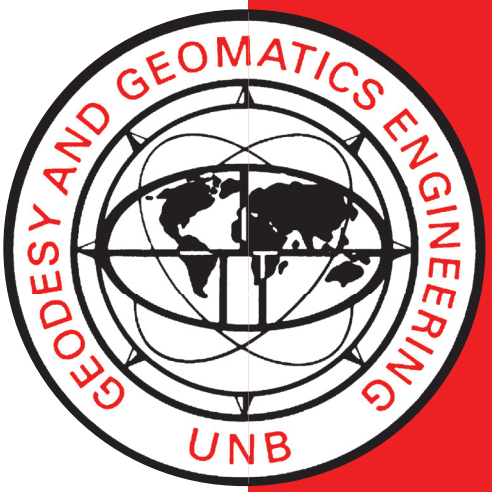
THE EVALUATION AND IMPLEMENTATION OF AN APPLE II MICRO-COMPUTER AS AN INTERACTIVE GRAPHICS TERMINAL

LEONARD B. SLIPP

February 1983



TECHNICAL REPORT
NO. 89

PREFACE

In order to make our extensive series of technical reports more readily available, we have scanned the old master copies and produced electronic versions in Portable Document Format. The quality of the images varies depending on the quality of the originals. The images have not been converted to searchable text.

THE UNIVERSITY OF NEW BRUNSWICK

SCHOOL OF COMPUTER SCIENCE

Revised

CS4993

Project Report

The Evaluation and Implementation of an APPLE II Micro-Computer as an Interactive Graphics Terminal

by

Leonard B. Slipp

Fredericton, New Brunswick, Canada

February, 1983

Reprinted October 1986

ACKNOWLEDGEMENTS

I would like to thank the following people whose assistance and suggestions greatly assisted me in the completion of this project.

Dr. Bernd Kurz -- School of Computer Science

Dr. Richard Langley -- Dept. of Surveying Engineering

Mr. Brad Nickerson -- Dept. of Surveying Engineering

Mr. See Hean Quek -- Dept. of Surveying Engineering

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Project Objective

The topic of this report was suggested by members of the Department of Surveying Engineering who required a facility that would take a set of coordinate pairs from an IBM 3032 computer, and plot them as a graph upon the screen of an Apple II micro-computer, running under the Apple Pascal Language System. The software developed to create the graph was required to have an interactive conversation with the user, and provide him with a variety of options from which he could control the quality of the finished plot. The applications of such a facility could vary from the straight forward plotting of X versus Y point plots, to using longitude and latitude coordinates to trace the coast-line of a continent.

The project was broken up into two phases. The communications phase (see chapter 2) is responsible for bringing in data from the contents of a VSPC file, or the output of a VS Fortran program, and writing the information onto a diskette file. The plotting phase (see chapter 3) takes this file, and plots the graph subject to the user's specifications (see figure 1-1).

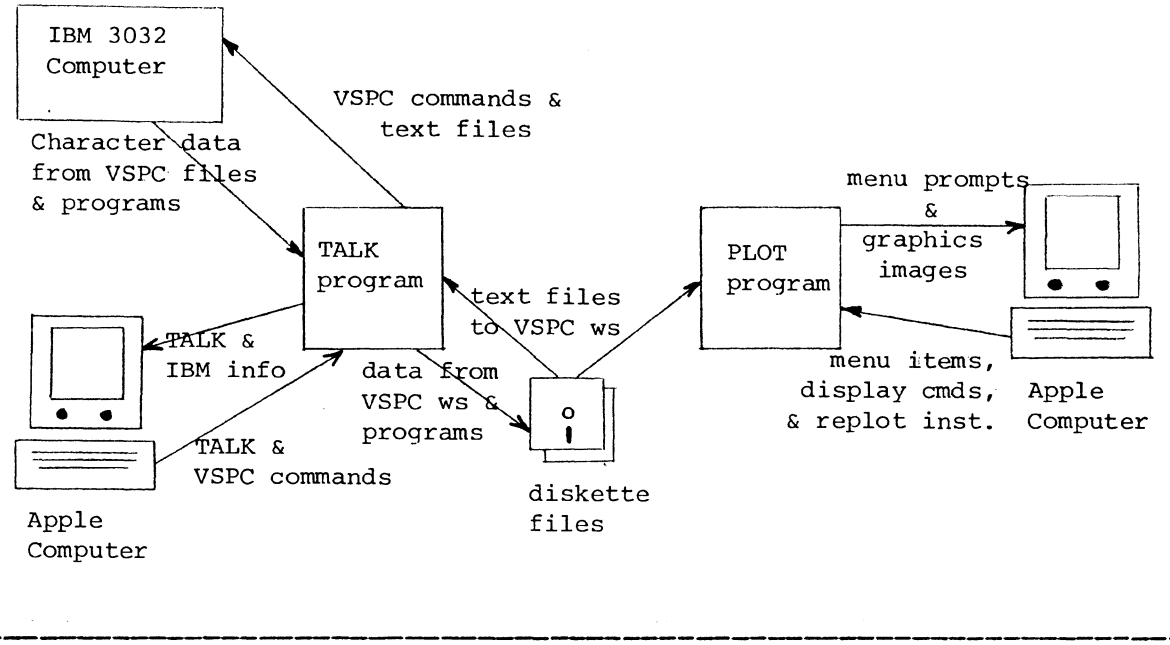


Figure 1-1: Flow of the Plot Package

* * * * * *

Apple II Display Modes

Depending upon the language and operating system used, the Apple II micro-computer has the capability of using three distinct display modes: Text, Low-Resolution ("Low-Res") graphics, and High-Resolution ("HI-Res") graphics. The Apple Pascal Language System will support only the Text and High-Resolution graphics display modes.

Text Mode

A general property of the Apple II Language System is that the Text mode of the Apple will display 24 lines of text on the screen with each line up to 40 characters in length. This implies that although the Apple Pascal Language System has been designed to work with lines that are up to 80 characters long, only 40 characters of the full Pascal page will be visible at any one time. In order to view a complete 80 character line, the user must toggle between the lefthand and righthand portions of the Pascal text page by typing "CTRL A".

Each character on the screen represents the contents of a data byte from a text page within the Apple. The character image is created from a dot matrix five pixels wide by seven pixels high. Characters are bordered on each side by a single pixel space to separate one character from another within the same line, and by a single pixel space above each character to separate successive lines of text. The characters are usually formed as white dots on a dark background, but the uppercase letters may be displayed in a reverse video mode, as black dots against a white background, by typing "CTRL R" while in the command mode of the Pascal Operating System, or by a WRITE(CHR(18)) statement within a Pascal program. The character display will return to the normal video mode when the user types "CTRL T" if in the command mode, or by a WRITE(CHR(20)) statement in a program.

Hi-Res Graphics Mode

The High-Resolution graphics mode of the Apple II micro-computer can display 53,760 dots in an array 280 pixels wide by 192 pixels high. Note that within the following section, any number enclosed within parentheses and preceeded by a $ sign is a hexidecimal number.

Picture Buffer Specifications

Hi-Res graphics images are created from one of two 8,192-byte areas of random access memory called "picture buffers". The first of these buffers is called the primary page, which begins at memory location 8192 ($2000), and extends through to location 16,383 ($3FFF). The remaining buffer is called the second page, and immediately follows the first by starting at location 16,384 ($4000), and ending at memory location 24,575 ($5FFF) (Apple, 1981a).

Each dot on the screen represents one bit from the picture buffer. Seven of the eight bits in each byte are displayed on the screen, with the remaining undisplayed eighth bit used to select the colours of the dots in that byte. Therefore, forty bytes are required to create each line on the screen. The least significant bit of the first byte in the line is displayed on the left edge of the screen, followed by the second bit, then the third, and so on up to the seventh bit. The first bit of the next byte then follows, and so on. Hence, a total of 280 dots are displayed on each of the 192 lines of the screen (Apple, 1981b).

Pixel Colour

On a black-and-white monitor or TV set, the dots whose corresponding bits are "on" (or equal to 1) will appear white, while those dots whose corresponding bits are "off" (or equal to 0) will appear black. If the video device is a colour monitor or TV, the result is not so straight forward due to the rather intricate way in which the National Television Screen Committee (NTSC) colour system produces colour, combined with the method that the Apple uses to get a lot of colour very economically (Apple, 1980). If a bit is "off" then its corresponding dot will always be black, but if a bit is "on", its colour will depend upon the position of that dot on the screen, and the value of the colour (eighth) bit within that byte. If the colour bit is off, and the dot is in the leftmost column on the screen, called "column 0", or in any even-numbered column, then the dot will appear violet. If, while the colour bit is off, the dot is in the rightmost column (column 279) or any odd-numbered column, then it will appear green. If two dots are located side-by-side, then they will appear white to the user. If the colour bit of a byte is turned on, then the colours blue and red are substituted for violet and green, respectively (see figure 1-2). Thus, there are six colours available in the High-Resolution graphics mode, subject to the following conditions:

(1) Dots in the even columns must be black, violet, or blue.

(2) Dots in the odd columns must be black, green, or red.

(3) Each byte must be either a violet/green or a blue/red byte. It is not possible to mix green and blue, green and red, violet and blue, or violet and red in the same byte.

(4) Two coloured dots side-by-side always appear white, even if they are in different bytes.

Status Of Colour Bit	Screen Column 0 1 2 . . . . . 277 278 279
off	violet green violet . . . . green violet green
on	blue red blue . . . . red blue red

Figure 1-2: Pixel Colour Combinations

* * * * * *

Apple Pascal Graphics Software

Turtlegraphics

Turtlegraphics is the standard Apple II Pascal Language System graphics package, and is stored as an Intrinsic unit in the Apple Pascal's SYSTEM.LIBRARY. It aquired its curious name from the "turtles" created by S. Papert and his colleagues at the Massachusetts Institute of Technology. During their development of the Logo computer language, they invented the idea of a "turtle" which could crawl around the screen and draw lines by lowering its "tail" which had a pen attached to it (Shapiro, 1982).

The coordinates of the Turtlegraphics window are identical to the pixel dimensions of the Apple's high resolution graphics screen where the X coordinates vary from 0 to 279, the Y coordinates from 0 to 191. The point (x=0,y=0) is in the lower left hand corner of the screen, and the point (x=279,y=191) in the upper right hand corner. Any line which is drawn outside these limits is clipped so that only the portion which is within the window is visible.

The functions supplied by Turtlegraphics perform the following actions:

(1) Initialization of the graphics screen, and switching the display between the Text and High-Resolution graphics pages.

(2) Creation of a viewport, specified in Apple II pixel dimensions, which will restrict the area of the screen in which the graphics image will appear.

(3) Change the colour of the turtle's pen and that of the background from a palette of white, black, green, blue, red and violet, conditional to the previously mentioned limitations of Apple graphics colour.

(4) Turn the turtle a specified number of degrees counter-clockwise from its present direction, or to a specific direction relative to the monitor's screen.

(5) Move the turtle a specified number of pixel units in the direction it is pointing, or to a specific screen coordinate.

(6) Write character strings onto the graphics screen.

(7) Transfer an array of bits to the screen, with each bit of the array mapped onto a screen pixel.

Applegraphics II

Applegraphics II is a graphics utility package developed by Apple Computer Inc., and consists of a reference manual, and a diskette containing the library file with the Intrinsic units required to implement Applegraphics and several demonstration programs. These Intrinsic units are a collection of powerful high-resolution graphics procedures which provide much greater flexibility in using the graphics capability of the Apple than the Turtlegraphics unit.

The desirable feature of Applegraphics is the series of drawing procedures which allows the programmer to work within the world coordinate system. The world coordinate system can be thought of as a giant, invisible piece of graph paper with X and Y axes crossing at right angles to each other, thus defining the Cartesian plane. The range of values of the X and Y coordinates may be any real number extending from the largest negative value (approx. -3.40282E38) to the largest positive value (approx. 3.40282E38) (Luehrmann et al., 1981). The advantage of this system is that a rectangular "window" may be specified anywhere on the world coordinate grid, thus enabling the user to work with units that he is familiar with rather than the awkward dimensions of the output device. The window will then be automatically mapped into the defined viewport, so that every point in the window will have a corresponding point in the viewport (Apple, 1981a).

The functions supplied by Applegraphics perform the following actions:

(1) Display either the Text or High-Resolution graphics pages upon the Apple's monitor.

(2) Create a viewport specified in "normalized device coordinates" (NDC) units.

(3) Change the colour of the line drawn, and that of the background to any one of the colours available in the Apple's colour palette.

(4) Write real and integer values and character strings upon the graphics screen.

(5) Move or draw from the current pen position to an absolute world coordinate point, to a specific screen position, or in terms of a relative displacement of either value.

(6) Project a 3 dimensional object onto a 2 dimensional display.

(7) Perform common geometric transformations such as scaling, translating, rotation and skewing.

Evaluation

Both packages are quite easy to use and are well documented, however a comparison of their features resulted in the choice of the Applegraphics II utility. Turtlegraphics is a good package to use if one is experimenting with the fundamentals of graphics, but it does not have the powerful routines that were required for this project. The plotting package would be required to handle numbers of arbitrary size. The Applegraphics II utility, because of its ability to work within the world coordinate system, would facilitate the implementation of this capability. The only apparent disadvantage with the Applegraphics package is that its three Intrinsic units are quite large, and consequently limit the size of the program in which they can be used.

Installing Applegraphics

The Applegraphics procedures are stored as a set of three Intrinsic units, and must be in the SYSTEM.LIBRARY of Pascal's "boot" diskette to be accessable to a program. The reference guide states that if the user has not modified his SYSTEM.LIBRARY, then he can simply transfer the SYSTEM.PASLIB file from the Applegraphics diskette to the SYSTEM.LIBRARY. However the SYSTEM.LIBRARY contained the PEEKPOKE unit required by the TALK program, so the LIBRARY utility on the APPLE3 diskette of the Pascal Language System was initially used to install the Applegraphics units individually into the SYSTEM.LIBRARY. A problem developed in that every time the TALK program, or a program which used the Applegraphics procedures was run, the Apple repeatedly gave errors stating that the program stack had overflowed. This problem was solved by putting the PEEK and POKE functions of the PEEKPOKE

unit directly into the TALK program, and transferring the SYSTEM.PASLIB file of the Applegraphics diskette to the SYSTEM.LIBRARY of the boot diskette.

* * * * * *

CHAPTER 2

COMMUNICATIONS

The TALK program was originally written by Lord (1982) to handle communications with VSPC through the Apple, and provide a method of transferring data files, collected by the RECEIVER program from a satellite Doppler receiver, to VSPC files. Since a major portion of the present project was to bring data in from the IBM, the original TALK program was modified to meet these requirements.

ASCII Terminal Option

This option of the TALK program [see Appendix D] essentially turns the Apple keyboard and display monitor into an ASCII terminal operating in a half-duplex mode at a transmission rate of 300 baud. This mode has aquired a limited intelligence through the addition of an improved editing facility.

Keyboard Buffer

Lord's original TALK program sent characters typed from the keyboard directly through the Asynchronous Communications Interface Adaptor (ACIA) to VSPC, but a limitation with this method was that it could not support an improved editing facility. Such a facility required a buffer which would store the characters typed from the keyboard, and then send them to VSPC once the return key was struck.

The keyboard buffer which was implemented is an 80 element character array in which the characters typed from the keyboard are stored.

When the user hits the return key, signifying the end of the command, the characters stored within the buffer are sent individually through the ACIA to VSPC. Should the user type more characters than the keyboard buffer will hold, each successive keystroke will produce a warning beep, indicating that the keyboard buffer is full. The buffer size can easily be changed by assigning a new value to the LINELEN constant declared at the beginning of the TALK program [see Appendix D].

Cursor Control Keys

The cursor direction keys are used to position the cursor throughout a VSPC command, and thus provide the user with a character editing ability. To edit a command, the non-destructive cursor is moved to the incorrect character, and the correct character is then typed.

Left arrow

Each time the left-arrow key is pressed, the cursor will move one position to the left. If the cursor is at the first character position of a command string when the left-arrow key is pressed, the Apple will respond with a warning beep, and the cursor will remain at its position.

Right arrow

Each time the right-arrow key is pressed, the cursor will move one position to the right. If the cursor is positioned immediately following the last character in a line when the right arrow key is pressed, the Apple will respond with a warning beep, and the cursor

will remain at its position.

Delete Mode

The Delete mode is used to remove characters from the keyboard buffer. To use this mode, move the cursor to the offensive character, and type "CTRL D". The character at that position will be removed from the screen, and all characters to the right of the cursor will move left one position to fill in the space. The cursor will then return to the position in the command where the deleted character had been, and so rest upon the letter which followed the deleted one. If the user needs to remove a string of characters, he may type "CTRL D" while holding the "REPEAT" key. This action will make successive calls to the deletion routine as long as the "REPEAT" key is depressed.

Insert Mode

The Insert mode is used to insert additional characters within VSPC commands. To use this mode, simply move the cursor to the position at which the characters are to be inserted, and type "CTRL I". All of the characters, starting at the cursor position up to the end of the command string, are copied from the keyboard buffer into a storage area and then erased from the Apple's screen. The user may then type the additional characters to complete the command. While the user is in this mode, he may use the cursor direction keys and the Delete mode to further edit the remaining characters on the screen.

To restore the saved portion of the command, type "CTRL Q". The cursor will move to the end of the characters, and write the stored characters from the storage area onto the screen while copying them

back into the keyboard buffer. The cursor will stop at the position following the last character. If the user should hit the "RETURN" key before he restores the command by typing "CTRL Q", the program will automatically restore the characters, and then send the command to VSPC.

One note of caution with respect to this mode is in order. If the user saves part of a command by typing "CTRL I", makes additional changes to the remaining portion, and then types "CTRL I" again without restoring the saved characters, then the first set of characters stored will be replaced by the characters from the present cursor position to the end of the character string. This means that if the user needs to make several insertions throughout a command, then he should always remember to restore the saved portion by typing "CTRL Q" after making additions, and before typing "CTRL I" to make subsequent insertions.

* * * * * *

Copy VSPC Files/VS Fortran Program Output Option

This option provides the user with the ability to copy data from the IBM computer onto a diskette for later use. Upon entering this mode, the program will ask the user to specify the group of characters which will be accepted into the character buffer (see figure 2-1). If the user selects the "ASCII CHARACTERS" option, then the program will accept all of the valid ASCII characters, except for the DLE, DC3 and LF characters, which will disrupt the Pascal Editor during the editing of a text file. If the user indicates that he wants to accept "NUMERIC TYPE ONLY" information, then only the following characters will be permitted entry into the internal character buffer:

+	--	Positive Sign
-	--	Negative Sign
.	--	Decimal Point
0,1,...,9	--	Numeric Digits
E	--	Exponential Format
[ ]	--	Blank Character
[CR]	--	Carriage Return

After the user has indicated the type of information which he wants to accept, the program will display another menu, giving the user the option of copying the contents of a VSPC file, or copying the output of a VS Fortran program. If the user indicates that he wants to copy the contents of a VSPC file, then the procedure will ask him to specify the file name (see figure 2-1). If he chooses to copy the output of a program, the procedure will ask for the name of the VS Fortran

program (see figure 2-2). Regardless of the option chosen, should the user reply to either name prompt by pressing the return key without entering anything, then the program will leave this mode, and the TALK program's main command menu will reappear on the screen.

```
        == PROCEDURE TO COPY IBM FILES ==

        == SPECIFY VALID CHARACTER SET ==

           1. - ASCII CHARACTERS
           2. - NUMERIC TYPE ONLY

        == ENTER 1 OR 2 :

        == SELECT 1 OF THE FOLLOWING OPTIONS ==

           1. - COPY VSPC FILE CONTENTS
           2. - RUN A VS FORTRAN PROGRAM

        == ENTER 1 OR 2 : 1

        == ENTER THE VSPC FILE NAME ==
        == FILE NAME ==>
```

Figure 2-1: Menu to Copy the Contents of a VSPC File

```
== PROCEDURE TO COPY IBM FILES ==

== SPECIFY VALID CHARACTER SET ==

   1. - ASCII CHARACTERS
   2. - NUMERIC TYPE ONLY

== ENTER 1 OR 2 :

== SELECT 1 OF THE FOLLOWING OPTIONS ==

   1. - COPY VSPC FILE CONTENTS
   2. - RUN A VS FORTRAN PROGRAM

== ENTER 1 OR 2 : 2

== ENTER THE FORTRAN PROGRAM'S NAME ==
== PROGRAM NAME ==>
```

Figure 2-2: Menu to Copy Output from a VS Fortran Program

The procedure receives data from the IBM by issuing a series of VSPC commands through the ACIA. The first instruction sent is the "TAPE" command, which instructs VSPC to append a "DC1" control character at the end of a string of characters sent by the IBM. As the data is coming into the Apple through the input side of the ACIA, the procedure will search for this character to determine when VSPC has responded to the command, and is ready for more input. The instructions that follow the "TAPE" command depend upon the option selected from the menu. If the user indicated that he wanted to copy the contents of a VSPC file, then the procedure will send a "LOAD filename" command to VSPC, where filename is the name of the file specified by the user. The procedure will then issue a "LIST NOLINE" command to display the contents of the file upon the screen without the accompanying VSPC

line numbers. If the user indicated that he wanted to copy the output of a VS Fortran program, then the procedure will send a "RUN pgmname" command to VSPC, where pgmname is the name of the VS Fortran program specified by the user. Note that this option does not provide an opportunity for the user to issue the "ALLOCATE" command to assign a file to a processor unit-number. If the VS Fortran program references any I/O files, then the user will have to issue the "ALLOCATE" command through the "ASCII Terminal" mode of the TALK program, before he enters the "COPY" mode to copy the output onto a diskette.

As the data is coming into the Apple, all of the incoming characters will be displayed upon the monitor's screen, but the invalid characters will be screened out before they are stored within the internal buffer. The characters are continuously displayed and stored until the IBM has sent the "DC1" character, indicating the end of file, or until the character buffer has filled to capacity. Once either of these events occurs, the program will clear the screen and display the following information:

```
                    == COPY IBM FILES ==

          1. IBM --> MEMORY

               SOURCE FILE : xxxxxxxx

               nnnnn INVALID CHARACTERS DETECTED

               nnnnn CHARACTERS WITHIN BUFFER

          2. MEMORY --> DISK

               ESTIMATED BLOCK REQUIREMENT  nn

               RECEIVING FILE :
```

Figure 2-3: Buffer Status Display from the Copy Output Option

The first item in the display is the name of the source file or program, which was specified by the user upon entering the "COPY" option. The second item indicates the number of invalid characters which were detected by the procedure. This may include such characters as line feeds, device control characters such as DCE, DLE, and other incidental characters, and depends upon the decision of the user to accept most of the ASCII characters or only numeric type information. The third item represents the number of characters which are actually held within the character buffer.

The first item of the second section gives the user an estimate of the size of the file to be created when the buffer is written onto the diskette. This information is very useful to the user in that he has advance knowledge of the amount of unused, contiguous blocks required,

and may insert a diskette with sufficient space. If the number of characters within the buffer is greater than 16,000, the program will create two files. This is in order that the user may, if necessary, edit the created files (16,000 characters is the maximum file size that can be handled by the Apple Pascal Editor).

If there are fewer than 16,000 characters within the buffer, then the procedure will prompt the user for the name of the receiving diskette file. The user should respond with the file name in the form "vol:file.TEXT". If the user presses the return key without entering anything, then he will leave this mode, and the main command menu of the TALK program will reappear on the monitor's screen. The user is forwarned that he will lose all of the information stored within the buffer if he does not specify a diskette file to receive the buffer's contents. If the user enters a volume name which is not online, then the procedure will issue a message stating the I/O error value, and then prompt the user to re-enter the file name. The values of the I/O errors may be found on page 32 of the Apple Pascal Language Reference Manual. Once the program accepts the file name, then it will dump the buffer's contents into the file. If two files are necessary to hold the information within the buffer, then it will automatically append a "1" and "2" to the file name entered by the user during file creation, so the user and the system may distinguish between the files. Once the files have been created, then the main command menu of the TALK program will reappear on the monitor's screen.

Input Buffer

The input buffer is necessary due to an unfortunate feature of the Pascal Operating System. When a program writes data onto a diskette, the Pascal Operating System will disable all of the program's interrupts, so any characters transmitted to the Apple during this time will be lost since an interrupted program cannot return and pick them up before they have been overwritten by following characters. The solution to this problem was the creation of an input buffer, which would store the characters received through the ACIA, and then write them to the diskette file once the IBM had sent a "DC1" control character, indicating the end of file, or when the input buffer had filled to capacity. The obvious limitation with this method is that the amount of information that can be copied onto diskette is limited to the size of the buffer, so the possibility certainly exists that the input buffer will hold only a portion of the data sent by the IBM. If the amount of information sent by the IBM is greater than the storage capacity of the input buffer, then the program will determine the last complete record sent by VSPC, and then write all of the information up to that point into the file. The procedure will also write the message "*** BUFFER OVERFLOW ***" upon the screen to indicate to the user that the buffer had indeed overflowed.

The size of the input buffer is dependent upon the the available memory size of the Apple that the program is run on. The 64K Apple systems belonging to the Department of Surveying Engineering will support an input buffer of 25,000 characters; any larger array will force the program stack to overflow. Should the user need to reduce to size of the buffer, it may be changed by reassigning the value of the BUFFLIMT constant declared at the beginning of the TALK program [see Appendix D].

Transfer Text Files

Selection of this mode will allow the user to specify any diskette file of type ".TEXT" to be transferred to a VSPC workspace. This option was written by Lord, and has not been modified in any way. Although it is not used with the creation of files for the PLOT program, it does provide a very useful feature for transferring Pascal programs and character data to the IBM for further processing.

* * * * * *

CHAPTER 3

PLOTTING

The PLOT program [see Appendix D] takes a text file created by the TALK program, converts this information into numeric values, and then creates a high-resolution plot on the Apple's screen. To run the PLOT program, select the "X)CUTE" option in the main command menu of the Pascal Operating System. Pascal will reply with a prompt for the name of the file, to which the user should type "APLEPLT:PLOT.CODE". The operating system will load the program into memory, and execution will begin.

Source Files

The first request of the PLOT program is for the name of the file which contains the data to be plotted. The user should reply with the file name in the form "vol:file.TEXT" (eg. MYDISK:PLTDAT.TEXT). If the user hits the return key without entering anything, execution of the program will cease, and the Apple will return to the main command menu of the Pascal Operating System. If the user enters a volume or file name which is not online, the program will issue an error message indicating the value of the I/O error, and then prompt the user to once again enter the proper file name. The values of the I/O errors are found on page 32 of the Apple Pascal Language Reference Manual.

File Format

Since the source file containing the data is a text file, all of the numbers which will be plotted are stored as a sequence of ASCII characters. The characters within the file should be of numeric type only, that is they must be from the set of:

+	--	Positive Sign
-	--	Negative Sign
.	--	Decimal Point
0,1,...,9	--	Numeric Digits
E	--	Exponential Format
[ ]	--	Blank Character
[CR]	--	Carriage Return

If the user had selected the "NUMERIC TYPE ONLY" option within the "COPY" mode of the "TALK" program (see Chapter 2) before the data was transferred to the Apple, then the file will contain only the above mentioned numeric characters.

The conversion necessary between the string of characters, and the real values required by the Applegraphics procedures, is done internally by the Pascal "READLN" procedure. If any character is encountered which is not of the numeric form described above, then the program will issue an error message indicating that an invalid character had indeed been encountered, and program execution will cease. Numbers on the file should be separated from one another by at least one blank character so the "READLN" procedure will be able to separate one value from another. An example of a line numbered VSPC file contain-

ing numbers to be plotted is shown in figure 3-1.

```
        10   1.00000   0.98007   0.19867   0.08885   1.00000
        20   2.00000   0.92106   0.38942   0.24629   1.41421
        30   3.00000   0.82534   0.56464   0.43737   1.73205
         .         .         .         .         .         .
         .         .         .         .         .         .
         .         .         .         .         .         .
         .         .         .         .         .         .
         .         .         .         .         .         .
         .         .         .         .         .         .
         .         .         .         .         .         .

       330  33.00000   0.95023   0.31154   0.80036   5.74456
       340  34.00000   0.86494   0.49411   1.28849   5.83095
```

Figure 3-1: Sample Contents of a VSPC File

* * * * * *

Plot Menu

The menu of the PLOT program (see figure 3-2) is a series of questions that asks the user to specify the organization of the data to be plotted, and the type of output desired. The content and structure of the menu was taken largely from the Autoplot facility of the Hewlett Packard 2648A graphics terminal. A great deal of attention was paid to the inclusion of statements which would detect or correct any invalid user responses.

```
== ENTER THE NAME OF THE SOURCE FILE ==
== FILE NAME :

      PLOT MENU

A. PLOT SPECIFICATIONS
 1. NO. OF COLUMNS
 2. X IS COLUMN
 3. NO. OF Y COLUMNS (1 OR 2)
 4. Y IS COLUMN
 5. PLOT TYPE - 1 = . (POINT)
                2 = -- (LINE)
 6. MIN X =
 7. MAX X =
 8. MIN Y =
 9. MAX Y =

B. AXIS SPECIFICATIONS
 1. PLOT THE AXES ?  (Y/N)
 2. ENCLOSE THE PLOT AREA ? (Y/N)
 3. NAME OF X AXIS :
 4. UNITS BETWEEN X TICS
 5. UNITS BETWEEN X LABELS
 6. NAME OF Y AXIS :
 7. UNITS BETWEEN Y TICS
 8. UNITS BETWEEN Y LABELS

C. PLOT OPTIONS
 1. PLOT TITLE :
 2. SKIP NUMBER OF POINTS
 3. PLOT NUMBER OF POINTS
```

Figure 3-2: The Plot Menu With One Y Value

```
== ENTER THE NAME OF THE SOURCE FILE ==
== FILE NAME :

      PLOT MENU

A. PLOT SPECIFICATIONS
 1. NO. OF COLUMNS
 2. X IS COLUMN
 3. NO. OF Y COLUMNS (1 OR 2)
 4. Y COLUMN - POINT PLOT :
 5. Y COLUMN -  LINE PLOT :
 6. MIN X =
 7. MAX X =
 8. MIN Y =
 9. MAX Y =

B. AXIS SPECIFICATIONS
 1. PLOT THE AXES ?  (Y/N)
 2. ENCLOSE THE PLOT AREA ? (Y/N)
 3. NAME OF X AXIS :
 4. UNITS BETWEEN X TICS
 5. UNITS BETWEEN X LABELS
 6. NAME OF Y AXIS :
 7. UNITS BETWEEN Y TICS
 8. UNITS BETWEEN Y LABELS

C. PLOT OPTIONS
 1. PLOT TITLE :
 2. SKIP NUMBER OF POINTS
 3. PLOT NUMBER OF POINTS
```

Figure 3-3: The Plot Menu With Two Y Values

Plot Specifications

The nine items in the Plot Specifications section of the menu asks the user to specify the physical organization of the data to be plotted, the type of line the coordinates are to be represented by, and the scale limits of the graph.

No. of Columns

This value specifies the number of columns which make up the data. It must be a positive integer less than or equal to the MAXCOL constant declared at the beginning of the PLOT program [see Appendix D]. Currently, the value of MAXCOL is 10.

X Is Column

This value specifies which column of the data array will be used for the X values. It must be a positive integer less than or equal to the number of columns specified in the first item of the menu.

No. of Y Columns (1 or 2)

This value specifies the number of Y columns to be plotted against the values in the X column. The program will only accept the value 1 or 2.

If the user answered this question by indicating that he wanted to plot only one Y value against each X point, then the program will ask the following two questions (see figure 3-2):

Y Is Column

This value specifies which column will be used for the Y values. It must be a positive integer less than or equal to the number of columns specified in the first item of the menu.

Plot Type

This value specifies the line type to be used when the program is

creating the plot. Specifying "1" will indicate to the program that a dot should appear at each point while a "2", or any other character specifies that a line should be drawn between successive points.

If the user responded to the third item of the menu by indicating that he wanted to plot two values of Y for each X point, then the program will ask the following questions (see figure 3-3):

Y Column - Point Plot

This value specifies which column of data is to be used as the first set of Y values. The data will be represented as points on the graphics output. The number must be a positive integer less than or equal to the number of columns specified in the first item of the menu.

Y Column - Line Plot

This value indicates which column of data is to be used as the set of Y values, and be represented as a line on the graph. The number must be a positive integer less than or equal to the number of columns specified in the first item of the menu.

The following four items specify the scale limits of the graph. All numbers must be within the ranges of real numbers permitted in the Apple Pascal language (i.e. -3.40282E38 to 3.40282E38).

Min X

This number specifies the minimum value to be assigned to the X axis. Any value within the range of real numbers permitted by the Apple Pascal Language will be accepted.

Max X

This number specifies the maximum value to be assigned to the X axis. Any real value which is greater than the minimum value of X will be accepted.

Min Y

This number specifies the minimum value to be assigned to the Y axis. Any value within the range of real numbers allowed by the Apple Pascal Language will be accepted.

Max Y

This number specifies the maximum value to be assigned to the Y axis. Any real value which is greater than the minimum value of Y will be accepted.

Axis Specifications

The following eight items in the Axis Specifications section provide the user with several options concerning the appearance of the axes on the graph.

Plot The Axes

The first item of this section asks the user if he would like the axes (includes the bisecting lines X=0, Y=0 if they are within the plot window) to appear upon the graph. If he types a "Y", then the axes will be drawn upon the screen. If he enters an "N" , or any other character, they will not appear.

Enclose The Plot Area

This prompt asks the user if he would like a line drawn around the limits of the graph at the values specified by the minimum and maximum values of X and Y in the Plot Specifications section. If he types a "Y", then a rectangle will be drawn to enclose this area. If he enters an "N" , or any other character, the rectangle will not appear.

If the user indicates to the program that he does not want either the axes or rectangle drawn, then the following items of the Axis Specifications section will not appear.

Name of X Axis

This item allows the user to specify the title of the X axis. The number of characters that can be printed is limited to 34, so should the user enter a title which is greater than 34 characters in length, only the first 34 will appear. If the user hits the return key without entering any characters, then this field will remain blank.

Units Between X Tics

This real number specifies how many units on the X axis each tick mark will represent. For example, if the range of X values varied from -1 to 1, and the user specified tick marks at every tenth of a unit, i.e. "0.1", then a tick would appear at the -1.0, -0.9, -0.8, .., -0.1, 0, 0.1, .. , 0.9, 1.0 points of the X axis. If the user enters a negative number, then the program will use the absolute value of that number to calculate the tick positions. If the user enters zero, no tick marks will appear along the X axis.

Units Between X Labels

This integer number specifies the number of tick marks between labels (i.e. the numeric values assigned to certain tick marks) on the X axis. In our example, if the user specifies 2 tick intervals between each X label, then the numeric values will appear at tick marks -1.0, -0.8, .. , -0.2, 0, 0.2, .., 0.8, 1.0. If the user specifies a negative number, the program will use its absolute value to calculate the label position. If the user enters zero, then labels will not appear on the X axis.

Name of Y Axis

This item allows the user to specify the title of the Y axis. The number of characters that can be printed is limited to 18, so should the user enter a title which is greater than 18 characters in length, only the first 18 will appear. If the user hits the return key without entering any characters, then this field will remain blank.

Units Between Y Tics

This real number specifies how many units on the Y axis each tick mark will represent. For example, if the range of the Y values is between -10 and 10, and the user has specified tick marks at every unit, then a tick would appear at the -10, -9, -8, .., -1, 0, 1, .. , 9, 10 points of the Y axis. If the user enters a negative number, the program will use the absolute value of that number to calculate the tick positions. If the user enters zero, no tick marks will appear along the Y axis.

Units Between Y Labels

This integer number specifies the number of tick marks between labels (i.e. the numeric values assigned to certain tick marks) on the Y axis. In our example, if the user specifies 5 tick intervals between each Y label, then the numeric values will appear at tick marks -10, -5, 0, 5, 10. If the user specifies a negative number, the program will use its absolute value to calculate each label position. If the user enters zero, then labels will not appear on the Y axis.

Plot Options

The three items in the Plot Options section allow the user to enter the title of the plot, and also specify the number of points to be skipped and plotted.

Plot Title

This item allows the user to enter the title of the graph, which will appear along the top of the screen. The number of characters that can be printed is limited to 40, so should the user enter a title which is greater than 40 characters in length, then only the first 40 characters will be printed. If the user hits the return key without typing anything, then the title field will remain blank.

Skip Number Of Points

This integer value represents the number of points the program will skip when it begins plotting. If the user enters a negative number, the program will take the absolute value, and skip that number of points. If the user enters zero, the program will not skip any

points.

Plot Number Of Points

This integer value represents the number of points the program will plot. If the user enters a negative number, the program will take its absolute value, and then plot that number of points. If the user enters zero, then the program will plot all of the remaining points in the file.

* * * * * *

Screen Initialization

There are three procedures within the PLOT program that are used for preparing the high resolution screen from the options specified by the user through the program's menu. They begin initialization by clearing the graphics buffer, and then writing the plot and axis names upon the screen if they were entered. They will also draw the axes and enclosing rectangle, and position the axis ticks and labels if they have been specified by the user.

* * * * * *

Point Plotting

The main purpose of the PLOT program is to plot the data on the screen, and it centres around two routines which are responsible for the creation of the graph.

Points Procedure

The Points procedure builds an array of real numbers, which is used by the Draw procedure to determine the points to be plotted that the user specified. The size of the array is equal to the MAXCOL constant assigned at the beginning of the PLOT program, but the number of elements actually used in the array is equal to the number of columns of data in the source file, which was specified by the user through the program's menu.

Draw Procedure

The main purpose of the Draw procedure is to control the plotting of points upon the graphics screen. The procedure initially calls the Points procedure, which creates an array of real numbers. The Draw procedure selects the points to be plotted from the array by using the column numbers, entered by the user through the menu, as subscripts within the array. For instance, if the user indicated that he wanted the first column of data to be used for values of X, and the third column to be used for the values of Y, the Draw procedure will use the first element of the array returned by the Points procedure as the X coordinate, and the third element as the Y coordinate of the point. As the procedure creates the appropriate line or dot on the screen, it also writes the point coordinates onto the text page of the Apple, so

the user may view the point values by displaying the Text page on the screen. The Draw procedure will continue plotting points until it reaches the end of file, or the user halts the plotting process. When the program finally reaches the end of file, the Apple will "beep", indicating that it has finished plotting all of the points in the file.

Screen Display Modes

There are four commands available during the plotting stages of the graph which the user may find useful:

"CTRL G" - This command will instruct the program to display the contents of the Hi-Res Graphics buffer upon the video screen.

"CTRL H" - This command will instruct the Apple to halt further drawing upon screen, but remain within the PLOT program. It is used to prematurely halt the creation of the plot to allow the user to redefine a menu item, and then recreate the graph as described in the Replot facility in the following section.

"CTRL Q" - This command will halt any further execution of the program, and return the user to the main menu of the Pascal Operating System.

"CTRL T" - This command will instruct the program to display the contents of the Text buffer upon the screen.

Replot Facility

Once the program has completed drawing the graph, the user may change any number of the Plot menu items, and then redraw the graph subject to the new plot specifications.

Display the Current Menu

The current values of the Plot menu may be reviewed by typing "CTRL D". This command will only be recognized once the program has stopped plotting values, which occurs when the program has reached the end of the source file, or the user halts the plotting process by typing "CTRL H". The program will respond by writing a condensed version of the plot menu with the corresponding values written beside each item. A group of instructions will appear at the bottom of the screen to assist the user in changing a menu item, or terminating the PLOT program.

Changing a Menu Item

Any item in the menu, except the name of the source file, may be changed by typing "CTRL C". As with the menu display command, this key will only be recognized once the program has halted the plotting process. The menu change option will initially ask the user to enter the section letter ["A","B",or "C"] that contains the item he wants to alter. If the user types any other character, or hits the return key without entering anything, then he will leave the menu change section. After selecting a valid section letter, the program will ask for the number of the item the user wants to change within that section. If he replies with an item number that is outside the range of that par-

ticular section, the program will respond with the message "INVALID ITEM", and the section prompt statement will reappear. If the user replies with a "0", or any non-numeric character, the program will leave the menu change option.

Once the user has entered both a valid section and item, the specified menu question will appear. Due to the error checking by the program, the user may have to answer two or three questions in order to change the value of one item. For instance, if he wanted to change the value of MAX Y by specifying section A, item 9, the program will ask him to re-enter the MIN Y value along with the MAX Y value. This is necessary to ensure that the program will check that the new MAX Y value is greater than the MIN Y value.

After the user has left the menu change mode by typing an invalid response to either the section or item prompts, a group of instructions will roll up onto the screen to assist him in displaying the new menu, replotting the data under the new specifications, or terminating the PLOT program.

Replotting the Data

The command "CTRL P" will instruct the program to replot the data under the new specifications of the PLOT menu. As with the two other commands, this instruction will not be recognized until the program has stopped plotting. This command will erase the graphics buffer, and then call the procedures to recreate the image from scratch. While the plot is being redrawn, the user is free to use the "CTRL G" and "CTRL T" commands to view the graphics or text pages, the "CTRL H"

command to halt plotting, or "CTRL Q" to terminate the program, and return to the main command menu of the Pascal Operating System. Once the plot has been recreated, the user may change the PLOT menu again, and replot the data as many times as necessary.

Overlay Facility

In light of the fact that it may be desirable to create a plot using values contained in several files, the user may overlay a graph formed from the contents of another source file on top of the first graph by typing "CTRL O". The program will reply to this command by asking the user to specify the name of the new source file. The user should reply with the file name in the form "vol:file.TEXT" (eg. MYDISK:FILE2.TEXT). If the user hits the return key without entering anything, execution of the program will cease, and the program will return to the main command menu of the Pascal Operating System. If the user enters a volume or file name which is not online, then the program will issue an error message indicating the value of the I/O error, and then prompt the user to once again enter the proper file. The values of the I/O errors are found on page 32 of the Apple Pascal Language Reference Manual.

Once the program has accepted the file name, it will ask the user seven questions from the plot menu to determine the organization of the data on the file, which columns are to be used as X and Y values, how the new values should be represented on the graph, and lastly, how many points should be skipped and plotted. After the user has supplied all of this information, the program will begin plotting the

values from the second file subject to the menu specifications. As during the initial plotting, the user may use the CTRL G, CTRL H, CTRL Q and CTRL T commands to select the different view modes. After the program has finished plotting the second file, the user may overlay a third file onto the screen, and repeat this process as many times as necessary.

The user should take caution in one aspect of this facility. If he views the graph after one or more overlays of other files on top of the first one, decides to change one or more of the menu items, and then replot the graph (as opposed to an overlay), the program will clear the graphics screen, and then replot only the points from the most recent source file.

Hardcopy Facility

The image within the high resolution graphics page may be dumped onto an Epson MX-80 dot matrix printer (provided that it is driven by a "Grappler+" printer interface board) by typing "CTRL N". The program will reply to this command by asking the user to specify the various print options that he desires. These commands may be found in the Grappler+ operating manual, and they are also listed in Appendix C of this report. After the user has entered the desired options, the program will then attempt to dump the contents of Hi-Res page 1 onto the printer. If the printer is online, then printing will begin. If the printer is not online, then the warning message "NOT SELECTED" will appear upon the screen to inform the the user that the printer is offline. Once the printer has been brought back online, then printing

will commence. After printing is complete, the program will display the graphics image in Hi-Res graphics page 1 on the monitor screen.

Image Storage Facility

The graphics image within the first high resolution graphics page may be stored on a diskette file for later retrieval by typing "CTRL L". The program will respond to this command by prompting the user to enter the name of the receiving file. The user should reply with the file name in the form "vol:file.FOTO" (eg. MYDISK:GRAF1.FOTO). After the user has entered the file name, the program will examine the diskette to ensure that there is sufficient room on the diskette for the new file (each picture image requires 16 contiguous blocks of disk space). If the required space is found, then the picture buffer will be written into the diskette file, and the display will return to the graphics page. A separate program, called "LOOKC", has been written to retrieve graphics images for viewing only. The PLOT program will be modified in the near future to allow it to retrieve, display and modify graphics images which have been stored in a diskette file.

CHAPTER 4

CONCLUSIONS AND RECOMMENDATIONS

The software developed for this project fulfills the requirement for the implementation of a facility to create a "quick and simple" plot of numbers taken from an IBM 3032 computer. It has also demonstrated several advantages of the Apple II micro-computer. The project was further proof that the Pascal Language System is a very powerful package, and that it greatly enhances the flexibility of the Apple. It has also demonstrated the capabilities of the Apple's high resolution graphics display mode in conjunction with the procedures of the Apple-graphics utility package. I feel that the quality of Apple graphics is sufficient to enable it to be used in other applications within the Department of Surveying Engineering.

During the course of the project, some disadvantages of the Apple became apparent, notably the lack of an interrupt driven I/O handler, which restricted the amount of information the Apple could copy from the IBM through the TALK program. An attempt was also made to increase the transmission rate of the TALK program to 1200 baud, but the program was unable to keep up with the characters sent by VSPC before they were overwritten by following characters. This problem could possibly be overcome by rewriting the SCANACIA procedure in assembler language (this is being done now).

In closing, I hope that the users of this package find it helpful in their work, and that the results inspire them to develop new applications for the Apple.

Appendix A

APPLE PLOT USER'S GUIDE

The information within this appendix supplies the directions necessary to use the Plot package developed for this project.

System Diskettes

There are two diskettes which contain the software developed for this project.

APLEPLT diskette

This diskette is a copy of the APPLE1 diskette, to which has been added the code files required to run the TALK and PLOT programs. This diskette is also called the "boot" diskette, and will normally be in drive #4.

PLTBKP diskette

This diskette is used as a backup diskette, and contains copies of the source and code files for the TALK and PLOT programs as well as a copy of the SYSTEM.PASLIB file from the Applegraphics package. The SYSTEM.PASLIB file contains the Intrinsic units required to implement Applegraphics as well as the standard Intrinsic units found in the SYSTEM.LIBRARY, with the exception of the Turtlegraphics units.

* * * * * *

Booting the System

In order to access the programs described within this report, insert the APLEPLT diskette into drive #4, and a diskette that contains, or will receive, files to be used with the PLOT program into drive #5. Power on the Apple. After a bit of disk activity, one of the two fol-

lowing events should normally occur:

(1) You will be greeted with a musical jingle, and a short message indicating the system date and time according to the clock module within the Apple, or

(2) You will be greeted with a short "beep beep beep" sound, and a message asking you to re-initialize the system by typing the letter "I". After typing this letter, you should be greeted as described in event 1 above.

* * * * * *

Special Pascal System Keys

The Pascal Reference manuals describe several special keys which are used by the operating system to give the user limited control over the execution of most programs. A brief summary of these keys follows:

"CTRL RESET" - This command initiates a "warm boot" of the system which will stop almost any ongoing process at the expense of losing whatever is in the Apple's memory at that time. This command is usually necessary when the system "hangs" (stops and does not respond to the keyboard).

"CTRL @" - This command gives the user the ability to interrupt the execution of the current program. The Pascal Operating System will issue a message stating "PROGRAM INTERRUPTED BY USER". Execution of the interrupted program cannot be resumed, and the system must be re-initialized by pressing the space bar.

"CTRL A" - This command will toggle the text display between the righthand and lefthand portions of the 80 character Pascal text page.

"CTRL F" - This command causes all of the following program's output to be "flushed" (i.e. the program will continue running, and any characters which were written on the screen will remain, but no subsequent output will appear on the screen). Output to the screen will resume when the user types another "CTRL F".

"CTRL S" - This command will halt execution of a program until the user types another "CTRL S".

"CTRL Z" - This command initiates the Auto-follow mode or horizontal scrolling. The screen will scroll right and left to follow the cursor as it moves across the Pascal page. The horizontal scrolling can be disabled by typing "CTRL A".

If the user has any questions or problems concerning the use of these commands, he should consult the Apple Pascal Operating System Reference Manual.

* * * * * *

The TALK Program

The TALK program is used to communicate to VSPC for the main purpose of copying information from the IBM onto a diskette for use with the PLOT program. To run the TALK program, the APLEPLT diskette should be in drive #4, and the Pascal Operating System should be in the main command mode. The user should then type "X" indicating to Pascal that he wants to execute a program, and respond to the system's prompt for a file name by typing "#4:TALK". After the operating system loads the program into memory, execution will begin, and the main command menu will appear (see figure A-1).

```
        == SERIAL COMMUNICATIONS PROGRAM ==

        == COMMAND MENU ==

        OPTIONS ARE :
           A = ASCII TERMINAL MODE
           C = COPY VSPC FILES / PROGRAM OUTPUT
           T = TRANSFER ANY TEXT FILE
           Q = QUIT

        == ENTER COMMAND ==>
```

Figure A-1: Main Command Menu of the TALK Program

ASCII Terminal Option

This option enables the Apple to operate as an ASCII terminal to establish communications with the IBM computer. The user must enter this mode (by typing "A" from the main command menu of the TALK program) to initally sign on to VSPC so that he can use the other options of the TALK program to copy data onto a diskette or send information to a VSPC file. He must also use it to sign off VSPC before he terminates the TALK program.

There are several commands within this mode which the user may find useful:

Escape key - This key is equivalent to the "BREAK" or "ATTN" keys found on other ASCII terminals.

LEFT/RIGHT ARROW - These keys are used to position the cursor anywhere in a command, and thus provide the user with a character editing facility.

"CTRL C" - This command terminates the ASCII Terminal mode, and returns the user to the main command menu of the TALK program.

"CTRL D" - This command activates the Delete mode which will remove the character at the cursor position from the VSPC command.

"CTRL I" - This command activates the Insertion mode which gives the user the ability to insert additional characters within a VSPC command.

"CTRL Q" - This command terminates the Insertion mode.

If the user has any questions concerning the use of these commands, he should refer to Chapter 2 of this report.

Copy VSPC Files/Program Output

This option allows the user to copy data from the contents of a VSPC file, or the output of a VS Fortran program, onto a diskette file. Before selecting this mode, the user should have signed onto VSPC using the "ASCII Terminal" mode described above. Although this option is used primarily to bring information in for the PLOT program, it can be used to copy programs, character data, etc., from the IBM for use with other Pascal programs.

Upon selection of this mode, the program will initially prompt the user to select one of two character filters which the program will use to screen out invalid incoming characters. Only valid characters will be stored within the program's internal buffer. The "ASCII CHARACTERS" option will permit most of the valid ASCII characters within the buffer, with the exception of the DLE, DC3, and LF characters, which will disrupt the Pascal Editor during the editing of a text file. The "NUMERIC TYPE ONLY" option, on the other hand, will restrict the characters allowed within the internal character buffer to the following:

+	--	Positive Sign
-	--	Negative Sign
.	--	Decimal Point
0,1,...,9	--	Numeric Digits
E	--	Exponential Format
[ ]	--	Blank Character
[CR]	--	Carriage Return

If the user expects to be using the information that he is about to copy as a source file for the PLOT program, then he should select the "NUMERIC TYPE ONLY" option.

Once the user has replied to the prompt, the program will display another brief menu which will ask the user if the source of the information is a VSPC file, or a VS Fortran program. If the information is held within a VSPC file, then the user should enter a '1', and reply to the program's next prompt with the name of the appropriate VSPC file. If the information is generated as the output of a VS Fortran program, then the user should enter a '2', and reply to the program's next prompt with the name of the VS Fortran program.

The TALK program retrieves the information by issuing a series of VSPC commands, which will appear on the screen as they are sent. As the information is sent to the Apple, the invalid characters will be screened out, and the valid characters stored within the buffer. Once all of the information has been sent by the IBM, or the internal buffer has filled to capacity, a new screen display will appear on the monitor, which will indicate the number of invalid characters detected, the number of characters held within the buffer, and the number of free, contiguous blocks required to create this file on disk. The last statement on the screen is a prompt to specify the name of the receiving diskette file, to which the user should reply with a file name in the form of "vol:file.TEXT". If the number of characters held within the buffer exceeds 16,000, then the program will create 2 files, automatically appending a '1' and a '2' to the file name to differentiate between the two files. Once the files have been created, the main command menu of the program will reappear on the screen.

If the user has any difficulity with this mode, then he should

refer to chapter 2 of this report. The following examples illustrate the uses of this option.

Example 1

Figures A-2 and A-3 illustrate the program's prompts, and the user's replies to copy the contents of a VSPC file named APLEDATA into the diskette file DISK1:APLEINFO.TEXT.

```
        == PROCEDURE TO COPY IBM FILES ==

        == SPECIFY VALID CHARACTER SET ==

           1. - ASCII CHARACTERS
           2. - NUMERIC TYPE ONLY

        == ENTER 1 OR 2 : 1

        == SELECT 1 OF THE FOLLOWING OPTIONS ==

           1. - COPY VSPC FILE CONTENTS
           2. - RUN A VS FORTRAN PROGRAM

        == ENTER 1 OR 2 : 1

        == ENTER THE VSPC WORKSPACE NAME ==
        == WORKSPACE NAME ==> APLEDATA
```

Figure A-2: Example 1 -- Copying the Contents of a VSPC File

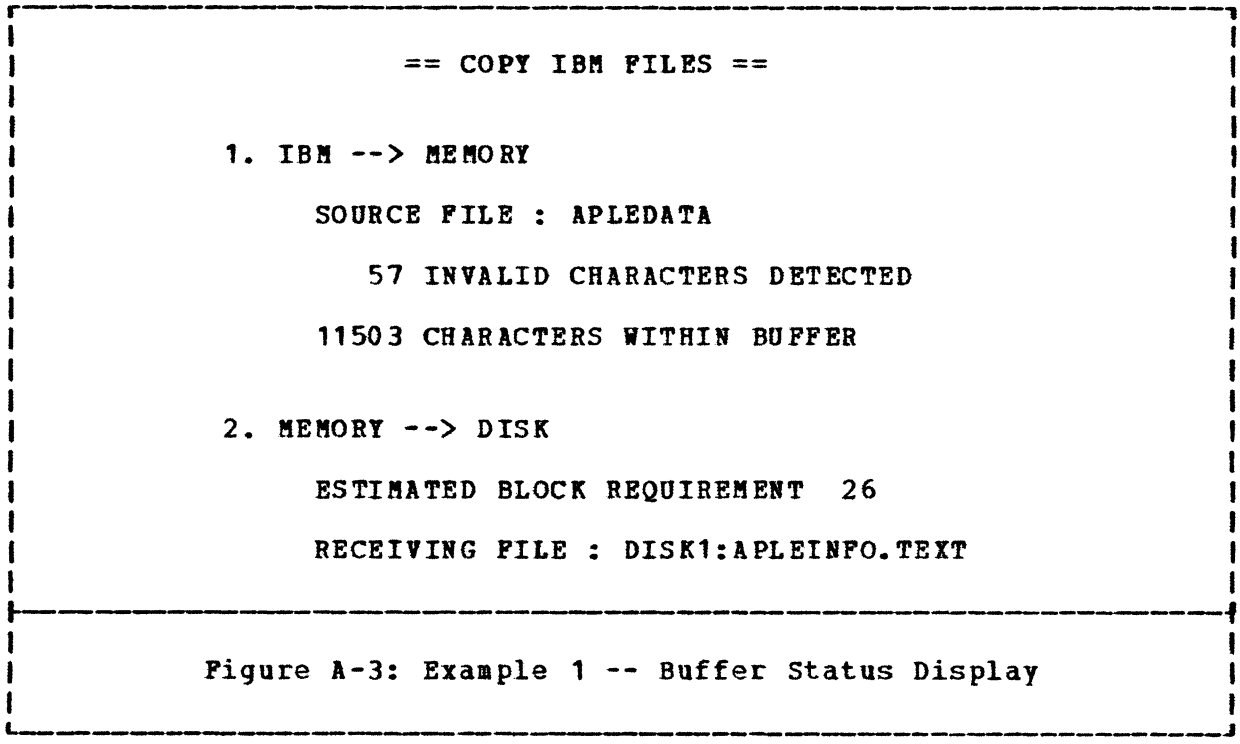


Figure A-3: Example 1 -- Buffer Status Display

Example 2

Figures A-4 and A-5 illustrate the program's prompts and the user's replies to copy the numeric output generated by a VS Fortran program named FORTPGM into the diskette file DISK1:APLEFORT.TEXT.

```
          == PROCEDURE TO COPY IBM FILES ==

          == SPECIFY VALID CHARACTER SET ==

             1. - ASCII CHARACTERS
             2. - NUMERIC TYPE ONLY

          ==  ENTER 1 OR 2 : 2

          == SELECT 1 OF THE FOLLOWING OPTIONS ==

             1. - COPY VSPC FILE CONTENTS
             2. - RUN A VS FORTRAN PROGRAM

          == ENTER 1 OR 2 : 2

          == ENTER THE FORTRAN PROGRAM'S NAME ==
          == PROGRAM NAME ==> FORTPGM
```

Figure A-4: Example 2 -- Copy Output from a VS Fortran Program

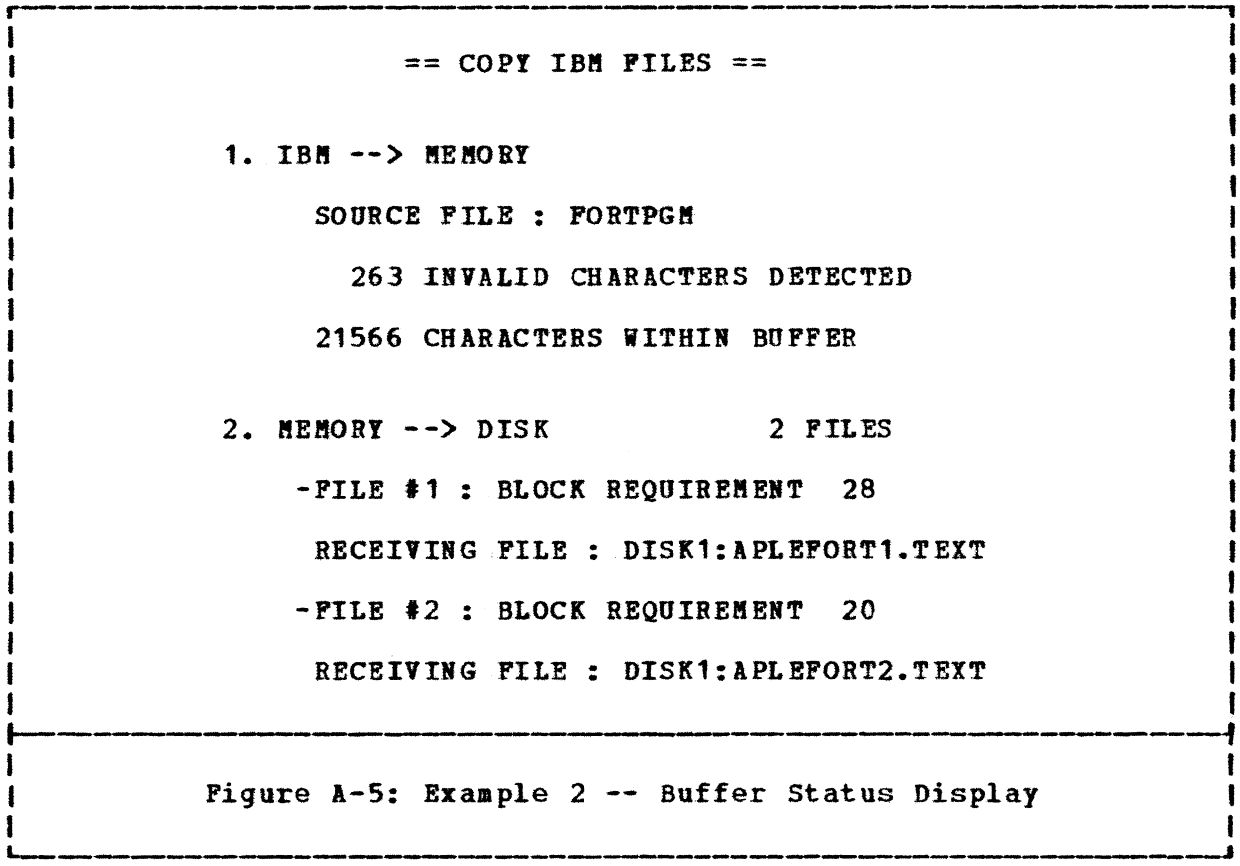


Figure A-5: Example 2 -- Buffer Status Display

Text File Transfer Option

This option will transfer any .TEXT file on a diskette into a VSPC file. The procedure will prompt the user for the file name, which should be in the form "vol:file.TEXT" (eg. MYDISK:FILE1.TEXT), and then the file name which VSPC will save the information under. When the user has entered this information, the program will send a series of VSPC commands to prepare the workspace, and then send the text file, line by line. Once all of the information has been sent, the TALK program will issue a "SAVE" command to save the information within the user's VSPC library.

Quit Option

The Quit option allows the user to terminate the TALK program, and return to the command mode of the Pascal Operating System. The user is reminded that this command will not sign him off VSPC, which he must do by entering the "ASCII Terminal" mode and entering the VSPC "OFF" command.

* * * * * *

The PLOT Program

The PLOT program is used to plot the data from text files created by the TALK program on the screen of the Apple. To run the PLOT program, the APLEPLT diskette should be in drive #4, and the Pascal Operating System should be in the main command menu. The user should type an "X", indicating to Pascal that he wants to execute a program, and then respond to the system's prompt for a file name by typing "#4:PLOT". After the operating system loads the program into memory, execution will begin, and the first item of the PLOT menu will appear on the screen.

Plot Menu

The Plot menu asks the user a series of questions concerning the organization of the data on the source file, and the desired appearance of the graph upon the Apple's screen. A detailed explanation of each question, and the type of response expected, is found in Chapter 3. The program will issue helpful error messages to the user should he supply any improper answers to the menu questions.

To clear up any possible misconceptions about the menu, the following example should be helpful. Suppose a user wanted to plot the values from the diskette file "DISK1:TSTPLT.TEXT". The information is arranged in 5 columns, with the X values in column 2, and the Y values in column 5. Each coordinate is to be represented as a dot on the screen. The Y values, which are temperature readings in Celsius degrees, vary from 0 to 25, while the X values, which are the corresponding depths of a thermometer in the ocean, vary from 0 (sea level) to 500 fathoms. The plot area is to be enclosed within a rectangle and the X and Y axes are not required. Tick marks are to appear along the Y edge of the rectangle at every 5 units, with labels at every tick unit (i.e. at tick positions 0, 5, 10, 15, 20, 25). Tick marks are to appear along the X edge of the rectangle at every 100 units, but the tick labels are not required. The title of the plot is "Ratio of Ocean Temp. to Depth", and all the points in the file are to be plotted.

The Plot menu to implement such a graph appears in figure A-6.

```
== ENTER THE NAME OF THE SOURCE FILE ==
== FILE NAME : DISK1:TSTPLT.TEXT

      PLOT MENU

A. PLOT SPECIFICATIONS
 1. NO. OF COLUMNS  5
 2. X IS COLUMN     2
 3. NO. OF Y COLUMNS (1 OR 2) 1
 4. Y IS COLUMN     5
 5. PLOT TYPE - 1 = . (POINT)
                2 = -- (LINE) 1
 6. MIN X = 0
 7. MAX X = 500
 8. MIN Y = 0
 9. MAX Y = 25

B. AXIS SPECIFICATIONS
 1. PLOT THE AXES ?  (Y/N) N
 2. ENCLOSE THE PLOT AREA ? (Y/N) Y
 3. NAME OF X AXIS : DEPTH - FATHOMS
 4. UNITS BETWEEN X TICS   100
 5. UNITS BETWEEN X LABELS 0
 6. NAME OF Y AXIS : DEG. CELSIUS
 7. UNITS BETWEEN Y TICS   5
 8. UNITS BETWEEN Y LABELS 1

C. PLOT OPTIONS
 1. PLOT TITLE : RATIO OF OCEAN TEMP. TO DEPTH
 2. SKIP NUMBER OF POINTS 0
 3. PLOT NUMBER OF POINTS 0
```

Figure A-6: Menu Implementing the Example Specifications

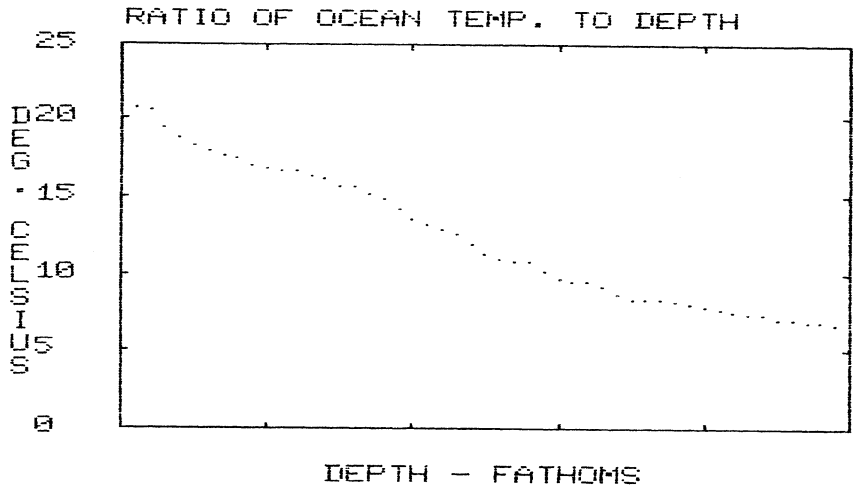


Figure A-7: Output from the Plot Example

Creating the Graph

After the user has entered the last item of the Plot menu, the program will switch the display to the graphics page of the Apple, and begin initialization of the screen according to the options specified through the menu. The initialization procedures will draw the enclosing rectangle with its respective tick marks and labels, and write the plot and axis titles upon the graphics screen. Once initialization is complete, the program will begin plotting the points of the file on the screen. The resulting plot is shown in figure A-7.

During the plotting stage, the user may find the following commands useful:

"CTRL G" - This command will instruct the program to display the contents of the Hi-Res Graphics buffer upon the monitor's screen.

"CTRL H" - This command will instruct the Apple to stop further drawing upon the screen, but to remain within the PLOT program. It is used to prematurely halt the creation of the plot to allow the user to redefine a menu item, and then recreate the graph.

"CTRL Q" - This command will halt any further execution of the program, and return the user to the main menu of the Pascal Operating System.

"CTRL T" - This command will instruct the program to display the contents of the Text buffer upon the monitor's screen.

Replotting the Graph

Once the program has completed plotting, the user may change any number of items within the Plot menu, and then recreate the graph under the new specifications. To view a condensed version of the present plot menu, type "CTRL D". The condensed menu of the example mentioned earlier in this section appears in figure A-8.

```
FILE : DISK1:TSTPLT.TEXT

      PLOT MENU

A. PLOT SPECIFICATIONS
 1. NO. COLS = 5  2. X COL = 2
 3. NO. Y COLS = 1
 4. Y COL = 5  5. PLOT TYPE = 1
 6. MIN X = 0.00000  7. MAX X = 5.00000E2
 8. MIN Y = 0.00000  9. MAX Y = 2.50000E1

B. AXIS SPECIFICATIONS
 1. AXIS N   2. ENCLOSED Y
 3. X AXIS : DEPTH - FATHOMS
 4. X TIC = 1.00000E2  5. X LABEL = 0
 6. Y AXIS : DEG. CELSIUS
 7. Y TIC = 5.00000  8. Y LABEL = 1

C. PLOT OPTIONS
 1. PLOT : RATIO OF OCEAN TEMP. TO DEPTH
 2. SKIP = 0  3. PLOT = 32767

CHANGE MENU   - CTRL C
OVERLAY FILE  - CTRL O
TERMINATE PGM - CTRL Q
```

Figure A-8: Condensed Menu for the Example

After the condensed menu has appeared, the user may change a menu item by typing "CTRL C". The procedure will then prompt the user to enter the letter of the section which contains the item he wants to change. If the user types any character other than "A","B", or "C", then the procedure will leave the menu change option. If the user has entered a valid section letter, then the procedure will prompt the user for the number of the item that he wants to change. If the user enters a number which is out of the range of the specified section, then the procedure will write "INVALID ITEM" on the screen, and then prompt for a new section. If the user enters a "0" or any other non-numeric character, the program will leave the menu change option.

If the user has specified both a valid section letter and item number, then the desired question will appear. Once the user has answered it, the section prompt will reappear to give the user the opportunity to change another menu item. When the user has finished making all of the necessary changes to the plot menu, then he may leave the menu change option by typing an invalid response to the section or item prompts.

Continuing with our example, suppose that after viewing the plot, the user decided to label the X axis at each tick mark, and to to draw the axes in the plot rather than enclose the plot area within a rectangle. To do this, he would initially type "CTRL C" to enter the menu change option, and then type the letter "B" in response the section prompt since the "UNITS BETWEEN X LABELS" is within the Axis Specifications section. When the item prompt appears, the user should reply with the number "5" and hit the return key. The first statement

that would appear is the "UNITS BETWEEN X TICS" item. Due to the large size of the PLOT program and the limited program size permited by the Pascal editor, the Plot menu was broken up into several groups of menu questions, hence the appearance of the question. Since the user did not need to change this value, he would enter the original value, which was 100, and then press the return key. Once the user replies to the first item, then the second item "UNITS BETWEEN X LABELS" will appear. Since the user wanted tick labels to appear at every tick mark, he should now reply to this prompt by typing a "1". After entering the new value for the tick label positions, the section prompt will reappear, to which the user would type "B", since the next item he wants to change is also within the Axis Specification section. The item prompt will appear, to which the user should reply with the number 1. The question "PLOT THE AXES ? (Y/N)" will appear, to which the user should type "Y", indicating that he wants the axis drawn on the graph. A second question "ENCLOSE THE PLOT AREA ? (Y/N)" will appear, and the user should type "N" to instruct the program not to enclose the plot area within a rectangle when the next plot image is created. After the user has answered the last question, the section prompt will again reappear. Having completed all of the changes to this menu, the user can leave the menu change mode simply by pressing the return key without entering any characters. A group of instructions will roll up onto the screen to assist the user in displaying the new menu, replotting the graph, or terminating the program (see figure A-9).

```
                MENU CHANGE

          SECTION : B
          ITEM    : 5
           4. UNITS BETWEEN X TICS   100
           5. UNITS BETWEEN X LABELS 1

          SECTION : B
          ITEM    : 1

          B. AXIS SPECIFICATIONS
           1. PLOT THE AXES ?  (Y/N) Y
           2. ENCLOSE THE PLOT AREA ? (Y/N) N

          SECTION : [user replies by hitting the RETURN key]

          DISPLAY NEW MENU - CTRL D
          OVERLAY NEW FILE - CTRL O
          REPLOT GRAPH     - CTRL P
          TERMINATE PROGRM - CTRL Q
```

Figure A-9: Example Menu Changes

After the user has changed the menu specifications, he may replot the data by typing "CTRL P". This command will erase the screen, and then construct the image from scratch. The new plot is shown in figure A-10. While the program is recreating the graph, the user may use the "CTRL G" and "CTRL T" commands to view the graphics and text pages, or he may terminate the program at any time by typing "CTRL Q". Once the program has recreated the graph, the user may change the menu items, and replot the image as many times as he wants.

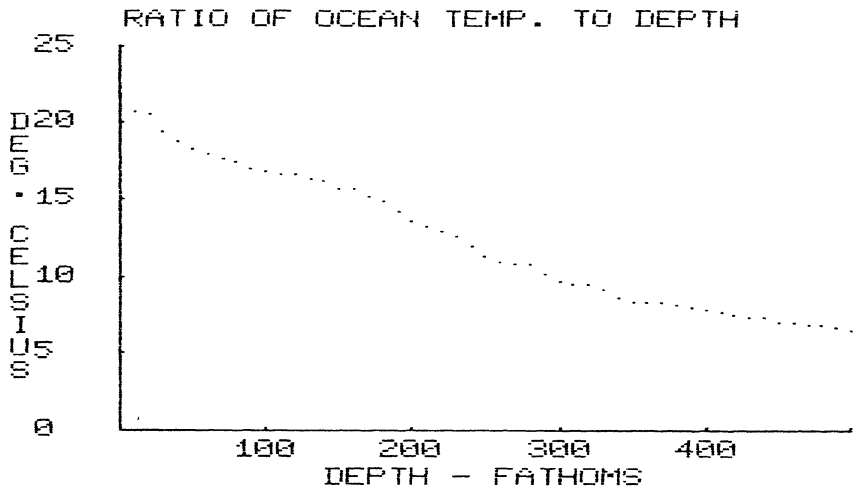


Figure A-10: Output from the Replot Example

Overlaying Values from Another File

Once the program has completed replotting the graph under the new menu specifications, our user would now like to overlay the contents of another file on top of the existing graph. These new values represent the ocean temperature against depth at another test site. The data is stored on file DISK2:TSTOCN2.TEXT, and is arranged in 4 columns, with the X values in column 4 and the Y values in column 2. The new points are to be represented by a line, and all of the points on the line are to be plotted. The overlay menu to implement these specifications appears in figure A-11, and the resulting graph appears in figure A-12.

```
                OVERLAY FILE

          == ENTER THE NAME OF THE SOURCE FILE ==
          == FILE NAME : DISK2:TSTOCN2.TEXT

                PLOT MENU

          A. PLOT SPECIFICATIONS
           1. NO. OF COLUMNS  4
           2. X IS COLUMN     4
           3. NO. OF Y COLUMNS (1 OR 2) 1
           4. Y IS COLUMN     2
           5. PLOT TYPE - 1 = . (POINT)
                          2 = -- (LINE) 2

         C. PLOT OPTIONS
          2. SKIP NUMBER OF POINTS 0
          3. PLOT NUMBER OF POINTS 0
```

Figure A-11: Example of Overlaying a File

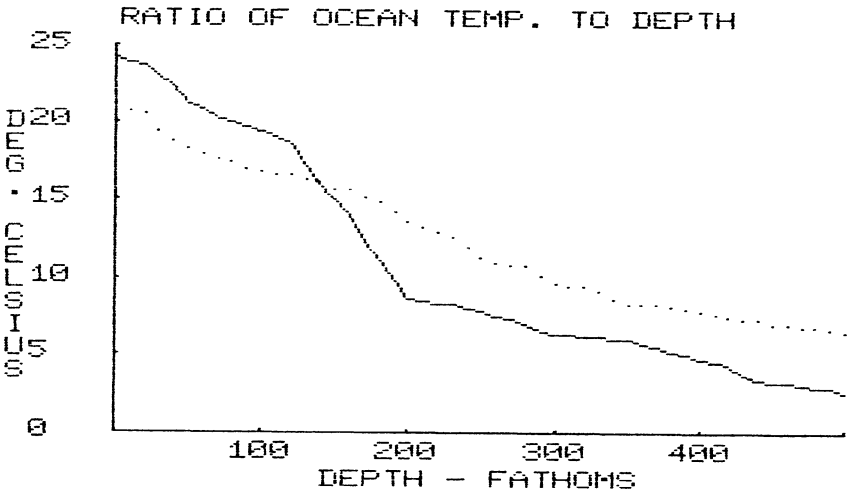


Figure A-12: Output from the Overlay Example

Hardcopy Dump of a Graph

Once the graph has been created to the user's satisfaction, he may dump the graphics image onto an Epson MX-80 printer, supported by a Grappler+ printer interface card, by typing "CTRL N". A prompt will appear asking the user to enter the Grappler+ printer interface commands to produce the form of output required. The Grappler+ commands may be found in the Grappler+ Printer Interface Operator's Manual, and they are also listed in Appendix C of this report.

Continuing with our example, our user now wants a emphasized image of his graph. The commands to produce this image appears in figure A-13.

GRAPHICS HARDCOPY

ENTER THE PRINT OPTIONS : GE

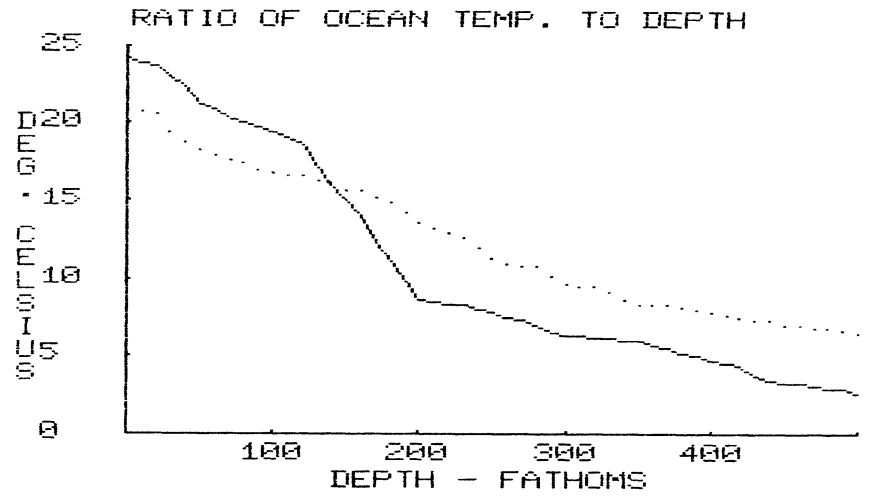


Figure A-13: Output from the Hardcopy Option

Saving the Graph on a Diskette File

Once the graph has been created, the user may store the graphics image on a diskette file by typing "CTRL L". A new screen will appear, and prompt the user to enter the file name which will contain the picture image. The user should reply to the prompt with a file name in the form "vol:file.FOTO". Once the user enters a file name acceptable to the system, the procedure will write the image to disk, a process which takes a few seconds. Once completed, the display will return to the graphics page.

Continuing with our example, our user now wants to save his work on the diskette file "DISK2:OCNPIC.FOTO". The screen showing the program's prompts and the user's replies appears in figure A-14.

```
== HIRES GRAPHICS STORAGE ==
== ENTER NAME : DISK2:OCNPIC.FOTO
```

Figure A-14: Example of Saving a Graphics Image

* * * * * *

Appendix B

PROGRAM COMMANDS

This appendix summarizes the user commands for the programs developed in this project, and is included for user reference. Further information concerning each command may be found in the appropriate chapters of this report.

TALK Program

The following instructions are available at anytime during execution of the TALK program. A full explaination of each command may be found in Chapter 2 of this report.

Key(s)	Command	Command Action
Left Arrow	Cursor Control	Moves the cursor one character position to the left.
Right Arrow	Cursor Control	Moves the cursor one character position to the right.
Control C	Return to the Program Menu	Exits from the present program option, and returns the user to the main menu of the program.
Control D	Delete Mode	Removes the character at the cursor's position from the screen and the keyboard buffer.
Control I	Insert Mode	Stores the characters from the present cursor position into the insertion buffer, and then erases those characters from the screen.
Control Q	Terminates the Insert Mode	Terminates the Insertion mode, and restores the characters held within the insertion buffer into the keyboard buffer, and onto the monitor's screen.
Escape	Break	Acts as a "Break" or "Attn" key.

Plot Program

The following commands are available at anytime during the execution of the PLOT program. A full explanation of each command may be found in Chapter 3 of this report.

Key(s)	Command	Command Action
Control G	Display the Graphics Page	Displays the Graphics page upon the monitor's screen.
Control H	Halt Plotting	Halts the active plotting of points, but remains within the Plot program.
Control Q	Quit	Terminates execution of the Plot program, and return control to the Pascal Operating System.
Control T	Display the Text Page	Displays the Text page upon the monitor's screen.

Once the program has stopped plotting, which occurs when the program has reached the end of the source file, or the user halts plotting by typing "CTRL H", the following commands become available to the user:

Key(s)	Command	Command Action
Control C	Change Menu	Allows the user to change any menu item.
Control D	Display Menu	Displays the current menu values.
Control L	Save the Picture Image	Saves the contents of the graphics buffer on a diskette file.

Key(s)	Command	Command Action
Control N	Hardcopy	Dumps the graphics image onto an Epson dot matrix printer.
Control O	Overlay File	Overlays the values from another source file onto an existing graph.
Control P	Replot Graph	Replots the values in the source file subject to the specifications of the Plot menu.

Appendix C

GRAPPLER+ INTERFACE COMMANDS

This appendix summarizes the Grappler+ printer interface commands used with the Hardcopy mode of the PLOT program.

The following set of Grappler+ interface commands is a list of those used most frequently within the Department of Surveying Engineering, however it is not a complete list of the commands recognized by the Grappler+. Readers who would like a complete list of the commands available should consult the Grappler+ Printer Interface Operator's Manual.

Command Summary

The Hardcopy mode of the PLOT program can easily dump high resolution graphics images to an Epson printer by specifying a few, simple Grappler+ commands. After the user has entered the Hardcopy mode, and the program prompt has appeared, the user may specify the desired printer options, which consist of the letter 'G' followed by any of the options described below. These printer options may be entered in any order, and the last one must be followed by a carriage return.

Command	Command Action
D	Print the Graphics screen double size. Since the Epson MX-80 does not have sufficient room to print the double size image horizontally, the user must also use the "R" option to rotate the image 90 degrees.
E	Print an "emphasized" image, in which the printer will print two closely spaced dots for every one it would normally print. The result is a denser image, but the printing time is twice as long.

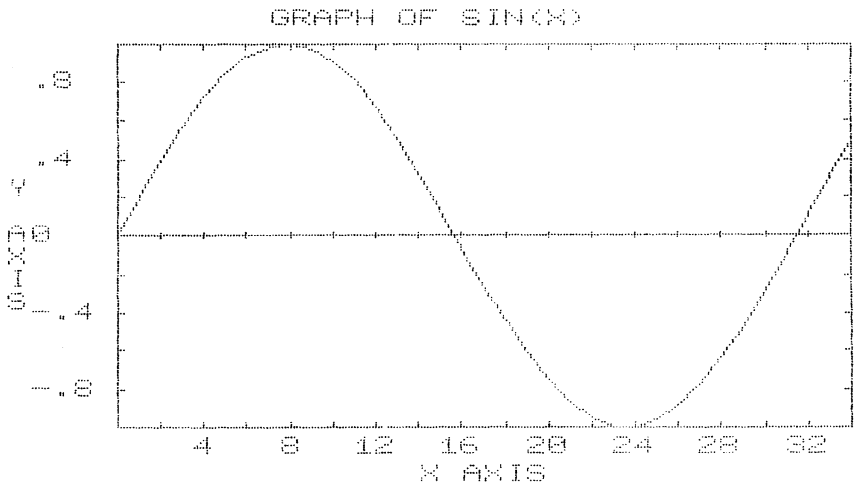
Command	Command Action
I	Invert the image before printing. Normally, every white dot on the screen is printed as a black dot on the paper however, this command will print the black portions of the screen as black on the paper, thus allowing the image to appear as it does on the screen.
R	Rotate the picture 90 degrees in a counterclockwise direction. The Epson MX-80 printer requires this option when printing an image double size.

Hardcopy Examples

The following examples illustrate the use of the Hardcopy mode in the PLOT program, and the Grappler+ printer interface commands.

```
GRAPHICS HARDCOPY
ENTER THE PRINT OPTIONS : G
```

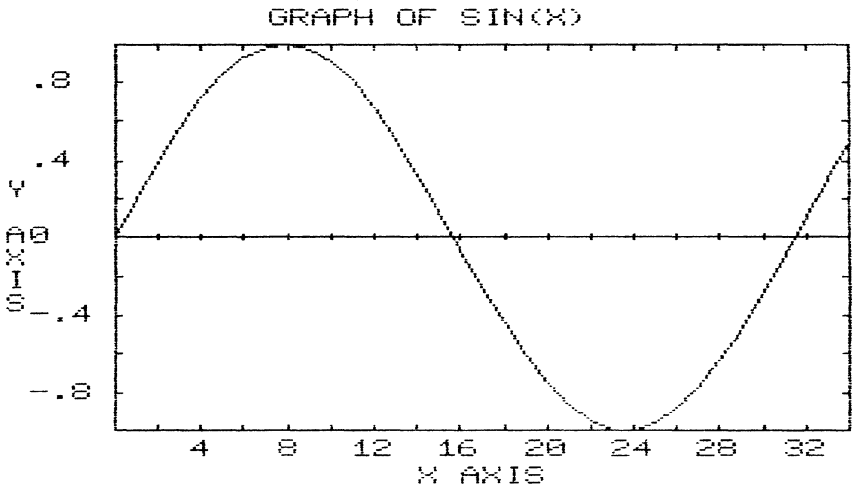
Figure C-1: Hardcopy Mode Example -- G



```
GRAPHICS HARDCOPY

ENTER THE PRINT OPTIONS : GE
```

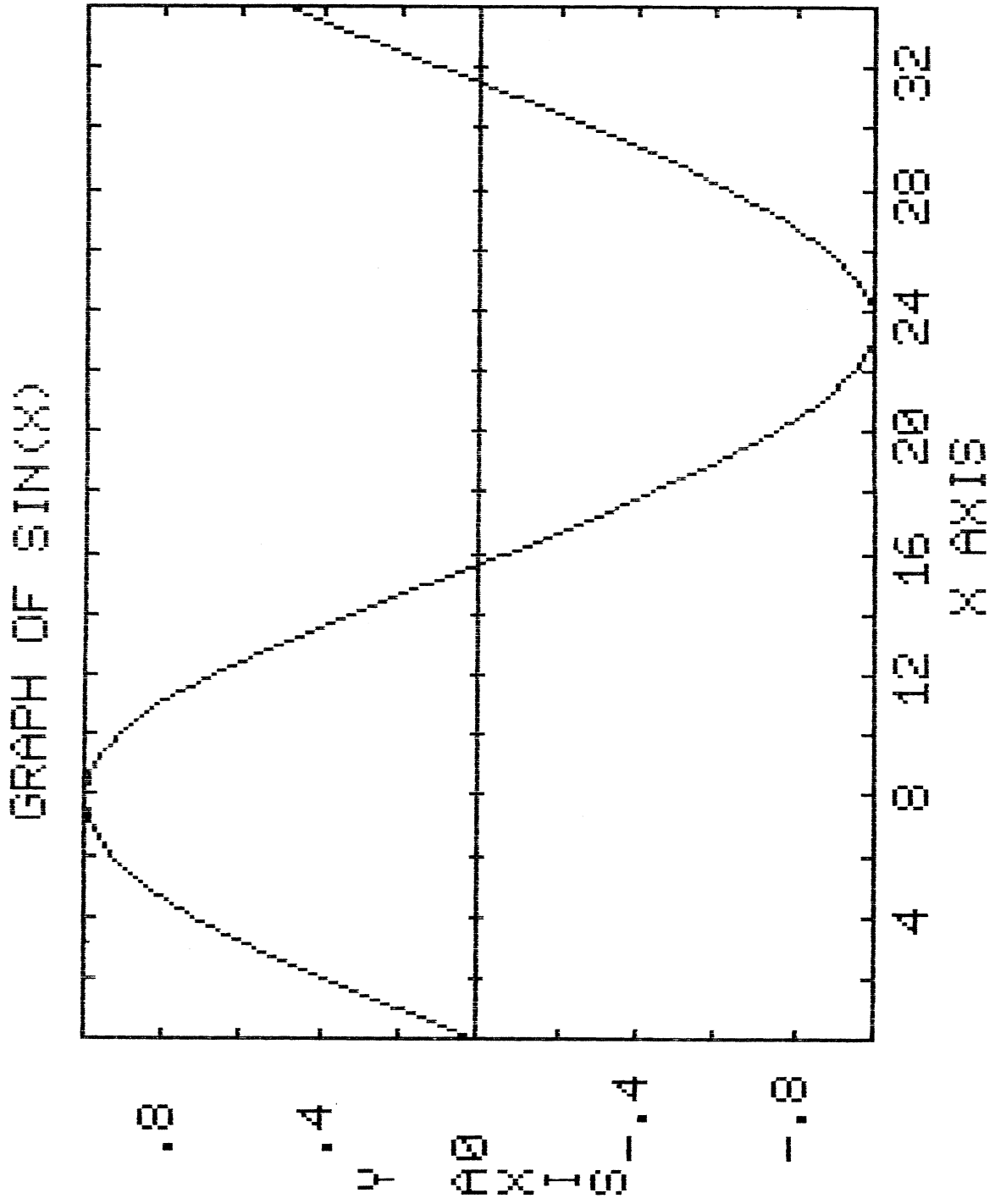
Figure C-2: Hardcopy Mode Example -- GE



```
GRAPHICS HARDCOPY

ENTER THE PRINT OPTIONS : GERD
```

Figure C-3: Hardcopy Mode Example -- GERD



Appendix D

PROGRAM LISTINGS

This appendix contains the source listings of the software developed for this project.

```
(*$S+*)
(*****************************************
 * PROGRAM : TALK                        *
 * WRITTEN :19-APR-82 BY MARK S LORD     *
 * MODIFIED: 9-JUL-82 BY LEONARD SLIPP   *
 * MODIFIED:20-DEC-82 BY SEE HEAN QUEK   *
 *---------------------------------------*
 * THIS PROGRAM ALLOWS COMMUNICATIONS    *
 * BETWEEN THE APPLE COMPUTER AND AN     *
 * OUTSIDE SOURCE, VIA THE SERIAL I/O    *
 * INTERFACE CARD IN APPLE SLOT #2.      *
 *                                       *
 * A)SCII TERMINAL MODE :                *
 *                                       *
 * T)RANSFER TEXT MODE :                 *
 *                                       *
 * C)OPY VSPC FILES / PROGRAM OUTPUT :   *
 *****************************************)

PROGRAM TALK;

USES  APPLESTUFF;

CONST ACIASTATUS =-16210; (* ADDRESS OF ACIA STATUS REG. *)
      ACIADATA   =-16209; (* ADDRESS OF ACIA DATA REG.   *)
      ACIABREAK  =    96; (* ACIA COMMAND FOR "BREAK"    *)
      ACIARESET  =     3; (* ACIA COMMAND FOR CHIP RESET *)
      ACIASPEED  =    82; (* ACIA SPEED SELECT = 300 BAUD*)
      BUFFLIMT   = 25000; (* MAX SIZE OF VSPC CHAR BUFFER*)
      ESC        =    27; (* ASCII CODE FOR <ESC> CHAR.  *)
      LINEFEED   =    10; (* ASCII CODE FOR <LF> CHAR.   *)
      LINELEN    =    80; (* MAX VSPC COMMAND LINE LENGTH*)
      LEFT       =     8; (* CODE FOR SPECIAL APPLE KEY  *)
      RIGHT      =    21; (* CODE FOR SPECIAL APPLE KEY  *)
      CTRLC      =     3; (* ASCII CODE FOR CONTROL-C    *)
      CTRLD      =     4; (* ASCII CODE FOR CONTROL-D    *)
      CTRLI      =     9; (* ASCII CODE FOR CONTROL-I    *)
      CTRLQ      =    17; (* ASCII CODE FOR CONTROL-Q    *)
      DC1        =    17; (* ASCII CODE FOR DC1 CHAR.    *)

TYPE  LONGSTRING = STRING[255];
      BYTE  = PACKED ARRAY [0..1] OF 0..255;
      DIRTY = RECORD
                CASE BOOLEAN OF
                  TRUE :(INT: INTEGER);
                  FALSE:(PTR: ^BYTE);
                END;

VAR   CMDSTR, STORE             :ARRAY [1..LINELEN] OF CHAR;
      QUITREQUESTED, ONEFILE    :BOOLEAN;
      NUMERIC                   :BOOLEAN;
      KBCHR                     :CHAR;
      KBVAL,REPLYVAL,CMDLEN     :INTEGER;
      CURSOR, SAVE, BUFFSIZE    :INTEGER;
```

```
      FILE1, IERCHAR            :INTEGER;
      CHARBUFF                  :PACKED ARRAY [1..BUFFLIMT] OF CHAR;
      TEXTFILE                  :TEXT;
      BLKS, FILENAME, VSPCNAME  :STRING[20];


FUNCTION PEEK(ADDR:INTEGER):INTEGER;
VAR  TRICK:DIRTY;
BEGIN
   TRICK.INT:=ADDR;
   PEEK:=TRICK.PTR^[0]
END; (* PEEK *)


PROCEDURE POKE(ADDR,DATA:INTEGER);
VAR  TRICK:DIRTY;
BEGIN
   DATA:=ABS(DATA MOD 256);
   TRICK.INT:=ADDR;
   TRICK.PTR^[0]:=DATA
END; (* POKE *)


PROCEDURE SCANACIA;
(****************************************
 * THIS ROUTINE SCANS THE ACIA FOR      *
 * INCOMING DATA.  IF DATA IS PRESENT,  *
 * IT IS DISPLAYED ON THE APPLE MONITOR*
 * AND THE ASCII NUMERIC VALUE IS       *
 * PLACED IN "REPLYCHR".  OTHERWISE,    *
 * "REPLYCHR" IS SET TO ZERO.           *
 ****************************************)
BEGIN (* SCANACIA *)
   IF ODD(PEEK(ACIASTATUS)) THEN
      BEGIN
         REPLYVAL:=PEEK(ACIADATA);
         WRITE(CHR(REPLYVAL))
      END
   ELSE
      REPLYVAL:=0;
END; (* SCANACIA *)


PROCEDURE SENDACIA(OUTVALUE:INTEGER);
(****************************************
 * THIS ROUTINE WILL TRANSMIT A BYTE    *
 * OUT THROUGH THE ACIA.  IT WAITS      *
 * UNTIL THE "READY" FLAG OF THE ACIA   *
 * IS SET, AND THEN TRANSFERS THE DATA  *
 * BYTE SPECIFIED BY ITS ASCII NUMERIC  *
 * VALUE IN "OUTVALUE".                 *
 ****************************************)
VAR STATUS:INTEGER;
BEGIN (* SENDACIA *)
```

```
   REPEAT
      STATUS:=PEEK(ACIASTATUS) DIV 2
   UNTIL ODD(STATUS);
   POKE(ACIADATA,OUTVALUE)
END; (* SENDACIA *)


PROCEDURE DELETION;
  VAR I : INTEGER;
  BEGIN
    IF CURSOR < (CMDLEN+1) THEN
      BEGIN
        FOR I := CURSOR TO (CMDLEN-1) DO
          BEGIN
            CMDSTR[I] := CMDSTR[I+1];
            WRITE (CMDSTR[I+1])
          END;
        WRITE(' ');
        FOR I := CMDLEN DOWNTO CURSOR DO
          WRITE (CHR(8));
        CMDLEN := CMDLEN - 1
      END
  END; (* DELETION *)


PROCEDURE INSERTION;
  VAR I,J : INTEGER;
  BEGIN
    IF CURSOR < (CMDLEN+1) THEN
      BEGIN
        I := 1;
        FOR J := CURSOR TO CMDLEN DO
          BEGIN
            STORE[I] := CMDSTR[J];
            WRITE(' ');
            I := I + 1
          END;
        I := I - 1;
        FOR J := 1 TO I DO
          WRITE (CHR(8));
        SAVE := I;
        CMDLEN := CMDLEN - SAVE
      END
  END; (* INSERTION *)


PROCEDURE TERMINSERT;
  VAR I : INTEGER;
  BEGIN
    IF CURSOR < (CMDLEN+1) THEN
      BEGIN
        FOR I := CURSOR TO CMDLEN DO
          BEGIN
            WRITE (CHR(28));
```

```
            CURSOR := CURSOR + 1
          END
      END;
    FOR I := 1 TO SAVE DO
      BEGIN
        WRITE (STORE[I]);
        CMDSTR[CURSOR] := STORE[I];
        CURSOR := CURSOR + 1
      END;
    CMDLEN := CMDLEN + SAVE;
    SAVE := 0
  END; (* TERMINSERT *)


PROCEDURE PROCESSCOMMAND; FORWARD;


PROCEDURE SCANKEYBOARD;
(*************************************
 * THIS ROUTINE CHECKS TO SEE IF ANY   *
 * MORE KEYBOARD INPUT HAS BEEN ENTERED*
 * BY THE USER.  IF SO, IT IS PROCESSED*
 * AS DESCRIBED AT THE TOP OF THIS     *
 * PROGRAM IN THE D)DUMB TERMINAL CMD. *
 *************************************)
VAR I : INTEGER;
BEGIN (* SCANKEYBOARD *)
   IF KEYPRESS THEN
     BEGIN
       READ(KEYBOARD,KBCHR);
       IF EOLN(KEYBOARD) THEN
         BEGIN
           IF SAVE <> 0 THEN
             TERMINSERT;
           KBCHR := CHR(13);
           WRITE(KBCHR);
           CMDSTR[CMDLEN+1] := KBCHR;
           I := 0;
           REPEAT
             I := I + 1;
             SENDACIA(ORD(CMDSTR[I]));
             SCANACIA
           UNTIL CMDSTR[I] = CHR(13);
           CURSOR := 1;
           CMDLEN := 0;
           SAVE   := 0
         END
       ELSE
         BEGIN
           KBVAL:=ORD(KBCHR);
           IF KBVAL IN [ESC,CTRLC,LEFT,RIGHT,CTRLD,CTRLI,CTRLQ] THEN
              CASE KBVAL OF
                ESC:
                  BEGIN
```

```
                    POKE(ACIASTATUS,ACIABREAK);
                    NOTE(40,25);
                    POKE(ACIASTATUS,ACIASPEED)
                  END;
                CTRLC:
                  BEGIN
                    WRITELN(CHR(7),'<CTRL-C>');
                    EXIT(PROCESSCOMMAND)
                  END;
                LEFT:
                  BEGIN
                    IF CURSOR > 1 THEN
                      BEGIN
                        CURSOR := CURSOR - 1;
                        WRITE (KBCHR)
                      END
                    ELSE
                      NOTE (40, 25)
                  END;
                RIGHT:
                  BEGIN
                    IF CURSOR <= CMDLEN THEN
                      BEGIN
                        CURSOR := CURSOR + 1;
                        WRITE (CHR(28))
                      END
                    ELSE
                      NOTE (40, 25)
                  END;
                CTRLD: DELETION;
                CTRLI: INSERTION;
                CTRLQ: TERMINSERT
              END (* CASE *)
            ELSE
              BEGIN
                IF (CMDLEN+SAVE) < (LINELEN-1) THEN
                  BEGIN
                    WRITE(KBCHR);
                    CMDSTR[CURSOR] := KBCHR;
                    IF CURSOR = (CMDLEN+1) THEN
                      CMDLEN := CMDLEN + 1;
                    CURSOR := CURSOR + 1
                  END
                ELSE
                  NOTE (20,20)
              END
          END
      END;
END; (* SCANKEYBOARD *)


PROCEDURE ASCIITERMINAL;
(*************************************
 * THIS ROUTINE ALLOWS DIRECT USER     *
```

```
 * COMMUNICATIONS WITH A REMOTE DEVICE *
 * BY CAUSING THE APPLE TO BEHAVE AS A *
 * SEMIINTELLIGENT ASYNC ASCII TERMINAL*
 *************************************)
BEGIN
   WRITELN('== ASCII TERMINAL MODE ==');
   WRITELN;
   WRITELN('== HIT <CTRL-C> TO QUIT ==');
   WRITELN(CHR(7));
   CMDLEN := 0;
   KBVAL  := 0;
   SAVE   := 0;
   CURSOR := 1;
   REPEAT
      SCANACIA;
      SCANKEYBOARD
   UNTIL KBVAL=CTRLC;
END; (* ASCIITERMINAL *)


PROCEDURE XMITVSPC(MESSAGE:LONGSTRING);
(*************************************
 * THIS ROUTINE USES "SENDACIA" TO    *
 * TRANSMIT A LINE OF CHARACTERS TO   *
 * VSPC.  A CARRIAGE-RETURN IS SENT AT *
 * THE END OF THE LINE, AND ALL CHARS  *
 * SENT ARE ALSO ECHOED ON THE APPLE'S *
 * MONITOR AS THEY ARE TRANSMITTED.    *
 *************************************)
VAR I:INTEGER;
BEGIN
   MESSAGE:=CONCAT(MESSAGE,' ');
   MESSAGE[LENGTH(MESSAGE)]:=CHR(13);
   I:=0;
   REPEAT
      I:=I+1;
      SCANKEYBOARD;
      SCANACIA;
      WRITE(MESSAGE[I]);
      SENDACIA(ORD(MESSAGE[I]))
   UNTIL MESSAGE[I]=CHR(13);
   REPEAT
      SCANKEYBOARD;
      SCANACIA
   UNTIL REPLYVAL=DC1;
END; (* XMITVSPC *)


PROCEDURE TEXTTRANSFER;
(*************************************
 * THIS ROUTINE HANDLES TRANSFERING OF *
 * NORMAL TEXT FILES TO VSPC. THE USER *
 * IS PROMPTED FOR A FILE SPECIFICATION*
 * IN WHICH THE ".TEXT" IS OPTIONAL,   *
```

```
 * AND THEN PROCEEDS TO TRANSFER THE   *
 * FILE TO A USER-SPECIFIED VSPC WS.   *
 ***************************************)
VAR TEXTLINE:LONGSTRING;
    IOERR:INTEGER;
BEGIN
   WRITELN('== PROCEDURE TO TRANSFER TEXT FILES ==');
   WRITELN;
   WRITELN('== ENTER NAME OF DISKETTE FILE ==');
   REPEAT
      WRITE('== FILE NAME ==> ');
      READLN(FILENAME);
      IF LENGTH(FILENAME)=0 THEN
         EXIT(TEXTTRANSFER);
      (*$I-*)
      RESET(TEXTFILE,FILENAME);
      IOERR:=IORESULT;
      IF IOERR=10 THEN
         BEGIN
            INSERT('.TEXT',FILENAME,(1+LENGTH(FILENAME)));
            RESET(TEXTFILE,FILENAME);
            IOERR:=IORESULT
         END;
      (*$I+*)
      IF IOERR<>0 THEN
         BEGIN
            IF IOERR=10 THEN
               WRITELN('FILE NOT FOUND - RE-ENTER')
            ELSE
               WRITELN('OPEN ERROR #',IOERR,' - RE-ENTER')
         END
   UNTIL IOERR=0;
   WRITELN;
   WRITELN('== ENTER NAME OF VSPC FILE ==');
   WRITE('== FILE NAME ==> ');
   READLN(VSPCNAME);
   IF 0=LENGTH(VSPCNAME) THEN
      BEGIN
         CLOSE(TEXTFILE);
         EXIT(TEXTTRANSFER)
      END;
   XMITVSPC('TAPE');
   XMITVSPC('');
   XMITVSPC('CLEAR');
   XMITVSPC(CONCAT('NAME ',VSPCNAME));
   XMITVSPC('INPUT 1 1');
   IF NOT EOF(TEXTFILE) THEN
      REPEAT
         READLN(TEXTFILE,TEXTLINE);
         IF LENGTH(TEXTLINE)=0 THEN
            TEXTLINE:=' ';
         XMITVSPC(TEXTLINE)
      UNTIL EOF(TEXTFILE);
   XMITVSPC('');
```

```
   XMITVSPC(CONCAT('SAVE ',VSPCNAME));
   CLOSE(TEXTFILE);
END; (* TEXTTRANSFER *)


PROCEDURE DUMP (NAME : STRING;
                START, STOP : INTEGER);
VAR
  I, IOERR : INTEGER;
  SUFFIX   : STRING[1];
  TEXTNAME : STRING[40];
BEGIN
  REPEAT
    WRITE('     RECEIVING FILE : ');
    IF ONEFILE THEN
      SUFFIX := ''
    ELSE
      IF 0=LENGTH(NAME) THEN
        SUFFIX :='1'
      ELSE
        SUFFIX := '2';
    IF 0=LENGTH(NAME) THEN
      BEGIN
        READLN(FILENAME);
        IF 0=LENGTH(FILENAME) THEN
          EXIT(PROCESSCOMMAND);
        I := POS('.TEXT',FILENAME);
        IF I<>0 THEN
          FILENAME := COPY(FILENAME,1,I-1)
      END
    ELSE
      BEGIN
        WRITELN(NAME,SUFFIX,'.TEXT');
        FILENAME := NAME
      END;
    TEXTNAME := CONCAT(FILENAME,SUFFIX,'.TEXT[',BLKS,']');
    (*$I-*)
    REWRITE(TEXTFILE,TEXTNAME);
    IOERR := IORESULT;
    (*$I+*)
    WRITELN;
    IF IOERR<>0 THEN
      BEGIN
        WRITE(CHR(7));
        WRITE('       ');
        IF IOERR=8 THEN
          WRITELN('NO ROOM ON DISK')
        ELSE
          WRITELN('I/O ERROR #',IOERR);
        WRITELN
      END;
  UNTIL IOERR=0;
  FOR I := START TO STOP DO
    WRITE (TEXTFILE, CHARBUFF[I]);
```

```
  WRITELN (TEXTFILE);
  CLOSE (TEXTFILE, LOCK)
END; (* DUMP *)


PROCEDURE COPYWS1;
(****************************************
 * THIS ROUTINE COPIES A BLOCK OF DATA *
 * FROM A USER SPECIFIED VSPC WORKSPACE*
 * OR THE OUTPUT FROM A FORTRAN PROGRAM*
 * AND STORES  IT ON A DISKETTE.  THE  *
 * BLOCK SIZE  IS DEPENDENT  UPON  THE *
 * "BUFFLIMT" CONSTANT DECLARED AT THE *
 * BEGINNING OF THIS PROGRAM.          *
 ****************************************)
VAR MESSAGE :STRING;
    IOERR,I :INTEGER;
BEGIN
   WRITELN('== PROCEDURE TO COPY IBM FILES ==');
   WRITELN;
   WRITELN('== SPECIFY VALID CHARACTER SET ==');
   WRITELN;
   WRITELN('    1. - ASCII CHARACTERS');
   WRITELN('    2. - NUMERIC TYPE ONLY');
   WRITELN;
   WRITE('== ENTER 1 OR 2 : ');
   REPEAT
     READ (KEYBOARD,KBCHR)
   UNTIL KBCHR IN ['1','2'];
   WRITELN(KBCHR);
   IF KBCHR = '1' THEN
     NUMERIC := FALSE
   ELSE
     NUMERIC := TRUE;
   WRITELN;
   WRITELN('== SELECT 1 OF THE FOLLOWING OPTIONS ==');
   WRITELN;
   WRITELN('    1. - COPY VSPC FILE CONTENTS');
   WRITELN('    2. - RUN A VS FORTRAN PROGRAM');
   WRITELN;
   WRITE('== ENTER 1 OR 2 : ');
   REPEAT
     READ(KEYBOARD, KBCHR)
   UNTIL KBCHR IN ['1','2'];
   WRITELN(KBCHR);
   WRITELN;
   IF KBCHR = '1' THEN
     BEGIN
       WRITELN('== ENTER THE VSPC FILE NAME ==');
       WRITE('== FILE NAME ==> ');
       READLN(VSPCNAME);
       IF 0=LENGTH(VSPCNAME) THEN
         EXIT (PROCESSCOMMAND);
       PAGE(OUTPUT);
```

```
        XMITVSPC('TAPE');
        XMITVSPC('');
        XMITVSPC(CONCAT('LOAD ',VSPCNAME));
        MESSAGE:='LIST NOLINE '
      END
    ELSE
      BEGIN
        WRITELN('== ENTER THE FORTRAN PROGRAM''S NAME');
        WRITE('== PROGRAM NAME ==> ');
        READLN (VSPCNAME);
        IF 0=LENGTH(VSPCNAME) THEN
          EXIT (PROCESSCOMMAND);
        PAGE(OUTPUT);
        XMITVSPC('TAPE');
        XMITVSPC('');
        MESSAGE := CONCAT('RUN ',VSPCNAME,' ')
      END;
    MESSAGE[LENGTH(MESSAGE)] := CHR(13);
    I:=0;
    REPEAT
       I:=I+1;
       SCANKEYBOARD;
       SCANACIA;
       WRITE(MESSAGE[I]);
       SENDACIA(ORD(MESSAGE[I]))
    UNTIL MESSAGE[I] = CHR(13)
  END; (* COPYWS1 *)


PROCEDURE COPYWS2;
VAR LASTCR :BOOLEAN;
    I      :INTEGER;
BEGIN
   I:=1; IERCHAR := 0;
   SCANACIA;
   WHILE (REPLYVAL <> DC1) AND (I <= BUFFLIMT) DO
     BEGIN
       IF REPLYVAL <> 0 THEN
         BEGIN
           REPLYVAL := REPLYVAL MOD 128;
           IF NUMERIC THEN
             IF (REPLYVAL IN [13,32,43,45,46,48..57,69]) THEN
               BEGIN
                 CHARBUFF[I] := CHR(REPLYVAL);
                 IF (I<13000) AND (REPLYVAL=13) THEN
                   FILE1 := I;
                 I := I + 1
               END
             ELSE
               IERCHAR := IERCHAR + 1
           ELSE
             IF NOT (REPLYVAL IN [10,16,19]) THEN
               BEGIN
                 CHARBUFF[I] := CHR(REPLYVAL);
```

```
                IF (I<13000) AND (REPLYVAL=13) THEN
                  FILE1 := I;
                I := I + 1
              END
            ELSE
              IERCHAR := IERCHAR + 1
        END;
      SCANKEYBOARD;
      SCANACIA
    END;
  IF I > BUFFLIMT THEN
    BEGIN
      WRITELN(' *** BUFFER FULL ***');
      WRITE(CHR(7));
      LASTCR := FALSE;
      I := BUFFLIMT;
      REPEAT
        I := I - 1;
        IF(ORD(CHARBUFF[I])=13) THEN
          LASTCR := TRUE;
      UNTIL LASTCR;
    END;
  BUFFSIZE := I - 1;
  PAGE(OUTPUT)
END; (* COPYWS2 *)


PROCEDURE COPYWS3;
VAR I1, I2 : INTEGER;
BEGIN
  IF BUFFSIZE <= 16000 THEN
    BEGIN
      I1 := 2*ROUND(1.5+BUFFSIZE/1024);
      ONEFILE := TRUE
    END
  ELSE
    BEGIN
      I1 := 2*ROUND(1.5+FILE1/1024);
      I2 := 2*ROUND(1.5+(BUFFSIZE-FILE1-1)/1024);
      ONEFILE := FALSE
    END;
  GOTOXY (10,1);WRITELN('== COPY IBM FILES ==');
  GOTOXY (0,4); WRITELN('1. IBM --> MEMORY');
  GOTOXY (31,4);
  IF NUMERIC THEN
    WRITELN('NUMERIC')
  ELSE
    WRITELN('ASCII');
  GOTOXY (5,6); WRITELN('SOURCE FILE :  ',VSPCNAME);
  GOTOXY (5,8); WRITELN(IERCHAR:5,' INVALID CHARACTERS DETECTED');
  GOTOXY (5,10);WRITELN(BUFFSIZE:5,' CHARACTERS WITHIN BUFFER');
  GOTOXY (0,13);WRITELN('2. MEMORY --> DISK');
  IF ONEFILE THEN
    BEGIN
```

```
        GOTOXY (5,15);WRITELN('ESTIMATED BLOCK REQUIREMENT',I1:4);
        WRITELN; STR(I1,BLKS);
        DUMP('',1,BUFFSIZE)
      END
    ELSE
      BEGIN
        GOTOXY (30,13); WRITELN('2 FILES');
        GOTOXY (4,15);WRITELN('-FILE #1 : BLOCK REQUIREMENT',I1:5);
        WRITELN; STR (I1,BLKS);
        DUMP('',1,FILE1);
        WRITELN;
        WRITELN('    -FILE #2 : BLOCK REQUIREMENT',I2:5);
        WRITELN; STR (I2,BLKS);
        DUMP (FILENAME,FILE1+1,BUFFSIZE)
      END
END; (* COPYWS3 *)


PROCEDURE PROCESSCOMMAND;
(****************************************
 * THIS ROUTINE SERVES AS A COMMON      *
 * INTERFACE BETWEEN THE MAIN PROGRAM   *
 * AND THE COMMAND-PROCESSING PROC'S.   *
 * IT'S PRESENCE IS REQUIRED IN ORDER   *
 * TO ALLOW SCANKEYBOARD TO HAVE A      *
 * COMMON EXIT POINT FOR HANDLING A     *
 * USER <CTRL¬C> COMMAND.               *
 ****************************************)
BEGIN (* PROCESSCOMMAND *)
   CASE KBCHR OF
      'Q':QUITREQUESTED:=TRUE;
      'A':ASCIITERMINAL;
      'C':BEGIN
            COPYWS1;
            COPYWS2;
            COPYWS3
          END;
      'T':TEXTTRANSFER;
   END; (* CASE *)
END; (* PROCESSCOMMAND *)


(****************************************
 * THE MAIN ROUTINE (BELOW) HANDLES     *
 * GENERAL INTIALIZATION AND THE        *
 * PROMPTING FOR, AND INPUT OF, USER    *
 * COMMAND OPTIONS FROM ITS MAIN MENU.  *
 ****************************************)
BEGIN (* TALK *)
   QUITREQUESTED:=FALSE;
   POKE(ACIASTATUS,ACIARESET);
   POKE(ACIASTATUS,ACIASPEED);
   REPEAT
      PAGE(OUTPUT);
```

```
      WRITELN('== SERIAL COMMUNICATIONS PROGRAM ==');
      WRITELN;
      WRITELN('== COMMAND MENU ==');
      WRITELN;
      WRITELN('OPTIONS ARE :');
      WRITELN('    A = ASCII TERMINAL MODE');
      WRITELN('    C = COPY VSPC FILES / PROGRAM OUTPUT');
      WRITELN('    T = TRANSFER ANY TEXT FILE');
      WRITELN('    Q = QUIT');
      WRITELN;
      WRITE('== ENTER COMMAND ==> ');
      REPEAT
        WRITE(CHR(7));
        READ(KEYBOARD,KBCHR)
      UNTIL KBCHR IN ['A','C','T','Q'];
      PAGE(OUTPUT);
      PROCESSCOMMAND;
      WRITELN;
      WRITELN;
   UNTIL QUITREQUESTED;
   PAGE(OUTPUT)
END.
```

```
(*$S+*)
PROGRAM PLOT;

  USES APPLESTUFF, TRANSCEND,
       STRINGSTUF, UOUT, APPLEGRAPHICS;

  CONST
     CTRLC = 3; CTRLD = 4;
     CTRLG = 7; CTRLH = 8;
     CTRLL = 12; CTRLN = 14;
     CTRLO = 15; CTRLP = 16;
     CTRLQ = 17; CTRLT = 20;
     HIRES1 = 8192; MAXCOL  =  10;

  TYPE
    DISKBLOCK = RECORD
                  CASE BOOLEAN OF
                    TRUE  :(INTPART : INTEGER);
                    FALSE :(PTRPART : ¬INTEGER);
                  END;

  VAR
    AXIS, BOX, FINISHPLT  : BOOLEAN;
    ALPH, BAD, QUITREQ, REPLT  : BOOLEAN;
    IOERR,NUMCOL,XCOL,YPT,YLN,YCOL: INTEGER;
    XLABEL, YLABEL,PTSKIP, NUMPTS : INTEGER;
    PLOTPTR  : PORTPOINTER;
    XMIN, XMAX, YMIN, YMAX   : REAL;
    XLIM, YLIM, XTIC, YTIC   : REAL;
    PT : ARRAY [1..MAXCOL] OF REAL;
    TEXTNAME, TITLE, XNAME, YNAME : STRING;
    FILECHR, CH, NUMY, PLT        : CHAR;
    TEXTFILE, PRINT : TEXT;


  FUNCTION DIGITS (NUM:REAL) : INTEGER;
    VAR SUM, NUB : INTEGER;
    BEGIN
      IF NUM = 0 THEN
        BEGIN
          DIGITS := -3;
          EXIT (DIGITS)
        END;
      IF NUM < 0 THEN
        BEGIN
          SUM := 1;
          NUM := ABS (NUM)
        END
      ELSE
        SUM := 0;
      IF NUM < 1.0 THEN
        BEGIN
          DIGITS := -1 * (SUM*7) - 3;
          EXIT (DIGITS)
```

```
      END;
    SUM := SUM + TRUNC(LOG(NUM)+1);
    NUB := SUM DIV 2;
    IF ODD(SUM) THEN
        DIGITS := -1 * (NUB*7) - 3
    ELSE
        DIGITS := -1 * (NUB * 7)
  END; (* DIGITS *)


PROCEDURE OPENFILE;
  BEGIN
    WRITELN('== ENTER THE NAME OF THE SOURCE FILE ==');
    REPEAT
      WRITE('== FILE NAME : ');
      READLN(TEXTNAME);
      IF LENGTH(TEXTNAME) = 0 THEN
        EXIT(PLOT);
      (*$I-*)
      RESET(TEXTFILE,TEXTNAME);
      IOERR := IORESULT;
      (*$I+*)
      IF IOERR <> 0 THEN
        WRITELN('I/O ERROR #',IOERR,' : RE-ENTER')
    UNTIL IOERR = 0
  END; (* OPENFILE *)


PROCEDURE MENU1;
  BEGIN
    WRITELN;
    WRITELN('      PLOT MENU');
    WRITELN;
    WRITELN('A. PLOT SPECIFICATIONS');
    WRITE(' 1. NO. OF COLUMNS  ');
    REPEAT
      READLN(NUMCOL);
      IF NUMCOL < 2 THEN
        WRITE('ERROR - MUST BE >= TO 2 : ');
      IF NUMCOL > MAXCOL THEN
        WRITE('ERROR - TOO MANY COLUMNS : ')
    UNTIL (NUMCOL <= MAXCOL) AND (NUMCOL >= 2)
  END; (* MENU1 *)


PROCEDURE MENU2;
  BEGIN
    WRITE(' 2. X IS COLUMN     ');
    REPEAT
      READLN(XCOL);
      IF (XCOL < 1) OR (XCOL > NUMCOL) THEN
        WRITE('MUST BE BETWEEN 1..',NUMCOL,' : ')
    UNTIL (XCOL >= 1) AND (XCOL <= NUMCOL)
  END; (* MENU2 *)
```

```
PROCEDURE MENU3;
  BEGIN
    WRITE(' 3. NO. OF Y COLUMNS (1 OR 2) ');
    REPEAT
      READLN(NUMY)
    UNTIL NUMY IN ['1','2'];
    IF NUMY = '1' THEN
      BEGIN
        WRITE(' 4. Y IS COLUMN     ');
        REPEAT
          READLN(YCOL);
          IF (YCOL < 1) OR (YCOL > NUMCOL) THEN
            WRITE('MUST BE BETWEEN 1..',NUMCOL,' : ')
        UNTIL (YCOL >= 1) AND (YCOL <= NUMCOL);
        WRITELN(' 5. PLOT TYPE - 1 = . (POINT) ');
        WRITE('                  2 = -- (LINE) ');
        READLN(PLT);
        IF PLT = '1' THEN
          BEGIN
            YPT := YCOL;
            YLN := 0
          END
        ELSE
          BEGIN
            PLT := '2';
            YPT := 0;
            YLN := YCOL
          END
      END
    ELSE
      BEGIN
        WRITE(' 4. Y COLUMN - POINT PLOT : ');
        REPEAT
          READLN(YPT);
          IF (YPT < 1) OR (YPT > NUMCOL) THEN
            WRITE ('MUST BE BETWEEN 1..',NUMCOL,' : ')
        UNTIL (YPT >= 1) AND (YPT <= NUMCOL);
        WRITE(' 5. Y COLUMN -  LINE PLOT : ');
        REPEAT
          READLN(YLN);
          IF (YLN < 1) OR (YLN > NUMCOL) THEN
            WRITE ('MUST BE BETWEEN 1..',NUMCOL,' : ')
        UNTIL (YLN >= 1) AND (YLN <= NUMCOL)
      END
  END; (* MENU3 *)


PROCEDURE MENU4;
  BEGIN
    WRITE(' 6. MIN X = ');
    READLN(XMIN);
    WRITE(' 7. MAX X = ');
```

```
    REPEAT
      READLN(XMAX);
      IF XMAX <= XMIN THEN
        WRITE('MAX X MUST BE > THAN MIN X : ');
    UNTIL XMAX > XMIN;
    XLIM := 0;
    IF XMIN > 0 THEN
      XLIM := XMIN;
    IF XMAX < 0 THEN
      XLIM := XMAX
  END; (* MENU4 *)


PROCEDURE MENU5;
  BEGIN
    WRITE(' 8. MIN Y = ');
    READLN(YMIN);
    WRITE(' 9. MAX Y = ');
    REPEAT
      READLN(YMAX);
      IF YMAX <= YMIN THEN
        WRITE('MAX Y MUST BE > THAN MIN Y : ')
    UNTIL YMAX > YMIN;
    YLIM := 0;
    IF YMIN > 0 THEN
      YLIM := YMIN;
    IF YMAX < 0 THEN
      YLIM := YMAX
  END; (* MENU5 *)


PROCEDURE MENU6;
  BEGIN
    WRITELN;
    WRITELN('B. AXIS SPECIFICATIONS');
    WRITE(' 1. PLOT THE AXIS ?  (Y/N) ');
    READLN(CH);
    IF CH = 'Y' THEN
      AXIS := TRUE
    ELSE
      AXIS := FALSE;
    WRITE(' 2. ENCLOSE THE PLOT AREA ? (Y/N) ');
    READLN(CH);
    IF CH = 'Y' THEN
      BOX  := TRUE
    ELSE
      BOX  := FALSE
  END; (* MENU6 *)


PROCEDURE MENU7;
  BEGIN
    IF BOX OR AXIS THEN
      BEGIN
```

```
        WRITE(' 3. NAME OF X AXIS : ');
        READLN(XNAME)
      END
  END; (* MENU7 *)


PROCEDURE MENU8;
  BEGIN
    IF BOX OR AXIS THEN
      BEGIN
        WRITE(' 4. UNITS BETWEEN X TICS   ');
        READLN(XTIC);
        XTIC   := ABS (XTIC);
        WRITE(' 5. UNITS BETWEEN X LABELS ');
        READLN(XLABEL);
        XLABEL := ABS (XLABEL)
      END
  END; (* MENU8 *)


PROCEDURE MENU9;
  BEGIN
    IF BOX OR AXIS THEN
      BEGIN
        WRITE(' 6. NAME OF Y AXIS : ');
        READLN(YNAME)
      END
  END; (* MENU9 *)


PROCEDURE MENU10;
  BEGIN
    IF BOX OR AXIS THEN
      BEGIN
        WRITE(' 7. UNITS BETWEEN Y TICS   ');
        READLN(YTIC);
        YTIC   := ABS (YTIC);
        WRITE(' 8. UNITS BETWEEN Y LABELS ');
        READLN(YLABEL);
        YLABEL := ABS (YLABEL)
      END
  END; (* MENU10 *)


PROCEDURE MENU11;
  BEGIN
    WRITELN;
    WRITELN('C. PLOT OPTIONS');
    WRITE(' 1. PLOT TITLE : ');
    READLN(TITLE)
  END; (* MENU11 *)


PROCEDURE MENU12;
```

```
    BEGIN
      WRITE(' 2. SKIP NUMBER OF POINTS ');
      READLN(PTSKIP);
      PTSKIP := ABS (PTSKIP);
      WRITE(' 3. PLOT NUMBER OF POINTS ');
      READLN(NUMPTS);
      NUMPTS := ABS (NUMPTS);
      IF NUMPTS = 0 THEN
        NUMPTS := MAXINT
    END; (* MENU12 *)


  PROCEDURE MENUPRT1;
    BEGIN
      TEXTMODE;
      PAGE(OUTPUT);
      WRITELN('FILE : ',TEXTNAME);
      WRITELN;
      WRITELN('      PLOT MENU');
      WRITELN;
      WRITELN('A. PLOT SPECIFICATIONS');
      WRITELN(' 1. NO. COLS = ',NUMCOL,'  2. X COL = ',XCOL);
      WRITELN(' 3. NO. Y COLS = ',NUMY);
      IF NUMY = '1' THEN
        WRITELN(' 4. Y COL = ',YCOL,'  5. PLOT TYPE = ',PLT)
      ELSE
        WRITELN(' 4. Y COL PT = ',YPT,'  5. Y COL LN = ',YLN);
      WRITELN(' 6. MIN X = ',XMIN,'  7. MAX X = ',XMAX);
      WRITELN(' 8. MIN Y = ',YMIN,'  9. MAX Y = ',YMAX);
      WRITELN
    END; (* MENUPRT1 *)


  PROCEDURE MENUPRT2;
    BEGIN
      WRITELN('B. AXIS SPECIFICATIONS');
      WRITE(' 1. AXIS ');
      IF AXIS THEN
        WRITE('Y')
      ELSE
        WRITE('N');
      WRITE('   2. ENCLOSED ');
      IF BOX THEN
        WRITELN('Y')
      ELSE
        WRITELN('N');
      IF AXIS OR BOX THEN BEGIN
        WRITELN(' 3. X AXIS : ',XNAME);
        WRITELN(' 4. X TIC = ',XTIC,'  5. X LABEL = ',XLABEL);
        WRITELN(' 6. Y AXIS : ',YNAME);
        WRITELN(' 7. Y TIC = ',YTIC,'  8. Y LABEL = ',YLABEL)
      END;
      WRITELN;
      WRITELN('C. PLOT OPTIONS');
```

```
    WRITELN(' 1. PLOT : ',TITLE);
    WRITELN(' 2. SKIP = ',PTSKIP,'  3. PLOT = ',NUMPTS);
    WRITELN;
    WRITELN('CHANGE MENU   - CTRL C');
    WRITELN('OVERLAY FILE  - CTRL O');
    WRITELN('TERMINATE PGM - CTRL Q');
    WRITELN
  END; (* MENUPRT2 *)


PROCEDURE CHNGMENU;
  VAR SECTION, ITEM     : CHAR;
      QUITCHNG, BADITEM : BOOLEAN;
  BEGIN
    TEXTMODE;
    WRITELN('       MENU CHANGE');
    QUITCHNG := FALSE;
    REPEAT
      BADITEM := FALSE;
      WRITELN;
      WRITE('SECTION : ');
      READLN(SECTION);
      IF NOT(SECTION IN ['A'..'C']) THEN
        QUITCHNG := TRUE;
      IF NOT QUITCHNG THEN
        BEGIN
          WRITE('ITEM    : ');
          READLN(ITEM);
          IF NOT(ITEM IN ['1'..'9']) THEN
            QUITCHNG := TRUE
          ELSE
            BEGIN
              IF (SECTION='B') AND (ITEM>'8') THEN
                BADITEM := TRUE;
              IF (SECTION='C') AND (ITEM>'3') THEN
                BADITEM := TRUE;
              IF BADITEM THEN
                WRITELN('INVALID ITEM')
            END
        END;
      IF NOT(BADITEM OR QUITCHNG) THEN
        BEGIN
          IF SECTION = 'A' THEN
            CASE ITEM OF
                       '1' : MENU1;
                       '2' : MENU2;
              '3','4','5' : MENU3;
                  '6','7' : MENU4;
                  '8','9' : MENU5
            END (* CASE *)
          ELSE
            BEGIN
              IF SECTION = 'B' THEN
                CASE ITEM OF
```

```
                    '1','2' : MENU6;
                        '3' : MENU7;
                    '4','5' : MENU8;
                        '6' : MENU9;
                    '7','8' : MENU10
                END (* CASE *)
              ELSE
                CASE ITEM OF
                        '1' : MENU11;
                    '2','3' : MENU12
                END (* CASE *)
            END
        END
    UNTIL QUITCHNG;
    WRITELN;
    WRITELN('DISPLAY NEW MENU - CTRL D');
    WRITELN('OVERLAY NEW FILE - CTRL O');
    WRITELN('REPLOT GRAPH     - CTRL P');
    WRITELN('TERMINATE PROGRM - CTRL Q')
  END; (* CHNGMENU *)


PROCEDURE SAVEPIC;
  VAR PICIMAGE :DISKBLOCK;
      PICFILE  :FILE;
      IO       :INTEGER;
      FILENAME :STRING[30];
  BEGIN
    PAGE(OUTPUT);
    TEXTMODE;
    WRITELN('== HIRES GRAPHICS STORAGE ==');
    WRITELN;
    WRITE('== ENTER NAME : ');
    READLN(FILENAME);
    PICIMAGE.INTPART := HIRES1;
    REWRITE (PICFILE,CONCAT(FILENAME,'[16]'));
    IO := BLOCKWRITE(PICFILE,PICIMAGE.PTRPART¬,16);
    CLOSE (PICFILE, LOCK);
    GRAFMODE
  END; (* SAVEPIC *)


PROCEDURE PGMCTRL; FORWARD;


PROCEDURE OVERLAY;
  BEGIN
    REPLT := TRUE;
    TEXTMODE;
    WRITELN;
    WRITELN('      OVERLAY FILE');
    WRITELN;
    OPENFILE;
    MENU1;
```

```
      MENU2;
      MENU3;
      WRITELN;
      WRITELN('C. PLOT OPTIONS');
      MENU12;
      FINISHPLT := FALSE;
      GRAFMODE;
      PGMCTRL;
      REPLT := FALSE;
    END; (* OVERLAY *)


  PROCEDURE SCANMODE;
    VAR KBCHR : CHAR;
        KBVAL : INTEGER;
        OPTIONS : STRING[10];
    BEGIN
      IF KEYPRESS THEN
        BEGIN
          READ(KEYBOARD, KBCHR);
          KBVAL := ORD(KBCHR);
          IF KBVAL IN [3,4,7,8,12,14,15,16,17,20] THEN
            CASE KBVAL OF
              CTRLG : GRAFMODE;
              CTRLT : TEXTMODE;
              CTRLH : BEGIN
                        FINISHPLT := TRUE;
                        CLOSE (TEXTFILE);
                        TEXTMODE;
                        EXIT (PGMCTRL)
                      END;
              CTRLQ : BEGIN
                        QUITREQ := TRUE;
                        IF NOT FINISHPLT THEN
                          EXIT (PGMCTRL)
                      END;
              CTRLD : IF FINISHPLT THEN
                        BEGIN
                          MENUPRT1;
                          MENUPRT2
                        END;
              CTRLC : IF FINISHPLT THEN
                        CHNGMENU;
              CTRLP : IF FINISHPLT THEN
                        BEGIN
                          RESET(TEXTFILE,TEXTNAME);
                          PGMCTRL
                        END;
              CTRLO : IF FINISHPLT THEN
                        OVERLAY;
              CTRLL : SAVEPIC;
              CTRLN : IF FINISHPLT THEN
                        BEGIN
                          TEXTMODE;
```

```
                        PAGE(OUTPUT);
                        WRITELN('   GRAPHICS HARDCOPY');
                        WRITELN;
                        WRITE(' ENTER THE PRINT OPTIONS : ');
                        READLN(OPTIONS);
                        REWRITE(PRINT,'PRINTER:');
                        WRITELN(PRINT,CHR(25),OPTIONS);
                        CLOSE(PRINT);
                        GRAFMODE
                      END
          END (* CASE *)
        ELSE
          NOTE (40, 25)
      END
  END; (* SCANMODE *)


 PROCEDURE POINTS(MAXCOL:INTEGER);
 BEGIN
   CASE MAXCOL OF
(*$I-*)
 2: READLN(TEXTFILE,PT[1],PT[2]);
 3: READLN(TEXTFILE,PT[1],PT[2],PT[3]);
 4: READLN(TEXTFILE,PT[1],PT[2],PT[3],PT[4]);
 5: READLN(TEXTFILE,PT[1],PT[2],PT[3],PT[4],PT[5]);
 6: READLN(TEXTFILE,PT[1],PT[2],PT[3],PT[4],PT[5],PT[6]);
 7: READLN(TEXTFILE,PT[1],PT[2],PT[3],PT[4],PT[5],PT[6],PT[7]);
 8: READLN(TEXTFILE,
      PT[1],PT[2],PT[3],PT[4],PT[5],PT[6],PT[7],PT[8]);
 9: READLN(TEXTFILE,
      PT[1],PT[2],PT[3],PT[4],PT[5],PT[6],PT[7],PT[8],PT[9]);
 10: READLN(TEXTFILE,
      PT[1],PT[2],PT[3],PT[4],PT[5],PT[6],PT[7],PT[8],PT[9],PT[10]);
       END;
 (*$I+*)
        IF(IORESULT<>0) THEN
          BEGIN
            WRITELN('BAD INPUT FORMAT ENCOUNTERED ');
            WRITELN('**FATAL**  PROGRAM TERMINATED');
            EXIT(PROGRAM)
          END;
END; (* POINTS *)


 PROCEDURE DRAW;
   VAR FIRST, ENDPTS : BOOLEAN;
       I,MAXCOL      : INTEGER;
       XMOV, YMOV    : REAL;
   BEGIN
     ENDPTS := FALSE;
     MAXCOL := XCOL;
     IF(MAXCOL<YPT) THEN MAXCOL := YPT;
     IF(MAXCOL<YLN) THEN MAXCOL := YLN;
     WHILE NOT(ENDPTS OR EOF(TEXTFILE)) DO
```

```
      BEGIN
        FIRST := TRUE;
        IF PTSKIP <> 0 THEN
          BEGIN
            I := 1;
            WRITE('SKIPPING POINTS');
            WHILE (I<=PTSKIP) AND (NOT EOF(TEXTFILE)) DO
              BEGIN
                SCANMODE;
                POINTS(MAXCOL);
                I := I + 1;
                WRITE('.')
              END;
            WRITELN
          END;
        I := 1;
        POINTS(MAXCOL);
        WHILE ((I<=NUMPTS) AND NOT(EOF(TEXTFILE))) DO
          BEGIN
            SCANMODE;
            WRITE('X=',PT[XCOL]);
            IF YPT <> 0 THEN
              BEGIN
                WRITE(' YPT=',PT[YPT]);
                MOVE2(PT[XCOL],PT[YPT]);
                RDRAW3(0,0,1)
              END;
            IF YLN <> 0 THEN
              BEGIN
                WRITE(' YLN=',PT[YLN]);
                IF FIRST THEN
                  BEGIN
                    FIRST := FALSE;
                    MOVE2(PT[XCOL],PT[YLN])
                  END
                ELSE
                  BEGIN
                    MOVE2(XMOV,YMOV);
                    DRAW2(PT[XCOL],PT[YLN])
                  END;
                XMOV := PT[XCOL];
                YMOV := PT[YLN];
              END;
            I := I + 1; POINTS(MAXCOL);
            WRITELN
          END;
        IF I > NUMPTS THEN
          ENDPTS := TRUE
      END;
    NOTE (40, 25);
    FINISHPLT := TRUE;
    CLOSE (TEXTFILE)
  END; (* DRAW *)
```

```
PROCEDURE SCREEN1;
  VAR BOTTOM,DOTVL,LEFT,RIGHT,TOP : REAL;
      I,ST,UNITS : INTEGER;
  BEGIN
    FINISHPLT := FALSE;
    GRAFMODE;
    FILLSCREEN (BLACK);
    IF YLABEL = 0 THEN
      BEGIN
        DOTVL := (XMAX - XMIN)/267;
        LEFT  := XMIN - DOTVL * 11
      END
    ELSE
      BEGIN
        DOTVL := (XMAX - XMIN)/242;
        LEFT  := XMIN - DOTVL * 36
      END;
    RIGHT  := XMAX + DOTVL * 2;
    DOTVL  := (YMAX - YMIN)/154;
    BOTTOM := YMIN - DOTVL * 22;
    TOP    := YMAX + DOTVL * 16;
    WINDOW (LEFT,RIGHT,BOTTOM,TOP);
    IF BOX OR AXIS THEN
      BEGIN
        IF LENGTH(XNAME) <> 0 THEN
          BEGIN
            UNITS := LENGTH(XNAME);
            IF UNITS > 34 THEN
              UNITS := 34;
            ST := 157 - ROUND(3.5*UNITS);
            MOVES (ST,0);
            TXT (XNAME,1,UNITS)
          END;
        IF LENGTH(YNAME) <> 0 THEN
          BEGIN
            UNITS := LENGTH(YNAME);
            IF UNITS > 18 THEN
              UNITS := 18;
            ST := 87 + ROUND(4.5*UNITS);
            FOR I := 1 TO UNITS DO
              BEGIN
                MOVES (0,ST);
                TXTCHR (YNAME[I]);
                ST := ST - 9
              END
          END
      END;
    IF LENGTH(TITLE) <> 0 THEN
      BEGIN
        UNITS := LENGTH(TITLE);
        IF UNITS > 40 THEN
          UNITS := 40;
        ST := 140 - ROUND(3.5*UNITS);
```

```
        MOVES (ST,182);
        TXT (TITLE,1,UNITS)
      END;
    IF BOX THEN
      BEGIN
        MOVE2 (XMIN,YMIN);
        DRAW2 (XMAX,YMIN);
        DRAW2 (XMAX,YMAX);
        DRAW2 (XMIN,YMAX);
        DRAW2 (XMIN,YMIN)
      END;
    IF AXIS THEN
      BEGIN
        MOVE2 (XMIN,YLIM);
        DRAW2 (XMAX,YLIM);
        MOVE2 (XLIM,YMAX);
        DRAW2 (XLIM,YMIN)
      END
  END; (* SCREEN1 *)


PROCEDURE SCREEN2;
  VAR TIC  : REAL;
      I, D, T : INTEGER;
  BEGIN
    I   := 0;
    TIC := XLIM;
    MOVE2 (XLIM,YMIN);
    WHILE (TIC <= XMAX) AND (XTIC <> 0) DO
      BEGIN
        IF (XLABEL<>0) THEN
          BEGIN
            IF (((I MOD XLABEL) = 0) AND (TIC <> XMIN)) THEN
              BEGIN
                MOVE2 (TIC,YMIN);
                D := DIGITS(TIC);
                RMOVES (D,-11);
                T := PLOTPTR↑.HLAST + 2 * ABS(D);
                IF T <= 279 THEN
                  RNUM (TIC,12,5)
              END
          END;
        IF BOX THEN
          BEGIN
            MOVE2 (TIC,YMIN);
            RDRAWS (0,2);
            MOVE2 (TIC,YMAX);
            RDRAWS (0,-2)
          END;
        IF AXIS THEN
          BEGIN
            MOVE2 (TIC,YLIM);
            IF YLIM = YMIN THEN
              RDRAWS (0,1)
```

```
            ELSE IF YLIM = YMAX THEN
              RDRAWS (0,-1)
            ELSE
              BEGIN
                RMOVES (0,1);
                RDRAWS (0,-2)
              END
          END;
        TIC := TIC + XTIC;
        I   := I + 1
      END; (* WHILE *)
    I   := 0;
    TIC := XLIM;
    MOVE2 (XLIM,YMIN);
    WHILE (TIC >= XMIN) AND (XTIC <> 0) DO
      BEGIN
        IF (XLABEL<>0) THEN
          BEGIN
            IF (((I MOD XLABEL) = 0) AND (TIC <> XMIN)) THEN
              BEGIN
                MOVE2 (TIC,YMIN);
                D := DIGITS(TIC);
                RMOVES (D,-11);
                T := PLOTPTR↑.HLAST + 2 * ABS(D);
                IF T <= 279 THEN
                  RNUM (TIC,12,5)
              END
          END;
        IF BOX THEN
          BEGIN
            MOVE2 (TIC,YMIN);
            RDRAWS (0,2);
            MOVE2 (TIC,YMAX);
            RDRAWS (0,-2)
          END;
        IF AXIS THEN
          BEGIN
            MOVE2 (TIC,YLIM);
            IF YLIM = YMIN THEN
              RDRAWS (0,1)
            ELSE IF YLIM = YMAX THEN
              RDRAWS (0,-1)
            ELSE
              BEGIN
                RMOVES (0,1);
                RDRAWS (0,-2)
              END
          END;
        TIC := TIC - XTIC;
        I   := I + 1
      END (* WHILE *)
  END; (* SCREEN2 *)
```

```
PROCEDURE SCREEN3;
  VAR TIC  : REAL;
      I    : INTEGER;
  BEGIN
    I   := 0;
    TIC := YLIM;
    MOVE2 (XMIN,YLIM);
    WHILE (TIC <= YMAX) AND (YTIC <> 0) DO
      BEGIN
        IF (YLABEL<>0) THEN
          BEGIN
            IF (I MOD YLABEL) = 0 THEN
              BEGIN
                MOVE2 (XMIN,TIC);
                RMOVES (-28,-3);
                RNUM (TIC,12,5)
              END
          END;
        IF BOX THEN
          BEGIN
            MOVE2 (XMIN,TIC);
            RDRAWS (2,0);
            MOVE2 (XMAX,TIC);
            RDRAWS (-2,0)
          END;
        IF AXIS THEN
          BEGIN
            MOVE2 (XLIM,TIC);
            IF XLIM = XMIN THEN
              RDRAWS (1,0)
            ELSE IF XLIM = XMAX THEN
              RDRAWS (-1,0)
            ELSE
              BEGIN
                RMOVES (1,0);
                RDRAWS (-2,0)
              END
          END;
        TIC := TIC + YTIC;
        I   := I + 1
      END; (* WHILE *)
    I   := 0;
    TIC := YLIM;
    MOVE2 (XMIN,YLIM);
    WHILE (TIC >= YMIN) AND (YTIC <> 0) DO
      BEGIN
        IF (YLABEL<>0) THEN
          BEGIN
            IF (I MOD YLABEL) = 0 THEN
              BEGIN
                MOVE2 (XMIN,TIC);
                RMOVES (-28,-3);
                RNUM (TIC,12,5)
              END
```

```
            END;
          IF BOX THEN
            BEGIN
              MOVE2 (XMIN,TIC);
              RDRAWS (2,0);
              MOVE2 (XMAX,TIC);
              RDRAWS (-2,0)
            END;
          IF AXIS THEN
            BEGIN
              MOVE2 (XLIM,TIC);
              IF XLIM = XMIN THEN
                RDRAWS (1,0)
              ELSE IF XLIM = XMAX THEN
                RDRAWS (-1,0)
              ELSE
                BEGIN
                  RMOVES (1,0);
                  RDRAWS (-2,0)
                END
            END;
          TIC := TIC - YTIC;
          I   := I + 1
        END (* WHILE *)
    END; (* SCREEN3 *)


  PROCEDURE PGMCTRL;
    BEGIN
      IF NOT REPLT THEN
        BEGIN
          SCREEN1;
          IF BOX OR AXIS THEN
            BEGIN
              SCREEN2;
              SCREEN3
            END
        END;
      DRAW
    END; (* PGMCTRL *)


BEGIN
  MAKEPORT (PLOTPTR);
  REPLT   := FALSE;
  QUITREQ := FALSE;
  PAGE(OUTPUT);
  OPENFILE;
  MENU1;
  MENU2;
  MENU3;
  MENU4;
  MENU5;
  MENU6;
```

```
  MENU7;
  MENU8;
  MENU9;
  MENU10;
  MENU11;
  MENU12;
  PGMCTRL;
  WHILE NOT QUITREQ DO
    BEGIN
      SCANMODE
    END;
  PAGE(OUTPUT)
END. (* PLOT *)
```

REFERENCES

Apple Computer Inc., 1980 **Apple Pascal Language Reference Manual**, Cupertino, California

Apple Computer Inc., 1981a **Applegraphics User's Manual**, Cupertino, California

Apple Computer Inc., 1981b **Apple II Reference Manual**, Cupertino, California

Luehrmann, Arthur & Herbert Peckham, 1981 **Apple Pascal - A Hands-on Approach**, Monterey, California: McGraw-Hill, Inc.

Lord, Mark Steven, 1982 **"A Digital Data Recorder and Transfer Device for the Marconi 722B Satellite Navigation Receiver"**, CS4993 Undergraduate Report, School of Computer Science, University of New Brunswick, Fredericton, New Brunswick; Dept. of Surveying Engineering Technical Report No. 88.

Orange Micro Inc., 1982 **Grappler+ Printer Interface Operators Manual**, Anaheim, California

Shapiro, Neil, 1982 **"Logo: The Newest Computer Language"** **Popular Mechanics Magazine**, New York, New York: Hearst Corporation, September 1982

BIBLIOGRAPHY

Barden, William, 1982 "High-Resolution Graphics System" Popular Computing Magazine, Peterborough, New Jersey: BYTE Publications, July 1982

Housley, Trevor, 1979 Data Communications and Teleprocessing Systems, Englewood Cliffs, New Jersey: Prentice-Hall, Inc.

Hume, J.N.P. & R.C. Holt, 1982 USCD Pascal: A Beginner's Guide to Programming Micro-computers, Reston, Virginia: Reston Publishing Company, Inc.

International Business Machines, 1979 Product# SH20-9062 VSPC Fortran Terminal User's Guide, San Jose, California: IBM Corporation

International Business Machines, 1980 Product# SH20-9071 VSPC General User's Guide and Command Language, San Jose, California: IBM Corporation

Shinshu Seiki Co. Ltd., 1982 Dot Matrix Printer Operation Manual, Nagano, Japan