TEKTRONIX®

excellence in information display

PLOT-10

TERMINAL CONTROL SYSTEM – 4010
USER'S MANUAL

INFORMATION DISPLAY PRODUCTS

Tektronix, Inc. • P.O. Box 500 • Beaverton, Oregon 97005 • Phone 644-0161

Document No. 062-1474-00

CUSTOMER AGREEMENT

REFERENCE MATERIAL

Additional copies of this reference material may be ordered using the Document number 062-1474-00.

This program is available in the following machine-entry forms:

User's Manual and Paper Source Tape:	062-1474-01
User's Manual and Source Card Deck:	062-1474-02

Please place all orders through your Tektronix Application Engineer.

Date: 4/15/72

This manual refers to Release 2 of the Terminal Control System manual.

TERMINAL CONTROL SYSTEM USER'S MANUAL

CONTENTS

TERMINAL CONTROL SYSTEM

USER'S MANUAL

1.0 INTRODUCTION

1.1 Software, Standards, and the User

One of the major difficulties in the development of Computer Graphics has been the lack of standard basic graphic software. As a result there has been a tendency to re-develop the basic software for each installation and, in some cases, for each application. In the past this software has often been oriented towards one system, applicable to only one type of terminal, and frequently had peculiar features facilitating a particular application and precluding others.

The software thus developed was often too complex for the occasional user to use conveniently and frequently too inflexible for the needs of the sophisticated programmer. As a result, graphic application software using such a base tended to have limited use and life.

To meet the need of the different users and the multiplicity of systems, Tektronix has developed the Terminal Control System. The Terminal Control System is a comprehensive set of functionally modular subroutines which allows essentially terminal-independent programming. The user needs only to select the proper modules at load time. The design is basically system and computer independent and allows the experienced programmer to work at the basic terminal level and also provides the facilities for the occasional user to operate easily at the conceptual level. The PLOT-1Ø version of the Terminal Control System consists of those modules which support the Tektronix 4Ø1Ø Computer Display Terminal. Properly written programs using the PLOT-1Ø/Terminal Control System should function with little or no modification on another model terminal (e.g. the Tektronix 4002A Graphic Computer Terminal) when loaded with the modules supporting that device.

The Terminal Control System will be used as a base for the future development of Tektronix Application Software and it is hoped that it will serve as an industry-wide standard for basic software for interactive graphic terminals.

1.2 The Tektronix 4Ø1Ø Terminal

The Tektronix 4Ø1Ø Computer Display Terminal is capable of displaying both alphanumeric characters and vectors. The display remains visible once written and until erased. It is not necessary to continually regenerate the output data or refresh the screen.

The 4Ø1Ø Terminal has a display area of 7.5 inches by 5.6 inches (19.0 centimeters by 14.3 centimeters) and contains 1Ø24 by 1Ø24 addressable points, of which 1Ø24 by 781 are in the viewable area* of the screen. In the alphanumeric mode, the 4Ø1Ø can display 35 lines of 72 characters each. There are 63 printable characters plus the space character. Graphic data may be displayed using the vector mode. Positional data may be input with a thumbwheel-controlled graphic cursor.

The Terminal Control System expands upon these basic hardware functions to provide more sophisticated means of handling the interactive capabilities of the terminal and frees the user from a number of basic "housekeeping" chores.

1.3 Terminal Control System Overview

The ideal that the Terminal Control System strives for is to make the terminal as easy to use as a pencil and a piece of paper. The detailed programming and general I/O handling are contained within the Terminal Control System. The basic terminal capabilities are expanded upon and made available to the user in a natural and practical manner.

The Terminal Control System modules communicate with each other primarily through the Terminal Status Area, a set of common variables which continuously represent the current state of the terminal and maintain the data and flags necessary to generate output according to the user's current level of usage. Terminal status will be lost whenever output to the terminal is generated other than through the appropriate Terminal Control System routine, or whenever the user changes status locally (e.g. uses the Page or Reset key). Terminal status should be saved before allowing these events to occur and should be restored afterwards (see Section 8.0).

*Vectors just above 78Ø on the Y-axis may be visible but marginal in quality. For the purposes of this manual such vectors are considered part of the unviewable area.

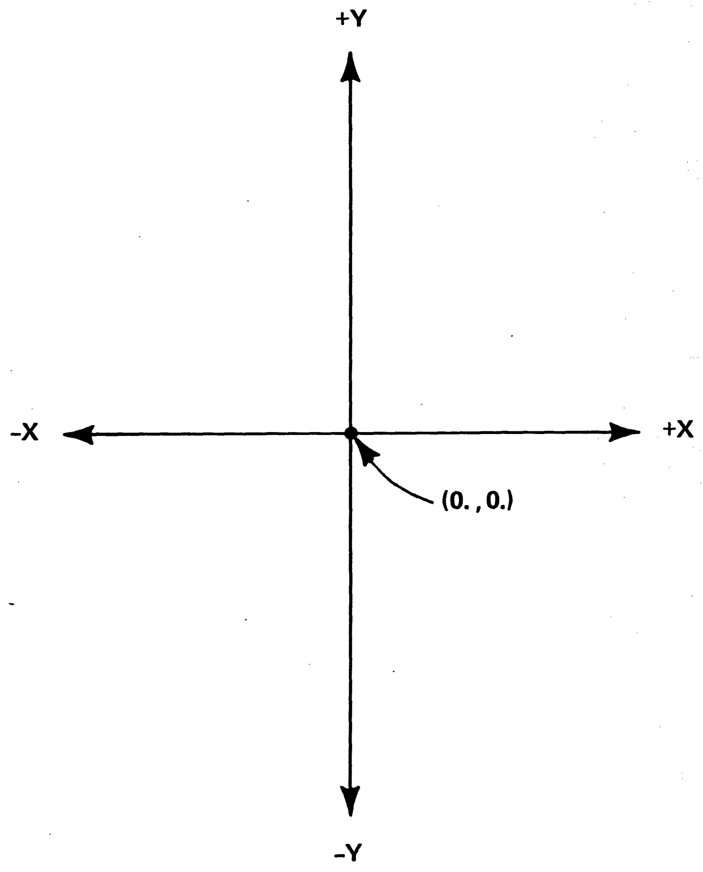
1.3 Terminal Control System Overview (continued)

As most users conceive of their graphic data as existing on a sheet of paper of arbitrary size, the Terminal Control System allows them to maintain this concept within their program through the use of a Virtual Display. The Virtual Display is a two-dimensional surface of indeterminable size, limited only by the numeric processing capability of the computer. All or a portion of the Virtual Display may be viewed at any time. The user is only responsible for defining what portion he wishes to have displayed. This is done by establishing a window which specifies the portion of the Virtual Display to be viewed and where on the terminal screen it is to be placed. The Terminal Control System will handle the conversions and details.

The Virtual Display is in contrast to the terminal screen (Figure 1.0). The user may address the terminal screen directly in Screen Coordinates or he may use the inch and centimeter conversion functions. By referring directly to the screen, the user can easily and naturally control the layout of his display.

The operations which are applied to the screen are called Direct Graphics while the operations which are applied to the Virtual Display are known as Virtual Graphics. The user is able to switch freely between Virtual Graphics and Direct Graphics according to his requirements.

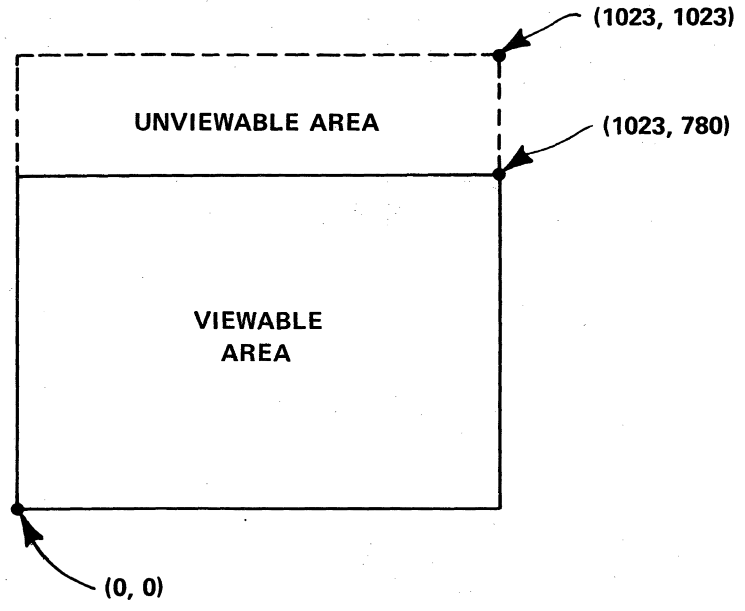
Along with graphical data handling, the Terminal Control System also aids in the output of alphanumeric data. The user is able to set and reset both horizontal and vertical tabs and may dynamically define left and right margins. The Terminal Control System automatically monitors alphanumeric output and the alphanumeric control commands.



VIRTUAL DISPLAY

Bounded only by the single precision floating point range.

(a)



TERMINAL SCREEN

Bounded by 0 and 1023 on the X-Axis and by 0 and 1023 on the Y-Axis, but only 0 thru 780 on the Y-Axis is in the viewable area.

(b)

Fig. 1.0.

2.0 FUNCTION CONTROL ROUTINES

2.1 Initialization

Initialization of the terminal and the Terminal Status Area must be accomplished as the initial step in using the Terminal Control System. This may be done quickly and easily by calling the initialization routine, INITT. When INITT is called, the following events occur:

a. The screen is erased and the cursor moves to the HOME position (upper left hand corner).

b. Alphanumeric mode is entered.

c. The margin variables are set to the left and right screen extremes.

d. The window is defined so that the portion of the Virtual Display which is equivalent in coordinates with the screen will be displayed [i.e. (275.,763.) in Virtual Coordinates is equivalent to (275,763) in screen coordinates].

e. The relative vector scaling is set to unity and the relative vector rotation to zero.

INITT requires the rate of character transmission from the computer to the terminal as an input argument in order that appropriate delays may be produced during screen erasure and hard copy generation. This will prevent loss of data on remotely connected terminals during these periods.

CALLING SEQUENCE:

```
                    CALL INITT (ICPS)
```

where: ICPS - the transmission rate in characters per seconds. ICPS = Ø implies the terminal is directly connected and no delays are required.

NOTE: For certain systems, additional in-line initialization may be required. Please check the Terminal Control System Implementation Notes for your computer and system.

2.2 Termination

When terminating a program which uses the Terminal Control System, it is necessary that the terminal be returned to alphanumeric mode and that the beam position be moved to a point that will not interfere with any previous output.

2.2 Termination (continued)

The Terminal Control System provides the routine FINITT, which will automatically perform these functions and then terminate the program. FINITT should be always used in place of the HALT command or the FORTRAN STOP statement. (NOTE: If FINITT places the beam above the Screen Y-Coordinate of the Home position (KHOMEY), the terminal will automatically lower the cursor to that Y-position on entry to alphanumeric mode.)

CALLING SEQUENCE:

```
CALL FINITT (IX, IY)
```

where: IX - The Screen X-Coordinate of the position to which the beam is moved before program termination.

IY - The Screen Y-Coordinate of the beam termination position.

2.3 Erasing the Screen

The terminal screen may be erased without changing the mode or beam position. The Terminal Control System will prevent generation of additional output until the erase is completed.

CALLING SEQUENCE:

```
CALL ERASE
```

2.4 Bell or Audible Output

An audible tone may be output at any time to call the user's attention to a particular event. Often a continued audible output, which may be generated by a series of calls to the bell routine, is used for an alarm. The "bell" may be sounded while in any mode and has no affect on terminal status.

CALLING SEQUENCE:

```
CALL BELL
```

2.5 Hard Copy Generation

A permanent copy of the current display may be obtained any time the optional hard copy unit is attached by having the computer initiate hard copy generation. This may be done

2.5 Hard Copy Generation (continued)

while in any mode and does not affect the Terminal Control System status. The Terminal Control System will prevent generation of additional output until the hard copy is completed.

CALLING SEQUENCE:

```
                CALL HDCOPY
```

```
C*** INITIALIZE TERMINAL CONTROL SYSTEM ***
C*** BAUD RATE IS 3Ø CHARS/SEC(3ØØ BAUD) ***
      CALL INITT(3Ø)
           .
           .
           .
C*** ERASE SCREEN ***
      CALL ERASE
           .
           .
           .
C*** SOUND BELL ***
      CALL BELL
           .
           .
           .
C*** GENERATE A HARD COPY FASCIMILE OF CURRENT DISPLAY ***
      CALL HDCOPY
           .
           .
           .
C*** MOVE BEAM TO CENTER SCREEN AND ***
C*** TERMINATE PROGRAM ***
      CALL FINITT(511,125)
      END
```

Using the Basic Routines

Example 2.0

3.0 VIRTUAL GRAPHICS

3.1 The Virtual Display

The Virtual Display is an imaginary two-dimensional surface with a range in both the X and Y directions equal to the range of a single precision floating point number. Using the Virtual Display the user may construct drawings, pictures, and graphs of extreme complexity and detail.

Since the unit of measurement of the Virtual Display is arbitrary, it may be assumed to be representative of any measurement unit from microns to light-years, with all measurements translated to the assumed unit for the given drawing. For example, the user decides that the basic unit of the Virtual Display will represent inches. Then the Virtual Coordinate (2., Ø.5) represents a point two inches to the right of the origin on the X-axis and one half inch up on the Y-axis. To indicate the point one mile (63,360 inches) to the left of the origin along the X-axis, the Virtual Coordinate (-6336Ø.Ø, Ø.Ø) would be used.

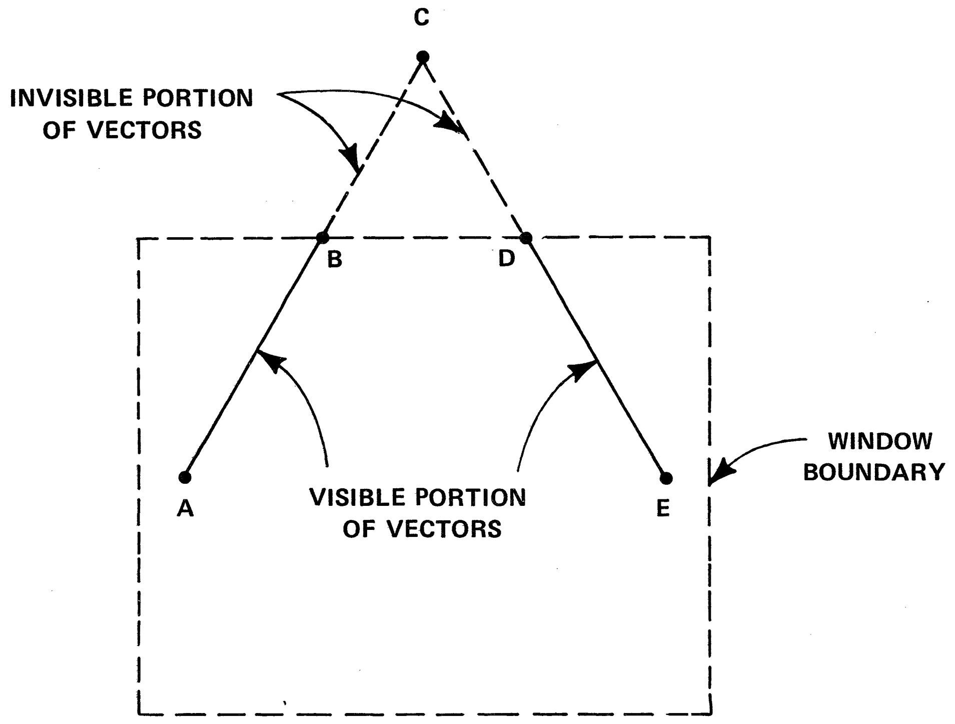
The Virtual Display is similar to normal displays and plotting devices in that there is a movable point which may be thought of as the writing cursor on the Virtual Display. This point is called the Imaginary Beam, and its position is the Virtual Coordinate which represents the location of the writing cursor as if the Virtual Display were an actual device.

Since only the portions of vectors and the points which lie within the current window are displayed, the Imaginary Beam position does not always represent the actual storage beam position. The actual beam is represented on the Virtual Display by the Real Beam, which is updated to reflect the actual output to the terminal. Figure 3.1 illustrates the differences between the Imaginary Beam and the Real Beam. When entering Virtual Graphics or whenever the window is redefined, both the Imaginary Beam and the Real Beam are set at the Virtual Coordinate representation (according to the latest window definition) of the actual beam position.

3.2 Windowing

All or any portion of the Virtual Display may be viewed at any time through the technique of windowing. The portion of the Virtual Display to be shown is defined by rectangular boundaries. This rectangle is called the Virtual Window, and only those vectors which pass through the Virtual Window will be displayed.

It is not necessary to use all of the screen for display of the Virtual Window. The user may define a rectangular section of



ACTION	IMAGINARY BEAM	REAL BEAM
1) Vector drawn from A to C.	Moved from vector start point, A, to vector end point, C.	Moved from vector start point, A, to vector intercept with window boundary, B.
2) Vector drawn from C to E.	Moved from vector start point, C, to vector end point, E.	Moved from B to vector intercept, D, then move with drawing of vector to end point, E.

Fig. 3.1. Imaginary and Real Beams.

any size and location on the screen as the area in which the window will appear. This rectangle is called the Screen Window and together with the Virtual Window defines the transformation between the Virtual Display and the screen (Figure 3.2).

Elimination of vectors and portions of vectors which lie outside of the window will be done automatically by the Virtual Graphic routines as well as the scaling and conversion of these vectors that are contained in or pass through the window.

It should be noted here that the scaling is not related to the size of the Virtual Display or the screen, but is determined solely by the window definition. Also, since the X and Y extents of the window may be separately defined (see 3.2.1, 3.2.2 below), the X and Y scaling are independent. This allows for the emphasis of either X or Y data values, (Figure 3.3). Care must be taken that unwanted distortion is not introduced by erroneous window definitions. The initial window definition is set so that the portion of the Virtual Display with coordinates equivalent to the screen will be displayed:

Virtual Window Initial Values:

X minimum - Ø., X extent - 1Ø23.
Y minimum - Ø., Y extent - 78Ø.

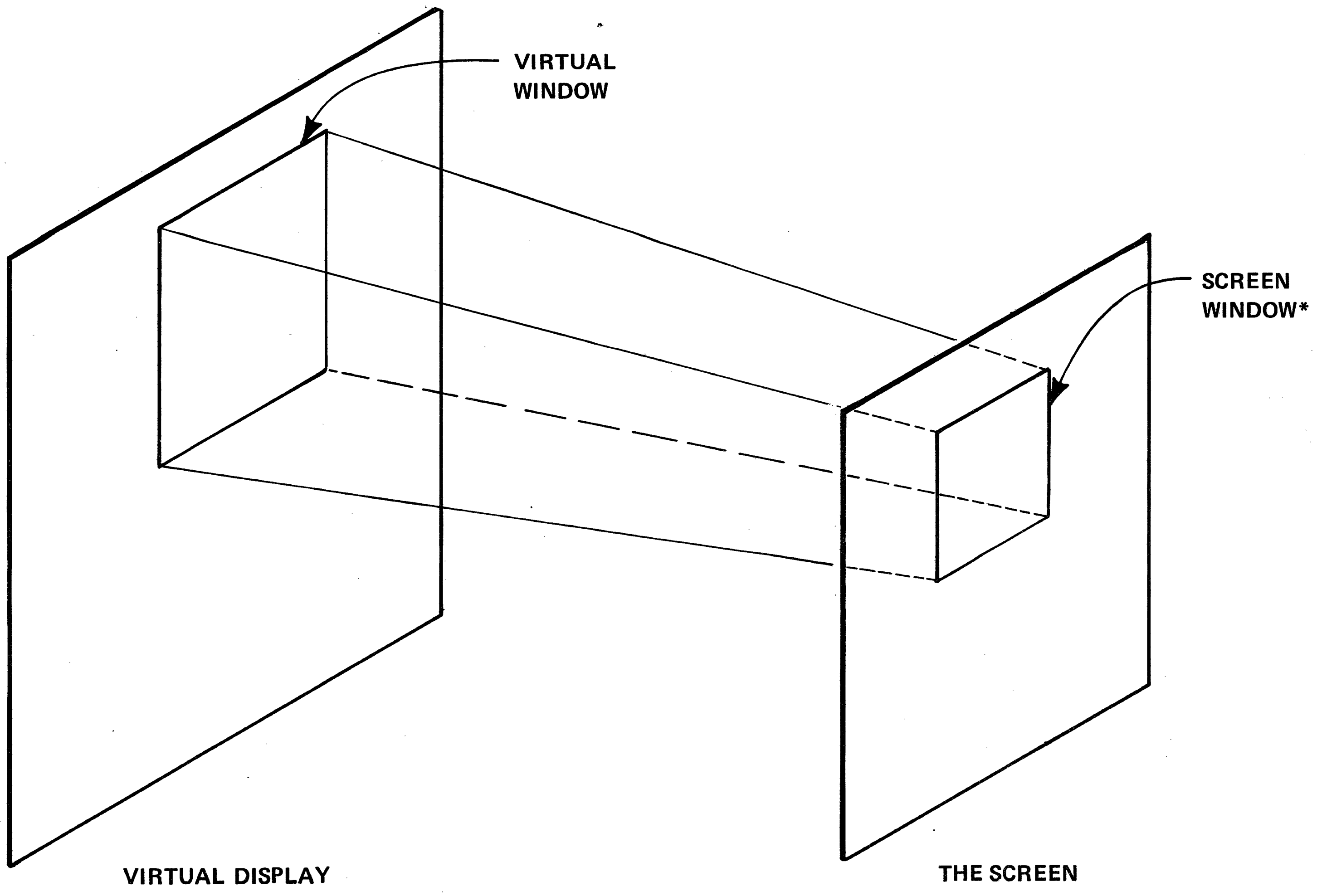
Screen Window Initial Values:

X minimum - Ø, X extent - 1Ø23
Y minimum - Ø, Y extent - 78Ø

The user utilizes the Virtual Display by first defining his window and then constructing his drawing, picture, or graph with the use of the Virtual Graphic routines. The user may display several portions of the Virtual Display at one time by redefining the window and reprocessing the Virtual Display for each (Figure 3.4) or may superimpose data from "several" Virtual Displays by using a common Screen Window (Figure 3.5). All transformations between the Virtual Display and the Screen will be based upon the latest window definitions.

3.2.1 Setting the Virtual Window

The portion of the Virtual Display to be viewed is determined by the Virtual Window. The Virtual Window is defined by a point which represents its lower left corner and the extent of the window in the X and Y directions.



*Lines outlining the Screen Window position are not automatically drawn. They are used here for illustrative purposes only.

Fig. 3.2. Windowing.

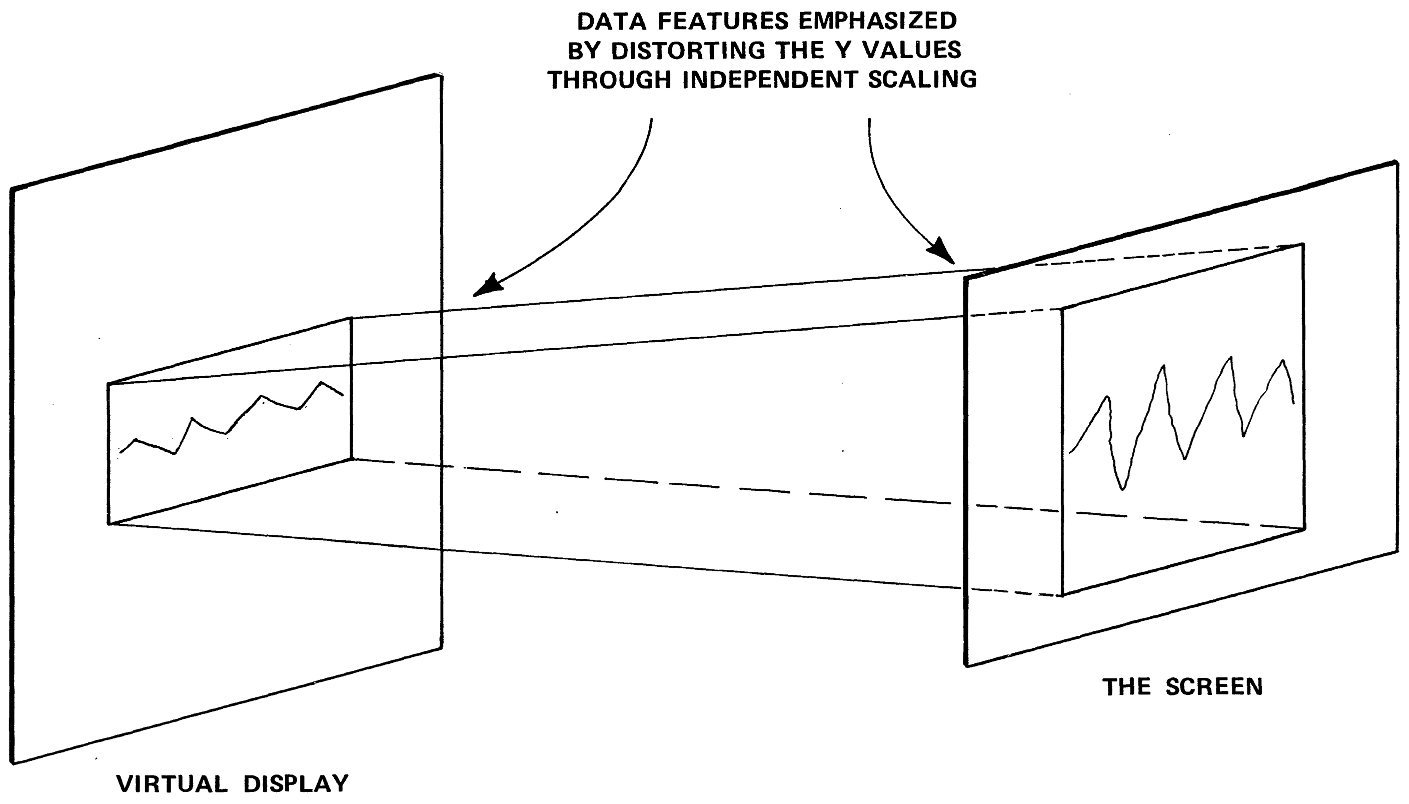

DATA FEATURES EMPHASIZED BY DISTORTING THE Y VALUES THROUGH INDEPENDENT SCALING
THE SCREEN
VIRTUAL DISPLAY

Fig. 3.3. Independent X, Y Window Scaling.

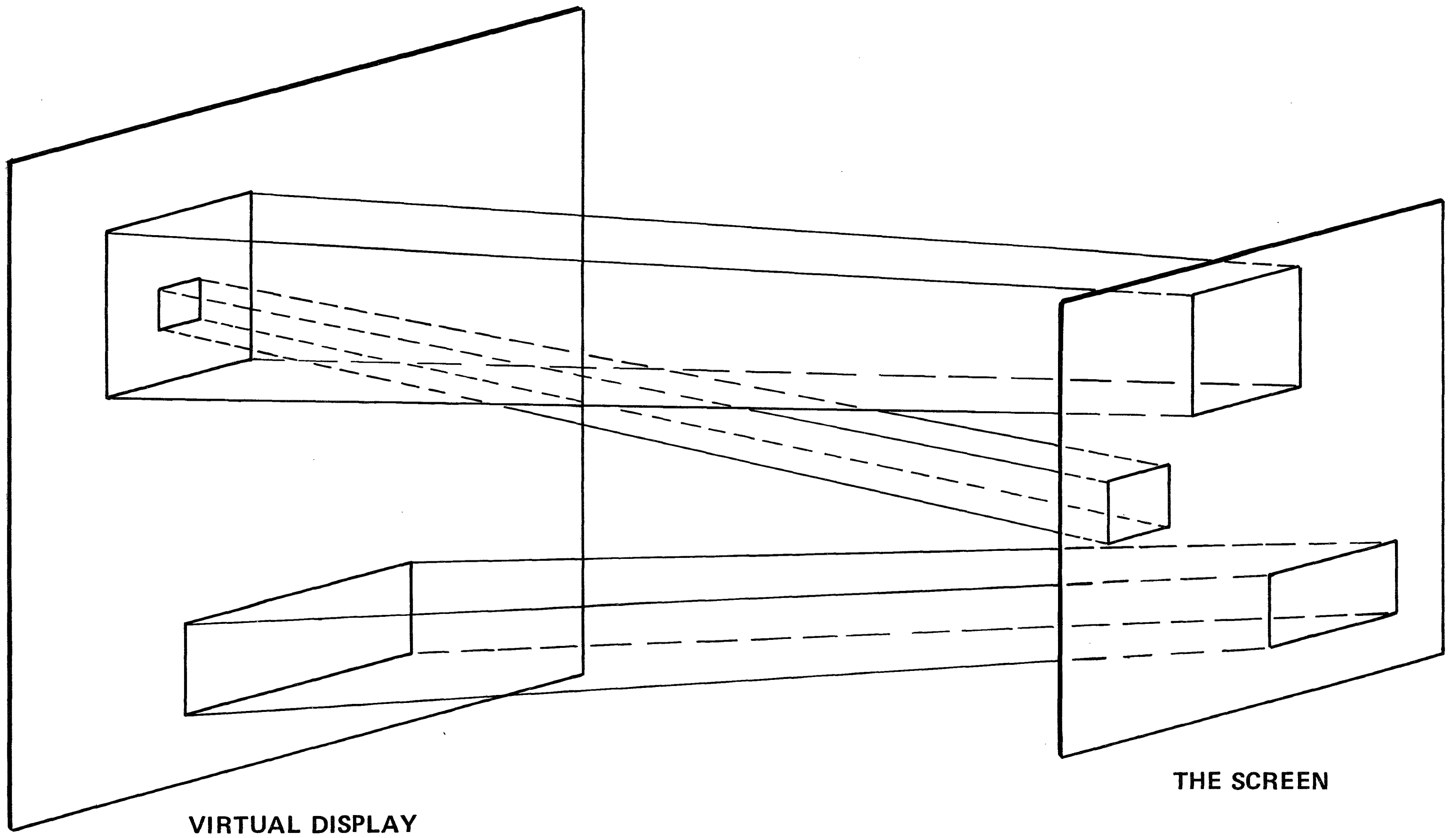


Fig. 3.4. Use of Several Windows.

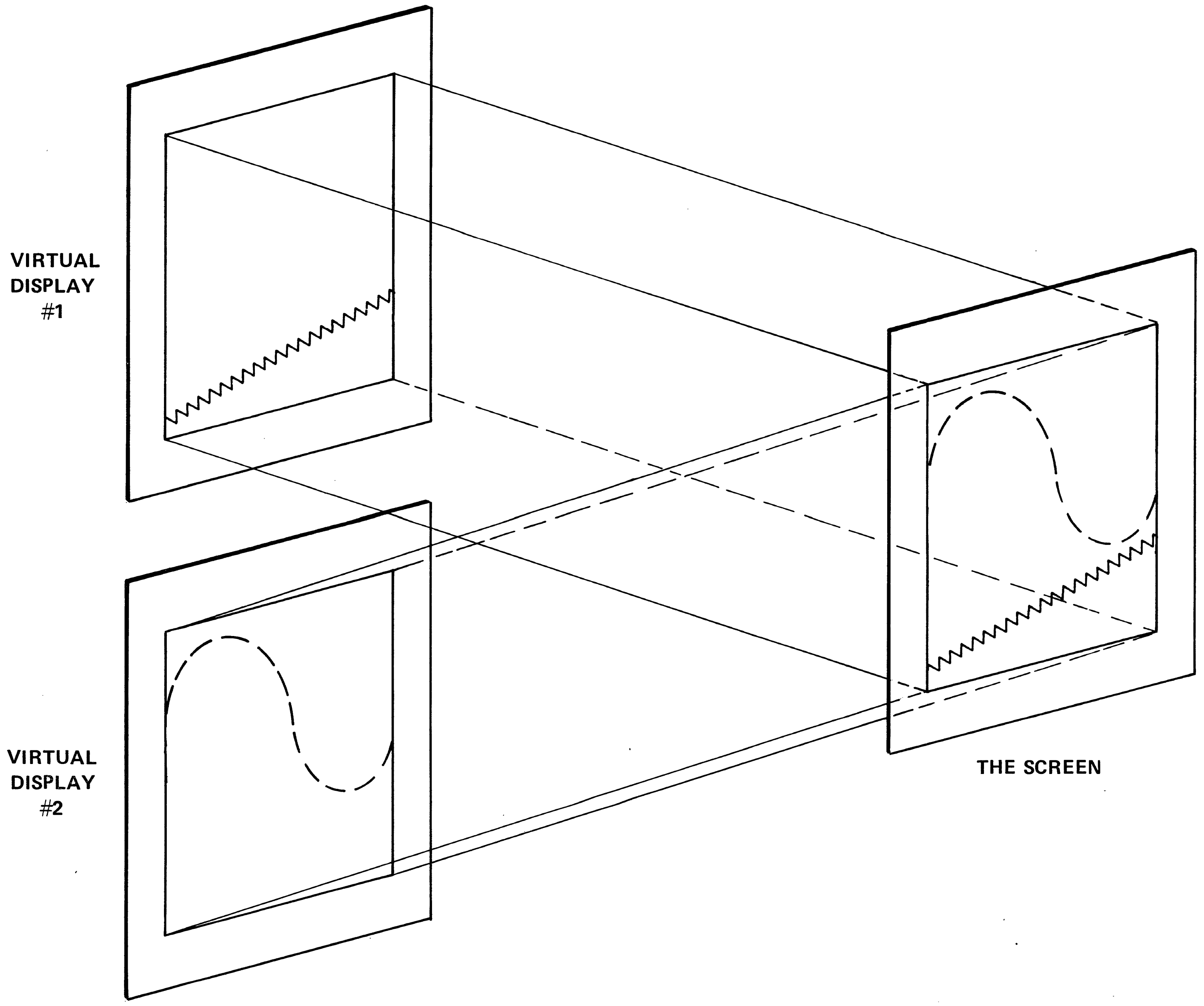


Fig. 3.5. Common Screen Window with several Virtual Displays.

3.2.1 Setting the Virtual Window (continued)

CALLING SEQUENCE:

```
CALL VWINDO (X, XL, Y, YL)
```

where:
- X - Minimum X-Coordinate of the Virtual Window.
- XL - Extent of the Virtual Window in the X-direction.
- Y - Minimum Y-Coordinate of the Virtual Window.
- YL - Extent of the Virtual Window in the Y-direction.

3.2.2 Setting the Screen Window

The Screen Window defines the section of the screen into which the Virtual Window will be transformed. Its definition is similar to that of the Virtual Window.

CALLING SEQUENCE:

```
CALL SWINDO (IX, LX, IY, LY)
```

where:
- IX - Minimum Screen X-Coordinate of the Screen Window.
- LX - Extent of the Screen Window in the X-direction.
- IY - Minimum Screen Y-Coordinate of the Screen Window.
- LY - Extent of the Screen Window in the Y-direction.

3.3 Absolute Vectors

Virtual Graphics allow the user to draw, move, or point plot to any particular point on the Virtual Display with an absolute vector. An absolute vector extends from the current Imaginary Beam position to the location specified by the given Virtual Coordinates, (X,Y). Mode entry and transformation to screen vectors, including windowing and clipping, is automatic.

3.3.1 Draw

A vector may be drawn from the last point on the Virtual Display at which the Imaginary Beam was positioned to a specified point with DRAWA. Only that portion, if any, of the vector which passes through the Virtual Window will be visible. On return from this routine, the Imaginary Beam will be positioned at the given Virtual Coordinates.

3.3.1 Draw (continued)

CALLING SEQUENCE:

```
CALL DRAWA (X,Y)
```

where: X - Virtual X-Coordinate of the point.
Y - Virtual Y-Coordinate of the point.

3.3.2 Move

A move (an invisible vector) to any particular point on the Virtual Display may be made by calling MOVEA. On return from this routine, the Imaginary Beam will be positioned at the given Virtual Coordinates.

CALLING SEQUENCE:

```
CALL MOVEA (X,Y)
```

where: X - Virtual X-Coordinate of the point.
Y - Virtual Y-Coordinate of the point.

3.3.3 Point Plot

A point may be plotted at any location on the Virtual Display with POINTA. Only if the given Virtual Coordinates are within the Virtual Window will a point actually be displayed. On return from this routine, the Imaginary Beam will be positioned at the given Virtual Coordinates.

CALLING SEQUENCE:

```
CALL POINTA (X,Y)
```

where: X - Virtual X-Coordinate of the point.
Y - Virtual Y-Coordinate of the point.
Y - Virtual Y-Coordinate of the

3.3.4 Dash

A dashed line may be drawn from the last point at which the Imaginary Beam was positioned to a specified point with DASHA. Only that portion, if any, which passes through the Virtual Window will be visible. On return from this routine, the Imaginary Beam will be positioned at the given Virtual Coordinate.

3.3.4 Dash (continued)

CALLING SEQUENCE:

```
CALL DASHA (X,Y,L)
```

where: X - Virtual X-Coordinate of the point.
Y - Virtual Y-Coordinate of the point.
L - Dashed line specification.
A dashed line is specified by concatenating integers describing the line segment length and visibility. All codes except 9 should have 2 or more integers.

1	5 raster units, visible.
2	5 raster units, invisible.
3	10 raster units, visible.
4	10 raster units, invisible.
5	25 raster units, visible.
6	25 raster units, invisible.
7	50 raster units, visible.
8	50 raster units, invisible.
9	alternate bright and dark between points.

NOTE: Screen definition does not affect dash size.

3.4 Relative Vectors

Virtual Graphics also allow the user to define a displacement of given length and direction on the Virtual Display through the use of relative vectors. Relative vectors offer the ability to create similar structures at different positions on the Virtual Display with one set of display commands. Conversion of the relative vector to an absolute vector, mode entry, and transformation to screen vectors, including windowing and clipping, is automatic. On return from a relative vector routine, the Imaginary Beam will be located at the point defined by its initial position plus the displacement value.

3.4.1 Draw

A relative vector may be drawn on the Virtual Display from the current Imaginary Beam location with DRAWR. The X and Y displacement values which define the length and direction of the relative vector are input arguments to DRAWR. Only that portion, if any, of the resultant vector which passes through the Virtual Window will be displayed.

```
                          .
                          .
                          .
C*** THIS PROGRAMMING EXAMPLE DEFINES AND ***
C***      OUTLINES THE WINDOW             ***
C***
C***
C*** DEFINE VIRTUAL WINDOW ***
      CALL VWINDO(1ØØ., 2ØØ., -5Ø., 1ØØ.)
C*** DEFINE SCREEN WINDOW ***
      CALL SWINDO(6ØØ, 4ØØ, 4ØØ, 2ØØ)
C*** MOVE TO LOWER LEFT OF WINDOW ***
      CALL MOVEA(1ØØ., -5Ø.)
C*** OUTLINE WINDOW ***
      CALL DRAWA(3ØØ., -5Ø.)
      CALL DRAWA(3ØØ., 5Ø.)
      CALL DRAWA(1ØØ., 5Ø.)
      CALL DRAWA(1ØØ., -5Ø.)
C*** PLOT A POINT IN CENTER OF WINDOW ***
      CALL POINTA(2ØØ., Ø.)
C*** MOVE TO LOWER LEFT OF WINDOW ***
      CALL MOVEA(1ØØ., -5Ø.)
C*** DRAW A DASHED LINE TO THE CENTER OF WINDOW ***
      CALL DASHA(2ØØ., Ø., 12)
C*** CONTINUE DASHED LINE TO LOWER RIGHT
      CALL DASHA(1ØØ., 5Ø., 12)
                          .
                          .
                          .
```

Window Definition and Virtual Absolute Vectors

Example 3.1

3.4.1 Draw (continued)

CALLING SEQUENCE:

```
CALL DRAWR (X,Y)
```

where: X - X-value of the displacement.
Y - Y-value of the displacement.

3.4.2 Move

A relative move on the Virtual Display may be generated by calling MOVER with the X and Y displacements as arguments.

CALLING SEQUENCE:

```
CALL MOVER (X,Y)
```

where: X - X-value of the displacement.
Y - Y-value of the displacement.

3.4.3 Point Plot

Points may also be plotted relative to the current Imaginary Beam location on the Virtual Display. If the resultant point is not within the Virtual Window it will not be displayed.

CALLING SEQUENCE:

```
CALL POINTR (X,Y)
```

where: X - X-value of the displacement.
Y - Y-value of the displacement.

3.4.4 Dash

A dashed line may be drawn on the Virtual Display from the current Imaginary Beam location to a point displaced by X and Y with DASHR. Only that portion, if any, of the line which passes through the Virtual Window will be displayed.

CALLING SEQUENCE:

```
CALL DASHR (X,Y,L)
```

where: X - X-value of the displacement.
Y - Y-value of the displacement.
L - Dashed line specification.

```
                          .
                          .
C*** THIS EXAMPLE DRAWS TWO TRIANGLES OF ***
C***      DIFFERENT SIZE AND ORIENTATION WITH ***
C***      THE SAME RELATIVE VECTORS ***
                          .
                          .
C*** SCALE AND ROTATION FACTORS STILL AT INITIAL VALUE ***
      CALL TRIANG(2ØØ., 2ØØ.)
                          .
                          .
C*** DOUBLE SCALE SIZE ***
      TRSCAL  = 2.
C*** ROTATE 90 DEGREES ***
      TRCOSF = COSD(9Ø.)
      TRSINF = SIND(9Ø.)
C*** REDRAW TRIANGLE ***
      CALL TRIANG(7ØØ., 4ØØ.)
                          .
                          .
      SUBROUTINE TRIANG(X,Y)
C*** INPUT IS CENTER OF TRIANGLE, MUST MOVE ABSOLUTE ***
      CALL MOVEA(X,Y)
C*** MOVE TO LOWER LEFT VERTEX ***
      CALL MOVER(-1ØØ., -1ØØ.)
C*** DRAW TRIANGLE ***
      CALL DRAWR(2ØØ., Ø.)
      CALL DRAWR(-1ØØ., 2ØØ.)
      CALL DRAWR(-1ØØ., -2ØØ.)
C*** RETURN TO CENTER AND PLOT POINT ***
      CALL POINTR(1ØØ., 1ØØ.)
      RETURN
                          .
                          .
                          .
```

Virtual Graphics Relative Vectors, Scaling and Rotating

Example 3.2

3.4.4 Dash (continued)

A dashed line is specified by concatenating integers describing the line segment length and visibility. All codes except 9 should have 2 or more integers.

1	5 raster units, visible.
2	5 raster units, invisible.
3	10 raster units, visible.
4	10 raster units, invisible.
5	25 raster units, visible.
6	25 raster units, invisible.
7	50 raster units, visible.
8	50 raster units, invisible.
9	alternate bright and dark between points.

NOTE: Screen definition does not affect dash size.

3.5 Scaling and Rotating

Relative vectors are used primarily to construct objects or entities which must be displayed at a number of different locations on the Virtual Display. However, the size and orientation of these objects is not always the same. For this reason, relative vectors are automatically scaled and rotated by the relative vector routines according to the scaling factor, TRSCAL, and the rotation factors, TRCOSF and TRSINF. TRSCAL, TRCOSF, and TRSINF are all Terminal Status Area variables. All input arguments to the relative vector routines are unscaled and unrotated. The input arguments define the normal size and orientation for a relative vector. Scaling and rotation will not effect absolute vectors.

3.5.1 Setting the Scale

The relative vector scale factor, TRSCAL, can be used to alter the length of a relative vector. All relative vectors are scaled according to the current value. For example, if a section of relative vector coding will construct a given object and you require the object to be constructed again at twice the normal size, then set TRSCAL to 2.Ø and re-execute the code which will construct the object.

3.5.1 Setting the Scale (continued)

The relative scale factor may be set in the same fashion that any variable is set. It is necessary however that the Terminal Status Area be defined as a set of COMMON variables for reference (see Appendix A). If no scaling is desired, TRSCAL should be assigned the value 1.Ø which is the initial value of TRSCAL set by INITT.

3.5.2 Setting the Rotation

Relative vectors may also have their direction altered through the relative vector rotation factors, TRCOSF and TRSINF. TRCOSF represents the cosine value of the rotation and TRSINF represents the sine value. It should be noted that if the sum of the squares of the cosine and sine values do not equal 1.Ø, then there will be a distortion in length.

All relative factors are rotated according to the values of the current rotation factors. If the user wishes to construct an object defined by relative vectors at an angle different from the normal orientation, he sets the rotation factors to the cosine and sine values of the angle and executes the code for the object.

The relative vector scale factors may be set in the same fashion that any variable is set, as long as the Terminal Status Area be defined as a set of COMMON variables for reference (see Appendix A). If no rotation from the normal orientation of the relative vector is desired, the rotation factors should be set to: TRCOSF = 1.Ø; TRSINF = Ø.Ø. These are also the initial values set by INITT.

3.6 Virtual Cursor

It is often useful to be able to indicate a point on the Virtual Display with the graphic cursor. The routine VCURSR allows the user to do this by enabling the graphic cursor. After the graphic cursor has been positioned, its Screen Coordinates may be transmitted to the computer by striking a keyboard character. VCURSR constructs the Virtual Cursor by transforming the input data into Virtual Coordinates according to the current window definition (Figure 3.6). The Virtual Cursor does not affect the Imaginary or Real Beam position.

3.6 Virtual Cursor (continued)

The transformation assumes that all of the screen is a continuation of the Virtual Display with the scale implied by the current window. This allows the user to receive valid Virtual Coordinate data even if the graphic cursor is positioned outside the current window. However, in such a case, the general error flag KERROR is set to one as an aid to the user. If the graphic cursor is inside the window, KERROR is zero. KERROR is a Terminal Status Area variable.

The keyboard character which triggers input of the graphic cursor's position, is also returned as an argument. This character may be used for command purposes, data identification, or ignored.

CALLING SEQUENCE:

```
CALL VCURSR (IC,X,Y)
```

where: IC - Keyboard character, 7-bit ASCII, right adjusted.
X - Virtual X-Coordinate of graphic cursor.
Y - Virtual Y-Coordinate of graphic cursor.

Note: Cursor control requires an accessory for 4002A.

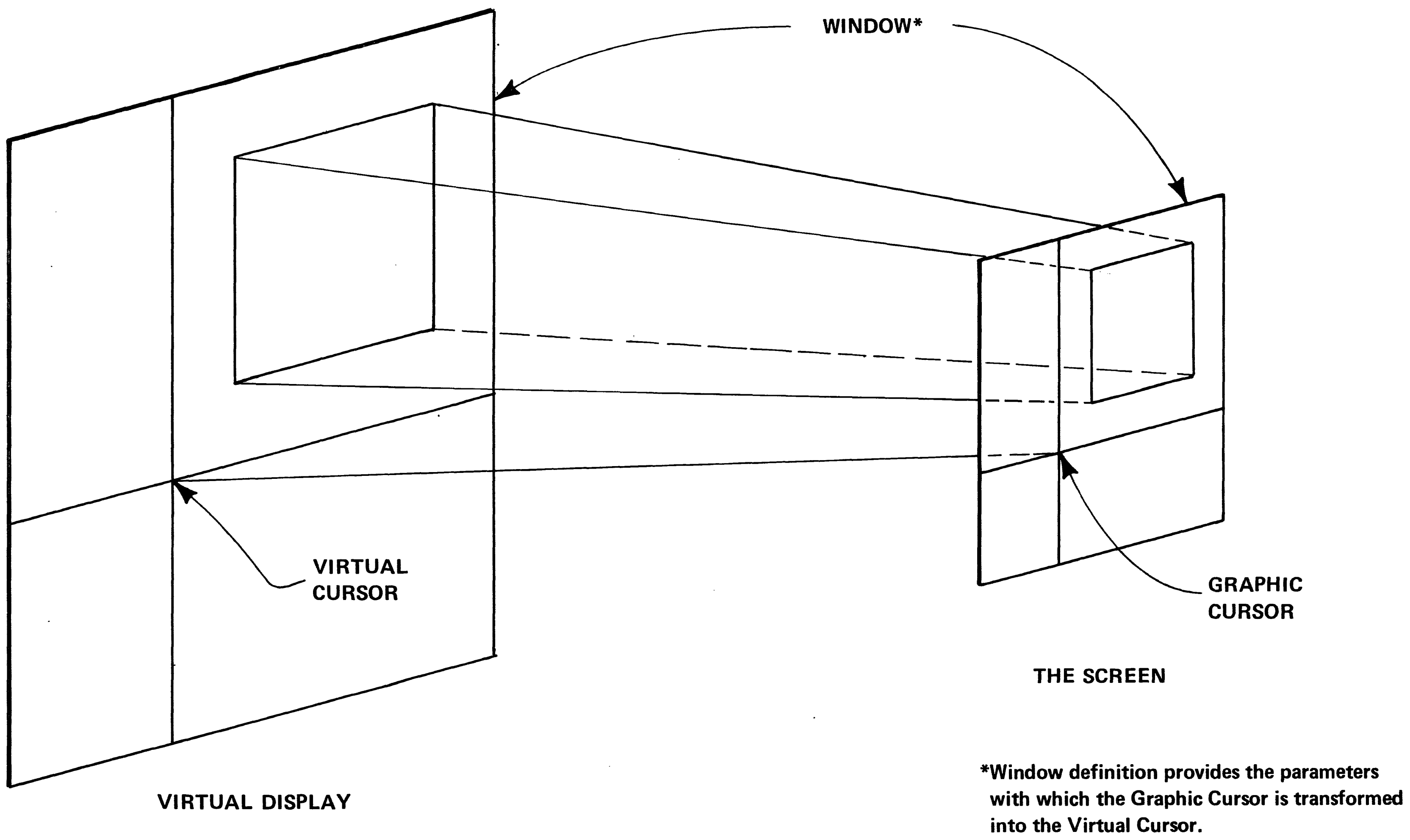

WINDOW*
VIRTUAL CURSOR
GRAPHIC CURSOR
THE SCREEN
VIRTUAL DISPLAY
*Window definition provides the parameters with which the Graphic Cursor is transformed into the Virtual Cursor.

Fig. 3.6. The Virtual Cursor

```
                      .
                      .
                      .
C*** THIS EXAMPLE DRAWS, MOVES, OR POINT PLOTS ***
C***      TO THE INPUT VIRTUAL CURSOR POSITION ***
1ØØ   CALL VCURSR(ICHAR,X,Y)
C*** "D" IMPLIES DRAW ***
      IF(ICHAR.NE.68) GO TO 2ØØ
      CALL DRAWA(X,Y)
      GO TO 4ØØ
C*** "M" IMPLIES MOVE ***
2ØØ   IF(ICHAR.NE.77) GO TO 3ØØ
      CALL MOVEA(X,Y)
      GO TO 4ØØ
C*** "P" IMPLIES POINT PLOT, RE-INPUT FOR ANY OTHER CHAR ***
3ØØ   IF(ICHAR.NE.8Ø) GO TO 1ØØ
      CALL POINTA(X,Y)
4ØØ
                      .
                      .
                      .
```

Virtual Cursor

Example 3.3

4.0 DIRECT GRAPHICS

4.1 The Screen

The terminal screen is a two-dimensional surface consisting of a discrete 1Ø24 x 1Ø24 matrix of addressable points, of which 1Ø24 x 781 of these points lie in the viewable area* of the terminal screen (Figure 1.0). The origin of the screen lies at the extreme lower left corner.

Operations on the screen are called Direct Graphics, and allow the user to relate directly with the visible surface of the terminal. Direct Graphics allow the user to work at a basic graphic level and avoid the overhead of the Virtual clipping and transformation routines. The user has the responsibility of remaining on screen as all coordinate input to Direct Graphic routines are interpreted as MOD 1Ø24.

Direct Graphics are primarily used with alphanumeric output and for display layout. The user may freely alternate between Direct and Virtual graphics. (NOTE: When using a Virtual Graphic routine after use of Direct Graphics or alphanumeric output, the Imaginary Beam is considered to be positioned at the Virtual Coordinate that is equivalent to the Screen Coordinate of the beam position under the current window transformation.)

4.2 Absolute Vectors

An absolute vector in Direct Graphics is a draw, move, or point plot from the current beam position to a specified Screen Coordinate. No windowing or clipping is performed. Mode entry and appropriate output handling is automatic.

4.2.1 Draw

A line may be drawn from the current beam position to any point on the screen with DRWABS. On return from this routine, the beam position is at the given Screen Coordinate.

CALLING SEQUENCE:

```
CALL DRWABS (IX,IY)
```

*Vectors just above 78Ø on the Y-axis may be visible but marginal in quality. For the purposes of this manual such vectors are considered part of the unviewable area.

4.2.1 Draw (continued)

where: IX - Screen X-Coordinate of the given point.
IY - Screen Y-Coordinate of the given point.

4.2.2 Move

The beam may be moved to any point on the Screen with MOVABS.

CALLING SEQUENCE:

```
CALL MOVABS (IX,IY)
```

where: IX - Screen X-Coordinate of the given point.
IY - Screen Y-Coordinate of the given point.

4.2.3 Point Plot

A point may be plotted at any location on the screen with PNTABS. On return, the beam position is at the given Screen Coordinates.

CALLING SEQUENCE:

```
CALL PNTABS (IX,IY)
```

where: IX - Screen X-Coordinate of the given point.
IY - Screen Y-Coordinate of the given point.

4.2.4 Dash

A dashed line may be drawn from the current beam position to any point on the screen with DSHABS. On return from this routine, the beam position is at the given Screen Coordinate.

CALLING SEQUENCE:

```
CALL DSHABS (IX,IY,L)
```

where: IX - Screen X-Coordinate of the given point.
IY - Screen Y-Coordinate of the given point.
L - Dashed line specification.

```
                  .
                  .
                  .
C*** DRAW BOX USING DIRECT GRAPHICS ***
C*** MOVE TO LOWER LEFT OF BOX ***
      CALL MOVABS(2ØØ, 1ØØ)
C*** DRAW SIDES ***
      CALL DRWABS(8ØØ, 1ØØ)
      CALL DRWABS(8ØØ, 65Ø)
      CALL DRWABS(2ØØ, 65Ø)
      CALL DRWABS(2ØØ, 1ØØ)
C*** MOVE TO CENTER
     MOVABS(5ØØ,375)
C*** DRAW DASHED TRIANGLE ***
      CALL DSHABS(5ØØ,15Ø,2325)
      CALL DSHABS(3ØØ,15Ø,2325)
      CALL DSHABS(5ØØ,375,2325)
                  .
                  .
                  .
```

Direct Absolute Vectors

Example 4.1

4.2.4 Dash (continued)

A dashed line ss specified by concatenating integers describing the line segment length and visibility. All codes except 9 should have 2 or more integers.

1	5 raster units, visible.
2	5 raster units, invisible.
3	10 raster units, visible.
4	10 raster units, invisible.
5	25 raster units, visible.
6	25 raster units, invisible.
7	50 raster units, visible.
8	50 raster units, invisible.
9	alternate bright and dark between points.

4.3 Relative Vectors

Relative vectors may also be drawn on the screen. However, no scaling or rotational transformations are applied to these. Mode entry and appropriate output handling is automatic. Direct Graphic relative vectors will cause the beam to move from its present position to the point specified by the direct displacement.

The user again has the responsibility of remaining on the screen. All resultant vectors will have their coordinates interpreted as MOD 1Ø24.

4.3.1 Draw

A relative line may be drawn on the screen from the current beam position according to a given X and Y displacement with DRWREL.

CALLING SEQUENCE:

```
CALL DRWREL (IX,IY)
```

where: IX - X-Displacement in Screen Coordinates.
IY - Y-Displacement in Screen Coordinates.

4.3.2 Move

A relative move may be generated by MOVREL.

CALLING SEQUENCE:

```
CALL MOVREL (IX,IY)
```

where: IX - X-Displacement in Screen Coordinates.
IY - Y-Displacement in Screen Coordinates.

```
              :
C*** THIS EXAMPLE FILLS SCREEN WITH TREES ***
      DO 1ØØ IX = Ø, 9ØØ, 1ØØ
      DO 1ØØ IY = Ø, 78Ø, 15Ø
C*** POSITION TREE START ***
      CALL MOVABS(IX+2Ø,IY+4Ø)
      CALL TREE
1ØØ   CONTINUE
              :

      SUBROUTINE TREE
C*** DRAW TREE BODY ***
      CALL DRWREL(6Ø,Ø)
      CALL DRWREL(-3Ø,6Ø)
      CALL DRWREL(-3Ø,-6Ø)
C*** DRAW FRUIT ***
      CALL PNTREL(3Ø,4Ø)
      CALL PNTREL(-1Ø,-2Ø)
      CALL PNTREL(2Ø,Ø)
C*** DRAW TRUNK ***
      CALL MOVREL(-1Ø,-2Ø)
      CALL DRWREL(Ø,-4Ø)
      RETURN
      END
              :
```

Direct Relative Vectors

Example 4.2

4.3.3 Point Plot

A point may be plotted relative to the current beam position with PNTREL.

CALLING SEQUENCE:

```
CALL PNTREL(IX,IY)
```

where: IX - X-Displacement in Screen Coordinates.
IY - Y-Displacement in Screen Coordinates.

4.3.4 Dash

A dashed line may be drawn on the Screen relative to the current beam position according to a given X and Y displacement with a DSHREL.

CALLING SEQUENCE:

```
CALL DSHREL (IX,IY,L)
```

where: IX - X-Displacement in Screen Coordinates.
IY - Y-Displacement in Screen Coordinates.
L - Dashed line specification.
A dashed line is specified by concatenating integers describing the line segment length and visibility. All codes except 9 should have 2 or more integers.

1	5 raster units, visible.
2	5 raster units, invisible.
3	10 raster units, visible.
4	10 raster units, invisible.
5	25 raster units, visible.
6	25 raster units, invisible.
7	50 raster units, visible.
8	50 raster units, invisible.
9	alternate bright and dark between points.

4.4 Units of Length

Direct Graphics allow the specification of points in inches and centimeters as well as Screen Coordinates through the use of conversion functions. This allows the user to specify output with reference to a familiar length.

4.4.1 Inches

The functional routine KIN is used to transform inches to Screen Coordinates. The input argument is the number of inches specified as a single precision real variable. The function then has the integer value of the appropriate number of Screen Coordinates.

Example:

```
IX = KIN(3.5)
```

where: IX would be assigned the number of Screen Coordinates equal to 3.5 inches.

4.4.2 Centimeters

The functional routine KCM similarly transforms centimeters to Screen Coordinates.

Example:

```
IX = KCM(3.5)
```

where: IX would be assigned the number of Screen Coordinates equal to 3.5 centimeters.

4.5 Direct Cursor Input

The graphic cursor may be used to specify Screen Coordinates directly. Calling DCURSR will activate the graphic cursor, allowing the user to position it. The cursor position is transmitted to the computer when a keyboard character is struck. This character along with the position input is returned as arguments by DCURSR. The graphics cursor position does not affect the beam position.

CALLING SEQUENCE:

```
CALL DCURSR (IC,IX,IY)
```

where: IC - Keyboard character, 7-bit ASCII, right-adjusted.
IX - Screen X-Coordinate of graphic cursor.
IY - Screen Y-Coordinate of graphic cursor.

4.6 Incremental Plotting (Restricted to 4002A Terminals)

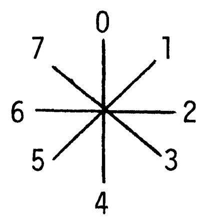
This routine is used to perform incremental plotting. The user specifies the direction, whether it is to be visible or invisible, and the number of times he wishes this plot character to be output.

CALLING SEQUENCE:

```
CALL INCPLT (IONOFF,IDIR,NO)
```

where:	IONOFF	= Ø; Beam off (invisible). = 1; Beam on (visible).
	IDIR	= Direction code (0-7; see below).
	NO	= Number of times plot character is to be repeated.

Direction codes:



Each incremental plot character will move the beam one raster unit in the given direction.

```
                    .
                    .
                    .
C*** THIS EXAMPLE DRAWS A 5" X 4" BOX ***
C***      CENTERED ON SCREEN ***
C*** MOVE TO LOWER LEFT OF BOX ***
      CALL MOVABS(KIN(1.25), KIN(Ø.8))
C*** DRAW BOX ***
      CALL DRWREL(KIN(5.),Ø)
      CALL DRWREL(Ø,KIN(4.1))
      CALL DRWREL(KIN(-5.),Ø)
      CALL DRWREL(Ø,KIN(-4.))
C***
C*** NOW DRAW 5 CM. by 4 CM. BOX, CENTERED ***
C*** MOVE TO LOWER LEFT OF BOX ***
      CALL MOVABS(KCM(7.), KCM(5.15))
C*** DRAW BOX ***
      CALL DRWREL(KCM(5.),Ø)
      CALL DRWREL(Ø,KCM(4.))
      CALL DRWREL(KCM(-5.),Ø)
      CALL DRWREL(Ø,KCM(-4.))
                    .
                    .
                    .
```

Units of Length

Example 4.3

```
                   :
C*** THIS EXAMPLE DRAWS, MOVES, OR POINT PLOTS ***
C***      TO THE DIRECT CURSOR INPUT POSITION ***
1ØØ   CALL DCURSR(ICHAR,IX,IY)
C*** "D" IMPLIES DRAW ***
     IF(ICHAR.NE.68) GO TO 2ØØ
     CALL DRWABS(IX,IY)
     GO TO 4ØØ
C*** "M" IMPLIES MOVE ***
2ØØ  IF(ICHAR.NE.77) GO TO 3ØØ
     CALL MOVABS(IX,IY)
     GO TO 4ØØ
C*** "P" IMPLIES POINT PLOT, RE-INPUT FOR ANY OTHER CHAR ***
3ØØ  IF(ICHAR.NE.8Ø) GO TO 1ØØ
     CALL PNTABS(X,Y)
4ØØ
                   :
```

Direct Cursor Input

Example 4.4

5.0 A/N OUTPUT

By allowing the Terminal Control System to monitor alphanumeric output, it is possible to maintain terminal status especially the tracking of the beam position, which is required for tab and margin control as well as facilitating the mixture of A/N and vector output.

5.1 Entering A/N Mode

At times the user may wish to output A/N data other than through the Terminal Control System. In such cases it is the user's responsibility to insure that the terminal is in A/N mode. This can be done without the output of extraneous data by using ANMODE. It is not necessary to call ANMODE when using the Terminal Control System routines as they will automatically enter A/N mode whenever necessary.

CALLING SEQUENCE:

```
CALL ANMODE
```

5.2 A/N Character Output

Non-control alphanumeric characters are monitored when output through ANCHO. A/N mode will be entered if necessary and the Terminal Status Area representation of the beam position is updated as characters are output. If the outputting of the character advances the beam beyond the right margin setting, a new line is automatically generated. This routine does NOT check the input variable which is assumed to be a 7-bit ASCII non-control character right-adjusted within an integer word. Any other input will result in erroneous beam status information.

CALLING SEQUENCE:

```
CALL ANCHO(ICHAR)
```

where: ICHAR - 7-bit ASCII non-control character, right-adjusted.

5.3 New Line

A new line may be generated with NEWLIN. The alphanumeric mode will be entered if necessary and the A/N cursor will be moved to the left margin and down one line.

CALLING SEQUENCE:

```
CALL NEWLIN
```

```
                .
                .
                .
C*** MOVE TO CLEAN AREA FOR FORMATTED INPUT ***
      CALL MOVABS(6ØØ, 1ØØ)
C*** ENTER A/N MODE ***
      CALL ANMODE
C*** REQUEST USER INPUT ***
      TYPE 1ØØ
1ØØ   FORMAT(' INPUT:')
      ACCEPT 2ØØ, IVAR
2ØØ   FORMAT(I5)
C*** GO TO DRAWING AREA ***
      CALL MOVABS(IX,IY)
                .
                .
                .
```

Using A/N Mode for Formatted Input

Example 5.1

5.4 Carriage Return

The A/N cursor can be moved directly to the left margin without moving down a line with CARTN. A/N mode is entered automatically if necessary.

CALLING SEQUENCE:

```
CALL CARTN
```

5.5 Line Feed

Similarly the A/N cursor may be moved down a line without returning to the left margin with LINEF.

CALLING SEQUENCE:

```
CALL LINEF
```

5.6 Backspace

The A/N cursor may be moved back one character location with the backspace routine. The backspace routine may be used to move the A/N cursor to the left of a non-zero left margin, but will cause the cursor to "wrap-around" if the cursor is backed up beyond the zero X-axis location.

CALLING SEQUENCE:

```
CALL BAKSP
```

5.7 Home

The HOME routine will move the A/N cursor to the left margin at the home Y-position without erasing the current display.

CALLING SEQUENCE:

```
CALL HOME
```

5.8 New Page

The routine NEWPAG will cause the screen to be erased and will move the A/N cursor to the left margin at the home Y-location.

CALLING SEQUENCE:

```
CALL NEWPAG
```

5.9 Italic Mode (Restricted to 4002A Terminals)

This routine will output the proper control character to enter italic mode. This routine does not enter alphanumeric mode automatically.

CALLING SEQUENCE:

```
CALL ITALIC
```

5.10 Italic Mode Reset (Restricted to 4002A Terminals)

Resets to non-italic mode and enters alphanumeric mode. Double size mode is not affected by this routine.

CALLING SEQUENCE:

```
CALL ITALIR
```

5.11 Double Size Mode (Restricted to 4002A Terminals)

This routine will output the proper control character to enter double size mode. This routine does NOT enter alphanumeric mode automatically.

CALLING SEQUENCE:

```
CALL DBLSIZ
```

5.12 Normal Size Mode (Restricted to 4002A Terminals)

Resets to normal size characters and enters alphanumeric mode. Italic mode is not affected by this routine.

CALLING SEQUENCE:

```
CALL NRMSIZ
```

NOTE: Italic and double size modes are set and reset only by the above routines. Entering graphic, point plot, or incremental plot modes will not affect these settings.

5.13 Character Size

The subroutine returns the size of the character for use in label positioning and other operations dependent on character size. When used, it enables applications and software using TCS to maintain terminal independence.

CALLING SEQUENCE:

```
CALL CSIZE (IHORZ,IVERT)
```

5.13 Character Size (continued)

where: IHORZ - Height of the character in screen coordinates.

IVERT - Width of the character in screen coordinates.

```
                :
C*** FILL HEADER(8)
      DATA HEADER/68,88,65,77,8Ø,76,69,32/
      CALL CSIZE(KHORSZ,KVERSZ)
                :
C*** GET NEW PAGE ***
      CALL NEWPAG
C*** GO TO HEADER POSITION ***
      CALL MOVREL (KIN(3.),Ø)
C*** OUTPUT HEADER
      DO 1ØØ I = 1,8
1ØØ   CALL ANCHO(HEADER(I))
C*** DRAW BOX AROUND HEADER ***
      CALL DRWREL(Ø,KVERSZ)
      CALL DRWREL(-9*KHORSZ,Ø)
      CALL DRWREL(Ø,-KVERSZ-3)
      CALL DRWREL(9*KHORSZ,Ø)
C*** RETURN TO LEFT MARGIN AND GO DOWN 2 LINES ***
      CALL NEWLIN
      CALL LINEF
                :
C*** OVERPRINT "O" WITH "X" ***
      CALL ANCHO(79)
      CALL BAKSP
      CALL ANCHO(88)
                :
C*** RETURN TO LEFT MARGIN, DO NOT GO DOWN A LINE ***
      CALL CARTN
                :
C*** RETURN TO HOME POSITION
      CALL HOME
                :
```

<u>NOTE:</u> KHORSZ and KVERSZ are variables containing horizontal and vertical character size, respectively, in Screen Coordinates.

A/N Output (mixed with Graphics)

Example 5.2

6.0 TABS AND MARGINS

The Terminal Control System allows the user to set and reset tabs and margins to facilitate format layout. The tabs and margin settings are software generated and as such are only useful for A/N data output through Terminal Control System routines. All tab and margin values are in Screen Coordinates.

Both horizontal and vertical tabs and left and right margins are available. Horizontal and vertical tabs are limited to ten positions each.

6.1 Tab Setting

Tab settings for both horizontal and vertical tabs are kept in two ten-word integer arrays. The settings are ordered with ascending Screen X-Coordinates with the first zero value indicating the end of the settings.

6.1.1 Set Tab Routine

The routine SETTAB takes a given tab setting in Screen Coordinates and inserts it into the given tab table. If the tab table is full, the maximum setting will be lost in order that a lesser tab setting may be inserted. When this occurs, the general error flag, KERROR, is set. Although duplicate tab settings are not inserted, SETTAB does not generally check the tab setting for validity and does not check if the given tab table is KHORZT or KVERTT, the horizontal and vertical tab tables respectively.

CALLING SEQUENCE:

```
        CALL SETTAB(ITAB,ITABLE)
```

where:	ITAB -	Tab setting in either X or Y Screen Coordinates.
	ITABLE -	Horizontal or vertical tab table (i.e. KHORZT, KVERTT).

6.1.2 Setting Through COMMON

Both the horizontal and vertical tab table (KHORZT(1Ø) and KVERTT(1Ø) respectively) can be set directly if the Terminal Status Area is available as a set of common variables (see Appendix A). If set directly, the user must insure that the tabs are in increasing order with the first zero value following the valid tab settings. Negative values should never be used.

6.2 Tab Resetting

6.2.1 Reset Single Tab

To selectively reset a tab, its position in Screen Coordinates must be input to the tab resetting routine with the given tab table. Non-zero values which do not correspond to a current tab setting are ignored.

CALLING SEQUENCE:

```
          CALL RSTTAB(ITAB,ITABLE)
```

where: ITAB - X or Y Screen Coordinate of tab to be reset.
ITABLE - Horizontal or vertical tab table. (i.e. KHORZT,KVERTT).

6.2.2 Reset All Tabs

An entire tab table may be reset by using a zero for the tab position to be reset.

CALLING SEQUENCE:

```
          CALL RSTTAB(Ø,ITABLE)
```

where: ITABLE - Horizontal or vertical tab table.(i.e. KHORZT,KVERTT)

6.3 Horizontal Tab

Calling the horizontal tab routine will cause the alphanumeric cursor to be moved with a constant Y-value to the position specified by the first non-zero entry in the horizontal tab table, KHORZT, which is greater than the current Screen X-Coordinate of the cursor or beam position. If the horizontal tab table is empty, no action will occur. If the tab table is not empty and no entry exists which is greater than the current Screen X-Coordinate of the cursor or beam position, or if the first non-zero entry greater than the Screen X-Coordinate is also greater than the right margin setting, a new line will be generated.

CALLING SEQUENCE:

```
          CALL TABHOR
```

6.4 Vertical Tab

Vertical tabbing will cause the alphanumeric cursor to be moved with a constant X-value to the position specified by the last non-zero entry in the vertical tab table, KVERTT, which is less than the current Y-Coordinate of the cursor or beam position. If no entry in the vertical tab table exists which is non-zero and yet less than the current Y-Coordinate, then no action occurs.

CALLING SEQUENCE:

```
CALL TABVER
```

6.5 Margins

6.5.1 Left Margin

The left margin is the Screen X-Coordinate at which a line of A/N output starts. The Carriage Return, Home, and New Page routines cause the A/N cursor to move to the current left margin.

The left margin setting is contained in the Terminal Status Area variable, KLMRGN (see Appendix A). KLMRGN may be set in the same manner as any other variable. However, its value should always be greater than Ø and less than the right margin value. The initial value of the left margin as set by INITT is Ø.

6.5.2 Right Margin

The right margin is the rightmost position at which A/N output may be output. Any attempt to output beyond the right margin using the A/N output routine will cause a new line to be generated.

The right margin value is a Screen X-Coordinate and is contained in the Terminal Status Area variable, KRMRGN. KRMRGN may be set in the same manner as any other variable.

However, its value should always be less than 1Ø23 and greater than the left margin variable. The initial value as set by INITT is 1Ø1Ø.

```
                    .
                    .
                    .
C*** THIS EXAMPLE ILLUSTRATES TABBING CAPABILITY ***
C*** SET HORIZONTAL TABS USING TAB SETTING ROUTINE ***
      DO 1ØØ I = 1ØØ, 6ØØ, 1ØØ
1ØØ   CALL SETTAB(I,KHORZT)
C*** SET VERTICAL TABS DIRECTLY THROUGH COMMON
      DO 2ØØ I = 1,4
2ØØ   KVERTT(I) = KHOMEY-(4-I)*1ØØ
C*** INSURE UNUSED PORTION OF VERTICAL TAB TABLE IS ZERO ***
      DO 3ØØ I = 5,1Ø
3ØØ   KVERTT(I) = Ø
C*** OUTPUT CHAR'S ***
      DO 42Ø I = 1,4
      DO 41Ø J = 1,1Ø
C*** OUTPUT 'ABCDE' ***
      DO 4ØØ K = 1,5
4ØØ   CALL ANCHO(K+64)
41Ø   CALL TABHOR
      CALL TABVER
C*** RETURN TO LEFT MARGIN AT VERTICAL TAB POSITION ***
42Ø   CALL CARTN
C*** RESET EVERY OTHER HORIZONTAL TAB ***
      DO 5ØØ I = 1ØØ, 6ØØ, 2ØØ
5ØØ   CALL RSTTAB(I,KHORZT)
      DO 51Ø I = 1,3
51Ø   CALL NEWLIN
C*** OUTPUT CHAR'S ***
      DO 62Ø I = 1,2
      DO 61Ø J = 1,1Ø
C*** OUTPUT 'ABCDE' ***
      DO 6ØØ K = 1,5
6ØØ   CALL ANCHO(K+64)
61Ø   CALL TABHOR
      CALL NEWLIN
      CALL LINEF
                    .
                    .
                    .
C*** RESET ALL TABS ***
      CALL RSTTAB(Ø,KHORZT)
      CALL RSTTAB(Ø,KVERTT)
                    .
                    .
                    .
```

Using the Tab Routines

Example 6.1

```
              :
C*** THIS EXAMPLE OUTPUTS CHAR'S IN 3 COLUMNS ***
C***      EACH COLUMN IS 1Ø CHARACTERS WIDE ***
      DO 2ØØ I = 1,3
C*** SET MARGINS ***
      KLMRGN = 1ØØ+2ØØ*(I-1)
      KRMRGN = KLMRGN+1Ø*KHORSZ
C*** RETURN TO CURRENT HOME POSITION
      CALL HOME
      DO 2ØØ J = 1,5
      DO 1ØØ K = 1,15
1ØØ   CALL ANCHO(K+64)
2ØØ   CALL NEWLIN
              :
```

Using Margins

Example 6.2

7.0 A/N INPUT

A/N characters may be input one at a time through the general input routine, TINPUT, Characters input will be in 7-bit ASCII and right adjusted. TINPUT will not cause an echo* to be generated and no beam movement will occur. This allows the user to interact with his program while in vector mode.

CALLING SEQUENCE:

```
                    CALL TINPUT(ICHAR)
```

where: ICHAR - 7-bit ASCII character right adjusted.

NOTE: If the user wishes to input data other than through the Terminal Control System routines, he should position the beam at an appropriate position, and enter A/N mode before requesting his input. Also he should expect a non-monitored echo of his input data to occur.

*Check the Terminal Control System Implementation Notes for more information regarding this matter.

```
                :
C*** THIS EXAMPLE INPUTS AN UNPACKED STRING ***
      DO 1ØØ I = 5
C*** INPUT CHARACTER ***
      CALL TINPUT(ISTRNG(I))
C*** ECHO INPUT CHARACTER ***
1ØØ     CALL ANCHO(ISTRNG(I))
                :
```

A/N Input

Example 7.0

8.0 TERMINAL STATUS

The Terminal Status Area is a set of variables which are kept in a common block (see Appendix A) and represent the current state of the terminal. The Terminal Control System allows the user to save the current terminal status and return to it at a later time.

Although it does not save the displayed data, this facility does allow the user to interrupt his processing, move to another location, do other processing there or interact with the user, and then return to his original processing.

Since the user allocates the save areas, he may easily save more than one level of status and may restore any of his saved states at any time.

8.1 Save Status

The current state of the terminal may be saved by providing the status saving routine with a 6Ø-word real array in which the current Terminal Status Area may be stored.

CALLING SEQUENCE:

```
CALL SVSTAT(ARRAY)
```

where: ARRAY - 6Ø-word real array.

8.2 Restore Status

The terminal may be restored to any previously saved state at any time by providing the status restoring routine with the 6Ø-word real array in which the previous Terminal Status Area was stored.

CALLING SEQUENCE:

```
CALL RESTAT(ARRAY)
```

where: ARRAY - 6Ø-word real array containing previously stored terminal state.

```
                          .
                          .
C*** THIS EXAMPLE SAVES STATUS DURING FORMATTED I/O ***
C***      AND THEN RESTORES STATUS ***
                          .
                          .
      DIMENSION SAVE1(6Ø)
                          .
                          .
C*** SAVE CURRENT TERMINAL STATUS ***
      CALL SVSTAT(SAVE1)
C*** MOVE TO UNUSED AREA OF SCREEN ***
      CALL MOVABS(IX,IY)
      CALL ANMODE
      TYPE 1ØØ

1ØØ   FORMAT('INPUT DATA:')
      ACCEPT 2ØØ, IVAR1, IVAR2
2ØØ   FORMAT(2I5)
                          .
                          . (Interpret Input)
                          .
C*** RESTORE STATUS
      CALL RESTAT(SAVE1)
                          .
                          . (Continue Graphic Processing)
                          .
```

Use of Status Routines

Example 8.0

9.0 SCRATCHPAD SUPPORT (Restricted to 4002A Terminals)

One of the major features of the 4002A terminal is the computer-addressable scratchpad. Some basic routines are included in the 4002A version of the Terminal Control System to assist in the use of the scratchpad. These are described below.

NOTE: It is firmly recommended that status be saved before using the scratchpad and restored after use.

9.1 Enter Scratchpad Mode

This routine enters scratchpad mode. Future output will be directed to the scratchpad until this mode is left. Display of data output to the scratchpad will not occur until scratchpad mode is exited (see ENLCM and EDITSP below).

CALLING SEQUENCE:

```
CALL ENSPM
```

9.2 Clear Scratchpad

The scratchpad is cleared and the scratchpad cursor is set to the beginning of the buffer.

CALLING SEQUENCE:

```
CALL CLRSP
```

9.3 Enter Local Compose Mode

Scratchpad mode is exited, the scratchpad data is displayed, and local compose mode is entered. The user may now modify the output or clear and enter his own data. When he presses the SEND button while still in local compose mode, the entire buffer will be sent to the computer. After calling this routine, the program should be set for input as all output will be ignored until a reply from the user has been received.

CALLING SEQUENCE:

```
CALL ENCLM
```

9.4 Enter Local Edit Mode

The scratchpad mode is exited, the output data is displayed, (with a terminating question mark), and local edit mode is entered. The user may now enter his own data. If he remains in local edit mode and presses the SEND button, only the data entered after the computer output will be returned to the computer. After calling this routine, the program should be set for input as all output will be ignored until a reply has been received.

CALLING SEQUENCE:

```
CALL EDITSP
```

APPENDIX A

Terminal Control System Common (Global) Variables

A.1 TERMINAL STATUS AREA

The Terminal Control System maintains a representation of the current state of the terminal and the user's output mode and level with a set of common (or global) variables referred to as the Terminal Status Area. The Terminal Status Area should be set up in each implementation of the Terminal Control System as a block of common storage, easily accessible to all user routines.

Some of the information contained within the Terminal Status Area, such as character width (KHORSZ) and height (KVERSZ), provide a significant aid for all and increase the ability to program in a terminal independent fashion. Other variables, such as the relative vector scale (TRSCAL) and rotation factors (TRCOSF, TRSINF), and the margin variables (KLMRGN, KRMRGN), must be available to the routines which require use of these facilities. The sophisticated user of the Terminal Control System will also find that the information in, and the appropriate use of, the other variables will significantly increase his programming capability.

All of the Terminal Status Area variables are not used in all implementations. However, in order to retain consistency and increase the ease of transference of application software from one system to another, it is required that the standard Terminal System Area layout indicated below be used by all.

Two names have been assigned to each Terminal Status Area variable and appear in the upper left of the description paragraph. The first is the normal 6-character name. The second is a 4-character name to be used for those implementations which do not permit a full 6-character name. In all Terminal Control System documentation, the Terminal Status Area variables will be referenced by the 6-character name.

A.2 COMMON LAYOUT

The Terminal Status Area is defined below as a labeled COMMON block as used in FORTRAN IV implementations. The name of the COMMON block is TKTRNX for all such implementations. The order of the variables in the COMMON block for all implementations is the same as that described in the Floating Point COMMON below, with the only difference being that which exists for Fixed Point COMMON, also described below.

A.2 COMMON LAYOUT (continued)

All Terminal Status Area variables for implementation which utilize floating point will be integer or real according to the implicit FORTRAN definition associated with their names. Although, the same names will be retained (with the exception of TRSCAL), all Terminal Status Area variables will be integers for those implementations with processing restricted to integer arithmetic.

Floating Point COMMON:

```
      COMMON /TKTRNX/ KBAUDR,KERROR,KGRAFL,KHOMEY,KKMODE,
     1 KHORSZ,KVERSZ,KITALC,KSIZEF,KLMRGN,KRMRGN,
     2 KTBLSZ,KHORZT(1Ø),KVERTT(1Ø),
     3 KBEAMX,KBEAMY,KMOVEF,KPCHAR(4),KDASHT,
     4 KMINSX,KMINSY,KMAXSX,KMAXSY,TMINVX,TMINVY,TMAXVX,TMAXVY,
     5 TREALX,TREALY,TIMAGX,TIMAGY,TRCOSF,TRSINF,TRSCAL
```

Fixed Point COMMON:

Same as Floating Point COMMON (with all variables defined as integers) except for line 5:

```
     5 TREALX,TREALY,TIMAGX,TIMAGY,TRCOSF,TRSINF,KUPSCA,KDWNSC
```

where the two-variable integer scale factor replaces the real single variable scale factor.

A.3 GENERAL VARIABLES

The following variables are generally used throughout the Terminal Control System.

A.3.1 Baud Rate — KBAUDR, KBDR

The number of characters per second which can be transmitted to the terminal. For directly connected terminals, this variable will have a zero value.

A.3.2 General Error Flag

The flag set or reset by various Terminal Control System routines to indicate whether or not certain anomalistic conditions occurred.

A.3.3 Graphic Level Flag — KGRAFL, KGFL

Flag which indicates the user is currently in Virtual Graphics mode when set. When reset, user is assumed to be at Direct Graphic Level.

A.3.4 Home Y-Value — KHOMEY, KHMY

Screen Y-Coordinate of the terminal home position.

A.3.5 Mode — KKMODE, KMOD

Status variable indicating current terminal mode:

Ø - Alphanumeric
1 - Vector
2 - Point Plot
3 - Incremental Plot*
4 - Dash

A.4 A/N VARIABLES

The following variables are used primarily in the processing of A/N data of the Terminal Control System.

A.4.1 Character Horizontal Size — KHORSZ, KHSZ

Number of Screen Coordinates that the beam is horizontally displaced when a hardware-generated character is output.

A.4.2 Character Vertical Size — KVERSZ, KVSZ

Number of Screen Coordinates that the beam is vertically displaced when a hardware-generated line feed is output.

A.4.3 Italic Flag* — KITALC, KITL

Flag set to indicate the enabling of italic output.

A.4.4 Size Flag* — KSIZEF, KSIZ

Flag set to indicate the enabling of double size alphanumeric output.

*Used with implementations supporting the Tektronix 4002A Graphic Computer Terminal.

A.4.5 Left Margin — KLMRGN, KLMG

Left margin setting as a Screen X-Coordinate.

A.4.6 Right Margin — KRMRGN, KRMG

Right margin setting as a Screen X-Coordinate.

A.4.7 Tab Table Size — KTBLSZ, KTBS

The number of words in each of the tab tables.

A.4.8 Horizontal Tab Table — KHORZT, KHOT

Ten word integer array containing the current horizontal tab settings. The entries must be Screen X-Coordinate values in ascending order. The first zero value is used to indicate the end of the tab settings.

A.4.9 Vertical Tab Table — KVERTT, KVET

Ten word integer array containing current vertical tab settings. The entries must be Screen Y-Coordinate values in ascending order. The first zero value is used to indicate the end of the tab settings.

A.5 DIRECT GRAPHIC VARIABLES

The following variables are used at the basic graphic output level.

A.5.1 Beam X-Coordinate — KBEAMX, KBMX

The Screen X-Coordinate of the current storage beam position. Updated whenever beam is moved through output to the terminal.

A.5.2 Beam Y-Coordinate — KBEAMY, KBMY

The Screen X-Coordinate of the current storage beam position. Updated whenever beam is moved through output to the terminal.

A.5.3 Move Flag — KMOVEF, KMVF

Flag set to indicate terminal is primed for a blank vector when in vector mode.

A.5.4 Previous Plot Characters — KPCHAR, KPCH

Four word integer array containing the plot characters which define the last vector or point plot output.

A.5.5 Dashed Line Specification — KDASHT, KDST

Defines the lengths of the visible and invisible portions of a dashed line.

A.6 VIRTUAL GRAPHIC VARIABLES

The following variables are used in conjunction with Basic Graphic output.

A.6.1 Screen Window Minimum X — KMINSX, KSX1

Minimum Screen X-Coordinate of the current Screen Window.

A.6.2 Screen Window Minimum Y — KMINSY, KSY1

Minimum Screen Y-Coordinate of the current Screen Window.

A.6.3 Screen Window Maximum X — KMAXSX, KSX2

Maximum Screen X-Coordinate of the current Screen Window.

A.6.4 Screen Window Maximum Y — KMAXSY, KSY2

Maximum Screen Y-Coordinate of the current Screen Window.

A.6.5 Virtual Window Maximum X — TMINVX, TVX1

Minimum Virtual X-Coordinate of the current Virtual Window.

A.6.6 Virtual Window Minimum Y — TMINVY, TVY1

Minimum Virtual Y-Coordinate of the current Virtual Window.

A.6.7 Virtual Window Maximum X — TMAXVX, TVX2

Maximum Virtual X-Coordinate of the current Virtual Window.

A.6.8 Virtual Window Maximum Y — TMAXVY, TVY2

Maximum Virtual Y-Coordinate of the Virtual Window.

A.6.9 Real Beam X TREALX, TRLX

Virtual X-Coordinate of the current Real Beam position.

A.6.10 Real Beam Y TREALY, TRLY

Virtual Y-Coordinate of the current Real Beam position.

A.6.11 Imaginary Beam X TIMAGX, TIMX

Virtual X-Coordinate of the current Imaginary Beam position.

A.6.12 Imaginary Beam Y TIMAGY, TIMY

Virtual Y-Coordinate of the current Imaginary Beam position.

A.6.13 Relative Vector Cosine Factor TRCOSF, TRCF

Cosine value used for rotation of relative vectors on the Virtual Display.

A.6.14 Relative Vector Sine Factor TRSINF, TRSF

Sine value used for rotation of relative vectors on the Virtual Display.

A.6.15 Relative Vector Scale Factor TRSCAL, TRSC

Value used for the scaling of relative vectors on the Virtual Display. (For implementations utilizing floating point.)

A.6.15a Relative Vector Up Scale Factor KUPSCA, KUPS

Numerator value of scaling factor for relative vectors on The Virtual Display. (For implementations where only integer arithmetic is available.

A.6.15b Relative Vector Down Scale Factor KDWNSC, KDWN

Denominator value of scaling factor for relative vectors on the Virtual Display. (For implementations where only integer arithmetic is available.)

A.7 VARIABLE NAMES IN ALPHABETICAL ORDER

Name	Use	Description
KBAUDR	General	Characters per Second
KBEAMX	Direct Graphics	Beam X-Coordinate
KBEAMY	Direct Graphics	Beam Y-Coordinate
KDASHT	Virtual Graphics	Dash Specification
KDWNSC*	Virtual Graphics	Relative Vector Down Scale Factor
KERROR	General	General Error Flag
KGRAFL	General	Graphic Level Flag
KHOMEY	General	Home Y-Value
KHORSZ	A/N	Character Horizontal Size
KHORZT	A/N	Horizontal Tab Table
KITALC**	A/N	Italic Flag
KKMODE	General	Mode
KLMRGN	A/N	Left Margin
KMAXSX	Virtual Graphics	Screen Window Maximum X
KMAXSY	Virtual Graphics	Screen Window Maximum Y
KMINSX	Virtaul Graphics	Screen Window Minimum X
KMINSY	Virtual Graphics	Screen Window Minimum Y
KMOVEF	Direct Graphics	Move Flag
KPCHAR	Direct Graphics	Previous Plot Characters
KRMRGN	A/N	Right Margin
KSIZEF**	A/N	Size Flag
KTBLSZ	A/N	Tab Table Size
KUPSCA*	Virtual Graphics	Relative Vector Up Scale Factor
KVERSZ	A/N	Character Vertical Size
KVERTT	A/N	Vertical Tab Table
TIMAGX	Virtual Graphics	Imaginary Beam X
TIMAGY	Virtual Graphics	Imaginary Beam Y
TMAXVX	Virtual Graphics	Virtual Window Maximum X
TMAXVY	Virtual Graphics	Virtual Window Maximum Y
TMINVX	Virtual Graphics	Virtual Window Minimum X
TMINVY	Virtual Graphics	Virtual Window Minimum Y
TRCOSF	Virtual Graphics	Relative Vector Cosine Factor
TREALX	Virtual Graphics	Real Beam X
TREALY	Virtual Graphics	Real Beam Y
TRSCAL	Virtual Graphics	Relative Vector Scale Factor
TRSINF	Virtual Graphics	Relative Vector Sine Factor

*Used only for implementation where only integer arithmetic is available.

**Used with implementations supporting the Tektronix 4002A Graphic Computer Terminal.

APPENDIX B

Other Terminal Control System Routines

B.1 GENERAL

The Terminal Control System consists of a set of highly modular routines in order that implementation and applicability would cover a number of terminals, systems, and users. A number of support routines not described in the main portion of this manual exist. These routines and a brief explanation of their function are described below.

B.2 BASIC I/O ROUTINES

B.2.1 Output Character — TOUTPT

Sets parity if necessary and outputs given character to terminal.

B.2.2 X,Y Conversion — XYCNVT

Screen X,Y Coordinates are translated to the minimum set of plot characters required for vector or point plot output. This routine performs the point plot simulation required for 4Ø1Ø implementations.

B.2.3 Forced I/O Delay — IOWAIT

Timesharing and remote temrinals will lose any output sent while a hard copy is being generated or the screen is being erased. The IOWAIT routine forces an appropriate delay in output to allow these events to occur without loss of information.

B.2.4 Output Dashed Line — TKDASH

Draws a dashed line as specified in KDASHT.

B.3 MODE CONTROL ROUTINES

B.3.1 Enter Vector Mode — VECMOD

Causes the terminal to enter the vector mode.

B.3.2 Enter Point Plot Mode — PNTMOD

Signals the X,Y Conversion routine to simulate point plotting for the 4Ø1Ø.

B.3.3 Enter Dash Mode — DSHMOD

Sets the dash type specification and enters dash mode.

B.3.4 Mode Check — MODCHK

Determines present system mode.

B.4 GRAPHIC TRANSFORM ROUTINES

B.4.1 Virtual Graphics to Screen Transformation — V2ST

Transforms a Virtual Display vector or point into output according to the current window definition. The General Error Flag is set whenever the given vector or point is outside the current window, and no output is generated. This routine maintains the Real Beam and Imaginary Beam positions in Virtual Graphics.

B.4.2 Clip — CLIPT

Clips Virtual Display vectors according to the current window definition. Returns start and end points of the visible segment of the vector. If vector does not pass through the window at all, the General Error Flag is raised.

B.4.3 Parallel Clip — PARCLT

Clips horizontal and vertical vectors on the Virtual Display according to the current window definition. Assumes vector passes through window and returns start and end points of the visible segment of the vector.

B.4.4 Point Clip — PCLIPT

Determines if given point is within the current window. Sets the General Error Flag if point is not within the current window.

B.4.5 Window Coordinate Transform — WINCOT

Scales and outputs a given Virtual Space vector or point according to the current window definition.

B.4.6 Reverse Window Coordinate Transform REVCOT

Transforms a given Screen Coordinate into a Virtual Coordinate according to the current window definition.

B.4.7 Graphic Level Check LVLCHT

Checks the current graphic level. If in Direct Level on entry, this routine resets the Real and Imaginary Beam and enters Virtual Graphics.

B.4.8 Relative to Absolute Conversion REL2AB

Scales and rotates relative vectors on the Virtual Display and converts them to absolute vectors.

APPENDIX C

Terminal Control System

Glossary

ABSOLUTE VECTOR

A directed line segment from a given start point to a given end point. In DIRECT GRAPHICS, the start point is defined by the beam position and the end point is an absolute SCREEN COORDINATE as specified by a DISPLAY COMMAND. In VIRTUAL GRAPHICS, the start point is defined by the IMAGINARY BEAM POSITION and the end point is an absolute VIRTUAL COORDINATE as specified by a DISPLAY COMMAND.

ALPHANUMERIC CURSOR

A rectangular non-stored movable marker which indicates the next position at which a character will be displayed.

ALPHANUMERIC MODE

The TERMINAL mode in which ASCII OUTPUT will be interpreted as characters to be displayed.

A/N

Abbreviation for "alphanumeric".

ASCII

American Standard Code for Information Interchange: A standard code consisting of 7-bit elements for information interchange among data processing communication systems. This code is usually broken up into two groups: a control set referred to as "CONTROL CHARACTERS" and a set which defines the character output when in ALPHANUMERIC MODE. Sometimes referred to as ANSCII or USASCII.

CHARACTER GENERATOR

A hardware or software device which draws the appropriate character when given a non-control ASCII character.

CLIPPING

The modification of VIRTUAL GRAPHICS Vectors so that the portion of these vectors which lie outside the WINDOW will not be displayed on the screen. The end points of such vectors are represented by the IMAGINARY BEAM POSITION so that sequential vectors defined by a series of end points will not be erroneously displayed.

CONTROL CHARACTER

The group of ASCII elements used to change the state of the TERMINAL or to perform functions other than the display of characters or the generation of vectors. Control characters are often used as data delimiters as well.

COORDINATE

An ordered pair (X,Y) of numbers which uniquely represent a point on either the screen or the VIRTUAL DISPLAY. The ordered pair of numbers used in the normal coordinate system (Cartesian Coordinates) represent the point according to its distance from the ORIGIN along the X-axis and Y-axis respectively.

CRT

Cathode Ray Tube. A device in which an electron beam emitted by a cathode strikes a phosphor screen to generate a visible image. The display surface of the Tektronix terminals is the viewing surface of a direct view bistable storage CRT.

CURSOR

A movable marker used as a reference.

DIRECT GRAPHICS

The set of DISPLAY COMMANDS which operate directly on the screen. Direct Graphics do not undergo CLIPPING and WINDOWING transformations.

DISPLAY COMMAND

A command which affects the display of data. Often an output command to the TERMINAL.

DRAW

The DISPLAY COMMAND which causes a visible vector to appear. In DIRECT GRAPHICS, the vector is from the current beam position to the given SCREEN COORDINATE. In VIRTUAL GRAPHICS, the vector is from the current IMAGINARY BEAM POSITION to the given VIRTUAL CO-ORDINATES. Note that in VIRTUAL GRAPHICS only the portion of the vector which passes through the window will appear.

ERASE

The procedure of clearing the TERMINAL screen.

GRAPHICS

The operations used to display data. Often refers only to the vector operations.

GRAPHIC CURSOR

A cross-hair CURSOR used to specify positional input.

GRAPHIC INPUT

Positional data consisting of an X- and Y-COORDINATE and specified by the location of the GRAPHIC CURSOR.

GRAPHIC LEVEL

The level (DIRECT or VIRTUAL) at which a display is being generated.

GRAPHIC TRANSFORM ROUTINES

The routines which transform VIRTUAL GRAPHICS into DIRECT GRAPHICS.

HARD COPY

A permanent copy of a display image. Also the operation which produces a permanent copy.

HARDWARE CHARACTER

A character displayed by the hardware CHARACTER GENERATOR internal to the TERMINAL.

HOME POSITION

The location on the screen in the upper-left hand corner at which the first character of a page is normally printed.

IMAGINARY BEAM POSITION

The VIRTUAL COORDINATE which corresponds to the position at which the STORAGE BEAM would be located if the entire VIRTUAL DISPLAY could be viewed.

INPUT

Data sent from the TERMINAL to the computer. Also data provided to a subroutine.

JOYSTICK

A device used to control the GRAPHIC CURSOR.

KEYBOARD

The portion of the TERMINAL which allows a user to enter A/N data into the computer.

LEFT MARGIN

The SCREEN X-COORDINATE which represents the starting position of a line of alphanumeric output.

MOVE

The DISPLAY COMMAND which causes an invisible vector to be generated.

NEW LINE

The operation which causes the ALPHANUMERIC CURSOR to go to the LEFT MARGIN and down one line.

NEW PAGE

The operation which ERASES the screen and moves the ALPHANUMERIC CURSOR to the HOME POSITION.

ORIGIN

The COORDINATE represented by (Ø,Ø). The origin of the screen is located at the lower left-hand corner. The VIRTUAL DISPLAY, by definition, has its origin at its center.

OUTPUT

Data sent from the computer to the TERMINAL. Also data generated by a subroutine.

PLOT CHARACTERS

A set of 1 to 4 non-control ASCII elements which represents a SCREEN COORDINATE to the terminal vector drawing hardware. Position data transmitted to and from the terminal must be in Plot Characters.

POINT PLOT

A DISPLAY COMMAND which causes an invisible vector to be generated and a point to be plotted at the end point of the vector. In BASIC GRAPHICS, no point will be plotted if the end point is outside the WINDOW.

POINT PLOT MODE

The TERMINAL mode which causes a set of PLOT CHARACTERS to be interpreted as a POINT PLOT vector.

RASTER UNIT

The distance between two adjacent points on the screen. The basic resolution element of the TERMINAL.

REAL BEAM POSITION

The point which represents the beam position transformed into VIRTUAL COORDINATES.

RELATIVE COSINE FACTOR

The cosine value used to rotate RELATIVE VECTORS.

RELATIVE ROTATION

The rotational transformation applied to RELATIVE VECTORS.

RELATIVE SCALE FACTOR

The value used to scale RELATIVE VECTORS.

RELATIVE SCALING

The linear transformation applied to RELATIVE VECTORS.

RELATIVE SINE FACTOR

The sine value used to rotate RELATIVE VECTORS.

RELATIVE VECTOR

A directed displacement used to construct an absolute vector according to current beam status. In DIRECT GRAPHICS, the vector constructed uses the beam position as the start point and the beam position plus displacement for an end point. In VIRTUAL GRAPHICS, the IMAGINARY BEAM POSITION provides the start point. The given displacement is then scaled and rotated before being added to the IMAGINARY BEAM POSITION to produce the end point.

RIGHT MARGIN

The SCREEN X-COORDINATE which represents the rightmost limit of alphanumeric output. Any attempt to output beyond the right margin using the A/N output routine will cause a NEW LINE to be generated.

screen

The portion of the TERMINAL on which output from the computer is displayed. The screen has 1Ø24 x 1Ø24 addressable points, although points with Y-COORDINATES greater than 78Ø will be off-screen. The ORIGIN for the screen is the extreme lower left point. Plotting on the screen without the use of the CLIPPING and WINDOW functions may be accomplished through the use of DIRECT GRAPHICS.

SCREEN COORDINATES

The set of points which constitutes the screen. These points form a discrete two-dimensional space and range from (Ø,Ø) to (1Ø23,1Ø23) inclusive. SCREEN COORDINATES MUST ALWAYS BE INTEGERS.

SCREEN WINDOW

The section of the screen into which the VIRTUAL WINDOW is transformed. No VIRTUAL GRAPHIC vectors may be displayed outside the Screen Window.

SOFTWARE

The programs and routines used to operate a computer. Also, the documentation, diagrams, and manuals for these routines, the computer, and associated peripheral devices.

SOFTWARE CHARACTERS

A character displayed by a software CHARACTER GENERATOR. Software Characters may be displayed in any size or rotation.

STORAGE BEAM

The electron beam which is directed by the output to draw characters and vectors on the TERMINAL screen.

STORAGE TUBE

A CRT which will maintain a display, written once, for an indefinite period until an erasure is made.

TERMINAL

A console which accepts data from or sends data to a computer. Used here to refer to the Tektronix 4Ø1Ø Computer Display Terminal which consists of a screen to display data and a keyboard to send data.

TERMINAL STATUS

The current state of the TERMINAL.

TERMINAL STATUS AREA

The set of common variables which represent the current TERMINAL STATUS.

TIMESHARING

The use of a computer to service a number of individuals in an effectively simultaneous fashion. Communication with a time-sharing computer is usually through an interactive TERMINAL.

VECTOR MODE

The TERMINAL mode which causes a set of PLOT CHARACTERS to be interpreted as a MOVE or DRAW vector. The first set of PLOT CHARACTERS output after entering Vector Mode will cause a MOVE to occur; sequential sets of PLOT CHARACTERS output without mode change will cause DRAW's to occur.

VIRTUAL COORDINATE

The set of points which constitute the VIRTUAL DISPLAY. These points form an effectively continuous two-dimensional space with a range equivalent to that of single precision floating point.

VIRTUAL CURSOR

The representation of the GRAPHIC CURSOR transformed into VIRTUAL COORDINATES according to the current WINDOW definition. The Virtual Cursor is not required to be within the current window.

VIRTUAL DISPLAY

An extensive imaginary display area independent of TERMINAL size restrictions. Displays may be constructed on the VIRTUAL DISPLAY using VIRTUAL GRAPHICS and may be inspected in part or totally through the definition of the WINDOW.

VIRTUAL GRAPHICS

The set of DISPLAY COMMANDS which operate on the VIRTUAL DISPLAY and perfrom CLIPPING and WINDOWING.

VIRTUAL WINDOW

The portion of the VIRTUAL DISPLAY which is displayed in the area defined by the SCREEN WINDOW. Only the portion of the VIRTUAL GRAPHICS vectors on the VIRTUAL DISPLAY which are contained within the Virtual Window will be displayed.

WINDOW

A transformation defined by the VIRTUAL WINDOW and the SCREEN WINDOW which allows a portion of the VIRTUAL DISPLAY to be viewed on a section of the screen. The transform itself consists of the elimination of vectors outside of the VIRTUAL WINDOW and the scaling of those inside to fit the SCREEN WINDOW.

WRAP-AROUND

The effect where a cursor or vector is moved to one side of the screen and reappears on the other side.

X-COORDINATE

The first (abscissa) value of a COORDINATE.

Y-COORDINATE

The second (ordinate) value of a COORDINATE.

4010
USASCII CODE FUNCTIONS

B7				ø	ø	ø	ø	1	1	1	1
B6				ø	ø	1	1	ø	ø	1	1
B5				ø	1	ø	1	ø	1	ø	1
BITS				CONTROL		HIGH ORDER X & Y		LOW ORDER X		LOW ORDER Y	
B4	B3	B2	B1								
ø	ø	ø	ø	NUL ø 0_8	DLE 16 20_8	SP 32 40_8	ø 48 60_8	@ 64 100_8	P 8ø 120_8	` 96 140_8	p 112 160_8
ø	ø	ø	1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
ø	ø	1	ø	STX 2	DC2 18	" 34	2 5ø	B 66	R 82	b 98	r 114
ø	ø	1	1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
ø	1	ø	ø	EOT 4	DC4 2ø	$ 36	4 52	D 68	T 84	d 1øø	t 116
ø	1	ø	1	ENQ 5 STATUS *	NAK 21	% 37	5 53	E 69	U 85	e 1ø1	u 117
ø	1	1	ø	ACK 6	SYN 22	& 38	6 54	F 7ø	V 86	f 1ø2	v 118
ø	1	1	1	BEL 7 BELL	ETB 23 * HARD COPY	' 39	7 55	G 71	W 87	g 1ø3	w 119
1	ø	ø	ø	BS 8 10_8 BACK SPACE	CAN 24 30_8	(4ø 50_8	8 56 70_8	H 72 110_8	X 88 130_8	h 1ø4 150_8	x 12ø 170_8
1	ø	ø	1	HT 9 SPACE	EM 25	) 41	9 57	I 73	Y 89	i 1ø5	y 121
1	ø	1	ø	LF 1ø LINE FEED	SUB 26 GRAPH IN *	* 42	: 58	J 74	Z 9ø	j 1ø6	z 122
1	ø	1	1	VT 11 REVERSE LINE FEED	ESC 27 33_8	+ 43	; 59	K 75	[91	k 1ø7	{ 123
1	1	ø	ø	FF 12 NEW PAGE *	FS 28	, 44	< 6ø	L 76	\ 92	l 1ø8	| 124
1	1	ø	1	CR 13 RETURN	GS 29 VECTOR	– 45	= 61	M 77	] 93	m 1ø9	125 ALT MODE
1	1	1	ø	SO 14	RS 3ø	. 46	> 62	N 78	^ 94	n 11ø	~ 126
1	1	1	1	SI 15	US 31 ALPHA	/ 47	? 63	O 79	_ 95	o 111	RUB OUT 127
						GRAPHIC INPUT				PRINT IN UPPER CASE	

* CHAR IS PRECEDED BY ESC CHAR TO PERFORM FUNCTION

TEKTRONIX®
excellence in information display

TEKTRONIX, INC. INFORMATION DISPLAY PRODUCTS
P.O. Box 500, Beaverton, Oregon 97005
(503) 644-0161

APPENDIX E

Index

READER'S COMMENT FORM

Your comments about this publication may be helpful to us.

If you wish to comment, please use the space provided below, giving specific page and paragraph reference.

Please do not use this form to ask technical questions about the equipment or to make requests for copies of publications. Instead, make such inquiries to your Tektronix Application Engineer.

Reply requested

Yes ☐

No ☐

Name ____________________

Job Title ____________________

Address ____________________

____________________ Zip ________

DOCUMENT NO. 062-1474-00

YOUR COMMENTS PLEASE

If you have any comments on this publication, please write them on the reverse side of this sheet.

Your comments will help us produce better publications for your use. Each reply will be carefully reviewed by the persons responsible for writing and publishing this material. All comments and suggestions become the property of Tektronix.

Note: **Please direct any requests for copies of publications, or for assistance in using your Tektronix equipment to your Tektronix Application Engineer.**

'd *fold*

FIRST CLASS
PERMIT NO. 61
BEAVERTON, OREGON

BUSINESS REPLY MAIL
No postage necessary if mailed in the United States

Postage will be paid by

TEKTRONIX, INC.
P.O. BOX 500
BEAVERTON, OREGON 97005
U.S.A.

ATTN: TEKTRONIX USER'S LIBRARY

'd *fold*

TEKTRONIX, INC.
P.O. BOX 500
BEAVERTON, OREGON 97005
U.S.A.

ATTN: TEKTRONIX USER'S LIBRARY

4/25/72

TERMINAL CONTROL SYSTEM
4002A ADDENDA

General

The Terminal Control System was originally released for the Tektronix 4010 Computer Display Terminal. All current documentation has been oriented towards that version. Additional routines have been written to support features of the 4002A not available on the 4010. These are INCPLT, ITALIC, ITALIR, DBLSIZ, NRMSIZ, ENSPM, CLRSP, ENCLM, and EDITSP. ~~In addition the following routines have been incorporated into the Terminal Control System but have not yet been included in the User's Manual: DASHA, DASHR, DSHABS, DSHREL, and CSIZE.~~

INCPLT

This routine hangles incremental plotting mode entry and handling of plot characters is entirely contained within this routine. The user specifies the direction, whether it is to be visible or invisible, and the number of times he wishes this plot character to be output.

CALLING SEQUENCE:

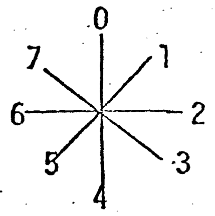
```
        CALL INCPLT (IONOFF,IDIR,NO)

where:  IONOFF = Ø; Beam off (invisible).
               = 1; Beam on (visible).

        IDIR   = Direction code (0-7; see below).

        NO     = Number of times plot chaaacter is to be
                 repeated.
```

Direction codes:



Each incremental plot character will move the beam one raster unit in the given direction.

ITALIC

This routine will output the proper control character to enter italic mode. This routine does NOT enter alphanumeric mode automatically.

CALLING SEQUENCE:

```
        CALL ITALIC
```

ITALIR

Resets to non-italic mode and enters alphanumeric mode. Double size mode is not affected by this routine.

CALLING SEQUENCE:

```
CALL ITALIR
```

DBLSIZ

This routine will output the proper control character to enter double size mode. This routine does NOT enter alphanuemric mode automatically.

CALLING SEQUENCE:

```
CALL DBLSIZ
```

NRMSIZ

Resets to normal size characters and enters alphanumeric mode. Italic mode is not affected by this routine.

CALLING SEQUENCE:

```
CALL NRMSIZ
```

NOTE: Italic and double size modes are set and reset only by the above routines. Entering graphic, point plot, or incremental plot modes will not affect these settings.

SCRATCHPAD

One of the major features of the 4002A terminal is the computer-addressable scratchpad. Some basic routines are included in the 4002A version of the Terminal Control System to assist in the use of the scratchpad. These are described below.

NOTE: It is firmly recommended that status be saved before using the scratchpad and restored after use.

ENSPM

This routine enters scratchpad mode. Future output will be directed to the scratchpad until this mode is left. Display of data output to the scratchpad will not occur until scratchpad mode is exitted (see ENLCM and EDITSP below).

CALLING SEQUENCE:

```
CALL ENSPM
```

CLRSP

The scratchpad is cleared and the scratchpad cursor is set to the beginning of the buffer.

CALLING SEQUENCE:

```
        CALL CLRSP
```

ENCLM

Scratchpad mode is exited, the output data is displayed, and local compose mode is entered. The user may now modify the output or clear and enter his own data. When he presses the SEND button while still in local compose mode, the entire buffer will be sent to the computer. After calling this routine, the program should be set for input as all output will be locked out until a reply from the user has been received.

CALLING SEQUENCE:

```
        CALL ENCLM
```

EDITSP

The scratchpad mode is exitted, the output data is displayed, (with a terminating question mark), and local edit mode is entered. The user may now enter his own data. If he remains in local edit mode and presses the SEND button, only the data entered after the computer output will be returned to the computer. After calling this routine, the program should be set for input as all output will be locked out until a reply has been received.

CALLING SEQUENCE:

```
        CALL EDITSP
```