1976-03

Users manual for the vector general graphics display unit

Thorpe, Lloyd Allen
Monterey, California. Naval Postgraduate School

http://hdl.handle.net/10945/29586

Downloaded from NPS Archive: Calhoun
NAVAL POSTGRADUATE SCHOOL
Monterey, California

USERS MANUAL FOR THE VECTOR GENERAL GRAPHICS DISPLAY UNIT

by
L. A. Thorpe
G. M. Raetz

March 1976

Approved for public release; distribution unlimited.

Prepared for: Naval Electronics Systems Command
(ELEX 320)
Washington, D. C.
REPORT DOCUMENTATION PAGE

1. REPORT NUMBER
NPS-72Rr76031

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)
USERS MANUAL FOR THE VECTOR GENERAL
GRAPHICS DISPLAY UNIT

5. TYPE OF REPORT & PERIOD COVERED
Technical Report

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)
L. A. Thorpe
G. M. Raetz

8. CONTRACT OR GRANT NUMBER(s)

9. PERFORMING ORGANIZATION NAME AND ADDRESS
Naval Postgraduate School
Monterey, California 93940

10. PROGRAM ELEMENT, PROJECT, TASK
AREA & WORK UNIT NUMBERS
N0003975WR 59051
PDM 000320

11. CONTROLLING OFFICE NAME AND ADDRESS
Naval Electronics Systems Command
(ELEX 320)
Washington, D. C. 20360

12. REPORT DATE
15 March 1976

13. NUMBER OF PAGES
85

14. MONITORING AGENCY NAME & ADDRESS (IF different from Controlling Office)
Computer Science Group (Code 72)
Naval Postgraduate School
Monterey, California 93940

15. SECURITY CLASS. (of this report)
Unclassified

15a. DECLASSIFICATION/ Downgrading
SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)
Approved for public release; distribution unlimited.

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

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)
Real time
Interactive graphics
Graphics

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)
This report is the description of a users manual for an interactive graphics package designed to allow programmatic control of the Vector General Graphics Display Unit from a Digital Equipment Corporation PDP-11/50 computer. The manual requires a knowledge of the C-programming language and a general familiarity with graphics terminology. Included is a brief description of the Vector General Graphics Display Unit, a
20. (cont.)

description of the interface routines, and a description of the Vector General graphics display instructions.
ABSTRACT

This report is the description of a users manual for an interactive graphics package designed to allow programmatic control of the Vector General Graphics Display Unit from a Digital Equipment Corporation PDP-11/50 computer. The manual requires a knowledge of the C-programming language and a general familiarity with graphics terminology. Included is a brief description of the Vector General Graphics Display Unit, a description of the interface routines, and a description of the vector General graphics display instructions.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I.</td>
<td>INTRODUCTION</td>
<td>4</td>
</tr>
<tr>
<td>II.</td>
<td>VECTOR GENERAL DISPLAY SYSTEM</td>
<td>5</td>
</tr>
<tr>
<td>A.</td>
<td>VISIBLE SPACE</td>
<td>7</td>
</tr>
<tr>
<td>B.</td>
<td>PICTURE SPACE</td>
<td>9</td>
</tr>
<tr>
<td>C.</td>
<td>IMAGE SPACE</td>
<td>11</td>
</tr>
<tr>
<td>D.</td>
<td>THREE-DIMENSIONAL DISPLAY</td>
<td>11</td>
</tr>
<tr>
<td>E.</td>
<td>PICTURE CONTROL</td>
<td>12</td>
</tr>
<tr>
<td>F.</td>
<td>CHARACTER GENERATION</td>
<td>18</td>
</tr>
<tr>
<td>G.</td>
<td>CONTROL CHARACTERS</td>
<td>21</td>
</tr>
<tr>
<td>III.</td>
<td>CONTROL DEVICE DESCRIPTION</td>
<td>21</td>
</tr>
<tr>
<td>A.</td>
<td>ALPHANUMERIC KEYBOARD</td>
<td>21</td>
</tr>
<tr>
<td>B.</td>
<td>LIGHTED FUNCTION SWITCHES</td>
<td>22</td>
</tr>
<tr>
<td>C.</td>
<td>LIGHT PEN</td>
<td>22</td>
</tr>
<tr>
<td>D.</td>
<td>CONTROL DIALS</td>
<td>23</td>
</tr>
<tr>
<td>IV.</td>
<td>USING THE VECTOR GENERAL</td>
<td>23</td>
</tr>
<tr>
<td>V.</td>
<td>DISPLAY INSTRUCTIONS</td>
<td>27</td>
</tr>
<tr>
<td>A.</td>
<td>WORD FORMATS</td>
<td>28</td>
</tr>
<tr>
<td>B.</td>
<td>CONTROL DISPLAY INSTRUCTIONS</td>
<td>29</td>
</tr>
<tr>
<td>1.</td>
<td>No Operation</td>
<td>29</td>
</tr>
<tr>
<td>2.</td>
<td>Halt</td>
<td>29</td>
</tr>
<tr>
<td>C.</td>
<td>DISPLAY WRITE INSTRUCTIONS</td>
<td>30</td>
</tr>
<tr>
<td>1.</td>
<td>Vector Relative</td>
<td>31</td>
</tr>
<tr>
<td>2.</td>
<td>Vector Relative Auto-x</td>
<td>34</td>
</tr>
<tr>
<td>3.</td>
<td>Vector Relative Auto-y</td>
<td>35</td>
</tr>
</tbody>
</table>
I. INTRODUCTION

This manual is a user oriented description of a C-callable interactive graphics package designed to allow programmatic control of the Vector General Graphics Display Unit (vector general). A knowledge of the C-programming language and a general familiarity with graphics terminology are assumed.

The vector general is an interactive graphics display system interfaced with the PDP-11/50 computer. The display interacts with a PDP-11 user by displaying vectorial information on the surface of a cathode ray tube and by accepting inputs from external control devices. The inputs are requested and processed by PDP-11 computer programs that alter and maintain the output picture being presented to the user.

With the permission of Vector General Corporation, portions of the material in sections II, III, and V have been reproduced from the Vector General Graphics Display Unit Reference Manual (VG 10105a).
II. VECTOR GENERAL DISPLAY SYSTEM

The vector general contains the following features for interactive interactive graphics display:

- Interface unit
- Display controller
- Digital to analog converter
- Vector generator
- Display monitor
- Character generator
- Circle/Arc generator
- Three dimension coordinate transform generator

Four interactive control devices are connected to the system. The devices are:

- Alphanumeric keyboard
- Thirty-two lighted function switches with manual interrupt
- Ten control dials
- Light pen

The computer and display controller communicate by way of the interface unit through three types of channels. These channels are:
Data Channel - Direct memory access channel. Used by the vector general to directly access memory without interfering with the operation of the PDP-11 central processor.

Programmed Input/Output Channel - Used to start the controller and acknowledge interrupts.

Interrupt Channel - Used by the display to interrupt the PDP-11 computer.

The display controller processes all display functions while running asynchronously with the PDP-11 central processor. The controller also receives inputs from the external control devices.

The digital to analog converter converts the digital values from the display controller into analog signals for use in the vector generator.

The cathode ray tube generates an electron beam that shows as a spot of light on the face of the tube. An electromagnetic deflection system causes the spot to move in a direction on the tube face in response to signals from the vector generator. An input from the vector generator causes the brightness of the spot to vary.

The method of controlling the movement of the spot is called the raster scan method. This involves steering the spot in a straight line between two points on the display screen. A series of these straight lines constitute an image. To present a clear image, the pattern traced on
the tube must be repeated approximately thirty to forty times per second. Each repetition is called a frame and the frequency at which it is generated is called the refresh rate. The default rate is forty hertz but can be varied from thirteen to one hundred twenty hertz by the routine vaclock().

A. VISIBLE SPACE

The rectangular portion of the CRT which can be viewed by a user is called the visible space. The visible space is limited by an opaque mask with a rectangular cutout. See Figure 2-1.

The maximum picture space is larger than the visible space. This permits limited zooming but primarily allows fully visible objects to be rotated and positioned to the extreme limits of the visible space and yet draw any remaining visible portions without distortion.

The picture being generated is adjusted in size (scaled) to present the desired output by means of two controls:

a. The manually adjustable gain controls on the CRT deflection hardware.

b. The program controlled picture scale transformation hardware. See pscal() routine in section VI.
Dynamic Range of Vector General = Picture Space

Figure 2-1. Image Areas
The picture can be generated on a picture space coordinate system and scaled for viewing through the visible space. See vgoost() and vpscal() routines in section VI.

With the gain control knobs at standard midrange calibrated settings, the maximum picture space (over which the vector generator accurately reproduces images) is a thirty inch by thirty inch plane of which the visible space is a thirteen inch by fourteen inch rectangle in the center. See Figure 2-1.

II. PICTURE SPACE

The transformation hardware permits the coordinates defining an element to be transformed prior to display generation. For the input coordinates \((x, y, z)\) the output transformed \(x\) and \(y\) are used to generate the element's horizontal and vertical picture space position respectively. Thus, the picture space is the \(X-Y\) projection of the transformed element definition.

If no transformation is performed, or for zero rotation, zero displacement, and full scale size transformation, an element coordinate \((X, Y, Z)\) will correspond directly to the picture space \((X, Y)\), with the positive \(X\) term horizontal towards the right of a viewer and positive \(Y\) being vertical. With the gain knobs at the calibrated settings and the picture scale set to maximum
(1.0), a plus full scale X element coordinate value (+2047) transforms into an X picture space coordinate value which corresponds to a horizontal displacement of seven and one-half inches to the right of center or one-half to the right of the visible space. Similarly, for no transformation and maximum picture scale, a full scale Y element coordinate value (+2047) corresponds to a picture space position seven and one-half inches up from the center.

To view a centered two-dimensional object defined over the entire X-Y coordinate range (such as a page of text), the picture scale can be set to 0.92 or the gain knobs turned down. To view an entire centered three-dimensional object which is defined over the entire (X, Y, Z) image space, a 0.577350 factor is needed to view the maximum length of the projected diagonals of the image space. The picture scale is defined as 1.0 unless explicitly changed by the routine vooscal().

Since the picture space is larger than the visible space, each element may be positioned out of the viewing area in any direction without distorting any remaining visible portions. This capability is termed the hardware scissoring facility.
C. IMAGE SPACE

Prior to transformation and projection onto the picture space an element is defined in a coordinate system referred to as the image space. All separately transformed objects of a displayed picture are defined in their respective untransformed image spaces.

The image space coordinate system is defined with positive X horizontal to the right of the viewer and Y being vertical. The Z axis is perpendicular to the X-Y plane and intersects the X and Y axes at the zero point. Positive Z is toward the viewer. The image space is a fifteen inch by fifteen inch rectangle. See Figure 2-1.

To exploit maximum use of transformation ranges and coordinate resolution, all elements should be defined as large as possible. Elements are defined primarily in terms of generated visual elements: Vectors and Characters.

D. THREE-DIMENSIONAL DISPLAY

Three-dimensional presentation involves a third, or Z, axis that is perpendicular to the face of the screen and intersects the X and Y picture space axis at the zero point. The Z axis represents depth into and out of the display screen. The illusion of depth may be achieved by
varying the light intensity of the fluorescent spot in proportion to the value of the Z coordinate. The intensity increases exponentially with the intensity value over the range of minus full-scale intensity to one-half full-scale intensity. Maximum intensity is at the face of the screen.

E. PICTURE CONTROL

The picture control is used for picture transformation after all transformations of individual objects have been completed. The registers used for this feature are the twelve bit intensity offset register, the twelve bit intensity scale register, and the twelve bit picture scale register. The value in the picture scale register is multiplied by each of the transformed X, Y, and Z coordinates to establish the final picture size. This scaling applies also to characters in the picture.

The intensity scale register is used in conjunction with the intensity offset register to provide depth cueing, or shading of the intensity of the picture according to the values of the Z coordinate. The hardware calculations of the spot intensity at any instant can be represented by the following C-programming language routine:
int is;  // intensity scale register value
int io;  // intensity offset register value
int i;  // resulting intensity
int imax;  // maximum intensity
int z;  // transformed Z coordinate value
int zl;
int k;  // constant
int e;

z1 = 0.5 * abs(is) * z + io;
if (is >= 0 && z1 <= 0.02) i = imax * e ** (k * z1);
if (is >= 0 && z1 > 0.02) i = saturation;
if (is < 0 && z1 <= 0.02) i = imax * e ** (k * z1);
if (is < 0 && is > 0.02) i = 0;

The intensity cutoff plane is established by the value in the intensity offset register. Within the depth range of an image, the intensity is blanked between the viewer and the screen. The intensity is at its maximum at the face of the screen and decreases exponentially with decreasing values of Z toward the back of the image. Figure 2-3a and Figure 2-3b show the effect of intensity offset variation. As the value in the intensity offset register is changed, the element moves forward or backward through the intensity range, to vary the section that is intensified and the part that is blanked out.

The intensity range, or apparent depth of the image, is determined by the value in the intensity scale register. If the value is 1.0, the maximum intensity is achieved. If the value is zero, the intensity is constant and the element has no depth-cuing. Figure 2-4a and Figure 2-4b show how a variation in intensity scale changes the depth of the element.
Figure 2-3A. Intensity Offset Variation
Figure 2-33. Effect of Intensity Offset Variation
Figure 2-4A. Intensity Scale Variation
Large Intensity Scale:

Back Much Dimmer Than Front

Maximum Intensity

Back Dim Front Bright

Small Intensity Scale:

All Nearly the Same Brightness

Maximum Intensity

Figure 2-4B. Effect of Intensity Scale Variation
F. CHARACTER GENERATION

The character generator accepts coded inputs from the display list and produces text strings composed of ASCII characters and special characters. Characters are drawn on the screen as a series of short vectors and curves. Unlike the vector generator, however, the character generator draws are generated automatically by the character generator each time a character code is received.

The program can select one of four character sizes. The intensity is determined by the intensity scale register. Continuous character scaling allows the picture scale and coordinate scale to scale the image and characters proportionally. The program can also specify whether the text lines are to be displayed horizontally on the screen or are to be positioned as if on a page that has been rotated ninety degrees clockwise. One of the characters is a cursor. The cursor differs from other displayed characters in that the character following the cursor is drawn in the same place without a column feed. This permits the cursor to be moved over the screen as desired with manual inputs. A hardware feature causes the cursor to blink twice per second.

The dimensions for character generator outputs in number space units are given in Figure 2-5. The standard character set font is shown in Figure 2-6. The codes for each character can be found in Appendix A.
Figure 2-5. Character Generator Outputs
G. CONTROL CHARACTERS

Fourteen codes in the character set are used for control purposes only and do not cause a display on the screen. The control characters and their functions are listed in section V.

III. CONTROL DEVICE DESCRIPTION

A. ALPHANUMERIC KEYBOARD

The alphanumeric keyboard is used as an entry device for manual input to the display system. Pressing a key on the keyboard enters an eight bit ASCII character code into the keyboard character queue. The character entered in the keyboard character queue does not directly affect the display on the screen. The program can read the keyboard character via vqetcar() and use the information in its operation. Holding any key down will maintain the correct code and, after an initial delay, will repeat the character. Appendix A lists the codes generated by the keyboard for shifted and unshifted key combinations.
B. LIGHTED FUNCTION SWITCHES

This device contains thirty-two function switches plus a manual interrupt switch. The function switch registers in the display controller have one bit corresponding to each function switch. While any function switch is depressed, the corresponding bit in the function switch register is set. The program can then read the contents of the registers via vnoontfsw(). The function switch lamps can be lighted via volamps().

The manual interrupt switch can be used to cause an interrupt. This feature allows the operator to intercept the program at any desired point. When the manual interrupt switch is pressed, the manual interrupt counter, vmanint, is incremented. The program may reset this counter when desired.

C. LIGHT PEN

The light pen can be used to point at an element of a display or to create information by drawing on the display. The light pen, a wand containing a photocell, is held over the face of the cathode ray tube by the viewer. When the light pen is held over a line or point on the display, the light pen interrupt flag, volnflag, is set to indicate a light pen interrupt condition. If the light
pen switch is activated, the light pen sense switch interrupt flag, voPsFlg, is set to indicate this condition.

A hardware delay feature permits the light pen interrupt state of the Vector General to be obtained prior to additional display processing. The values retained at interrupt time can be obtained by calling vaoetlpn().

D. CONTROL DIALS

Ten control dials may be used to send digital numerical information to the computer for any purpose specified by the program. Each dial is associated with a twelve bit dial input register in the display controller. As a dial is turned, the corresponding register is updated to reflect the dial value. These values may be read at any time by calling vvodial().

IV. USING THE VECTOR GENERAL

The vector general interface design concept is to define high level constructs which the interface routines convert into vector general commands. There are three classes of constructs defined: objects, elements, and the picture. An object is the lowest level construct which
can be displayed alone. Each object is independently rotatable, scalable, and translatable into any portion of the thirty inch by thirty inch picture space. An object can be as large as fifteen inches by fifteen inches and be rotated or positioned to the extreme limits of the picture space without distortion to any of the remaining visible portion. Each object is composed of one or more independently light pen hookable elements. An element is composed of a series of user-drawn images or characters relative to the untransformed image space of its object. An object can be defined unrotated in such a way as to fill the entire object space and then be scaled, rotated, and moved so that the image space is the appropriate size, is viewed from the appropriate aspect, and is in the appropriate area of the picture. The picture defines the picture scale and screen coordinates for all objects.

The user is responsible for the generation and content of each element. Prior to its inclusion within the display list, the user must fill each element with the necessary draw and move commands. In addition, the user must provide three unused words succeeding the draw-move commands. These three words are needed by the interface routines for proper display list termination.

The generation and contents of all objects and the picture is the responsibility of the interface software. A set of routines are provided to link elements to objects and objects to the picture. Dynamic modification of
objects and picture parameters is also provided. The routines for manipulation and modification of the display data structure are:

- `voaddele(abo,num,size)`
- `voblink(type,num,action)`
- `voclock(rate)`
- `vocoord(num,x,y,z)`
- `vocrsr(num,val)`
- `vodelele(num)`
- `vodeleorj(num)`
- `voidal(abo)`
- `voietcar()`
- `voetfsw(abo)`
- `voetion(abo)`
- `voinit()`
- `vqioffset(num,val)`
- `voiscal(num,val)`
- `volamps(abo,action)`
- `vqlen(type,num,action)`
- `vomkorj()`
- `vopicture()`
- `vopost(px,py)`
- `voyscale(val)`
- `vqpost(px,py)`
- `vqrotate(num,x,y,z)`
- `voterm()`

All picture and object display parameters are initialized to default values. The user may then modify these parameters as desired. The default parameters are:

- **Display Blink**: The picture, all objects, and all elements are defined as non-blink. `voblkink()` can set or clear the blink mode on any of the constructs.
- **Coordinate Position**: The X, Y, and Z coordinates of each object are (0,0,0). The routine `vocoord()` can move an object to any area of the image space by varying these coordinate values.
- **Picture Position**: The location of the picture coordinates is defined as the center of the screen.
(0,0,0). The entire picture can be moved on the X-Y plane by calling vqpost().

Coordinate Scale - The default is one, the maximum size. This parameter may be be scaled between zero and one by calling vocsr().

Picture Scale - The default is one. Reducing the picture scale will shrink all of the display. voscal() modifies this parameter.

Intensity Offset - The maximum intensity offset, value of one, is set as the default parameter. A call to vooffset() can reduce this value.

Intensity Scale - Initially set to a one, this maximum value can be reduced by the routine voscal().

Refresh Rate - A refresh rate of forty hertz has been assigned as the default value. vlock() can modify this value.

Light Pen Enable - The light pen is initially disabled. A call to volpen() will enable it.

Function Switch Lamps - All function switch lamps are extinguished at display time and can be set by the routine volamps().

In order to compile a program using the display subroutines, global names, and structures the user should invoke the following statement:

```
cc -f -O filename -lv
```
This will cause the lexical scope of the vector general display system to extend through the program.

Certain global names and routines are not visible to the user. To avoid colliding with them, the user should refrain from defining external variables and routines that start with the letters "vo".

The user must preface his display actions by a call to voinit() and should call v任期() before process termination.

V. DISPLAY INSTRUCTIONS

Instructions used in the display system fall into two main types: control instructions (used to control the display operation) and output instructions (used to generate image vectors or characters on the display screen).

A description of the control instructions and their effect on the vector general can be found in the Vector General Graphics Display Unit Reference Manual (VG 101050).

The following paragraphs contain functional descriptions of the various display-list instruction configurations processed by the display system. Each instruction discussion includes a format diagram, a listing of the octal codes for the instruction variations, a definition
of the applicable code, and a description of the purpose of the instruction.

The octal code given assumes the instruction to be an eighteen bit instruction with the first two bits zero.

Operation of the display system consists of processing data words in accordance with their instructions. Instructions that draw lines or manipulate text strings process the data words following the instruction command. The data words give the end point coordinates of the lines or character codes of the text.

Data words are transmitted in a string or block following the applicable instruction. The last data word in the string must contain a coded terminate bit, field, or character to indicate that it is the last word for that instruction.

4. WORD FORMATS

The display system uses as its basic informational element a sixteen bit word with the bit positions numbered zero through fifteen as shown in the following diagram. Bit zero is the most significant bit.
B. CONTROL DISPLAY INSTRUCTIONS

1. No Operation

```
000000
```

The **N0OP** instruction may be used to hold data or addresses. The information in bits 4-15 is not interpreted by the vector general.

2. Halt

```
030000
```

The **HLT** instruction causes the display system to cease all operations. No further instructions or data words are
accepted. The display system state is set to not-run, not-wait. This command does not disable any of the vector general interrupts.

C. DISPLAY WRITE INSTRUCTIONS

These display instructions and their following data are output as lists over the direct memory access channel to generate visual display elements. The basic word format is as shown in the following diagram.

```
<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1</td>
</tr>
<tr>
<td>Character or Vector Data List</td>
</tr>
</tbody>
</table>
```

The image generation instructions are used to present display elements consisting of solid lines, dashed lines, dotted lines, or dash-dot-dashed lines between two positions on the display screen.

The modifier bits (12-15) of the image generation instruction specify if the data words that follow it are to be used for characters, absolute or relative vectors, X, Y, or 2D auto-incrementing, or 2D or 3D incremental vectors. The instruction also indicates the type of display (normal, dashed, dot, or point) and the incremental
resolution or character scaling to be used.

The character generation instruction indicates the size of the characters to be displayed and whether they are to be displayed horizontally or vertically.

The following descriptions are given for no transformations imposed on the generated image prior to display. The user must specify any desired transformation prior to processing any display generating instructions whose output is to be affected.

1. **Vector Relative**


```
010000
```

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>1</th>
<th>VM</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>±</td>
<td>Δ Coordinate</td>
<td></td>
<td>OF</td>
<td>CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>±</td>
<td>Δ Coordinate</td>
<td>1</td>
<td>1</td>
<td>CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The display instruction for relative vectors generates a vector display whose coordinates are relative to the initial contents of the coordinate registers, $X_R$, $Y_R$, and $Z$. The type of display generated by the moving beam is specified by the vector mode field ($VM$).
VECTOR MODE

VM
000  Line
001  Dashed Line
010  Dotted Line
011  End Point
101  Dash-Dot-Dash
110  Dash-Dot-Dash

The operation field (OF) of each data word specifies if the beam is to be moved to a new position held in the coordinate registers; also, when moving the beam, it specifies if a vector type (V) is to be drawn. The OF field also specifies the end of the data list.

OF
00  Load Register
01  Load, then draw vector
10  Load, move beam (no draw)
11  Load, draw, terminate

Each data word has a signed twelve bit coordinate increment to be added to a coordinate register or to the auto-increment register (AIR). The coordinate field (CF) of each data word specifies which register is to be updated by the coordinate increment.

CF
00  Auto-increment register (AIR)
To draw an arc with the circle arc generator use the following procedure:

1. Move the beam or draw to the start point of the arc.
2. Load the end point into the coordinate registers.
3. Issue an arc command. The arc commands are:

<table>
<thead>
<tr>
<th>Clockwise arc</th>
<th>Counter clockwise arc</th>
</tr>
</thead>
<tbody>
<tr>
<td>000004</td>
<td>000010</td>
</tr>
</tbody>
</table>

4. Issue a draw or draw-terminate command to the center point of the arc. This defines the radius of the arc.

A circle may be drawn by either using the procedure described above and specifying the same value for both the start point and end point of the arc or by leaving step two of the procedure described above out entirely and just moving or drawing to the start point before issuing the arc command.

If the radius to start and to the end point are not equal, the start point will be used and the circle arc
generator will use either the X or Y coordinate for terminating the draw in accordance with the following diagram. A straight line will be drawn from the terminating point on the radius and the specified end point.

![Diagram showing use of X or Y coordinate for drawing]

2. **Vector Relative Auto-X**

![Table showing vector relative auto-X instructions]

The display instruction for vector relative auto-X is processed as a relative vector. Each coordinate value is added to the register designated by CF; then the vector generator performs any function specified by VM and CF. With each move or draw operation, the X-coordinate register (Xr) is incremented by the value in the auto-increment
register (AIR) following the load instruction but preceding the move or draw command of the operation.

The type of vectors generated is specified by VM as described above.

Control of beam motion and blanking or list termination is specified by OF as described for relative vectors.

Specification of the register to be incremented by the coordinate value is given by CF as described above.

3. Vector Relative Auto-Y

![Table]

<table>
<thead>
<tr>
<th>VM</th>
<th>OF</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 1</td>
<td>0 0 1 0</td>
<td>1 1 CF</td>
</tr>
</tbody>
</table>

The display instruction for vector relative auto-Y is processed as a relative vector. Each coordinate value is added to the register designated by CF; then the vector generator performs any function specified by VM and OF.

With each move or draw operation the Y-coordinate register (YP) is incremented by the value in the auto-increment register (AIR) following the load operation but preceding the move or draw portion of the operation. The VM, OF, and CF fields are as described for relative vectors.
The display instruction for vector relative auto-Z is processed as a relative vector. Each coordinate value is added to the register designated by CF; then the vector generator performs any function specified by VM and OF. With each move or draw operation, the \( Z \) coordinate register \((Z^2)\) is incremented by the value in the auto-increment register \((AIK)\) following the load portion but preceding the move or draw portion of the operation. The VM, tF, and CF fields are as described for relative vectors.
5. Vector Absolute

<table>
<thead>
<tr>
<th>VM</th>
<th>OF</th>
<th>CF</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>± Coordinate</td>
<td></td>
<td></td>
</tr>
<tr>
<td>± Coordinate</td>
<td>1 1</td>
<td>CF</td>
</tr>
</tbody>
</table>

The vector absolute display instruction loads the coordinate value from each of its data words directly into the register specified by CF, replacing the previous contents. The beam position is moved if called for by OF and a vector of type VM is drawn if required by CF. The VM, OF, and CF fields for absolute vectors are the same as described for vector relative.

To draw an arc using the circle arc generator use the procedure described for vector relative.
The display instruction for vector absolute auto-X processes its data list as absolute vectors. Each coordinate value is loaded into the register designated by CF; then the vector generator performs any move or VM type draw operation if called for by CF. With each move or move operation, the X-coordinate register (XR) is incremented by adding the value from the auto-increment register (AIX) following the load portion but preceding the move or draw portion of the operation. The VM, OF, CF fields are used as described for vector relative.
7. **Vector Absolute Auto-Y**

01000b

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>VM</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>±</td>
<td>Coordinate</td>
<td>OF</td>
<td>CF</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The display instruction for vector absolute auto-Y processes its data list as absolute vectors. Each coordinate value is loaded into the register designated by CF; then the vector generator performs any move or VM type draw operation if called for by OF. With each draw or move operation, the i-coordinate register (IR) is incremented by adding the value from the auto-increment register (AIR) following the load portion but preceding the move or draw portion of the operation. The VM, OF, CF fields are used as described for vector relative.
8. Vector Absolute Auto-Z

010007

<table>
<thead>
<tr>
<th>0 0 0 1</th>
<th>VM</th>
<th>0 1 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>± Coordinate</td>
<td>OF</td>
<td>CF</td>
</tr>
</tbody>
</table>

The display instruction for vector absolute auto-Z processes its data list as absolute vectors. Each coordinate value is loaded into the register designated by CF; then the vector generator performs any move or V1-type draw operation if called for by OF. With each move or move operation, the Z-coordinate register (Zr) is incremented by adding the value from the auto-increment register (ALK) following the load portion but preceding the move or draw portion of the operation. The VM, OF, CF fields are used as described for vector relative.
9. Incremental Vectors, 2D

010010

<table>
<thead>
<tr>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
<th>13</th>
<th>14</th>
<th>15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>S</td>
<td>VM</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The 2D incremental vector display instructions generate an XY vector display whose coordinates are relative to the initial contents of the coordinate registers. Also, the maximum possible data rate has been doubled and the storage requirements halved (over those of relative vectors). This is done by reducing the coordinate data field by seven-twelfths and packing two values per data word. This performance increase can be exploited where the lower resolution data is adequate and the processing of packed values is not detrimental. The applicability of incremental vectors is enhanced by the scale field (S) which permits the data values to be applied as increments over a coarse or fine grid.

S  INCREMENT SCALE
0  no magnification: add to seven low-order bits
1  magnified: add to seven high-order bits
by specifying magnification, the coordinate increments are added to the high-order bits of the register being updated; otherwise the increment is sign-extended and added to the low-order bits:

The type of display generated by the moving beam is specified by the vector mode (VM) field.

The 1-field of the incremental vector data word controls beam blanking for processing of the entire data word.

The last bit of an incremental vector data-list word is used to flag the end of the data list.
10. Incremental Vectors, 2D Auto-X

010011

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1 S VM 1 0 0 1</td>
</tr>
<tr>
<td>± ΔY I ± ΔY 0</td>
</tr>
<tr>
<td>± ΔY I ± ΔY 1</td>
</tr>
</tbody>
</table>

The 2D auto-X display instruction generates a two-dimensional, relative vector display from packed data increments: but the data words supply only the \( x \) increments. The corresponding \( x \)-increments are taken as the constant held in the auto-increment register (AIR). This further doubles the possible vector rate and halves the memory requirements for the displays such as graphs where one coordinate is steered by a constant.

Each data word supplies two \( x \)-increments and, therefore, is used to generate two vectors.

The \( S, \) \( VM, \) \( I, \) and \( 1 \) fields are coded and used as described for incremental vector 2D, but the \( 1 \)-field applies to both vectors generated from its data word, and both vectors are generated from the final data word.
The 2D auto-Y display instruction generates a two-dimensional, relative vector display from packed data increments; but the data words supply only the $X$ increments. The corresponding $Y$-increments are taken as the constant held in the auto-increment register ($\Delta Y$). This further doubles the possible vector rate and halves the memory requirements for the displays such as graphs where one coordinate is stepped by a constant.

Each data word supplies two $X$-increments and, therefore, is used to generate two vectors.

The S, VM, I, and L fields are coded and used as described for incremental vector 2D, but the L-field applies to both vectors generated from its data word, and both vectors are generated from the final data word.
The 3D incremental vector display instructions generate an XYZ vector display whose coordinates are relative to the initial contents of the coordinate registers. Also, the maximum possible data rate has been increased and the storage requirements reduced (over those of relative vectors). This is done by shortening the coordinate data field width seven-twelfths and packing up to two values per data word. This performance increase can be exploited where the lower resolution data is adequate and the processing of packed values is not detrimental. The applicability of incremental vectors is enhanced by the scale field (S) which permits the data values to be applied as increments over a coarse or fine unit.

One vector is generated for every two data words processed.

The S, VM, I, and T fields are coded and used as
described for incremental vectors, 2D.

13. Character Generation

<table>
<thead>
<tr>
<th>Bit 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 - No slant</td>
</tr>
<tr>
<td>0 - slant</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 0 0 1</td>
</tr>
<tr>
<td>Character</td>
</tr>
<tr>
<td>Character</td>
</tr>
</tbody>
</table>

The character generation display instruction processes its data as a string of extended ASCII character codes packed two per word.

Each successive character displays a symbol or performs a control function until a terminate character, ASCII code DC4, is processed signaling the end of the instruction's data list.

The symbols available include all of the ninety-six ASCII graphics, plus a standard set of ninety-six additional symbols (programming, math, Greek, etc.), and an optional set of 32 user-specified special symbols.

The standard symbols and their codes are given in Appendix A.

The direction field (4) when set causes the characters to be displayed as if on a page which has been rotated.
ninety degrees counter clockwise.

**CHARACTER WRITE-DIRECTION**

- W  Write characters clockwise
- U  Write characters horizontally
- 1  Write characters vertically

The size field (SZ) is used to specify one of the four available string-controlled character sizes. The size-enable bit (E) causes the contents of the SZ field to be instated as the new character size for the subsequent character generation.

<table>
<thead>
<tr>
<th>E</th>
<th>SZ</th>
<th>CHARACTER SIZE CONTROL</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>xx</td>
<td>Use previous character size</td>
</tr>
<tr>
<td>1</td>
<td>00</td>
<td>Set size to one hundred columns by sixty lines</td>
</tr>
<tr>
<td>1</td>
<td>01</td>
<td>Set size to eighty-one columns by forty-one lines</td>
</tr>
<tr>
<td>1</td>
<td>10</td>
<td>Set size to sixty columns by thirty lines</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>Set size to thirty-two columns by sixteen lines</td>
</tr>
</tbody>
</table>

**CONTROL CHARACTERS**

- **DELETE**
  - No display is generated and the tape is not stepped to the next character.
- **NULL**
  - Same as DELETE.
<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BACKSPACE</td>
<td>Causes the positioning to revert to the previous character position.</td>
</tr>
<tr>
<td>LINE FEED</td>
<td>Causes the current line position to be increased by one line.</td>
</tr>
<tr>
<td>FORM FEED</td>
<td>Instates the current character positioning at the first character of line one.</td>
</tr>
<tr>
<td>CARRIAGE</td>
<td>Resets the current column position to position one, the left margin, and increases the current line position by one line.</td>
</tr>
<tr>
<td>RETURN</td>
<td>Reduces the current line position by one line.</td>
</tr>
<tr>
<td>DC1</td>
<td>Reduces the current line position by one line.</td>
</tr>
<tr>
<td>DC2</td>
<td>Decreases the current character size by one size. Permits subscript and superscript sizes to be encoded in text. Size zero is changed to size three.</td>
</tr>
<tr>
<td>DC3</td>
<td>Increases the current character size by one size. Size three is changed to size zero.</td>
</tr>
<tr>
<td>DC4</td>
<td>Then encoded in a display list, this code terminates the data associated with a character generation instruction. If input via the keyboard, it is interpreted as a process termination code. It has the same effect as</td>
</tr>
</tbody>
</table>
the "rubout" key on a DATAKEDIA terminal.

DLE

When encoded in a display list, this code is ignored. When input from the keyboard, it is interpreted as a command to clear all characters from the keyboard character queue.

ESC

When encoded in a display list, this code is ignored. When input from the keyboard, it is interpreted as an escape code for the next character. When ESC is depressed, the next character will be passed as data regardless of the content. This allows the DCL code to be input as data from the keyboard.

HORIZONTAL

Resets the current column position to horizontal center and increases the current line position by one line.

VERTICAL

Instates the current character position to horizontal center of line one.
VI. USER ROUTINES

The description of the user routines are described in the same format as the routine descriptions given in the Unix Reference Manual.
NAME:

vgaddobj - add object

SYNOPSIS:

vgaddobj(num)
int num;

DESCRIPTION:

The object indicated by num is added to the active display list.

Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-1 The object number is negative, zero, or greater than the maximum permitted in the system.
-2 The object doesn't exist
-3 The picture cannot link any more objects
NAME:

vgaddele - add element

SYNOPSIS:

vgaddele(abp,num,size)
int *abp;
int num,size;

DESCRIPTION:

abp is a pointer to a display list with all draw-move commands defined. The byte count of the
display list is the parameter size. This value includes the three words (six bytes) that must be
added after the draw-move commands. The display list is linked to the object, num.

The value returned is the element number of the display list and should be retained for subsequent
element identification.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-1 The object number provided is negative or zero.
-2 The object has not been defined by vamkobj().
-3 The object cannot link additional elements.
-4 The system has generated the maximum number of elements permitted.
-5 The display list is less than six bytes long.
-6 The display list buffer pointer is zero.

SEE ALSO:

vamkobj(), vgdelele()
NAME:

vgblink - set or clear display blink mode

SYNOPSIS:

vgblink(type, num, action)
char type, action;
int num;

DESCRIPTION:

The display blink mode of the entire picture, a single object, or a single element may be set or cleared. The following parameter values apply:

TYPE:

0 - Modify the entire picture.
1 - Modify the object, num.
2 - Modify the element, num.

ACTION:

0 - Clear the display blink mode
1 - Set the display blink mode

Modifying the blink mode of an object affects all elements linked to that object. Likewise, modification of the picture affects all objects comprising the picture.

The normal return value is a zero.

DIAGNOSTICS:

All errors are indicated by negative return values.
The error values and their meanings are as follows:
-1 Object number is negative or zero.
-2 The object or element does not exist.
-4 The element number is negative, zero, or greater than the maximum number of elements permitted in the entire system.
NAME:

vqclock - set refresh rate

SYNOPSIS:

vqclock(rate)
int rate;

DESCRIPTION:

Set the refresh rate of the display. The parameter, rate, is in hertz. The value should be an integer divisor of one hundred twenty. The default value of display initialization is forty hertz.
NAME:

`vqcoord` - modify object X-, Y-, and Z-coordinates

SYNOPSIS:

```c
vqcoord(num,x,y,z)
int num,x,y,z;
```

DESCRIPTION:

The image space X, Y, Z coordinate values of the object, `num`, are set to the values of `x`, `y`, `z` respectively. Each coordinate point of all elements linked to the object is translated by the `x`, `y`, `z` values at display time. The contents of the original display list remain unchanged.

The range of values is `-2047` through `2047`. The system default is zero for the X, Y, Z coordinates.

The normal return value is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follow:
- `-1` The object number is negative or zero.
- `-2` The object is not defined.
NAME:

vqcsr - modify object coordinate scale

SYNOPSIS:

vqcsr(num, val)
    int num;
    double val;

DESCRIPTION:
Change the image space coordinate scale of the object, num, to val. Each coordinate of all elements linked to the object is scaled to val at this time. The contents of the original display that remain unchanged.

The range of val is zero to one. The default default is one, the maximum scale.

The normal return value is zero.

DIAGNOSTICS:
All errors are indicated by negative return values.

The error values and their meanings are in terms of:
-1 The object number is negative or zero.
-2 The object is not defined.

SEE ALSO:

vqscal()

LIMITATIONS:
Negative input values cause the coordinates to be inverted. This gives the appearance of viewing the object from the negative Z axis.
NAME:

`vqdelele - delete element`

SYNOPSIS:

```c
vqdelele(num) 
int num;
```

DESCRIPTION:

Remove the element, `num`, from the display system. The contents of the display list remain unchanged; however, the display cannot be accessed by the display system until again linked to some object.

The normal return value is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-2 The element does not exist.
-4 The element number is negative, zero, or greater than the maximum number of elements permitted in the system.

SEE ALSO:

`vqanielle()`, `vqanooonj()`
NAME:

vadelobj - delete object

SYNOPSIS:

vadelobj(num)
int num;

DESCRIPTION:

Remove the object, num, from the display system. This also deletes all elements linked to the object. The contents of the display lists linked to the object remain unchanged; however, the display cannot access any of the elements until again linked to some object.

The normal return value is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-1 The object number is negative or zero.
-2 The object has not been defined.

SEE ALSO:

vadellete(), voremobj()
NAME:

vgdial - get analog dial values

SYNOPSIS:

vgdial(ahp)

int *ahp;

DESCRIPTION:

Obtain the ten vector general dial values and return them to the caller beginning at the ten vector buffer pointed to by ahp.

DIAGNOSTICS:

The only error code is -6. This means the buffer pointer is zero.
NAME:

vgoetcar - get keyboard character

SYNOPSIS:

vgoetcar()

DESCRIPTION:

An eight bit ASCII character is returned to the caller. If no character has been input a -1 is returned.

A four character queue is maintained by the system. This permits some variation in processing time without character loss. The queue can be cleared of all previous characters by a DEL character from the vector general keyboard.
NAME:

vggetfsw - get function switch values

SYNOPSIS:

vggetfsw(argc)
int *argv;

DESCRIPTION:

Read the thirty-two function switches and return
then to the caller starting at the two word buffer
pointed to by argv. Each bit represents one function
switch. If a bit is set, the function switch is
depressed.

DIAGNOSTICS:

A -10 is returned if the buffer pointer is zero.

SEE ALSO:

vala-ps()
NAME:

vgaeltpn - get light pen interrupt values

SYNOPSIS:

vgaeltpn(ahp)
int *ahp;

DESCRIPTION:

Seven light pen interrupt values are read into a seven word buffer beginning at the buffer pointed to by ahp.

The meaning of each word is:

Word 0 - The element number of the display list being displayed at the time of the interrupt.

Word 1 - Instruction word being executed at the time of the interrupt.

Word 2 - The number of words executed from the beginning of the display.

Word 3,4,5 - The X,Y,Z coordinates respectively at the time of the interrupt.

Word 6 - Pen resolution count.

DIAGNOSTICS:

A return of -6 indicates a buffer counter of zero.

SEE ALSO:

vgltpen()
NAME:

vginit - vector general display initialization

SYNOPSIS:

vginit()

DESCRIPTION:

This routine performs all display initialization and default parameter assignment. The vector general is accessed, cleared, and reset for display operations.

This routine must be called before any display actions.

DIAGNOSTICS:

If the initialization cannot be completed, an error message is printed and the process is terminated. Error messages usually occur because another process is a real-time process or because the vector general has been accessed by another user.
NAME:

vgiofset - modify intensity offset

SYNOPSIS:

vgiofset(num, val)
int num;
double val;

DESCRIPTION:

Set the maximum level of intensity permitted for the object, num. val, between zero and one, is a measure of the maximum intensity of the object. The intensity is maximum at the face of the screen and decreases exponentially toward the back of the image. If val is negative, a screen cut-off is imposed. If the transformed values of the Z coordinate axis are greater than 0.5, they are blanked.

The normal return value is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-1 The object number is negative or zero.
-2 The object does not exist.

SEE ALSO:

voiscal()
NAME:

vqiscal - modify object intensity scale

SYNOPSIS:

vqiscal(num,val)
int num;
double val;

DESCRIPTION:

The intensity range of the object, num, is determined by val. If val is one, the maximum intensity range is achieved. If the value is zero, the intensity is constant and the image has no depth-cueing. Normal return is zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:
-1 The object number is negative or zero.
-2 The object does not exist.

SEE ALSO:

vqisfset()
NAME:

vqlamps - light function switch lamps

SYNOPSIS:

vqlamps(abo)
int *abo;

DESCRIPTION:

Light the function switch lamps of those function switches with a bit set in the two word buffer pointed to by abo. Each bit position corresponds to one of the thirty-two function switch lamps.

DIAGNOSTICS:

If the buffer pointer is zero, a -1 is returned.

SEE ALSO:

vucetfsa()
NAME:

volpen - set or clear light pen hookability

SYNOPSIS:

volpen(type,num,action)
char type;
int num, action;

DESCRIPTION:

The light pen hookability of the entire picture, a single object, or a single element may be set or cleared. The following parameter values apply:

TYPE

0      "Modify the entire picture."
1      "Modify the object, num."
2      "Modify the element, num."

ACTION

0      "Clear the light pen hookability."
1      "Set the light pen hookability."

The normal return value is a zero.

DIAGNOSTICS:

All errors are indicated by negative return values. The error values and their meanings are as follows:

-1 The object number is negative or zero.
-2 The object or element does not exist.
-4 The element number is negative, zero, or greater than the maximum number of elements permitted in the entire system.
NAME:

vgmkobj - make object

SYNOPSIS:

vgmkobj()

DESCRIPTION:

Create an empty object structure and return the object number to the caller.

DIAGNOSTICS:

A -3 is returned if all available objects have been previously assigned.
NAME:

vgpicture - display picture

SYNOPSIS:

vgpicture()  

DESCRIPTION:
Send the vector general display list and start the display.
NAME:

**vqpost** - modify the picture coordinate axis

SYNOPSIS:

```c
vqpost(px,py)
int px,py;
```

DESCRIPTION:

Modify the transformed X,Y coordinate points of each image in the picture by ox and oy respectively. The original display lists remain unchanged. The range of values are -2047 through +2047.

SEE ALSO:

vacoord()
NAME:

vqpscal - modify picture scale

SYNOPSIS:

vqpscal(val)

double val;

DESCRIPTION:

Scale each coordinate of all elements by val. The original display list remains unchanged. A value of one gives the largest picture. A value of zero gives the smallest.
NAME:

`vgrave;rotate - rotate object`

SYNOPSIS:

```c
vgrave;rotate(num,x,y,z)
int num;
double x,y,z;
```

DESCRIPTION:

Rotate the object, `num`, about the x, y, and z axis. The input parameters are radian measure rotation about the axis indicated. All elements linked to the object are affected. The original display lists remain unaffected.
NAME:

vgterm - terminate vector general display

SYNOPSIS:

vgterm()

DESCRIPTION:

Terminate access to the vector general and release all system resources. This should be called at the conclusion of all display operations.
Table A-1 lists the ASCII codes used by the display system for the various general and special characters. The codes are given in octal notation. The octal codes are given as though there were eighteen bits in the data word instead of sixteen bits. Since two characters can be given in each data word, the octal codes are given for the right half-word and the left half-word. The left half-word code is given as though there were no character in the right half-word. To obtain the complete code for the two characters in a word, the user must add the two codes together. For example, if the character C is to be in the left half-word and the character A is to be in the right half-word, the code would be:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>041400</td>
</tr>
<tr>
<td>A</td>
<td>0101</td>
</tr>
<tr>
<td>CA</td>
<td>041501</td>
</tr>
<tr>
<td>OCTAL</td>
<td>LEFT</td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
</tr>
<tr>
<td>000000</td>
<td>000</td>
</tr>
<tr>
<td>000400</td>
<td>001</td>
</tr>
<tr>
<td>001000</td>
<td>002</td>
</tr>
<tr>
<td>001400</td>
<td>003</td>
</tr>
<tr>
<td>002000</td>
<td>004</td>
</tr>
<tr>
<td>002400</td>
<td>005</td>
</tr>
<tr>
<td>003000</td>
<td>006</td>
</tr>
<tr>
<td>003400</td>
<td>007</td>
</tr>
<tr>
<td>004000</td>
<td>010</td>
</tr>
<tr>
<td>004400</td>
<td>011</td>
</tr>
<tr>
<td>005000</td>
<td>012</td>
</tr>
<tr>
<td>005400</td>
<td>013</td>
</tr>
<tr>
<td>006000</td>
<td>014</td>
</tr>
<tr>
<td>006400</td>
<td>015</td>
</tr>
<tr>
<td>007000</td>
<td>016</td>
</tr>
<tr>
<td>007400</td>
<td>017</td>
</tr>
<tr>
<td>010000</td>
<td>020</td>
</tr>
<tr>
<td>010400</td>
<td>021</td>
</tr>
<tr>
<td>011000</td>
<td>022</td>
</tr>
<tr>
<td>011400</td>
<td>023</td>
</tr>
<tr>
<td>012000</td>
<td>024</td>
</tr>
<tr>
<td>012400</td>
<td>025</td>
</tr>
<tr>
<td>013000</td>
<td>026</td>
</tr>
<tr>
<td>013400</td>
<td>027</td>
</tr>
<tr>
<td>014000</td>
<td>030</td>
</tr>
<tr>
<td>014400</td>
<td>031</td>
</tr>
<tr>
<td>015000</td>
<td>032</td>
</tr>
<tr>
<td>015400</td>
<td>033</td>
</tr>
<tr>
<td>016000</td>
<td>034</td>
</tr>
<tr>
<td>016400</td>
<td>035</td>
</tr>
<tr>
<td>001000</td>
<td>036</td>
</tr>
<tr>
<td>017400</td>
<td>037</td>
</tr>
<tr>
<td>020000</td>
<td>040</td>
</tr>
<tr>
<td>020400</td>
<td>041</td>
</tr>
<tr>
<td>021000</td>
<td>042</td>
</tr>
<tr>
<td>021400</td>
<td>043</td>
</tr>
<tr>
<td>022000</td>
<td>044</td>
</tr>
<tr>
<td>022400</td>
<td>045</td>
</tr>
<tr>
<td>023000</td>
<td>046</td>
</tr>
<tr>
<td>023400</td>
<td>047</td>
</tr>
<tr>
<td>023800</td>
<td>048</td>
</tr>
<tr>
<td>024000</td>
<td>050</td>
</tr>
<tr>
<td>024400</td>
<td>051</td>
</tr>
<tr>
<td>025000</td>
<td>052</td>
</tr>
<tr>
<td>025400</td>
<td>053</td>
</tr>
<tr>
<td>026000</td>
<td>054</td>
</tr>
<tr>
<td>026400</td>
<td>055</td>
</tr>
<tr>
<td>027000</td>
<td>056</td>
</tr>
</tbody>
</table>

75
<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td>027400</td>
<td>057</td>
<td>/</td>
</tr>
<tr>
<td>030000</td>
<td>060</td>
<td>0</td>
</tr>
<tr>
<td>030400</td>
<td>061</td>
<td>1</td>
</tr>
<tr>
<td>031000</td>
<td>062</td>
<td>2</td>
</tr>
<tr>
<td>031400</td>
<td>063</td>
<td>3</td>
</tr>
<tr>
<td>032000</td>
<td>064</td>
<td>4</td>
</tr>
<tr>
<td>032400</td>
<td>065</td>
<td>5</td>
</tr>
<tr>
<td>033000</td>
<td>066</td>
<td>6</td>
</tr>
<tr>
<td>033400</td>
<td>067</td>
<td>7</td>
</tr>
<tr>
<td>034000</td>
<td>070</td>
<td>8</td>
</tr>
<tr>
<td>034400</td>
<td>071</td>
<td>9</td>
</tr>
<tr>
<td>035000</td>
<td>072</td>
<td>:</td>
</tr>
<tr>
<td>035400</td>
<td>073</td>
<td>;</td>
</tr>
<tr>
<td>036000</td>
<td>074</td>
<td>&lt;</td>
</tr>
<tr>
<td>036400</td>
<td>075</td>
<td>=</td>
</tr>
<tr>
<td>037000</td>
<td>076</td>
<td>&gt;</td>
</tr>
<tr>
<td>037400</td>
<td>077</td>
<td>?</td>
</tr>
<tr>
<td>040000</td>
<td>100</td>
<td>A</td>
</tr>
<tr>
<td>040400</td>
<td>101</td>
<td>B</td>
</tr>
<tr>
<td>041000</td>
<td>102</td>
<td>C</td>
</tr>
<tr>
<td>041400</td>
<td>103</td>
<td>D</td>
</tr>
<tr>
<td>042000</td>
<td>104</td>
<td>E</td>
</tr>
<tr>
<td>042400</td>
<td>105</td>
<td>F</td>
</tr>
<tr>
<td>043000</td>
<td>106</td>
<td>G</td>
</tr>
<tr>
<td>043400</td>
<td>107</td>
<td>H</td>
</tr>
<tr>
<td>044000</td>
<td>108</td>
<td>I</td>
</tr>
<tr>
<td>044400</td>
<td>109</td>
<td>J</td>
</tr>
<tr>
<td>045000</td>
<td>110</td>
<td>K</td>
</tr>
<tr>
<td>045400</td>
<td>111</td>
<td>L</td>
</tr>
<tr>
<td>046000</td>
<td>112</td>
<td>M</td>
</tr>
<tr>
<td>046400</td>
<td>113</td>
<td>N</td>
</tr>
<tr>
<td>047000</td>
<td>114</td>
<td>O</td>
</tr>
<tr>
<td>047400</td>
<td>115</td>
<td>P</td>
</tr>
<tr>
<td>050000</td>
<td>116</td>
<td>Q</td>
</tr>
<tr>
<td>050400</td>
<td>117</td>
<td>R</td>
</tr>
<tr>
<td>051000</td>
<td>118</td>
<td>S</td>
</tr>
<tr>
<td>051400</td>
<td>119</td>
<td>T</td>
</tr>
<tr>
<td>052000</td>
<td>120</td>
<td>U</td>
</tr>
<tr>
<td>052400</td>
<td>121</td>
<td>V</td>
</tr>
<tr>
<td>053000</td>
<td>122</td>
<td>W</td>
</tr>
<tr>
<td>053400</td>
<td>123</td>
<td>X</td>
</tr>
<tr>
<td>054000</td>
<td>124</td>
<td>Y</td>
</tr>
<tr>
<td>054400</td>
<td>125</td>
<td>Z</td>
</tr>
<tr>
<td>055000</td>
<td>126</td>
<td>I</td>
</tr>
<tr>
<td>055400</td>
<td>127</td>
<td>\</td>
</tr>
<tr>
<td>056000</td>
<td>128</td>
<td>J</td>
</tr>
<tr>
<td>056400</td>
<td>129</td>
<td>^</td>
</tr>
<tr>
<td>057000</td>
<td>130</td>
<td>(sub-, superscript)</td>
</tr>
<tr>
<td>057400</td>
<td>131</td>
<td></td>
</tr>
</tbody>
</table>

\* shift :
| 060400 | 141 | a             | A             |
| 061000 | 142 | b             | B             |
| 061400 | 143 | c             | C             |
| 062000 | 144 | d             | D             |
| 062400 | 145 | e             | E             |
| 063000 | 146 | f             | F             |
| 063400 | 147 | g             | G             |
| 064000 | 150 | h             | H             |
| 064400 | 151 | i             | I             |
| 065000 | 152 | j             | J             |
| 065400 | 153 | k             | K             |
| 066000 | 154 | l             | L             |
| 066400 | 155 | m             | M             |
| 067000 | 156 | n             | N             |
| 067400 | 157 | o             | O             |
| 070000 | 160 | p             | P             |
| 070400 | 161 | q             | Q             |
| 071000 | 162 | r             | R             |
| 071400 | 163 | s             | S             |
| 072000 | 164 | t             | T             |
| 072400 | 165 | u             | U             |
| 075000 | 172 | r             | R             |
| 075400 | 173 | {             | L shift       |
| 076000 | 174 | :             | \ shift       |
| 076400 | 175 | }             | \ shift       |
| 077000 | 176 | ~             | ^ shift       |
| 077400 | 177 | del           | DEL           |

<p>| 100000- | 200- | space         |
| 117400  | 237  | \ shift spec  |
| 120000  | 240  | 2 shift spec  |
| 120400  | 241  | 3 shift spec  |
| 121000  | 242  | 4 shift spec  |
| 121400  | 243  | 5 shift spec  |
| 122000  | 244  | 6 shift spec  |
| 122400  | 245  | 7 shift spec  |
| 123000  | 246  | 8 shift spec  |
| 123400  | 247  | 9 shift spec  |
| 124000  | 250  | 0 shift spec  |
| 124400  | 251  | , shift spec  |
| 125000  | 252  | ; shift spec  |
| 125400  | 253  | . shift spec  |
| 126000  | 254  | , spec        |
| 126400  | 255  | - spec        |
| 127000  | 256  | , spec        |
| 127400  | 257  | / spec        |
| 130000  | 260  | 0 spec        |</p>
<table>
<thead>
<tr>
<th>TablE A-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>171400</td>
</tr>
<tr>
<td>172000</td>
</tr>
<tr>
<td>172400</td>
</tr>
<tr>
<td>173000</td>
</tr>
<tr>
<td>173400</td>
</tr>
<tr>
<td>174000</td>
</tr>
<tr>
<td>174000</td>
</tr>
<tr>
<td>174400</td>
</tr>
<tr>
<td>175000</td>
</tr>
<tr>
<td>175400</td>
</tr>
<tr>
<td>176000</td>
</tr>
<tr>
<td>176400</td>
</tr>
<tr>
<td>177000</td>
</tr>
</tbody>
</table>

* Superscript
APPENDIX B
SAMPLE PROGRAM

The sample program described here is a box containing a zigzag line and the word box. This program is not intended to illustrate all of the features of the vector general interface but to illustrate one way of developing a display picture.

```c
#define _OBJ 0    // make object
#define _ELE 1    // and element
#define _OBJ1 2   // and object

// Display list describing a box
int box [10]
{
  0 100004,    // vector absolute inst
  0100001,    // load x coord, and move
  0100012,    // load y coord, and move
  077705,    // load x coord, and draw
  077700,    // load y coord, and draw
  0100005,    // load x coord, and draw
  0100016,    // load y coord, draw & term.
  0000000,    // term.
  0000000,    // term.
  0000000,    // term.
  0000000,    // term.
};

// Character display list spelling the word box
int word [7]
{
  010157,    // char generation inst
  010410,    // neg. line feed & space
  041157,    // ASCII bytes, - and u
  074024,    // ASCII bytes, a & term.
  000000,    // term.
  000000,    // term.
  000000,    // term.
};
```
// Zioza line display list
int ziq[12]
{
  010011, // 2D vector incremental,
  077176, // X auto-increment
  077602, // move Y
  077602, // increment X, draw Y
  077602, // increment X, draw Y
  077602, // increment X, draw Y
  077602, // increment X, draw Y
  077602, // increment X, draw Y
  000000, // Noop
  000000, // Noop
  000000, // Noop
};

main()
{
  int ele1, ele2, ele3;
  int obj1, obj2, obj3;

  vinit(); // initiate vector general

  // Make three objects
  if ((obj1=vqmnobjO) < 0) error(EMOBJ);
  if ((obj2=vqmnobj(word, objj, 14)) < 0)
    error(ADELE);
  if ((obj3=vqmnobj(zio, orj3, 2)) < 0)
    error(ADELE);

  // Add the display lists to the objects
  if ((ele1=vqaddele(box, obj1, 20)) < 0)
    error(ADELF);
  if ((ele2=vqaddele(word, objj, 14)) < 0)
    error(ADELE);
  if ((ele3=vqaddele(zio, orj3, 2)) < 0)
    error(ADELE);

  // Add the objects to the picture
  if (vqaddobj(obj1) < 0) error(ADOBJ);
  if (vqaddobj(objj) < 0) error(ADOBJ);
  if (vqaddobj(obj3) < 0) error(ADOBJ);

  vopicture(); // start display of picture
  sleep(60); // wait 60 seconds
  vqterm(); // terminate display operations
}
error(msq)
int msg;
{
    switch(msq)
    {
    case MKOBJ:
        {
            printf("make object error");
            break;
        }
    case ADELE:
        {
            printf("add element error");
            break;
        }
    case ADOBJ:
        {
            printf("add object error");
            break;
        }
    }
    vterm();
    exit();
}


INITIAL DISTRIBUTION LIST

Dean of Research
Code 023
Naval Postgraduate School
Monterey, California 93940

Defense Documentation Center
Cameron Station
Alexandria, Virginia 22314

Library (Code 0212)
Naval Postgraduate School
Monterey, California 93940

. K. Church Computer Center
Naval Postgraduate School
Monterey, California 93940

Naval Electronics Systems Command (LEZ C 527)
Attn: Cdr. A. Miller
Department of the Navy
Washington, D.C. 20360

Naval Electronics Laboratory
Center Library
271 Catalina Boulevard
San Diego, California 92152

Ltjg. Gary V. Paetz, USN (Code 720)
Naval Postgraduate School
Monterey, California 93940

Lt. Lloyd A. Horns, USN
415 24th Street West
Billings, Montana 59102

Copies
1
2
2
1
1
10
1