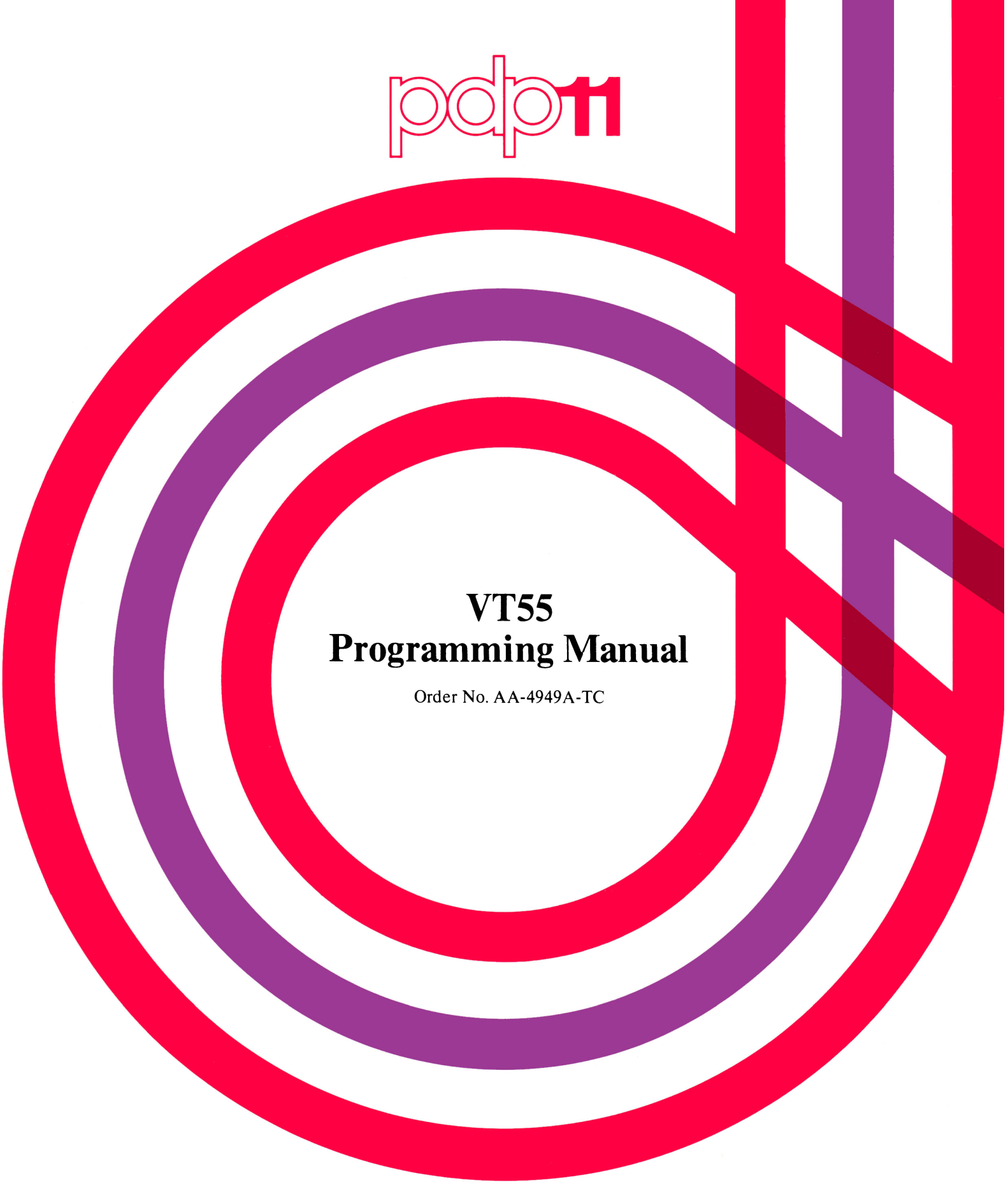

VT55
Programming Manual

Order No. AA-4949A-TC

February 1977

This document describes methods and available software for programming the VT55 terminal in the FORTRAN and BASIC-PLUS languages.

VT55
Programming Manual

Order No. AA-4949A-TC

SUPERSESSION/UPDATE INFORMATION:	For RT-11/FORTRAN users, this manual supersedes Chapter 1 of the FORTRAN/RT-11 Extensions Manual (DEC-11-LRTEA-C-D). For the other operating systems it contains new information.
OPERATING SYSTEM AND VERSION:	RT-11 V02C RSX-11M V03 RSX-11D V6.2 IAS V1.1 RSTS/E V6A
SOFTWARE VERSION:	FORTRAN IV V1C BASIC-PLUS V1 FORTRAN-IV PLUS V02

First Printing February 1977

CONTENTS

CONTENTS (Cont.)

PREFACE

This manual will introduce you, a new VT55 user, to the methods and resources that take fullest advantage of the VT55's capabilities. The VT55 can be used as the programming console of a very small system such as might be found in a laboratory or as one of a large number of terminals for an extensive multiuser computer system, as well as the range of intermediate applications. The programming methods outlined in this book are standard for all applications under the programming languages that are covered.

In order to understand this manual fully, you should be familiar with PDP-11 **FORTRAN** (for users of the RT-11, RSX-11M, RSX-11D, and IAS operating systems) or with **BASIC-PLUS** (for users of the RSTS/E operating system). If you are new to programming or need to refresh your memory, consult the list of associated documents in Section 1.6. All users should first read Chapter 1 and then the chapter describing the programming language to be employed.

Acquiring a high degree of skill in programming takes time, but the VT55 software can still be used by relatively unsophisticated programmers. You will find that, as with any sort of practical experience, VT55 programming is easiest to learn by simply working examples and progressing to more sophisticated applications as your expertise increases. This book is therefore oriented toward examples which illustrate VT55 features in simple terms and which provide good starting points for beginning graphic programmers. The chapter for each programming language will discuss the fundamental FORTRAN and BASIC-PLUS commands and concepts that a VT55 programmer needs to know to get started.

CHAPTER 1
GENERAL INFORMATION

1.1 GETTING USED TO GRAPHIC TERMS

There are many words which have a special meaning in graphic programming, and you should try to be familiar with them before getting started.

The VT55 terminal has more than one **terminal mode**. The difference between one terminal mode and another has to do with the way data are interpreted by the computer when it is either sending data to the VT55 or receiving data from it. There are three modes for the VT55 terminal:

Alphanumeric mode is the one which is in effect when you first turn on the power to a VT55. This mode implies that letters and numbers you type on the terminal will be received and interpreted by the computer according to a binary code called the American Standard Code for Information Interchange (ASCII). ASCII characters are the symbols used by the computer to perform calculations and most of the other functions that we normally associate with computers as strictly computational devices. In fact, the old-style "teletypewriter" terminal might never leave alphanumeric mode at all in the normal course of operation, so in this sense alphanumeric mode is the "normal" mode of the VT55.

The VT55, of course, can also display graphic figures on its screen such as function graphs, markers, and lines. These figures are put together from a special set of graphic instructions. In order for the computer to realize that a stream of data contains graphic instructions rather than the usual ASCII characters, the VT55 must enter a different mode called **graphic mode**. While in this mode the VT55 will send graphic instructions only, and the data received by the VT55 will be displayed on the screen in graphical form rather than as alphabetic or numeric characters. In other words, the graphic mode changes the way data are interpreted, not the data themselves.

The VT55 software discussed in this manual will switch the VT55 from one mode to the other automatically when you run programs. It is this feature which allows you to combine statements that display graphs and statements that display alphanumeric characters in the same program.

The VT55 also has a third, temporary mode called **escape mode**. Data that are received by a VT55 in escape mode are interpreted as "immediate" control instructions for the VT55. These instructions do such things as erasing portions of the screen, changing the position at which a line of text will appear, and, if your terminal has a "hard copy" unit built in, making a paper copy of the VT55 screen. When the terminal has completed the function commanded by such an **escape sequence**, it returns to its previous mode, and so the escape mode is only a temporary state. VT55 software covered in this book allows you to send escape sequences to the VT55 which force it to enter escape mode, perform a function such as the ones mentioned, and then to return to the former mode, all with a single subroutine call. The chapters for each programming language discuss escape sequences in more detail.

1.2 GRAPHIC FIGURES

The VT55 allows you to display the following types of graphic figures on its screen:

Graphs, which are plotted on the screen as a series of points or line segments;

Shaded Graphs, with the screen "filled in" underneath the graphed function;

Markers, which are short vertical line segments that can be displayed at interesting points on a graph; and

Horizontal or **vertical lines**, which span the entire width or height of the screen and can be used, for example, to display a grid or to lay out coordinate axes on the screen.

At any given time, you can simultaneously display on the screen up to two functions (two graphs, two shaded graphs, or one of each), up to 24 lines of 80 alphanumeric characters, 236 horizontal lines, 512 vertical lines, and 512 graph markers.

As you might expect, the VT55 has to be in **graphic mode** for any of these graphic features to operate; only in that mode will the graphic data be properly interpreted. Additionally, you must specifically **enable** each graphic figure you wish to use; the graphic figures are not available immediately upon entering graphic mode. Enabling a figure such as "Shaded Graph 1" does not mean that a shaded graph immediately appears on the screen but only that the VT55 is now capable of displaying a shaded graph; nor does the enabling of this figure mean that the next function to be plotted will be identified thereafter as Shaded Graph 1. Putting a shaded graph on the screen, to continue with the example, is a three-step process in both programming languages:

Step 1 invokes graphic mode and **enables** Shaded Graph 1 in a single subroutine call;

Step 2 is a second subroutine call to **select** Shaded Graph 1 as the next figure to be plotted;

Step 3 is a third subroutine call that tells the VT55 to **plot** Shaded Graph 1.

In the same subroutine call employed in Step 1, you can also selectively **disable** each graphic figure. As you will see in some of the examples in later chapters, repeated calls to the subroutines referred to here can make various graphic figures appear on the screen (two superimposed graphs, for example), and then some of the figures can be erased without affecting the others.

Each of the three steps must be done for every graphic figure you display on the VT55 screen, but the steps do not necessarily have to occur in the above order. For example, the subroutine referred to in Step 3, which plots graphs and shaded graphs, is actually telling the VT55 to store a value or series of values in its **graphic memory**. When you plot a single point a single value is stored, and a graph is plotted by storing a series of related points. There are two sections, or **registers**, in the graphic memory, one for Graph 0 and one for Graph 1. When, in Step 3, you plot Shaded Graph 1, you have loaded a series of values into Register 1, **whether or not Shaded Graph 1 is enabled**. If you enable Shaded Graph 1 later, these values will still be in Register 1, so the shaded graph will appear on the screen immediately. It follows that if you then disable Shaded Graph 1, the shaded graph will vanish from the screen, but the plotting values will still remain in Register 1. Both programming languages covered by this manual have a facility for clearing the graphic memory, that is, resetting all the values in both registers to zero.

1.3 CURSOR CONTROL

When you first turn on a VT55, a small flashing horizontal line appears in the upper left corner of the screen when the terminal has warmed up. This flashing figure is called the **cursor** and the upper left corner is the cursor's **home position**. When the terminal is in alphanumeric mode, the cursor shows you the position at which the next alphanumeric character will appear.

When the terminal is first switched on, it is in alphanumeric mode. Any system commands, text editing, and other normal programming dialog will appear on the VT55 screen and remain there. The text of these dialogs will stay on the screen even after you execute a graphic command; graphic commands switch to graphic mode but do not contain any instructions to automatically erase alphanumeric characters from the screen. VT55 graphic commands also are designed to automatically return the terminal to alphanumeric mode after they complete the designated graphic process. You must decide whether you want only your graphic program's output or the program's output plus system dialogs on the screen. If you want to draw a graph, for example, on a blank screen, move the cursor to the home position and then command the VT55 to erase all the text from the cursor to the bottom of the screen. Both the FORTRAN and the BASIC-PLUS VT55 software contain commands that simplify this operation.

1.4 MATHEMATICAL TERMS

This manual also uses a few terms borrowed from mathematics to describe plotting graphic figures on a coordinate system.

VT55 software under both languages allows you to display line segments on the screen of any length and at any position or angle (except vertical). These line segments (not to be confused with the vertical and horizontal lines described previously), are drawn on the screen from a particular starting point until they connect with a stated destination. The language of geometry calls this sort of "directed" line segment a **vector**, and this book uses the term to distinguish your variable line segments from the previously mentioned lines, which are always either vertical or horizontal in direction and have a fixed length (the height and width of the VT55 screen, respectively).

VT55 software allows you to construct a figure from a series of vectors and "label" the figure as either Graph 0 or Graph 1 (with a Select command like the command described in connection with plotting a shaded graph). Once again, you can use this feature to make figures appear and disappear, or by labeling portions of the figure differently, to selectively erase part of a drawing.

In both programming languages, the VT55 constructs graphs from one-dimensional **arrays**. As you will recall, arrays in a program are algebraic variables that are created by a statement such as (for FORTRAN):

```
      INTEGER DISPLAY(100)
```

If you began your VT55 program with this statement, you could then write additional statements to "fill" the 100 positions of DISPLAY with an ordered list of numbers, for example:

```
      DO 1 I=1,100
1     DISPLAY(I)=SIN(I*3.14159/50.)
```

In both languages, the VT55 software is designed to plot graphs by reading values from such arrays, with a single subroutine call. The same subroutine call can, with a minor change, plot a single isolated point without reference to an array; see the chapter appropriate to your programming language.

In both programming languages, you can change the starting coordinates of graphs and vectors. You can, for example, declare the starting coordinates to be the middle of the screen (x=236 and y=118). If so, the next graph to be displayed starts at x=236, or the next vector starts at the point (236,118).

When you draw a series of vectors, each vector will start at the destination point of the previous vector (or at the starting coordinates you have defined, in the case of the first vector to be drawn). Note also that vectors **do not have to be enabled**, as other figures do.

NOTE

The vector-drawing routines in both languages can be used to clear a single register in the graphic memory without disturbing the other register. This operation is done by setting the graph number to the number of the register you want to clear and then drawing a vector from the point (0,0) to the point (512,0). Doing so will load the register with zeroes.

1.5 TERMS FOR LARGE SYSTEMS

Each of the input/output paths in a RSTS/E or RSX-11 computer system is assigned a number called a **logical unit number**. This number, in effect, tells the computer to either send or receive data through a specific communication channel. If the data are being sent to some output device, it is obviously important that they end up at the right place, e.g., a line printer rather than a paper tape punch. The logical unit number tells the computer what the proper destination is for a certain stream of data, or in the case of input, where to look to find data on which to operate. For instance, the number 5 in the FORTRAN statement WRITE (5,1) is a logical unit number that is normally assigned by the FORTRAN compiler to the user's terminal. VT55 software allows you to override this "default" assignment and change the logical unit number of your VT55. This feature will be discussed in detail in the chapters that follow.

In the RSTS/E operating system, there can be a large number of terminals attached to the host computer. Each terminal, VT55s included, has an identification code called a **terminal number**. A particular installation will assign codes such as KB20:, for "Keyboard 20," to identify an individual device. The number 20 would be used in the Open VT55 command described in Section 3.1.1 of this manual.

In the RSX-11 and IAS systems, **event flags** are used to "synchronize" several programs (called "tasks") that are competing for use of the computer. If several tasks are ready to run, event flags keep track of, for example, input or output operations that are currently underway. When a high-priority task is waiting for a data transfer operation to be completed, the resources of the computer can be used by lower priority tasks until an event flag informs the system that the data transfer is finished. This type of synchronization can be valuable to VT55 programmers using large systems. If you fall into this category, consult your system's **Executive Reference Manual** for information about system directives used for controlling event flags. One form of the subroutine PLOT55, described in Section 2.2.1 of this manual, names a specific **event flag number** that can be used to synchronize your VT55 program.

1.6 FURTHER READING

You will find the following manuals helpful if you need further information on BASIC-PLUS, FORTRAN, operating systems, or the VT55 hardware.

Languages

BASIC-PLUS Language Manual
(DEC-11-ORBPB-A-D)

IAS/RSX-11 FORTRAN IV User's Guide
(DEC-11-LMFUA-C-D)

PDP-11 FORTRAN Language Reference Manual
(DEC-11-LFLRA-C-D)

RT-11/RSTS/E FORTRAN IV User's Guide
(DEC-11-LRRUA-A-D)

Operating Systems

RSX-11M Operator's Procedures Manual
(DEC-11-OMOGA-C-D)

RT-11 System Reference Manual
(DEC-11-ORUGA-C-D, DN1, DN2)

RSTS/E Programming Manual
(DEC-11-ORPMA-A-D)

RSX-11M Executive Reference Manual
(DEC-11-OMERA-C-D)

RSX-11D Executive Reference Manual
(DEC-11-OXERA-A-D)

IAS Executive Reference Manual, Volume 1
(DEC-11-OIE1A-A-D)

IAS User's Guide
(DEC-11-OIUGA-A-D)

VT55 Hardware

VT55-E,F DECgraphic Scope Users' Manual
(EK-VT55E-TM-001)

CHAPTER 2
FORTRAN PROGRAMMING

2.1 INTRODUCTION TO PLOT55

FORTRAN control of the VT55 is provided by a single subroutine called PLOT55. A call to this routine from your main program has one of the following two forms:

```
CALL PLOT55(ICMD,IX,IY,ITBL)

        or

CALL PLOT55(ICMD,IX,IY)
```

ICMD, IX, IY, and ITBL, the **arguments** of PLOT55, must all be **integers** or, in certain cases, integer **arrays**. The arguments have the following meanings in the three-argument form of PLOT55:

ICMD defines the **procedure** that will be carried out by a single call to PLOT55.

IX is, in most cases, the **x coordinate** of, for example, a point to be plotted on the VT55 screen. The specific use of this argument varies depending on the procedure being performed (i.e., depending on the value of ICMD). Its use will be explained for each procedure in the next section.

IY is, similarly, the **y coordinate** of a point. Its use will also be covered for each procedure.

The purpose of ITBL, which is a **16-element integer array**, is to provide a storage space that PLOT55 can use to store status information (e.g., which graphic figures are enabled) from one PLOT55 call to the next. For this reason, ITBL is called a **status table**. If you do not specify the fourth argument, PLOT55 will use an internally defined array for the same purpose. The use of this fourth argument in RT-11 FORTRAN programs usually makes no difference in the way your PLOT55 program functions (see Appendix B of this manual for some important exceptions). When you do specify ITBL, begin your program with the statements:

```
INTEGER ITBL(16)
DATA ITBL/16*0/
```

These statements a) define a 16-element array called ITBL, and b) fill the array with zeroes. Remember that the ITBL must be **initialized** (as with the DATA statement above) at the beginning of each new PLOT55 program.

NOTES

1. Do not use both three- and four-argument forms of PLOT55 in the same program, since doing so would effectively create two separate ITBLs.

2. If you use the four-argument form in an overlaid program, ITBL must be placed in the **root of the overlay** so that all PLOT55 calls can gain access to it.

3. RSX-11 and IAS users **must** use the four-argument version of PLOT55 at all times.

2.2 PLOT55 PROCEDURES

There are 14 procedures that you can control by changing the value of ICMD in CALL PLOT55 (ICMD, IX, IY [,ITBL]). ICMD can be any integer in the range 0 to 13, with the results described in this section.

NOTE

In many cases, you do not have to specify all three (or four) arguments in a PLOT55 call. However, three-argument calls must always contain **two commas**, and four-argument calls, **three commas**. PLOT55 programs will function unpredictably if any commas are left out.

2.2.1 Attaching and Detaching a Terminal (RSX-11 and IAS Only)

Format: CALL PLOT55(0,ILUN,IEFN,ITBL)

ILUN is the **logical unit number** and IEFN is the **event flag number**.

If ILUN is -1, the VT55 terminal is **detached**. If ILUN is a positive integer (1-12), the logical unit specified by ILUN is **attached** to the task being executed and will not accept input from another task.

IEFN, the event flag number (1-32), is important in controlling the scheduling of tasks in a multitask system. A detailed explanation of this number is beyond the scope of this book, but more information can be found on this number (and the logical unit number) in the **Executive Reference Manual** for your system.

The actual function of ICMD=0 is to store the logical unit number (ILUN) and event flag number (IEFN) in the status table (ITBL). Once this function is performed, all output from PLOT55 calls (graphic and alphanumeric instructions) is sent to the device associated with ILUN.

The ICMD=0 call **does not** associate ILUN with the VT55; you must do that job separately, using any of several methods available to users of IAS, RSX-11M, and RSX-11D. The system directive ASNLUN is one such method. ASNLUN can be written into your program as if it were a FORTRAN subroutine:

```
CALL ASNLUN (1,'TT',17)
```

In this example, 17 is the octal terminal number of the VT55, so that logical unit 1 is associated with physical device TT17:. Note that if this call were written into a program that was run from TT15:, for example,

the VT55 would only display information created by PLOT55. Unless you reassigned logical units 5 and 6 as well, these numbers would continue to refer to the terminal interface (TT15:) and the line printer, respectively. You could therefore keep the VT55 screen free of unwanted alphanumeric displays, such as the messages created by PAUSE statements.

In any case, it is best to use ASNLUN or some other method to assign ILUN to the VT55 before you call ICMD=0. See the documentation for your operating system for alternative methods of making this assignment.

Appendix B suggest a FORTRAN subroutine that performs the assignment and attachment in a single call.

NOTE

ICMD=0 is useful only under the RSX-11 and IAS systems. This command form can appear in RT-11 programs, too, but will not perform any operation.

2.2.2 Graphic Procedures

Select Graph Number (ICMD=1)

Format: CALL PLOT55 (1,IGRF,[,ITBL])

IGRF, always either 0 or 1, specifies the **graph number** of the next graph (regular or shaded) to be plotted. This number will be the label of the next regular or shaded graph to be plotted and will be the label of that figure as long as the figure is enabled. Note that this value of ICMD **does not** display or plot a graphic figure but only identifies the register that will contain the plotting values for the graph. Plotting the figure actually loads this register with plotting values (see ICMD=3), and enabling the figure (ICMD=2) allows it to appear on the screen as soon as it is plotted.

Enable or Disable Graphic Figures (ICMD=2)

Format: CALL PLOT55 (2,IENAB,IDISAB[,ITBL])

This call is the "fundamental" form of PLOT55, because graphic figures cannot be displayed until they have been enabled. Therefore, you will, in most cases, want this form to be one of the first calls to PLOT55 in your program. Of course, this form can also appear later in the program if you want to enable some new figure or to disable a figure that is already on the screen.

IENAB and IDISAB are lists of figures to be enabled or disabled, respectively. You can use the following values in these two lists:

1 = Enter (leave) graphic mode.

2 = Enable (disable) Graph 0.

4 = Enable (disable) Graph 1.

8 = Enable (disable) Shaded Graph 0.

16 = Enable (disable) Shaded Graph 1.

32 = Enable (disable) horizontal lines.

64 = Enable (disable) vertical lines.

128 = Enable (disable) markers for Graph (or Shaded Graph) 0.

256 = Enable (disable) markers for Graph (or Shaded Graph) 1.

512 = Disable all graphic figures and clear the graphic memory (IENAB only).

To enable or disable more than one figure in a single PLOT55 call, you can enter lists of numbers, separated by plus signs, for IENAB or IDISAB. For example, instead of putting the number 1 in the IENAB position (to enter graphic mode), you could put 1+2+4+512, which would first clear the graphic memory and then enter graphic mode and enable Graph 0 and Graph 1.

NOTES

1. If you enter the same number for both IENAB and IDISAB, the graphic figure for that number will be **enabled**.

2. You can also enter the sum of several numbers rather than a list, e.g., the number 24 (instead of 8+16) to represent both shaded graphs. However, your program will be more easily understood by other users if you avoid this practice.

3. As a matter of good practice you should always include the 512 argument in the IENAB position **the first time** you call PLOT55 with ICMD=2, e.g.,

   CALL PLOT55(2,1+512+2,)

   This three-argument example would put the terminal in graphic mode, enable Graph 1, and also would clear all previous graphic information from the status table and memory. If you were running several graphic programs in sequence, the 512 would erase the graphic output of the last program from the screen before displaying a new figure. (Notice also that in the sample call above, the IDISAB position is empty, indicating that no graphic figures are being disabled, but that the "trailing" comma is still included.)

4. It is usually not desirable to enable different figures with the same label (for instance, both Graph 0 and Shaded Graph 0).

Plot Graph (ICMD=3)

After you have enabled a graph and selected its label (ICMD=2 and ICMD=1, respectively), you can use ICMD=3 to plot the figure.

Format: CALL PLOT55(3,IX,IY[,ITBL])

or

CALL PLOT55(3,IX,IARRAY[,ITBL])

When the first format is used, ICMD=3 will display a **single point** on the screen at the coordinates given by IX (0-511) and IY (0-235).

When you use the second format, IX should be a negative integer and IARRAY an integer array that you have previously filled with a series of values. In this form, ICMD=3 will plot a **series of points** on the screen. The number of points plotted will be the absolute value of IX; the same number of y coordinates will be selected from IARRAY, starting with the first element of the array; and the points will be separated by one unit in the x direction.

For example,

CALL PLOT55(3,100,100)

will plot a single point on the screen at a position 100 units from the bottom of the screen (y=100) and 100 points from the left edge (x=100).

The other form,

CALL PLOT55(3,-100,ISINE)

would plot a sine function on the screen (assuming that ISINE has been identified in a DIMENSION statement and filled with sine values **before** this call to PLOT55). The sine function would be made of 100 points (the absolute value of IX=-100), and each point would be separated from neighboring points by one x unit on the screen.

NOTE

It is generally necessary to define a starting x position (ICMD=7) before plotting regular or shaded graphs. It is **not possible** to assign two or more y values to the same x value in a single graph; therefore, vertical lines cannot be plotted with ICMD=3.

Example 1 demonstrates the use of ICMD=1, ICMD=2, and ICMD=3 to display a series of functions on the VT55 screen.

Example 1

```
      INTEGER ARYSIN(500),ARYCOS(500),ARYEXP(500)
      DATA PI/3.14159/
C
C     ENABLE GRAPHIC MODE AND GRAPHS 0 AND 1
C
      CALL PLOT55(2,1+2+4+512,)
C
C     FILL UP THE COSINE, SINE AND EXPONENTIAL ARRAYS
C
      DO 3 I=1,500
      EXPARG=I/50.
      THETA=PI*EXPARG
      ARYSIN(I)=SIN(THETA)*100.+100.
      ARYCOS(I)=COS(THETA)*100.+100.
3     ARYEXP(I)=EXP(EXPARG)
C
C     DEFINE STARTING COORDINATES AT (0,0), SELECT GRAPH 0, AND
C     PLOT THE SINE FUNCTION
C
      CALL PLOT55(7,0,0)
      CALL PLOT55(1,0,)
      CALL PLOT55(3,-500,ARYSIN)
      READ(5,2) KR
2     FORMAT(I2)
C
C     PLOT THE COSINE FUNCTION AS GRAPH 1
C
      CALL PLOT55(1,1,)
      CALL PLOT55(3,-500,ARYCOS)
      READ(5,2) KR
C
C     ENABLE THE SHADED GRAPHS AND DISABLE THE GRAPHS
C
      CALL PLOT55(2,8+16,2+4)
C
C     PLOT THE EXPONENTIAL AS SHADED GRAPH 0 AND THE
C     COSINE AS SHADED GRAPH 1.
C
      CALL PLOT55(1,0,)
      CALL PLOT55(3,-500,ARYEXP)
      CALL PLOT55(1,1,)
      CALL PLOT55(3,-500,ARYCOS)
      READ(5,2) KR
C
C     DISABLE SHADED GRAPH 1 (THE COSINE FUNCTION)
C
      CALL PLOT55(2,,16)
      READ(5,2) KR
C
C     REENABLE GRAPH 0 AND DISABLE SHADED GRAPH 0, CHANGING
C     THE EXPONENTIAL TO A REGULAR GRAPH.
C
      CALL PLOT55(2,2,8)
      READ(5,2) KR
      END
```

Notes on Example 1:

1. The subroutine PLOT55(7,0,0) defines a starting x position (x=0) for all the graphs to be displayed by the program. Unless you want to change the starting position of graphic figures, you do not have to repeat this call.

2. The statements READ(5,2) KR are used in this example to make the program pause between each graphic display. Each time this statement is reached, the computer will attempt to read a character from the keyboard (logical unit 5); when you type a carriage return on the keyboard the program will continue.

   FORTRAN provides a better method, the PAUSE statement, for doing the same thing, but using it will make the letters "PAUSE--" appear on the screen each time the program reaches a PAUSE statement. A later example will show you how to remove this sort of undesired text from the screen. (IAS/RSX-11 users: see Section 2.2.3 for a further discussion of the PAUSE statement.)

3. At this point, you should try running this example program. It will function as follows:

   a) First, Graph 0, a sine function, will be traced out on the screen of the VT55 by 100 dots.

   b) Type a carriage return, and Graph 1, a cosine function, will also appear. Notice that the sine and cosine functions have the same starting x position.

   c) Type another carriage return, and both graphs will disappear and be replaced by Shaded Graphs 0 and 1, which represent the exponential and cosine functions, respectively.

   d) Type another carriage return and Shaded Graph 1 will be disabled, making the cosine function vanish.

   e) Typing a final carriage return will disable the shaded graphs and reenable Graph 0. This time the Graph 0 label is given to the exponential function, so the shaded graph of the exponential will be replaced with a regular graph of the same function.

Plot Horizontal Line (ICMD=4)

Format: CALL PLOT55(4,IDISP,IY[,ITBL])

This form is used to either erase or display horizontal lines that span the entire width of the screen.

IDISP is set to 0 to erase a line or to 1 to display a line.

IY gives the y coordinate (0 to 235) of the line to be erased or displayed.

Plot Vertical Line (ICMD=5)

Format: CALL PLOT55(5,IX,IDISP[,ITBL])

This form is very similar to the previous ICMD=4, but controls vertical rather than horizontal lines. Note that the position of IDISP is different for this format than for ICMD=4.

IX gives the x coordinate (0 to 511) of a vertical line to be erased or displayed.

IDISP is set to 0 to erase a line or to 1 to display a line.

Example 2 demonstrates the use of both line types.

Example 2

```
C
C       THIS SAMPLE PROGRAM SHOWS HOW TO DRAW AND ERASE
C       HORIZONTAL AND VERTICAL LINES
C
C       BEGIN BY RESETTING THE GRAPHIC MEMORY (THE 512
C       ARGUMENT) AND ENABLING BOTH LINE TYPES
C
        CALL PLOT55(2,1+32+64+512,)
C
C       NEXT DRAW 51 VERTICAL LINES
C
        DO 1 I=2,502,10
1       CALL PLOT55(5,I,1)
        READ(5,2) KR
2       FORMAT(A2)
C
C       NOW DRAW IN 24 HORIZONTAL LINES TO FORM A GRID
C
        DO 3 I=2,232,10
3       CALL PLOT55(4,1,I)
        READ(5,2) KR
C
C       ERASE THE VERTICAL LINES
C
        DO 4 I=2,502,10
4       CALL PLOT55(5,I,0)
        END
```

Notes on Example 2:

As the example shows, ICMD=4 and ICMD=5 can each be used to either erase or display horizontal and vertical lines. Again, the READ statement is used to insert temporary pauses, preventing unwanted text from appearing on the screen.

Plot Markers (ICMD=6)

Format: CALL PLOT55(6,IX,IDISP[,ITBL])

With this command, you can place markers (short vertical line segments) at any points on a regular or shaded graph. The markers for Graph 0 and for Graph 1 **must be enabled separately** (see ICMD=2).

When you call PLOT55 with ICMD=6, markers will be displayed on the graph that has currently been selected (see ICMD=1). The following example illustrates a common error:

```
.
.
.
.
CALL PLOT55(2,1+2+4+256,)  (Enable both graphs and markers for Graph 1)
CALL PLOT55(1,0,)          (Select Graph 0)
.
.
CALL PLOT55(6,50,1)        (Plot marker at x=50)
.
.
```

This example **will not work**, because the second call selected Graph 0, and only the markers for Graph 1 have been enabled. If you plan to use markers in a graphic display, avoid this problem by enabling **both** sets of markers when you use both Graph 0 and Graph 1 (or Shaded Graph 0 and Shaded Graph 1). You will still have to be careful, of course, to ensure that a marker is placed on the correct graph.

IX, an integer from 0 to 511, gives the x coordinate at which the marker will appear. If a graph exists at that x position, the marker will appear on the graph; if there is no graph at that x position on the screen, the marker will appear on the y=0 line at the bottom of the screen.

IDISP will cause the marker to either be erased (IDISP=0) or displayed (IDISP=1).

Example 3 demonstrates the proper use of markers.

Example 3

```
C       THIS PROGRAM DEMONSTRATES THE USE OF MARKERS
C       ON GRAPHS AND SHADED GRAPHS
C
        INTEGER ARYSIN(500)
        DATA PI/3.14159/
C
C       RESET THE GRAPHIC MEMORY, ENTER GRAPHIC MODE,
C       AND ENABLE GRAPH 0 AND MARKERS FOR GRAPH 0
C
        CALL PLOT55(2,512+1+2+128,)
C
C       FILL THE ARRAY WITH A SINE FUNCTION
C
        DO 1 I=1,500
1       ARYSIN(I)=SIN(PI*I/50.)*100.+100.
C       DEFINE STARTING X COORDINATE, SELECT GRAPH 0, AND
C       PLOT THE SINE FUNCTION.
C
        CALL PLOT55(7,0,0)
        CALL PLOT55(1,0,)
        CALL PLOT55(3,-500,ARYSIN)
        READ(5,2) KR
2       FORMAT(I2)
C
C       PUT A MARKER ON GRAPH 0 EVERY 25 X UNITS
C
        DO 3 I=25,500,25
3       CALL PLOT55(6,I,1)
        READ(5,2) KR
C
C       NOW DISABLE GRAPH 0 AND ENABLE SHADED GRAPH 0;
C       PLOTTING THE SINE AS A SHADED GRAPH
C
        CALL PLOT55(2,8,2)
        READ(5,2) KR
C
C       REMOVE THE MARKERS, BUT LEAVE THE SHADED GRAPH
C
        DO 4 I=25,500,25
4       CALL PLOT55(6,I,0)
        READ (5,2) KR
        END
```

Define Starting Coordinates (ICMD=7)

Format: CALL PLOT55(7,IX,IY[,ITBL])

This call defines the starting x position for regular or shaded graphs (ICMD=3) and the starting x and y positions for vectors (ICMD=8). PLOT55 will read the x and y coordinates that you supply as IX and IY and will store them in its status table.

IX should be an integer between 0 and 511.

IY, also an integer, must be between 0 and 235.

Note that when you draw **vectors** (ICMD=8), the starting coordinates change each time you draw a vector. If, for instance, you want several vectors to start from the same point, you have to redefine the starting coordinates (with ICMD=7) before drawing each vector.

When you draw **graphs**, the starting y coordinate will be supplied in your ICMD=3 call, as the first element of IARRAY.

NOTE

The starting coordinates you define do not affect the placement of **single points** by the ICMD=3 call. For example,

```
CALL PLOT55(3,100,100)
```

will **always** plot a point 100 units from the left of the screen and 100 units from the bottom, regardless of the starting coordinates you have defined.

Draw Vector (ICMD=8)

Format: CALL PLOT55(8,IX,IY[,ITBL])

After you have defined starting coordinates with ICMD=7, you can use this call to draw line segments on the screen. The first vector you draw will start at the previously defined starting coordinates and end at the point (IX,IY).

IX, an integer from 0 to 511, is the x coordinate of the destination of the vector.

IY, which can be from 0 to 235, is the y coordinate of the destination.

For example, the series

```
CALL PLOT55(7,100,100)
CALL PLOT55(8,150,150)
```

will draw a vector from (100,100) to (150,150).

NOTE

After drawing a vector to (IX,IY), ICMD=8 **resets the starting coordinates** to (IX,IY). If you draw a series of vectors, each one (except the first) will start where the previous vector ended. It is **not possible** to draw an absolutely vertical vector, because no two y values can apply to the same x coordinate for the same graph number.

Example 4 demonstrates the use of Define Starting Coordinates and Draw Vector calls. Notice that ICMD=7 is only used once, so that the vectors will be connected to each other.

Example 4

```
C       THIS PROGRAM SHOWS HOW TO DEFINE STARTING COORDINATES
C       AND HOW TO PLOT VECTORS.  IT WILL DRAW A DIAMOND IN THE
C       MIDDLE OF THE VT55 SCREEN.
C
C       FIRST CLEAR THE SCREEN OF GRAPHIC FIGURES.
        CALL PLOT55(2,512,)
C       THEN ENTER GRAPHIC MODE AND ENABLE GRAPHS 0 AND 1.
        CALL PLOT55(2,1+2+4,)
C       SELECT GRAPH 0 FOR THE TOP OF THE DIAMOND AND DEFINE
C       STARTING COORDINATES OF (100,100)
C
        CALL PLOT55(1,0,)
        CALL PLOT55(7,100,100)
C
C       NOW DRAW THE TOP TWO LINES USING ICMD=8 (DRAW VECTOR).
C       NOTICE THAT THE SECOND LINE WILL BE DRAWN STARTING
C       WHERE THE FIRST LINE ENDED.
C
        CALL PLOT55(8,150,150)
        CALL PLOT55(8,200,100)
C
C       NOW SELECT GRAPH 1 FOR THE BOTTOM TWO LINES AND DRAW
C       THEM.
C
        CALL PLOT55(1,1,)
        CALL PLOT55(8,150,50)
        CALL PLOT55(8,100,100)
        READ (5,1) KR
1       FORMAT(I2)
C
C       TO DEMONSTRATE THE FACT THAT THERE ARE TWO SEPARATE
C       FIGURES ON THE SCREEN, ERASE THE BOTTOM OF THE
C       DIAMOND BY DISABLING GRAPH 1.
C
        CALL PLOT55(2,,4)
        END
```

2.2.3 Alphanumeric Procedures

Position Cursor (ICMD=9)

Format: CALL PLOT55(9,IX,IY[,ITBL])

This simple command positions the cursor at the point (IX,IY).

IX can be from 0 to 79; and
IY can be from 0 to 23

You can use this command to decide where the next display of alphanumeric characters will appear on the screen. IX will give the column number (with 0 being the leftmost column) and IY will be the line number (line 0 is the top line on the screen).

Erase from Cursor to End of Screen (ICMD=10)

Format: CALL PLOT55(10,,[,ITBL])

NOTE
The "x" and "y" positions are always empty, but you must still include the two commas (three if ITBL is specified).

This command erases all alphanumeric characters from the present cursor position to the end of the screen. The position of the cursor **is not** changed by ICMD=10.

Graphic figures are not affected by ICMD=10.

The subroutine CLEAR, shown in Example 5, uses ICMD=9 and ICMD=10 to clear the entire screen.

CLEAR functions by first returning the cursor to the home position:

```
CALL PLOT55(9,0,0)
```

and then erasing from this position to the end of the screen:

```
CALL PLOT55(10,,)
```

Erase from Cursor to End of Line (ICMD=11)

Format: CALL PLOT55(11,,[,ITBL])

This call performs the same sort of procedure as ICMD=10; but it only erases text from the present cursor position to the end of the line. As with ICMD=10, the erasure does not change the cursor's position, nor does it change the status of graphic figures.

Display Text (ICMD=12)

Formats: CALL PLOT55(12,,'ANY TEXT STRING'[,ITBL])

or

CALL PLOT55(12,ICHARS,IALPHA[,ITBL])

Both formats will display strings of alphanumeric characters, with the first character appearing at the cursor position.

The first format will display any text string that is inserted between the single quotation marks; in the example shown here, the letters ANY TEXT STRING would appear at the cursor position. The second format can function in two modes, both of which display characters from the "alphanumeric array," IALPHA. These modes are selected by the argument ICHARS.

If ICHARS is omitted or is 0, PLOT55 will display characters that are listed in the third argument until it finds a NUL character (000). The example with a text string in single quotes shows this mode in operation, because FORTRAN automatically puts a NUL at the end of text strings in single quotes.

If ICHARS is some positive integer, it will determine the number of characters from IALPHA that will be displayed.

Example 5 shows how to display alphanumeric text with the use of ICMD=10, ICMD=11, and ICMD=12. Notice that a NUL character (000) is loaded into the last word of IALPHA to provide an end mark.

Example 5

```
C
C     THIS PROGRAM DEMONSTRATES ALL OF THE "ALPHANUMERIC"
C     FORMS OF PLOT55.
C
C     ICMD = 9 AND ICMD = 10 ARE USED IN A SUBROUTINE CALLED
C     "CLEAR," WHICH ERASES EXTRA TEXT FROM THE SCREEN.
C
C     AS A PRELIMINARY STEP TO ICMD = 12, YOU MUST DEFINE AN
C     "ALPHANUMERIC ARRAY" THAT CAN BE USED TO STORE TEXT:
C
      LOGICAL*1 IALPHA(81)
      DATA IALPHA/80*' ',0/
C
C     THE LAST WORD IN IALPHA MUST BE 000, WHICH WILL BE
C     INTERPRETED AS A "NULL" CHARACTER AND WILL TELL ICMD = 12
C     TO STOP DISPLAYING CHARACTERS FROM IALPHA.
C
C     CLEAR THE STATUS TABLE AND THE SCREEN:
C
      CALL CLEAR
      CALL PLOT55(2,512,)
```

```
C
C       PUT THE CURSOR AT THE BEGINNING OF LINE 12 AND
C       DISPLAY A MESSAGE:
C
        CALL PLOT55(9,0,12)
        CALL PLOT55(12,,'WHEN YOU TYPE A CARRIAGE RETURN, THE
     X  SYMBOLS WILL VANISH:    [1234567890]<>*+@#%&')
        READ(5,1) KR
1       FORMAT(I2)
C
C       THESE TWO PLOT55 CALLS WILL ERASE ALL THE CHARACTERS
C       AFTER "VANISH:".
C
        CALL PLOT55(9,60,12)
        CALL PLOT55(11,,)
        PAUSE
C
C       TYPE ANOTHER CARRIAGE RETURN TO ERASE THE SCREEN AGAIN.
C
        CALL CLEAR
C
C       NOW WE'LL USE THE ALPHANUMERIC ARRAY, FILLING IT WITH
C       CHARACTERS YOU TYPE IN FROM THE KEYBOARD.
C
        CALL PLOT55(9,0,12)
        CALL PLOT55(12,,'TYPE UP TO ONE LINE OF CHARACTERS AND
     X  A CARRIAGE RETURN')
        CALL PLOT55(9,0,14)
        READ(5,2) (IALPHA(J),J=1,80)
2       FORMAT(80A1)
        CALL CLEAR
C
C       STATEMENT 3 WILL DISPLAY ALL THE CHARACTERS YOU ENTERED,
C       BECAUSE PLOT55 WILL KEEP GOING UNTIL IT HITS THE
C       "NULL" CHARACTER IN IALPHA(81).
C
C       YOU WILL DECIDE HOW MANY CHARACTERS WILL BE DISPLAYED
C       BY STATEMENT 5 BY TYPING AN INTEGER (1-80) IN RESPONSE
C       TO THE PROGRAM'S REQUEST.
C
        CALL PLOT55(9,0,12)
3       CALL PLOT55(12,,IALPHA)
        CALL PLOT55(9,0,14)
        CALL PLOT55(12,,'TYPE A NUMBER OF CHARACTERS AND A <CR>:')
        READ(5,4) ICHARS
4       FORMAT(I2)
        CALL PLOT55(9,0,14)
        CALL PLOT55(11,,)
5       CALL PLOT55(12,ICHARS,IALPHA)
        PAUSE
        CALL CLEAR
        END
```

```
SUBROUTINE CLEAR
CALL PLOT55(9,0,0)
CALL PLOT55(10,,)
RETURN
END
```

NOTE

In IAS operation, the PAUSE statement is ignored. The READ statement should be used instead. In RSX-11 systems the PAUSE statement will suspend the task, which must be restarted with the MCR command RESUME:

>RES taskname <CR>

Merely typing a carriage return will not resume the task in RSX-11. When you run an RSX-11 task containing PAUSE statements, give it a specific task name, for instance:

RUN EXAM1/TASK=EXAM <CR>

where EXAM1.TSK is the name of the file containing the task and EXAM is the name that would be used in the RES command.

2.2.4 Sending Escape Sequences

Format: CALL PLOT55(13,ICHAR,[,ITBL])

This command reads a number that you insert in place of ICHAR, puts an <ESC> character (033 octal) in front of it, and sends those two characters to the VT55. This sort of character pair is called an "escape sequence," and it tells the VT55 to do the following:

1. Enter **escape mode**.
2. Perform a procedure specified by your choice of ICHAR.
3. Return to the previous mode (i.e., graphic or alphanumeric) when the procedure is finished.

ICHAR is a **decimal** integer. The values you can enter for ICHAR are listed below along with the procedures they invoke:

ICHAR value	Procedure
65	Move cursor up one line
67	Move cursor right one position.
72	Move cursor to home position.
74	Erase from cursor to end of screen.

75	Erase from cursor to end of line.
91	Hold screen.
92	Release screen.
93	Copy from top of screen to cursor.
94	Start automatic copy.
95	Stop automatic copy.

NOTE

If you send an escape sequence that does not constitute an executable command, the VT55 will enter escape mode and then automatically return to the previous mode, **without performing any operation**. The same holds true if you use an ICHAR in the range 93-95 and your VT55 has no hard copy unit.

Several of the PLOT55 forms, particularly the ones which are alphanumeric functions, work by sending escape sequences to the VT55. For example, the forms

```
CALL PLOT55(11,,) and
CALL PLOT55(13,75,)
```

do exactly the same thing: erase from the cursor to the end of the line.

By using escape sequences, you can simplify some operations considerably. Suppose you wanted to move the cursor along the 20th line of text one character at a time. You could say

```
      DO 1 I=1,80
      J=I-1
1     CALL PLOT55(9,J,19)
```

But the same procedure is accomplished by saying

```
      CALL PLOT55(9,0,19)
      DO 1 I=1,80
1     CALL PLOT55(13,67,)
```

Using ICMD=13 eliminates the computation of J=I-1 (which would be done 80 times), but more importantly, ICMD=13 is a "simpler" command on the assembly language level and will be executed more quickly. Each of the 80 times ICMD=9 is used, the ICMD=9 macro routine is called and then it calls seven additional routines to home the cursor, read the value of IX (even though, in this case, IX is the same all 80 times), and repeatedly send escape sequences to move the cursor 20 lines down from the home position.

Therefore, the execution speed of your program will nearly always benefit if you use ICMD=13 as much as possible.

The Hold Screen and Release Screen commands disable and reenable (respectively) the VT55's automatic scrolling feature, that is, the feature that moves text up on the screen when you type more than 24 lines.

After you have displayed a graph on the screen, you may want to temporarily prevent the computer from printing out the usual STOP message (or whatever your operating system normally displays at the end of a FORTRAN program). Suppressing such messages is useful for appearance' sake and nearly always desirable when you make a hard copy of the screen contents. The following sequence at the end of your program will accomplish this task:

```
900     CALL PLOT55(9,79,23)
901     CALL PLOT55(13,91,)
902     PAUSE
903     CALL PLOT55(13,92,)

904     END
```

Statement 900 moves the cursor to the last position on the screen. Then Line 901 effectively 'freezes' the screen so that the computer cannot display characters; the characters 'PAUSE--' are transmitted back to the VT55, but because the scrolling is disabled they are not immediately displayed. The computer can still read the VT55 keyboard, however, so that when you type a carriage return (or a RESUME command in RSX-11), Statement 903 will be executed, releasing the screen. If your VT55 has a built-in hard copier, you can copy the contents of the screen (both graphic and alphanumeric) by pressing the orange COPY key on the keyboard **before** typing the carriage return to release the screen. In any case, after releasing the screen, you will have to push the orange SCROLL key a few times to scroll up the messages that were transmitted while the terminal was in Hold Screen mode.

The escape sequence value 93 is the programmed equivalent of manually pressing the COPY key on the keyboard; it will create a copy of everything on the screen from the top up to the cursor position, including graphic displays. Remember to position the cursor at the end of the screen--

```
CALL PLOT55(9,79,23)
```

before copying the screen.

The escape sequence values 94 and 95 start and stop the automatic copy feature of terminals with a built-in hard copier. **Automatic copying should only be used with alphanumeric text**; graphic displays will be distorted if copied in this mode. When the VT55 receives a command to start automatic copying, it copies the contents of the screen beginning at the top and continuing up to but not including the line containing the cursor. Then one additional line of text will be copied each time you type a LINE FEED on the keyboard. Automatic copying is especially useful for creating permanent listings of programs and subroutines that you write. This automatic mode can be started and stopped manually by holding down the SHIFT key while you press the COPY key.

Example 6 contains a program that allows you to examine the effects of various escape sequences.

For more information on escape sequences, see the **VT55-E,F DECgraphic Scope User's Manual** (EK-VT55E-TM-001).

Example 6

```
C       THIS PROGRAM DEMONSTRATES THE USE OF ICMD = 13,
C       WHICH SENDS "ESCAPE SEQUENCES" TO THE VT55.
C
C       THE PROGRAM WILL READ A TWO-DIGIT INTEGER THAT
C       YOU TYPE ON THE VT55 KEYBOARD .  THE INTEGER
C       WILL BECOME AN ARGUMENT IN A PLOT55 CALL AND WILL
C       PERFORM SOME OPERATION ON THE VT55 SCREEN.
C
C       THE INTEGERS YOU CAN ENTER, AND THE OPERATIONS THAT
C       WILL RESULT, ARE AS FOLLOWS:
C
C               65 = MOVE CURSOR UP ONE LINE.
C               67 = MOVE THE CURSOR ONE POSITION TO THE RIGHT.
C               72 = MOVE CURSOR TO THE HOME POSITION (0,0)
C               69 = ERASE FROM CURSOR TO END OF SCREEN.
C               91 = HOLD SCREEN.
C               92 = RELEASE SCREEN.
C
C       AND, IF YOUR VT55 HAS A HARD COPY UNIT BUILT IN,
C
C               93 = COPY FROM TOP OF SCREEN TO CURSOR.
C               94 = START AUTOMATIC PRINT.
C               95 = STOP AUTOMATIC PRINT.
C
C       TO STOP THE PROGRAM, TYPE 99.
C
1       READ(5,2) ICHAR
2       FORMAT(I2)
        IF (ICHAR.EQ.99) GO TO 3
        CALL PLOT55(13,ICHAR,0)
        GO TO 1
3       END
```

CHAPTER 3
BASIC-PLUS PROGRAMMING

Four BASIC-PLUS functions exist to aid VT55 graphic programming. The names and purposes of the functions are as follows:

FNV5, for the display of graphs, markers, vectors, vertical or horizontal lines, and shaded graphs;

FNV6, for the display of alphanumeric characters on the screen;

FNV7, for the initialization of the screen; and

FNV8, for the display of special "step" histograms.

Each of the functions is used in the form:

D= FNVx(arguments)

where x is the function number and arguments are the integers or character strings you supply in a particular call. After a line containing one of these functions is executed, D will equal zero if the function executed correctly. If some error is detected in the function, control is immediately returned to the calling program in most cases, and D is set to -1. D is therefore a kind of "condition code" that can be examined after a function call to ensure that no errors have occurred.

The BASIC-PLUS definitions for these four functions are contained in the file VT55.BAS.

The function definitions in VT55.BAS must be added to every BASIC-PLUS program you write for the VT55. The following dialog shows how to add the definitions to your program (the underlined statements are the ones that you would type).

READY

.
.
.

(The numbered lines for your program.)

.
.
.

APPEND VT55 <CR> (The function definitions are added to your program.)

```
READY

RUN or RUNNH <CR>        (The program will be executed.)
```

Notice that there is no line number for the APPEND statement.

This procedure will add the lines in VT55.BAS to the end of your program, so that it will now contain both the function definitions and an END statement. Be sure to observe the following rules:

1. Put an END statement with line number 32767 at the end of each program.

2. The file VT55.BAS contains BASIC-PLUS statements with line numbers from 20000 to 29999. Avoid using these line numbers in your program.

3. Similarly, the function definitions use the following "internal" variables to store graphic information, so they should not be used as variables in your main program:

   C9%: Channel number for terminal.
   C8$: Output buffer for alphanumeric text.
   C7%: Current graph number.
   O7%: X value of origin.
   Q7%: Y value of origin.
   O6%: Enable/disable memory.
   Q6%: Enable/disable memory.

4. The following variables will be destroyed by the functions in VT55.BAS. Do not use them in your program if they should be unchanged after a call to one of the functions:

   C9$: Output string for FNV6.
   O9%, Q9%, O8%, Q8%, O9, Q9, O8, Q8, O7, A2, A7: Temporary "scratch" variables.

In addition to VT55.BAS, a demonstration program, DEMO55.BAS, has been supplied in your BASIC-PLUS kit. After you have read and understood this manual, you should run DEMO55 (preferably after listing the program on a line printer) so that you can observe the functions in operation:

```
READY
OLD DEMO55 <CR>
READY
APPEND VT55 <CR>
READY
RUNNH <CR>
```

NOTE

See the introduction to Appendix C for instructions on loading VT55.BAS from the distribution medium to your system disk.

3.1 FNV5, THE GENERAL GRAPHIC FUNCTION

Basic Call Format: D = FNV5(A1%,A2%,A3%)

A1% can be any integer from 0% to 13% **except 12%**. The number you insert in place of A1% will select the procedure to be performed by the FNV5 call. The allowable values for A2% and A3% are explained later for each procedure. An error return (with D = -1) will result if A1% is less than 0%.

3.1.1 Opening and Closing a VT55 for Output

Call Format: D = FNV5(0%,A2%,A3%)

A2%, an integer from 0% to 63%, is the **terminal number** of the VT55 terminal. If the VT55 terminal is designated "KB1:", for example, then A2% would be 1%. Any time you make A2% greater than 0% in this manner, the VT55 will be **opened** for output from your BASIC-PLUS program.

Note that A2% is the number of the VT55, which is not necessarily the same as the number of the terminal used to command RSTS/E (that is, the terminal with which you logged on to the system). In fact, if convenient you should log on to a terminal that is not a VT55, with the VT55 screen in plain view. If you log on to KB1: and the VT55 is KB2:, A2% should be 2% when you run the program. Now the VT55 will only display output from the special functions FNV5, FNV6, FNV7, and FNV8, which are discussed in this chapter. All other program output, such as the question marks created by INPUT statements, will appear on your "command terminal," KB1:. Therefore the VT55 screen, and any reproduction you make of the screen, will be free of unwanted messages.

A3% is the **logical unit number** opened for the terminal and can be any integer from 1% to 12%. As long as the VT55 is open, the logical unit number (or "channel") specified by A3% will be used for your program's output.

To **close** the terminal just call FNV5 again, but this time make A2% a negative integer (any negative value will work). You should always enter a 0% for A3% in a closing call, for example:

D = FNV5(0%,-5%,0%)

This call would close the terminal, closing whatever logical unit number was opened by the previous command opening the terminal. Closing your VT55 terminal also erases any graphic figure on the screen, disables the graphic features you have chosen, and resets the starting coordinates (see A1% = 7) to (0,0). Therefore, the best practice in VT55 graphic programming is to make the opening call (A2% > 0%) **the first FNV5 call** and the closing procedure the last call in your program.

NOTE

Any alphanumeric text that was displayed by your program will remain on the screen whether or not you close the VT55. If you enter a logical unit number outside the range 1%-12% or a terminal number outside the range 0%-63%, an error return will result (D = -1).

3.1.2 Graphic Display with FNV5

There are eight forms of FNV5 that control graphic displays, the forms being selected by your choice of A1%. Pay particular attention to the order in which graphic commands are given, as illustrated in the examples that accompany the command descriptions.

Select Graph Number (A1% = 1%)

Call Format: D = FNV5(1%,A2%,0%)

A2% should either be 0% or 1% and will select the **graph number** for the next regular or shaded graph to be plotted. This number will serve as the "label" for the figure as long as the figure is enabled. **A3% is always 0%** in this call. Note that selecting this graph number is only a labeling procedure (that is, it selects a register to contain the graph plotting values) and **does not display a graph**. Displaying a graphic figure is a three-step process:

1. Enable the figure (A1% = 2%).
2. Select graph number (A1% = 1%).
3. Plot the figure (A1% = 3%).

Enable or Disable Graphic Figures (A1% = 2%)

Call Format: D = FNV5(2%,A2%,A3%)

This command is the "fundamental" form of FNV5, because all the graphic figures **must be enabled** before they will appear on the VT55 screen. These figures are all disabled at first, and once enabled they will remain so until specifically disabled.

A2% and A3% tell which figures will be enabled or disabled, respectively. The following numbers can be used for either A2% or A3%:

1%	=	Enter (or leave) graphic mode.
2%	=	Enable (disable) Graph 0.
4%	=	Enable (disable) Graph 1.
8%	=	Enable (disable) Shaded Graph 0.
16%	=	Enable (disable) Shaded Graph 1.
32%	=	Enable (disable) horizontal lines.
64%	=	Enable (disable) vertical lines.
128%	=	Enable (disable) markers on Graph or Histogram 0.
256%	=	Enable (disable) markers on Graph or Histogram 1.
512%	=	Disable all figures and clear the graphic memory (A2% only).

When you are enabling and disabling figures, you can either put single integers in the call for A2% and A3%, or you can substitute expressions that combine any or all of the figures. For example,

D = FNV5(2%,1%+2%+512%,8%)

would do the following procedures:

1. Erase any graphic figures left on the screen by previous programs (512%);
2. Enter graphic mode and enable Graph 0 (1%+2%); and
3. Disable Shaded Graph 0 (A3% = 8%).

NOTES

1. If you enter the same number for A2% and A3%, the corresponding graphic figure will be **enabled**.

2. Instead of an expression like 1%+2%, you can enter the **sum** of several numbers to enable or disable figures, such as 24% to refer to both Shaded Graph 0 and Shaded Graph 1. However, the expression form will be more readily understood by other persons reading your program.

3. It is a good practice to always include the number 512% in the A2% position the first time you enable graphic figures. Doing so will eliminate graphic displays left on the screen by a previous program.

4. It is usually undesirable to enable a graph and a shaded graph with the same graph number (e.g., Graph 0 and Shaded Graph 0).

5. An error return will occur (with D set to -1) if the sums of the arguments are outside the range 0%-1023% (for A2%) or 0%-511% (for A3%).

Plot Graph (A1% = 3%)

Call Formats: D = FNV5(3%,A2%,A3%)

or

D = FNV5(3%,A2%,0%)

The first call format shown is used to plot a **single point** at the coordinates given by A2% (0% to 511%) and A3% (0% to 235%). For example,

```
850     FOR I% = 1% TO 511%
860     D = FNV5(3%,I%,235%*RND)
870     NEXT I%
```

will plot points on the screen with x values from 1 to 511 and with y varying randomly between 0 and 235.

The second format plots points with y values from an **array** called **V5**. You must dimension V5 and fill it with y values before trying to plot the graphic figure. **V5 must be dimensioned to at least -A2%-1%**. Otherwise, the FNV5 call will cause a fatal subscript error. Remember that BASIC-PLUS arrays begin with a zero subscript, so that the statement DIM V5(511) creates a 512-element array. For example,

```
340     DIMENSION V5(511)
350     FOR I% = 0% TO 511%
360     V5(I%) = 118%+100%*SIN(I%/50.)
370     NEXT I%
380     D = FNV5(1%,0%,0%)
390     D = FNV5(2%,1%+2%,0%)
400     D = FNV5(3%,-512%,0%)
```

Lines 340-370 set up V5 as a 512-element array and then fill up V5 with sine values.

Line 380 selects 0 as the **label** of the next graphic figure. Line 390 enters graphic mode and enables Graph 0.

Line 400 now plots the sine function in the form of a graph (Graph 0). Notice that A2% is a **negative integer**. When A2% is negative, FNV5 plots a number of y values from V5 equal to the absolute value of A2%. In the example shown here, 512 values (the entire contents of V5) will be used, so the sine function will extend over the entire width of the screen.

NOTE

An error return (D = -1) will occur if A2% is ever less than -512% (for the second format). D is also set to -1 if you plot a point with coordinates outside the screen area (A2% < 0%, A3% < 0%, A2% > 511%, or A3% > 235%).

Example 1

```
        !THIS SAMPLE BASIC-PLUS PROGRAM USES THE
        !FOUR FUNCTION STATEMENTS NEEDED TO PLOT
        !GRAPHS AND SHADED GRAPHS.
        !
        !BEGIN BY CREATING THREE ARRAYS; A1% WILL BE
        !THE SINE FUNCTION, A2% THE COSINE FUNCTION
        !AND A3% THE EXPONENTIAL FUNCTION.
        !
100     DIM A1%(499),A2%(499),A3%(499),V5(499)
110     FOR I% = 0% TO 499%
120             F1=I%/50.
130             T1=P1*F1
140             A1%(I%) = SIN(T1)*100+100
150             A2%(I%) = COS(T1)*100+100
160             A3%(I%)=EXP(F1)
170     NEXT I%
180     INPUT "WHAT IS THE VT55'S KEYBOARD NUMBER";I%
190     D = FNV5(0%,I%,1%)      !"CHANNEL 1" ATTACHED.
        !
        !DEFINE STARTING POSITION, SELECT GRAPH 0,
        !AND PLOT THE SINE FUNCTION.
        !
200     D = FNV5(7%,0%,0%)      !START AT X=0.
210     D = FNV5(2%,1%+2%,0%)
        !ENABLE GRAPHIC MODE AND
        !GRAPH 0.
220     D = FNV5(1%,0%,0%)      !SELECT GRAPH 0.
230     V5(I%)=A1%(I%) FOR I%=0% TO 499%
        !V5 IS NOW THE SINE FUNCTION.
240     D = FNV5(3%,-500%,0%)   !PLOT THE SINE GRAPH.
250     INPUT "CONTINUE";C$
        !PAUSE AND WAIT FOR CARRIAGE RETURN; THEN
        !PUT THE COSINE FUNCTION ON THE SCREEN, TOO.
        !
```

```
260       D = FNV5(2%,4%,0%)        !ENABLE GRAPH 1.
270       D = FNV5(1%,1%,0%)        !SELECT GRAPH 1.
280       V5(I%)=A2%(I%) FOR I%=0% TO 499%
          !CHANGE V5 TO COSINE.
290       D = FNV5(3%,-500%,0%)     !PLOT THE COSINE GRAPH.
300       INPUT "CONTINUE";C$
          !
          !ERASE BOTH GRAPHS (BY DISABLING THEM) AND
          !PREPARE TO DISPLAY SHADED GRAPHS INSTEAD.
          !
310       D = FNV5(2%,8%+16%,2%+4%)
          !
          !SHADED GRAPH 0 WILL BE THE EXPONENTIAL AND
          !SHADED GRAPH 1 WILL BE THE COSINE.
          !
320       D = FNV5(1%,0%,0%)        !SELECT SHADED GRAPH 0.
330       V5(I%)=A3%(I%) FOR I%=0% TO 499%
          !V5 = EXPONENTIAL.
340       D = FNV5(3%,-500%,0%)     !PLOT THE EXPONENTIAL.
350       D = FNV5(1%,1%,0%)        !SELECT SHADED GRAPH 1.
360       V5(I%)=A2%(I%) FOR I%=0% TO 499%
          !NOW V5 = COSINE.
370       D = FNV5(3%,-500%,0%)     !PLOT THE COSINE.
380       INPUT "CONTINUE";C$
390       D = FNV5(2%,0%,16%)       !ERASE THE COSINE SHADED GRAPH.
400       INPUT "CONTINUE";C$
410       D = FNV5(2%,2%,8%)        !CHANGE EXPONENTIAL TO A GRAPH.
          !ENABLED GRAPH 0 AND
          !DISABLED SHADED GRAPH 0.
420       D = FNV5(0%,-1%,0%)       !CLOSE CHANNEL 1.
32767     END
```

Plot Horizontal Line (A1% = 4%)

Call Format: D = FNV5(4%,A2%,A3%)

A3% gives the **y coordinate** (0%-235%) of the horizontal line.

A2% is either 0% or 1%. A2% = 0% will **erase** the line at the y position given by A3%. A2% = 1% will **display** the line.

Remember that the horizontal line feature must have been enabled previously by a call such as

D = FNV5(2%,1%+32%,0%)

NOTE

An error return will occur if you try to display a line off the screen (A3% < 0% or A3% > 235%) or if A2% is any number other than 0% or 1%.

Plot Vertical Line (A1% = 5%)

Call Format: D = FNV5(5%,A2%,A3%)

This command is very similar to A1% = 4%, but it uses a different combination of arguments to plot vertical lines instead of horizontal lines.

In this call A2% is the **x coordinate** at which a vertical line will either be displayed or erased. A2% can take values between 0% and 511%.

A3% = 0% will **erase** the line and A3% = 1% will **display** the line.

Again, any values entered for A2% or A3% that are not within the legal range will cause an error return.

Example 2:

```
        !THIS PROGRAM DRAWS A "GRID" ON THE SCREEN WITH
        !THE VERTICAL AND HORIZONTAL LINE FIGURES.
10      INPUT "WHAT IS THE VT55'S TERMINAL NUMBER";T%
20      D = FNV5(0%,T%,1%)                  !OPEN AS CHANNEL 1.
30      D = FNV5(2%,1%+512%+32%+64%,0%)
        !GRAPHIC MEMORY CLEARED; GRAPHIC MODE AND
        !BOTH LINE TYPES ENABLED.
40      FOR I%=2% TO 502% STEP 10%
50              D = FNV5(5%,I%,1%)
60      NEXT I%
        !51 VERTICAL LINES ARE ON THE SCREEN; PAUSE.
70      INPUT "CONTINUE";C$
        !TYPE A CARRIAGE RETURN TO CONTINUE.
80      FOR I%=2% TO 232% STEP 10%
90              D = FNV5(4%,1%,I%)
100     NEXT I%
        !24 HORIZONTAL LINES ADDED.
110     INPUT "CONTINUE";C$
120     FOR I%=2% TO 502% STEP 10%
130             D = FNV5(5%,I%,0%)
140     NEXT I%
        !VERTICAL LINES ERASED.
150     D = FNV5(0%,-1%,0%)                 !CLOSE CHANNEL 1.
32767   END
```

Plot Markers (A1% = 6%)

Call Format: D = FNV5(6%,A2%,A3%)

This command will display or erase markers on regular and shaded graphs plotted with A1% = 3%. The markers are short vertical line segments that will appear on the graph at the x position you specify in the call.

A2% is the x position of the marker (0%-511%).

A3% = 0% will erase the marker, and A3% = 1% will display it.

NOTE

This command will not work properly unless A2% and A3% are within the stated ranges.

Example 3:

```
        !THIS PROGRAM DEMONSTRATES THE USE OF MARKERS ON
        !GRAPHS AND HISTOGRAMS.
100     DIM V5(499)
110     INPUT "WHAT IS THE VT55'S TERMINAL NUMBER";T%
120     D = FNV5(0%,T%,1%)        !OPEN FOR OUTPUT AS CHANNEL 1.
130     V5(I%)=SIN(PI*I%/50.)*100.+100. FOR I%=0% TO 499%
        !V5 NOW CONTAINS SINE VALUES.
140     D = FNV5(2%,1%+512%+2%+128%,0%)
        !RESET THE GRAPHIC MEMORY AND
        !ENABLE GRAPHIC MODE, GRAPH 0, AND MARKERS FOR GRAPH 0,
150     D = FNV5(7%,0%,0%)                !START AT X=0.
160     D = FNV5(1%,0%,0%)                !SELECT GRAPH 0.
170     D = FNV5(3%,-500%,0%)             !PLOT THE SINE GRAPH.
180     INPUT "CONTINUE";C$
190     FOR I% = 50% TO 500% STEP 50%
200             D = FNV5(6%,I%,1%)
210     NEXT I%
        !PUT A MARKER ON THE GRAPH EVERY 50 X UNITS.
220     INPUT "CONTINUE";C$               !PAUSE.
230     D = FNV5(2%,8%,2%)                !CHANGE SINE TO SHADED GRAPH.
        !ENABLE SHADED GRAPH 0,
        !DISABLE GRAPH 0.
240     INPUT "CONTINUE";C$               !PAUSE.
250     D = FNV5(2%,0%,128%)
        !REMOVE THE MARKERS, BUT LEAVE THE SHADED GRAPH.
260     D = FNV5(0%,-1%,0%)               !CLOSE CHANNEL 1.
32767   END
```

Define Starting Coordinates (A1% = 7%)

Call Format: FNV5(7%,A2%,A3%)

This command defines new starting coordinates for vectors, graphs, and shaded graphs. For regular and shaded graphs, the Define Starting Coordinates command supplies a **starting x coordinate** for the figure, without affecting the starting y coordinate. This command has no effect on single points plotted with A1% = 3%).

NOTE

An error return will occur unless A2% is in the range 0%-511% and A3% is in the range 0%-235%.

Draw Vector (A1% = 8%)

Call Format: D = FNV5(8%,A2%,A3%)

This command draws line segments called vectors, which extend from the starting coordinates to a point on the screen given by A2% and A3%.

A2% is the x coordinate of the vector's **destination point** and should be a number from 0% to 511%. A3% is the **destination y coordinate**, with allowable values between 0% and 235%. (An error return will occur if A2% or A3% is outside its legal range.)

If you have previously defined starting coordinates (A1% = 7%) when you draw a vector, the vector will begin at those coordinates. Remember that the "initialization" function FNV7 (described later) resets the starting coordinates to (0,0).

The Draw Vector command itself also **resets the starting coordinates to those of the destination point** supplied in your call. This feature allows you to "chain" vectors together to form geometric figures or other straight-line drawings. If you want to draw a series of vectors that originate at the same point, you can simply interweave Define Starting Coordinates commands with your Draw Vector commands.

Vectors are not graphic figures, strictly speaking, because they **do not have to be enabled**. However, you can (and should) label a series of vectors that make a design as Graph 0 or Graph 1, and Graphs 0 and 1 do, of course, have to be enabled. Remember that you **cannot** draw a vector that is perfectly vertical.

Example 4:

```
        !THIS SAMPLE PROGRAM DEMONSTRATES THE USE
        !OF VECTORS, WHICH CAN BE USED TO DRAW
        !GEOMETRIC FIGURES (A DIAMOND IN THIS CASE)
        !AND CAN BE LABELED LIKE GRAPHS.
        !
100     INPUT "WHAT IS THE VT55'S TERMINAL NUMBER";T%
110     D = FNV5(0%,T%,1%)              !OPEN "CHANNEL 1."
120     D = FNV5(2%,1%+512%+2%+4%,0%)
        !CLEAR THE SCREEN OF GRAPHIC DISPLAYS, ENTER
        !GRAPHIC MODE, AND ENABLE BOTH GRAPHS.
130     D = FNV5(1%,0%,0%)              !SELECT GRAPH 0.
140     D = FNV5(7%,100%,100%)
        !(100,100) IS THE STARTING POINT OF THE FIRST VECTOR.
150     D = FNV5(8%,150%,150%)          !FIRST VECTOR.
160     D = FNV5(8%,200%,200%)          !SECOND VECTOR.
170     D = FNV5(1%,1%,0%)              !SELECT GRAPH 1.
        !THE BOTTOM OF THE DIAMOND WILL BE GRAPH 1.
180     D = FNV5(8%,150%,50%)           !THIRD VECTOR.
190     D = FNV5(8%,200%,100%)          !FOURTH (FINAL) VECTOR.
        !THE DIAMOND IS COMPLETE.
200     INPUT "CONTINUE";C$
        !TYPE A CARRIAGE RETURN TO CONTINUE.
210     D = FNV5(2%,0%,4%)      !DISABLE GRAPH 1.
        !TO DEMONSTRATE THE FACT THAT TWO SEPARATE
        !"GRAPHS" ARE BEING DISPLAYED, GRAPH
        !1 (THE BOTTOM OF THE DIAMOND) WAS ERASED.
220     D = FNV5(0%,-1%,0%)     !CLOSE CHANNEL 1.
32767   END
```

3.1.3 Alphanumeric Display with FNV5

The actual output of alphanumeric characters to the VT55 screen is done with a separate function, FNV6. However, FNV5 contains three command forms that are useful for alphanumeric display programming.

Position Cursor (A1% = 9%)

Call Format: D = FNV5(9%,A2%,A3%)

This command positions the cursor at a new point whose coordinates are given by A2% and A3%.

A2% is the number of character positions from the left edge of the screen at which the cursor will appear. A2% can therefore take values from 0% to 79%.

A3% is the number of lines from the top of the screen at which the cursor will appear. Since there are 24 lines per screen, A3% can be any integer from 0% to 23%.

Note that an error return (D = -1) will occur if you enter values for A2% or A3% that exceed the legal ranges.

Note also that moving the cursor with this command has **no effect** on graphic or alphanumeric displays on the screen.

Erase from Cursor to End of Screen (A1% = 10%)

Call Format: D = FNV5(10%,0%,0%)

This command erases all alphanumeric text from the screen, starting at the cursor position and continuing to the bottom of the screen.

This command does not erase graphic figures from the screen, nor does it change the cursor position.

Notice that the call format is completely standard for this command; the arguments are always 10%, 0%, and 0%.

Erase from Cursor to End of Line (A1% = 11%)

Call Format: D = FNV5(11%,0%,0%)

This command is very similar to A1% = 10%, the difference being that the erasure stops at the end of the cursor's line instead of continuing to the end of the screen.

Note again that the format is constant.

As with the previous command, the status of graphic figures and the cursor position are unaffected by the erasure.

3.1.4 Producing Hard Copy with FNV5

FNV5 can be used to send special instructions called "escape sequences" to the VT55 terminal. An escape sequence consists of the ASCII code 155, which is interpreted as an <ESC> character in RSTS/E, and a value you supply as A2% in the following call:

D = FNV5(13%,A2%,0%)

As is discussed in the **VT55-E,F DECgraphic Scope User's Manual**, escape sequences can be used to perform a variety of functions, including "cursor control" and screen erasure. Of course, these two types of control are provided by forms of the FNV5 function, so that there is no advantage to using escape sequences.

The most helpful application of escape sequences is in producing a paper copy of the screen display. This procedure is only valid for VT55 terminals with a **built-in hard copy unit**; the same program can be run on a terminal with no hard copier, but the hard copy commands (93%-95%) will be ignored.

Following is a list of values for A2% that will perform hard copy operations:

A2% Value	Procedure
91%	Hold screen.
92%	Release screen.
93%	Copy from top of screen to cursor.
94%	Start automatic copying.
95%	Stop automatic copying.

The following example would copy all graphic and alphanumeric displays from the screen:

```
100     D  = FNV5(9%,79%,23%)          !Cursor to bottom right.
110     D  = FNV5(13%,91%,0%)          !Hold screen.
120     D  = FNV5(13%,93%,0%)          !Copy entire screen.
130     D  = FNV5(13%,92%,0%)          !Release screen.
```

As shown in Line 100, you have to first move the cursor to the bottom right corner of the screen, because the copy operation will stop when the cursor is reached.

Line 110 will "**hold" the screen**, temporarily preventing the display of further text or graphic figures. While the VT55 is in this mode, the computer will still read characters typed on the keyboard, but will not "echo" them, so they will not appear on the screen.

Line 120 now copies everything on the screen, from the topmost line to the cursor.

Now that the copy operation is finished, the screen is released by Line 130, which cancels the previous Hold Screen command. After the screen is released, you will have to press the SCROLL key a few times before you can continue. The SCROLL key will display any messages that were received while the screen was still being held, for instance, the READY message that tells you the system is ready for new commands.

If you have a screen display consisting entirely of **alphanumeric text**, you can use the **automatic copy** commands to produce "line printer" type listings. When the VT55 is in automatic copy mode, it will first copy everything up to but **not including** the line containing the cursor; then it will copy one additional line each time you push the <LINE FEED> key on the keyboard. The following example shows a programmed automatic copy operation:

```
100     D  = FNV5(9%,0%,0%)            !Cursor to home position.
110     D  = FNV5(13%,91%,0%)          !Hold screen.
120     D  = FNV5(13%,93%,0%)          !Start automatic copying.
130     INPUT A$
        !A LINE FEED WILL COPY ONE MORE
        !LINE. THE LETTER 'S' WILL STOP
        !AUTOMATIC COPYING.
```

```
140     IF A$='S' GOTO 160              !Type S to stop.
150     GOTO 130                        !Read the next character.
160     D  = FNV5(13%,95%,0%)           !Stop automatic copy.
```

This example will read each line feed that you enter to copy lines. It will keep the terminal in automatic copy mode until you enter the character S.

NOTES

1. Automatic copying should only be used with alphanumeric text. The hard copy of a graphic figure will be distorted if produced in line-by-line fashion.

2. The screen will temporarily go blank while a copy operation is in progress.

3. The copy operation can be **manually** started any time the program pauses by simply pressing the orange COPY key on the VT55 keyboard. Similarly, pressing the COPY key while holding down the SHIFT key will both start and stop automatic copying.

3.2 FNV6, THE TEXT OUTPUT FUNCTION

Call Format: D = FNV6(N%,C1$)

FNV6 displays a **character string** on the VT55 screen. The string will appear with the first character at the cursor position, so that FNV6 is often used in combination with the "alphanumeric" forms of FNV5, which control the cursor position and the erasure of characters (see Section 3.1.3).

As the format shows, there are **only two arguments** for FNV6, both of which are required at all times.

C1$, a string variable, stores the text that you wish to display on the screen. C1$ can be either a regular or subscripted variable, depending on the type of display you are designing. A regular string variable can accept a string of 80 characters, which is the maximum width of a line of characters on the screen. Therefore, you can use **regular variables** when the string will be displayed **horizontally**. **Vertical displays** require either **subscripted variables**, so that one element can be displayed per line, or alternatively, the use of the BASIC-PLUS function **MID**, which selects "substrings" from a regular character string variable and thereby allows you to address individual characters or groups of characters in a string. Example 5 shows how MID can be used to put a vertical label on a graph.

3.3 FNV7, THE INITIALIZATION FUNCTION

Call Format: D = FNV7

This simple function (notice that there are **no arguments**) resets the screen to an "initial state," that is:

1. The **starting coordinates are reset to the point (0,0)**, the lower left corner of the screen;

2. All **graphic figures are erased** from the screen, leaving only alphanumeric text;

3. The internal variables O6% and Q6%, which store information about the graphic figures, are **reset to 0, disabling all graphic figures**; and

4. The terminal is returned to alphanumeric mode.

All four of these initializations can, of course, be performed by FNV5 calls, but FNV7 is a more direct and simple method. Therefore, it is good practice to call FNV7 immediately after opening the terminal (see Example 5).

NOTE

FNV7 should not be used in the middle of a program unless you want to totally eliminate the effects of previous FNV5 calls.

Example 5:

```
        !THIS SAMPLE PROGRAM USES SEVERAL FORMS OF FNV5
        !AS WELL AS THE ALPHANUMERIC FUNCTION FNV6 AND
        !THE INITIALIZATION FUNCTION FNV7.
90      DIM V5(499)
100     INPUT "WHAT IS THE VT55'S TERMINAL NUMBER";T%
110     D = FNV5(0%,T%,1%)      !OPEN "CHANNEL 1."
120     D = FNV7
        !FNV7 CLEARS THE SCREEN OF GRAPHIC FIGURES AND
        !RESETS THE STARTING COORDINATES TO (0,0).
130     GOSUB 1000
        !THE SUBROUTINE AT LINE 1000 CLEARS ANY STRAY TEXT
        !FROM THE SCREEN.
140     D = FNV5(9%,0%,12%)
        !
        !PUT THE CURSOR AT THE BEGINNING OF LINE 12 AND
        !DISPLAY A MESSAGE.
        !
150     C9$='TYPE A CARRIAGE RETURN; SYMBOLS WILL VANISH: 1234567890<>+'
160     D = FNV6(58%,C9$)       !DISPLAY THE CHARACTERS.
170     INPUT "CONTINUE";C$     !PAUSE FOR CARRIAGE RETURN.
180     D = FNV5(9%,44%,12%)    !REPOSITION CURSOR.
190     D = FNV5(11%,0%,0%)     !ERASE SYMBOLS.
200     INPUT "CONTINUE";C$     !PAUSE.
210     GOSUB 1000              !CLEAR SCREEN.
        !
        !USE FNV6 TO LABEL A GRAPH.
        !
220     D = FNV5(2%,1%+2%+32%+64%+128%,0%)
230     PRINT 'TYPE X AXIS LABEL (UP TO 80 CHARACTERS):'
240     D = FNV5(9%,0%,2%)      !POSITION CURSOR.
250     INPUT X1$
260     GOSUB 1000              !CLEAR SCREEN.
270     PRINT 'TYPE Y AXIS LABEL (UP TO 24 CHARACTERS):'
280     D = FNV5(9%,0%,2%)
290     INPUT Y1$
```

```
300       GOSUB 1000                !CLEAR SCREEN.
310       FOR I%=0% TO 499%
320               V5(I%) = SIN(PI*I%/50.)*100.+100.  !V5=SINE.
330       NEXT I%
340       D = FNV5(5%,0%,1%)\D = FNV5(4%,1%,0%)   !DISPLAY AXES.
350       D = FNV5(9%,0%,23%)       !CURSOR TO BOTTOM LINE.
360       D = FNV6(80%,X1$)         !DISPLAY X AXIS LABEL.
370       FOR I% = 1% TO LEN(Y1$)
380               L2% = I%-1%               !L2% IS A LINE COUNTER.
390               D = FNV5(9%,0%,L2%)       !POSITION CURSOR FOR 1 CHARACTER.
400               D = FNV6(1%,MID(Y1$,I%,1%))     !DISPLAY ONE CHARACTER.
410       NEXT I%                   !CONTINUE UNTIL Y AXIS LABEL IS COMPLETE.
420       D = FNV5(1%,0%,0%)        !NOW DISPLAY THE SINE FUNCTION.
430       D = FNV5(3%,-500%,0%)
440       D = FNV5(9%,40%,0%)
450       INPUT "X POSITION OF MARKER";M1%
460       D = FNV5(9%,40%,0%)
470       D = FNV5(11%,0%,0%)       !ERASE THE PREVIOUS MESSAGE.
480       D = FNV5(6%,M1%,1%)       !PUT A MARKER AT X=M1%.
          !
1000      !THIS SUBROUTINE CLEARS THE SCREEN.
1010      D = FNV5(9%,0%,0%)        !CURSOR TO HOME POSITION.
1020      D = FNV5(10%,0%,0%)       !ERASE SCREEN.
1030      RETURN
32767     END
```

3.4 FNV8, THE STEP HISTOGRAM FUNCTION

Call Format: D = FNV8(N%,O$,Q$)

FNV8 creates **step histograms**, or vertical bar graphs, from the array V5. V5 must be properly **dimensioned** and filled with values in the main program, and you must select either Shaded Graph 0 or Shaded Graph 1 before the FNV8 call. After these preliminaries, a call to FNV8 has the following results:

1. The first N% values are selected from V5.

2. The screen width (512 x units) is divided into N% equal-sized segments.

3. N% **horizontal vectors** are drawn, each with a width of 512%/N%. The screen area beneath each vector is then **shaded** to create N% vertical bars.

4. The horizontal vectors have been **enabled separately** as Graph 0 (or 1 if you selected that number before the FNV8 call). The entire step figure is then enabled as a Shaded Graph with the same number as the vectors' graph number. Because of this dual use of the same graph number, the **envelope** of the step histogram (that is, the vectors) will have a higher display intensity than the rest of the figure. As was mentioned previously (Section 3.1.2), the effect of using the same graph number twice is usually undesirable, although it adds clarity to the final figure in this case.

5. The different bars will have heights equal to the corresponding "y values" selected from V5.

6. Finally, the character strings you have supplied for O$ and Q$ are displayed as **labels** for the horizontal and vertical axes of the histogram, respectively. O$ can be a string of up to 80 characters, and Q$ as many as 24 characters.

Consider the following calling sequence:

```
.
.
.
10      DIM V5(19)
20      D = FNV5(1%,0%,0%)              !Select graph number.
30      FOR I% = 0% TO 19%
40      V5(I%) = 10%*(I% + 1%)          !Fill V5 with integers.
50      NEXT I%
60      D = FNV8(20%,'FIRST 20 INTEGERS','INTEGERS TIMES TEN')
.
.
.
```

This fragment would display twenty bars, each about 25 x units wide (512%/20%). The first bar would be 10 y units tall and the twentieth bar, 200 units.

Besides being a useful function in its own right, FNV8 is a good model for new functions you can create for your special needs. BASIC-PLUS has easy procedures for writing new functions, and your growing experience with VT55 programming will allow you to combine FNV5, FNV6, FNV7, and FNV8 to create increasingly powerful graphic commands. Appendix C of this manual suggests other functions that you may find useful.

APPENDIX A
FORTRAN (PLOT55) ASSEMBLY AND LINKING PROCEDURES

PLOT55 is supplied in both source and object form. If the PLOT55 file in your kit is labeled PLOT55.MAC, it is a MACRO-11 source file and must be assembled (Section A.1) and then linked to each program you write that contains PLOT55 calls (Section A.2). If you are given a preassembled object file (PLOT55.OBJ), you only have to follow the linking procedures for your operating system.

A.1 ASSEMBLING AN OBJECT FILE

This procedure will translate the MACRO-11 assembly language code in PLOT55.MAC into binary machine instructions. You must do this job and the linking procedure shown in Section A.2 before your program will work.

Suppose you are using a PDP-11/34 computer with a pushbutton console, two RK05 disk drives, and the RT-11 operating system.

1. Put the DECpack containing PLOT55.MAC in Disk Drive 1 (DK1:) and close the door. Push the switch on the disk drive panel from LOAD to RUN.

2. In the same manner, load your "system disk" into Drive 0 (DK0:). This disk should contain an RT-11 operating system, the MACRO-11 assembler (MACRO.SAV), and the System MACRO Library (SYSMAC.SML).

3. Start the RT-11 system by these procedures:

   A. Push the button marked CTRL on the PDP-11/34 and hold it down while you press the BOOT button. The terminal will display a list of octal numbers followed by the $ symbol.

   B. Type the two-letter device code on the terminal that represents the storage device containing your system. Because we are using RK05 disks in our example, you would type the underlined symbols:

      ```
      077776  012456  127645
      $DK <CR>
      ```

   C. The computer will read the bootstrap from your system in Drive 0, start the RT-11 system, and display the system title, for example:

      ```
      RT-11 F/B V02C
      ```

      Then the system will display a period (.) at the left margin, meaning that it is ready to accept system commands.

4. Assemble the PLOT55 object file by typing the underlined parts of these lines:

```
.R MACRO <CR>
*PLOT55=DK1:PLOT55 <CR>
*↑C
```

The second asterisk shown above will not appear until the assembly of the object file is complete. To exit from MACRO, you then type "Control C" by holding down the CTRL key on the terminal while you type a C. At this point PLOT55.OBJ has been created on Disk 0. If you want the new object file to be placed on Disk 1, type DK1:PLOT55 on both sides of the equal sign rather than only the right side.

If you are using a different computer:

Follow the customary procedures for bootstrapping the system and then begin with Step 4 of the previous assembly procedure.

If you are using an RSX-11 system:

When your PLOT55 source is on a DECtape (DT:) or RK disk (DK:):

```
>MOU DK:(or DT:)PLOT55 <CR>
>MAC PLOT55=DK:(or DT:)[200,200]PLOT55 <CR>
```

When your source is on magnetic tape:

```
>FLX SY:/RS=MT: (or MM:) [200,200] PLOT55.MAC/DOS <CR>
>MAC PLOT55=PLOT55 <CR>
```

Both procedures will assemble PLOT55.OBJ on the system device, placing the object file in the UIC under which you have logged on to the system. The additional FLX command is necessary in the second step because the magnetic tape is in DOS-11 format.

If you are using an IAS system:

When your PLOT55 source is on a DECtape (DT:) or RK disk (DK:):

```
PDS>MOUNT <CR>
DEVICE? DK: (or DT:) <CR>
VOLUME-ID? PLOT55 <CR>
PDS>MACRO DK: (or DT:) [200,200]PLOT55 <CR>
```

When your PLOT55 source is on magnetic tape:

```
PDS>MOUNT/FOREIGN/OVERRIDE:VOLUMEID<CR>
DEVICE? MT: (or MM:) <CR>
VOLUME-ID? PLOT55 <CR>
PDS>COPY <CR>
FROM? MT: (or MM:) [200,200] PLOT55.MAC/DOS <CR>
TO? *.* <CR>
PDS>MACRO PLOT55 <CR>
```

After either procedure, the assembled file PLOT55.OBJ will exist on the system device with the same UIC under which you logged on to IAS.

If your storage devices are not RK05 disks:

The only change is the two-letter code for the device. Standard codes include:

SY:	=	Any device identified beforehand as the system device for your installation.
DB:	=	RP04, RP05, or RP06 disk drive.
DX:	=	RX01 "floppy disk" drive.
DT:	=	DECtape drive.
CT:	=	DECcassette tape drive (RT-11 only).
DP:	=	RP02/03 disk drive.
MT:	=	TU10 magnetic tape drive.
MM:	=	TJU16 magnetic tape drive.

A.2 LINKING AN OBJECT FILE TO YOUR PROGRAM

A.2.1 Procedure for RT-11 Users

Type the underlined parts of the following RT-11 commands:

```
.R LINK <CR>
*program=object,PLOT55/F <CR>
*↑C
```

where "program" is the name you want for the finished program, and "object" is the name of the object file for that program (that is, the output of the FORTRAN compiler). PLOT55 is the object file that was either supplied in your software kit or was produced by the assembly procedure in Section A.1. The "/F" switch tells the LINK program that your program is using FORTRAN statements and library functions. LINK will look for the FORTRAN library (FORLIB.OBJ) on the system disk. If FORLIB is on some other storage device, follow the PLOT55 name with ",dc:FORLIB" instead of the /F, where dc: is the code for the device containing FORLIB.

LINK will print the second asterisk shown to indicate that the linkage was successful. To exit from LINK, type CTRL C.

Your VT55 program can now be run by typing

```
.R program <CR>
```

A.2.2 Procedure for RSX-11 Users

Type the underlined part of the following command in response to the prompt symbol (>):

```
>TKB task=object/FP,PLOT55 <CR>
```

The name you insert for "task" will be assigned to the finished, executable task image of your FORTRAN program. The label "object" refers to the compiled form of your program, and "PLOT55" is actually PLOT55.OBJ, the object file that was either supplied in your software kit or created by the assembly procedures in Section A.1. The /FP switch means that the Floating Point Processor will be used.

Notice that the command line has no references to a FORTRAN library. The task builder will expect SYSLIB to contain the FORTRAN object code needed by your program. If SYSLIB is not on your disk, or if

you are using a separate FORTRAN library, you have to state the exact name and location of the library in the TKB command, for instance:

```
>TKB task=object/FP,PLOT55,DK1:FORLIB/LB <CR>
```

Regardless of the particular TKB command you use, the prompt symbol will reappear when the task image is ready. You are then ready to run your VT55 program by typing

```
>RUN task <CR>
```

If you received the binary file PLOT55.OBJ in your kit rather than the source file PLOT55.MAC, you should transfer the object file to your UIC before running TKB. Use the RSX-11 MOU and FLX commands described in Section A.1. The only change required is making the file name extension .OBJ instead of .MAC.

A.2.3 Procedure for IAS Users

Type the underlined part of the following command in response to the Program Development System's prompt symbol:

```
PDS>LINK/TASK:task object PLOT55 <CR>
```

where task is the name you want for the finished, executable task image and object is the name of the file containing the compiled form of your main program. PLOT55 represents the file PLOT55.OBJ, which was either supplied in your software kit or created by a previous assembly of the file PLOT55.MAC. The PDS> prompt symbol will reappear when the task is finished and ready to run. The IAS LINK utility assumes that the Floating Point Processor will be used.

When the LINK program is finished building the task, the task can be executed with the command

```
PDS>RUN task<CR>
```

If you received the binary file PLOT55.OBJ in your kit rather than the source file PLOT55.MAC, you should transfer PLOT55.OBJ to your UIC before running LINK. Use the same IAS MOUNT and COPY commands described in Section A.1, changing the .MAC extension in the COPY command to .OBJ.

APPENDIX B
SUGGESTED PLOT55 APPLICATIONS

This appendix contains examples of FORTRAN subroutines that combine various PLOT55 calls to achieve "higher level" functions. For example, your use of a VT55 might require the display of a large number of graphs, in which case you can avoid a large number of PLOT55 calls in your program by following the techniques suggested in this appendix.

Not all of these routines would be needed by every user, and there is rarely any saving in execution speed, because the routines result in just about the same number of PLOT55 calls as would occur anyway. The advantage of subroutines is that they allow you to define, debug, and save an entire procedure, which can then be reused as often as necessary without the risk of new programming errors.

You can try the routines suggested in this appendix by compiling and linking them as you did for PLOT55. You may find that they can be used as they are shown here or that some additional modifications would make them more applicable to your needs.

Notice that these subroutines use the four-argument form of PLOT55. The fourth argument is the 16-element array ISTAT, and it is placed in a common area labeled STATUS. The use of such a common status table allows you to share the same status information between any number of routines. Within each routine, individual elements of ISTAT can be examined and changed to control the overall status of the VT55 display. For example, the element ISTAT(8) stores the current graph number; ISTAT(8) equals 0 if Graph 0 is in use and 8 if Graph 1 is in use. The full list of status table elements and their uses is as follows:

ISTAT(1):	Logical unit number of terminal.
ISTAT(2):	Event flag number to be used.
ISTAT(3):	X value of origin.
ISTAT(4):	Y value of origin.
ISTAT(5):	Work register (do not change contents!).
ISTAT(6):	Work register.
ISTAT(7):	Graphic status (holds the current total from IENAB).
ISTAT(8):	0 if Graph 0 is in use; 8 if Graph 1.
ISTAT(9):	Graphic mode switch (0=Off, 1=On).
ISTAT(10):	Work register for vectors (do not change!).
ISTAT(11):	Character count (RSX-11 only).
ISTAT(12)- ISTAT(16):	Character output buffer (RSX-11).

B.1 INITIALIZATION

In most applications, it is useful to have a routine such as the one shown here that will return the VT55 to some initial state as preparation for running a new program.

```
SUBROUTINE INIT
COMMON/STATUS/ISTAT(16)
DATA ISTAT/16*0/
```

```
CALL PLOT55(13,72,,ISTAT)
CALL PLOT55(13,74,,ISTAT)
CALL PLOT55(2,1+512,,ISTAT)
RETURN
END
```

The statement CALL INIT will first clear all alphanumeric text from the screen and then will disable all graphic figures, clearing the screen of graphic displays as well. INIT is therefore a good example of a "starting routine" for a graphic program.

B.2 GRIDS AND COORDINATE AXES

```
      SUBROUTINE GRID(IDX,IDY)
      COMMON/STATUS/ISTAT(16)
      CALL PLOT55(2,1+32+64,,ISTAT)
      DO 1 I=1,512
1     CALL PLOT55(5,I-1,0,ISTAT)
      DO 2 I=1,236
2     CALL PLOT55(4,0,I-1,ISTAT)
      DO 3 I=1,512,IDX
3     CALL PLOT55(5,I-1,1,ISTAT)
      DO 4 I=1,236,IDY
4     CALL PLOT55(4,1,I-1,ISTAT)
      RETURN
      END
```

The numbers you supply for IDX and IDY in a CALL GRID statement will be the spacing in screen units between the vertical and horizontal lines, respectively. GRID will therefore display a rectangular grid on the screen, which can be superimposed on other figures such as graphs and histograms. If GRID is called several times in the same program, it will erase any old grid before displaying a new one.

To display only a y axis at the left edge and an x axis at the bottom of the screen, use a CALL GRID statement with IDX equal to any integer greater than 511 and IDY any integer greater than 235.

The markers described in Chapter 2 (ICMD = 6) are the best way to mark off a graph in the x direction. You may, however, want to put a vertical scale on a graphic figure as well. With a routine such as GRID, you could, for example, scale the screen vertically in increments of 20 units by calling GRID with IDX > 511 and IDY = 20.

B.3 GRAPH PLOTTING ROUTINES

The examples in this section suggest "canned" procedures for the most common use of VT55: plotting graphs or shaded graphs of already computed data.

```
      SUBROUTINE GRAPH(N,IARRAY)
      COMMON/STATUS/ISTAT(16)
      DIMENSION IARRAY(512)
      NUMBER=ISTAT(8)/8
      CALL PLOT55(7,0,0,ISTAT)
      CALL PLOT55(8,512,0,ISTAT)
      CALL PLOT55(2,1+(NUMBER+1)*2,(NUMBER+1)*10,ISTAT)
      CALL PLOT55(3,-N,IARRAY,ISTAT)
      CALL PLOT55(1,1-NUMBER,,ISTAT)
      RETURN
      END
```

When you write a CALL GRAPH statement, a graph will appear that displays the first N points from IARRAY (where IARRAY is some integer array that you have filled with the points you want to display).

This particular GRAPH routine also switches automatically between Graph 0 and Graph 1, in the following manner:

1. After a CALL INIT statement, CALL GRAPH will use Graph 0 to plot the array you supply.

2. If GRAPH is called a second time in the same program (without calling INIT again), the second graph will appear as Graph 1, superimposed on Graph 0.

3. If GRAPH is called a third time, the new graph will be Graph 0 again, erasing the old Graph 0 but leaving Graph 1 on the screen. The erasure of the old graph is done by drawing a vector from the point (0,0) to the point (512,0), which is a short method for resetting all the y values of a graph to 0.

4. Consequently, all even-numbered calls to GRAPH will create a new Graph 1, and odd-numbered calls a new Graph 0.

```
SUBROUTINE SHADOW
COMMON/STATUS/ISTAT(16)
NUMBER=ISTAT(8)/8
CALL PLOT55(2,1+(1-NUMBER)*8+8,(1-NUMBER)*2+2,ISTAT)
RETURN
END
```

CALL SHADOW converts a regular graph to a shaded graph.

If you want to "shade" a particular graph, you must call SHADOW immediately after calling GRAPH, for example,

```
CALL GRAPH(500,SINE)
        [Plot the sine function on Graph 0.]
CALL SHADOW [Shade Graph 0.]
CALL GRAPH(500,COSINE)   [Graph 1.]
CALL SHADOW   [Shade Graph 1.]
```

Shading is a useful practice for distinguishing two graphs. For example, the sine and cosine graphs have exactly the same shape, and so are much easier to tell apart if one of them is shaded.

```
      SUBROUTINE BARS(N,IARRAY)
      COMMON/STATUS/ISTAT(16)
      DIMENSION IBAR(236,2),IARRAY(512)
      NUMBER=ISTAT(8)/8
      CALL PLOT55(7,0,0,ISTAT)
      CALL PLOT55(8,512,0,ISTAT)
      CALL PLOT55(2,1+(NUMBER+1)*2,,ISTAT)
      WIDTH=512./FLOAT(N)
      DO 1 I=1,N
      IBAR(I,1)=I*WIDTH
1     IBAR(I,2)=IARRAY(I)
      CALL PLOT55(7,0,IBAR(1,2),ISTAT)
      DO 2 I=1,N
```

```
            CALL PLOT55(8,IBAR(I,1),IBAR(I,2),ISTAT)
2           CALL PLOT55(7,IBAR(I,1),IBAR(I+1,2),ISTAT)
            CALL PLOT55(1,1-NUMBER,,ISTAT)
            CALL SHADOW
            RETURN
            END
```

BARS creates a special type of shaded graph called a step histogram, or "bar graph." BARS will display N vertical shaded bars of width 512/N on the screen. The bars will have heights equal to the first N elements of IARRAY. This example of a bar graph routine has the same mechanisms as the previous GRAPH routine for switching between Graph 0 and Graph 1 and for erasing old graphs (the "zeroing vector").

B.4 LABELING A GRAPHIC DISPLAY

```
            SUBROUTINE LABEL(IXLABL,IYLABL)
            COMMON/STATUS/ISTAT(16)
            LOGICAL*1 IYLABL(24)
            CALL PLOT55(9,0,23,ISTAT)
            CALL PLOT55(12,,IXLABL,ISTAT)
            DO 3 I=1,24
            IF(IYLABL(I).EQ.0) GO TO 4
            CALL PLOT55(9,0,I-1,ISTAT)
3           CALL PLOT55(12,1,IYLABL(I),ISTAT)
4           CALL PLOT55(9,0,23,ISTAT)
            CALL PLOT55(13,91,,ISTAT)
            PAUSE
            CALL PLOT55(13,92,,ISTAT)
            RETURN
            END
```

LABEL will use the two strings you enter for IXLABL and IYLABL as axis labels for the x and y axes, respectively. Note that the labels you supply must be enclosed in single quotation marks (') to identify them as alphanumeric strings in FORTRAN symbology.

The x axis label will appear on the very bottom line of the screen, below the x axis. It can be up to 80 characters long (including spaces).

The y axis label will read from top to bottom (one letter per line) and will appear in "Column 0," the leftmost column of the screen. This column is to the left of a vertical line at x = 0, because the graph area of the VT55 is slightly smaller than the alphanumeric area. The y axis label can be up to 24 characters long, including spaces.

Notice that the LABEL routine sends two escape sequences. The first one holds the screen to prevent the STOP message from interfering with the display. The screen will not be released until you type a carriage return (or a "RESUME task" command in RSX-11 systems). If you will be using IAS, replace the PAUSE statement with the statement pair

```
            READ(5,5)KR
5           FORMAT(I2)
```

The PAUSE statement will be ignored in an IAS execution. (To bring the terminal back to a completely normal state after this sort of procedure, press the orange SCROLL key until the STOP message and the operating system's prompt symbol reappear.)

B.5 ATTACHING THE VT55 AS AN OUTPUT DEVICE

The subroutine ATTACH assigns logical unit 1 to the terminal number of the VT55. Then the routine stores ILUN=1 and IEFN=1 in the status table. This subroutine should precede any other PLOT55 calls in a program.

```
      SUBROUTINE ATTACH
      COMMON/STATUS/ISTAT(16)
      WRITE(5,1)
1     FORMAT('WHAT IS THE VT55 TERMINAL NUMBER?')
      READ(5,2) NR
2     FORMAT(O2)
      CALL ASNLUN(1,'TT',NR)
      CALL PLOT55(0,1,1,ISTAT)
      RETURN
      END
```

When you use this routine or any other that attaches the VT55, remember to **detach** the VT55 at the end of the program.

```
      .
      .
      .
      CALL PLOT55(0,-1,0,ISTAT)
      END
```

NOTE

ATTACH will only work with IAS or RSX-11 systems.

APPENDIX C
SUGGESTED BASIC-PLUS APPLICATIONS

Chapter 3 suggested that you may want to design some BASIC-PLUS functions of your own, using the four functions supplied in your kit (FNV5, FNV6, FNV7, and FNV8) as building blocks. This appendix gives you a few samples that show the four functions as they might be combined to yield even higher-level, and therefore simpler, graphic support.

Your VT55 software kit is supplied either on RK05 disk, DECtape, or DOS-11 format magnetic tape. Use the following procedures to write the VT55 software onto your RSTS/E system disk:

For RK05 disk distribution:

1. Place the RK05 disk in disk drive 0 (DK0:). Close the door.

2. Log onto the system and type the following command in response to the READY message:

```
MOUNT DK0:VT55 <CR>
```

(Change the device number to DK1: if you are using drive 1.)

3. Now use the PIP program to copy the disk:

```
READY
PIP <CR>
.....(PIP VERSION printed by system)
#SY:VT55.BAS=DK0:VT55.BAS(200,200) <CR>
#↑Z
READY
DISMOUNT DK0:VT55 <CR>
```

For DECtape distribution:

The same procedure is used as for RK05 disks except that the device code, wherever it appears, should be DT0: (for DECtape drive 0) or DT1: (for drive 1). The MOUNT and DISMOUNT commands may not be necessary; check with your system manager.

For magnetic tape distribution:

```
READY
MOUNT MT0:<CR>
READY
```

```
PIP <CR>
#SY:VT55.BAS=MT0:[200,200]VT55.BAS/DOS <CR>
#↑Z
READY
DISMOUNT MT0:/UNLOAD <CR>
```

All of these procedures will write the file VT55.BAS onto the system disk under the account that you used to log on.

C.1 INITIALIZATION

The standard function FNV7 is called an initialization function because it erases old graphic displays from the screen and returns the VT55 terminal to its initial state, ready to receive new graphic programming instructions. What FNV7 does not do, however, is to also clear the screen of the letters and numbers that make up the alphanumeric part of a display. In other words, FNV7 performs a "single" initialization: the graphic display is cleared from the screen, but the alphanumeric display remains. FNV7 is thus useful when you want to change from one graph to the next without changing the labels on the axes. In other cases you might want to combine the graphic initialization of FNV7 with a second initialization that clears the alphanumeric information from the screen as well. Consider this example:

```
15000 DEF FNV9

15010 D = FNV5(9%,0%,0%)                !Cursor to home position.

15020 D = FNV5(10%,0%,0%)               !Erase screen.

15030 FNEND
```

FNV9 is a different kind of single initialization; it clears the alphanumeric display from the screen but leaves the graphic display alone. FNV9 could be used by itself in a program you write, or it could be combined with FNV7 in yet another function that performs a "dual" initialization:

```
15040 DEF FNV1

15050 D = FNV7

15060 D = FNV9

15070 FNEND
```

If your graphic programs contained these function definitions, then the statement D = FNV7 would clear only the graphic display, D = FNV9 would clear only the alphanumeric display, and D = FNV1 would clear both displays.

C.2 GRIDS AND COORDINATE AXES

```
16000    DEF FNV2(D1%,D2%)

16010    D = FNV5(2%,1%+32%+64%,0%)

16020    FOR I% = 1% TO 512%

16030    D = FNV5(5%,I%-1%,0%)
```

```
16040       NEXT I%

16050       FOR I% = 1% TO 236%

16060       D = FNV5(4%,0%,I%-1%)

16070       NEXT I%

16080       FOR I% = 1% TO 512% STEP D1%

16090       D = FNV5(5%,I%-1%,1%)

17100       NEXT I%

17110       FOR I% = 1% TO 236% STEP D2%

17120       D = FNV5(4%,1%,I%-1%)

17130       NEXT I%

17140       FNEND
```

The statement D = FNV2(10%,20%) would display a rectangular grid on the VT55 screen, with the vertical grid lines separated by 10 units and the horizontal lines separated by 20 units. This particular definition of FNV2 is written in such a way that if FNV2 is called several times in the same program, Lines 16020-16070 will erase the old grid before a new one is displayed. Grid lines help give a graphic display a visual scale; for example, when you display a graph on the screen, you may want to know where the points x = 0, x = 25, and x = 50 are located and what is the approximate value of the graph at these points. The markers described in Chapter 3 will mark graphs at chosen x positions, but that feature will not give quite as much numerical information as you want. As an alternative to actually labeling the screen with numbers, you could display the graph and then write the statement

```
D = FNV2(25%,5%)
```

This statement would cover your graph with a "checkerboard," scaling the graph in both directions. The second line from the left would mark the position x = 25, the third line, x = 50, and so forth; in the y direction, you could use the horizontal lines, which would be 5 units apart, to estimate the value of any point on the graph, with an accuracy of about 2 units.

You could also use FNV2 to put just two important lines, the x and y axes, on the screen, as follows:

```
D = FNV2(512%,236%)
```

If you look back to the definition of FNV2, you will see that this statement would make FNV2 stop after it had displayed the first line in each direction, so that you would have a single vertical line at x = 0 (the y axis) and a single horizontal line at y = 0 (the x axis).

Finally, notice that Line 16010 of the FNV2 definition enables both line types but does not affect any other type of graphic figure. Because FNV2 only affects the display of grid lines, you could display one graph and change the grid scale as many times as necessary to get the right result.

C.3 PLOTTING GRAPHS

Because the VT55 is so often used to display graphs, it is usually convenient to have a separate function prepared that performs all the necessary initialization, enabling, graph number selection, and display in one step. The function FNV3, defined below, is suggested for this purpose.

```
15100    DEF FNV3(N%)
15110    D = FNV5(7%,0%,0%)
15120    D = FNV5(8%,511%,0%)
15130    D = FNV5(2%,1%+(C7%+1%)*2%,(C7%+1%)*10%)
15140    D = FNV5(3%,-N%,0%)
15150    D = FNV5(1%,1%-C7%,0%)
15160    FNEND
```

You should take note of the following unusual features of this function definition:

1. Line 15120 draws a vector from the origin (0,0) to the bottom right corner (511,0). This is one of the simpler techniques for erasing the previous graph for a particular graph number, since drawing such a vector fills up the appropriate register of the VT55 graphic memory with 512 points, all of which have a y value of 0.

2. Line 15130 uses the variable C7%, which is the internal variable that stores the current graph number in use (C7% = 0% or 1%). You may recall from the introduction to Chapter 3 that you should avoid using internal variables in your program. In this case, however, the variable C7% is used in the program for the same purpose for which it is used internally. You should, of course, use such internal variables with caution, because changing their numerical values would affect the status of the display.

3. Line 15140 is simply the standard FNV5 call for creating a graph from the array V5. In other words, you should assign the values you want to graph to the array V5 before writing D = FNV3 to create the graph. Line 15140 will then plot the first N% points from V5, starting with an x position of 0.

4. Line 15150 will switch the graphic display from Graph 0 to Graph 1 and back to Graph 0 again automatically as FNV3 is called several times in the same program. The first time FNV3 is called from your program, the graph will appear as Graph 0; the second time, as Graph 1; and so forth. If a very large number of graphs are displayed, just remember that the even-numbered calls will create Graph 1 and the odd-numbered calls will create Graph 0.

C.4 SHADING A GRAPH

This section suggests another function, called FNV4 here, that "shades" a graph, converting it from a normal graph made up of points to a shaded graph.

```
15170    DEF FNV4
15180    D = FNV5(2%,1%+(1%-C7%)*8%+8%,(1%-C7%)*2%+2%)
15190    FNEND
```

FNV4 uses the variable C7%, the graph number, in the same way as described for FNV3. FNV4 should be called immediately after displaying the graph that you want to shade, because FNV4 will only shade the graph with the most recent graph number, for example:

```
                                        .
                                        .
                                        .
15200   V5(I%)=S1(I%)                   FOR I%=0% TO 511%
                                        !S1 CONTAINS SINE VALUES.
15210   D = FNV9                        !ERASE SCREEN.
15220   D = FNV3(512%)                  !PLOT 512-POINT SINE.
15230   D = FNV4                        !SHADE THE SINE GRAPH.
15240   V5(I%)=C1(I%)                   FOR I%=0% TO 511%
                                        !C1 CONTAINS COSINE.
15250   D = FNV3(512%)                  !PLOT 512-POINT COSINE.
                                        .
                                        .
                                        .
                                        .
```

In this example, the sine graph would be shaded, but the cosine graph would remain a standard "point-plot" graph. Shading is very useful in just such a case, because the sine and cosine graphs are identical in shape and can be hard to tell apart unless one of them is shaded.

INDEX

READER'S COMMENTS

NOTE: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. Problems with software should be reported on a Software Performance Report (SPR) form. If you require a written reply and are eligible to receive one under SPR service, submit your comments on an SPR form.

Did you find errors in this manual? If so, specify by page.

_____

_____

_____

_____

_____

_____

_____

Did you find this manual understandable, usable, and well-organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

Is there sufficient documentation on associated system programs required for use of the software described in this manual? If not, what material is missing and where should it be placed?

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent.

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Non-programmer interested in computer concepts and capabilities

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or
Country

Please cut along this line.

---------- Fold Here ----------

---------- Do Not Tear - Fold Here and Staple ----------

FIRST CLASS
PERMIT NO. 33
MAYNARD, MASS.

BUSINESS REPLY MAIL
NO POSTAGE STAMP NECESSARY IF MAILED IN THE UNITED STATES

Postage will be paid by:

digital

Software Documentation
146 Main Street ML 5-5/E39
Maynard, Massachusetts 01754

digital equipment corporation

Printed in U.S.A.