VPACK: A FORTRAN-CALLABLE GRAPHICS PACKAGE FOR THE RAND VIDEO GRAPHIC SYSTEM

PREPARED UNDER GRANT FROM THE NATIONAL INSTITUTES OF HEALTH

R. LAWRENCE CLARK
GABRIEL F. GRONER

R-1015-NIH
DECEMBER 1972
This research was supported by the National Institutes of Health under Grant No. 1 R01 CA12369-02. Views or conclusions contained in this study should not be interpreted as representing the official opinion or policy of Rand or of the National Institutes of Health.
VPACK: A FORTRAN-CALLABLE GRAPHICS PACKAGE FOR THE RAND VIDEO GRAPHIC SYSTEM

PREPARED UNDER GRANT FROM THE NATIONAL INSTITUTES OF HEALTH

R. LAWRENCE CLARK
GABRIEL F. GRONER

R-1015-NIH
DECEMBER 1972

Rand
SANTA MONICA, CA. 90406
PREFACE

This report, part of Rand's continuing effort in facilitating man-computer communication, describes VPACK, a package of subroutines that enable Fortran programs to communicate with the Rand Video Graphic System. Although this hardware currently exists only at Rand, the general principles outlined herein are applicable to a wide variety of display devices.

The report is directed toward Fortran programmers who wish to use the Rand system in an IBM System/360 environment, and toward systems programmers who are designing similar packages for other graphics hardware.

VPACK was developed for use in BIOMOD, an interactive computer graphics system for modeling biological and other continuous systems. The goal of VPACK was to provide a small package of relatively primitive routines that did not force the programmer to use any particular philosophy of interaction or display management, but rather, allowed him to deal with interaction and to create pictures that satisfied the needs of his particular application.

A by-product of this philosophy is that the package is relatively device-independent (except for geometrical considerations) and therefore facilitates transferring programs to other display systems. As a result, the package has found application in a number of programs in addition to BIOMOD.
SUMMARY

VPACK is a package of subroutines that enable Fortran programs to communicate with the Rand Video Graphic System. The package comprises routines for building and transmitting pictures, for receiving input from the Rand Tablet and the console keyboard, and for producing hardcopy of displayed images.

The VPACK routines perform relatively primitive functions, e.g., drawing a series of lines or positioning a string of characters. More complex functions can be created by combining calls to several VPACK routines in a Fortran subprogram.

VPACK operates on an IBM System/360 computer that is connected to the Rand Video Graphic System, a collection of hardware and software used to communicate with Video Graphic consoles. At present, this combination is available only at Rand. However, most of the VPACK capabilities could be implemented on a number of commercially available systems. In effect, VPACK serves to isolate the hardware dependencies to a relatively small package of code that can be replaced without requiring any change in the application program.

This report describes the VPACK subroutines and illustrates their use through a sample program.
## CONTENTS

PREFACE ......................................................... iii

SUMMARY ........................................................ v

Section

I. INTRODUCTION ............................................... 1
   The Console ................................................. 2
   The Screen ................................................. 2
   The Rand Tablet .......................................... 3
   The Keyboard .............................................. 4
   Hardcopy ................................................... 5
   Storage Requirements .................................... 5

II. SUBROUTINE DESCRIPTIONS ............................... 6
    VCHAR .................................................. 9
    VCOPY ............................................... 10
    VDIM ............................................... 11
    VGETCH .............................................. 12
    VINIT ............................................... 13
    VINTEN ............................................... 14
    VJLINE ............................................. 15
    VLINES ............................................. 16
    VLTYPE ............................................. 17
    VMODCH ............................................. 18
    VMOVE ............................................... 19
    VNULL ............................................... 20
    VORG ................................................ 21
    VPEN ............................................... 22
    VPOSIT ............................................. 24
    VSEND .............................................. 25
    VSIZE ............................................... 26
    VTABS ............................................... 27
VTRUNC ................................................................. 28
VWAIT ................................................................. 29

III. SAMPLE PROGRAM ........................................... 30

Appendix
A. LISTING OF SAMPLE PROGRAM ......................... 41
B. UTILITY ROUTINES ............................................. 45

REFERENCES .......................................................... 47
I. INTRODUCTION

VPACK is a package of subroutines that enable Fortran programs to communicate with the Rand Video Graphic System [1]. The package comprises routines for building and transmitting pictures, for receiving input from the Rand Tablet [2] and the console keyboard, and for producing hardcopy of displayed images.

The VPACK routines perform relatively primitive functions. More complex functions can be created by combining calls to several VPACK routines in a Fortran subprogram.

VPACK operates on an IBM System/360 computer that is connected to the Rand Video Graphic System (VGS), a collection of hardware and software used to communicate with Video Graphic consoles. At present, this combination is available only at Rand. However, most of the capabilities available through VPACK could be implemented on a number of commercially available systems, e.g., the IBM 2250. In effect, VPACK serves to isolate the hardware dependencies to a relatively small package of code. Should a need arise to transfer an application written with VPACK to another system, this small package of routines could be replaced by an equivalent package for the new system. The application program would not require significant changes unless the geometry of the screen or of the character sizes differed markedly from those of VGS.

The VPACK display routines operate on a buffer provided by the calling program. The programmer specifies the address and size of the buffer during initialization. Thereafter, VPACK retains this information so that the programmer no longer need be concerned with it.

Several versions of VPACK are currently in existence. The first, written in early 1970, provided a fairly general set of features, but did not allow for keyboard usage. Since then, various users with special needs have added their own extensions to the package. In this report we shall describe the version used in the BIOMOD system [3]. This version is especially useful in applications where the Rand Tablet serves as the primary input device, but where the keyboard is also available for entering voluminous text.
THE CONSOLE

The Video Graphic console (Fig. 1) consists of a screen on which images are displayed, a series of buttons used to log on and off, a keyboard for typing text, and (optionally) a Rand Tablet. Throughout this report we shall assume that a Tablet is available.

The Screen

The screen is described as a grid of 1024 by 1024 "raster units." The x or horizontal coordinates range from 0 at the extreme left to 1023 at the extreme right. Similarly, the y or vertical coordinates range from 0 at the bottom to 1023 at the top.

Fig. 1—The Video Graphic console with Rand Tablet
Two character sizes are available. A normal size character, about a quarter-inch high, occupies an envelope of 14 horizontal raster units by 20 vertical raster units. A large size character, about half an inch high, occupies 28 horizontal raster units by 40 vertical raster units. Thus, the screen may contain 51 lines of 73 normal size characters each, or 25 lines of 36 large characters each.

Character positions are described by the center of the envelopes they occupy. Although a program may position characters at will, VPACK assumes that characters typed or printed by the user will be positioned with centers at integer multiples of 14 and 20 raster units. This assumption is used to distinguish between hand-printed commas and prime signs (single quotes), to position keyboard characters, and to position the keyboard cursor, an underscore that appears on the screen to indicate where subsequent keyboard characters are to be placed. For these reasons, the VPACK programmer is advised to adhere to this convention when positioning subpictures on which the user can write or type; it may be disregarded for other subpictures.

The Rand Tablet

The Rand Tablet consists of a pen-like stylus (or pen) and a 10 × 10 in. horizontal writing surface located in front of the screen. The tablet surface and the screen may be regarded functionally as a single surface; each point on the tablet surface corresponds to a point on the screen, and vice versa. The user writes on the tablet and watches his writing on the screen. As the pen is moved above the tablet surface, a dot on the screen follows its motion; this direct feedback point helps the user position the pen for pointing or drawing. When the pen is pressed against the surface, a switch closes, notifying VPACK that an action is beginning.

The pen may be used to print or draw, to push displayed “buttons,” or to “drag” a figure or text to a new location. When printing, the user places the pen on the tablet surface, moves it, and lifts it off as he would in printing on paper. As the pen moves across the tablet, its track is displayed on the screen—the pen appears to have ink. When it is lifted, the switch opens and inking ceases. The pen track is analyzed by symbol-recognition routines [4]. When a symbol is recognized, the ink is erased, and the recognized symbol is returned to the calling program for interpretation. The recognized symbols appear in Fig. 2.

Pushing is usually associated with a software “button”—a displayed area on the screen which the user touches to perform a particular function. To push a button, the user simply touches the pen to the tablet in the button area, then lifts the pen.

Dragging is often used to reposition a geometric figure or to move a copy of displayed text, eliminating the need to reprint or type it. Once the user touches an appropriate figure with the pen, the application program can redisplay the figure at the current pen coordinates until the pen is lifted.
The Keyboard

The console keyboard is a typewriter-like device that may be used to enter text into an application program. Data from the keyboard are returned by VPACK as if they had been printed with the tablet.

In order to allow the user to position keyboard characters, VPACK uses a cursor. This character appears on the screen as an underscore immediately below the position where the next keyboard character will be placed. To establish a cursor, the user touches the pen to the tablet and lifts it again, making no other motion. This action is recognized as a period. VPACK returns the recognized period, and also displays the cursor.

Once the cursor is established, it may be used until the pen is again touched to the tablet surface. Each normal keyboard character is returned with the current cursor position, and the cursor is moved one position (14 raster units) to the right.

The cursor may be repositioned from the keyboard itself. The space bar is divided into two parts, marked SKIP and BLANK. Either of these moves the cursor one space to the right, but the BLANK key also returns a blank character; the SKIP key simply moves the cursor without returning a character. The BACKSPACE key moves the cursor one position to the left without returning a character. The RETURN key moves the cursor to the left margin and down one line (20 raster units). A termination code (X'00') is returned, indicating end of line.

The left margin is initially assumed at 28 raster units, but may be altered by calling subroutine VTABS. The CNTL key moves the cursor to the left margin and one line higher, returning a termination code. The TAB and BACKTAB keys move
the cursor to the next tab position to the right or left, respectively, but do not return a character. Tab positions are initially set at five-position (70-raster-unit) intervals, but may be altered by calling VTABS.

HARDCOPY

The VPACK hardcopy facility enables VPACK programs to obtain high-quality copy of the display. It is activated by calling VCOPY, usually in response to a user input (e.g., pushing a software button). The system writes an image of the current display screen contents onto disc for subsequent processing by microfilm equipment. At Rand both paper and 35mm film forms are available.

STORAGE REQUIREMENTS

The complete VPACK package occupies 25,216 (decimal) bytes. This space is broken down as follows:

- Display routines 2,468
- Tablet-handling routines 4,860
- Character recognition 13,560
- Hardcopy routines 4,328

For those applications that do not require character recognition or hardcopy, the unneeded routines can be replaced with short dummy routines to save storage space.
II. SUBROUTINE DESCRIPTIONS

The routines provided by VPACK are categorized below. More complete descriptions appear in alphabetical order on the following pages.

Initialization
VINIT—Initializes the buffer and tablet

Picture Transmission
VSEND—Sends a picture to the screen

Input
VPEN—Retrieves tablet and keyboard data

Character Displays
VCHAR—Adds a character string to the buffer
VGETCH—Extracts characters from the buffer
VMODCH—Modifies characters in the buffer
VSIZE—Adds a character-size code to the buffer

Line Drawing
VLINES—Adds discrete line segments to the buffer
VLINE—Adds joined lined segments to the buffer
VLTYPE—Adds a line-type code to the buffer

Intensity Control
VINTEN—Alters intensity codes in the buffer
VDIM—Adds a dim-mode code to the buffer

Repositioning
VMOVE—Moves a subpicture

Control
VPOSIT—Retrieves the current buffer position
VORG—Resets the current buffer position
VTRUNC—Truncates the buffer

Hardcopy
VCOPY—Produces hardcopy of the current display

Miscellaneous
VNULL—Adds null codes to the buffer
VWAIT—Waits a prescribed period of time
VTABS—Sets keyboard tab and margin positions

6
The VPACK display routines operate on a buffer provided by the calling program. The programmer specifies the address and size of the buffer during initialization. Thereafter, VPACK retains this information so that the programmer no longer need be concerned with it. VPACK also maintains information about the amount of buffer space currently being used (the current length) and the position into which the display commands are being stored (the current position). These data are automatically updated by the several routines as appropriate. (See Fig. 3.) The current length of a buffer may exceed the maximum length allowed by the Video Graphic System for a single picture segment. In this event, VPACK automatically segments the picture at appropriate points and passes the multiple segments to VGS.

More detailed descriptions of the subroutines follow, in alphabetical order.
Fig. 3—Use of current position and current length
VCHAR: Adds a Character String to the Buffer

CALL VCHAR (IX, IY, ST, N)

VCHAR adds N characters from the string ST to the buffer, beginning at the current position. IX and IY specify the center of the first character position. The intensity of the string is set to normal. The current position and current length are updated.
VCOPY: Produces Hardcopy of the Current Picture

CALL VCOPY (L)

VCOPY effectively produces hardcopy (both paper and 35mm film) of the current picture. More specifically, it reformats the buffer into a meta language, which it writes on a disc data set named SC4060ZZ. This data set is subsequently transferred to magnetic tape and processed off-line.

Parameter L specifies the action to be taken by the microfilm processor after drawing the image. Its values are:

0 Produce 11 × 14 copy and advance the frame
1 Produce 11 × 14 copy but do not advance frame
2 Produce 8 × 11 copy and advance the frame
3 Produce 8 × 11 copy but do not advance the frame
4 Close the output file

In normal operation, the frame is advanced after each frame is drawn. If the frame is not advanced, the next picture that is copied will be superimposed onto the current picture.

Programs that call VCOPY require a Job Control Language DD statement for the SC4060ZZ data set. Such a statement may be of the form

```
//SC4060ZZ DD DSN=\textit{name} DISP=MOD
```

where \textit{name} is the name chosen when the data set was created. The data set must first be created by running a job such as

```
// EXEC PGM=BR14
// anyname DD DSN=\textit{name},UNIT=SYSDA,VOL=SER=pack,
// DISP=(NEW,CATLG),SPACE=(240,(100,100)),
// DCB=(RECFM=FB,BLKSIZE=240,LBLRECL=240)
```

To copy the disc data set onto tape and reset the contents of the disc data set to the null state, use the following statements:

```
//GO EXEC PGM=IEBGENER
//SYSPRINT DD SYSPUT=A
//SYSPUT1 DD DSN=\textit{name},DISP=OLD
//SYSPUT2 DD DSN=METATAPE,VOL=SER=tapeno,
// LABEL=(1,BLP),UNIT=TAPE7,DISP=(NEW,KEEP),
// DCB=(*,SYSPUT1,DEN=2,TRTCH=C,BLKSIZ=240)
//SYSPIN DD DUMMY
// EXEC PGM=CLOSE,COND=(0,NE,GO)
//STEPLIB DD DSN=SYS1.RANDCOMP,DISP=SHR
//CARD DD DSN=\textit{name},DISP=OLD
```

If the last three cards are omitted, the data set will be copied onto tape but will not be reset.
VDIM: Adds Adds a Dim-Mode Code to the Buffer

CALL VDIM (L)

VDIM adds a dim-mode code to the buffer at the current position. The current position and current length are updated. The dim mode may be set either on or off. When it is on, all subsequent subpictures that would have appeared at normal or bright intensity are displayed as dim until a code is encountered that turns the dim mode off. Dim and off intensities are not affected. If no dim-mode code is specified, the dim mode is set off. The values of L are:

1. Dim mode off (intensities used as specified)
2. Dim mode on (normal and bright intensities treated as dim)
VGETCH: Extracts Characters from the Buffer

CALL VGETCH (J,L,ST,N)

VGETCH extracts N characters from the buffer and places them in the first N positions of string ST. The characters are extracted from the subpicture that begins at position J, beginning with the Lth character and extending through the (L + N - 1)th character. The current position and current length are unchanged.
VINIT: Initializes the Display, Tablet, and Keyboard

CALL VINIT(NAME,BUFF,N)

VINIT connects the program to the Video Graphic System and activates the tablet and keyboard. NAME is a string of exactly eight characters which the user will type to connect his terminal to the program (CREATE ← → NAME). If the name consists of fewer than eight characters, it must be padded with trailing blanks. BUFF is the buffer, which VINIT initializes with a RESET command. N is the maximum length of BUFF in bytes. The current position and current length are initialized.

An appropriate value for N can be estimated by considering the longest buffer that will be created, using the following guide:

- Each call to VCHAR requires n+6 bytes, where n is the length of the character string.
- Each call to VLINES requires 10n+1 bytes, where n is the number of discrete line segments.
- Each call to VJLINE requires 5n+6 bytes, where n is the number of joined line segments.
- Each call to VNNULL requires n bytes, where n is the number of null codes to be added.
- Each call to VSIZE, VTYPE, or VDIM requires 1 byte.
- All other calls require no space.
VINTEN: Alters Intensity Codes in the Buffer

CALL VINTEN (N,J,INT)

VINTEN changes the intensity of N subpictures whose positions appear in array J. The new intensity for all such subpictures is specified by the scalar INT. Unless changed, all subpictures appear at normal intensity. The values of INT are:

1  Off
2  Dim
3  Normal
4  Bright

The current position and current length are unchanged.
VJLINE: Adds Joined Line Segments to the Buffer

CALL VJLINE (N,IX,IY)

VJLINE adds N joined line segments to the buffer, beginning at the current position. The lines are set to normal intensity. The current position and current length are updated.

The lth line extends from the point IX(1),IY(1) to the point IX(l+1),IY(l+1). Arrays IX and IY must therefore contain at least N+1 entries. All coordinates after the first are entered in the buffer relative to the first; this enables the program to move the entire set of lines by a single call to VMOVE.
VLINES: Adds Discrete Line Segments to the Buffer

CALL VLINES (N,IXF,IYF,IXT,IYT)

VLINES adds N discrete line segments to the buffer, beginning at the current position. The line segments are set to normal intensity. The current position and current length are updated.

The lth line extends from \( \text{IXF}(l), \text{IYF}(l) \) to \( \text{IXT}(l), \text{IYT}(l) \). All coordinates after the first are entered in the buffer relative to the first; this enables the program to move the entire set of lines by a single call to VMOVE.
VLTYPE: Adds a Line-Type Code to the Buffer

CALL VLTYPE (L)

VLTYPE adds a line-type code to the buffer at the current position. The current position and current length are updated. The line-type code is a modal relative to the buffer; that is, once the line type has been specified, it applies to all lines appearing in the buffer until the next line-type code appears. If no line-type code is specified, solid lines are used. The values of L are:

1  Solid Lines
2  Dashed lines
3  End points only.
VMODCH: Modifies Characters in the Buffer

CALL VMODCH (J,L,ST,N)

VMODCH causes the first N characters of the string ST to replace the Lth through (L+N−1)th characters in the subpicture that begins at position J. (L+N−1) must not exceed the length of the original string; violation of this rule may result in garbled pictures. The current position and current length are unchanged.
VMOVE: Moves a Subpicture

CALL VMOVE (J,IX,IY)

VMOVE changes the starting coordinates of the subpicture that begins at position J. The new position IX,IY replaces the IX,IY specified in a call to VCHAR, or the IXF(1),YF(1) in a call to VLINES, or the IX(1),Y(1) in a call to VJLINE. Because all of these subpictures are located in relation to their starting coordinates, the entire subpicture is moved. The current position and current length are unchanged.
VNULL: Adds Null Codes to the Buffer

CALL VNULL (N)

VNULL places N null codes into the buffer at the current position. The current position and current length are updated. This routine enables the programmer to replace a subpicture with a shorter one without truncating the buffer.
VORG: Resets the Current Buffer Position

CALL VORG (J)

VORG sets the current position to the value of J. The current length is unchanged. The value of J must not exceed the current length. This routine allows the programmer to reconstruct part of the display by replacing a sequence of buffer orders with another sequence of the same length. To guarantee that identical lengths are used, it is good practice to replace character strings only with other character strings, sequences of joined lines with other sequences of joined lines, etc., and to ensure that the number of characters or lines is the same as the number being replaced.
VPEN: Communicates with the Tablet and Keyboard

CALL VPEN (IACT,ISIZE,ITIME,IDATA,IEXIT)

VPEN provides the user program with data from the tablet or keyboard. The parameter IACT allows it to be used for character recognition, pen tracking, or scrolling. Pen coordinates are always returned on first pen-down so that the calling program can use the positional information to determine how next to call VPEN.

IACT = 1 indicates the recognition mode. VPEN inks and recognizes characters as described in [4]; the recognized symbols appear in Fig. 2. In this mode, VPEN returns on first pen-down in a character and also each time a character is recognized, regardless of whether the user has started another character. It also returns if a specified time has elapsed since the last character was recognized; this enables the calling program to determine that the user has stopped writing. In this mode, if a keyboard character is typed and a cursor is active, VPEN returns as if the character had come from the tablet—i.e., it simulates a first pen-down exit and a character exit, using the cursor position as the character center.

IACT = 2 indicates the tracking mode. VPEN does not ink, but returns pen coordinates immediately. This mode is used for dragging figures and for tracing curves. The calling program may test for pen-up or for an elapsed time since pen-up to determine if the user has completed his action.

IACT = 3 indicates a combination of recognition and tracking. It causes VPEN to ink and recognize exactly as in the IACT = 1 call, but returns control immediately regardless of whether a character is recognized. This mode is used when the user’s intent is not clear—e.g., when he is moving the pen in an area where both writing and dragging are permitted. It should not be used when recognition alone is required because data may be lost while VPEN is returning pen coordinates.

IACT = 4 indicates the "scrolling" mode. It returns pen coordinates on pen-down, on pen-up, and whenever a specified time has elapsed since being called. No inking is performed. This mode is used for time-based operations such as scrolling (moving lines of text continuously).

Parameter ISIZE is the user’s expected character size. It is used to determine the size of ink vectors and to separate handwritten characters. It applies only when recognition is to be performed (IACT = 1 or 3). The values of ISIZE are:

1 Normal size characters (about 1/4 in. high)
2 Large characters (about 1/2 in. high)
3 Extra-large characters (higher than 3/4 in.)

When IACT = 2 or IACT = 4, ISIZE should be set to 1.

Parameter ITIME is the time-expired value, expressed in hundredths of a second. Its interpretation varies according to the value of IACT. For IACT = 1 or IACT = 3, it represents the time required between pen strokes for character separation, or the time required following a character for the "time-expired" exit. For character completion, ITIME = 50 is a good choice; for indicating that writing has stopped, ITIME
= 100 works well. For IACT = 2, ITIME is the time required after a pen-up for the "time-expired" exit. For IACT = 4, ITIME is the maximum elapsed time that VPEN should retain control before taking the "time-expired" exit.

Parameter IDATA is an array containing six words of output data. On exit from VPEN, it contains the following data:

\[
\begin{align*}
\text{IDATA}(1) &= \text{X coordinate of the pen location or character center} \\
\text{IDATA}(2) &= \text{Y coordinate of the pen location or character center} \\
\text{IDATA}(3) &= \text{character width} \\
\text{IDATA}(4) &= \text{character height} \\
\text{IDATA}(5) &= \text{character code, right aligned} \\
\text{IDATA}(6) &= \text{character code, left aligned}
\end{align*}
\]

For IACT = 1 or IACT = 3, all six entries are supplied; for IACT = 2 or IACT = 4, only the first two are meaningful.

Parameter IEXIT is an exit code indicating the reason for VPEN's return. Its six possible values are:

1. First pen-down (all)
2. Continued pen-down (2,3)
3. Pen-up (2,3,4)
4. Character recognized, followed by timeout (1,3)
5. Character recognized, more data pending (1,3)
6. Time expired (all)

The numbers in parentheses above indicate the values of IACT to which the exit applies—e.g., for IACT = 1, the possible exit codes are 1, 4, 5, and 6.
VPOSIT: Retrieves the Current Buffer Position

CALL VPOSIT (J)

VPOSIT returns the current position in location J. The current position and current length are unchanged.
VSEND: Sends a Picture to the Screen

CALL VSEND

VSEND sends an updated picture to the screen. Because all of the other picture manipulation routines merely alter the buffer, this routine must be called to make their effects visible to the user. The current position and current length are unchanged.
VSIZE: Adds a Character-Size Code to the Buffer

CALL VSIZE (L)

VSIZE adds a character-size code to the buffer at the current position. The current position and current length are updated. The size code is a modal relative to the buffer; that is, once a size has been specified, it applies to all characters until another size is specified. If no size code is specified, characters will be of normal size. The values of L are:

1. Normal size characters
2. Large size characters
VTABS: Sets Keyboard Tab and Margin Positions

CALL VTABS (ITABLE)

VTABS resets the positions of keyboard tab stops and margins. Parameter ITABLE is an array of at least 2*N + 5 words, where N is the number of tab stops desired. The contents of these words are:

ITABLE(1) = number of lines on the form
ITABLE(2) = unused
ITABLE(3) = X coordinate of leftmost character center
ITABLE(4) = Y coordinate of topmost character center
ITABLE(5) = number of fields on the form
ITABLE(2*i + 4) = position of leftmost character in field i
ITABLE(2*i + 5) = position of rightmost character in field i

Tab positions are set at the first character position of each field. The cursor's movement is restricted to remain between the first position of the first field and the last position of the last field, and between the top and bottom lines of the form. If VTABS is not called, the default values allow the cursor center to range from 28 to 1008 in the x dimension and from 20 to 1000 in the y dimension, with tabs positioned every 5 character spaces (70 raster units).
VTRUNC: Truncates the Buffer

CALL VTRUNC (J)

VTRUNC sets the current length to the value of J. If the current position exceeds the value of J, it too is reset to that value. The value of J must not exceed the current length. This routine effectively deletes all display code appearing in the buffer beyond the specified point.
VWAIT: Waits a Prescribed Period of Time

CALL VWAIT (N)

VWAIT places the user's task in a wait state for N/100 seconds. Although it is not directly related to the other VPACK routines, it may be useful in interactive programs.
III. SAMPLE PROGRAM

A sample Fortran program illustrates the use of VPACK. This program allows its user to enter data points from a tablet and to print the values of the points. The user specifies the scales of the axes; these values may be changed at any time to rescale the data curve. Existing points may be moved or deleted by dragging. The user may generate a hardcopy picture of the screen (as in Fig. 4), print the data, clear all existing points, or log off, simply by "pushing" the appropriate software button.

A complete listing of the sample program appears in Appendix A. A fragmented listing is interspersed with the descriptive text that follows.

```
DIMENSION BUFFER(1100)
DIMENSION PENDAT(6)
EQUIVALENCE (IXPEN, PENDAT(1)), (IYPEN, PENDAT(2)),
1 (IMPEN, PENDAT(3)), (IYPEN, PENDAT(4)), (IPENCH, PENDAT(5)),
2 (PENCHR, PENDAT(6))
DATA IAXLX/225/, IAXHIX/925/, IAXLOY/160/, IAXHIY/860/
```

BUFFER is the VPACK buffer. PENDAT is the package of pen data returned by VPEN; mnemonic names are also supplied for the individual entries. IAXLOX, IAXHIX, IAXLOY and IAXHIY are the boundaries of the graph area of the screen.

```
DIMENSION IRNLOX(4), IRNCNY(4)
DATA IRNLOX/144, 844, 32, 32/, IRNCNY/100, 100, 160, 860/
DIMENSION IBUTX(4)
DATA IBUTX/163, 363, 563, 763/, IBUTY/950/
```

Four range boxes at the extremes of the axes indicate the ranges of the axes. The leftmost edges of these boxes appear in array IRNLOX; the vertical centers are in array IRNCNY. Array IBUTX gives the coordinates of the leftmost character position of the four "buttons." Scalar IBUTY gives their common vertical position.
Fig. 4—Picture produced by sample program
Arrays IXMTR and IYMETR describe a "meter," positioned initially at coordinates 500,500. The meter is a distinctively shaped box that follows the pen inside the graph area, and states the coordinate value of the point which the pen is touching.

**DATA ISCUB/50/**
**DIMENSION TEMP(3)**

ISCUB is the character code returned by VPEN when the user scrubs (erases) characters. TEMP is a temporary buffer, often used to contain a string of twelve characters.

**DIMENSION RANGE(4), JRANGE(4)**

RANGE is an array containing the numerical values of the strings in the four range boxes. The indices are: 1 for the lower X range, 2 for the upper X, 3 for the lower Y, and 4 for the upper Y. JRANGE contains the buffer position of the four corresponding character strings.

**DIMENSION XPOINT(50), YPOINT(50), IXPNT(50), IYPNT(50)**

Arrays XPOINT and YPOINT contain the numerical values of up to 50 data points. IXPNT and IYPNT contain the same values converted to display coordinates, considering the current values in the range boxes.

**C--INITIALIZE VPACK**
**CALL VINIT('SAMPLE ', BUFFER, 4000)**

The program first calls VINIT to initialize VPACK and allow the user to connect. Note that the user will type "CREATE ←→ SAMPLE" to initiate the program. An eight-character string is supplied by padding with blanks at the right. Also notice that BUFFER, dimensioned for 1000 words, is described to VINIT as containing 4000 bytes.

**C--DRAW AXES**
**CALL VRECT(IAXLCX, IAXHIX, IAXLCY, IAXHIIY)**

Next the graph area is outlined by calling VRECT. This is a Fortran-coded subroutine that draws a rectangle, given its corner coordinates. It illustrates the ability to construct higher-level routines from the primitive VPACK functions. A listing of VRECT appears at the end of this section.
C--DRAW RANGE BOXES
  DO 50 I=1,4
    CALL VRRECT(IRNLX(I),IRNLOX(I)+174,IRNCVY(I)-15,IRNCNY(I)+15)
    RANGE(I)=0.
    CALL VPOSIT(JRANGE(I))
    CALL VCHAR(IRNLX(I)+10,IRNCNY(I),' '+0.00000E+00',12)
    IF (I.EQ.1).AND.(I.EQ.3) GO TO 50
    RANGE(I)=1.
  CALL VMODCH(JRANGE(I),2,'1',1)
50 CONTINUE

The four range boxes are placed by a DO-loop. Each box is drawn by calling VRRECT. The value in the range box is set initially to zero. After saving the current position by calling VPOSIT, a character string representing zero in scientific notation is displayed in the box. For the lower boxes, this completes the loop. For the upper boxes, the range value is reset to one, and the character string is altered accordingly by calling VMODCH. The string "+1.00000E+00" then appears in the upper boxes.

C--PLACE BUTTONS
  CALL VCHAR(IBUTX(1),IBUTY,'CLEAR',5)
  CALL VCHAR(IBUTX(2),IBUTY,'PRINT',5)
  CALL VCHAR(IBUTX(3),IBUTY,'HARDCOPY',8)
  CALL VCHAR(IBUTX(4),IBUTY,'LOGOFF',6)
C--PLACE MESSAGE (IN LARGE CHARACTERS)
  CALL VSIZE(2)
  CALL VPOSIT(JMESAG)
  CALL VCHAR(240,900,' READY FOR ACTION ',20)
  CALL VSIZE(1)

Four buttons, labeled CLEAR, PRINT, HARDCOPY, and LOGOFF, are then placed near the top of the screen. After changing to the large character size, a message area is established, initially reading "READY FOR ACTION." The character size is then reset to normal.

C--SAVE POSITION FOR FUTURE TRUNCATION
  CALL VPOSIT(JTRUNC)

This much of the picture, although subject to alteration, will remain on the screen at all times. Beyond this point in the buffer, the displayed information will be dependent on the user's data. Therefore, it is at this position that we must truncate the buffer to clear user data, so we save the position in variable JTRUNC by calling VPOSIT.

  90 NPOINT=0
C--SEND PICTURE
  100 CALL VSEND
C--GET PEN DATA
  200 CALL VPEN(1,1,50,PENDAT,EXIT)
  GO TO (300,200,200,530,500,200), EXIT

Statement 90 zeroes NPOINT, the number of data points. At statement 100, the entire picture is sent to the screen. VPEN is then called to access user pen (or keyboard)data; the recognition mode is specified initially. Upon return from VPEN, a branch is made, depending on the exit code. On first pen-down, control passes to
statement 300; on a character exit, with or without data pending, control passes to statement 500; otherwise VPEN is reentered to await further action.

C--FIRST PEN DOWN
300 CALL VMODCH(JMESAG,1,')
    CALL VSEND
    IF (IYPEN.GE.IAXLOX .AND. IYPEN.LE.IAXHIY .AND. IXPEN.GE.IAXLOX
1 .AND. IXPEN.LE.IAXHIX) GO TO 3000
    DO 350 I=1,4
        IF (IXPEN.GE.IRNLOX(I) .AND. IXPEN.LE.IRNLOX(I)+174 .AND.
1 IABS(IYPEN-IRNCNY(I)).LE.15) GO TO 2000
350 CONTINUE
    GO TO 200

After receiving a first pen-down, the message area is cleared by calling VMODCH, and the updated picture is sent. If the pen-down occurred in the graph area, control passes to statement 3000. If it was in a range box, control passes to statement 2000 with the variable I set to the box number. Otherwise, the pen-down is of no importance, so VPEN is reentered.

C--CHARACTER EXIT
500 IF (IABS(IYPEN-IBUTY).LT.20) GO TO 1000
    GO TO 200
C--CHARACTER EXIT IN BUTTON AREA
1000 DO 1100 I=1,4
    IF (IXPEN.GE.IBUTX(I)-7 .AND. IXPEN.LE.IBUTX(I)+105)
1 GO TO 1200
1100 CONTINUE
    GO TO 200
1200 GO TO (1300,1400,1500,1600), 1
C--USER HIT CLEAR BUTTON
1300 CALL VCLEAC(JTRUNC)
    GO TO 90
C--USER HIT PRINT BUTTON
1400 WRITE (6,9000)
    IF (NPOINT.EQ.0) GO TO 1450
    WRITE (6,9001) (XPOINT(M),YPOINT(M),M=1,NPOINT)
1450 CALL VMODCH(JMESAG,1,' PRINTING COMPLETED ',20)
    GO TO 100
C--USER HIT HARDCOPY BUTTON
1500 CALL VCOPY(0)
    CALL VMODCH(JMESAG,1,' HARDCOPY COMPLETED ',20)
    GO TO 100
C--USER HIT LOGOFF BUTTON
1600 CALL VCOPY(4)
    CALL VMODCH(JMESAG,1,' PROGRAM TERMINATED ',20)
    CALL VSEND
    RETURN

On character exit, the program checks whether the character is on the button line. If not, all other actions having already been taken care of on pen-down, the character is ignored. When CLEAR is pushed, the program truncates the buffer to position JTRUNC and returns to statement 90 to zero NPOINT and sends the picture. When PRINT is pushed, the program writes a heading line and then prints the data values, if any. When this is done, it reports completion to the user by modifying the message area and returning to statement 100 to send the picture. The HARDCOPY
button works similarly, except that VCOPY is called to copy the buffer for processing on microfilm and paper. The LOGOFF button closes the hardcopy file, displays a message indicating program termination, and returns control to the operating system.

C--FIRST PEN-DOWN IN RANGE BOX 1
2000 CALL VPEN(I1,1,50,PENDAT,IEXIT)
2100 IF (IABS(IYPEN-IRNCY(I1))+.75*IRX(I1)) GT 1.75 GO TO 2200
IF (IEXIT.NE.4 .AND. IEXIT.NE.5) GO TO 2000

When a first pen-down is detected in a range box, control is passed to statement 2000. Here, VPEN is again called with the recognition option. When it returns, a check is made to ensure that the pen is still within the range box. If not, writing is assumed to be completed. If the exit was due to other than a recognized character, the program loops back to the VPEN call until one is found.

IF (IPENCHEQ.0) GO TO 2200
IF (IPENCEQ.ISCRUB) GO TO 2150
ILO=MAXO(IYPEN-IRX(I1)+11/14,1)
IHI=MINDO(IYPEN-IRX(I1)+11/14,12)
NCHAR=IHI-ILO+1
CALL VMODCH(I RANGE(I1),ILO,*,NCHAR)
GO TO 2190

2150 ICHAR=(IYPEN-IRX(I1)+11/14)
IF (ICHAR.LT.1 .OR. ICHAR.GT.12) GO TO 2200
CALL VMODCH(I RANGE(I1),ICHAR,PENCHEQ)

Having received a character, the program determines if it is a termination character or a scrub. A termination character, which results from a keyboard return or control key, is assumed to indicate that the user has finished writing. In the case of a scrub, the program uses the character width to erase all of the characters covered by the scrub. These characters (from the ILOth to the IHIth) are changed to blanks by VMODCH. If the character is normal, the program calculates its position in the box (ICHR) and uses VMODCH to change this position to the new character. Note the use of the two different character values from VPEN: the right-justified value (IPENCEQ) is easily tested by Fortran statements; the left-justified value (PENCHEQ) is convenient for passing to routines such as VMODCH.

2190 CALL VSEND
    IF (IEXIT.EQ.5) GO TO 2000
    CALL VPEN(I1,1,100,PENDAT,IEXIT)
    IF (IEXIT.NE.6) GO TO 2100

After sending the updated picture, the program determines whether further data are already pending—i.e., whether VPEN returned exit code 5. If so, a loop is made to statement 2000 to retrieve the next character. If no data were pending, we should give the user a little longer to write before redisplaying the curves. VPEN is therefore called with a time constant of one second. If it returns other than a time-elapsed exit code, the new data are processed exactly as before by the statements beginning at 2100. If the time has elapsed, the curves are updated by the code beginning at 2200.
2200 CALL VGETCH(JRANGE(1),1,TEMP,12)
   CALL CONVRT(2,12,TEMP,RANGE(1))
   IF (NPOINT.EQ.0) GO TO 4000
   IF (J.GT.2) GO TO 2500
   DELT=IAXHIX-1AXLX
   DO 2300 J=1,NPOINT
   ITMP=(XPOINT(J)-RANGE(1))*DELT/(RANGE(2)-RANGE(1))+IAXLX
   2300 IXPNT(J)=MINO(MAXO(ITMP,IAXLX),IAXHIX)
   GO TO 4000
2500 DELT=1AXHY-1AXLO
   DO 2600 J=1,NPOINT
   ITMP=(YPOINT(J)-RANGE(3))*DELT/(RANGE(4)-RANGE(3))+IAXLO
   2600 IYPNT(J)=MINO(MAXO(ITMP,IAXLO),IAXHY)
   GO TO 4000

After the user has completed changing the value in a range box, the program
extracts the value by calling VGETCH. The character string is converted to a floating-
point value by the utility subroutine CONVRT. (See Appendix B.) It is then necessary
to recompute the screen coordinates of the data points to reflect the new range box
value. (This step is bypassed if no points exist.) If one of the X-axis ranges was
changed, the DO-loop ending at step 2300 computes the updated positions for the
IXPNT table. If one of the Y-axis ranges was changed, a similar loop ending at step
2600 recomputes the IYPNT table. Once these arrays are updated, the program
branches to statement 4000 to redisplay the curves.

C--FIRST PEN-DOWN IN GRAPH AREA
3000 CALL VPOSIT(JMETR1)
   CALL VLINE(5,IXMETR,1YMETR)
   CALL VPOSIT(JMETR2)
   CALL VCHAR(500,500,TEMP,12)
   CALL VPOSIT(1METR3)
   CALL VCHAR(500,500,TEMP,12)

Control reaches statement 3000 when a first pen-down is received in the graph
area. The program first draws a "meter," which will follow the pen movement and
display the coordinates of its position. The meter comprises three subpictures, one
consisting of five joined line segments forming its outline, and two strings of twelve
characters that display the coordinates. (The coordinates initially assigned to the
meter subpictures are unimportant, as they will be changed by VMOVE before the
picture is sent.)

   IF (NPOINT.EQ.0) GO TO 3500
   DO 3100 I=1,NPOINT
   IF (((IXPNT(I)-IXPEN)**2+(IYPNT(I)-IYPEN)**2).LT.50) GO TO 3200
   3100 CONTINUE
   GO TO 3500
3200 XPOINT(I)=XPOINT(NPOINT)
   YPOINT(I)=YPOINT(NPOINT)
   IXPNT(I)=IXPNT(NPOINT)
   IYPNT(I)=IYPNT(NPOINT)
   NPOINT=NPOINT-1

If points already exist, the program determines whether the pen-down position
is near an existing point; if so, that point is deleted by replacing it with the last point
and reducing NPOINT.
3500 XVAL=(IXPEN-IAXLOX)*(RANGE(2)-RANGE(1)) / 700.*RANGE(1)
YVAL=(IYPEN-IAXLOY)*(RANGE(4)-RANGE(3)) / 700.*RANGE(3)
IXSAVE=IXPEN
IYSAVE=IYPEN
CALL FORMAT(3,12,5,XVAL,TEMP)
CALL VMOCH(JMETR2,1,TEMP,12)
CALL FORMAT(3,12,5,YVAL,TEMP)
CALL VMOCH(JMETR3,1,TEMP,12)
CALL VMOVE(JMETR1,(IXPEN,IYPEN)
CALL VMOVE(JMETR2,(IXPEN-181,1,IYPEN+10)
CALL VMOVE(JMETR3,(IXPEN-181,1,IYPEN-10)
CALL VSEND

The program next converts the pen coordinates to values in the user's coordinate system, XVAL and YVAL. It also saves the actual pen coordinates for use when the pen is raised. XVAL and YVAL are then converted into character strings (in E12.5 format) by utility routine FORMAT (described in Appendix B), and these strings are placed into the subpictures of the meter. All three meter subpictures are moved by VMOVE to position the tip of the meter at the pen location. The picture is then sent by VSEND.

CALLVPEN2,1,1,PENDAT,1EXIT)
IF (1EXIT.LE.2) GO TO 3500
IF (IXPEN.LT.IAXLOX .OR. IXPEN.GT.IAXHIX .OR. IYPEN.LT.IAXLOY
1 .OR. IYPEN.GT.IAXHIY) GO TO 3600
IF (NPOINT.EQ.50) GO TO 3900
NPOINT=NPOINT+1
 XPAR(NPOINT)=XVAL
YPAR(NPOINT)=YVAL
IXPNT(NPOINT)=IXSAVE
IYPTT(NPOINT)=IYSAVE

The program then calls VPEN to check the status of the pen. If it is still down, the same procedure is used to modify the meter values and move the meter. If the pen is up, the program determines whether its coordinates are within the graph area. If so, and if NPOINT is less than 50, a new point is added by incrementing NPOINT and tabling the pen coordinates and their values in the user's coordinate system. If 50 points already exist, a message is displayed and the point is ignored.
C--SORT THE POINTS
3600 IF (NPOINT.LE.1) GO TO 4000
   NPTM1=NPOINT-1
   DO 3800 I=1,NPTM1
      IP1=I+1
      MIN=I
      DO 3700 J=IP1,NPOINT
         IF (XPOINT(J).LT.XPOINT(MIN)) MIN=J
      3700 CONTINUE
      IF (MIN.EQ.1) GO TO 3800
      ATEMP=XPOINT(MIN)
      XPOINT(MIN)=XPOINT(I)
      XPOINT(I)=ATEMP
      ATEMP=YPONT(MIN)
      YPOINT(MIN)=YPONT(I)
      YPOINT(I)=ATEMP
      ITEMP=IXPNT(MIN)
      IXPNT(MIN)=IXPNT(I)
      IXPNT(I)=ITEMP
      ITEMP=IYPNT(MIN)
      IYPNT(MIN)=IYPNT(I)
      IYPNT(I)=ITEMP
   3800 CONTINUE
   GO TO 4000

The code between statements 3600 and 3800 sorts the arrays XPOINT, YPOINT, IXPNT, and IYPNT into ascending sequence, based on the value of XPOINT. The points can then be displayed as a graph with connected line segments.

3900 CALL VMDCCH('JMESAG,1,' TOO MANY POINTS ',20)
   CALL VSEND

At statement 3900, the message area is modified to indicate that the user has entered more than 50 points. This message is then sent to the screen.

C--DISPLAY THE CURVES
4000 CALL VTRUNC(JTRUNC)
   IF (NPOINT.EQ.0) GO TO 100
   DO 4100 I=1,NPOINT
4100 CALL VCHAR(IXPNT(I),IYPNT(I),',',1)
   IF (NPOINT.GT.1) CALL VJLINE(NPOINT-1,IXPNT,IYPNT)
   GO TO 100
9000 FORMAT (' ',X,' Y*///')
9001 FORMAT (1X,2F14.5)
END

At statement 4000 the buffer is truncated, eliminating all displayed points and lines. If NPOINT is not zero, an asterisk is displayed at each data point. If more than one point exists, the points are connected by line segments by calling VJLINE. Control then returns to statement 100 to redisplay the picture and await the next pen action.
Subroutine VRECT draws a rectangle, given the coordinates of the corners.

```
SUBROUTINE VRECT(IXLO,IXHI,IYLO,IYHI)
DIMENSION IX(5),IY(5)
IX(1)=IXLO
IX(2)=IXLO
IX(3)=IXHI
IX(4)=IXHI
IX(5)=IXHI
IY(1)=IYLO
IY(2)=IYHI
IY(3)=IYHI
IY(4)=IYLO
IY(5)=IYLO
CALL VJLINE(4,IX,IY)
RETURN
END
```

The coordinates are first stored in local arrays IX and IY in such a way that the entries correspond to the lower left corner, the upper left, the upper right, the lower right, and back to the lower left corner. The lines are then drawn via a single call to VJLINE. This subroutine provides a simple example of the way in which new, higher-level functions can be created by using the primitive functions provided by VPACK.
Appendix A

LISTING OF SAMPLE PROGRAM

```
DIMENSION BUFFER(1900)
DIMENSION PENDAT(6)
EQUIVALENCE (IXPEN, PENDAT(1)), (IYPEN, PENDAT(2)),
    (IWPN, PENDAT(3)), (IHPEN, PENDAT(4)), (IPENCh, PENDAT(5)),
    (PENCn, PENDAT(6))
DATA IAXL0X/225//, IAXH0X/925/, IAXL0Y/160/, IAXHY/860/
DIMENSION IRNLOX(4), IRNCNY(4)
DATA IRNLOX/144, 844, 32, 32/, IRNCNY/100, 100, 160, 960/
DIMENSION IBUTX(4)
DATA IBUTX/163, 363, 563, 763/, IBUTY/950/
DIMENSION IXMTR(6), IYMTR(6)
DATA IXMTR/500, 480, 312, 312, 480, 500/, IYMTR/500, 520, 520, 480, 480, 500/
DATA ISCRUB/50/
DIMENSION TEMP(3)
DIMENSION RANGE(4), JRGAMG(4)
DIMENSION XPOINT(50), YPOINT(50), XPNT(50), YPNT(50)

C--INITIALIZE VPACK
CALL VINIT('SAMPLE', BUFFER, 4000)
C--DRAW AXES
CALL VRECT(IAXLOX, IAXH0X, IAXLOY, IAXHY)
C--DRAW RANGE BOXES
DO 50 I=1,4
   CALL VRECT(IRNLOX(I), IRNLOX(I)+174, IRNCNY(I)-15, IRNCNY(I)+15)
   RANGE(I)=0.
   CALL VPOSIT(JRANGE(I))
   CALL VCHAR(IRNLOX(I)+10, IRNCNY(I), '+0.00000E+00', 12)
   IF (.EQ.1+CR. 1.EQ.3) GO TO 50
   RANGE(I)=1.
   CALL VMOCH(JRANGE(I)+1,2,1,1)
50 CONTINUE
C--PLACE BUTTONS
CALL VCHAR(IBUTX(1), IBUTY, 'CLEAR', 5)
CALL VCHAR(IBUTX(2), IBUTY, 'PRINT', 5)
CALL VCHAR(IBUTX(3), IBUTY, 'HARDCOPY', 8)
CALL VCHAR(IBUTX(4), IBUTY, 'LOGOFF', 6)
C--PLACE MESSAGE (IN LARGE CHARACTERS)
CALL VSIZE(2)
CALL VPOSIT(JMESAG)
CALL VCHAR(240,900, ' READY FOR ACTION ', 20)
CALL VSIZE(1)
```

41
C--SAVE POSITION FOR FUTURE TRUNCATION
   CALL VPOSIT(JTRUNC)
90 NPPOINT=0
C--SEND PICTURE
100 CALL VSEND
C--GET PEN DATA
200 CALL VPen(1,1,50,PENDAT,IEEXIT)
   GO TO (300,200,200,530,500,200), IEEXIT
C--FIRST PEN DOWN
300 CALL VMODCH(JMESAG,1,'    ',20)
   CALL VSEND
   IF (IYPEN.GE.IAXLOX .AND. IYPEN.LE.IAXHIX .AND. IX PEN.GE.IAXLOX
      1 .AND. IX PEN.LE.IAXHIX) GO TO 3000
   DO 350 I=1,4
      IF (IX PEN.GE.IRNLOX(I) .AND. IX PEN.LE.IRNUX(I)+174 .AND.
         1 IABS(IYPEN-IRNCNY(I)).LE.15) GO TO 2000
   350 CONTINUE
   GO TO 200
C--CHARACTER EXIT
500 IF (IABS(IYPEN-IBUTY).LT.20) GO TO 1000
   GO TO 200
C--CHARACTER EXIT IN BUTTON AREA
1000 IF (1100 .LT. I .AND. 1100 .GE. I+105)
      1 GO TO 1200
   1100 CONTINUE
   GO TO 200
1200 GO TO (1300,1400,1500,1600), 1
C--USER HIT CLEAR BUTTON
1300 CALL VTRUNC(JTRUNC)
   GO TO 90
C--USER HIT PRINT BUTTON
1400 WRITE (6,9000)
   IF (NPPOINT.EQ.0) GO TO 1450
   WRITE (6,9000) (XPOINT(M),YPOINT(M),M=1,NPOINT)
1450 CALL VMODCH(JMESAG,1,' PRINTING COMPLETED ',20)
   GO TO 100
C--USER HIT HARDCOPY BUTTON
1500 CALL VCOPY(0)
   CALL VMODCH(JMESAG,1,' HARDCOPY COMPLETED ',20)
   GO TO 100
C--USER HIT LOGOFF BUTTON
1600 CALL VCOPY(4)
   CALL VMODCH(JMESAG,1,' PROGRAM TERMINATED ',20)
   CALL VSEND
   RETURN
C--FIRST PEN-DOWN IN RANGE BOX 1
2000 CALL VPen(1,1,50,PENDAT,IEEXIT)
2100 IF (IABS(IYPEN-IRNCNY(I)).GT.10 .OR. IX PEN.LT.IRNLOX(I)
      1 .OR. IX PEN.GT.IRNUX(I)+174) GO TO 2200
      IF (IEEXIT.NE.4 .AND. IEEXIT.NE.5) GO TO 2000
      IF (IPENCH.EQ.0) GO TO 2200
      IF (IPENCH.NE.ISCRUB) GO TO 2150
      ILO=MAXO((IX PEN-IPEN/2-IRNLOX(I)+11)/14,1)
      IHI=MNO((IX PEN+IPEN/2-IRNUX(I)+11)/14,12)
      NCHAR=IHI-ILO+1
      CALL VMODCH(JRANGE(I),ILO,'    ',NCHAR)
   GO TO 2190

42
2150  ICHAR=(IXPEN-IRNLOX(I)+11)/14
    IF (ICHAR.LT.1 .OR. ICHAR.GT.12) GO TO 2200
    CALL VMODCH(IJRANGE(I),ICHAR,PENCHR,1)
2190  CALL VSEND
    IF (IEEXIT.EQ.5) GO TO 2000
    CALL VPDEN(1,1,100,PENDAT,IEEXIT)
2200  CALL VGETCH(IJRANGE(I),1,TEMP,12)
    IF (IXPNT(2).EQ.0) GO TO 4000
    IF (I.GT.2) GO TO 2500
    DELT=IAHXIX-IAXLOX
    DO 2300 J=1,NPOINT
       TEMP=(XPOINT(J)-RANGE(I))*DELT/(RANGE(2)-RANGE(I))+IAXLOX
       GO TO 4000
2300  IXPNT(J)=MINO(MAXO(TEMP,IAXLOX),IAHXIX)
    GO TO 4000
2500  DELT=IAXHIY-IAXLOX
    DO 2600 J=1,NPOINT
       TEMP=(YPOINT(J)-RANGE(3))*DELT/(RANGE(4)-RANGE(3))+IAXLOY
2600  IYPNT(J)=MINO(MAXO(TEMP,IAXLOY),IAXHIY)
    GO TO 4000
C-FIRST PEN-DOWN IN GRAPH AREA
3000  CALL VPPOSIT(JMETR1)
    CALL VJLINE(5,IXMETR,IYMETR)
    CALL VPPOSIT(JMETR2)
    CALL VCHAR(500,500,TEMP,12)
    CALL VPPOSIT(JMETR3)
    CALL VCHAR(500,500,TEMP,12)
    IF (NPOINT.EQ.0) GO TO 3500
    DO 3100 I=1,NPOINT
       IF (IXPNT(I)-IXPEN)**2+(IYPNT(I)-IYPEN)**2.LT.50) GO TO 3200
3100  CONTINUE
    GO TO 3500
3200  XPOINT(I)=XPOINT(NPOINT)
    YPOINT(I)=YPOINT(NPOINT)
    IXPNT(I)=IXPNT(NPOINT)
    IYPNT(I)=IYPNT(NPOINT)
    NPOINT=NPOINT-1
3500  XVAL=(IXPEN-IAXLOX)*((RANGE(2)-RANGE(1))/700.*RANGE(1))
    YVAL=(IYPEN-IAXLOY)*((RANGE(4)-RANGE(3))/700.*RANGE(3))
    IXSAVE=IXPEN
    IYSAVE=IYPEN
    CALL FORMAT(3,12,5,XVAL,TEMP)
    CALL VMODCH(JMETR2,1,TEMP,12)
    CALL FORMAT(3,12,5,YVAL,TEMP)
    CALL VMODCH(JMETR3,1,TEMP,12)
    CALL VMODEL(JMETR1,IXPEN,IYPEN)
    CALL VMODEL(JMETR2,IXPEN-181,IYPEN+10)
    CALL VMODEL(JMETR3,IXPEN-181,IYPEN-10)
    CALL VSEND
    CALL VPEN(1,1,1,PENDAT,IEEXIT)
    IF (IEEXIT.LE.-2) GO TO 3500
    IF (IXPEN.LT.IAXLOX .OR. IXPEN.GT.IAHXIX .OR. IYPEN.LT.IAXLOY
     1 .OR. IYPEN.GT.IAXHIY) GO TO 3600
    IF (NPOINT.EQ.50) GO TO 3900
    NPOINT=NPOINT+1
    XPOINT(NPOINT)=XVAL
    YPOINT(NPOINT)=YVAL
    IXPNT(NPOINT)=IXSAVE
    IYPNT(NPOINT)=IYSAVE
C--SORT THE POINTS
3600 IF (NPOINT.LE.1) GO TO 4000
   NPTM1=NPOINT-1
   DO 3800 I=1,NPTM1
      IP1=I+1
      MIN=I
      DO 3700 J=IP1,NPOINT
         IF (XPOINT(J).LT.XPOINT(MIN)) MIN=J
      3700 CONTINUE
   IF (MIN.EQ.I) GO TO 3800
   ATEMP=XPOINT(I)
   XPOINT(I)=XPOINT(MIN)
   XPOINT(MIN)=ATEMP
   ATEMP=YPOINT(I)
   YPOINT(I)=YPOINT(MIN)
   YPOINT(MIN)=ATEMP
   ITEMP=IXPNT(I)
   IXPNT(I)=IXPNT(MIN)
   IXPNT(MIN)=ITEMP
   ITEMP=IYPNT(I)
   IYPNT(I)=IYPNT(MIN)
   IYPNT(MIN)=ITEMP
3800 CONTINUE
   GO TO 4000
3900 CALL VMGCH(JMESAG,1,' TOO MANY POINTS ',20)
   CALL VSSEND
C--DISPLAY THE CURVES
4000 CALL VTRUNC(JTRUNC)
   IF (NPOINT.EQ.0) GO TO 100
   DO 4100 I=1,NPOINT
4100 CALL VCHAR(IXPNT(I),IYPNT(I),'**1')
   IF (NPOINT.GT.1) CALL VJLINE(NPOINT-1,IXPNT(IYPNT))
   GO TO 100
9000 FORMAT ('1 X',Y/2)
9001 FORMAT (1X,2F14.5)
END
Appendix B

UTILITY ROUTINES

The following routines are provided in Rand's subroutine library. Although not related directly to VPACK, they may prove useful for manipulating character strings, a common requirement of VPACK programs.

MOVECH: Moves Characters

CALL MOVECH (SOURCE, ISOR, N, DEST, IDEST)

MOVECH moves N characters, beginning with the ISORth position of string SOURCE, to string DEST, beginning at the IDESTth position.

LXTRAC: Extracts and Left-Justifies a Character

IOUT = LXTRAC(STRING, ISTR)

LXTRAC extracts the ISTRth character from string STRING and stores it in the first position of variable IOUT. The second, third, and fourth positions of IOUT are set to blanks.

IXTRAC: Extracts and Right-Justifies a Character

IOUT = IXTRAC(STRING, ISTR)

IXTRAC extracts the ISTRth character from string STRING and stores it in the fourth position of variable IOUT. The first, second, and third positions of IOUT are zeroed.

45
NCOMP: Compares Character Strings

\[ \text{IOUT} = \text{NCOMP} (\text{ONE}, 11, N, \text{TWO}, 12) \]

NCOMP compares N characters of character string ONE, beginning with position 11, with N characters of string TWO, beginning with position 12. Standard EBCDIC collating sequence is used. The result IOUT is set to: -1 if ONE is less than TWO, 0 if ONE is identical to TWO, or +1 if ONE is greater than TWO.

FORMAT: Converts a Number to a Character String

CALL FORMAT (IFMT, IW, ID, VALUE, CHAR$)

FORMAT converts a number, either real or integer, to a character string in I,F, or E format. IFMT specifies the type of conversion to be performed: 1 to convert an integer to I format, 2 to convert a real number to F format, or 3 to convert a real number to E format. IW is the field width of the converted number. ID specifies the number of decimal places; ID = 0 if IFMT = 1. VALUE contains the number to convert; it may be either real or integer, depending on the value of IFMT. CHAR$ will contain the resulting character string; it must be an array large enough to hold at least IW characters.

CONVRT: Converts a Character String to a Number

CALL CONVRT (IFMT, IW, STRING, OUT)

CONVRT converts IW characters of the character string STRING to a number, either real or integer. IFMT specifies the conversion: 1 to convert to an integer; 2 to convert to a real; or 3 to convert to a real if a decimal point or the letter E appears in the string, but to an integer otherwise. On exit, OUT contains the converted value, either real or integer.
REFERENCES


