 0-1023
y : integer; - y coordinate, range 0-778

Fortran: call lin14(x,y)

integer x - x coordinate, range 0-1023
integer y - y coordinate, range 0-778

Programming Considerations: Must be in graphics mode when using these subroutines. The very first line14 or lin14 is normally preceded by move14 or mve14 to fix a starting point. This move sets up the terminal for one line, or a series of connected lines, to be drawn. It is not entirely necessary to precede the very first line with a move. However, when in graphics mode and not preceding the very first line with a move, then the very first line may or may not be drawn depending on the state of the terminal. Lines will be clipped or lost if they are outside the screen limits.

8. move14 / mve14

Purpose: Move the current Tektronics 4014 cursor to the specified x,y coordinates

Calling Sequence:

Pascal: move14(x,y);

x : integer; - x coordinate, range 0-1023
y : integer; - y coordinate, range 0-778

Fortran: call mve14(x,y)

integer x - x coordinate, range 0-1023
integer y - y coordinate, range 0-778
Programming Considerations: Do not have to be in graphics mode when accessing this routine. It is automatically placed into graphics mode. Normally precedes the very first line14 or lin14. Moves can occur outside the screen limits not aborting the program, but affecting characters and lines drawn.

9. size14 / siz14

Purpose: Set the character size for the Tektronics 4014

4 sizes possible:
1 = 74 characters, 35 lines (default)
2 = 81 characters, 38 lines
3 = 121 characters, 58 lines
4 = 133 characters, 64 lines

Calling Sequence:

Pascal: size14(num);
num : integer; - number (1-4) to set character size

Fortran: call siz14(num)
integer num - number (1-4) to set character size

Programming Considerations: This routine does not have to be called unless the character size needs to be set to other than the default value. When it is called, the Tektronics 4014 must be in graphics mode.

10. term14 / trm14

Purpose: Terminate the Tektronics 4014, and places it into alpha mode

Calling Sequence:

Pascal: term14;

Fortran: call trm14

Programming Considerations: Any text or graphics screen will be erased. The Tektronics 4014 will now be in alpha mode. clr14, init14, and term14 do the same thing.
II. **Hewlett Packard 7220A Plotter User Subroutines**

1. **charhp / chrhp**

   **Purpose:** Writes a string of characters to the Hewlett Packard plotter

   **Calling Sequence:**

   Pascal: `charhp(num, outary);`
   
   num: integer; - number of characters to output
   ary = packed array [1..??] of char;  
   ?? - maximum number in array
   outary : ary; - array of characters to output

   Fortran: `call chrhp(num, outary)`
   
   integer num - number of characters to output
   character outary(??)  
   ?? - maximum number in array

   **Programming Considerations:** An "out of range" error will occur and cause the plotter to abort if the x,y coordinates of the plotter are exceeded (x = 0-16000, y = 0-11400). The plotter must be online when using these subroutines.

2. **crosshp / crshp**

   **Purpose:** Reads the x,y coordinates of the current pen location on the Hewlett Packard plotter. A delay can be made to read the keyboard character.

   **Calling Sequence:**

   Pascal: `crosshp(x, y, icnt, d);`
   
   x : integer; - x coordinate, range 0-16000
   y : integer; - y coordinate, range 0-11400
   icnt : integer; - keyboard character
   d : integer; - delay 0=no delay, 1=delay

   Fortran: `call crshp(x, y, icnt, d)`
   
   integer x - x coordinate, range 0-16000
   integer y - y coordinate, range 0-11400
   integer icnt - keyboard character
   integer d - delay 0=no delay, 1=delay

   **Programming Considerations:** The plotter must be online when using these subroutines. To move the pen, use the 5
directional arrows on the plotter. They will move the pen in either the online or offline mode.

3. `inithp / inthp`

Purpose: Initializes the Hewlett Packard plotter, pen up is the default

Calling Sequence:

Pascal: `inithp;`

Fortran: `call inthp`

Programming Considerations: These subroutines move the pen to the lower right corner of the plotter, in the pen up position. The current pen selection is not affected. The plotter must be online when using these subroutines.

4. `linehp / linhp`

Purpose: Draws a line from the current pen location to the specified x,y coordinates on the plotter

Calling Sequence:

Pascal: `linehp(x,y);`

Fortran: `call linhp(x,y)`

Programming Considerations: The plotter must be online to use these subroutines. To draw a line the pen must be in the "down" position prior to calling these subroutines. An "out of range" error will occur if a line is drawn outside the x,y coordinate limits, causing the plotter to abort.

5. `movehp / mvehp`

Purpose: Moves the current Hewlett Packard plotter pen location to the specified x,y coordinates

Calling Sequence:

Pascal: `movehp(x,y);`

Fortran: `call mvehp(x,y)`
Fortran: call mvehp(x,y)

integer x - x coordinate, range 0-16000
integer y - y coordinate, range 0-11400

Programming Considerations: The plotter must be online when using these subroutines. The pen is automatically placed in the "up" position by this routine. An "out of range" error will occur if the pen is moved outside the x,y coordinate limits, causing the plotter to abort.

6. offhp / offhp

Purpose: Takes the Hewlett Packard plotter offline.

Calling Sequence:

Pascal: offhp;
Fortran: call offhp

Programming Considerations: To gain control of the Visual 100 that is connected to the plotter, the plotter must be offline.

7. onhp / onhp

Purpose: Puts the Hewlett Packard plotter online.

Calling Sequence:

Pascal: onhp;
Fortran: call onhp

Programming Considerations: The plotter must be online when commands are sent to the plotter.

8. penhp / penhp

Purpose: Place the Hewlett Packard plotter pen in the up or down position.

Calling Sequence:

Pascal: penhp(num);

num : integer; 0 = pen up
1 = pen down

Fortran: call penhp(num)

integer num - 0 = pen up
1 = pen down
Programming Considerations: The plotter must be online when using these subroutines.

9. selectpenhp / spnhp

Purpose: Selects the specified Hewlett Packard plotter pen, or returns current pen to its bin holder.

Calling Sequence:

Pascal: selectpenhp(num);

num : integer; - 0 = return pen to its bin
1 = pen 1
2 = pen 2
3 = pen 3
4 = pen 4

Fortran: call spnhp(num)

integer num - 0 = return pen to its bin
1 = pen 1
2 = pen 2
3 = pen 3
4 = pen 4

Programming Considerations: The plotter must be online when using these subroutines.

10. sizehp / sizhp

Purpose: Set the character size on the Hewlett Packard plotter in centimeters (range -128.000 to 127.000)

Calling Sequence:

Pascal: sizehp(width,height);

width : real; - width of character
default = 0.19
height : real; - height of character
default = 0.27

Fortran: call sizhp(width,height)

double precision width - width of character
default = 0.19
double precision height - height of character
default = 0.27

Programming Considerations: The plotter must be online when using these subroutines.

11. termhp / trmhp
Purpose: Terminates the Hewlett Packard plotter, pen up is the default, current pen is returned to its bin holder

Calling Sequence:

Pascal: termhp(num);

num : integer; - num = 0 to return pen to its bin (variable required)

Fortran: call tramhp(0)

Programming Considerations: The plotter must be online when using these subroutines. The pen is moved to the lower right corner of the plotter.
III. Visual 550 User Subroutines

1. alpha550 / alp50

Purpose: Enter the Visual 550 into alphanumerics mode

Calling Sequence:

Pascal: alpha550;

Fortran: call alp50

Programming Considerations: primarily used to set device into alpha mode for writing text

2. char550 / chr50

Purpose: Write a string of text to the Visual 550

Calling Sequence:

Pascal: char550(num, outary);

num : integer; - number of characters to print

ary = packed array [1..??] of char;

?? - maximum number in array

outary : ary; - array of characters to output

Fortran 77: call chr50(num, outary)

integer num - number of characters to print
character outary(??) - array of characters to output

?? - maximum number in array

Programming Considerations: must be in alpha mode when using these subroutines. Characters will be lost if they are drawn outside the limits of the screen.

3. clr550 / clr50

Purpose: Clears the Visual 550 screen, and places it into alpha mode

Calling Sequence:

Pascal: clr550;

Fortran: call clr50

Programming Considerations: Any text or graphics on screen will be erased. The Visual 550 will now be in the

K-13
alpha mode. clr550, init550, and term550 do the same thing.

4. cross550 / crs50

Purpose: Reads the cross hair position in x,y coordinates on the Visual 550, plus it returns the character from the keyboard used to initiate the cross hair read

Calling Sequence:

Pascal: cross550(x,y,icnt);

\[
\begin{align*}
\text{x : integer} & \quad \text{- x coordinate, range 0-1023} \\
\text{y : integer} & \quad \text{- y coordinate, range 0-778} \\
\text{icnt : integer} & \quad \text{- keyboard character returned in integer format}
\end{align*}
\]

Fortran: call crs50(x,y,icnt)

\[
\begin{align*}
\text{integer x} & \quad \text{- x coordinate, range 0-1023} \\
\text{integer y} & \quad \text{- y coordinate, range 0-778} \\
\text{integer icnt} & \quad \text{- keyboard character returned in integer format}
\end{align*}
\]

Programming Considerations: Must be in graphics mode when using these subroutines. To use, the cross hairs will appear on the screen, move them with the 8 arrow keys to the desired position, then depress any keyboard character to initiate the read. No delay need be added, the system will automatically wait for a keyboard character before continuing.

5. graph550 / gph50

Purpose: Enter the Visual 550 into graphics mode

Calling Sequence:

Pascal: graph550;

Fortran: call gph50

Programming Considerations: primarily used to set device into graphics mode for reading the cross hairs, drawing lines, and setting the character size

6. init550 / int50

Purpose: Initializes the Visual 550 screen, and places it into alpha mode

Calling Sequence:
Pascal: init550;
Fortran: call int50

Programming Considerations: Any text or graphics on screen will be erased. The Visual 550 will now be in the alpha mode. clr550, init550, term550 do the same thing.

7. line550 / lin50

Purpose: Draw a line from the current cursor position on the Visual 550 to the specified x,y coordinate

Calling Sequence:

Pascal: line550(x,y);

\[ x : \text{integer}; \quad \text{x coordinate, range 0-1023} \]
\[ y : \text{integer}; \quad \text{y coordinate, range 0-778} \]

Fortran: call lin50(x,y)

\[ \text{integer } x \quad \text{x coordinate, range 0-1023} \]
\[ \text{integer } y \quad \text{y coordinate, range 0-778} \]

Programming Considerations: Must be in graphics mode when using these subroutines. The very first line550 or lin50 is normally preceded by move550 or mve50 to fix a starting point. This move sets up the terminal for one line, or a series of connected lines, to be drawn. It is not entirely necessary to precede the very first line with a move. However, when in graphics mode and not preceding the very first line with a move, then the very first line may or may not be drawn depending on the state of the terminal. Lines will be clipped or lost if they are outside the screen limits.

8. mode550 / mde50

Purpose: Turns the pixels on or off for the Visual 550. With the pixels off, lines can be selectively deleted.

Calling Sequence:

Pascal: mode550(num);

\[ num : \text{integer}; \quad 0 = \text{pixels on (default)} \]
\[ 1 = \text{pixels off} \]

Fortran: call mde50(num)

\[ \text{integer } num \quad 0 = \text{pixels on (default)} \]
1 = pixels off

Programming Considerations: Must be in graphics mode when using these subroutines.

9. move550 / mve50

Purpose: Move the current Visual 550 cursor to the specified x,y coordinates

Calling Sequence:

Pascal: move550(x,y);

\[
\begin{align*}
\text{x} & : \text{integer} \ - \ \text{x coordinate, range 0-1023} \\
\text{y} & : \text{integer} \ - \ \text{y coordinate, range 0-778}
\end{align*}
\]

Fortran: call mve50(x,y)

\[
\begin{align*}
\text{integer x} & - \ \text{x coordinate, range 0-1023} \\
\text{integer y} & - \ \text{y coordinate, range 0-778}
\end{align*}
\]

Programming Considerations: Do not have to be in graphics mode when accessing this routine. It is automatically placed into graphics mode. Normally precedes the very first line550 or lin50. Moves can occur outside the screen limits not aborting the program, but affecting characters and lines drawn.

10. size550 / siz50

Purpose: Set the character size for the Visual 550

4 sizes possible:
1 = 80 characters, 34 lines (default)
2 = 40 characters, 17 lines
3 = 26 characters, 11 lines
4 = 20 characters, 8 lines

Calling Sequence:

Pascal: size550(num);

\[
\text{num} : \text{integer} \ - \ \text{number (1-4) to set character size}
\]

Fortran: call siz50(num)

\[
\text{integer num} - \ \text{number (1-4) to set character size}
\]

Programming Considerations: This routine does not have to be called unless the character size needs to be set to other than the default value. When it is called, the Visual 550 must be in graphics mode.
11. **term550 / trm50**

**Purpose:** Terminate the Visual 550, and places it into alpha mode

**Calling Sequence:**

Pascal: `term550`;

Fortran: `call trm50`

**Programming Considerations:** Any text or graphics screen will be erased. The Visual 550 will now be in alpha mode. `clr550`, `init550`, and `term550` do the same thing.
IV. Graphic Terminal/Plotter Usage

1. Tektronics 4014 Vector Terminal

***** Not yet available for use on the Vax *****

2. Hewlett Packard 7220A 4 Color Plotter

Setup of the Visual 100 connected to the plotter

a. Set "blue box" to "61" - for 9600 baud

b. Logon to the Vax - ****************************
    ****** warning ******
    ****************************

    Make sure that the Hewlett Packard plotter is
turned off when trying to logon to the Visual
100 terminal.

c. Set the Visual 100 terminal to 2400 baud

   (1) type "stty 2400" <cr> to start 2400 baud setup
   (2) Depress the "setup" key
   (3) Depress the "5" key
   (4) Transmit speed set to 2400 baud by depressing
       the "7" key until the "T.Speed" on the lower
       right screen is 2400
   (5) Receive speed set to 2400 baud by depressing
       the "B" key until the "R.Speed" on the lower
       right screen is 2400
   (6) Depress the "setup" key again
   (7) Now at 2400 baud

Setup of the Hewlett Packard Plotter

a. Set the Hewlett Packard plotter to "2400" baud, if
   necessary, by adjusting the knob to 2400 in the back
   of the plotter.

b. Turn the Hewlett Packard plotter on.

3. Visual 550 Raster Terminal

To run a graphics program on the Visual 550:

   (1) depress the "setup" key
   (2) depress the "F6" key until the F6 at the bottom of
       the screen reads "GRAPHICS"
(3) depress the "setup" key
(4) the terminal is ready to run a graphics program
V. Sample Pascal and Fortran 77 Programs

1. TK4014 in Pascal

-------------

test14.p file

program test14(input, output);
type
  ary = packed array [1..80] of char;
var
  x, y: integer; (* x, y coordinates *)
  num: integer; (* number to pass to subroutines *)
  icnt: integer; (* keyboard character returned *)
  outary: ary; (* character string *)
#include 'test14.h' (* include file needed to define *)
  (* external procedures and variables *)
begin
  (* initialize the screen *)
  init14;
  (* read the cross hair *)
  graph14;
  cross14(x, y, icnt);
  (* write out a sample text string *)
  clr14;
  outary = 'Sample character string';
  num := 23;
  alpha14;
  char14(num, outary);
  (* write out another, smaller text string *)
  clr14;
  graph14;
  num := 4;
  size14(num);
  alpha14;
  num := 23;
  char14(num, outary);
  (* draw a line *)
  clr14;
  graph14;
  x := 100;
  y := 200;
  move14(x, y);
  x := 600;
  y := 700;
  line14(x, y);
  (* terminate program *)
  term14;
end.

-------------

test14.h file
procedure alpha14; external;
procedure char14(var num : integer;
  var outary : ary); external;
procedure clr14; external;
procedure cross14(var x,y,icnt : integer); external;
procedure graph14; external;
procedure init14; external;
procedure line14(var x,y : integer); external;
procedure move14(var x,y : integer); external;
procedure size14(var num : integer); external;
procedure term14; external;

2. TK4014 in Fortran 77

---------------
test14.f file

program test14
  integer x,y,icnt
  integer num
  character outary*80
  c initialize screen
  call int14
  c read the cross hair position
  call gph14
  call crsl4(x,y,icnt)
  c write out sample text string
  call clr14
  num = 23
  call alp14
  outary = 'Sample character string'
  call chr14(num,outary)
  c write out another, smaller text string
  call clr14
  call gph14
  num = 4
  call siz14(num)
  call alp14
  num = 23
  call chr14(num,outary)
  c draw a line
  call clr14
  call gph14
  x = 100
  y = 200
  call mvel4(x,y)
  x = 600
  y = 700
  call lin14(x,y)
  c terminate program
  call trm14
end

---------------
3. HP7220m in Pascal

--------------
testhp.p file

program test(input, output);
type
 ary = packed array [1..80] of char;
var
 x, y : integer; (* x, y coordinates *)
 icnt : integer; (* keyboard character *)
 d : integer; (* delay *)
 num : integer; (* number to pass to subroutines *)
 outary : ary; (* character string *)
 width, height : real (* character size in centimeters *)
#include 'testhp.h' (* include file needed to define *)
 (* external procedures and variables *)
begin
 (* initialize plotter *)
onhp;
inithp;
 (* select a pen *)
 num := 1;
 selectpenhp(num);
 (* read the pen position *)
 d := 0;
crosshp(x, y, icnt, d);
 (* write a sample text string *)
 x := 100;
y := 2000;
movelp(x, y);
 outary := 'Sample character string';
 num := 23;
 charhp(num, outary);
 (* write another, larger sample text string *)
 x := 100;
y := 4000;
movelp(x, y);
 width := 1.0;
 height := 1.0;
 sizehp(width, height);
 charhp(num, outary);
 (* draw a line *)
 x := 1000;
y := 1000;
movelp(x, y);
 num := 1;
 penhp(num);
 x := 9000;
y := 9000;
 linehp(x, y);
 num := 0;
 penhp(num);
 (* terminate plotter *)
 num := 0;
termhp(num);
offhp;
end.

-------------
testhp.h

procedure charhp(var num : integer); external;
var outary : array*80;
procedure crosshp(var x,y,icnt,d : integer); external;
procedure inithp; external;
procedure linehp(var x,y : integer); external;
procedure movehp(var x,y : integer); external;
procedure offhp; external;
procedure onhp; external;
procedure penhp(var num : integer); external;
procedure selectpenhp(var num : integer); external;
procedure sizehp(var width,height : real); external;
procedure termhp(var num : integer); external;

4. HP7220A in Fortran 77

program testhp
integer x,y
integer icnt,d
integer num
character outary*80
double precision width,height

 c initialize plotter
 call onhp
call inthp
c select a pen
 num = 1
call spnhp(num)
c read pen position
 d = 0
call crshp(x,y,icnt,d)
c write a sample text string
 x = 100
 y = 2000
call movehp(x,y)
outary = 'Sample character string'
 num = 23
call charhp(num,outary)
c write a larger sample text string
 x = 100
 y = 4000
call movehp(x,y)
width = 1.0
height = 1.0
call sizehp(width,height)
call charhp(num,outary)
c draw a line
 x = 1000
5. Visual 550 in Pascal

**test550.p file**

```pascal
program test550(input, output);

type
  ary = packed array [1..80] of char;

var
  x, y : integer; (* x, y coordinates *)
  num : integer; (* number to pass to subroutines *)
  icnt : integer; (* keyboard character returned *)
  outary : ary; (* character string *)

#include 'test550.h' (* include file needed to define *)
(* external procedures and variables *)

begin
  (* initialize the screen *)
  init550;
  (* read the cross hair *)
  graph550;
  cross550(x, y, icnt);
  (* write out a sample text string *)
  clr550;
  outary := 'Sample character string';
  num := 23;
  alpha550;
  char550(num, outary);
  (* write out another, larger text string *)
  clr550;
  graph550;
  num := 2;
  size550(num);
  alpha550;
  num := 23;
  char550(num, outary);
  (* draw a line *)
  clr550;
  graph550;
  x := 100;
  y := 200;
end.
```

K-24
move550(x,y);
  x := 600;
  y := 700;
line550(x,y);

(* erase the same line *)
  num := 1;
  mode550(num);
  x := 100;
  y := 200;
move550(x,y);
  x := 600;
  y := 700;
line550(x,y);

(* terminate program *)
term550;
end.

-------------------------

6. Visual 550 in Fortran

-------------------------

test50.f file

program test50
  integer x, y, icnt
  integer num
  character outary*80
  c initialize screen
  call int50
  c read the cross hair position
  call gph50;
  call crs50(x, y, icnt)
  c write out sample text string
  call clr50
  num = 23
  call alp50
  outary = 'Sample character string'
  call chr50(num, outary)
  c write out another, larger text string

K-25
call clr50
call gph50
num = 2
call siz50(num)
call alp50
num = 23
call chr50(num, outary)
c draw a line
call clr50
call gph50
x = 100
y = 200
call mve50(x, y)
x = 600
y = 700
call lin50(x, y)
c erase the same line
num = 1
call mve50(num)
x = 100
y = 200
call mve50(x, y)
x = 600
y = 700
call lin50(x, y)
c terminate program
call trm50
end
VI. VAX 11/780 Execution

1. Compile and Run Pascal Programs

Compile: >pc file.p /u1/gcs-83d/jtaylor/lib2

***** warning *****
a message "warning: table of contents for archive is
out of date, rerun ranlib(1)" will appear, ignore it

lib2 - library containing pascal graphics
subroutines

Run: >a.out

2. Compile and Run Fortran Programs

Compile: >f77 file.f /u1/gcs-83d/jtaylor/lib2

***** warning *****
a message "warning: table of contents for archive is
out of date, rerun ranlib(1)" will appear, ignore it

lib2 - library containing fortran graphics
subroutines

Run: >a.out
VITA

John W. Taylor was born on 5 April 1953 in Bethesda, Maryland. He graduated from high school in Fall River, California in 1971 and attended San Jose State University from which he received the Degree of Bachelor of Science in Geology in January 1976. Upon graduation, he received a commission in the USAF through the ROTC program. He then served as a computer programmer/analyst in San Antonio, Texas until entering the School of Engineering, Air Force Institute of Technology, in June 1982.
Implementation and Evaluation of a Core Graphics System on a VAX 11/780 with a UNIX Operating System

A Pascal Core Graphics System from the University of Pennsylvania using the VAX VMS 11/780 Operating System was converted to run on the Air Force Institute of Technology (AFIT) VAX 11/750 UNIX operating system. Major problems in converting from the VMS operating system to the UNIX operating system were encountered, but they were solved and documented for others to use and avoid. The size of the Pascal Core System and its lengthy compile times were a major system limitation. Device drivers were written in C for the Visual 550, the Tektronics 4014, and the Hewlett Packard 7220A 4 color plotter and can be applied to other projects. The Core package that was converted follows the SIGGRAPH 1979 standard. An application program, called intcore.run, is available that demonstrates the Pascal Core routines that are supported by this system.