Tektronix®
COMMITTED TO EXCELLENCE

Please Check for
CHANGE INFORMATION
at the Rear of this Manual

PLOT 10
4010A02
ADVANCED GRAPHING II
USER'S MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077

MANUAL PART NO. 070-2244-00
PRODUCT GROUP 11

First Printing MAR 1977
Revised FEB 1982

SUPPORT POLICY

This software product is designated Support Category B, as shown on the applicable software data sheet existing at the time of order. Tektronix' sole obligation shall be to correct defects (non-conformance of the software to the data sheet) as described below, without additional charge.*

During the one (1) year period following delivery, if the customer encounters a problem with the software which his diagnosis indicates is caused by a software defect, the customer may submit a Software Performance Report (SPR) to Tektronix. Tektronix will respond to problems reported in SPRs which are caused by defects in the current unaltered release of the software via the Maintenance Periodical for the software, which reports code corrections, temporary corrections, generally useful emergency by-pass and/or notice of the availability of corrected code. Software updates, if any, released by Tektronix during the one (1) year period, will be provided to the customer on Tektronix' standard distribution media as specified in the applicable data sheet. The customer will be charged only for the media on which such updates are provided, unless otherwise stated in the applicable data sheet, at Tektronix' then current media prices.

*In addition to the locations within the contiguous forty-eight (48) United States and the District of Columbia, this service is available in those areas where Tektronix has software support capability.

4010A02 PLOT 10 ADVANCED GRAPHING II USER MANUAL

INTRODUCTION

4010A02 PLOT 10 Advanced Graphing II is designed for use with the 4010A01 PLOT 10 Terminal Control System. It can be used with (or without) the 4662A01 PLOT 10 Utility Routines.

Advanced Graphing II is the latest addition to the Tektronix PLOT 10 family of terminal software. It is the result of more than four years of experience with the original Advanced Graphing software, and is designed to provide the terminal user with a comprehensive capability to depict his data in the form of a graph.

The package consists of a number of FORTRAN subroutines which are carefully designed to combine simplicity of use with a highly flexible capability to tailor the size, shape, and format of the graph as it is displayed on the screen. Three calls to Advanced Graphing II subroutines are sufficient to create a complete, full-screen graph using the full set of default values. More than 40 small subroutines available to the user permit him to alter any of the details of graph construction to obtain the desired appearance.

For those who have never considered the creation of a graph from an analytical point of view, it may be startling to consider the possibility of more than 40 elements in a graph to change. That is where Tektronix's years of experience in the field of graphing can help. We recommend that you start at the simple end of the scale, and let the carefully considered default conditions operate. Then, as the parts of a graph and the capabilities of the package become more familiar, try to manipulate some of the more complex and subtle capabilities of Advanced Graphing II.

The Primer section of the manual provides the most fundamental information for graph production, and sections 2 and 3 provide more advanced information for varying the graph. For the most part, there is a simple way to perform nearly every task. A table of parameters is available to change each element of the graph individually, but where several changes are needed to perform one task, a single routine is generally available to specify all the parameters.

Appendix B contains descriptions of routines which allow incorporation of non-standard data arrays, lines, symbols, and labels. These routines are "dummies" only and must be written by the user.

Advanced Graphing II provides the tools; they need only be put to use.

CONTENTS

SECTION 1

THE PRIMER

The Primer provides basic information to guide the user through his early contact with the Advanced Graphing II Package. Explanations are detailed and specific. Upon completion of the section he should be able to create a basic graph with one or multiple curves using a long data form.

Initially, it is necessary to create a program which will call the AG-II (Advanced Graphing II) routines necessary to plot a graph. Your computer system will provide the necessary editing facility to create this program. Details of the actual program are described in the following section.

1.1 Creating a Basic Graph

To create a graph initially, it is first necessary to prepare the data which is to be plotted. Two data lists will be necessary - one for the horizontal (X) values, and one for the vertical (Y) values. In the long data form used here, the list will include in the first position the number of data values to be plotted; subsequent positions contain the data values. Each complete list will be called an array.

Example of Data Array in Standard Long Form:

```
DIMENSION XDATA(7),YDATA(7)
DATA XDATA/6.,1.,2.,3.,4.,5.,6./
DATA YDATA/6.,211.,114.,306.,354.,291.,325./
```

The first line is a standard FORTRAN DIMENSION statement assigning the names "XDATA and "YDATA" to the two arrays and setting the length of each to 7 (the total number of entries in the array).

The second line contains the number of data points to be plotted (6) followed by the actual data values (1, 2, 3, etc.) of the X array. The third line contains similar information for the Y array of data.

Once the data is in a usable form we are ready to graph.

Five CALL statements will produce results:

```
CALL INITT(3Ø)
CALL BINITT
CALL CHECK (XDATA,YDATA)
CALL DSPLAY (XDATA,YDATA)
CALL FINITT(Ø,7ØØ)
```

The call to INITT initializes the Terminal Control System (Tektronix Part Number 062-1474-xx) which provides the basic line drawing functions. The parameter (30) indicates that terminal transmission is at 30 characters per second (300 baud). A baud rate of 1200 would be indicated as 120 characters per second.

BINITT initializes the AG-II (Advanced Graphing II) system.

CHECK has two parameters which are the names of the data arrays to be plotted (the data arrays we created prior to calling INITT.) CHECK verifies that all the necessary information is present.

DSPLAY, which displays the graph, also requires the names of the data arrays as parameters (XDATA,YDATA).

Note that the first data array named in the parameter is plotted on the X axis and the second is plotted on the Y axis.

FINITT terminates use of the Terminal Control System. The two parameters are the position to which the alphanumeric cursor is to be moved before termination. (The exact position is not important to us here, but must be entered as a matter of form.)

A complete program and the graph it produces are shown in figure 1.1

```
DATA XDATA/6.,1.,2.,3.,4.,5.,6./
DATA YDATA/6.,211.,114.,306.,354.,291.,325./
CALL INITT(120)
CALL BINITT
CALL CHECK(XDATA,YDATA)
CALL DSPLAY(XDATA,YDATA)
CALL FINITT(0,700)
STOP
END
```

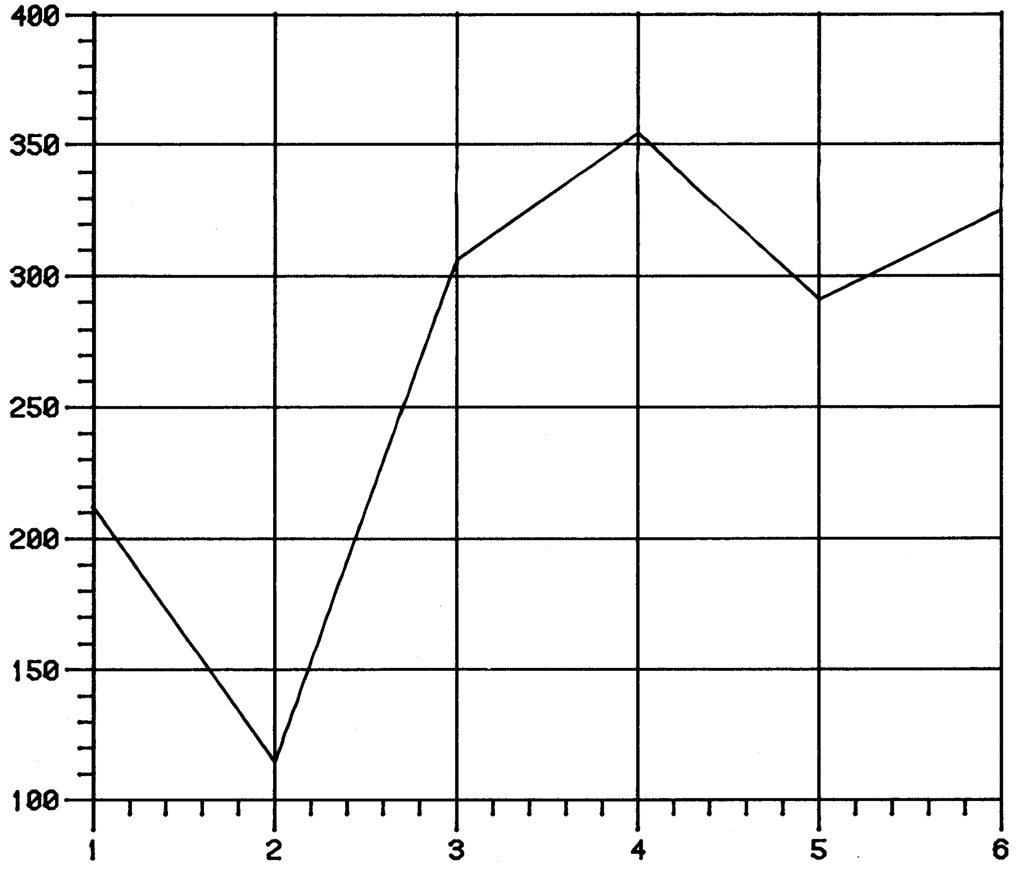


Figure 1.1

1.2 Multiple Curves

It is often desirable to plot two sets of data on the same plot. For example, a businessman might wish to plot sales of two different items for a comparable period of time. To add a second data curve to the plot we just created we will need to add another array of data. NOTE: Since the second curve will be plotted on the same graph as the first, the widest range of data should be plotted first. Following is the code which includes the added values. Since we will use the same horizontal values, we need add only one array for another curve line.

```
DIMENSION XDATA(7),YDATA1(7),YDATA2(7)
DATA XDATA/6.,1.,2.,3.,4.,5.,6./
DATA YDATA1/6.,211.,114.,306.,354.,291.,325./
DATA YDATA2/6.,367.,300.,179.,238.,320.,210./
CALL INITT(30)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL CPLOT(XDATA,YDATA2)
CALL FINITT(0,700)
END
```

Now we will need to add one more CALL statement immediately after the call to DSPLAY. It will draw the second curve.

```
            CALL CPLOT (XDATA,YDATA2)
```

Here XDATA is the name of the horizontal array (the same as for the first plot) and YDATA2 is the new data to be plotted in the second curve. Following is a copy of the complete code and the resulting graph.

```
DIMENSION XDATA(7),YDATA1(7),YDATA2(7)
DATA XDATA/6.,1.,2.,3.,4.,5.,6./
DATA YDATA1/6.,211.,114.,306.,354.,291.,325./
DATA YDATA2/6.,367.,300.,179.,238.,320.,210./
CALL INITT(30)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL CPLOT(XDATA,YDATA2)
CALL FINITT(0,700)
END
```

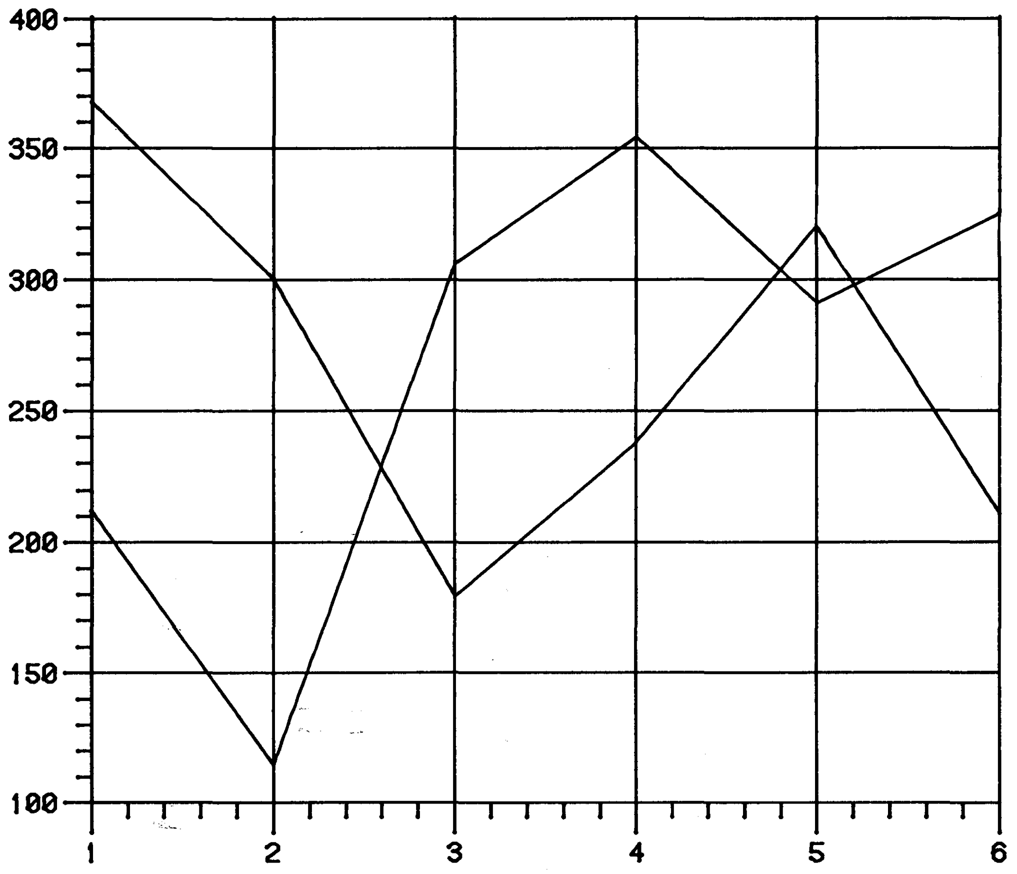


Figure 1.2

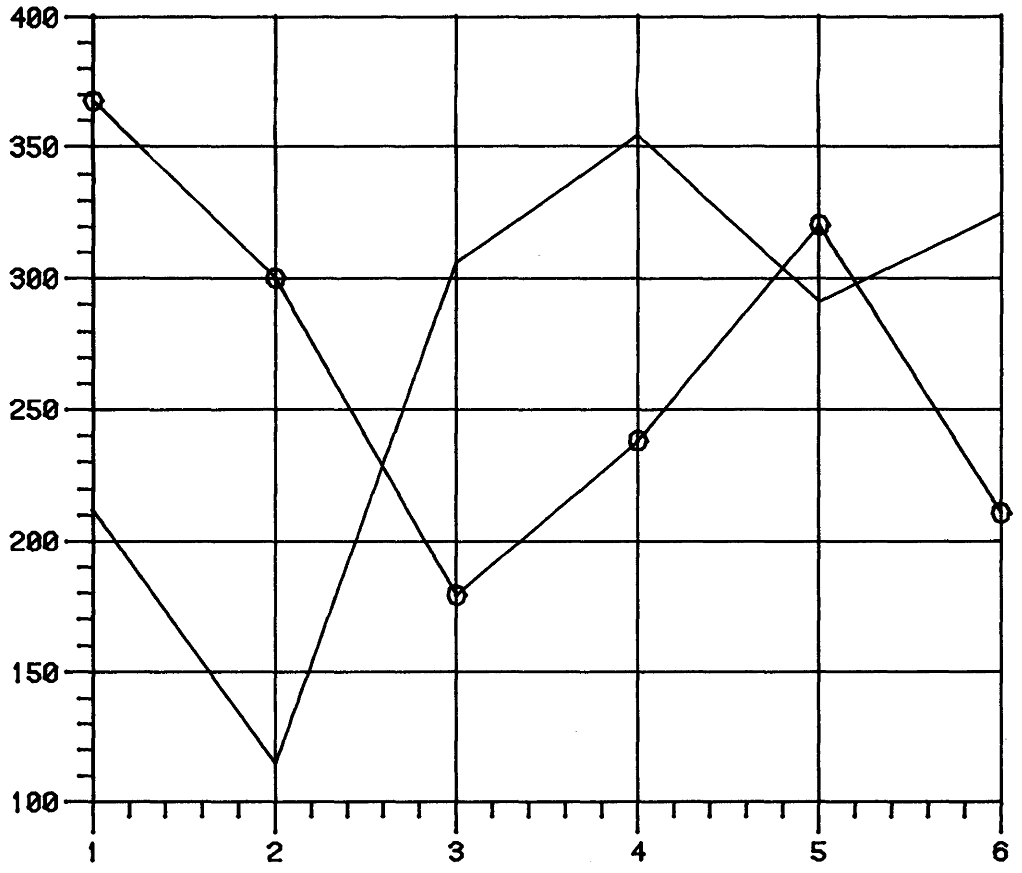
1.3 Introduction to Advanced Plots

Many options are available to change the appearance of the graph. To differentiate the second data curve from the first, we will circle each data point of the second curve by adding the statement:

```
CALL SYMBL(1)
```

immediately before the call to CPLOT. Many other symbols are available, as you will find by referring to section 3 of the User's Manual.

```
DIMENSION XDATA(7),YDATA1(7),YDATA2(7)
DATA XDATA/6.,1.,2.,3.,4.,5.,6./
DATA YDATA1/6.,211.,114.,306.,354.,291.,325./
DATA YDATA2/6.,367.,300.,179.,238.,320.,210./
CALL INITT(30)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL SYMBL(1)
CALL CPLOT(XDATA,YDATA2)
CALL FINITT(0,700)
END
```



Example 1.3

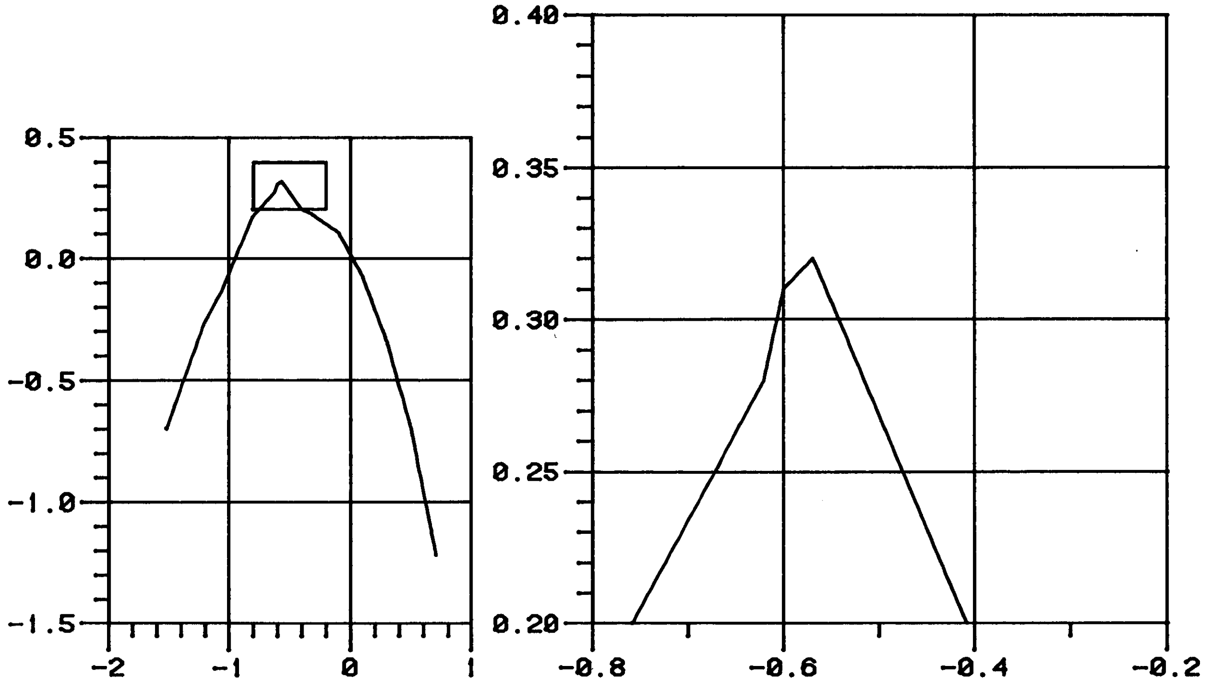
Log plots, bar charts, calendar plotting - what is your wish? Advanced Graphing II provides these options and more.

Perusal of the user's manual will supply the familiarity to allow your creativity full rein. Various kinds of dashed data curve lines, many kinds of tic marks and grid lines, symbols at incremented data points, and a variety of label forms are all available, as well as the option of extra labels entered at any point on the screen.

In the Primer we have used only the long data form. A short data form for regularly incremented data and a calendar form are also available as explained in section 2.2 of the User's Manual.

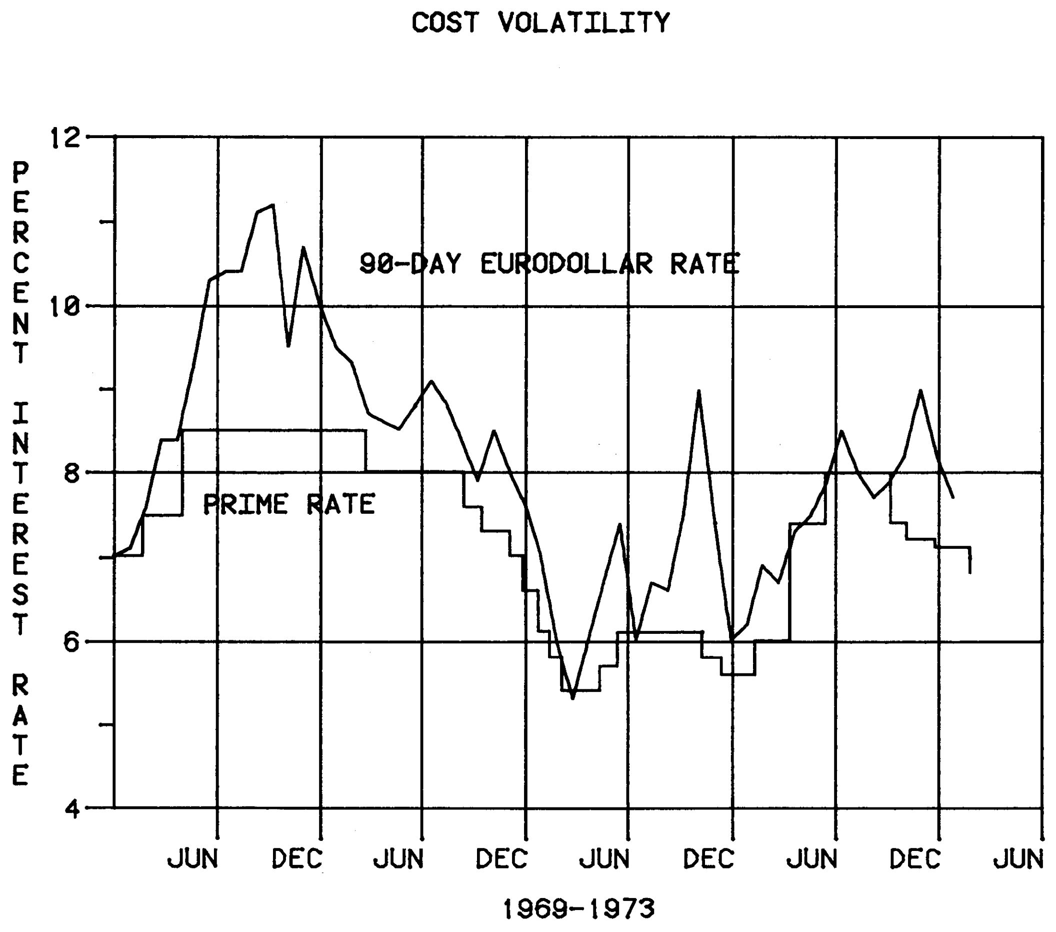
If you would prefer a different standard graph, this can be provided by changing a few internal values in the code as is explained in the systems manual.

The following graphs demonstrate a few of the many possibilities.



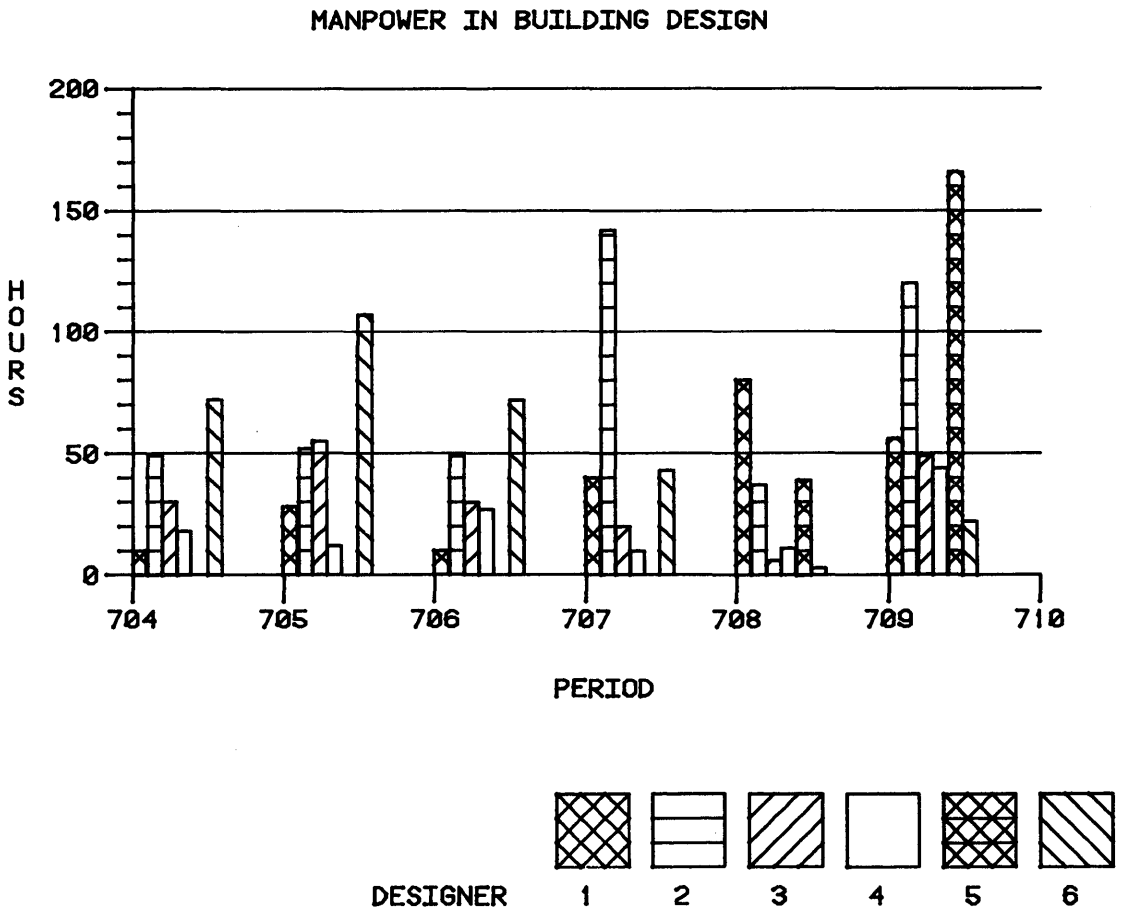
This display selects a portion of the first graph and displays it in detail in the second.

Example 1.4



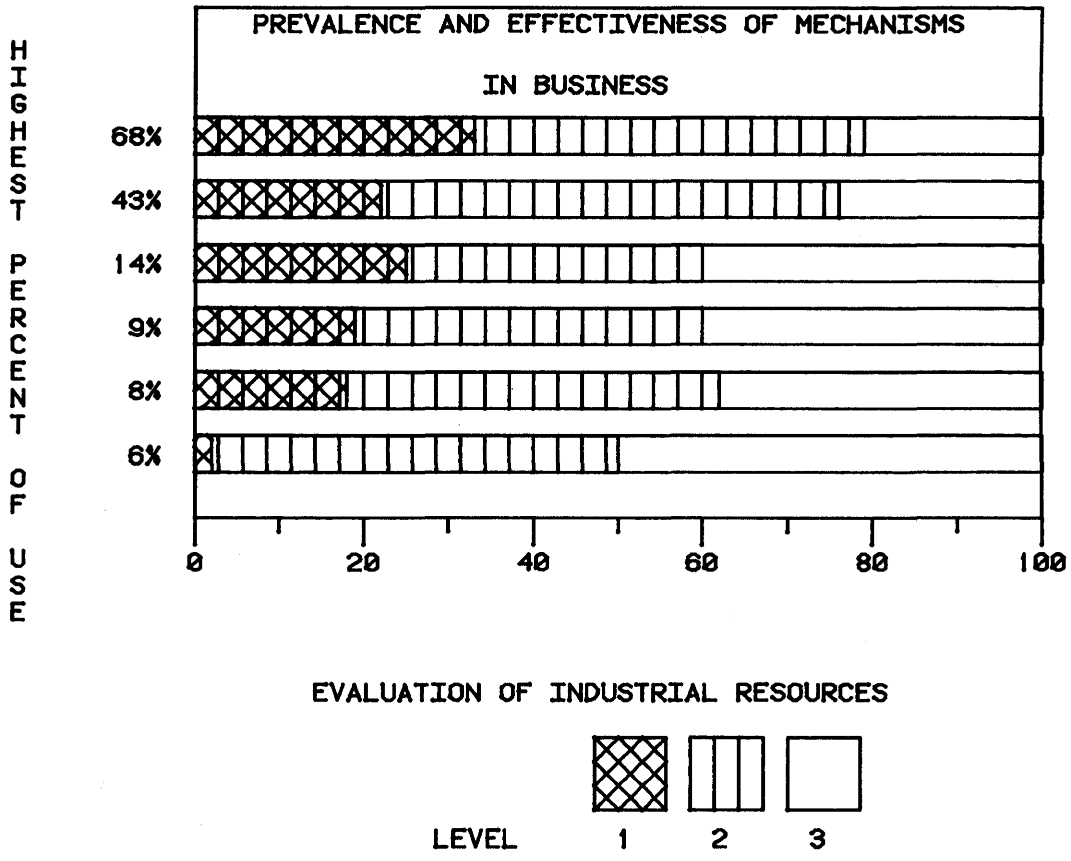
Displays two sets of data using TCS routines to form the stepped line.

Example 1.5



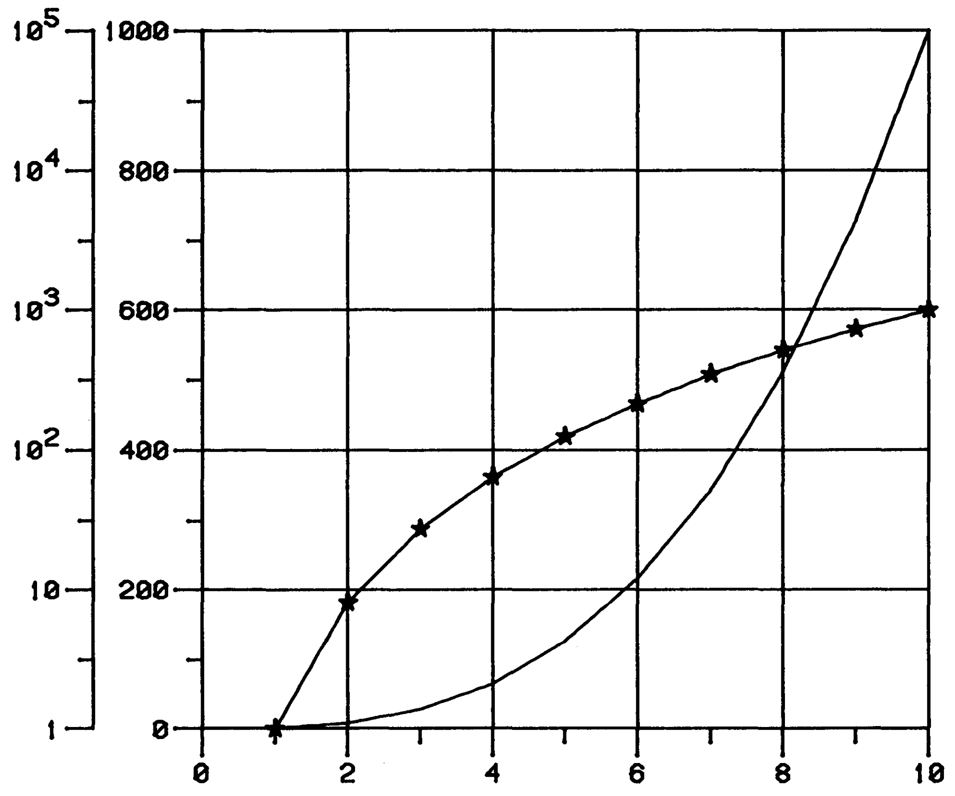
Uses a bar chart to show time spent per period per person.

Example 1.6



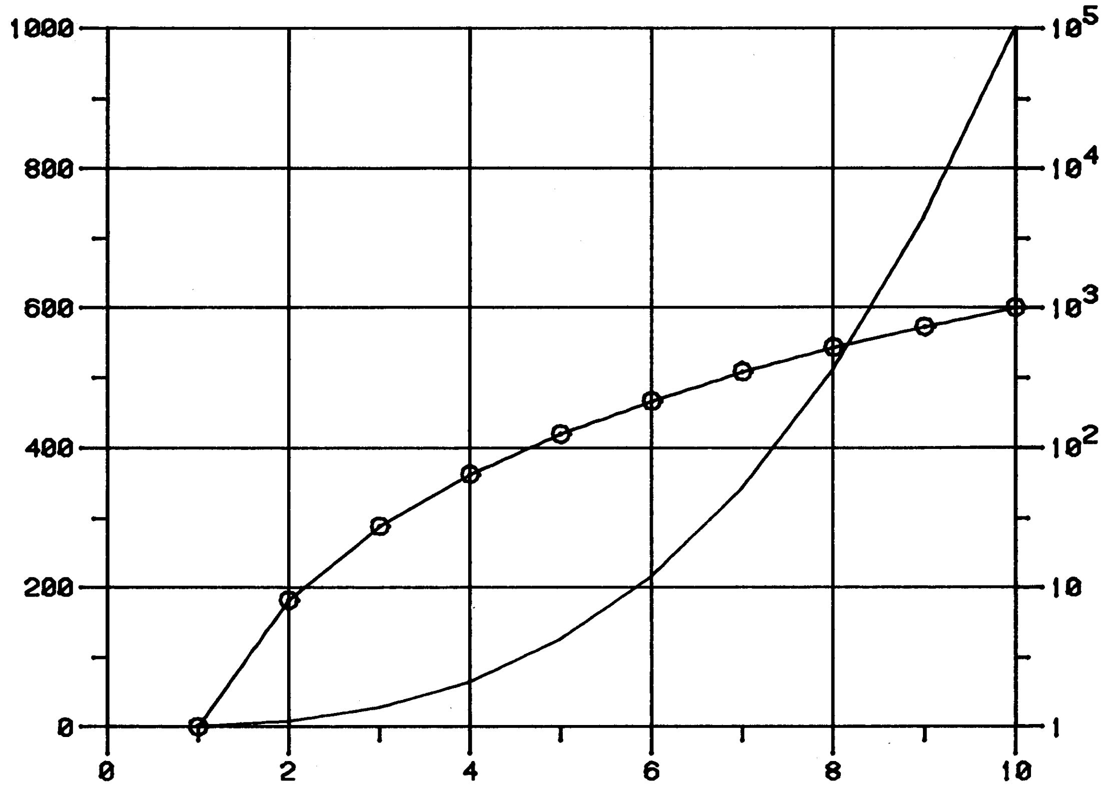
Another use of bar charts

Figure 1.6.1



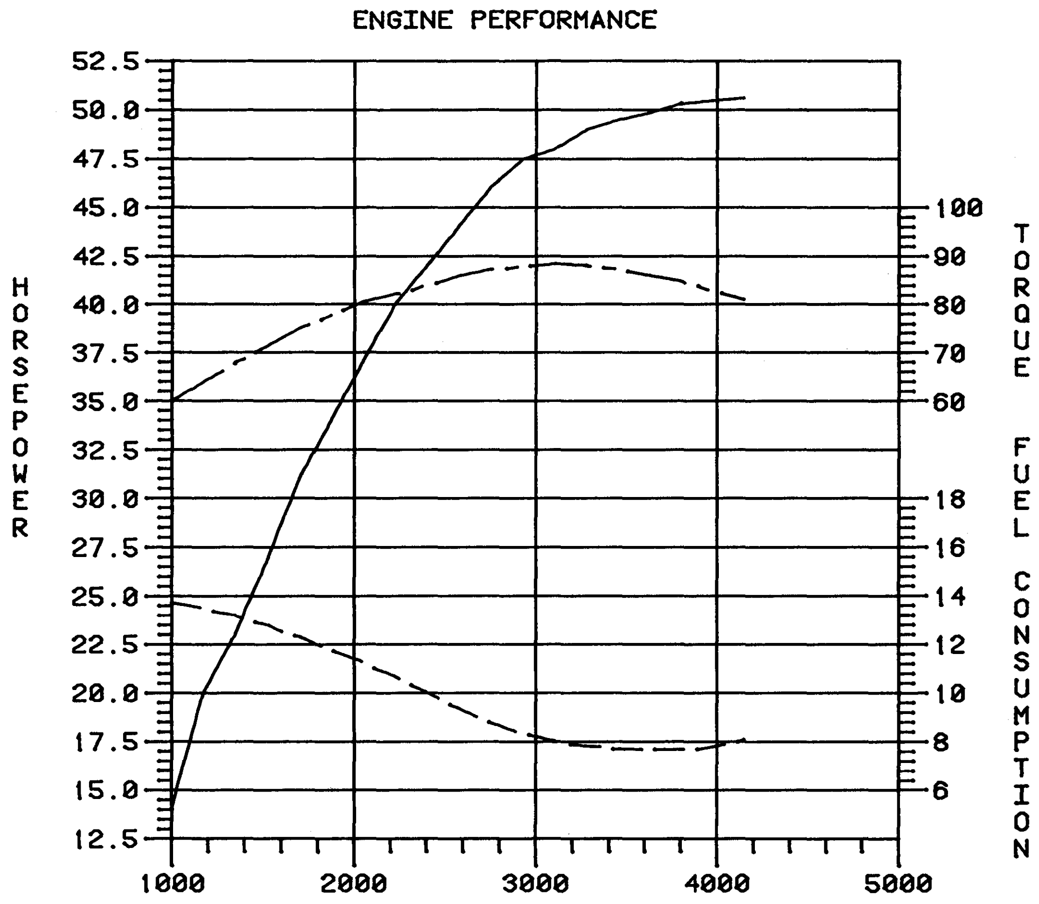
Plots a logarithmic and a linear plot with offset axes.

Figure 1.7



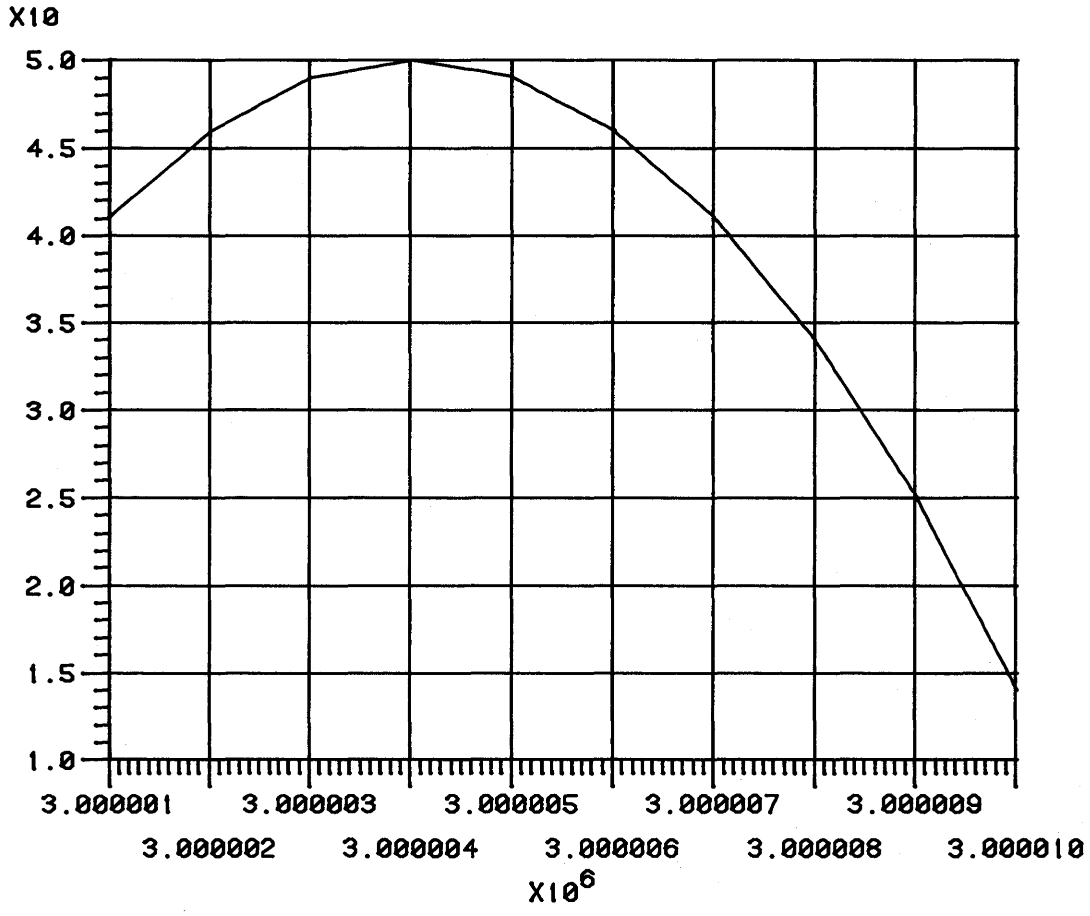
Plots two curves - one against a linear axis and the other against a logarithmic axis.

Figure 1.8



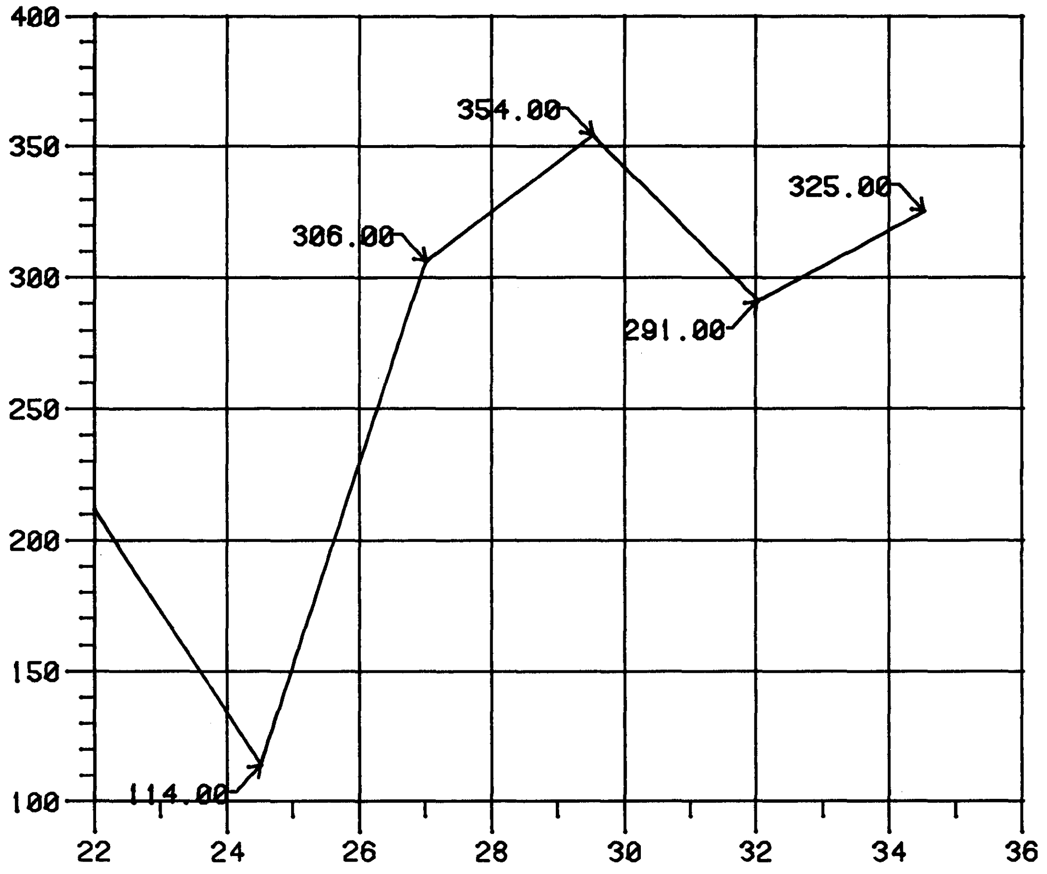
Demonstrates the use of multiple curves with different tic mark values for each.

Figure 1.9



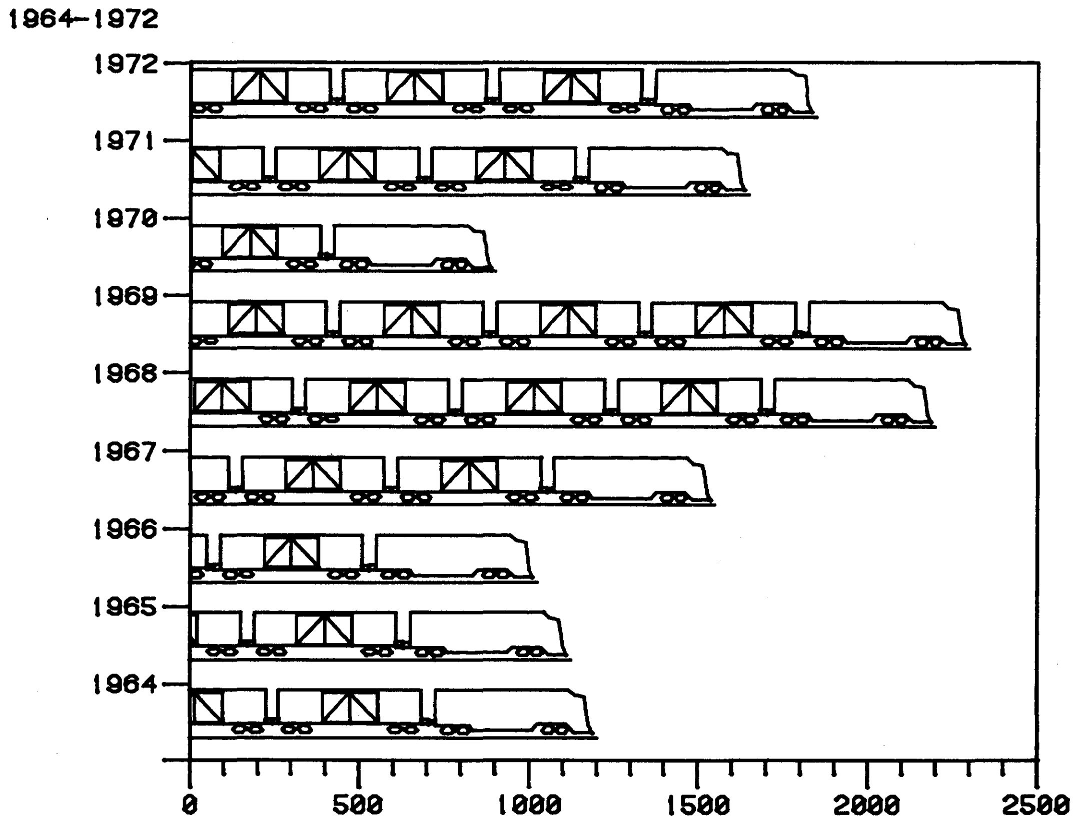
Shows the use of a remote exponent and staggered labels.

Figure 1.10



Shows the use of a user written symbol routine.

Figure 1.11



A bar chart with a user written line routine to draw the trains.

Figure 1.12

SECTION 2

BASIC ROUTINES

To produce graphs that are more detailed and specialized than the simple graphs shown in Section 1, much more information is needed. The following sections include explanations for producing logarithmic, calendar, and bar plots, as well as describe how to add alphanumeric labels.

2.1 Definition of Terms

AG-II Plot Table (also referred to as AG-II COMMON) will be referred to frequently throughout the manual. This is a global table of values, available to all system routines, which determines the various traits of the graph. The table, and all user called subroutines to set it, are included in Section 3. We will be referring to those routines in this section, also, as we describe the various kinds of plots possible.

Note: The general user will never need to allocate space for the Plot Table or be concerned with it directly.

Since we did not specify any graph characteristics in the Section 1 examples, they were set internally and stored in the AG-II Plot Table. These internally set values are called "default" values, since we did have the opportunity to set them but failed to exercise our right.

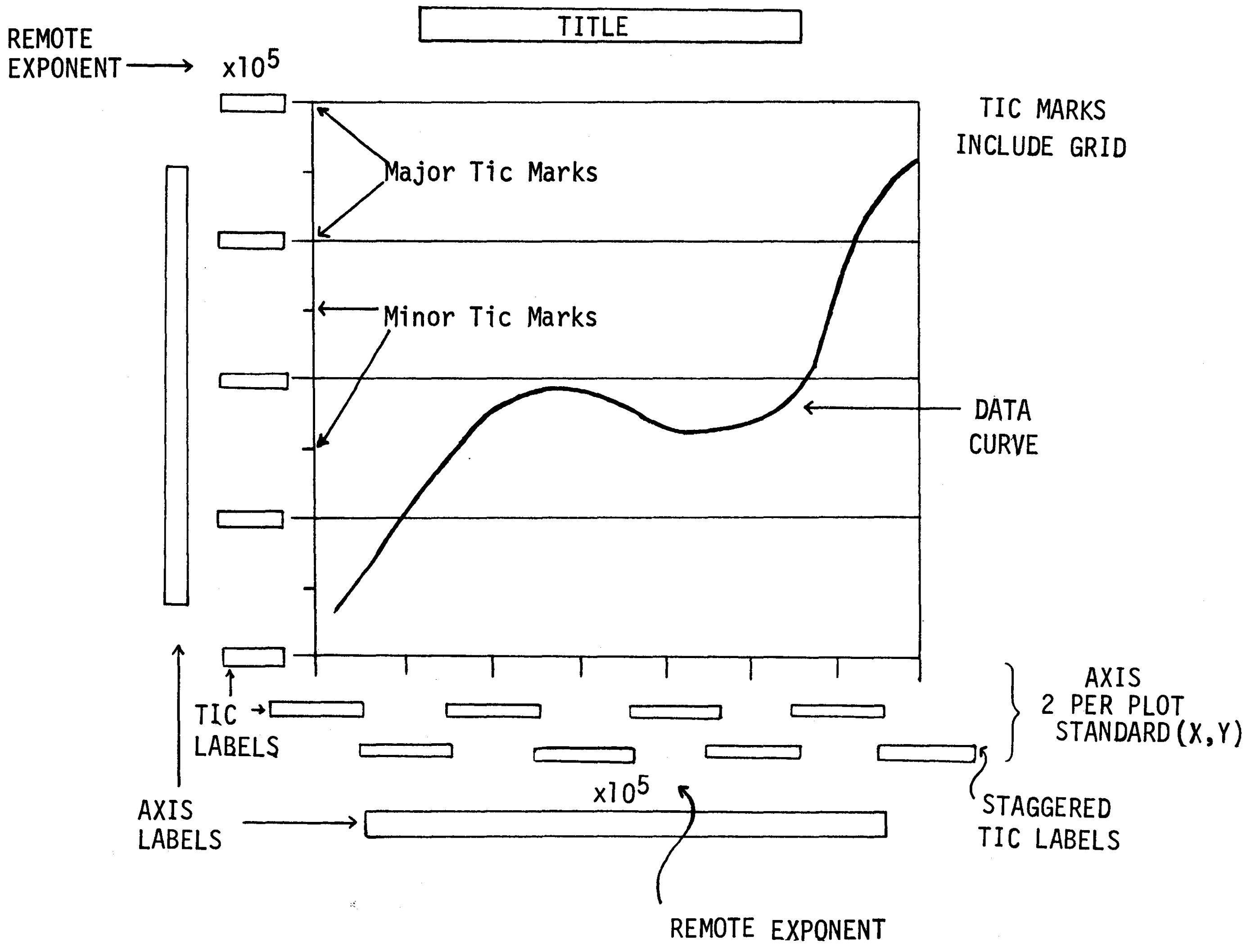
Figure 2.1 helps define the terms which will normally be used in discussing parts of the graph. The window is the portion of the display which contains the curve, not including the tic mark labels,titles, or tic marks which extend outside the axis lines. There are two axes in a standard plot - - the X axis which is the horizontal aspect of the graph and the Y axis which is the vertical aspect of the graph. In general discussion the term "axis" will be used to designate not only the window edge, but also the tic marks and labels which relate to it. X axis data is the data which will be plotted along the X axis, and Y axis data is plotted on the Y axis.

The "remote exponent" is the value by which every tic mark label along the axis is multiplied to obtain the actual value. It is used to eliminate Ø's from the labels, allowing large values in limited label space.

"Tic marks" are divided into two categories - - "major" and "minor". Only major tic marks are labeled; minor tic marks are merely reference points between. Minor tic marks are arbitrarily extended from the window edge half the length of major tic marks (excluding grid lines).

The screen is divided into a grid work of points which are referred to as raster units. These units are used to specify locations on the screen. For example, a screen location of 300,200 is a point 300 raster units across (the X coordinate) and 200 units up (the Y coordinate). The Tektronix 4010 terminal, for example, has a screen display area 1023 raster units by 780.

GRAPH NOMENCLATURE



Tic marks are classified as major or minor. Only major tic marks are labeled.

Figure 2.1
Graph Nomenclature

2.2 Data Structures

In the primer, we used only one standard data form. Two other forms are available for general use, as well as a non-standard long data form. (See Figure 2.2.)

2.2.1 Standard Long Form

The array contains the actual data values. The first element in the array is a positive value specifying the total number of points, followed by the individual data point values, in the order in which they will be plotted.

A short calendar form is provided for calendar data, but should a long form be used, all calendar values must be in UBGC (Universal Business Graphing Calendar) form. See Section 4.3 for details of conversion to UBGC values.

2.2.2 Short Linear Form

The short form may be used if the data is regularly incremented. The first element is -1, indicating the method of expansion (linear). The second element is the number of values to be plotted, the third is the first value to be plotted, and the fourth is the amount by which the value is to be incremented for each data point.

The data statement DATA YARRAY/-1.,6.,1.,2./ would plot the values 1, 3, 5, 7, 9, 11.

2.2.3 Short Calendar Form

The short calendar form is normally used for all calendar data. The first item is -2 which indicates the method of expansion (calendar). The second element is the number of values to be plotted, the third is the number of periods per year, the fourth is a four digit year in which the data begins, and the fifth is the starting period. The "number of periods per year" can only be 1, 4, 12, 13, 52, or 365 (366 cannot be used for a leap year).

The data statement for an array of six months beginning with July 1969 would appear as:

```
DATA XARRAY/-2.,6.,12.,1969.,7./
```

Note: If data is entered in short calendar form, labels will be in calendar form also, without extra Table setting calls.

2.2.4 Non-Standard Long Form *

Data from existing applications programs may not be available with the array length as the first element. In this case, the user may set the number of data points (CNPTS) in the Table by calling NPTS(IVALUE) which indicates that a non-standard array of length equal to IVALUE is being used. (See Section 3.1.5.)

The elements of the array are the actual values to be plotted.

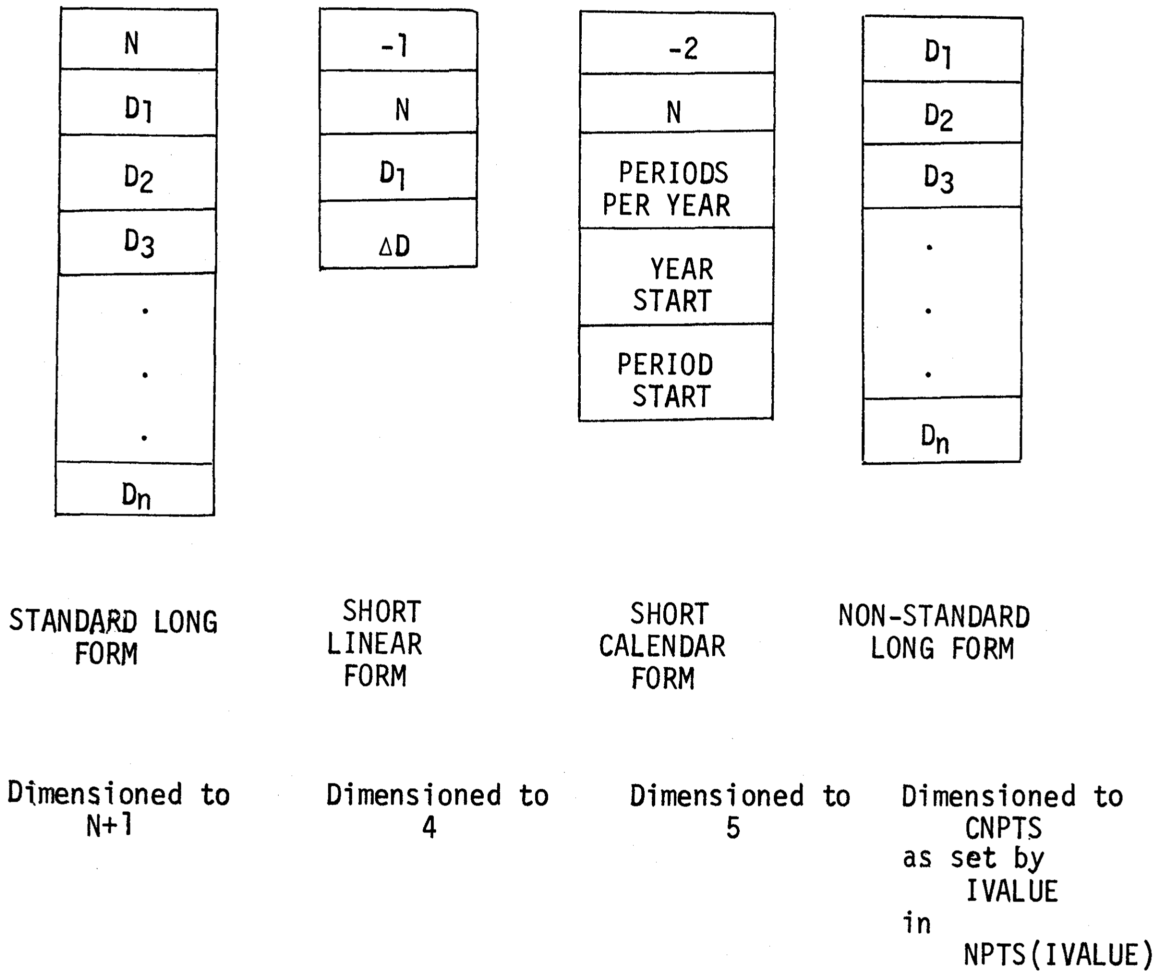


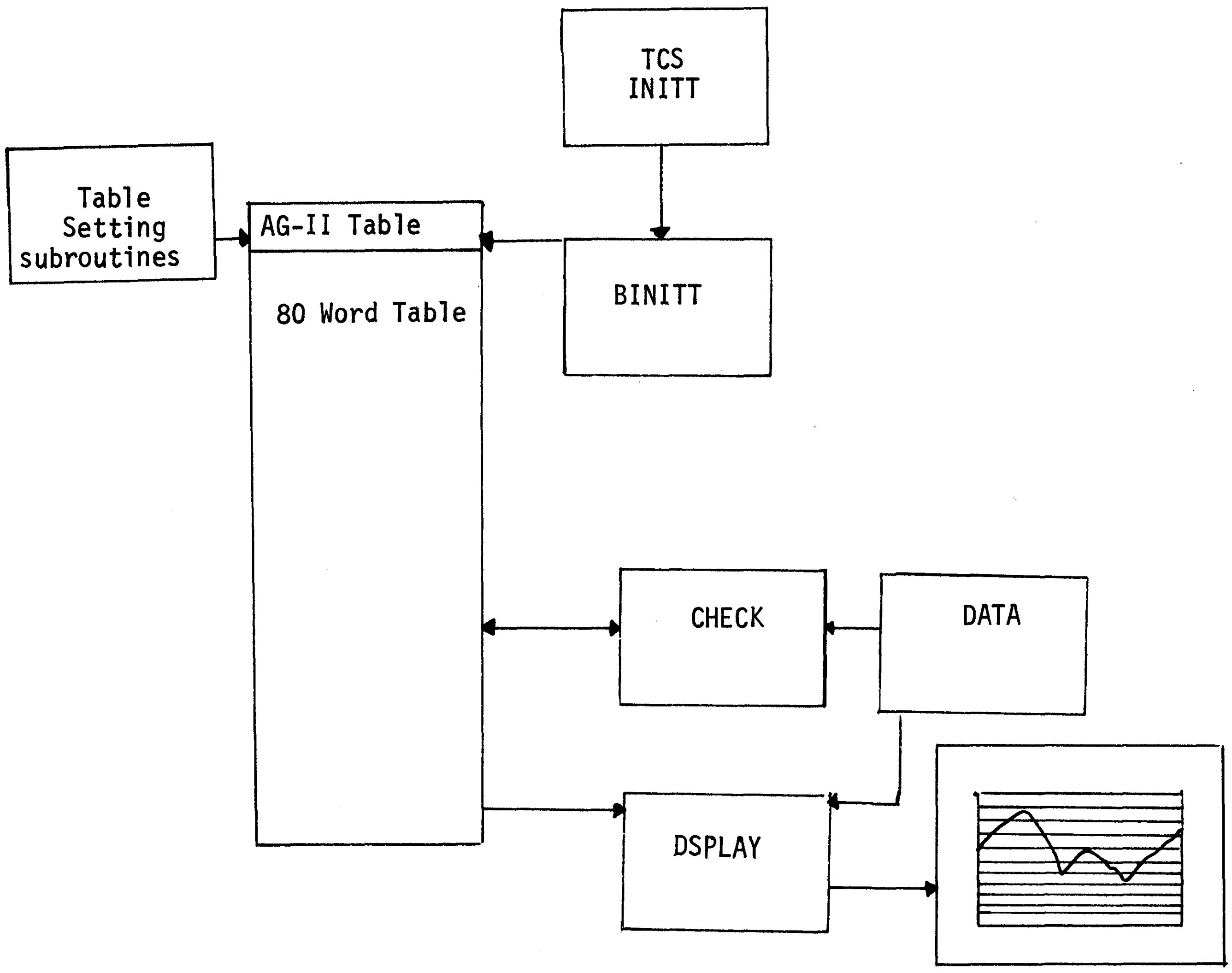
Figure 2.2

DATA STRUCTURE

*Note: A non-standard short form may also be used if the user writes a brief UMNMX subroutine and UPOINT function as described in Section 14 of the System Manual.

2.3 <u>Basic Plot Creating Routines</u>

The following section describes the basic plot creating routines which were used in The Primer.



BASIC GRAPH GENERATION

Figure 2.3

2.3.1 Advanced Graphing II Initialization Subroutine - BINITT

This routine sets the initial values of the Advanced Graphing II COMMON.

Calling Sequence:

```
CALL BINITT
```

Parameters Entered:

None

Description:

BINITT should be called before any other Advanced Graphing II calls are made. It sets the initial values of AG-II COMMON and calls INITT if the user has failed to do so. Note: Since INITT sets the transmission rate, failure to call it directly may result in much longer delays than are really necessary at low transmission speeds.

BINITT may be called at any time to set the table values back to the default values.

In some applications, different initial values may be desired to avoid routine calling of Table setting routines. The program source statements of subroutine BINITT may be altered to perform this function more efficiently than with subroutine calls.

2.3.2 Table Checking Subroutine - CHECK

This routine assures that values have been specified for all necessary words in COMMON.

Calling Sequence:

```
CALL CHECK (XARRAY,YARRAY)
```

Parameters Entered:

XARRAY	is the data to be used for the X axis.
YARRAY	is the data to be used for the Y axis.

This user called routine checks for the data minimums and maximums and the screen length. It then calls OPTIM which checks and figures COMMON values pertaining to spacing of tic marks and label values. CHECK then calls WIDTH and SPREAD which calculate the width and position of the labels based on the room available, and TSET which calculates grid parameters.

This routine does not display the graph, but only prepares all internal values necessary for display.

If BINITT has not been called before the user calls CHECK, CHECK will call it. Note: This is not the recommended calling sequence. An error message will result if BINITT has not been called, and any modifying calls made before CHECK will be cancelled.

Note: CHECK does not alter values which have previously been specified. Repeated use of CHECK without changing COMMON may result in wasted computer time and unchanging plots.

2.3.3 Graph Displaying Subroutine - DSPLAY

This routine displays the complete graph on the screen.

Calling Sequence:

```
CALL DSPLAY (XARRAY,YARRAY)
```

Parameters Entered:

XARRAY	is the array of data for the X axis.
YARRAY	is the array of data for the Y axis.

This subroutine calls all the subroutines necessary to actually display the completed graph. It sets up the window, plots the curve, draws the grid, and labels each set of major tic marks. Subroutines INITT, BINITT, and CHECK must be called prior to calling DSPLAY.

2.3.4 Curve Plotting Subroutine - CPLOT

This routine plots a curve, using the data window location values currently in COMMON. It does not draw the grid or labels, or alter scale factors that determine which data values will be plotted.

Calling Sequence:

```
CALL CPLOT (XARRAY,YARRAY)
```

Parameters Entered:

XARRAY	is the name of the X array.
YARRAY	is the name of the Y array.

This routine is called by subroutine DSPLAY to plot the curve portion of the graph. For multiple curves on the same graph, the user should call DSPLAY for the first curve and CPLOT for all additional curves. If the data values exceed the window boundaries as defined in the Table, the curve will be clipped.

If it is desired to draw several curves on a single scale without clipping, two methods are possible.

1. CALL DLIMY and/or DLIMX (Section 3.2.11) if the data minimums and maximums are known.
2. CALL MNMX (Section 4.1.1) once for each curve before calling CHECK and DSPLAY for the first curve if automatic setting of the minimums and maximums are desired.

See Figure 1.2 for an example of use.

2.4 Changing Standard Graph Form

Any time the user wishes to specify the exact number of tic mark intervals, raw (non-neat) data values for tic marks and labels, length and tic mark width of labels, or a specific remote exponent, etc., he may do so by setting the values with the routines described in Section 3.

Any such table setting routines which affect the initial display are inserted between the calls to BINITT and CHECK.

```
DIMENSION XDATA(4),YDATA(7)
DATA XDATA/-1.,6.,1.,1./
DATA YDATA/6.,211.,114.,3Ø6.,354.,291.,325./
CALL INITT(3Ø)
CALL BINITT
                  .
                  .
                  .
CALL CHECK (XDATA,YDATA)
CALL DSPLAY(XDATA,YDATA)
CALL FINITT(Ø,7ØØ)
STOP
END
```

(Note that the Y axis data has been entered in standard long form and that the X data is in short linear form.)

For example, inserting

```
       CALL XDEN(3)
```

as described in Section 3.2.5 will change the tic mark density factor to 3 which will eliminate minor tic marks.

```
       CALL YNEAT(Ø)
```

specifies raw data tic mark values instead of "neat" values. (See Section 3.2.1.)

Any number of calls may be inserted between the calls to BINITT and CHECK.

When a multiple curve is being plotted, the calls which are to affect only an additional curve should be placed immediately before the call to CPLOT which draws the additional curve. The use of CPLOT is described in Section 1.2.

```
DIMENSION XDATA(4),YDATA1(7),YDATA2(7)
DATA XDATA/-1.,6.,1.,1./
DATA YDATA1/6.,211.,114.,3Ø6.,354.,291.,325./
DATA YDATA2/6.,367.,3ØØ.,179.,238.,32Ø.,21Ø./
CALL INITT(3Ø)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
          .
          .
          .
CALL SYMBL(4)
CALL CPLOT(XDATA,YDATA2)
CALL FINITT(Ø,7ØØ)
STOP
END
```

2.5 Multiple Curves with Different Scales - DINITX, DINITY

A simple method of plotting a second curve was described in Section 1.2. However, if the second curve has data values which extend outside the first data limits, the curve will be clipped at the window edge. See example 2.4.

To prevent this from occurring,the user may make all the calls to display the first curve (including BINITT,CHECK, and DSPLAY), followed by calls to DINITX or DINITY, an axis location routine, CHECK, and DSPLAY for the additional curve. An axis location change is necessary to prevent the new label values from being printed over the old ones. The routines you might choose to use are XLOC or YLOC which offset the axes from the normal left hand and lower window boundaries, and XLOCTP or YLOCRT which move the axes to the opposite side of the window.

If the entire table were reinitialized with BINITT, the tic mark locations would also be recomputed. DINITX or DINITY will leave the tic mark locations the same.

The program might appear as shown in example 2.5.

Subroutine DINITX

This subroutine reinitializes the label values of the X axis in the AG-II Table, allowing the drawing of an additional curve with new label values.

Calling sequence:

```
CALL DINITX
```

Parameters Entered:

None

Description:

All values related to labeling the X axis are reinitialized to zero, so that new values can be computed for the display of an additional curve.

Subroutine DINITY

This subroutine reinitializes the label values of the Y axis in the AG-II Table. This allows the drawing of an additional curve with new label values.

Calling Sequence:

```
CALL DINITY
```

Parameters entered:

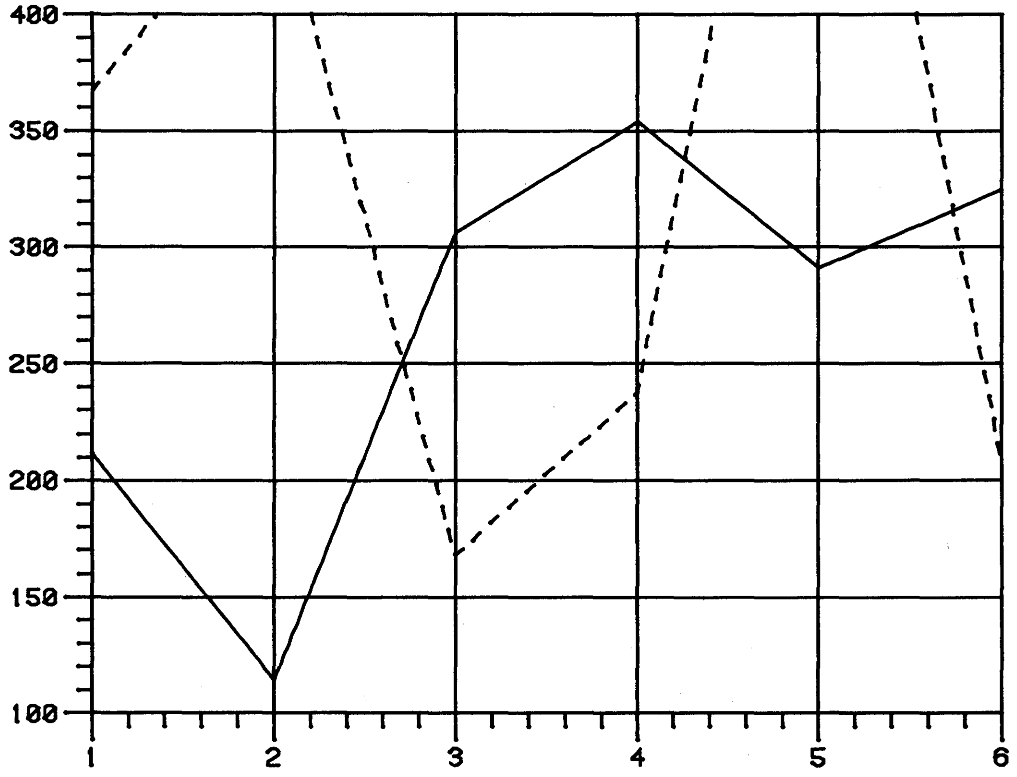
None

Description:

All values related to labeling the Y axis are reinitialized to zero, so that new values can be computed for the display of an additional curve.

```
DIMENSION XDATA(7),YDATA1(7),YDATA2(7)
DATA XDATA/-1.,6.,1.,1./
DATA YDATA1/6.,211.,114.,306.,354.,291.,325./
DATA YDATA2/6.,367.,458.,168.,238.,620.,210./
CALL INITT(30)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL LINE(34)
CALL CPLOT(XDATA,YDATA2)
CALL FINITT(0,700)
END
```

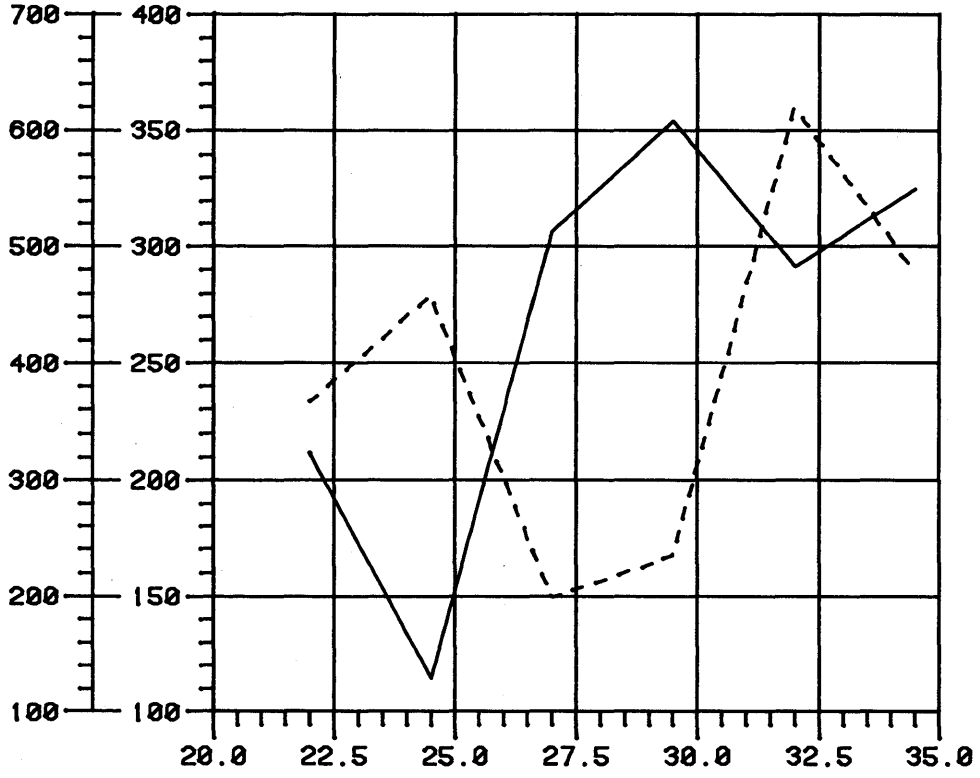
NOTE: The call to TINPUT is used as a pause throughout this manual so the user can study the graph or make a hard copy.



Clipped Data

Figure 2.4

```
      DIMENSION XARRAY(7),YARRAY(7),ZARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      DATA ZARRAY/6.,366.,458.,198.,235.,620.,478./
      CALL INITT(30)
      CALL BINITT
      CALL SLIMX(200,800)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
C * SECOND CURVE,HAS DIFFERENT Y RANGE.  REINITIALIZE.
      CALL DINITY
      CALL YLOC(-100)
      CALL LINE(34)
      CALL YFRM(4)
      CALL XLAB(0)
      CALL XFRM(0)
      CALL CHECK(XARRAY,ZARRAY)
      CALL DSPLAY(XARRAY,ZARRAY)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
```



Dual Scales

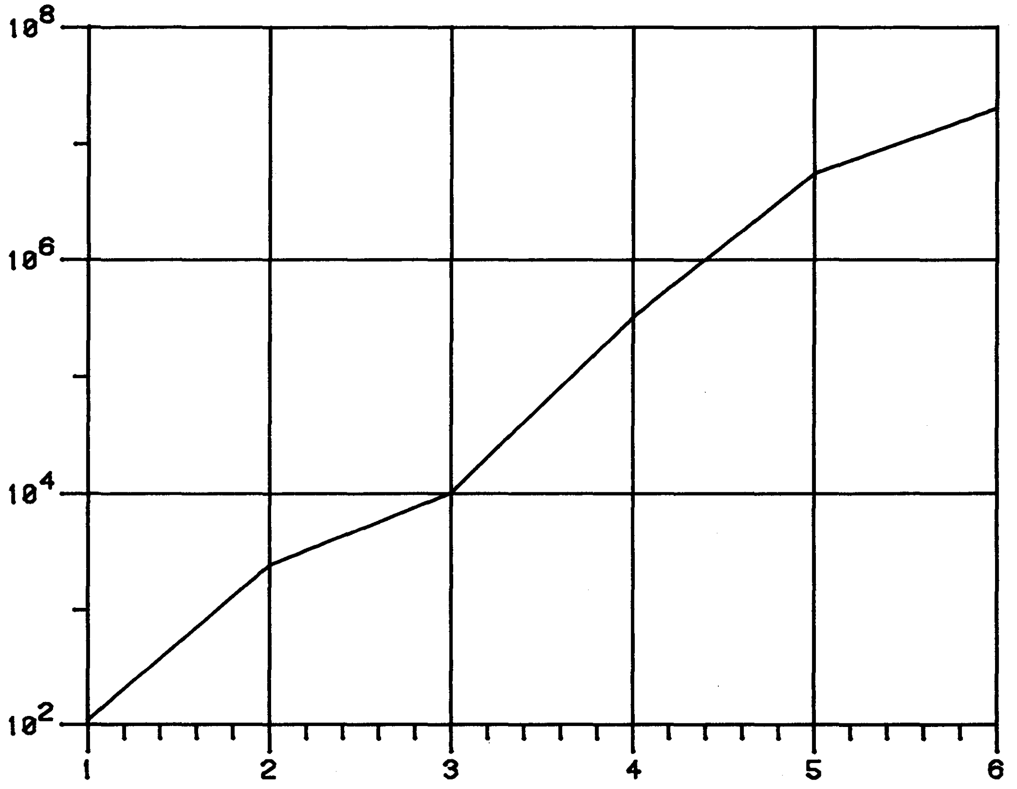
Figure 2.5

2.6 Logarithmic Plots

Logarithmic plots may be created by entering data in the usual manner and specifying YTYPE or ZTYPE of 2. XTYPE and YTYPE are described in Section 3.2.13. Either one or both axis may be made logarithmic on the same plot. The data and the axis limits set by SLIMX, SLIMY, must be greater than 0. Otherwise, the associated logarithm is not defined.

The sample results are shown.

```
DIMENSION XDATA(4),YDATA1(7)
DATA XDATA/-1.,6.,1.,1./
DATA YDATA1/6.,110.,2400.,10000.,320000.,5600000.,20000000./
CALL INITT (30)
CALL BINITT
CALL YTYPE(2)
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL FINITT(0,700)
END
```



Logarithmic Plot
Figure 2.6

2.7 Calendar Plotting

If data is entered in the short calendar form, appropriate label type will also be used. An appropriate data statement for monthly data would be

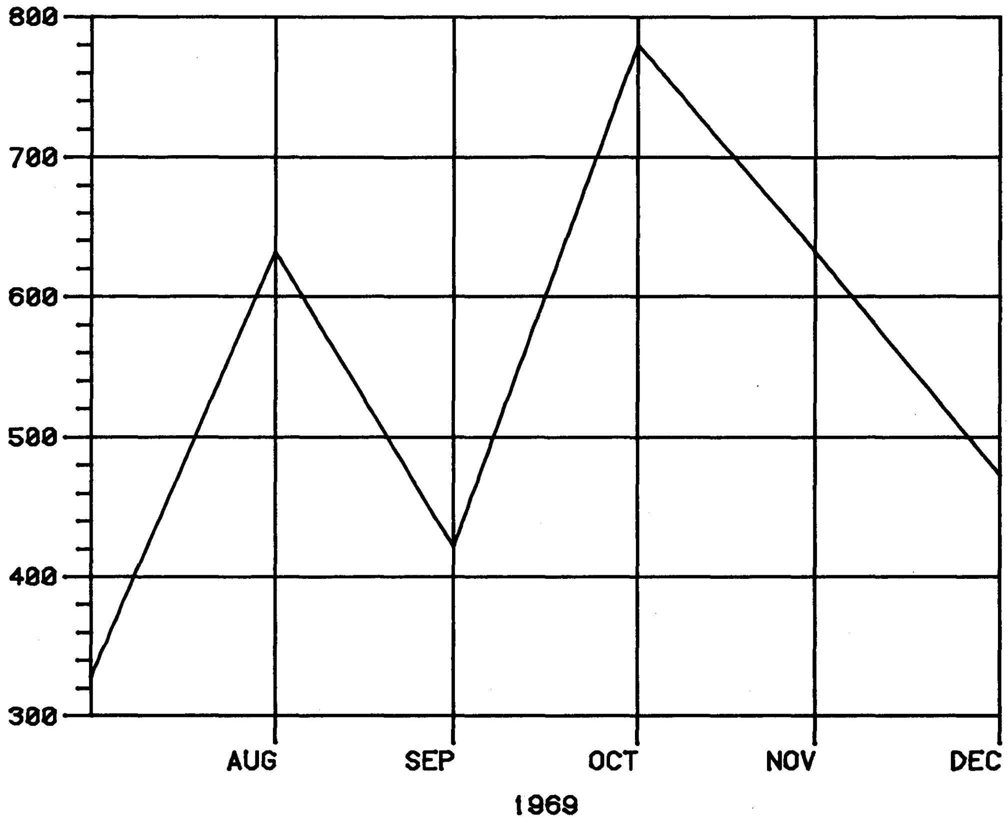
```
DATA XDATA/-2.,6.,12.,1969.,7./
```

Where -2 specifies calendar form, 6 is the number of points to be plotted, 12 is the number of periods per year, and 7 is the beginning period in the beginning year. A plot will appear as shown in Figure 2.8.

The label type and data type have been set internally to match the calendar data. However, if you want to plot daily data against a weekly grid, the data would be entered as daily, and the user would need to call XLAB or YLAB (depending on which axis the calendar data is plotted) to set the label type to be weekly. (See Section 3.2.4 for a description of XLAB and YLAB.)

If daily data is being plotted, the remote label will be the date of the first day of the grid. The remote labels for all other types of calendar data will be the year or span of years plotted (e.g., 1969 or 1969-1971).

```
C****************************************************************
      DIMENSION XDATA(5),YDATA(7)
      DATA XDATA/-2.,6.,12.,1969.,7./
      DATA YDATA/6.,328.,632.,421.,780.,632.,472./
      CALL INITT(120)
      CALL BINITT
C * MAXIMUM LENGTH OF XLABELS WILL BE 3
      CALL XWDTH(3)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
C****************************************************************
```



Calendar Plot

Figure 2.7

2.8 Bar Charts - HBARST, VBARST

Bar Charts may be specified by using one of two routines - HBARST for horizontal bars or VBARST for vertical bars. The same results could be obtained by making individual calls to LINE to specify bar chart, XFRM or YFRM to suppress the grid lines parallel to the bars, SYMBL to specify the type of shading lines in the bar, SIZEL to specify the width of the bar, and SIZES to specify the distance between the shading lines. These subroutines are described in detail in Section 3.

Subroutines HBARST and VBARST

Calling Sequences:

```
        CALL HBARST (ISHADE,IWBAR,IDBAR)
                     or
        CALL VBARST (ISHADE,IWBAR,IDBAR)
```

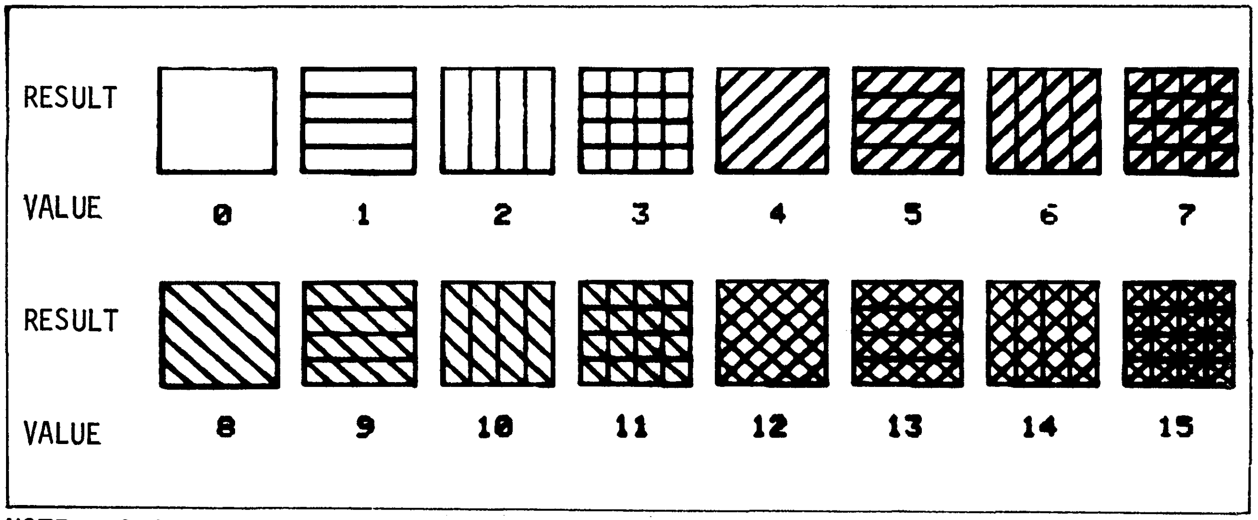
Parameters Entered:

ISHADE	is the integer value of the type of shading to fill in the bar. See chart below.
IWBAR	is the width of the bar in raster units. The width of the bar must be greater than 1, but Ø will result in a default width of 40 raster units.
IDBAR	is the distance between the shading lines in raster units. The distance must be greater than 2; however, if a Ø is entered, a default width of 20 raster units will result.

Description:

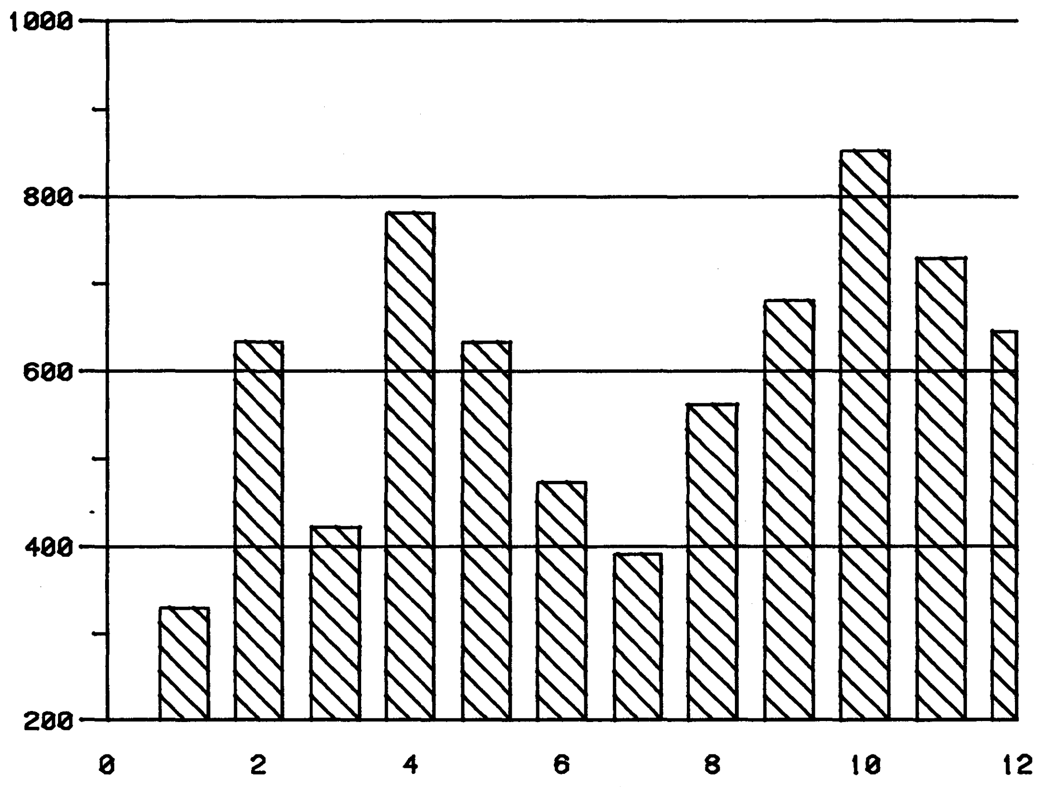
The types of shading are as follow:

BAR CHART SHADING



NOTE: Grid lines will be prevented from appearing parallel to these bars.

```
C******************************************************************
      DIMENSION XDATA(4),YDATA(13)
      DATA XDATA/-1.,12.,1.,1./
      DATA YDATA/12.,328.,632.,421.,780.,632.,472.,390.,562.,
     & 680.,852.,729.,645./
      CALL INITT(120)
      CALL BINITT
      CALL VBARST(8,0,0)
      CALL XFRM(1)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
C******************************************************************
```

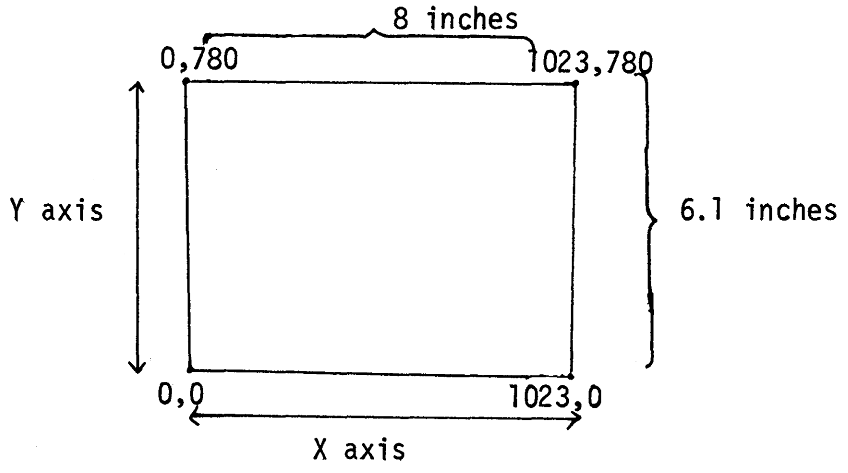


Bar Chart

Figure 2.8

2.9 Screen Window Location

Three different Table Setting subroutines are available to set screen window position, depending upon the manner in which you would prefer to enter the information. Any position defined is the position of the data window itself, not including the labels and other internal information, so it is important to leave room for these "extras" when defining the window location. On a Tektronix 4010 terminal there are 1024 visible addressable units on the X axis and 781 visible addressable units on the Y axis with 128 units to the inch. These raster unit designations are used to show screen location. The visible screen appears as:



2.9.1 Easy Method

The simplest method of entering the location is to use the routine PLACE which will position the window at predefined positions on the screen. The diagram in Section 3.2.12 shows the positions and the literals which may be used as parameters to designate them. The statements:

```
DATA IUL4/3HUL4/
CALL PLACE(IUL4)
```

would position the window in the upper left hand quarter of the terminal screen. See Section 3.2.12 for complete description.

2.9.2 <u>Screen Coordinate Method</u>

The limits of the X axis or the Y axis may be set directly in screen units by calling SLIMX or SLIMY in the form

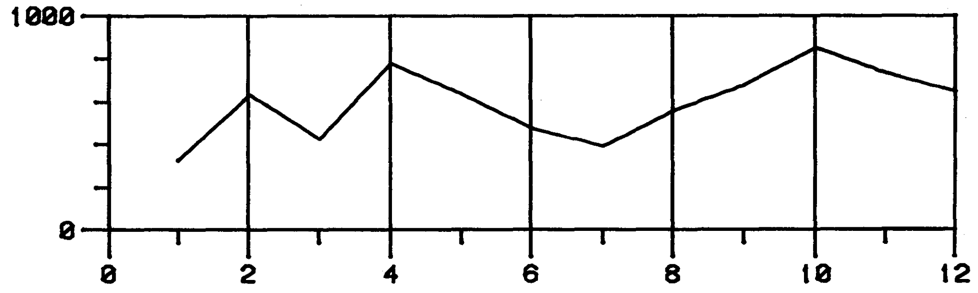
```
CALL SLIMX(3ØØ,8ØØ)
```

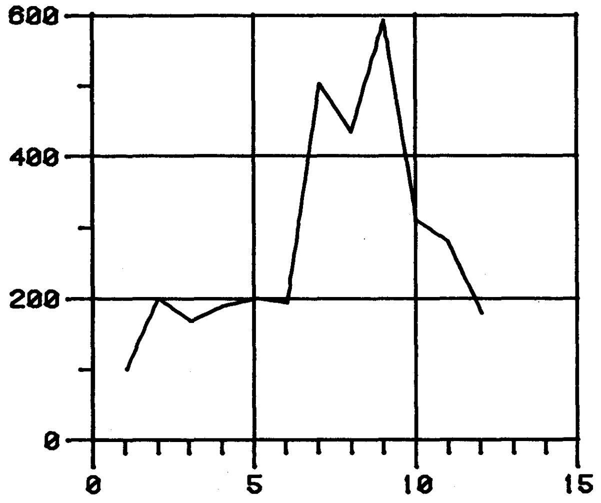
where 3ØØ is the X axis screen minimum in raster units, and 8ØØ is the X axis screen maximum. This would limit the width of the window. SLIMY would be called to set the height of the window in a similar manner.

2.10 <u>Multiple Graphs on the Same Page</u>

To display two or more separate graphs on the same page it is necessary to call BINITT, location setting routines, CHECK, and DSPLAY, in that order, once for each graph. An example, with two methods of positioning the window, follows:

```
C*************************************************************
      DIMENSION XDATA(4),YDATA(13),ZDATA(13)
      DATA XDATA/-1.,12.,1.,1./
      DATA YDATA/12.,328.,632.,421.,780.,632.,472.,390.,562.,
     & 680.,852.,729.,645./
      DATA ZDATA/12.,100.,199.,166.,188.,199.,192.,503.,
     &434.,592.,309.,277.,177./
C * VARIABLE UPH IS A 3 ELEMENT LITERAL
      DATA IUPH/3HUPH/
      CALL INITT(120)
      CALL BINITT
C * DISPLAY THE GRAPH IN THE UPPER HALF OF THE SCREEN AREA
      CALL PLACE(IUPH)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
C * CHANGE SCREEN LIMITS TO DISPLAY THE GRAPH
C * IN BOTTOM HALF OF SCREEN AREA
      CALL BINITT
      CALL SLIMX(400,800)
      CALL SLIMY(50,400)
      CALL CHECK(XDATA,ZDATA)
      CALL  DSPLAY(XDATA,ZDATA)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
C*************************************************************
```





Multiple Graphs

Figure 2.10

2.11 Additional Labels and Titles

Additional labels may be added to the display by establishing an array of the ASCII equivalents of the characters to be displayed, moving invisibly to the beginning point and calling either HLABEL (horizontal labeling routine) or VLABEL (vertical labeling routine) to display the label.

```
DATA ISTRIN/84,73,84,76,69/
```

Would establish a string which contains the ASCII equivalents of "TITLE". Appendix C charts the equivalents.

TCS subroutine MOVABS (see your PLOT-10/Terminal Control System User's Manual) provides one method of moving to the beginning location of the label.

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

where IX is the X screen coordinate and IY is the Y screen coordinate of the lower left corner of the first character.

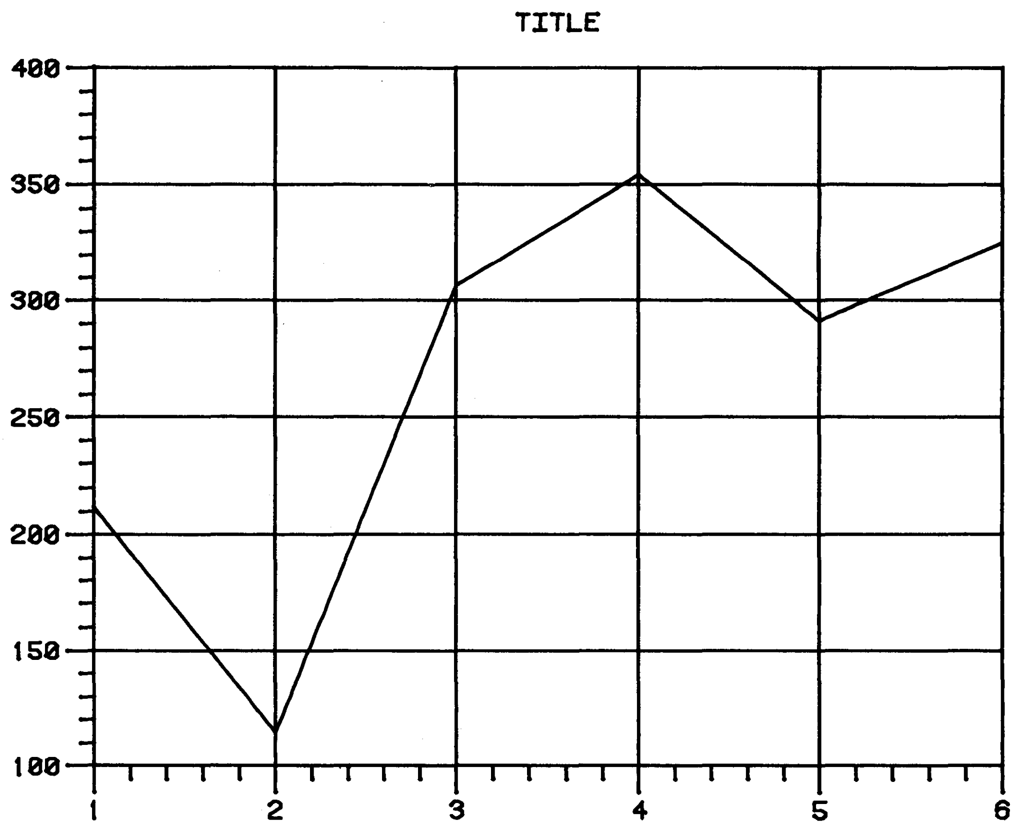
A call to HLABEL or VLABEL immediately following the call to MOVABS will display the label. For example,

```
       CALL HLABEL(5,ISTRIN)
```

would display the five characters contained in the string ISTRIN horizontally. VLABEL is used in the same manner for vertical labels with the location being the upper left corner of the first character.

Subroutine NOTATE may also be used to display horizontal labels, as described in Section 2.13.2.

```
DIMENSION XDATA(4),YDATA1(7),ISTRIN(5)
DATA XDATA/-1.,6.,1.,1./
DATA YDATA1/6.,211.,114.,306.,354.,291.,325./
DATA ISTRIN/84,73,84,76,69/
CALL INITT(30)
CALL BINITT
CALL CHECK(XDATA,YDATA1)
CALL DSPLAY(XDATA,YDATA1)
CALL MOVABS(500,730)
CALL HLABEL(5,ISTRIN)
CALL FINITT(0,700)
END
```



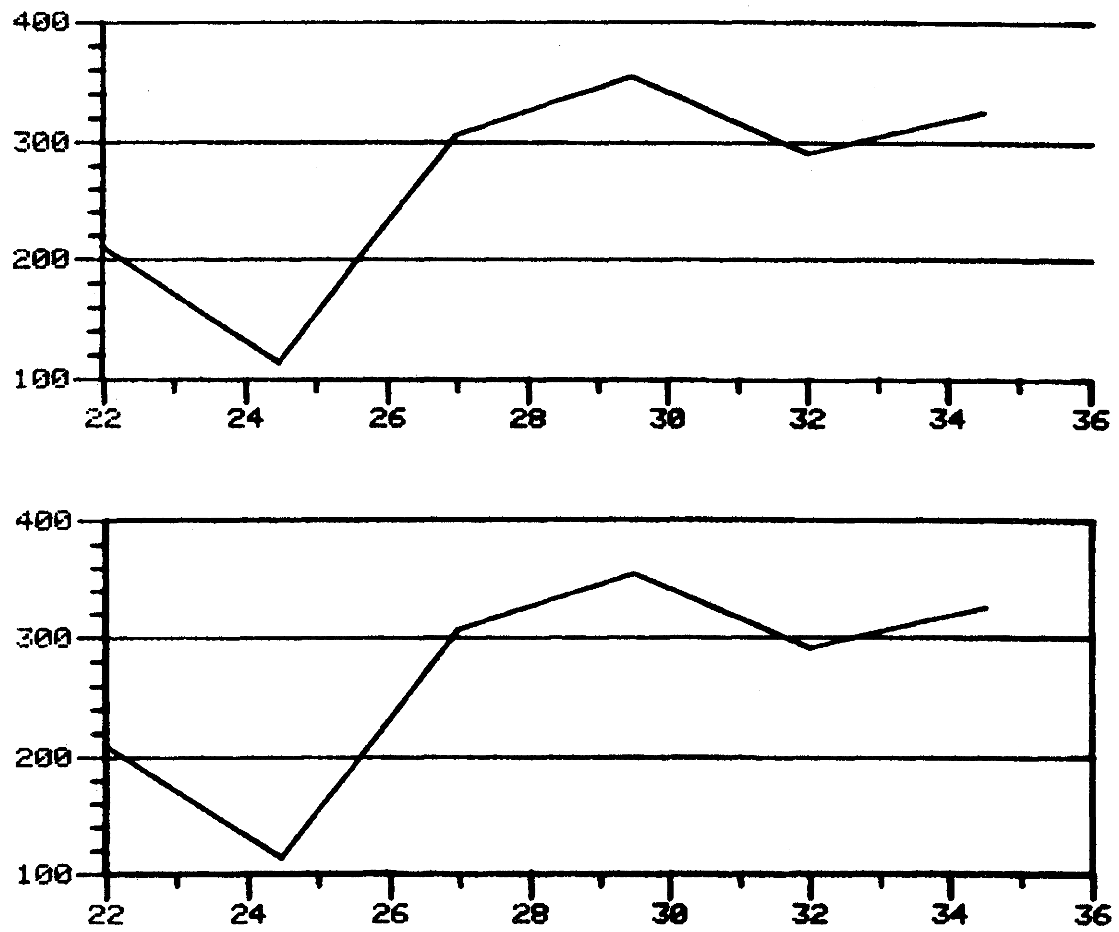
Labels and Titles

Figure 2.11

2.12 Framing the Data Window

With a standard default graph, the grid lines form a frame for the data window. However, if short major tic marks have been specified to omit grid lines (CALL XFRM(IVALUE) with IVALUE of 1, 2, 3, or 4), two sides of the window opposite the axes will not be framed. To create a frame,the user may call subroutine FRAME after DSPLAY.

```
C*************************************************************
      DIMENSION XARRAY(7),YARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(120)
C * DRAW A GRAPH THAT NEEDS FRAMING
      CALL BINITT
      CALL SLIMY(200,500)
      CALL XFRM(2)
      CALL YFRM(5)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(K)
C * DRAW A FRAME AROUND THE GRAPH
      CALL FRAME
      CALL TINPUT(K)
      CALL FINITT(0,700)
      END
C*************************************************************
```



Screen Framing

Figure 2.12

2.13 Frequently Used Routines

The routines which the user might call most frequently are described in detail in the following sections. The usage of most of these routines has been discussed in the previous sections.

2.13.1 Window Framing Subroutine - FRAME

This routine draws a frame around the window.

Calling Sequence:

```
CALL FRAME
```

Parameters Entered:

None

Description:

If short tic marks have been specified (without grid lines), the sides of the data window opposite the axes will not be framed. In this case, calling FRAME will add these lines.

Figure 2.12 shows the use of subroutine FRAME.

2.13.2 Horizontal Labeling Subroutines- HLABEL, HSTRIN, NOTATE

The two routines HLABEL*and HSTRIN print the label string on the screen in the position designated . They may be used interchangeably but use different methods for entering the length of the string.

The alphanumeric cursor must be moved to the label beginning point using a TCS command before calling either of these two routines. Subroutine NOTATE moves the cursor itself. NOTATE also allows subscripts and superscripts by means of special flags.

* For systems which use buffered I/O, the TCS, Level 1, subroutine ANSTR should be used in place of HLABEL. The parameters of ANSTR are the same as those of HLABEL.

1. Calling Sequence:

   CALL HLABEL (LENGTH,ISTRIN)

   Parameters Entered:

LENGTH	is the length of the label character string.
ISTRIN	is the label character string in ASCII decimal equivalents (numbers Ø through 127 representing alphanumeric characters).

   Figures 2.10 and 2.13.3 demonstrate the use of HLABEL.

2. Calling Sequence:

   CALL HSTRIN (ISTRIN)

   Parameters Entered:

ISTRIN	is the array of characters in ASCII decimal equivalents (numbers Ø through 127 representing alphanumeric characters.) The first element in ISTRIN is the number of characters in the alphanumeric string, i.e. length of ISTRIN less 1. Note: This is similar in form to the standard long data form.

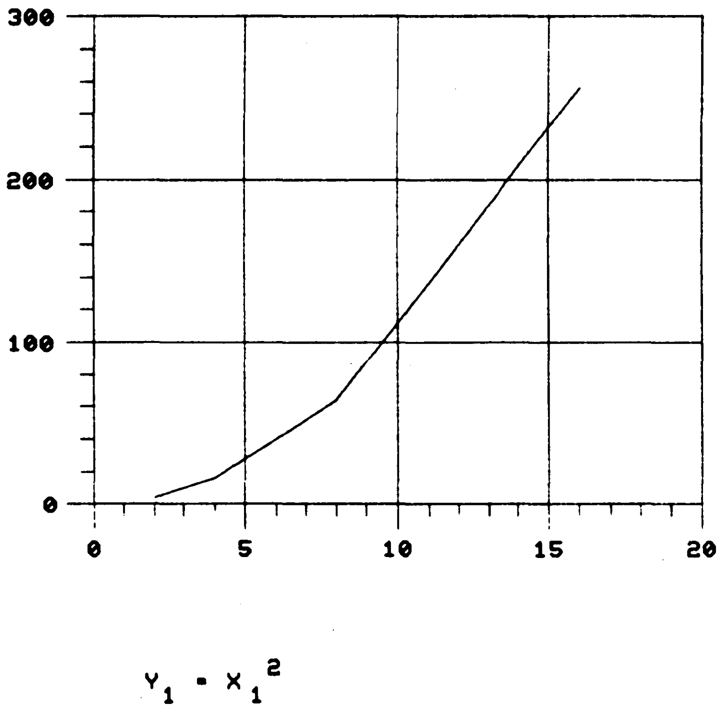
3. Calling Sequence:

   CALL NOTATE (IX,IY,LENCHR,ISTRIN)

   Parameters Entered:

IX,IY	designate the beginning point of the label in screen coordinates.
LENCHR	is the length of the character string for the given label, including the number of -1 and -2 tags.
ISTRIN	is the string of characters in integers from Ø to 127 (ASCII decimal equivalents), plus any -1 and -2 tags. A -1 tag in the character string causes the succeeding characters to be output one-half line upward. A -2 tag in the character string causes succeeding characters to be output one-half line downward. These offsets remain in effect until another tag is encountered. Thus, a -1 tag will cancel a previous -2 tag, and vice-versa.

```
      DIMENSION X1(5),Y1(5)
      DIMENSION ISTRIN(13)
C * CREATE A GRAPH OF Y = X * X
      DATA X1/4.,2.,4.,8.,16./
      DATA Y1/4.,4.,16.,64.,256./
      DATA ISTRIN/89,-2,49,-1,32,61,32,88,-2,49,-1,-1,50/
C * CHANGE THE SCREEN WINDOW SIZE
      CALL INITT(240)
      CALL BINITT
      CALL SLIMX(359,850)
      CALL SLIMY(250,650)
C * PRINT THE TITLE OF THE DISPLAY
      CALL NOTATE(400,100,13,ISTRIN)
      CALL CHECK(X1,Y1)
      CALL DSPLAY(X1,Y1)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



2.13.3 Vertical Labeling Subroutine - VLABEL, VSTRIN

These routines print a label string on the screen vertically. Note: The alphanumeric cursor must be moved to the label beginning point before calling VLABEL. TCS commands provide the moving capability. The top left of the first output character is at the current beam position.

1. Calling Sequence:

   CALL VLABEL (LENGTH,ISTRIN)

   Parameters Entered:

LENGTH	is the length of the label character string.
ISTRIN	is the label character string in ASCII equivalents.

   Figure 2.14 demonstrates the use of VLABEL.

2. Calling Sequence:

   CALL VSTRIN (ISTRIN)

   Parameters Entered:

ISTRIN	is the array including in the first position the number of characters in the label (the length of the array minus one), followed by the characters in ASCII decimal equivalents

```
      DIMENSION ISTRIN(22)
      DATA ISTRIN/48,32,49,32,50,32,51,32,52,32,53,32,54,32,55,32,
     &56,32,57,32,58,32/
      CALL INITT(30)
      CALL BINITT
C * LABEL IN THE HORIZONTAL DIRECTION
      CALL HLABEL(22,ISTRIN)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```

```
0 1 2 3 4 5 6 7 8 9 :
```

```
      DIMENSION ISTRIN(22)
      DATA ISTRIN/48,32,49,32,50,32,51,32,52,32,53,32,54,32,55,32,
     &56,32,57,32,58,32/
      CALL INITT(30)
      CALL BINITT
C * LABEL IN THE VERTICAL DIRECTION
      CALL VLABEL(22,ISTRIN)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```

```
0

1

2

3

4

5

6

7

8

9

:
```

Horizontal and Vertical Labels

Figure 2.13.3

2.13.4 Label Justification Subroutine - JUSTER

This subroutine right, left, or center justifies labels or other strings of characters to be displayed.

Calling Sequence:

```
CALL JUSTER (LENGTH,ISTRING,KEY,IFIL,LEN,IOFFSET)
```

Parameters Entered:

LENGTH	is the total length of the character string, including fill characters.
ISTRING	is the array of characters (in ASCII equivalents).
KEY POSITION	is an integer that designates if the string is to be right, left, or center justified. Values entered are + right justified 0 center justified - left justified
IFILL CHARACTER	is the character in ASCII code used as filler (usually a space [32]). JUSTER deletes fill characters occurring at the start of the ISTRING.

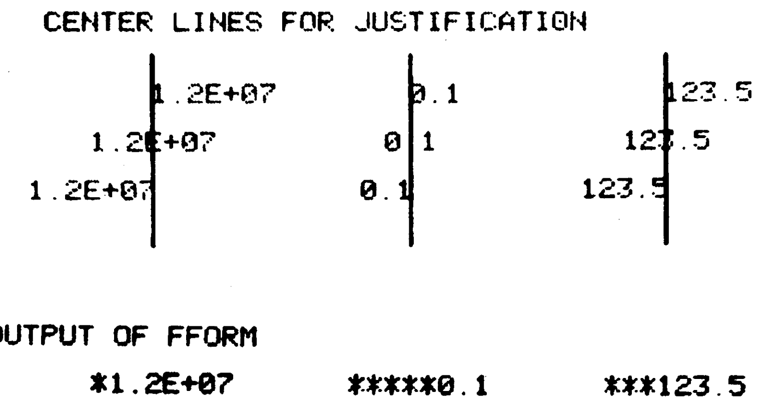
Parameters Returned:

LEN	is an output parameter giving the length of the character string without fill characters.
IOFFSET	is the distance in raster units between the reference point from which the string is to be justified and the actual starting point for the justified string.

JUSTER provides the information necessary for a call to NOTATE, (Section 2.13.2) which displays labels.

Note: If AG-II routines were used to create the strings to be justified, the specified fill character and width must be the same as those specified at creation time.

```
C * THIS ROUTINE DEMONSTRATES JUSTER
      DIMENSION A(3),IARRAY(8),LCENTR(30),LFFORM(15)
      DATA A/12345678.,.12345678,123.45678/
C * EXPLANATION LABEL:  CENTER LINES FOR JUSTIFICATION
      DATA LCENTR/67,69,78,84,69,82,32,76,73,78,69,83,32,70,79,82,32,74,
     &85,83,84,73,70,73,67,65,84,73,79,78/
C * FFORM OUTPUT LABEL
      DATA LFFORM/79,85,84,80,85,84,32,79,70,32,70,70,79,82,77/
      LOCREL=0
      CALL INITT(120)
C * OUTPUT EXPLANATION LABEL
      CALL NOTATE(120,760,30,LCENTR)
C * PRINT FFORM OUTPUT LABEL
      CALL NOTATE(120,340,15,LFFORM)
C * MOVE TO (200,740),(400,740), AND (600,740).  DRAW VERTICAL LINES.
      DO 50 K=1,3
      CALL MOVABS(K*200,740)
50    CALL DRWREL(0,-160)
      CALL ANMODE
C * OUTPUT 9 JUSTIFIED NUMBERS
      DO 200 J=1,3
      DO 100 I=1,3
C * FORMAT ENTRIES OF A, USING AG-II NUMERIC ROUTINES
C * FILL CHARACTER IS *
      CALL FFORM(A(I),8,1,IARRAY,42)
C * PRINT OUTPUT OF FFORM NEAR THE BOTTOM OF THE PAGE
      CALL NOTATE(200*I,300,8,IARRAY)
C * KEY TAKES VALUES -1,0,AND 1, SO LEFT, CENTER, AND RIGHT JUSTIFY
      KEY=J-2
C * WIDTH AND FILL CHARACTER INPUT TO JUSTER MUST BE THE SAME
C * AS IN FFORM
C * IOFSET IS AN INCREMENT TO BE ADDED TO THE X COORDINATE
C * OF THE OUTPUT POSITION
C * LEN IS THE LENGTH OF THE STRING RETURNED BY JUSTER, WITH
C * FILL CHARACTERS DELETED
      CALL JUSTER(8,IARRAY,KEY,42,LEN,IOFSET)
C * OUTPUT THE NUMBER
C * IF THE LAST ENTRY OF NOTATE IS ONE CHARACTER OF A LARGER ARRAY
C * THAT CHARACTER AND ALL THAT FOLLOW ARE OUTPUT
100   CALL NOTATE(I*200+IOFSET,700-LOCREL,LEN,IARRAY(8-LEN+1))
C * INCREASE THE ADJUSTMENT TO THE Y LOCATION OF OUTPUT
200   LOCREL=LOCREL+40
      CALL FINITT(0,150)
      END
```



```
CENTER LINES FOR JUSTIFICATION

             |1.2E+07       |0.1           |123.5
        1.2E+07           0 1           123.5
    1.2E+07|             0.1|         123.5|


OUTPUT OF FFORM

       *1.2E+07     *****0.1     ***123.5
```

Use of JUSTER
Figure 2.13.4

SECTION 3

DESCRIPTION OF AG-II TABLE CONTROLLING ROUTINES

Description of AG-II Table Controlling Routines

The Advanced Graphing II Package centers around the use of a COMMON Table which contains all the variables necessary to display a graph complete with axes, labelled tic marks, grid, and remote exponent where necessary. It also includes variables to specify the window position, the type of line (solid, dashed, etc.), data point symbols, and increments between data points or symbols. These variables are stored as full word real values, which are initialized by BINITT to reasonable default values. The COMMON setting routines included in this section may be called at any time after initialization to change the values in COMMON. The values are entered as integers in the COMMON setting routines (except DLIMX, DLIMY, SIZEL, SIZES, and INFIN),and are then converted to real values for storage. If the user has specific requirements for the appearance of the graph, individual routines are available to set the variables, with the exception of five variables concerning label length and position; these are set internally to prevent labels from being overcrowded. (See Table I on page 58-60). The values of the COMMON Table variables are available at any time after initialization for the use of the internal routines which set up and display the actual graph.

Section 4.4 contains the general COMMON access and storage routines. One subroutine, COMSET, may be used to set any variable in COMMON, and function COMGET can call forth the current value of any variable in COMMON. The use of these subroutines are described in detail in Section 4.4.

TABLE I

Advanced Graphing II Common Variables

Location	Variable	Description	Initial Value	Subroutines Used To Set Value
		SECTION I. DATA CURVE VARIABLES		
Ø	CLINE	Type of line used in plots	Ø.	LINE
1	CSYMBL	Symbol used for point plots	Ø.	SYMBL
2	CSTEPS	Increment between symbols	1.	STEPS
3	CINFIN	Infinity	1.E30	INFIN
4	CNPTS	Number of points for non-standard array	Ø.	NPTS
5	CSTEPL	Increment between points in lines	1.	STEPL
6	CNUMBR	Actual number of points in this plot.	Ø.	Set Internally
7	CSIZES	Symbol size - Shading density on bars	1.	SIZES
8	CSIZEL	Line size - Bar width	1.	SIZEL
		SECTIONS II AND III. AXIS VARIABLES[1]		
Ø	CXNEAT CYNEAT	Neat tic mark flag	1.(ON)	XNEAT YNEAT
1	CXZERO CYZERO	Zero suppression flag	1.(ON)	XZERO YZERO
2	CXLOC CYLOC	Location of axis	Ø.	XLOC XLOCTP YLOC YLOCRT
3	CXLAB CYLAB	Type of labels	1.	XLAB YLAB
4	CXDEN CYDEN	Density of tic marks	8.	XDEN YDEN
5	CXTICS CYTICS	Number of major tic mark intervals	Ø.	XTICS YTICS
6	CXLEN CYLEN	Length of major tic marks	Vertical height of current character size	XLEN YLEN

(continued)

[1]Sections II and III (X Axis Variables and Y Axis Variables) have been combined in this chart for ease of reference.

TABLE I (continued)

Advanced Graphing II Common Variables

Location	Variable	Description	Initial Value	Subroutines Used To Set Value
SECTION II AND III. AXIS VARIABLES				
7	CXFRM CYFRM	Form of major tic marks and Axis	5.	XFRM YFRM
8	CXMTCS CYMTCS	Number of minor tic mark intervals	Ø.	XMTCS YMTCS
9	CXMFRM CYMFRM	Form of minor tic marks	2.	XMFRM YMFRM
1Ø	CXDEC CYDEC	Number of decimal places in labels	Ø.	Set Internally
11	CXDMIN CYDMIN	Data Minimum	Ø.	DLIMX DLIMY or MNMX
12	CXDMAX CYDMAX	Data maximum	Ø.	
13	CXSMIN CYSMIN	Screen minimum	150,125 (1024 addressing)	SLIMX SLIMY or PLACE
14	CXSMAX CYSMAX	Screen maximum	900,700 (1024 addressing)	
15	CXTYPE CYTYPE	Type of data	1.	XTYPE YTYPE
16	CXLSIG CYLSIG	Least significant digit	Ø.	Set Internally
17	CXWDTH CYWDTH	Maximum label width	Ø.	XWDTH YWDTH
18	CXEPON CYEPON	Exponent	Ø.	Set Internally
19	CXSTEP CYSTEP	Number of label steps	1.	Set Internally
20	CXSTAG CYSTAG	Type of label form	1.	Set Internally
21	CXETYP CYETYP	Exponent Type	Ø.	XETYP YETYP
22	CXBEG CYBEG	Beginning location for major tic marks	Ø.	Set Internally
23	CXEND CYEND	Ending location for major tic marks	Ø.	Set Internally

(continued)

TABLE I (continued)

Location	Variable	Description	Initial Value	Subroutines Used To Set Value
24	CXMBEG CYMBEG	Beginning location for minor tic marks	Ø.	Set Internally
25	CXMEND CYMEND	Ending location for minor tic marks	Ø.	Set Internally
26	CXAMIN CYAMIN	Calculated data Minimum	Ø.	Set Internally
27	CXAMAX CYAMAX	Calculated data Maximum	Ø.	Set Internally

The location values are useful when using COMSET and COMGET to change or read any of these variables.

3.1 Individual Table Setting Routines

Descriptions of the routines which perform similar functions for each axis have been grouped in the following sections. For example a description of the neat tic mark variables (CXNEAT and CYNEAT) is followed by a combined subroutine description for the variable setting routines XNEAT and YNEAT.

Remember that all values except the data limits , line and symbol sizes, and machine infinity are entered as integers and are then converted internally to real values for storage.

3.1.1 Line Type - CLINE

This full word real value specifies the type of line to be drawn between points on the graph. Valid values for this field are specified in the chart below. The initial value is set at "1" for a solid line and can be changed by a call to LINE.

11	SOFTWARE* DASH
9	ALTERNATES MOVES AND DRAWS BETWEEN DATA POINTS
1	DOT (............)
2	DOT DASH (·—·—·—·—·—·—·—)
3	DASH (- - - - - - - -)
4	LONG DASH (— — — — — — — — — —)
0	SOLID
-1	NO LINE
-2	VERTICAL BAR
-3	HORIZONTAL BAR
-4	POINT PLOT
-11	USER LINE

* See the Terminal Control System User's Manual for details of the software dashed line.

Subroutine LINE

LINE specifies the type of line to be drawn between data points on the graph.

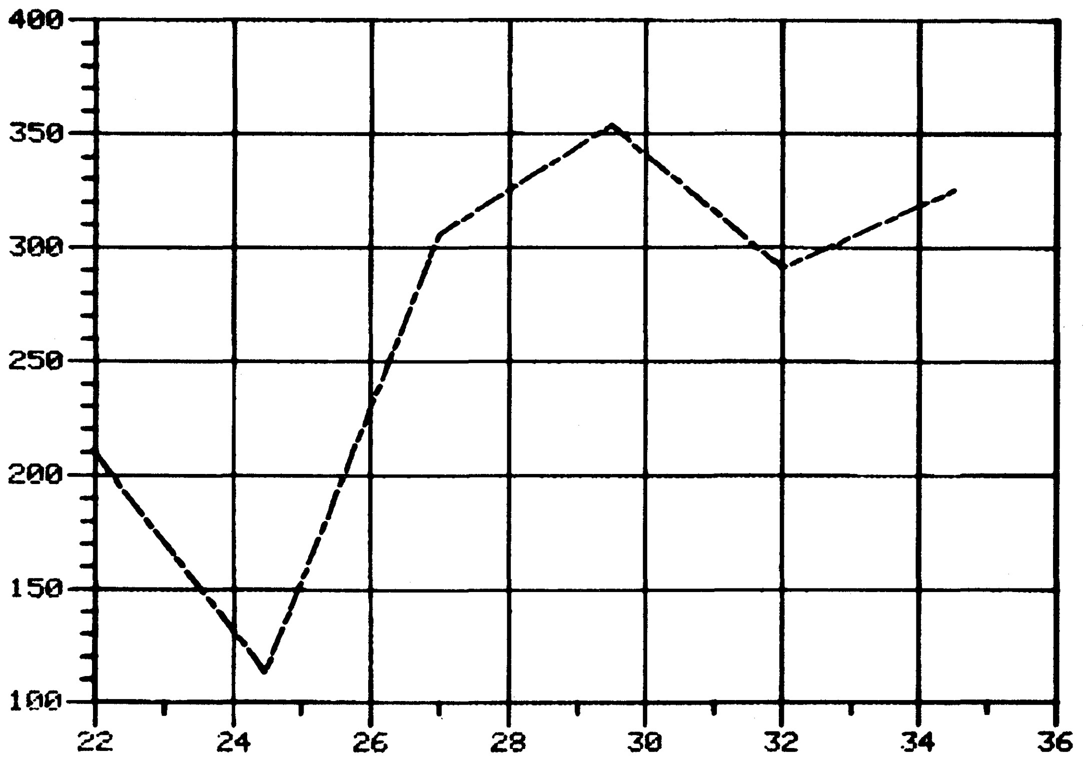
Calling Sequence:

```
     CALL LINE (IVALUE)
```

Parameter Entered:

IVALUE	is the coded integer value for the appropriate line style from the chart on page 3-6

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * DRAW DASHED LINES
      CALL LINE(5232)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Line Type Designation

Figure 3.1

3.1.2 Data Point or Bar Chart Symbols - CSYMBL

This full word real variable specifies the symbol to be drawn at the data point if a data curve is plotted, or the type of shading lines to be used if a bar chart is plotted. The following chart shows possible values for data point symbols. (Bar chart symbols are described in detail in section 2.8)

DATA POINT SYMBOLS

VALUE		RESULT
Ø	None	No symbol
1	○	Drawn circle
2	✕	Drawn "X"
3	△	Triangle
4	□	Square
5	✱	Star
6	◇	Diamond
7	|	Vertical bar
8	+	Cross
9	↑	Up arrow below point
1Ø	↓	Down arrow above point
11	▽	Reverse Triangle
12 -32	Reserved for further Tektronix Development	
33 ≤ SYMBOL ≤ 127	Corresponding ASCII hardware character centered on point (see ASCII Table in Appendix C).	
SYMBOL > 127	Reserved for further Tektronix development	
Negative Values	Control will be passed to user function routine (USERS).	

Initial value of this field is Ø, indicating that no symbol is to be used at the points and no shading lines in the bars.

Size may be adjusted by calling SIZES as described in Section 3.1.8.

Subroutine SYMBL

Subroutine SYMBL designates the symbols at data points, or the shading lines for bar charts.

Calling Sequence:

```
     CALL SYMBL (IVALUE)
```

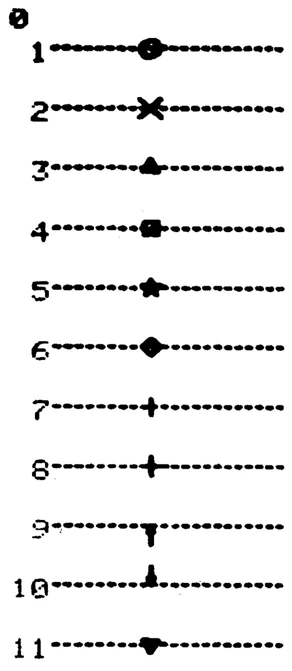
Parameter Entered:

IVALUE	is the value representing the symbols or shading desired.

For a scatter plot without a line through the data point, perform

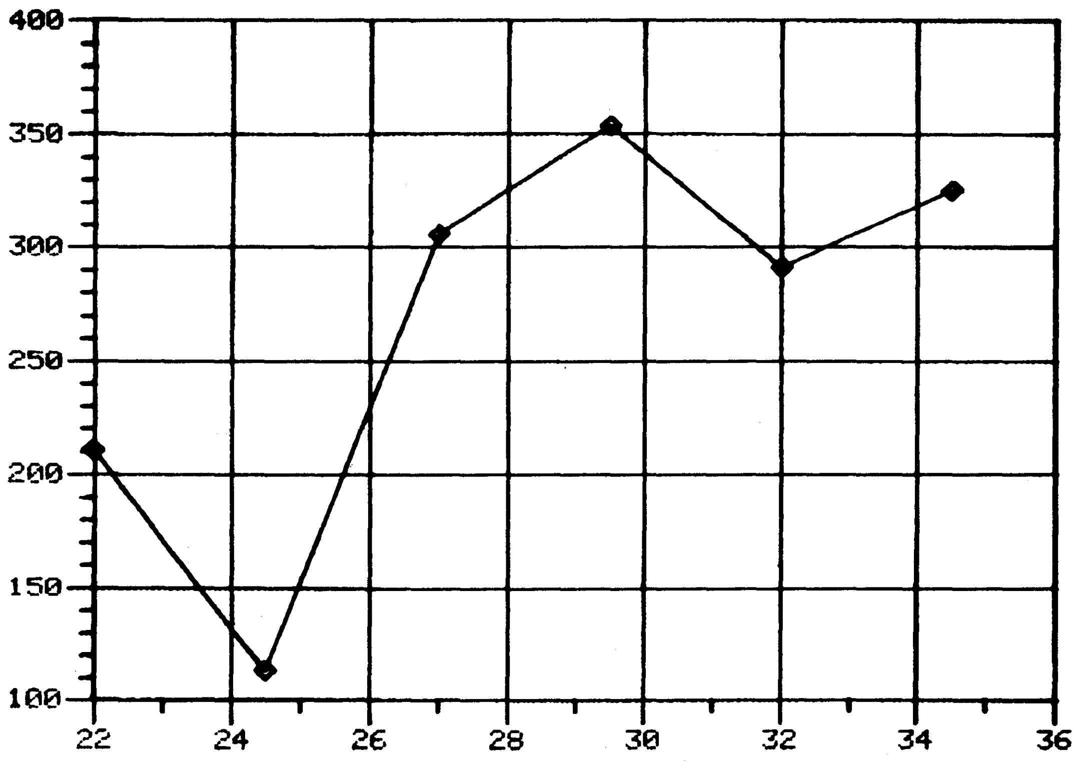
```
     CALL LINE(Ø)
     CALL SYMBL(IVALUE)
```

NOTE: Points and symbols drawn outside the virtual window will be clipped. Symbols drawn at points inside the window will not be clipped.



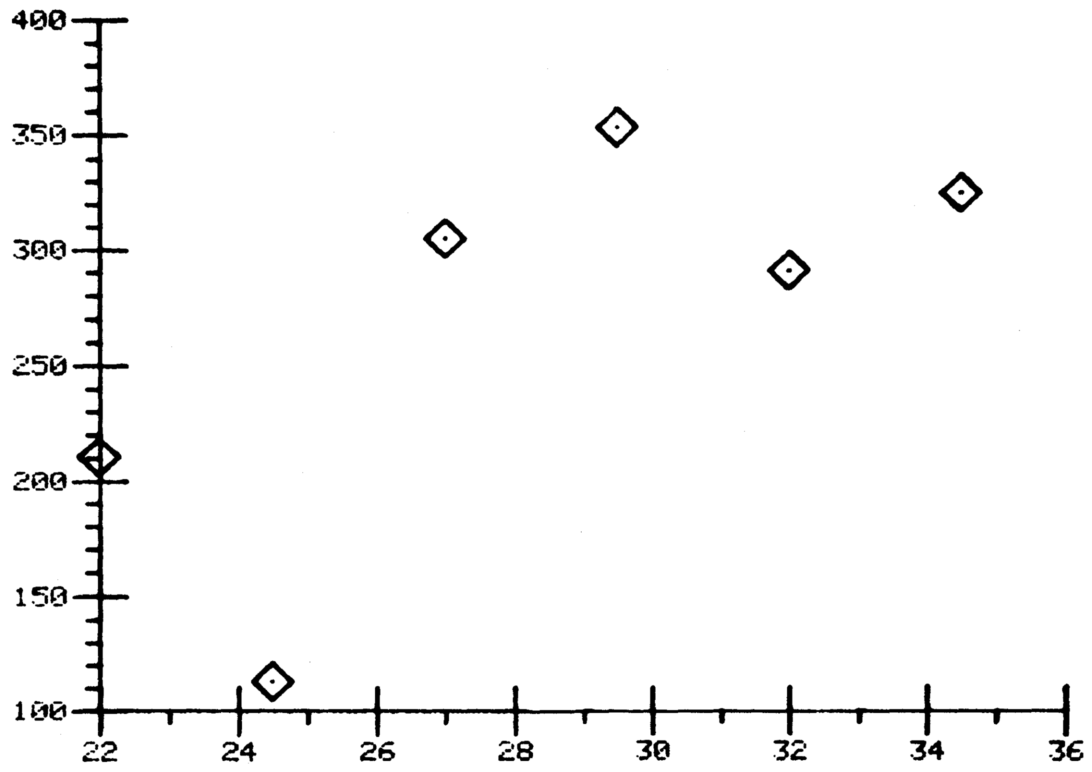
Symbols Available

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * DRAW A SYMBOL AT ALL DATA POINTS
      CALL SYMBL(6)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Data Point Symbols
Figure 3.2

```
        DIMENSION XARRAY(4),YARRAY(7)
        DATA XARRAY/-1.,6.,22.,2.5/
        DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
        CALL INITT(30)
        CALL BINITT
C * NO DATA LINE ONLY A SCATTER-PLOT
        CALL XFRM(4)
        CALL YFRM(4)
        CALL STEPS(1)
        CALL LINE(-4)
        CALL SIZES(2.)
        CALL SYMBL(6)
        CALL CHECK(XARRAY,YARRAY)
        CALL DSPLAY(XARRAY,YARRAY)
        CALL TINPUT(I)
        CALL FINITT(0,700)
        STOP
        END
```



Scatter Plot
Figure 3.2.1

3.1.3 Symbol Increments - CSTEPS

COMMON Table location CSTEPS contains a real variable which designates the increment between data point symbols.

Any time symbols are used, the first data point will be marked with a symbol. Thereafter, symbols will appear only at the designated increments. The following chart shows possible entries and results.

VALUE	RESULTS
1	Designates symbols on every data point.
2	Designates a symbol on every second data point after the first. For example, data points 1, 3, 5, 7, etc., will be marked with symbols.
3	Designates a symbol on every third data point after the first. In this case, the first, fourth, seventh, tenth data points will be marked with symbols.
N	Any increment desired.

Note: Lines will be drawn to every data point unless otherwise specified in CSTEPL, (Section 3.1.6) but the symbols will occur only at the increments designated. CSTEPS is checked for plotting only if symbols have been designated in CSYMBL. Otherwise it is disregarded. The initial value is 1.

Subroutine STEPS

Subroutine STEPS designates the increment between data point symbols.

Calling Sequence:

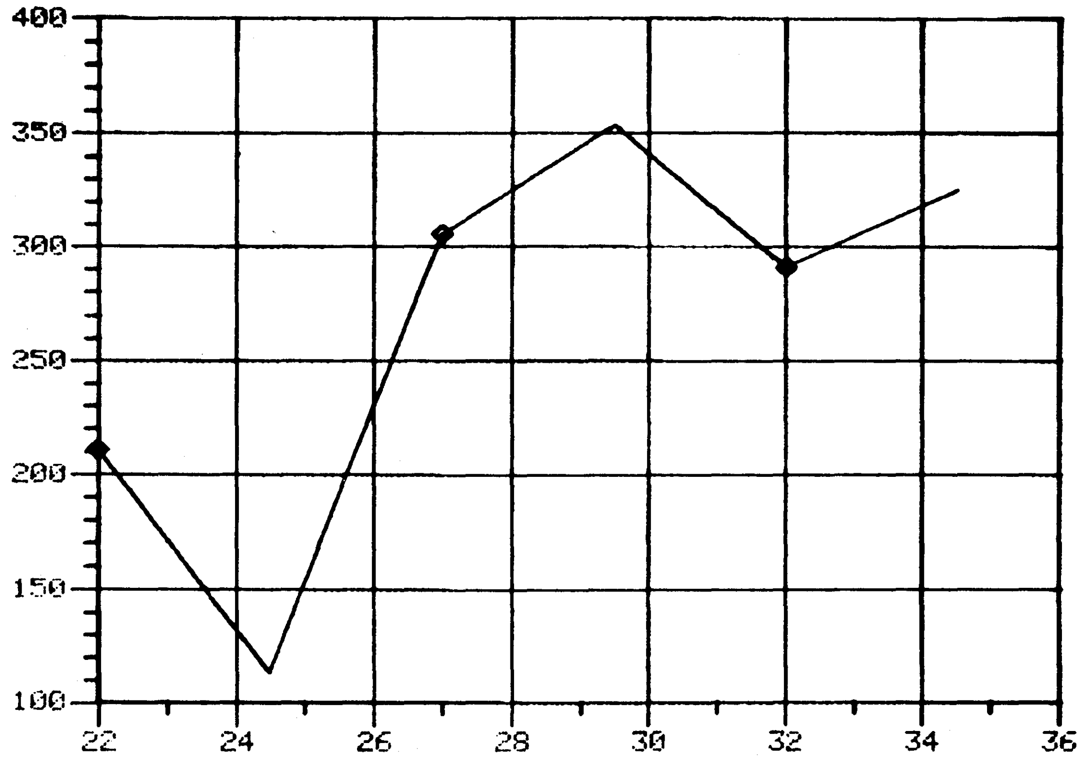
```
CALL STEPS (IVALUE)
```

Parameter Entered:

IVALUE	is an integer value which designates the frequency of the data point symbols.

Note: CSTEPS is defined in terms of points plotted. If every other point is used for the line portion (Section 3.1.6) and CSTEPS is set to two, then every second line point (every fourth data point) will have a symbol.

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * DRAW A SYMBOL ON EVERY OTHER DATA POINT
      CALL STEPS(2)
      CALL SYMBL(6)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Stepped Symbols

Figure 3.3

3.1.4 Machine Infinity - CINFIN

This real value designates a large floating point value acceptable by the hardware being used. The initial value will vary according to the system.

Subroutine INFIN

INFIN sets the value of the large floating point value acceptable to the hardware being used.

Calling Sequence:

```
      CALL INFIN (VALUE)
```

Parameter Entered:

VALUE	is a large floating point value acceptable.

This number is used in the data scanning routines for initialization and in the curve plotting routine to designate missing points. It does not need to be the largest number the machine will handle; it only needs to be larger than any legitimate data value.

CINFIN is primarily useful to designate missing data points. If any data point is encountered which is larger than CINFIN, it is ignored. It may be used to make several Y arrays line up against a single X array rather than require a separate X array for each Y array. (See Section 4.1.1.) An example of such use is found in the plot at the back of this manual titled "MANPOWER IN BUILDING DESIGN". Designer four was not hired until period 308.

3.1.5 Number of Data Points - CNPTS

The value in CNPTS designates the number of data points in a plot for a non-standard data array. This value is initialized to Ø.

Subroutine NPTS

NPTS sets the number of data points to be plotted for a non-standard array (i.e.,an array that does not have the data length as its first value).

Calling Sequence:

```
CALL NPTS (IVALUE)
```

Parameter Entered:

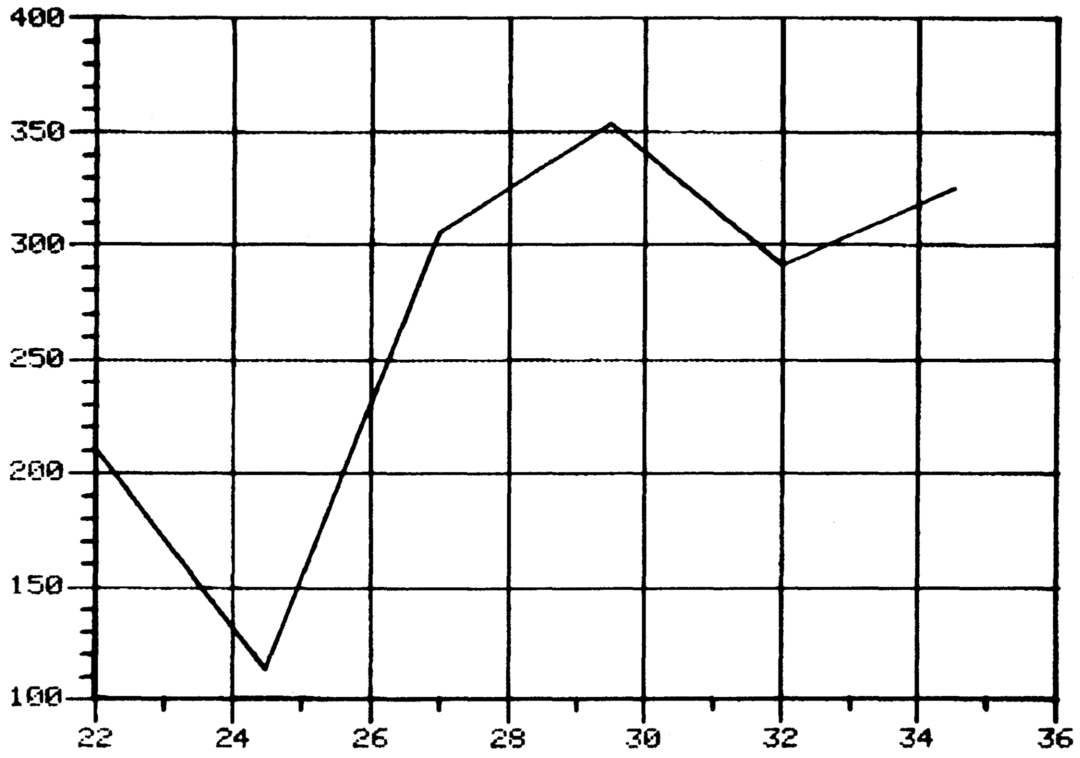
IVALUE	is the integer value specifying the number of data points to be plotted.

This feature was included to permit the use of this package with existing applications. It is suggested that this feature not be used if the standard form can be used, as the standard form has two advantages over the non-standard.

1. Standard form permits use of arrays of different lengths without setting CNPTS.

2. Standard form may be used with short form which is especially convenient for calendar data.

```
      DIMENSION XARRAY(6),YARRAY(6)
C * A NON-STANDARD ARRAY OF FINE POINTS
      DATA XARRAY/22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * ENTER THE NUMBER OF DATA PTS IN THE ARRAY
      CALL NPTS(6)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Use of NPTS for non-standard
Data Array

Figure 3.4

3.1.6 Increments between Line Connected Data Points - CSTEPL

The real variable CSTEPL indicates the increment between data points which the lines will connect. The first data point will always appear as the beginning point of the line, and the increment factor will be followed thereafter.

Subroutine STEPL

This subroutine is called to establish the increment between data points which will be connected by lines.

Calling Sequence:

```
CALL STEPL (IVALUE)
```

Parameter Entered:

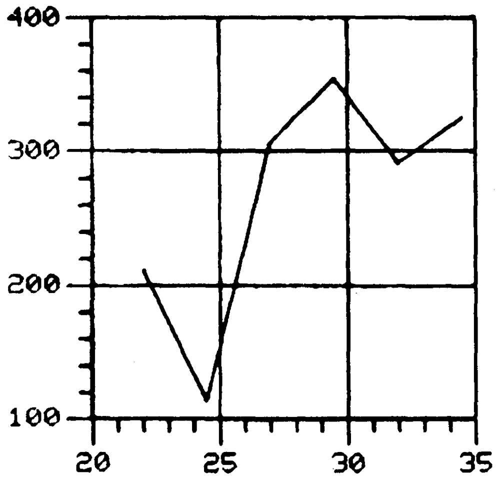
IVALUE	is an integer value specifying the frequency of the data points to be connected. See the value table below.

VALUE	RESULTS
1	Lines will be drawn between every data point.
2	Lines will be drawn between every other data point. (Data points 1, 3, 5, etc., will be connected by lines.)
3	Lines will be drawn between every third data point. (Lines will be drawn directly from data point 1 to 4 to 7, etc.)
etc.	

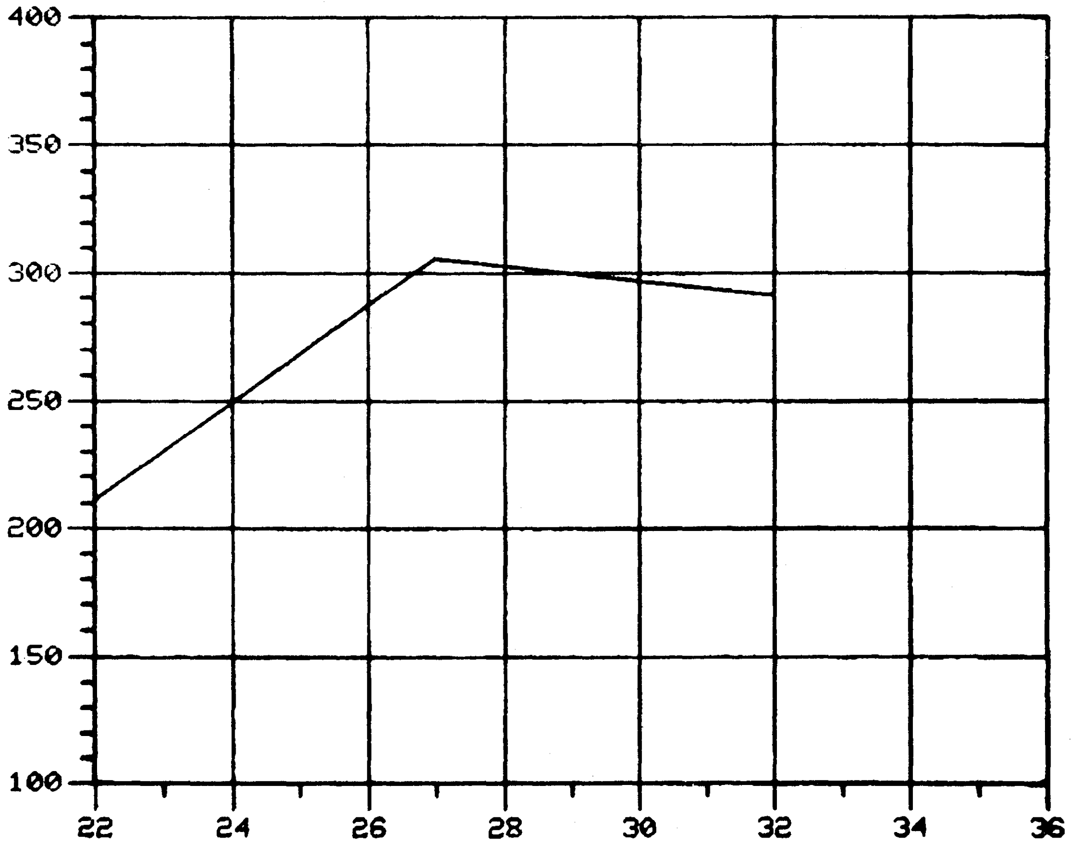
STEPL Value Table

Note: This variable also affects the appearance of symbols on data points. The variable CSTEPS is defined in terms of CSTEPL. If CSTEPL is set to 2 and CSTEPS is 3, then every sixth point will be designated. In other words, data points 1, 3, 5, 7, etc., would be connected with lines and symbols would appear on data points 1, 7, 13, etc.

Standard
Graph



```
      DIMENSION XARRAY(7),YARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C    DRAW THE CURVE SKIPPING EVERY OTHER DATA POINT
      CALL STEPL(2)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
```



Skipped Data Points
Figure 3.5

3.1.7 Number of Points in Plot - CNUMBR

The full word real variable CNUMBR contains the number of points in the plot. CNUMBR is set internally only.

3.1.8 Symbol Size - CSIZES

If a line is designated in CLINE, and symbols are specified, CSIZES is a multiplicative factor indicating symbol size. If a bar chart is designated in CLINE, CSIZES specifies the distance in raster units, between the shading lines, if used. All bar chart specifying information may be entered at once using HBARST or VBARST as described in Section 2.8.

Subroutine SIZES

SIZES sets the size of the data point symbol if a symbol has been specified with subroutine SYMBL.

Calling Sequence:

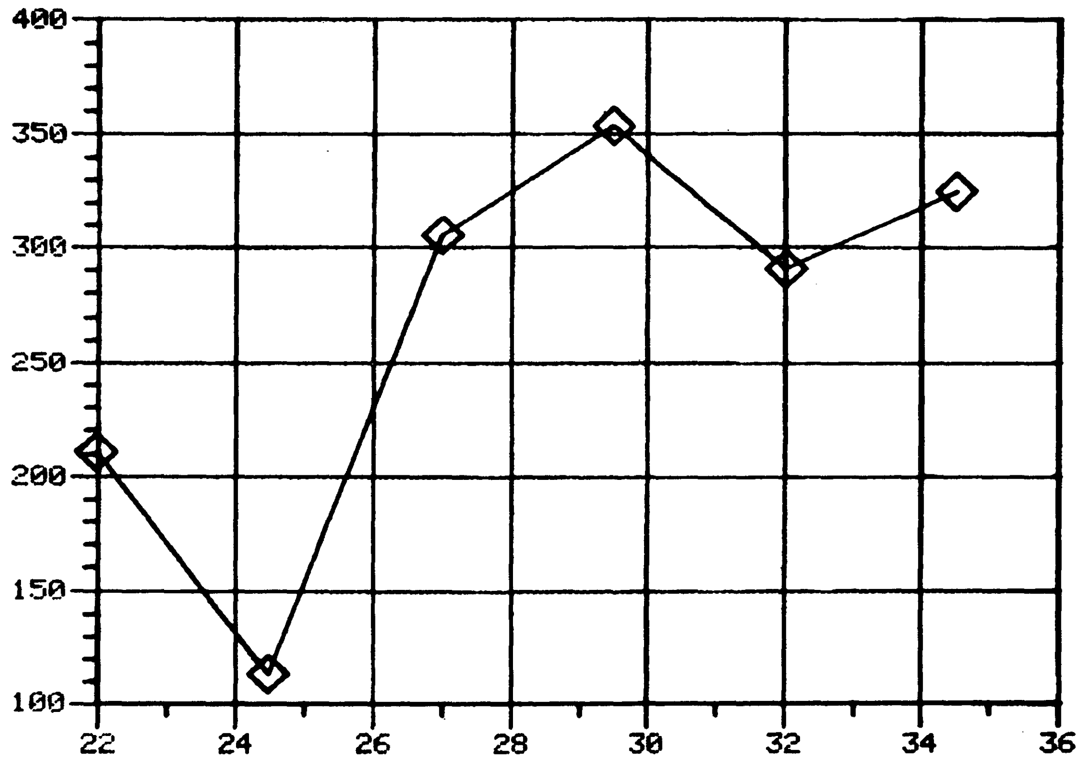
```
        CALL SIZES (VALUE)
```

Parameter Entered:

```
        VALUE      is the real multiplicative factor specifying
                   the size of the data point symbols, or the
                   distance in raster units between bar chart
                   shading lines.
                   Example of multiplicative factor:
                   VALUE=1.5 specifies a symbol 1 1/2 times the
                             standard size.
                        =2.  specifies a symbol twice the standard
                             size.
```

Note: This is one of the few routines which require real values as parameters.

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * CALL TO CSIZE TO CHANGE SIZE OF SYMBOLS
      CALL SIZES(2.)
      CALL SYMBL(6)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL FINITT(0,700)
      STOP
      END
```



Symbol Size Variation

Figure 3.6

3.1.9 Data Line Size (Width) - CSIZEL

The value in CSIZEL specifies the size of the line if a curve plot is being used, or the distance between shading lines if bar charts are being used.

A subroutine is not provided to vary line width, but the user may write his own routine for this purpose using dummy routine USERL. The width of the bar for bar charts is set by calling HBARST, VBARST, or SIZEL. All bar chart specifying information may be entered at once using HBARST or VBARST as described in Section 2.8. If SIZEL is used, bar charts and shading must have been specified with calls to LINE and SYMBL.

Subroutine SIZEL

SIZEL sets the width of the bar used for bar charts.

Calling Sequence:

```
CALL SIZEL(VALUE)
```

Parameters Entered:

VALUE	is a real value designating the width in raster units of the bars.

Note: This is one of the few routines which require real values as parameters.

3.2 X and Y Axis Variables and Setting Routines

The COMMON variables which contain the information to set the appearance of the X and Y axes are described in the following section. Each set of X and Y axis variables are followed by the subroutines which are called to set them.

3.2.1 Neat Tic Marks - CXNEAT, CYNEAT

These full word real variables contain the value "Ø" (off) or "1" (on) indicating whether "NEAT" tic marks are desired. Initial value will be "on", forcing the tic marks to fall at "neat" intervals rather than raw data intervals. The user programmer can alter the value of these words by calling XNEAT or YNEAT and passing the condition that he wishes to prevail.

As "NEAT" is defined, the size of the tic mark interval must be 1, 2, 2.5, or 5 times the integer power of 10. For example, "neat" tic mark intervals might be 100, 2000, 25, 50, etc.

Subroutines XNEAT and YNEAT

These two routines set the values of the neat tic mark variables in COMMON.

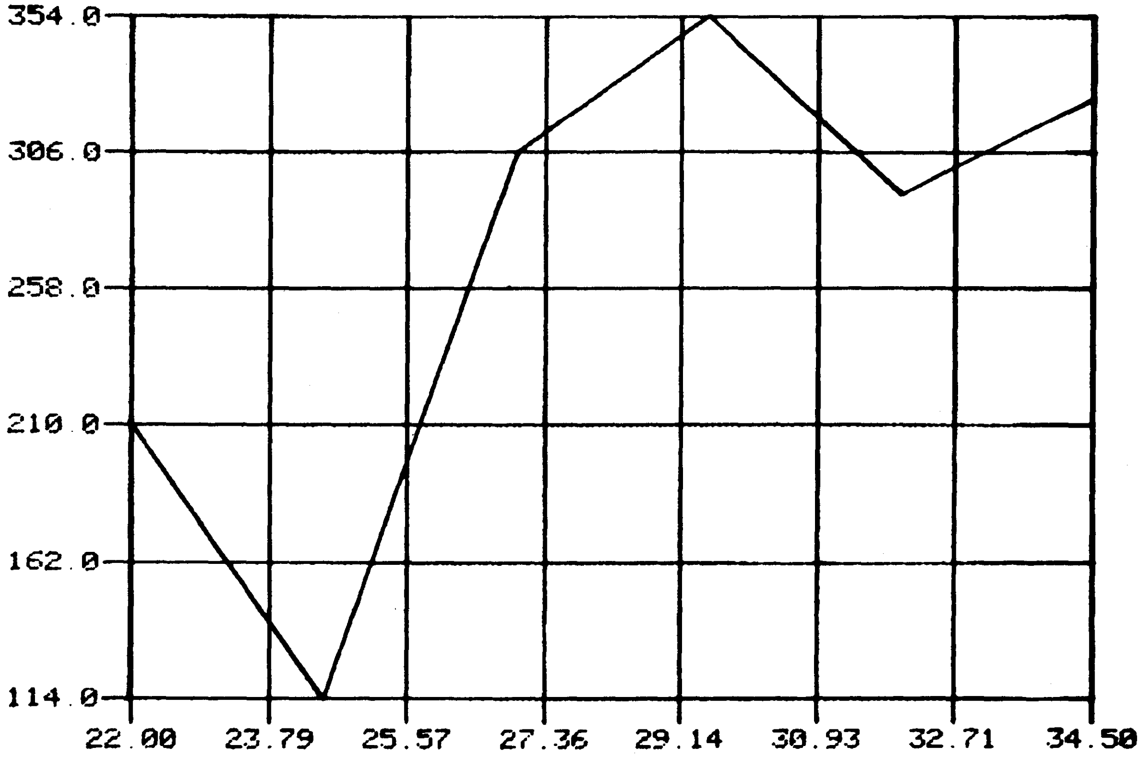
Calling Sequence:

```
CALL XNEAT (IVALUE)
CALL YNEAT (IVALUE)
```

Parameters Entered:

IVALUE	is an integer power desired for the neat tic mark flag. Ø = raw data tic mark values 1 = neat tic mark values

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * TURN OFF THE NEAT TIC FLAG
      CALL XNEAT(0)
      CALL YNEAT(0)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Non-Neat Tic Mark Values

Figure 3.7

3.2.2 Zero Suppression - CXZERO, CYZERO

These full word real variables contain the values "Ø" (off) or "1" (on) indicating whether zero is to be suppressed. An "on" indicates that the zero value will not appear on the graph unless the value of the data being plotted approaches zero. These variables are changed through routines XZERO and YZERO.

Subroutines XZERO and YZERO

These routines set the Ø suppression flag for the X and Y tic mark values in AG-II COMMON.

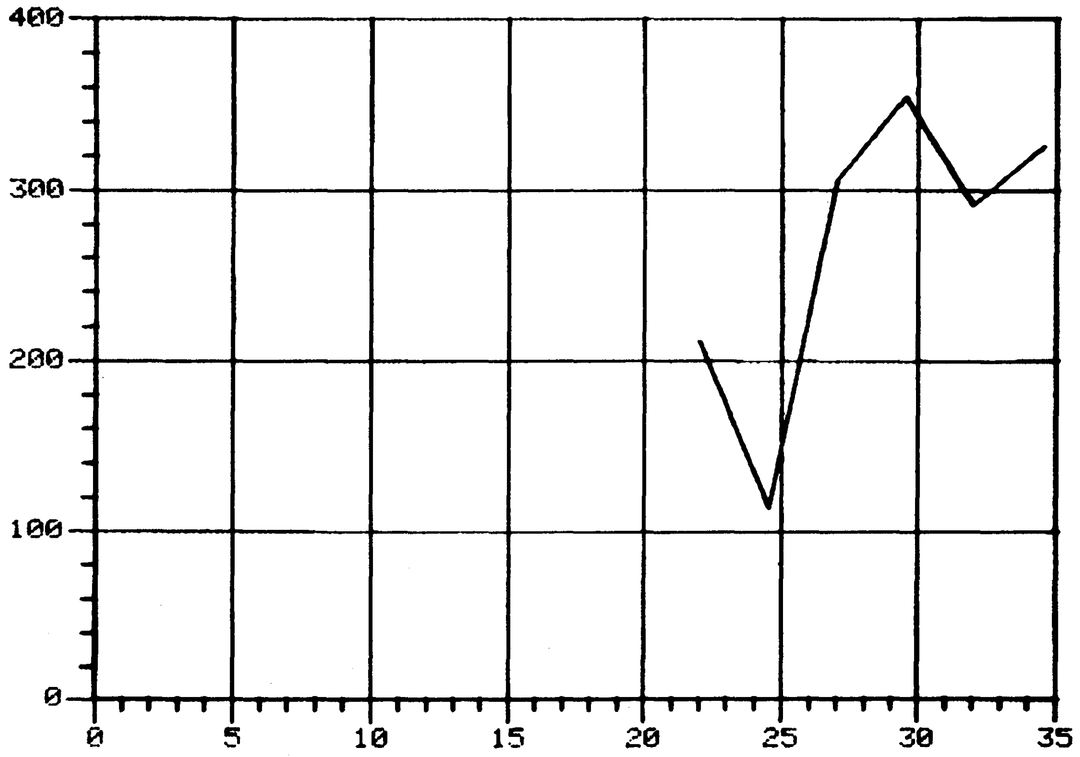
Calling Sequence:

```
CALL XZERO (IVALUE)
CALL YZERO (IVALUE)
```

Parameters Entered:

IVALUE	is the integer value wished for the zero suppression flag. Ø = Zero will always appear as part of the Graph. 1 = Zero will appear as part of the graph only if data includes zero or a point close to it.

```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * INCLUDE ZERO ON THE DISPLAY
      CALL XZERO(0)
      CALL YZERO(0)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Non-Suppression of Zero

Figure 3.8

3.2.3 Axis Location - CXLOC and CYLOC

These full word real variables specify the location of the axes (with all appended tic marks and labels) with respect to the screen minimums. The values placed in these variables are in raster units with a positive value indicating the distance above the X axis screen minimum or to the right of the Y axis screen minimum, and a negative value indicating the distance in the opposite direction.

Four user called routines are available to set the axis locations. XLOC and YLOC require parameters which designate the distance from the screen minimum locations at which the axes are to be located. XLOCTP and YLOCRT require parameters which designate the distance from the screen maximums at which the axes are to be located.

That is,to move the axis to the opposite side of the screen (to the top or left side) XLOCTP or YLOCRT should be called with a parameter value of Ø.

These variables are intitalized to Ø.

Subroutines XLOC and YLOC

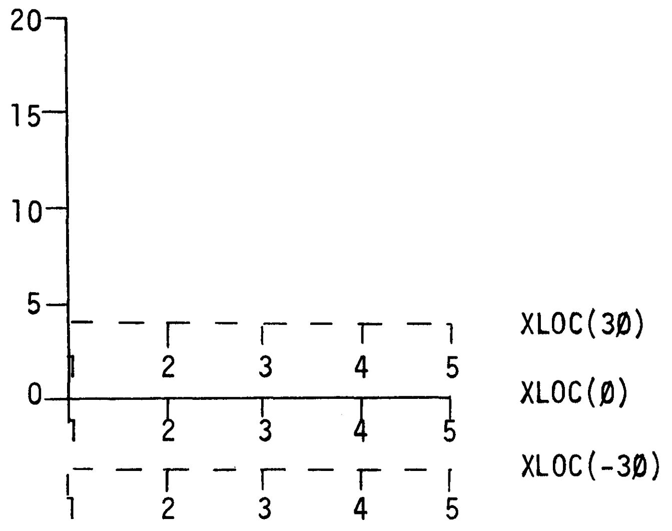
These routines set the locations of the X and Y axes, complete with labels and tic marks, in relation to the left and lower edges of the screen window. See Figure A.4.

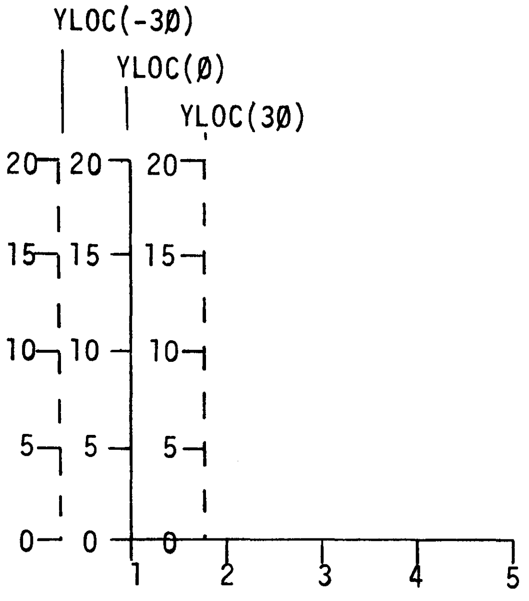
Calling Sequence:

```
CALL XLOC (IVALUE)
CALL YLOC (IVALUE)
```

Parameters Entered:

IVALUE	is the integer number of raster units the axes will be offset from the lower edge of the data window (subroutine XLOC) or from the left edge of the data window (subroutine YLOC). A positive value will move the axis upward (if XLOC is called) or to the right (if YLOC is called). A negative value will move the axis downward (if XLOC) or to the left (if YLOC). Example: The following figures show sample results with an offset of 30 raster units.





Axis Location Change Using
XLOC or YLOC

Figure 3.9.1

Subroutines XLOCTP and YLOCRT

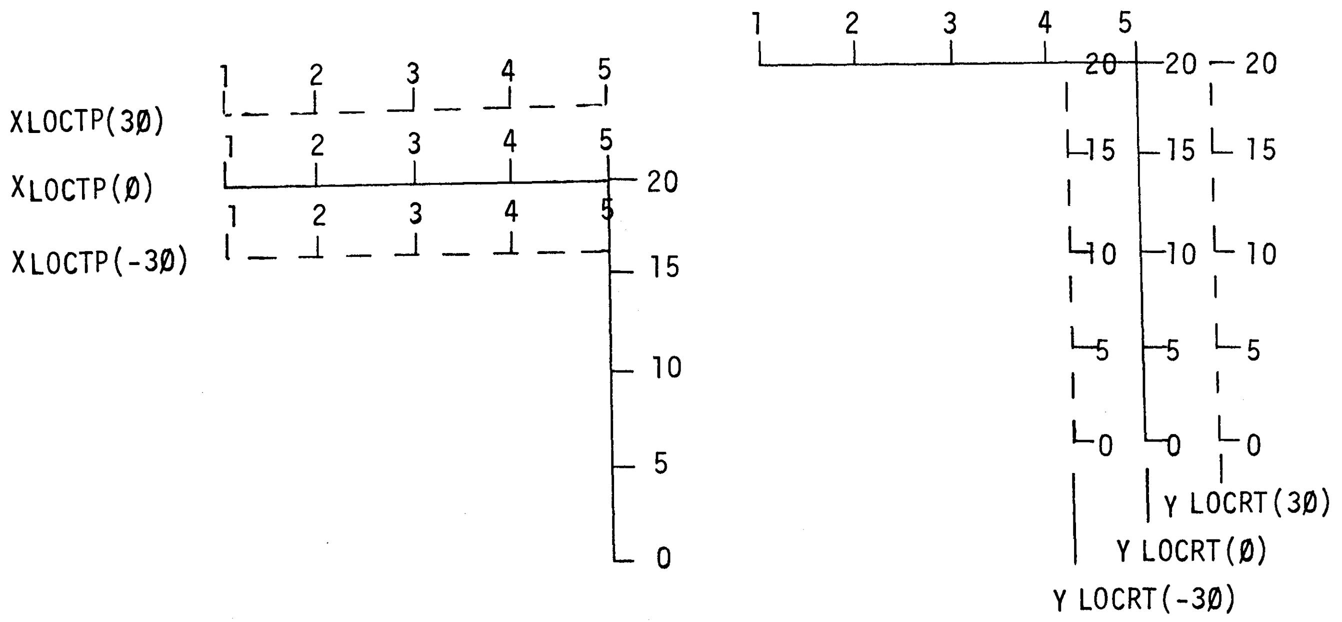
These routines set the locations of the X and Y axes, complete with labels and tic marks, in relation to the right and upper edges of the screen window.

Calling Sequence:

```
CALL XLOCTP (IVALUE)
CALL YLOCRT (IVALUE)
```

Parameters Entered:

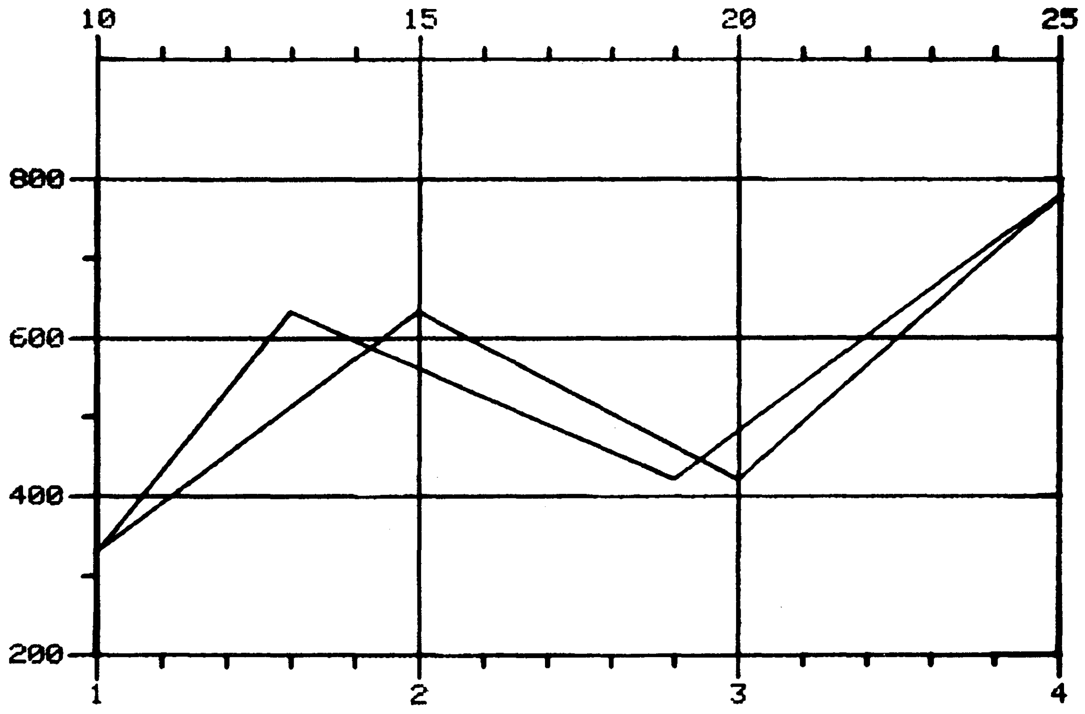
IVALUE	is the integer number of raster units the axes will be offset from the top of the data window (XLOCTP) or from the right edge of the data window (YLOCRT). A positive value will move the axis upward (XLOCTP) or to the right (YLOCRT). A negative value will move the axis downward (XLOCTP) or to the left (YLOCRT). Example: Values of +30, 0, and -30 would place the axes in the locations shown. See Figure A.5 and Figure A.6.



Axis Location Change Using XLOCTP or YLOCRT

Figure 3.9.2

```
       DIMENSION XDATA(5),YDATA(7),ZARRAY(5)
      DATA XDATA/4.,1.,2.,3.,4./
      DATA YDATA/6.,328.,632.,421.,780.,632.,472./
      DATA ZARRAY/4.,10.,13.,19.,25./
      CALL INITT(120)
      CALL BINITT
      CALL SLIMY(200,600)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
C * LOCATE SECOND AXIS AT TOP OF THE DISPLAY
      CALL DINITX
      CALL XLOCTP(100)
      CALL CHECK(ZARRAY,YDATA)
      CALL DSPLAY(ZARRAY,YDATA)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Axis Location Variation

Figure 3.9.3

3.2.4 Tic Mark Label Types - CXLAB, CYLAB

These full word real variables contain the coded values which specify the types of labels to be used for the X and Y axes, respectively. The default value is 1. If not user set, the label type will always match the transformation type (CXTYPE and CYTYPE). The user may call XLAB and YLAB to alter the label type values at any time before calling CHECK.

Subroutines XLAB and YLAB

These routines are called to set the X or Y axis tic mark label types in AG-II COMMON.

Calling Sequences:

```
CALL XLAB (IVALUE)
CALL YLAB (IVALUE)
```

Parameters Entered:

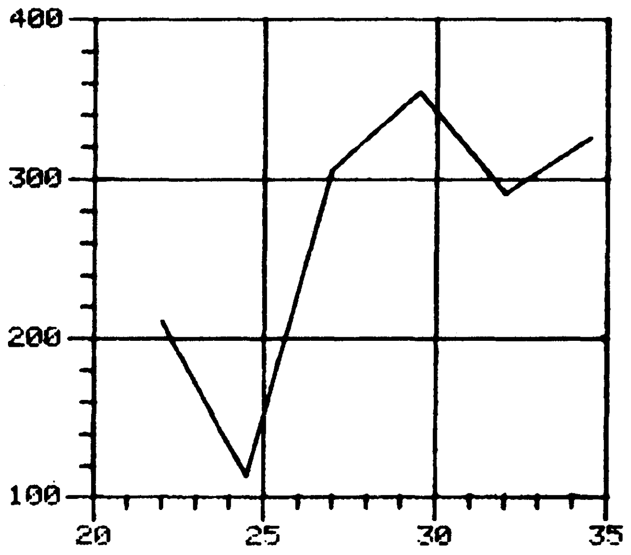
IVALUE	is an arbitrary integer value which refers to a label type as described in the table below.

Internal Code	Type
0	No Label
1	Matches transformation type
2	Logarithmic
3	Days
4	Weeks
5	Periods
6	Months
7	Quarters
8	Years
Negative	User Label +

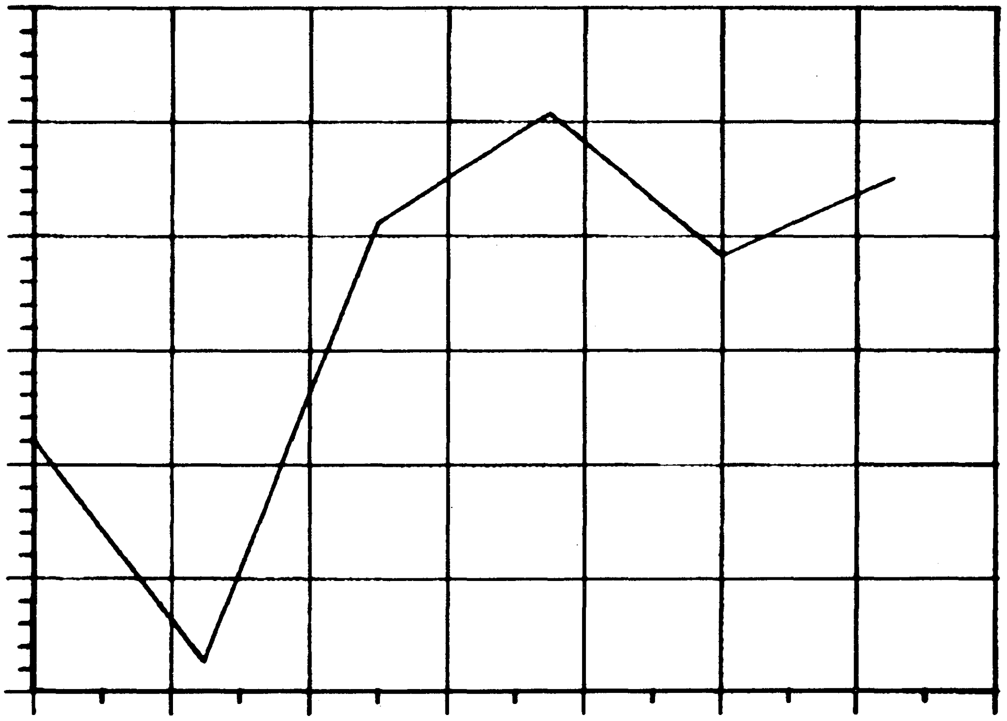
Note: This variable is not normally changed except in some cases involving calendar data where the labels may be different than the data. For example, daily data may be plotted with monthly tic marks and labels.

+ User defined label (See subroutine USESET, System Manual.)

Standard
Graph



```
      DIMENSION XARRAY(7),YARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * DELETE THE LABELS
      CALL XLAB(0)
      CALL YLAB(0)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Tic Mark Label Variation

Figure 3.10

3.2.5 Tic Mark Density - CXDEN, CYDEN

A parameter value of 1 to 10 will determine the density of the tic marks. If a parameter value between 1 and 5 is chosen, only major tic marks will appear; if a value from 6 to 10 is chosen, minor tics will appear, also. The coded values from 1 to 5 and from 6 to 10, respectively, will vary the number of tics from sparse (1 and 6) to dense (5 and 10).* See the chart below.

If the user has not set the exact number of tic marks he wants by calling XTICS or YTICS, subroutine OPTIM uses the density factor, along with other pertinent information from COMMON, to determine the number of tic marks and the intervals.

If calendar data is used, no minor tic marks will appear on the time axis. The density of the major tic marks will be controlled the same as with any other linear data.

The initial value is 8.

Subroutines XDEN and YDEN

These subroutines set the values of the density factors in AG-II COMMON.

Calling Sequences:

```
CALL XDEN (IVALUE)
CALL YDEN (IVALUE)
```

Parameters Entered:

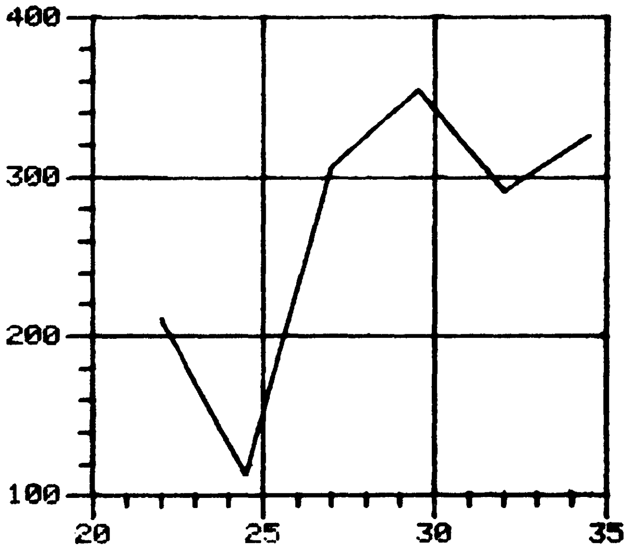
IVALUE	is the integer value of the density factor to be used in determining the frequency of tic marks. The table below contains the valid values.

	No Minor Tics	With Minor Tics
sparse	1	6
↓	2	7
	3	8
	4	9
dense	5	10

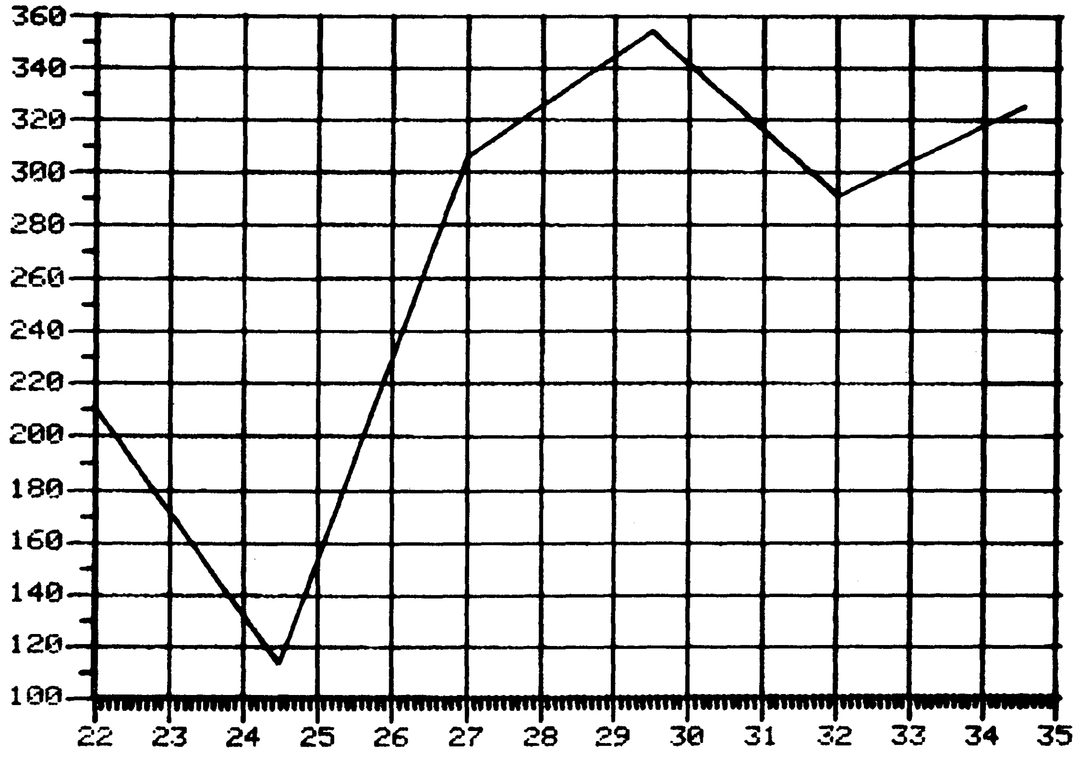
Tic Mark Density Table

* Density of tics on the 4014 or 4015 Terminal may vary between different character sizes. For example, a density value of 5 will be more dense for character size 4 than for size 1.

Standard Graph

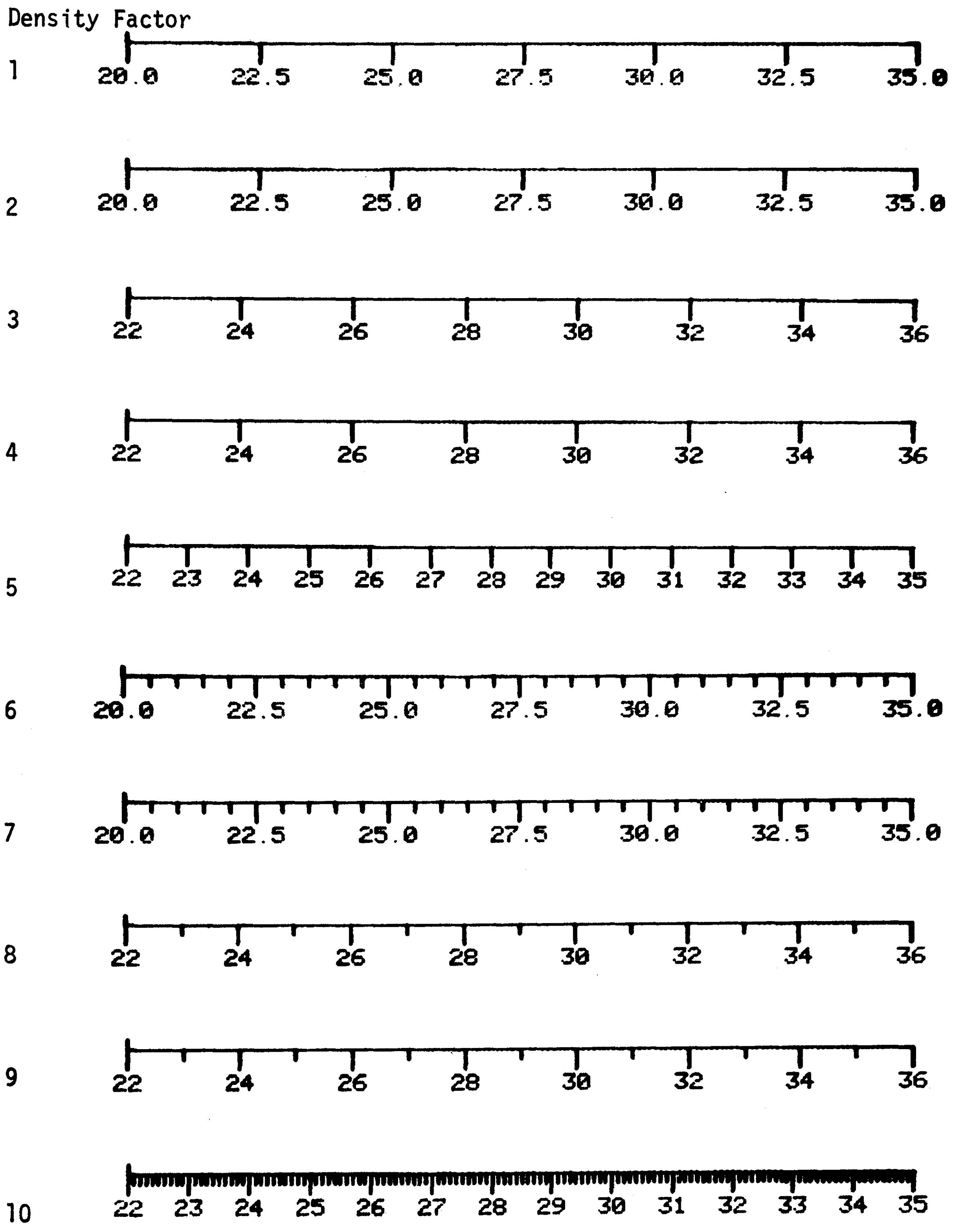


```
       DIMENSION XARRAY(4),YARRAY(7)
       DATA XARRAY/-1.,6.,22.,2.5/
       DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
       CALL INITT(30)
       CALL BINITT
C * SET THE DENSITY OF THE TICS
       CALL XDEN(10)
       CALL YDEN(10)
       CALL CHECK(XARRAY,YARRAY)
       CALL DSPLAY(XARRAY,YARRAY)
       CALL TINPUT(I)
       CALL FINITT(0,700)
       STOP
       END
```



Tic Mark Density Variation

Figure 3.11



Sample Density Factor Results

Figure 3.12

3.2.6 Major Tic Mark Intervals - CXTICS, CYTICS

These full word real variables contain the number of tic mark intervals to be used on the X and Y axes. Initial value for these variables is Ø, but the user may set the specific number of intervals desired by calling XTICS or YTICS. The actual value of each tic location will then be determined internally.

If the user does not specify the number of tic mark intervals, the number will be calculated by the density algorithm, based on the density value found in COMMON (CXDEN and CYDEN).

Subroutines XTICS and YTICS

With these routines the user sets the values of the number of tic mark intervals on the X and Y axes.

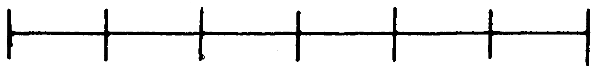
Calling Sequence:

```
CALL XTICS (IVALUE)
CALL YTICS (IVALUE)
```

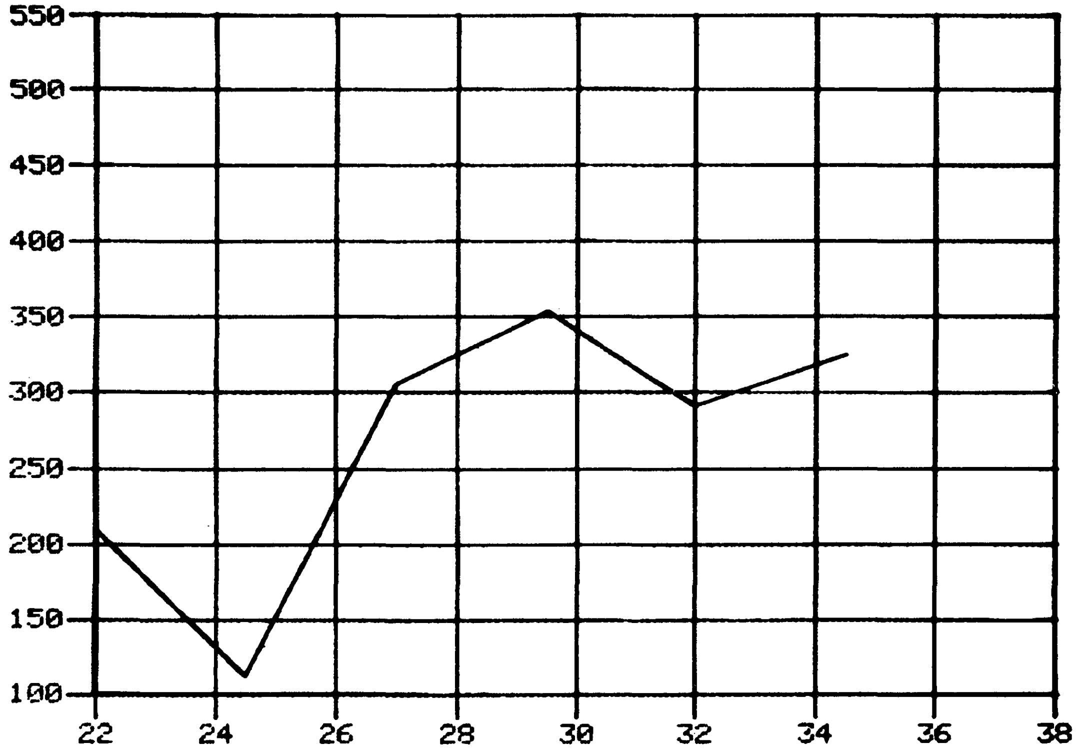
Parameters Entered:

IVALUE	is integer value indicating the number of tic mark intervals on the X or the Y axis.

Example: For the following axis with 7 tic marks, the value in COMMON would be 6 (the number of intervals).



```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * SET THE NUMBER OF TIC INTERVALS FOR EACH AXIS
      CALL XTICS(8)
      CALL YTICS(9)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



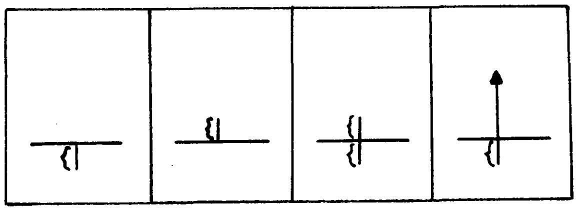
Tic Mark Intervals

Figure 3.13

3.2.7 Tic Mark Length - CXLEN, CYLEN

These full word real variables indicate the length (in raster units) of the major tic marks. Initial value of these variables is "20". The user may change these values by calling XLEN or YLEN. These values refer to the distance the tic marks are to extend outward from the axis line on each side, not including any grid lines.

Example: The part of the tic mark for which length is is specified is shown with brackets below.



Note: Minor tic marks, if specified, will be half the length specified by these fields.

Subroutines XLEN and YLEN

These subroutines specify the length of the major tic marks on each axis.

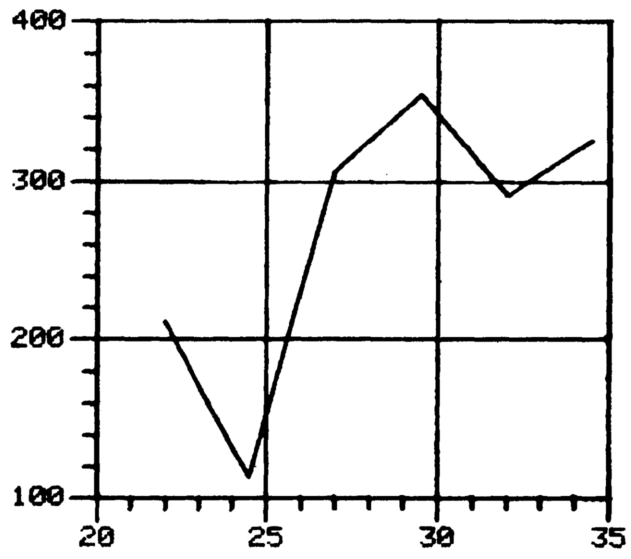
Calling Sequence:

```
CALL XLEN (IVALUE)
CALL YLEN (IVALUE)
```

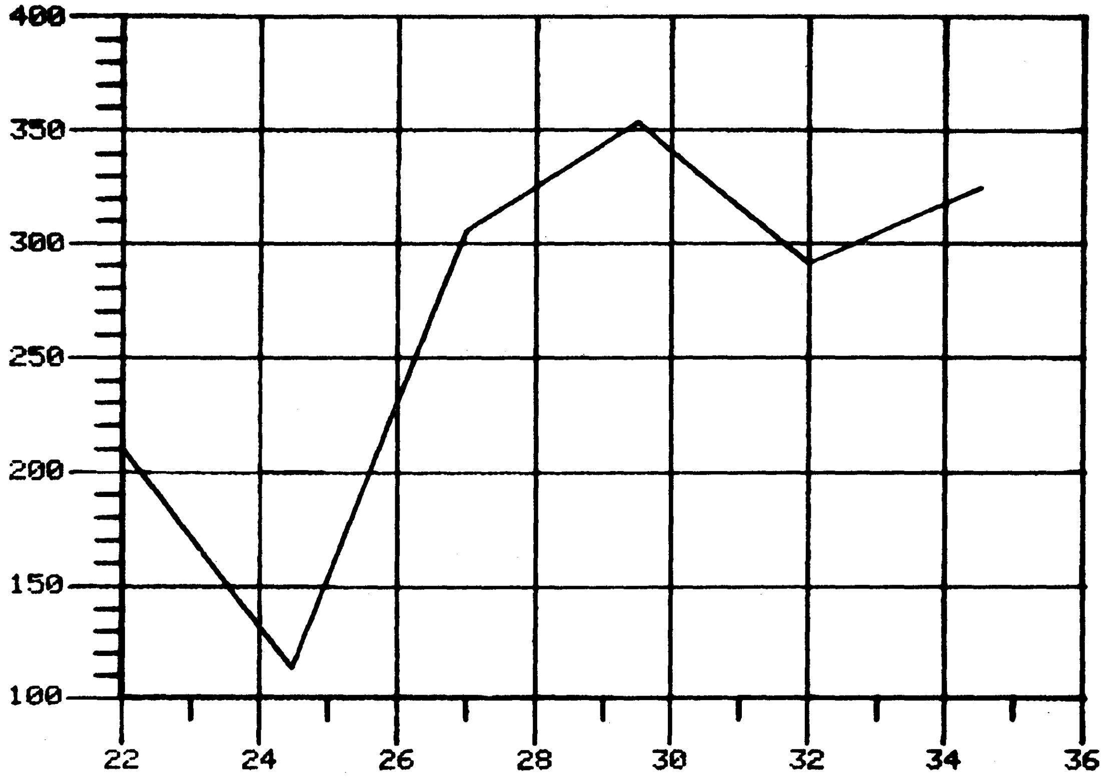
Parameters Entered:

IVALUE	is an integer specifying the length chosen for the tic marks on the X and the Y axes, respectively. The length designates the distance in raster units that the tic marks will extend outward from the axis, not including grid lines.

Standard Graph



```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * CHANGE THE LENGTH OF THE TICS
      CALL XLEN(40)
      CALL YLEN(40)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Tic Mark Length Variation

Figure 3.14

3.2.8 Major Tic Mark Form - CXFRM, CYFRM

These real variables stored in CXFRM and CYFRM determine the form of the major tic marks on the respective axes. The form for each value (1 through 6) is illustrated in the chart below.

These variables also determine whether grid lines are to appear, either alone or as a continuation of the tic mark (see codes 5 and 6). The initial value is 5 (a tic mark with a grid line).

XLEN and YLEN are called to specify the desired length of the tic marks, excluding grid lines. (Grid lines will extend exactly to the opposite side of the window if tic mark type 5 or 6 is chosen.)

If bar charts are drawn using a call to HBARST or VBARST (page 2-20), grid lines parallel to the bars will be suppressed.

Subroutines XFRM and YFRM

XFRM sets the form of the X axis major tic marks for the AG-II COMMON.

Calling Sequence:

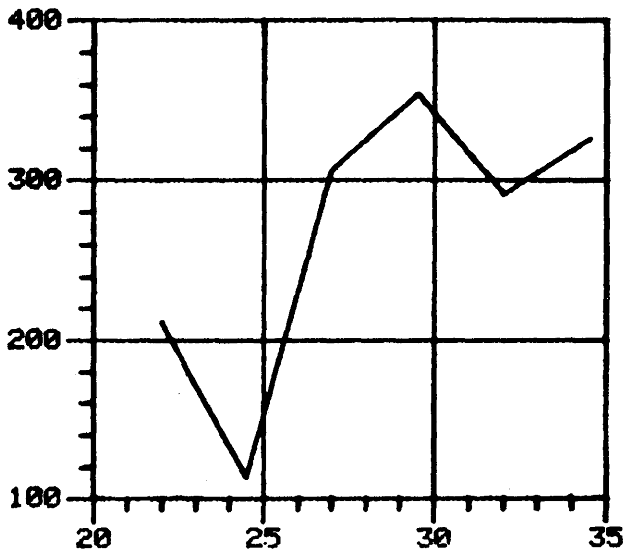
```
CALL XFRM (IVALUE)
CALL YFRM (IVALUE)
```

Parameters Entered:

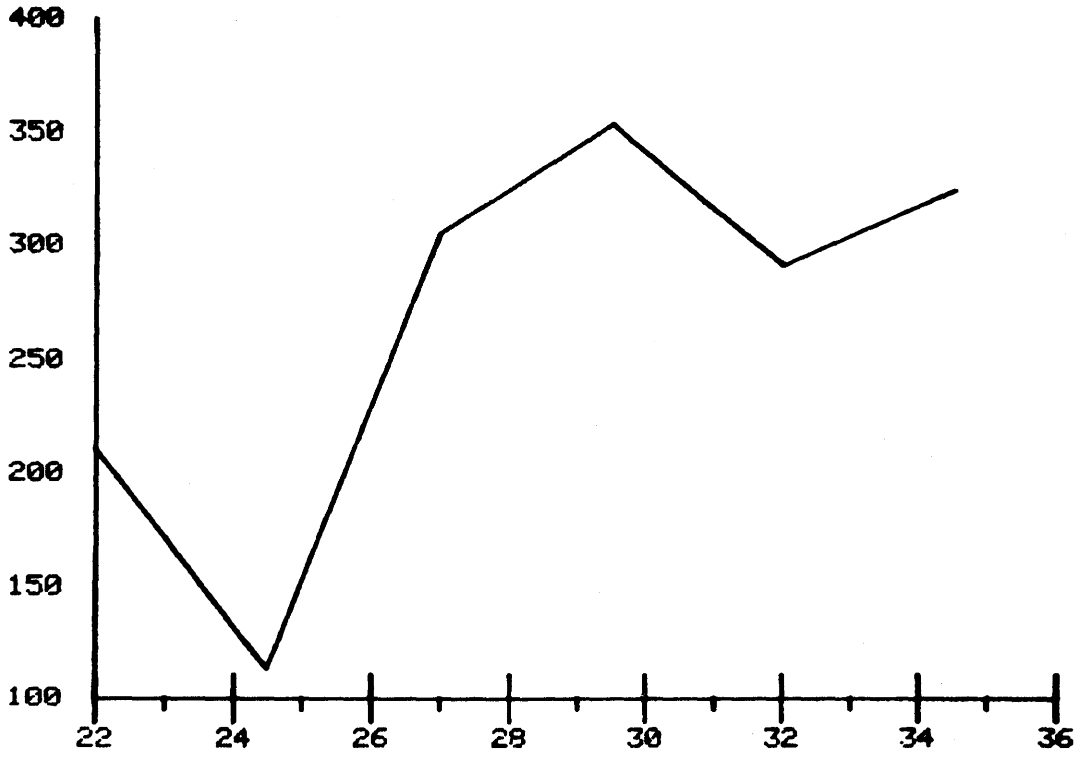
IVALUE	is the coded integer value of the form of the tic marks desired. Valid values are numbers ∅ through 6 as shown on the chart below. Types 5 and 6 include grid lines.

Results	No Tics or Axis	No Tics	⊤	⊥	+	↥	↥
Code	∅	1	2	3	4	5	6

Standard Graph



```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * CHANGE THE FORM OF THE TICS
      CALL XFRM(4)
      CALL YFRM(1)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Tic Mark Form Variation
Figure 3.15

3.2.9 Minor Tic Mark Intervals and Form - CXMTCS, CYMTCS, CXMFRM, CYMFRM

These full word real variables contain the number of minor tic mark intervals to appear between two adjacent major tic marks. The initial value for these variables is Ø, but the user may set the intervals by calling XMTCS or YMTCS. The actual value of each minor tic location will then be determined internally. See the diagram on page 105.

If the user does not specify the number of minor Tic mark intervals, the number will be calculated by the density algorithm, based on the density value found in COMMON (CXMTCS and CYMTCS).

Subroutine XMTCS and YMTCS

Set the number of minor tic mark intervals between adjacent major tic marks.

Calling Sequence:

```
CALL XMTCS (IVALUE)
CALL YMTCS (IVALUE)
```

Parameters Entered:

IVALUE	is the integer value indicating the number of minor tic mark intervals between adjacent major tic marks.

Minor Tic Mark Form - CXMFRM, CYMFRM

These full word real variables designate the minor tic mark form for each axis with a code value from the table below. (these routines use the same table of tic mark forms as is used for major tic marks.) Initial value is 2 and may be changed by calling XMFRM or YMFRM.

Note: Minor tic marks if used, will be half the length of the major tic marks. (Grid lines will still extend the full width of the screen.)

Subroutines XMFRM and YMFRM

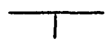
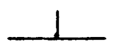
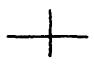
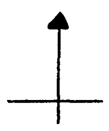
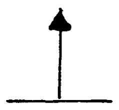
This routine sets the form of the minor tic marks on the X axis.

Calling Sequence:

```
CALL XMFRM (IVALUE)
CALL YMFRM (IVALUE)
```

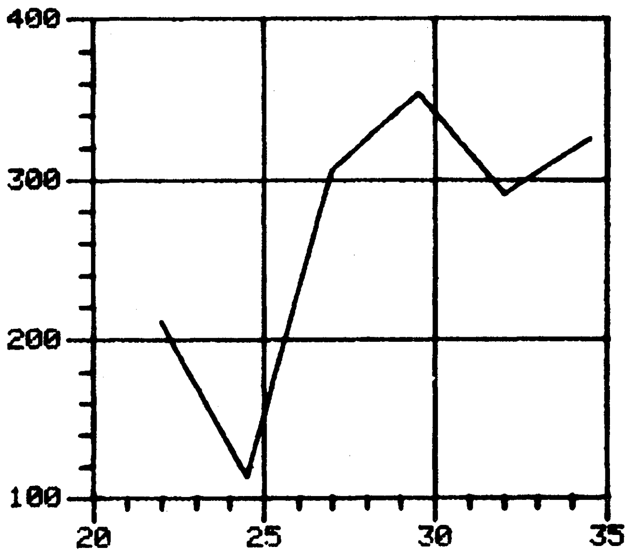
Parameters Entered:

IVALUE	is the code integer value designating the tic mark form desired (1 through 6 from the table below).

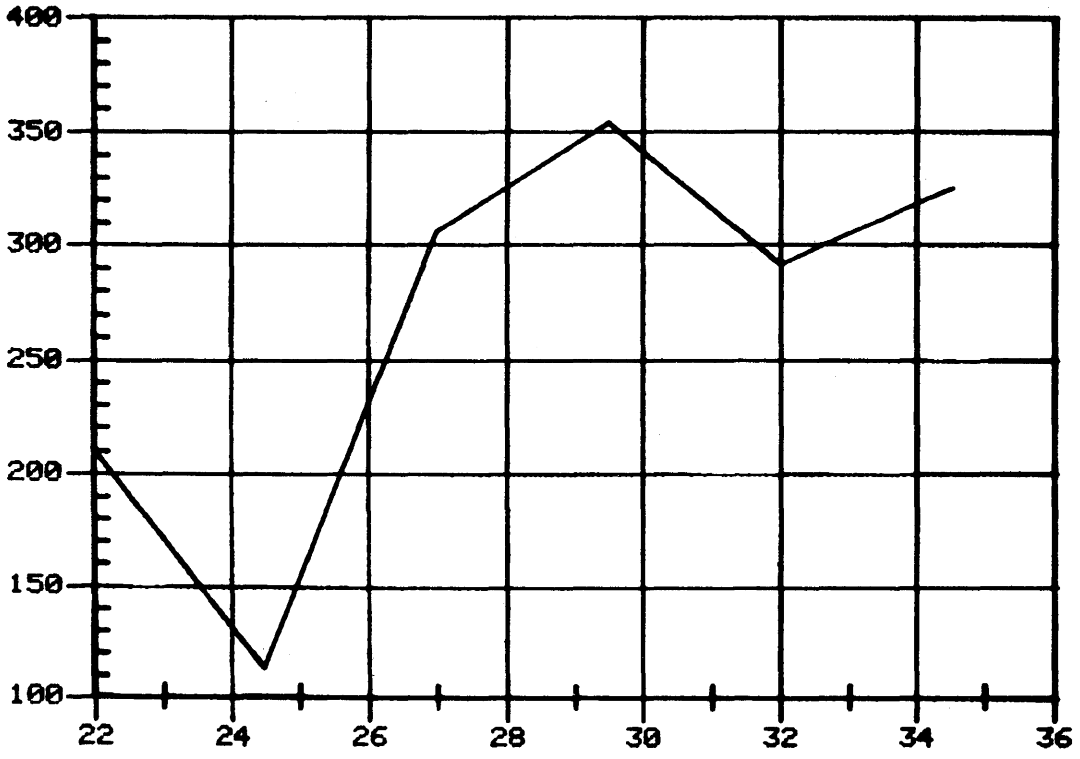
Results	No Tics					
Code	1	2	3	4	5	6

Types 5 and 6 extend to the opposite side of the screen, forming a grid.

Standard Graph



```
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
C * CHANGE THE FORM OF THE MINOR TICS
      CALL XMFRM(4)
      CALL YMFRM(3)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Minor Tic Mark Form Variation
Figure 3.16

3.2.10 Decimal Places in Labels - CXDEC, CYDEC

Common locations CXDEC and CYDEC contain the real variables which designate the number of decimal places to be used in the labels for the respective axis.

These values are set by subroutine LWIDTH which takes into consideration the space available for the label and whether or or not a remote exponent will be necessary.

Note: The user cannot set these values; they are set and used internally only.

3.2.11 Data Minimum and Maximum - CXDMIN, CYDMIN, CXDMAX, and CYDMAX

These full word real values are the data minimums and maximums for the X and Y arrays. The initial values of these fields are Ø. Subroutine CHECK checks to see if these fields have been user set. If not, CHECK calls RGCHEK which, in turn, calls MNMX which determines the minimum and maximum of each array. The values are then entered into the COMMON Table. The user may call MNMX which requires the array as a parameter and returns the data minimum and maximum which must be entered with DLIMX or DLIMY. If you already know the minimum and maximum to be entered use DLIMX and DLIMY only.

DLIMX and DLIMY may be used to force a data window to span a greater range than normal if the user sets the data minimum to a value less than that of the actual data and the maximum to be greater than the actual data maximum, or to clip data if the limits are set less than the actual data limits.

Subroutines DLIMX and DLIMY*

DLIMX and DLIMY are called to set the data minimums and maximums of the X and Y axes, respectively, in AG-II COMMON.

Calling Sequence:

```
    CALL DLIMX (XDMIN,XDMAX)
    CALL DLIMY (YDMIN,YDMAX)
```

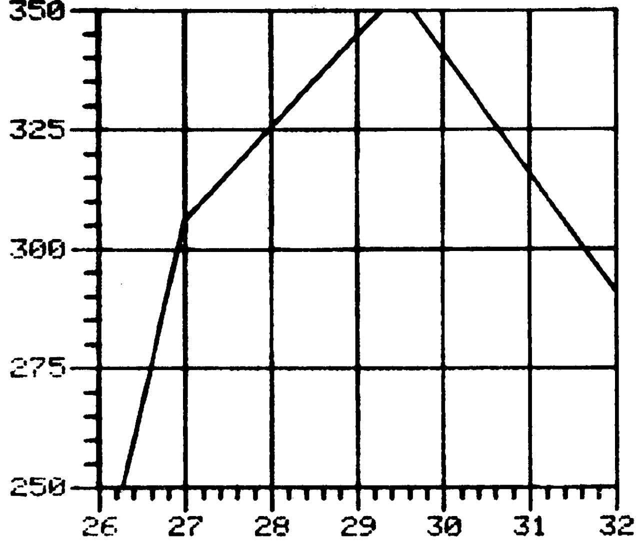
Parameters Entered:

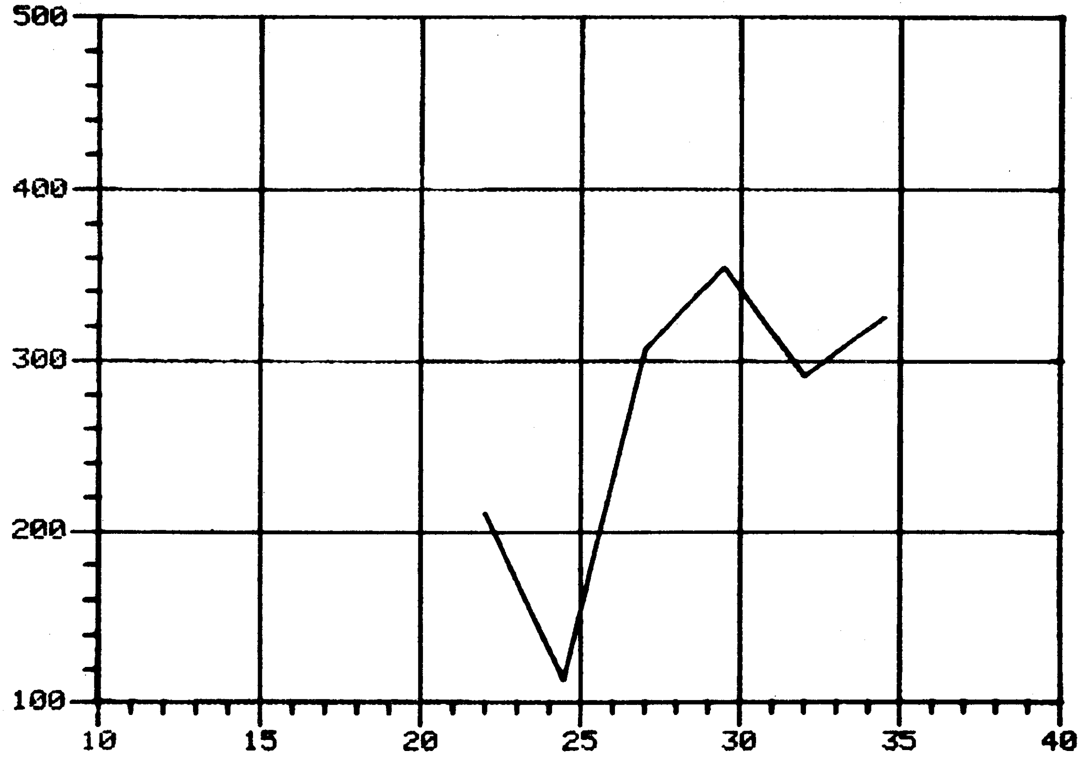
XDMIN or YDMIN	is a real value at which the data minimum for the X or Y axis,respectively, to be set in AG-II COMMON.
XDMAX or YDMAX	is the real value at which the data maximum for the X or Y axis, respectively, is to be set in AG-II COMMON.

*For calendar data, the OUBGC value is necessary when setting DLIMX or DLIMY.

```
      DIMENSION XARRAY(7),YARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(120)
      CALL BINITT
C * SET NEW DATA BOUNDARIES
      CALL DLIMX(11.,40.)
      CALL DLIMY(100.,500.)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
C * SHOW THE CLIPPING EFFECT
      CALL SLIMX(100,500)
      CALL SLIMY(200,600)
      CALL DLIMX(26.,32.)
      CALL DLIMY(250.,325.)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```

Clipped Window Area





Data Limit Usage
Figure 3.17

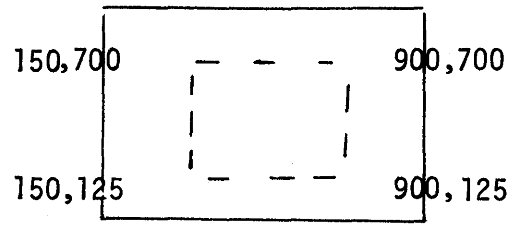
Note: SLIMX and SLIMY are called to fit the example into the limited page space available.
The neat tic mark algorithm is in effect, causing the user set data limits to be rounded to "neat" values.

3.2.12 Screen Location - CXSMIN,CYSMIN, CXSMAX, CYSMAX, and PLACE

These full word real variables are the screen minimums and maximums for the X and Y coordinates of the data window. The initial values of these positions in the table are set to give a maximum size window that will still result in minimum default labeling. The initial values are:

CXSMIN	150
CXSMAX	900
CYSMIN	125
CYSMAX	700

so that the screen window will be positioned as follows:



These variables may be set by individual calls to SLIMX and SLIMY.

Subroutine PLACE allows the user to select optional pre-defined areas on the screen such as "upper half" or "lower right sixth". (See description on page 112.)

Subroutines SLIMX and SLIMY

SLIMX and SLIMY are called to set the screen minimum and maximum for the X and Y axes in AG-II COMMON.

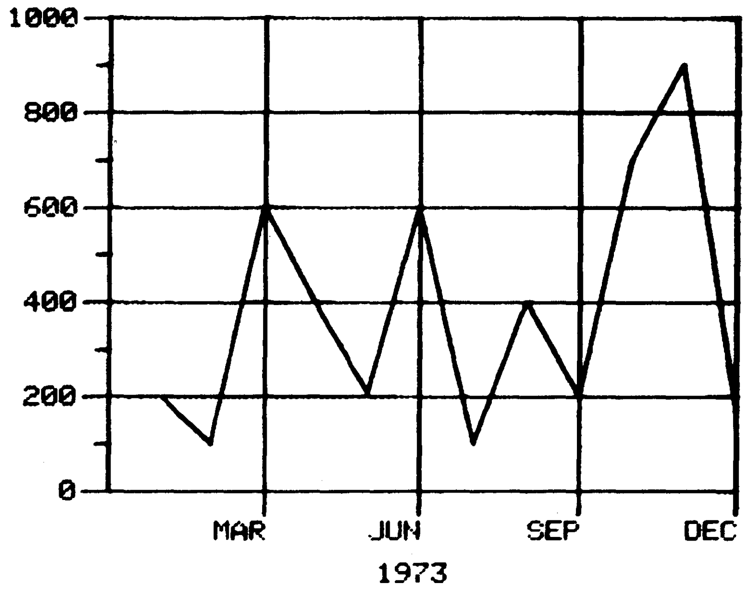
Calling Sequence:

```
CALL SLIMX (ISMIN, ISMAX)
CALL SLIMY (ISMIN, ISMAX)
```

Parameters Entered:

ISMIN	is the integer minimum screen location in raster units (the value to be entered in CXSMIN or CYSMIN).
ISMAX	is the integer screen location in raster units (the value to be entered in CXSMAX or CYSMAX).

```
      DIMENSION XARRAY(5),YARRAY(13)
      DATA XARRAY/-2.,12.,12.,1973.,1./
      DATA YARRAY/12.,200.,100.,600.,400.,209.,599.,
     &100.,400.,200.,700.,900.,166./
      CALL INITT(30)
      CALL BINITT
C * CHANGE THE RECTANGULAR DISPLAY AREA
      CALL SLIMX(359,850)
      CALL SLIMY(250,650)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Screen Limit Variation Using
Screen coordinates
Figure 3.18

Subroutine PLACE

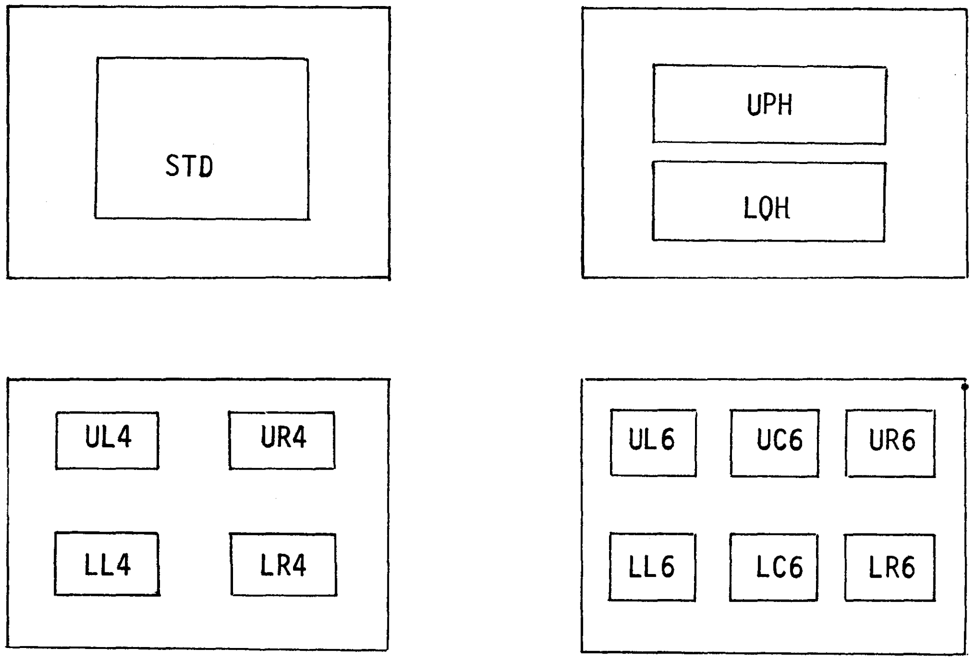
This routine sets the values of the screen minimums and maximums in COMMON to place the screen window at a predefined location on the display area. (See the chart below for the possible locations.)

Calling Sequence:

```
CALL PLACE (LIT)
```

Parameters Entered

'LIT'	is a literal string of three characters specifying the window location desired, i.e., STD, UPH, etc. See sample below. In most systems it will be necessary to enter the literal with a data statement. If preferred, the user may enter the number of the location as shown on the following page. For example a parameter value of 6 will specify window location in the lower lefthand quarter.



EXAMPLES:	STD	Standard	UR4	Upper Right Quarter
	UPH	Upper Half	LC6	Lower Center Sixth

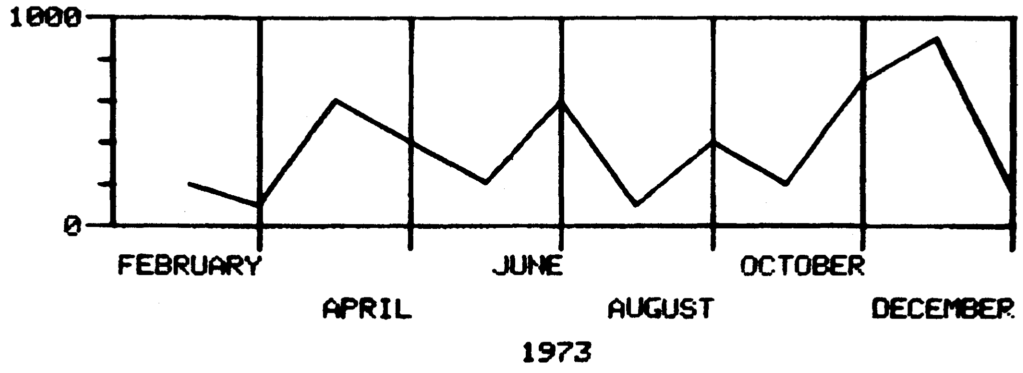
Screen location of each window

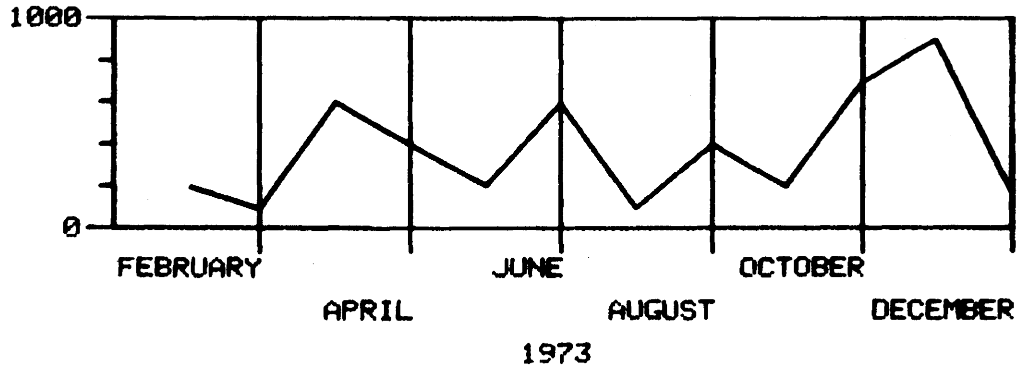
Location	Value	Screen Coordinates			
		XMIN	XMAX	YMIN	YMAX
Standard (STD)	1	150	900	150	700
Upper Half (UPH)	2	150	850	525	700
Lower Half(LOH)	3	150	850	150	325
Upper Left Quarter UL4)	4	150	450	525	700
Upper Right Quarter (UR4)	5	650	950	525	700
Lower Left Quarter (LL4)	6	150	450	150	325
Lower Right Quarter (LR4)	7	650	950	150	325
Upper Left Sixth (UL6)	8	150	325	525	700
Upper Center Sixth (UC6)	9	475	650	525	700
Upper Right Sixth (UR6)	10	800	975	525	700
Lower Left Sixth (LL6)	11	150	325	150	325
Lower Center Sixth (LC6)	12	475	650	150	325
Lower Right Sixth (LR6)	13	800	975	150	325

NOTE

The above Screen coordinate values reflect 1024 addressing. For 4096 addressing the values are multiplied by four.

```
      DIMENSION XARRAY(5),YARRAY(13)
      DATA XARRAY/-2.,12.,12.,1973.,1./
      DATA YARRAY/12.,200.,100.,600.,400.,209.,599.,
     &100.,400.,200.,700.,900.,166./
C * VARIABLES UPH AND LOH ARE 3 ELEMENT LITERALS
      DATA UPH,LOH/3HUPH,3HLOH/
      CALL INITT(120)
      CALL BINITT
C * DISPLAY THE GRAPH IN THE UPPER HALF OF THE SCREEN AREA
      CALL PLACE(UPH)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL BINITT
C * DISPLAY THE GRAPH IN THE LOWER HALF OF THE SCREEN AREA
      CALL PLACE(LOH)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```





Screen Limit Variation Using PLACE
Figure 3.19

3.2.13 Data and Transformation Type -CXTYPE, CYTYPE

These full word real values indicate the types of data that are in the X and Y arrays or the type of transformation to be used.

The acceptable values and external keys are shown in the figure below. Initial values of both CXTYPE and CYTPE are "1", representing normal or linear type data. The user may change the type of graph by calling XTYPE or YTYPE and specifying an alternate type. Label type, CXLAB or CYLAB will be changed internally to be the same as the transformation type.
Note: No remote exponent is used with logarithmic data.

Subroutines XTYPE and YTYPE

This routine is used to define to the system the type of data that is represented in the X or Y array.

Calling Sequence:

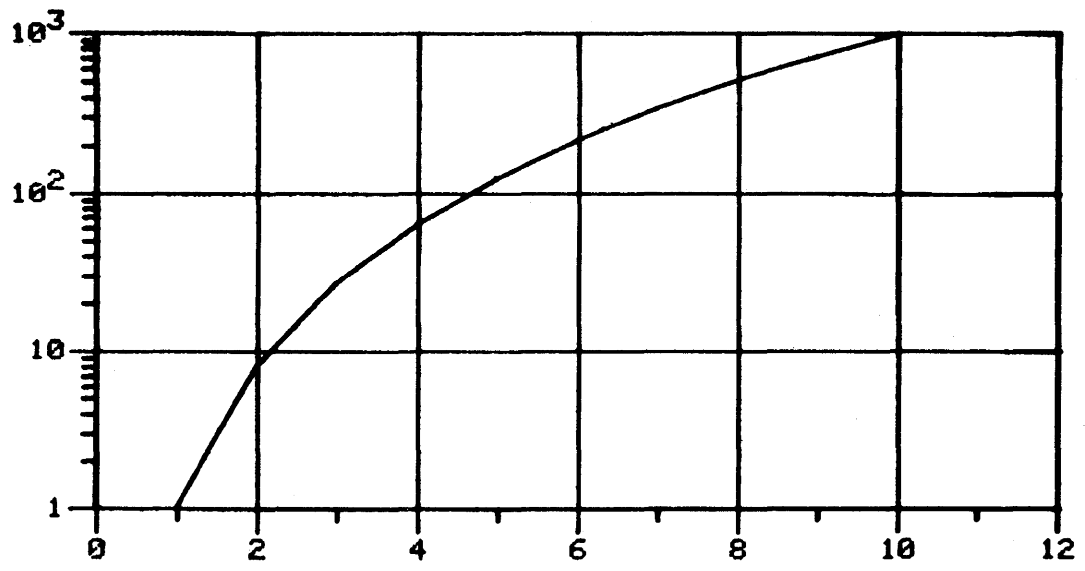
```
CALL XTYPE (IVALUE)
CALL YTYPE (IVALUE)
```

Parameters Entered:

IVALUE	is a coded integer value defining the type of data or the type of transformation to be used. Code values are defined below.

Code Value	Type
1	Normal (Linear) Default
2	Logarithmic (Ration Scale
3	Days
4	Weeks
5	Periods
6	Months
7	Quarters
8	Years

```
      DIMENSION XARRAY(13),YARRAY(11)
      DATA XARRAY/12.,1.,2.,3.,4.,5.,6.,7.,8.,9.,10.,11.,12./
      YARRAY(1)=10
      DO 100 I=2,11
100   YARRAY(I)=FLOAT(I-1)**3
      CALL INITT(30)
      CALL BINITT
      CALL SLIMY(300,700)
C * DATA ON X AXIS WILL BE LINEAR WITH LOG DATA ON THE Y
      CALL XTYPE(1)
      CALL YTYPE(2)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Data Type Variation
Figure 3.20

3.2.14 Least Significant Digit in Labels - CXLSIG, CYLSIG

These real variables specify the position of the least significant digit to be used in computing the width of the tic mark labels on each axis. The user cannot set this value; subroutine OPTIM will set the value when it computes the form of labels to be used for the available width.

A positive value designates a position to the left of the decimal point and a negative value designates a position to the right of the decimal point.

Example: The least significant digits of sample data are specified below.

Data	Least Significant Digit Position
143	+1
3	+1
143.1	-1
143.156	-3

If these four values were all to be used on the same axis, the least significant digit of that axis would be specified as -3.

3.2.15 Tic Mark Label Width - CXWDTH, CYWDTH

These real variables specify the maximum label width (number of characters) usable for tic mark labels on the X and Y axes. This information is used by OPTIM to determine the form in which the tic mark labels will appear (whether a remote exponent will be necessary, etc.).

These values are initialized to Ø, and if the user doesn't assign a value, OPTIM calls subroutine WIDTH which determines the values to be stored in C(X/Y)WDTH.

Subroutines XWDTH and YWDTH

These routines are used to set in COMMON the maximum number of characters allowable for the tic mark labels on the X axis.

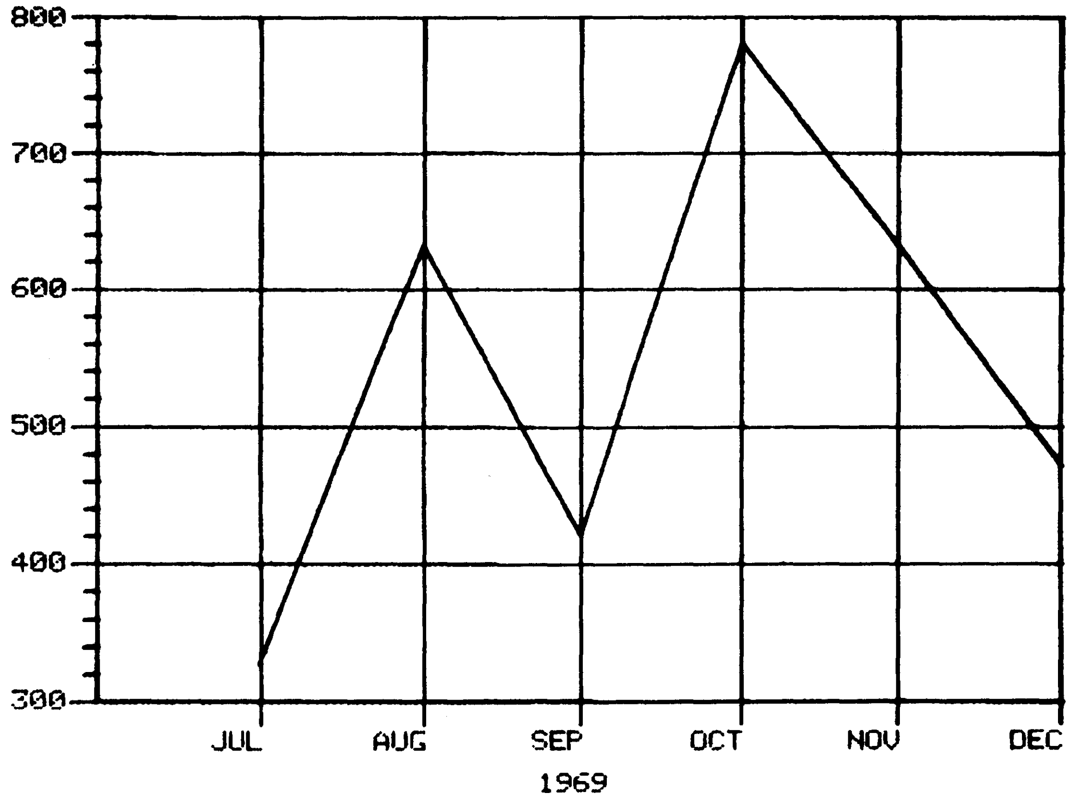
Calling Sequence:

```
CALL XWDTH (IVALUE)
CALL YWDTH (IVALUE)
```

Parameters Entered:

IVALUE	is the maximum number of characters allowable for the tic mark labels.

```
      DIMENSION XDATA(5),YDATA(7)
      DATA XDATA/-2.,6.,12.,1969.,7./
      DATA YDATA/6.,328.,632.,421.,780.,632.,472./
      CALL INITT(120)
      CALL BINITT
C * MAXIMUM LENGTH OF XLABELS WILL BE 3
      CALL XWDTH(3)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Label Length Variation
Figure 3.21

3.2.16 Remote Exponent Value - CXEPON, CYEPON

These full word real variables contain the value of the remote exponent to be used in labels on the respective X or Y axis. These values are set internally by subroutine OPTIM (there are no user routines available to set these values).

3.2.17 Label Frequency - CXSTEP, CYSTEP

These full word real variables designate the frequency of labels on major tic marks on the respective X or Y axis.

These variables are initialized to 1 which designates a label at every major tic mark. (A value of 2 would produce labels at every other tic mark and a value of 3 would produce tic marks at every third tic mark, etc.)

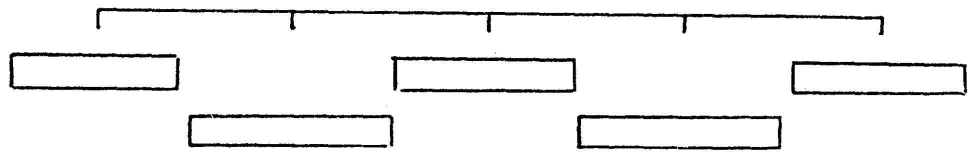
These values may not be set by the user. Subroutine SPREAD takes into consideration the size of the labels and the space available and determines the CXSTEP and CYSTEP values.

Note: The first tic mark is always labeled; thereafter labels appear only at the specific intervals.

3.2. 18 <u>Staggered Labels - CXSTAG, CYSTAG</u>

These full word real variables specify whether or not labels are to be staggered. These values are never user set, but are specificied by subroutine SPREAD which checks the size of the labels and the space available to determine if label staggering is necessary. The Y axis labels are never staggered.

CXSTAG and CYSTAG are initialized to 1 which specifies non-staggered labels. A value of 2 specifies staggered labels in two levels as shown below. Also, see Figure A.7



3.2.19 Remote Exponent Type - CXETYP, CYETYP

These full word real variables specify the type of remote exponent label to be used on the appropriate axis. The remote exponent will appear at the center of the X axis set out slightly farther than the individual tic mark labels, or centered above the Y axis. The various types of exponents available and their code values are described in the following table:

VALUE	RESULTING TYPE
1	$1Ø^n$
2	M, MM's, etc. (1 M per thousand)
3	Printed words, i.e. HUNDREDS
4	1Ø...Ø (1 plus appropriate number of Ø's)

The default value is 1.

If the label type is logarithmic, this variable specifies the form of the major tic mark labels. No remote exponents are formed for logarithmic axes.

Subroutines XETYP, YETYP

These routines are called to set the code value of the form of remote exponent label to be used on the appropriate axis.

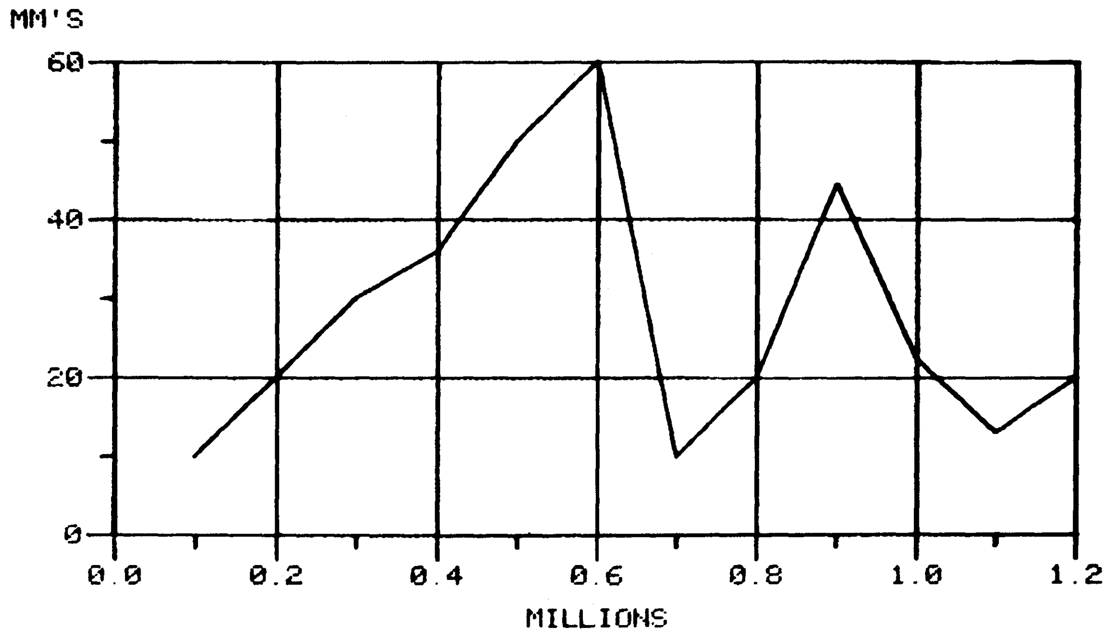
Calling Sequence:

```
CALL XETYP (IVALUE)
CALL YETYP (IVALUE)
```

Parameters Entered:

IVALUE	is the integer code value which specifies the form of remote exponenet to be used. See the table above.

```
      DIMENSION XARRAY(13),YARRAY(13)
      DATA XARRAY/12.,100000.,200000.,300000.,400000.,500000.,
     &600000.,700000.,800000.,900000.,1000000.,1100000.,1200000./
      DATA YARRAY/12.,10000000.,20000000.,30000000.,35900000.,50000000.,
     &60000000.,10000000.,20000000.,44400000.,22200000.,13000000.,
     &20000000./
      CALL INITT(30)
      CALL BINITT
      CALL SLIMY(200,600)
C * SET THE REMOTE LABEL EXPONENT TYPE
      CALL XETYP(3)
      CALL YETYP(2)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Remote Exponent Type
Figure 3.22

3.2.20 Major Tic Mark Starting and Ending Locations -

CXBEG, CXEND, CYBEG, CYEND

These full word real variables indicate the beginning (CXBEG and (CYBEG) and the ending (CXEND and CYEND) locations for the major tic marks. The starting location for the X axis for example, indicates the height on the screen at which the X axis major tic marks will begin.

The initial value of these variables is Ø. There are no user called routines to set these variables; they are set internally.

3.2.21 Calculated Data Minimums and Maximums

CXAMIN, CXAMAX, CYAMIN, CYAMAX

These full word real variables contain the calulated or adjusted data minimums and maximums from which tic mark values will be determined. These values are determined by subroutine OPTIM. (See Section 4.1.10 .)

There are no user called routines to set these variables; they are set internally only.

3.2.22 Calculated Data Minimums and Maximums

CXAMIN, CXAMAX, CYAMIN, CYAMAX

These full word real variables contain the calulated or adjusted data minimums and maximums from which tic mark values will be determined. These values are determined by subroutine OPTIM. (See Section 4.1.10)

There are no user called routines to set these variables; they are set internally only.

SECTION 4

ADVANCED SYSTEM USAGE

4. Advanced System Usage

For the programmer who wishes to create more advanced graphs or to set up a program to create regularly updated graphs, many of the routines in this section may prove useful.

4.1 Special Purpose Routines

Additional routines are provided which are used internally but may be called by the user for special purposes; others are simply provided for the convenience of the user. These include subroutines to save and restore a COMMON Table, and routines which create portions of a graph - the symbols, labels, grid, etc.

4.1.0 Determining the release or level of Software - AG2LEV

This subroutine returns the release or level number and last modification date of the software.

Calling Sequence

```
CALL AG2LEV (LEVEL)
```

Parameters Entered:

NONE

Parameters Returned

LEVEL	A coded 6 digit integer containing the following information:	
	First 2 digits:	year of last modification
	Next 3 digits:	Julian day of last modification
	Last 1 digit:	release number, 1

Description:

AG2LEV allows the user to obtain the release or level of his software so that he may determine exactly which features are present. The value returned should be updated in the source code when the package is updated.

4.1.1 Setting Data Limits - MNMX

This subroutine determines the minimum and maximum values of a data array.

Calling Sequence:

```
CALL MNMX (ARRAY,AMIN,AMAX)
```

Parameters Entered:

ARRAY	is the data array for which the minimum and maximum are to be found.

Parameters Returned:

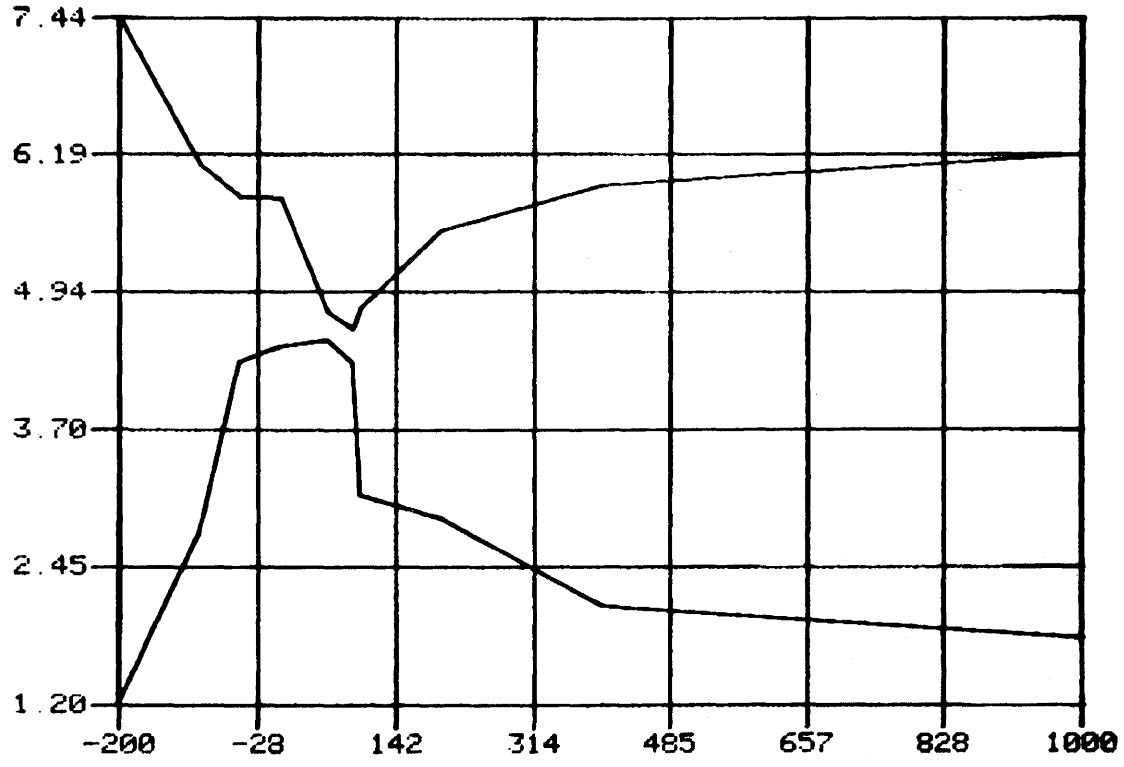
AMIN	is the minimum array value.
AMAX	is the maximum array value.

Description:

MNMX is used internally to determine data limits, and may be useful to the user in setting the minimum and maximum values for multiple overlapping curves. It is first necessary to set AMIN to an extremely large value and AMAX to an extreme negative value. Then, MNMX must be called once for each data array. Each time, the AMIN is set to be the smaller of the new and the old values, and AMAX to be the larger of the new and the old values. After MNMX has checked each data curve, DLIMX or DLIMY must be called with the new AMIN, AMAX values to set the limits in COMMON.

Note: If a data value is set at CINFIN, it will be disregarded. That is, if a user wishes to plot data, for which some values are missing, he can set the missing values at "infinity", and those points will be skipped.

```
      DIMENSION XARRAY(11),YARRAY(11),ZARRAY(11)
      DATA XARRAY/10.,-200.,-100.,-50.,0.,60.,90.,100.,199.,400.,1000./
      DATA YARRAY/10.,1.2,2.76,4.31,4.45,4.5,4.3,3.1,2.9,2.1,1.8/
      DATA ZARRAY/10.,7.44,6.1,5.81,5.8,4.75,4.6,4.8,5.5,5.9,6.2/
      CALL INITT(120)
      AMIN=1000.
      AMAX=-100000.
      BMIN=10000.
      BMAX=-10000.
      CALL BINITT
      CALL XNEAT(0)
      CALL YNEAT(0)
C * FIND MINIMUM AND MAXIMUM DATA VALUES
      CALL MNMX(XARRAY,AMIN,AMAX)
      CALL MNMX(YARRAY,BMIN,BMAX)
      CALL MNMX(ZARRAY,BMIN,BMAX)
      CALL DLIMX(AMIN,AMAX)
      CALL DLIMY(BMIN,BMAX)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL CPLOT(XARRAY,ZARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      END
```



Use of MNMX with a Multiple Curve

Figure 4.1

4.1.2 COMMON Table Saving Subroutine - SAVCOM

This routine saves the COMMON table as it currently exists.

Calling Sequence:

```
CALL SAVCOM (ARRAY)
```

Parameters Entered:

ARRAY	is the name assigned to the array of values from COMMON. This array should be dimensioned to 80.

Description:

SAVCOM allows the COMMON table to be saved, so that additional curves may be plotted using the same graph from without calling CHECK each time.

See Figure 4.2.

4.1.3 COMMON Restoring Subroutine - RESCOM

This subroutine restores COMMON values which have previously been saved.

Calling Sequence:

```
CALL RESCOM (ARRAY)
```

Parameters Entered:

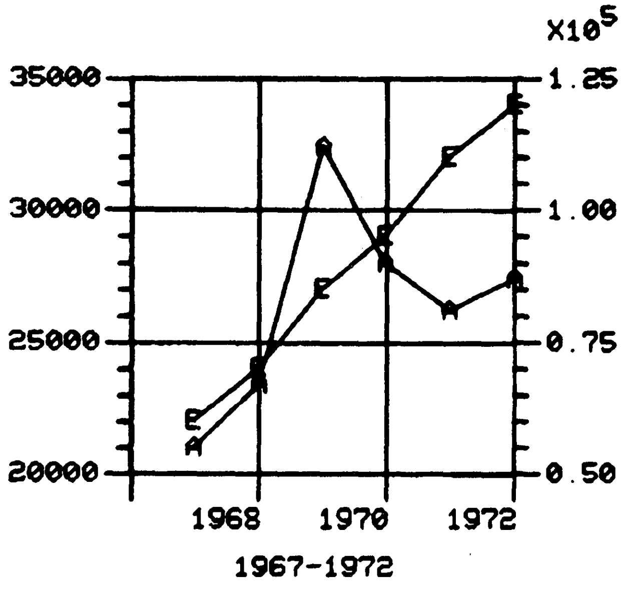
ARRAY	is the array of COMMON values previously saved with SAVCOM. This array should be dimensioned to 80.

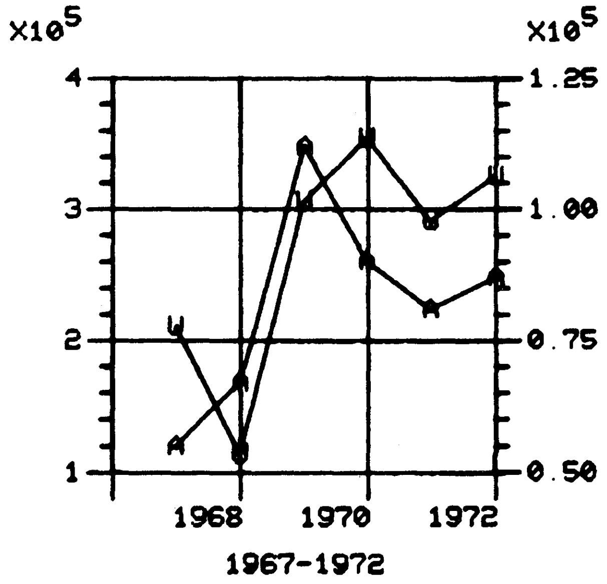
Description:

RESCOM restores COMMON to the values saved with SAVCOM. This eliminates the steps involved in checking the table each time a new graph is to be displayed.

See Figure 4.2.

```
C * THIS ROUTINE ILLUSTRATES THE USE OF SAVCOM AND RESCOM
      DIMENSION EARRAY(7),WARRAY(7),AARRAY(7),TARRAY(5),STAT(80),ST(80)
      DATA EARRAY/6.,22000.,24000.,27000.,29000.,32000.,34000./
      DATA WARRAY/6.,211000.,114000.,306000.,354000.,291000.,325000./
      DATA AARRAY/6.,55000.,67000.,112000.,90000.,81000.,87000./
      DATA TARRAY/-2.,6.,1.,1967.,1./
      CALL INITT(30)
      CALL BINITT
      CALL SLIMX(200,500)
      CALL SLIMY(200,500)
      CALL SAVCOM(ST)
      CALL SYMBL(69)
      CALL CHECK(TARRAY,EARRAY)
      CALL DSPLAY(TARRAY,EARRAY)
      CALL DINITY
      CALL XFRM(0)
      CALL YFRM(2)
      CALL SYMBL(65)
      CALL YLOCRT(0)
      CALL CHECK(TARRAY,AARRAY)
      CALL XLAB(0)
      CALL SAVCOM(STAT)
      CALL DSPLAY(TARRAY,AARRAY)
      CALL TINPUT(IVY)
      CALL RESCOM(ST)
      CALL SYMBL(87)
      CALL CHECK(TARRAY,WARRAY)
      CALL DSPLAY(TARRAY,WARRAY)
      CALL RESCOM(STAT)
      CALL DSPLAY(TARRAY,AARRAY)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
```





Save and Restore COMMON
Figure 4.2

4.1.4 Symbol Drawing Subroutine - BSYMS

This routine draws symbols for data points when required.

Calling Sequence:

```
CALL BSYMS (X,Y,ISYM)
```

Parameters Entered:

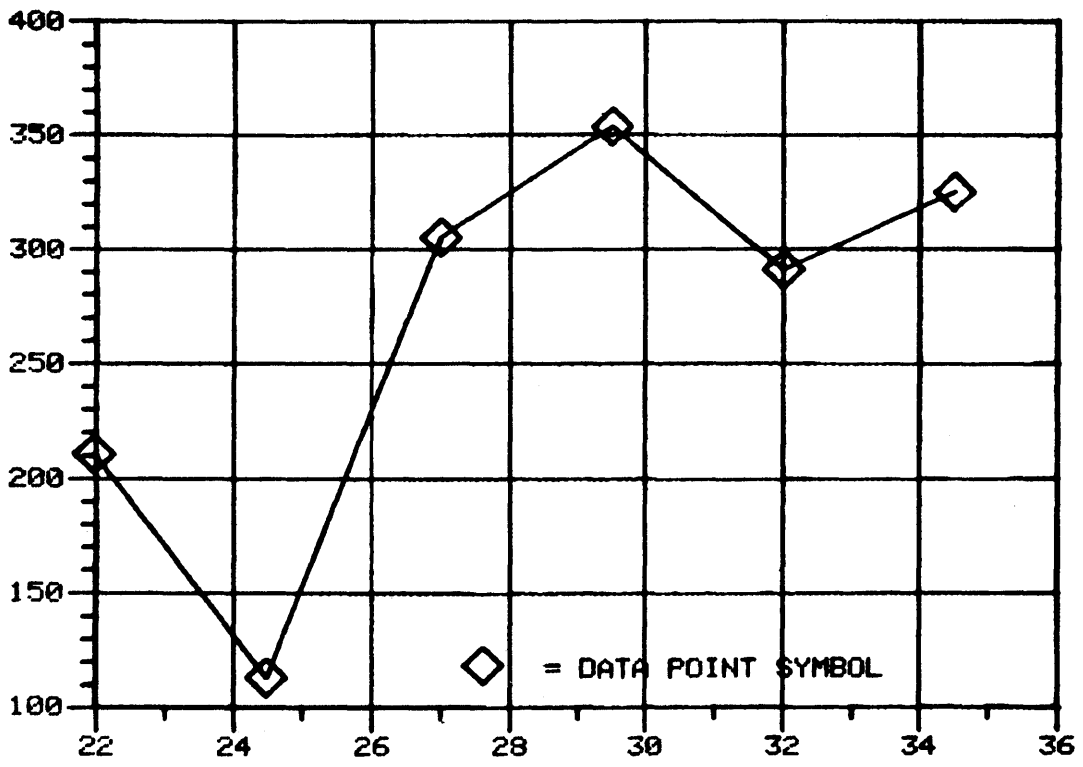
X, Y	are the virtual coordinates for the data point at which a symbol is to be drawn.
ISYM	is the value of the symbol type. See the chart on page 3-9.

Description:

BSYMS obtains the data point location and the symbol type and draws the symbol. It is used internally by CPLOT to draw the data point symbols, or may be called by the user if he wishes to draw symbols elsewhere.

NOTE: BSYMS draws the symbol at the current beam position; that is, BSYMS does not move to X, Y before drawing the symbol. X, Y is used to determine if the symbol location is inside the virtual window; if not, the symbol will not be drawn. This routine is intended for internal system use. The user is therefore encouraged to use subroutine SYMOUT for placing symbols and legends.

```
      DIMENSION ISTRIN(19)
      DIMENSION XARRAY(4),YARRAY(7)
      DATA XARRAY/-1.,6.,22.,2.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      DATA ISTRIN/61,32,68,65,84,65,32,80,79,73,78,84,
     &32,83,89,77,66,79,76/
      CALL INITT(120)
      CALL BINITT
      CALL SIZES(2)
      CALL SYMBL(6)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL MOVEA(27.5,120.)
C *  PLACE A SAMPLE OF THE SYMBOL WITHIN THE GRAPIC AREA
      CALL BSYMS(27.5,120.,6)
      CALL MOVABS(450,160)
      CALL HLABEL(19,ISTRIN)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
```



Using BSYMS

Figure 4.3

Symbol Output Subroutine - SYMOUT

This routine produces standard symbols at the current beam location. The beam is left at that location upon return. No clipping or windowing is done by this routine. The characters (exclusive of ASCII characters) may be scaled by a multiplicative factor; that is , a factor of 2 will produce a symbol two times the normal size.

Calling Sequence:

```
CALL SYMOUT(ISYM,FACTOR)
```

Parameters Entered:

ISYM	is the value of the symbol type. See the chart on page 3-9.
FACTOR	is the scaling factor of the symbol type (exclusive of hardware generated ASCII characters).

Bar Drawing Subroutine - BAR

This routine draws the bars for a bar chart.

Calling Sequence:

```
CALL BAR (X,Y,LINE)
```

Parameters Entered:

X,Y	are the screen coordinates to which the bar is to be drawn.
LINE	is the direction in which the line is to be drawn. 0 draws a bar similar to the last one drawn. 2 draws vertical bars and accesses COMMON for shading. 3 draws horizontal bars and accesses COMMON for shading.

Description:

BAR obtains all the information necessary for drawing and shading the bars for a bar chart. To do this it obtains CSIZEL for the width of the bar and CSIZES for the distances between shading lines. It then draws the bar and calls FILBOX to shade it.

4.1.5 Bar Chart Shading Subroutine - FILBOX

This routine shades in the bars of a bar chart.

Calling Sequence:

```
CALL FILBOX(MINX,MINY,MAXX,MAXY,ISYMB,LSPACE)
```

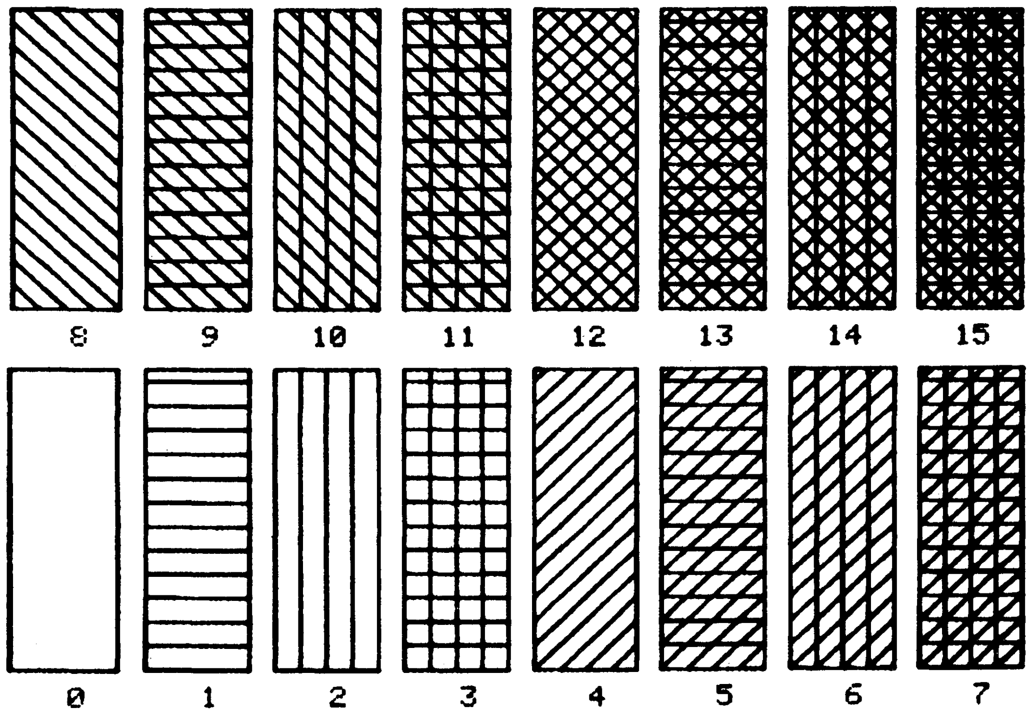
Parameters Entered:

MINX	is the X axis screen minimum of the bar.
MINY	is the Y axis screen minimum of the bar.
MAXX	is the X axis screen maximum of the bar.
MAXY	is the Y axis screen maximum of the bar.
ISYMB	is an integer symbol code from Ø through 15. It indicates the type of line shading to fill the box. (See Section 2.8)
LSPACE	is the distance in raster units between the lines shading the bar.

Description:

FILBOX fills each bar of the chart with the specified grid work for shading.

```
C * THIS ROUTINE TESTS FILBOX ROUTINE
      DIMENSION IARRAY(15)
      CALL INITT(120)
      CALL BINITT
      ISYMB=0
      DO 100 J=1,2
      MINY=100+300*(J-1)
      MAXY=MINY+250
      DO 100 I=1,8
      MINX=50+100*(I-1)
      MAXX=MINX+80
C * FILL THE BOXES WITH APPROPRIATE DESIGN
      CALL FILBOX(MINX,MINY,MAXX,MAXY,ISYMB,20)
      SYMB=FLOAT(ISYMB)
      CALL FFORM(SYMB,3,0,IARRAY,32)
      CALL MOVABS(MINX+30,MINY-30)
      CALL HLABEL(15,IARRAY)
      ISYMB=ISYMB+1
100   CONTINUE
      CALL TINPUT(KEY)
      STOP
      END
```



FILBOX

Figure 4.4

4.1.6 Tic Mark Labeling Subroutine - LABEL

This routine displays the tic mark labels on the graph.

Calling Sequence:

```
CALL LABEL (NBASE)
```

Parameters Entered:

NBASE	is the axis pointer, i.e. the location in COMMON of the first item referring to the X or Y axis, whichever is here indicated. (Function IBASEX or IBASEY provides this location. See Sections 4.4.5 and 4.4.6.)

Description:

LABEL obtains the appropriate axis values from COMMON for the number of tic marks, tic mark length and form, data minimum and maximum, screen minimum and maximum, type and width of labels, and remote exponent (if any), frequency of labels and whether staggered. It then calculates the labels, converts them to alphanumerics with NUMSET, justifies and displays them with NOTATE. If remote labels are used, LABEL calls REMLAB.

4.1.7 <u>Grid Drawing Subroutine - GRID</u>

This routine draws the axis lines, tic marks, and grid lines.

Calling Sequence:

```
CALL GRID
```

Parameters Entered:

None

Description:

GRID obtains the screen minimums and maximums, the axis location and tic mark form and number. It then draws the axis lines and the tic mark and grid lines.

4.1.8 Window Setting Subroutine - SETWIN

This subroutine transfers the window location from virtual to screen coordinates.

Calling Sequence:

```
CALL SETWIN
```

Parameters Entered:

None

Description:

SETWIN is a transformational routine which transfers the window size and location from virtual space to screen space. DSPLAY calls SETWIN before drawing the graph.

4.1.9 Label Width Subroutine - WIDTH

This subroutine calculates and stores in COMMON the width of the longest label on the given axis.

Calling Sequence:

```
CALL WIDTH (NBASE)
```

Parameter Entered:

NBASE	is the axis pointer, or the location in COMMON of the first item referring to the X or Y axis, whichever is here indicated. (IBASEX or IBASEY provides this location.)

Description:

This routine checks the data minimum and maximum for the given axis, the position of the least significant digit, and the type of label; it then calculates the number of characters necessary for the longest label on the axis. The number of characters computed is then stored in COMMON variable CXWDTH or CYWDTH.

4.1.10 Optimum Tic Mark Subroutine - OPTIM

Using the data from COMMON, this routine produces results necessary for appropriately spaced and labeled tic marks.

Calling Sequence:

```
CALL OPTIM (NBASE)
```

Parameters Entered:

NBASE	is the axis pointer to the location of the first item in COMMON which refers to the X or to the Y axis. (Functions IBASEX and IBASEY provide this location.)

Description:

If any necessary COMMON values have not been set, appropriate values will be obtained through this routine.

If the data range in COMMON is Ø, data will be scanned to find the minimum and maximum, and thus the range. Zero suppression (suppression of zero as a tic mark value unless the data approaches zero) is checked only if data scanning is necessary.

If the number of tics is not specified, the number is chosen based on the length of the axis and the density factor. The distance between the resulting tic marks will be no less than 50 raster units or more than 150 raster units.

Note: The density factor is a value in COMMON designating the desired density of tic marks. Values Ø through 5, and 6 through 1Ø create tic marks ranging from sparse (Ø or 6) to dense (5 or 1Ø) with values Ø through 5 omitting minor tic marks.

If the number of tics has been user specified, density factor is ignored.

The interval between tic mark values will be calculated. If neat tic marks have been requested, the label values will be rounded to neat intervals. Data minimum and maximum will be adjusted to coincide with label values. If the number of tics has not been specified, it may be altered at this time to fit the plot better; if the number of tic marks was user specified, tic mark labels and data limits will be adjusted to fit.

The position of the least significant digit in the tic mark label is calculated and this information utilized by WIDTH to select the number of digits to be displayed as part of the label. A remote exponent will be used, displayed as a general axis label, if the complete label will not fit.

If the number of minor tics has not been selected and the density factor is six or greater, the number of minor tics will be selected based on the tic mark label interval.

Note: The list of neat values used to compute neat major and minor tic mark values may be reprogrammed by the user.

4.2 Number Setting Routines

A group of routines called by NUMSET change floating point numbers into character strings which can be displayed as tic mark labels. Their interaction is shown in Figure 4.5.

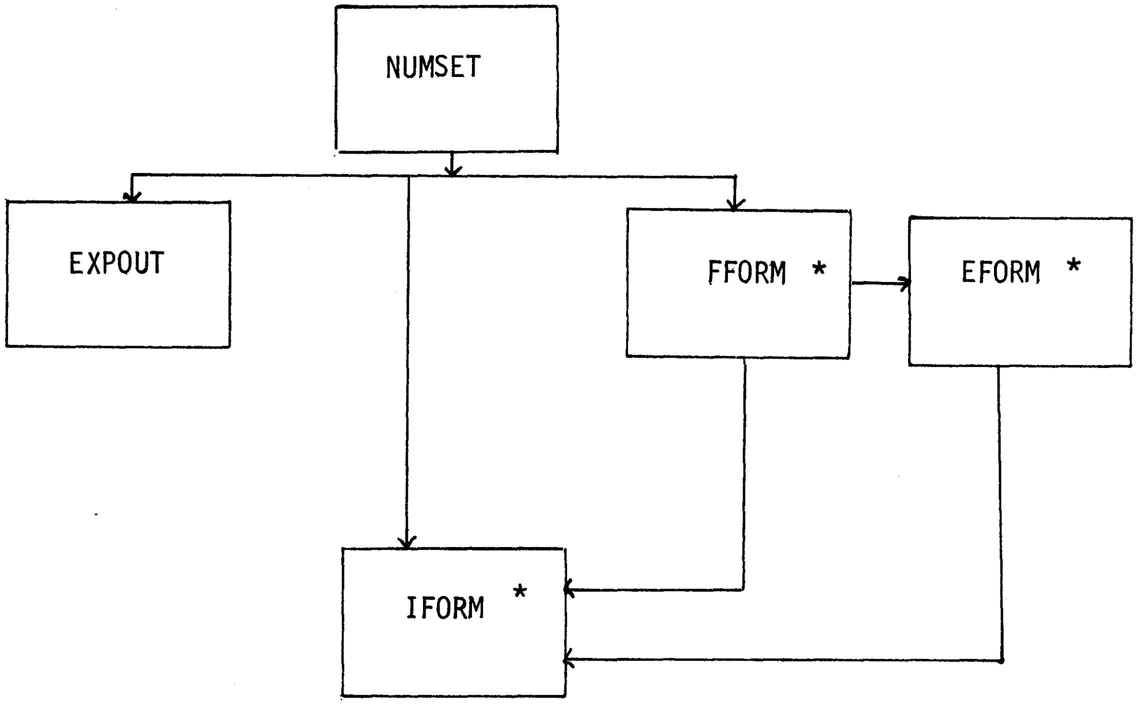


Figure 4.5

Number Handling Routines Utilized by NUMSET

* These routines may be used outside of Advanced Graphing II as they do not refer to the Global Table.

4.2.1 Subroutine NUMSET

This subroutine changes floating point numbers into character strings which can later be displayed as tic mark labels.

Calling Sequence:

```
CALL NUMSET (FNUMBER,IWIDTH,NBASE,IARRAY,IFILL)
```

Parameters Entered:

FNUMBER	is the floating point number to be converted.
IWIDTH	is the length of the resulting string, including fill characters.
NBASE	is the axis pointer, i.e., the location in COMMON of the first item referring to the X or Y axis, whichever is here indicated. (IBASEX or IBASEY provides this location.)
IFILL	is the character to be used as filler in any extra spaces to the left of the array (all numbers are right justified).

Parameters Returned:

IARRAY	is the string of characters returned in ASCII decimal equivalents.

NUMSET also uses the values from COMMON designating the number of decimal places required, the type of labels, and the exponent type if a logarithmic axis is involved.

NUMSET utilizes a group of other subroutines in determining the final form for the tic mark labels. The three routines which it calls directly are: IFORM, FFORM, and EXPOUT, descriptions of which are included in detail in following pages.

If the axis being considered is not logarithmic, NUMSET will call either

1. IFORM (if number is integer) which will result in integers without decimal points, or

2. FFORM (if real number) which results in ASCII decimal equivalents of a number with decimal points.

If the axis is logarithmic, NUMSET calls EXPOUT, resulting in exponential labels.

4.2.2 Subroutine EFORM

This routine converts a real value into an exponential character string.

Calling Sequence:

```
CALL EFORM (FNUMBER,IWIDTH,IDECIMAL,IARRAY,IFILL)
```

Parameters Entered:

FNUMBER	is the floating point number to be converted.
IWIDTH	is the length of the resulting string, including fill characters.
IDECIMAL	is the number of digits to follow the decimal point.
IFILL	is the character to be used as filler in any extra spaces to the left of the array (all numbers are right justified). Normally a Fill character of blank is used (ASCII [32]).

Parameters Returned:

IARRAY	is the string to characters returned in ASCII decimal equivalents.

Description:

EFORM utilizes subroutine ESPLIT to divide the real number into multiplier and exponent, based on the width designated and the number of digits following the decimal point.
The multiplier is then converted into a character string and IFORM converts the exponent into a character string.

4.2.3 Subroutine FFORM

This subroutine converts a real value into a character string which provides either a floating point number or a number with a decimal point, depending on which form will best fit the given width.

Calling Sequence:

```
CALL FFORM (FNUMBER,IWIDTH,IDECIMAL,IARRAY,IFILL)
```

Parameters Entered:

FNUMBER	is the floating point number to be converted.
IWIDTH	is the length of the resulting string, including fill characters.
IDECIMAL	is the number of digits to follow the decimal point.
IFILL	is the character to be used as filler in any extra spaces to the left of the array (all numbers are right justified as output in IARRAY). Normally IFILL is a Blank (32).

Parameters Returned:

IARRAY	is the string of characters returned in ASCII-decimal equivalents.

Description:

If the number will fit the given width, it is converted into a character string with a decimal. If it will not fit the given width, control is transferred to EFORM.

A minus sign (-) precedes the value if it is negative.

Example: See Section 2.13.4.

4.2.4 Subroutine IFORM

This routine converts a floating point number to a character string without a decimal point.

Calling Sequence:

```
CALL IFORM (FNUMBER,IWIDTH,IARRAY,IFILL)
```

Parameters Entered:

FNUMBER	is the floating point number to be converted.
IWIDTH	is the length of the resulting string, including fill characters.
IFILL	is the character to be used as filler in any extra spaces to the left of the array (all numbers are right justified at this time). Normally a FILL character of blank is used (ASCII [32]).

Parameters Returned:

IARRAY	is the string of characters returned in ASCII decimal equivalents.

Description:

If the integer is negative, a sign is placed in the position immediately preceding the number.

4.2.5 Subroutine EXPOUT

This routine constructs remote scale factors in various set forms or the labels for logarithmic tic marks.

Calling Sequence:

```
CALL EXPOUT (NBASE,IEXP,IARRAY,LENGTH,IFILL)
```

Parameters Entered:

NBASE	is the axis key or the location in COMMON of the first item referring to the X or Y axis, whichever is here indicated. (IBASEX or IBASEY provides this location.)
IEXP	is the integer value that defines to what power 1Ø is raised. This is necessary for either scale factors or logarithmic tic marks. (In 10^6, 6 is the integer exponent.)
LENGTH	is the integer length of the array.
IFILL	defines the character used to fill in the blanks (all labels are right justified). ASCII value 32 representing a blank is usually used.

Parameters Returned:

IARRAY	is the array in which the label is returned in ASCII decimal equivalents.

Description:

EXPOUT takes the integer supplied to it and translates it into a character string which can be justified. From COMMON it obtains the type of exponent (CXETYP or CYETYP) and the type of labels (CXLAB or CYLAB). The results are one character per word in a right justified array.

Logarithmic graphs never have a remote exponent.

4.3 Universal Business Graphing Calendar

The irregularities of the modern day calendar make the computer plotting of calendar data extremely difficult. In addition to the calendar irregularities are the irregularities of accounting periods as devised by various businesses.

To allow for these variations, the UBGC (Universal Business Graphing Calendar) has been developed to consider January 1, 1901, as day 1, and to number each day thereafter sequentially from day 1. By conversion of calendar dates to UBGC days, we can allow for such irregularities as daily data plotted against monthly labels, or any number of accounting periods per year.

The following routines provide for conversions from calendar dates into UBGC days or vice versa, and from year-day format to year-month-day or vice versa.

4.3.1 Conversion to UBGC Value - IUBGC

This subroutine is used for converting the year and day into a Universal Business Graphing Calendar value. The UBGC date is the number of days since December 31, 1900, e.g., UBGC of January 1, 1901, is 1.

Calling Sequence:

```
CALL IUBGC (IYEAR,IDAYS,IUBGCX)
```

Parameters Entered:

IYEAR	A four digit integer value year (e.g. 1972) representing the year to be converted.
IDAYS	A three digit integer value representing the day of the year.

Parameters Returned:

IUBGCX	An integer value returned by the subroutine representing the Universal Business Graphing Calendar value.

4.3.2 Conversion from UBGC Value - OUBGC

The purpose of the subroutine OUBGC is to convert from the Universal Business Graphing Calendar date to the appropriate year and day.

Calling Sequence:

```
CALL OUBGC (IYEAR,IDAYS,IUBGCX)
```

Parameters Entered:

IUBGCX	represents the integer Universal Business Graphing Calendar value.

Parameters Returned:

IYEAR	is an integer variable, containing the year in which the entered date (IUBGCX) lies.
IDAYS	is an integer variable representing the day of the year.

4.3.3 Year Day Conversion - YDYMD

This routine converts dates from year, day format (YYYYDDD) to year, month, day format (YYYYMMDD), or from Julian to Gregorian.

Calling Sequence:

```
CALL YDYMD(IYEAR,IDAYS,IYEAR2,MONTH,IDAY)
```

Parameters Entered:

IYEAR	is a four digit year.
IDAYS	is a three digit day of the year value.

Parameters Returned:

IYEAR2	is the year.
MONTH	is the month of the year
IDAY	is the day of the month

Description:

If IDAYS is outside the domain of 1 to 366, IYEAR2 will differ from IYEAR.

4.3.4 Year Month Day Conversion - YMDYD

This routine converts dates from the year, month, day format (YYYYMMDD) to a year, day format (YYYYDDD), a Gregorian to Julian conversion.

Calling Sequence:

```
CALL YMDYD (IYEAR,IDAYS,IYR,MONTH,IDAY)
```

Parameters Entered:

IYR	is a four digit year to be converted.
MONTH	is the month passed to the routine.
IDAY	is the day of the month.

Parameters Returned:

IYEAR	is the year
IDAYS	is the day of the year

Description:

If MONTH is outside the domain of 1 to 12, IYR will differ from IYEAR.

4.4 General COMMON Setting and Access Routines

Some users may prefer to use one single subroutine to set any variable in COMMON. Subroutine COMSET, used with functions IBASEC, IBASEX, and IBASEY to specify the item being set, provides this option.

The IBASE functions may be thought of as COMMON section pointers, as illustrated in the diagram below. The actual variables found in each section are shown with the function descriptions on pages

The table of variables and their locations follows.

AG-II COMMON

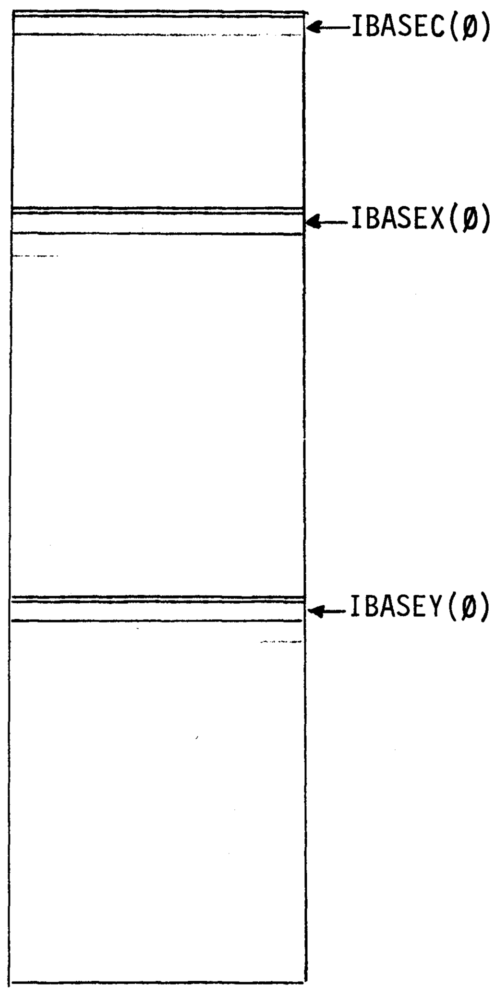


TABLE II

Advanced Graphing II Common Variables

SECTION I. DATA CURVE VARIABLES
IBASE Value

Location	Variable	Description	Initial Value
Ø	CLINE	Type of line used in plots	Ø.
1	CSYMBL	Symbol used for point plots	Ø.
2	CSTEPS	Increment between symbols	1.
3	CINFIN	Infinity	1.E30
4	CNPTS	Number of points for non-standard array	Ø.
5	CSTEPL	Increment between points in lines	1.
6	CNUMBR	Actual number of points in this plot	Ø.
7	CSIZES	Symbol size - Shading density on bars	1.
8	CSIZEL	Line size - Bar width	1.

SECTIONS II AND III. AXIS VARIABLES [1]
IBASEX/IBASEY Value

Location	Variable	Description	Initial Value
Ø	CXNEAT CYNEAT	Neat tic mark flag	1.(ON)
1	CXZERO CYZERO	Zero suppression flag	1.(ON)
2	CXLOC CYLOC	Location of axis	Ø.
3	CXLAB CYLAB	Type of labels	1.
4	CXDEN CYDEN	Density of tic marks	8.

[1]Sections II and III (X Axis Variables and Y Axis Variables) have been combined in this chart for ease of reference.

TABLE II (continued)

Advanced Graphing II Common Variables

Location	Variable	Description	Initial Value
		SECTIONS II AND III. AXIS VARIABLES	
5	CXTICS CYTICS	Number of major tic mark intervals	Ø.
6	CXLEN CYLEN	Length of major tic marks	Vertical height of current character size
7	CXFRM CYFRM	Form of major tic marks	5.
8	CXMTCS CYMTCS	Number of minor tic mark intervals	Ø.
9	CXMFRM CYMFRM	Form of minor tic marks	2.
10	CXDEC CYDEC	Number of decimal places in labels	Ø.
11	CXDMIN CYDMIN	Data Minimum	Ø.
12	CXDMAX CYDMAX	Data Maximum	Ø.
13	CXSMIN CYSMIN	Screen minimum	150,125 (1024 addressing)
14	CXSMAX CYSMAX	Screen maximum	900,700 (1024 addressing)
15	CXTYPE CYTYPE	Type of data	1.
16	CXLSIG CYSLIG	Least significant digit	Ø.
17	CXWDTH CYWDTH	Maximum label width	Ø.
18	CXEPON CYEPON	Exponent	Ø.

(continued)

TABLE II (continued)

Advanced Graphing II Common Variables

Location	Variable	Description	Initial Value
	SECTION II AND III. AXIS VARIABLES		
19	CXSTEP CYSTEP	Number of label steps	1.
20	CXSTAG CYSTAG	Type of label form	1.
21	CXETYP CYETYP	Exponent Type	Ø.
22	CXBEG CYBEG	Start location for tic marks	Ø.
23	CXEND CYEND	Ending location for tic marks	Ø.
24	CXMBEG CYMBEG	Starting location for tic marks	Ø.
25	CXMEND CYMEND	Ending location for minor tic marks	Ø.
26	CXAMIN CYAMIN	Calculated data Minimum	Ø.
27	CXAMAX CYAMAX	Calculated data Maximum	Ø.

The location values are useful when using COMSET and COMGET which may refer to any of these variables.

4.4.1 COMMON Section Pointers

All access to the global COMMON table is via a number of intermediate functions and subroutines. This assures flexibility in future versions of the package. Any one accessing COMMON by any method other than the one described in this section is running a risk of being incompatible with future releases.

Note: Most users will never need to access COMMON and will NOT need to read this section.

Function IBASEC (page 169) is a pointer to the first variable in the data curve section of COMMON. The function parameters are values from Ø to 5, designating the distance (the number of full words) the pointer should advance to locate the desired variable. For example, IBASEC(2) locates the variable CSTPL. IBASEX and IBASEY are pointers to the beginning variables in the X axis section and the Y axis section respectively. Since the same kinds of variables are necessary for both the X and the Y axes, the names and descriptions of these variables are combined into a single reference table. (See Table II on page 4-30.)

Internally, the same routines are often used to compute information for either the X or the Y axis. (This is true of all the axis labeling routines such as OPTIM, TSET and WIDTH.) Therefore, in these routines, the variable NBASE is used as the axis pointer designating the value in COMMON to be used for the computation. A higher level routine (such as CHECK) can then set NBASE to be either IBASEX or IBASEY, depending on which axis is being computed.

For example, NBASE+1 may refer to either CXZERO (X axis zero suppression variable) or CYZERO (Y axis zero suppression variable), depending on which function (IBASEX or IBASEY) is substituted for NBASE.

A set of five subroutines and functions are used to access COMMON in this manner.

4.4.2 General COMMON Setting Routine - COMSET

COMSET is a user called routine which sets the value of any item in COMMON. Its two arguments designate the location of the item in COMMON to be changed and the new value to be assigned.

Calling Sequence:

```
CALL COMSET (ITEM,VALUE)
```

Parameters Entered:

ITEM	is the number of word in COMMON to be set or changed.
VALUE	is the real value to be stored there.

See Example 3 on page 4-40

```
      DIMENSION IARRAY(10),VALUE(3)
      CALL INITT(30)
      CALL BINITT
C * OBTAIN THE LOCATION OF CLINE IN COMMON
      VALUE(1)=FLOAT(IBASEC(0))
C * OBTAIN THE VALUE OF CLINE
      VALUE(2)=COMGET(IBASEC(0))
C * CHANGE CLINE CONTENTS TO-2 (THE BAR CHART OPTION)
      CALL COMSET(IBASEC(0),2.)
C * OBTAIN THE NEW VALUE OF CLINE
      VALUE(3)=COMGET(IBASEC(0))
C * PRINT THE LOCATION OF CLINE, THE OLD VALUE AND THE NEW VALUE
C * OF CLINE.  USE AG-II NUMBER OUTPUT ROUTINES.
      DO 100 I=1,3
      CALL FONLY(VALUE(I),10,3,IARRAY,32)
      CALL MOVABS(0,750-I*50)
100   CALL HLABEL(10,IARRAY)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
```

```
8.000

1.000

2.000
```

Use of COMSET and COMGET
Figure 4.6

4.4.3 <u>COMMON Access Function - COMGET</u>

If the user wants to know the value assigned to an item in COMMON at any give time, the function COMGET provides it.

Calling Sequence:

```
COMGET (ITEM)
```

Parameters Entered:

ITEM	is the number of the word in COMMON for which we want to know the value.

In this function VALUE=COMGET(ITEM).

The internal location of an item in COMMON is found with the assistance of functions IBASEC, IBASEX, and IBASEY. These three functions coincide with the three sections of COMMON as described earlier.

See Examples 1 and 2 on page 4-40, and Figure 4.6 on preceding page.

Note: It is necessary to include a WRITE statement to obtain a value with COMGET.

4.4.4 General COMMON Section Pointer - IBASEC

Function IBASEC is used to find the location of any item in COMMON pertaining to the type of line and data point symbols to be used.

Calling Sequence:

```
IBASEC (IWORD)
```

Parameters Entered:

IWORD	is the integer used to designate the item in this section of COMMON. The first column of Table II on page 4-30. provides the item designation number. For example, IWORD equal to 2 designates variable CSTEPS.

4.4.5 X Axis COMMON Section Pointer - IBASEX

IBASEX is a function used to find the location of a particular item in COMMON which pertains to the X axis.

Calling Sequence:

```
IBASEX (IWORD)
```

Parameter Entered:

IWORD	is the integer used to designate the item in the X axis section of COMMON. The first column of Table II provides you with the appropriate IWORD value. For example, IWORD equal to 3 designates variable CXLAB.

See Examples on page 4-40.

4.4.6 Y Axis COMMON Section Pointer - IBASEY

IBASEY is used in the same way as IBASEX, but designates an item in COMMON which pertains to the Y axis.

Calling Sequence:

```
IBASEY (IWORD)
```

Parameter Entered:

IWORD	is the integer used to designate the item in the Y axis section of COMMON. The first column of Table II provides the appropriate IWORD value. For example, IWORD equal to 3 designates variables CYLAB.

See Example 2 on page 4-40.

4.4.7 Examples of Use of COMSET, COMGET, IBASEC, IBASEX, and IBASEY

1. To find the value that has been assigned in COMMON to the X axis neat flag (CXNEAT), it is necessary to use the two functions COMGET and IBASEX.

   ```
   ITEM = IBASEX(Ø)
   VALUE = COMGET(ITEM)

         or

   VALUE=COMGET(IBASEX(Ø))
   ```

   Ø is the integer assigned to CXNEAT in Table II.

2. To find the value of the Y axis data minimum (CYDMIN) in COMMON, functions COMGET and IBASEY are used.

   ```
   ITEM=IBASEY(11)
   VALUE=COMGET(ITEM)

         or

   VALUE=COMGET(IBASEY(11))
   ```

   11 is the number assigned to CYDMIN in Table II.

3. To set the value of the X axis neat flag to Ø (off), COMSET and IBASEX could be used in the following manner.

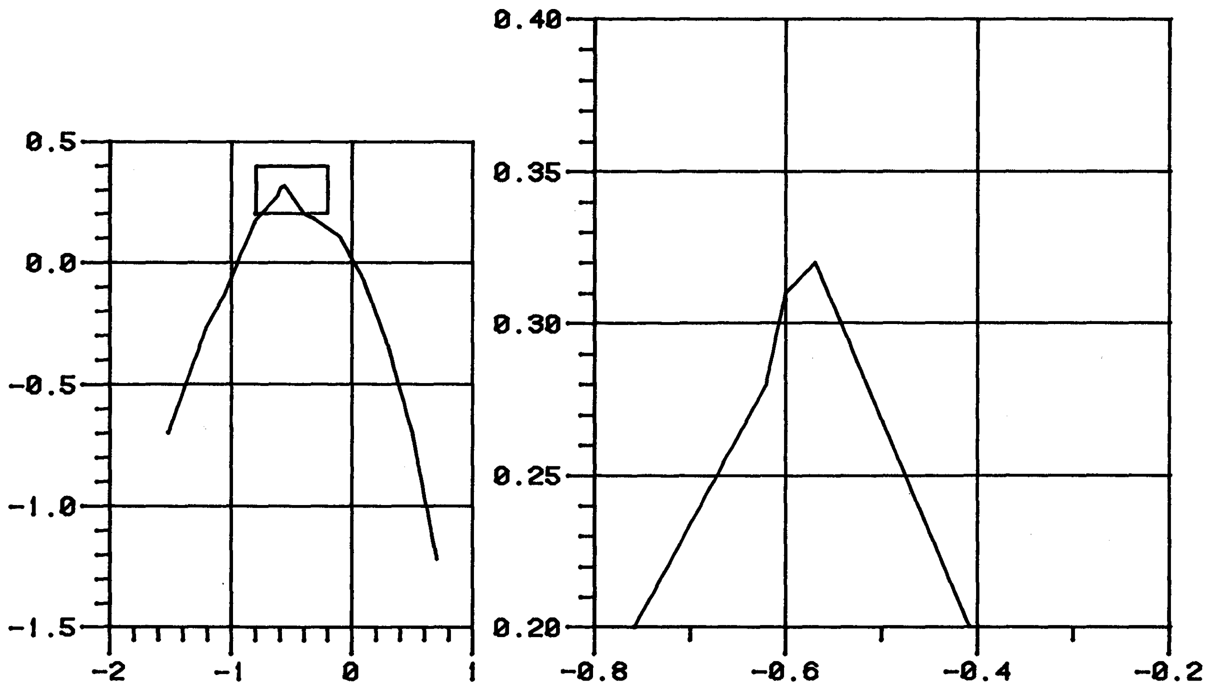
   ```
         CALL COMSET(IBASEX(Ø),Ø.)
   ```

   where IBASEX(Ø) provides the location in COMMON of the X axis neat tic flag (CXNEAT) and the second Ø is the value to be assigned to the neat flag.

APPENDIX A

Sample Graphs

A group of complex graphs were collected at the end of Section 1 to demonstrate the variety available with the package. The code for these examples is contained in the following section.



```
C******************************************************************
      DIMENSION XPTS(15),YPTS(15)
      DATA XPTS/14.,-1.52,-1.21,-1.05,-.81,-.62,-.60,-.57,-.41,-.32,
     &-.09,.1,.3,.5,.7/
      DATA YPTS/14.,-.70,-.27,-.12,.17,.28,.31,.32,.20,.18,.10,-.08,
     &-.35,-.71,-1.22/
      CALL INITT(120)
C * THE MAIN CURVE
      CALL BINITT
      CALL SLIMX(100,400)
      CALL SLIMY(200,600)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
C * DRAW A BOX AROUND THE PORTION OF THE GRAPH TO BE ENLARGED
      CALL MOVEA(-.8,.2)
      CALL DRAWA(-.2,.2)
      CALL DRAWA(-.2,.4)
      CALL DRAWA(-.8,.4)
      CALL DRAWA(-.8,.2)
      CALL ANMODE
C * THE ENLARGEMENT, PRODUCED BY RESETING DATA LIMITS
      CALL BINITT
      CALL DLIMX(-.8,-.2)
      CALL DLIMY(.2,.4)
      CALL SLIMX(500,975)
      CALL SLIMY(200,700)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
      CALL FINITT(0,700)
      END
C******************************************************************
```

Figure A.1

This display demonstrates the enlargement of a detail in the second graph on the same page.

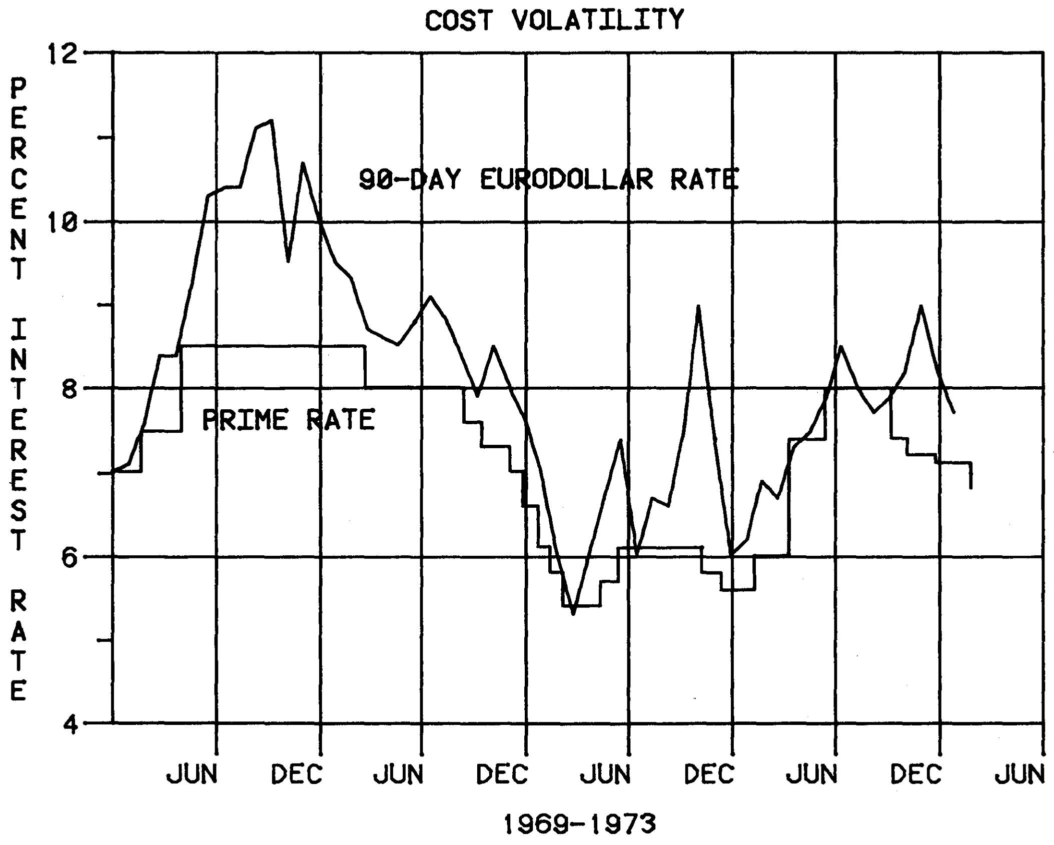


Figure A.2

The Cost Volatility graph demonstrates the use of a multiple curve with a user written subroutine ULINE to create a stepped data line.

```
C*************************************************************
      REAL XPTS(55),YPTS(55),UPTS(23),VPTS(23)
      INTEGER IST(22),ISTR(10),ISTRI(15),ISTRIN(21)
      LEN=22
      LENG=10
      LENGT=15
      LENGTH=21
C * CURVE LABEL:9  0   -   D  A   Y    E   U   R   O   D   O   L   L
C *    A   R     R  A   T   E
      DATA IST/57,48,45,100,97,121,32,69,117,114,111,100,111,108,108,
     &97,114,32,114,97,116,101/
C * CURVE LABEL:  PRIME RATE
      DATA ISTR/80,114,105,109,101,32,114,97,116,101/
C * GRAPH TITLE:  COST VOLATILITY
      DATA ISTRI/67,79,83,84,32,86,79,76,65,84,73,76,73,84,89/
C * SCALE LABEL:  PERCENT INTEREST RATE
      DATA ISTRIN/80,101,114,99,101,110,116,32,73,110,116,101,114,101
     &,115,116,32,114,97,116,101/
```

```
C * DEFINE LONG FORM DATA ARRAY IN UBGC DATES
      XPTS(1)=54.
      CALL IUBGC(1969,1,IXPTS)
      XPTS(2)=FLOAT(IXPTS)
      DO 100 I=3,55
100   XPTS(I)=XPTS(I-1)+28.
C * DATA ARRAY IN PERCENT
      DATA YPTS/54.,7.0,7.1,7.6,8.4,8.4,9.3,10.3,10.4,10.4,11.1,11.2,
     &9.5,10.7,10.0,9.5,9.3,8.7,8.6,8.5,8.8,9.1,8.8,8.4,7.9,8.5,
     &8.0,7.6,7.0,6.0,5.3,6.0,6.7,7.4,6.0,6.7,6.6,7.5,9.0,7.4,6.0,6.2,
     &6.9,6.7,7.3,7.5,7.9,8.5,8.0,7.7,7.9,8.2,9.0,8.2,7.7/
C * DATA ARRAY IN DAY INTERVALS
      DATA UPTS/22.,0.,50.,70.,325.,175.,30.,50.,25.,25.,20.,25.,65.
     &,30.,150.,35.,60.,60.,64.,115.,30.,50.,60./
C * DATA ARRAY IN PERCENT
      DATA VPTS/22.,7.0,7.5,8.5,8.0,7.8,7.3,7.0,6.6,6.1,5.8,5.4,5.7,
     &6.1,5.8,5.6,6.0,7.4,8.0,7.4,7.2,7.1,6.8/
C * CHANGE UPTS ARRAY TO UBGC DATE VALUES
      UPTS(2)=XPTS(2)
      DO 200 I=3,23
200   UPTS(I)=UPTS(I-1)+UPTS(I)
      CALL INITT(120)
      CALL BINITT
      CALL XWDTH(3)
      CALL SLIMY(200,690)
      CALL XTYPE(3)
      CALL XDEN(9)
      CALL XLAB(6)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
C * ON THE SECOND PLOT USER WRITTEN SUBROUTINE ULINE IS USED
      CALL LINE(-11)
      CALL CPLOT(UPTS,VPTS)
C * LABEL THE GRAPH INTERACTIVELY
      CALL DCURSR(IC,IX,IY)
      CALL NOTATE(IX,IY,LEN,IST)
      CALL DCURSR(IC,IX,IY)
      CALL NOTATE(IX,IY,LENG,ISTR)
      CALL DCURSR(IC,IX,IY)
      CALL NOTATE(IX,IY,LENGT,ISTRI)
      CALL DCURSR(IC,IX,IY)
      CALL MOVABS(IX,IY)
      CALL VLABEL(LENGTH,ISTRIN)
      CALL TINPUT(K)
      CALL FINITT(0,700)
      END
C * THIS SUBROUTINE IS CALLED TO GRAPH THE SECOND CURVE
C * SINCE CLINE WAS SET TO -1
      SUBROUTINE ULINE(X,Y,I)
      IF(I .EQ. 1) GO TO 100
      CALL DRAWA(X,YOLD)
      CALL DRAWA(X,Y)
100   YOLD=Y
      RETURN
      END
C***********************************************************
```

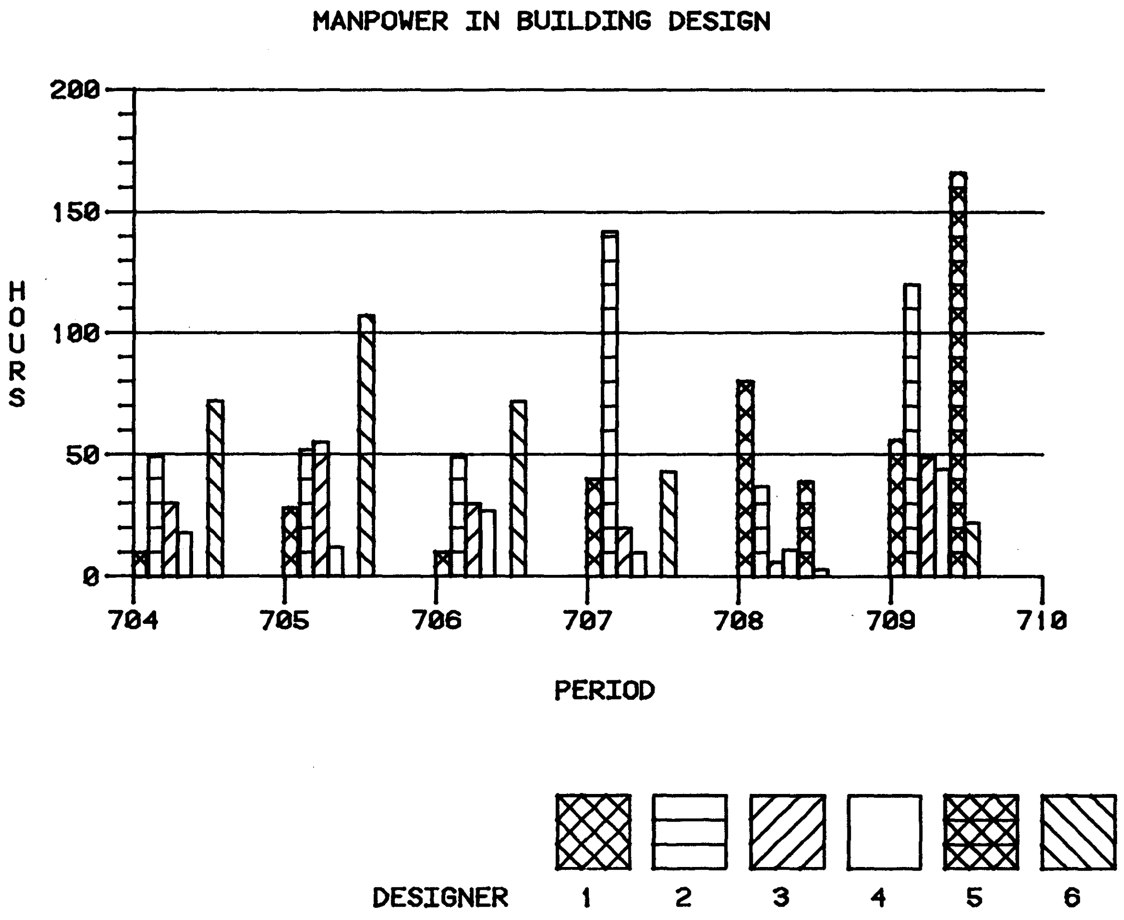


Figure A.3

This graph demonstrates the use of a bar chart with user written subroutines ULINE and LEDG. Subroutine FILBOX draws the shading lines for the legend.

```
C***********************************************************************
C * ROUTINE TO PLOT TIME INVESTED IN BUILDING DESIGN
C *  SINCE PERIOD 704
      DIMENSION PERIOD(7),DESINR(7,6)
      DIMENSION ITITLE(28),IH(6),IP(7),IDESIN(9)
C * ASCII EQUIVILENT OF 'MANPOWER IN BUILDING DESIGN'
      DATA ITITLE/27,77,65,78,80,79,87,69,82,32,73,78,32,66,85,73,
     1  76,68,73,78,71,32,68,69,83,73,71,78/
C * ASCII EQUIVILENT OF 'HOURS'
      DATA IH/6,72,79,85,82,83/
C * ASCII EQUIVILENT OF 'PERIOD'
      DATA IP/6,80,69,82,73,79,68/
C * ASCII EQUIVILENT OF ' DESIGNER '
      DATA IDESIN/8,68,69,83,73,71,78,69,82/
      DATA PERIOD/6.,704.,705.,706.,707.,708.,709./
      DATA DESINR/6.,10.,28.,10., 40.,80., 56.,
     1            6.,49.,52.,49.,142.,37.,120.,
     2            6.,30.,55.,30., 20., 6., 49.,
     5            6.,18.,12.,27.,10.,11.,44.,
     6            6., 4*1.E30,         39.,166.,
     7            6.,72.,107.,72.,43., 3., 22./
      CALL INITT(120)
      CALL BINITT
C * SET THE LINE TYPE IN COMMON
      CALL LINE (-11.)
C * SET THE COMMON FOR THE PROPER BAR TYPE
      CALL SIZEL(12.)
      CALL XFRM(2)
      CALL DLIMX(704.,710.)
      CALL DLIMY(0.,200.)
      CALL SLIMY(300,700)
      CALL XDEN(3)
      CALL CHECK(PERIOD,DESINR)
      CALL DSPLAY(PERIOD,DESINR)
C * PLOT THE REMAINING BARS
      DO 100 I=2,6
100   CALL CPLOT(PERIOD,DESINR(1,I))
C * PLACE TITLES ON THE GRAPH
      CALL NOTATE(300,750,ITITLE(1),ITITLE(2))
      CALL NOTATE(350,30,IDESIN(1),IDESIN(2))
      CALL NOTATE(500,200,IP(1),IP(2))
      CALL MOVABS(50,550)
      CALL VLABEL(IH(1),IH(2))
      CALL TINPUT(K)
      CALL FINITT(0,30)
      END
C * SUBROUTINE TO DECIDE THE BAR AND SHADING INSIDE
      SUBROUTINE ULINE(X,Y,I)
      DATA XOFF,DEL,J/-.06,.1,0/
      IF(I .NE. 1)GO TO 100
      XOFF=XOFF+DEL
      K=2
      J=J+1
```

```
C * LEGEND AT THE BOTTOM OF THE GRAPH
      CALL LEDG(500,30,J)
      IF(J .LE. 6)GO TO 100
      J=1
      XOFF=-.6+DEL
100   X2=X+XOFF
      IF(Y .GT. 1.E30)RETURN
      CALL SYMBL(IT(J))
      CALL BAR(X2,Y,K)
      CALL SYMBL(0)
      K=0
      RETURN
      END
```

```
C * FUNCTION WHICH HAS THE CODES FOR THE SHADING OF THE BARS
      FUNCTION IT(I)
      INTEGER ITT(6)
C * CODES FOR BAR SHADINGS
      DATA ITT/12,1,4,0,13,8/
      IT=ITT(I)
      RETURN
      END
C * ROUTINE TO PLACE LEGEND ON THE SCREEN
      SUBROUTINE LEDG(IX,IY,I)
      DATA J/0/
      J=J+1
      IF(J .GT. 1)GO TO 100
      ILX=IX-80
      ILY=IY+30
100   ILX=ILX+80
      CALL FILBOX(ILX,ILY,ILX+60,ILY+60,IT(J),20)
      CALL MOVABS(ILX+20,ILY-30)
      CALL ANCHO (J+48)
      RETURN
```

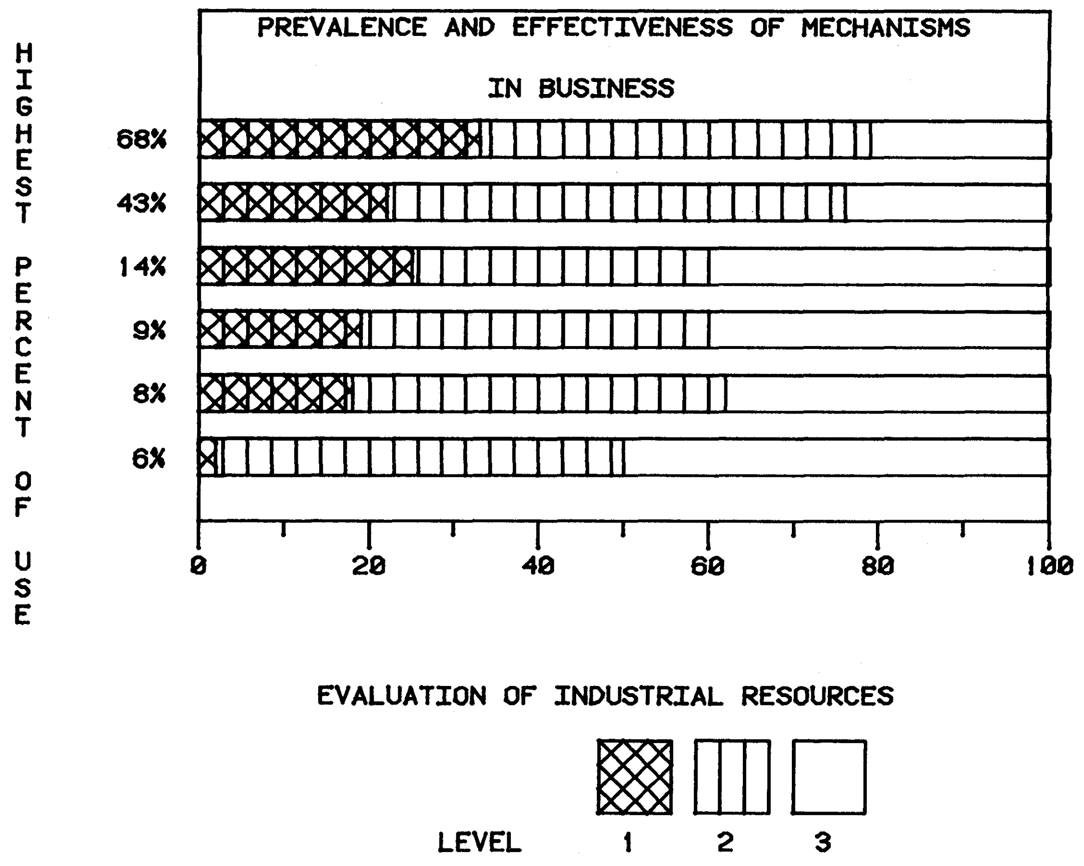


Figure A.3.2

```
      DIMENSION BAR1(7),BAR2(7,3)
      DIMENSION ILABEL(22),JLABEL(34),KLABEL(42),LABEL(11),LABEL1(5)
      DATA BAR1/6.,1.,2.,3.,4.,5.,6./
      DATA BAR2/6.,2.,18.,19.,25.,22.,33.,
     1          6.,50.,62.,60.,60.,76.,79.,
     2          6.,100.,100.,100.,100.,100.,100./
C * ASCII EQUIVILENT OF 'HIGHEST PERCENT OF USE'
      DATA ILABEL/72,73,71,72,69,83,84,32,80,69,82,67,69,78,84,
     & 32,79,70,32,85,83,69/
C * ASCII EQUIVILENT OF 'EFFECTIVENESS OF INDUSTRIAL RESOURCES'
      DATA JLABEL/69,86,65,76,85,65,84,73,79,78,32,79,70,32,
     &73,78,68,85,83,84,82,73,65,76,32,82,69,83,79,85,82,67,69,83/
C * ASCII EQUIVELENCE OF 'PREVELENCE OF EFFECTIVENESS OF MECHANISM'
      DATA KLABEL/80,82,69,86,65,76,69,78,67,69,32,65,78,68,32,
     &69,70,70,69,67,84,73,86,69,78,69,83,83,32,79,70,32,77,
     &69,67,72,65,78,73,83,77,83/
C * ASCII EQUIVELENCE OF 'IN BUSINESS'
      DATA LABEL/73,78,32,66,85,83,73,78,69,83,83/
```

```
C * ASCII EQUIVELENCE OF 'LEVEL'
      DATA LABEL1/76,69,86,69,76/
      CALL INITT(120)
      CALL BINITT
C * SET SCREEN LIMITS TO INCLUDE TITLES
      CALL SLIMX(200,900)
      CALL SLIMY(300,720)
C * SET LABELS FOR Y TO NEGATIVE VALUE
      CALL COMSET(IBASEY(3),-1.)
C * FORCE TICS FOR Y TO SPECIFICATION WANTED
      CALL YTICS(8)
      CALL YWDTH(20)
      CALL YFRM(1)
      CALL YMFRM(1)
      CALL XFRM(2)
      CALL FRAME
C * SET DATA LIMITS
      CALL DLIMX(0.,100.)
      CALL DLIMY(0.,8.)
C * DESIGN THE BARS AND DRAW THEM
      CALL HBARST(IT(1),30,20)
      CALL CHECK(BAR2,BAR1)
      CALL DSPLAY(BAR2,BAR1)
C * DRAW THE REST OF THE BARS
      DO 100 J=2,3
      CALL HBARST(IT(J),30,20)
C * PUT OUT THE LEGEND
      CALL LEDG(530,30,J)
100   CALL CPLOT(BAR2(1,J),BAR1)
      CALL LEDG(530,30,3)
C * PUT OUT VERTICAL LABEL
      CALL MOVABS(50,700)
      CALL VLABEL(22,ILABEL)
C * PUT OUT GRAPH TITLES
      CALL NOTATE (400,30,5,LABEL1)
      CALL NOTATE(300,150,34,JLABEL)
      CALL NOTATE(250,700,42,KLABEL)
      CALL NOTATE(440,650,11,LABEL)
      CALL TINPUT(I)
      END
C * FUNCTION TO DESIGNATE BAR SHADING
      FUNCTION IT(I)
      INTEGER ITT(3)
C * BAR SHADING CODE
      DATA ITT/12,2,0/
      IT=ITT(I)
      RETURN
      END
C * ROUTINE TO CALCULATE LEGEND
```

```
      SUBROUTINE LEDG(IX,IY,I)
      DATA J/0/
      J=J+1
      IF(J .GT. 1)GO TO 100
      ILX=IX-80
      ILY=IY+30
100   ILX=ILX+80
      CALL FILBOX(ILX,ILY,ILX+60,ILY+60,IT(J),20)
      CALL MOVABS(ILX+20,ILY-30)
      CALL ANMODE
      CALL TOUTPT(J+48)
      RETURN
      END
C * ROUTINE TO OUTPUT THE USER LABELS ON THE Y AXIS
      SUBROUTINE USESET(FNUM,IWIDTH,NBASE,LABELI)
      DIMENSION LABELI(2)
      DIMENSION IASCIL(27)
C * ASCII EQUIVELENCE OF ALL THE LABELS
      DATA IASCIL/32,32,32,32,54,37,32,56,37,32,57,37,
     & 49,52,37,52,51,37,54,56,37,32,32,32,32,32,32/
      IW=IWIDTH-3
      DO 100 I=1,IW
100   LABELI(I)=32
      IC=(IFIX(FNUM)*3)+1
      IW=IW+1
      DO 200 I=IW,IWIDTH
      LABELI(I)=IASCIL(IC)
200   IC=IC+1
      RETURN
      END
C*****************************************************************
```

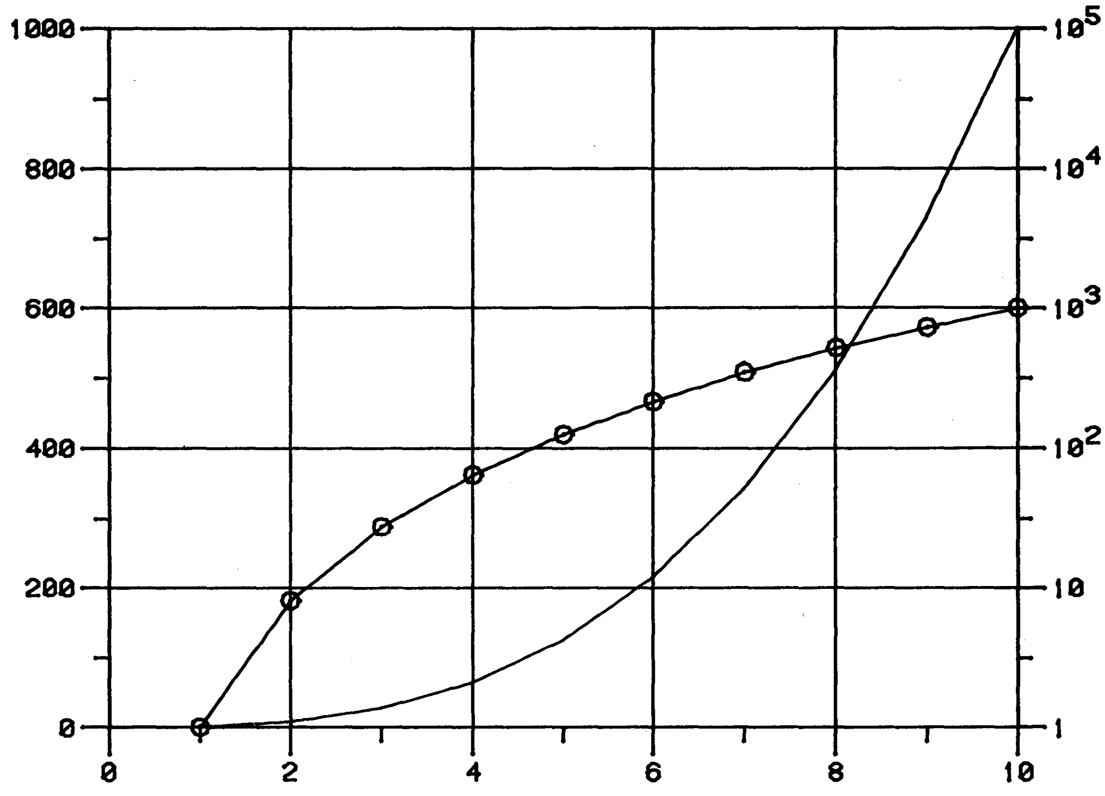


Figure A.4

This graph demonstrates the same data plotted against a linear and a logarithmic Y axis varying slightly from Figure A.

```
C*************************************************************
C * DEMO PROGRAM USING LOG AND LINEAR PLOTTING
      REAL XPTS(11),YPTS(11)
      XPTS(1)=10
      YPTS(1)=10
C * GENERATE DATA POINTS FOR Y=X**3
      DO 100 I=2,11
      XPTS(I)=FLOAT(I-1)
100   YPTS(I)=FLOAT(I-1)**3
      CALL INITT(120)
      CALL BINITT
C * SET THE HORIZONTAL SCREEN LIMITS
      CALL SLIMX(300,900)
C * USE CHECK TO COMPLETE THE SPECIFICATIONS
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
C * LOCATE Y AXIS TO LEFT OF FIRST AXIS
      CALL YLOC(-90)
C * PLOT LOG WITH CIRCLED DATA POINTS
      CALL SYMBL(5)
C * MAKE Y AXIS A LOGARITHMIC TRANSFORMATION
      CALL YTYPE(2)
C * SUPPRESS THE LABELING OF THE X AXIS A 2ND TIME
      CALL XLAB(0)
C * SUPPRESS THE DRAWING OF THE X AXIS A 2ND TIME
      CALL XFRM(0)
C * CHANGE THE FORM OF THE MAJOR TIC MARKS
      CALL YFRM(2)
C * RESET THE  Y DATA LIMITS
      CALL DLIMY(0.,0.)
C * USE CHECK TO COMPLETE COMMON
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
      CALL TINPUT(IVY)
C * DUMP COMMON TABLE IF C IS TYPED
      IF(IVY.NE.67)GO TO 200
      CALL COMDMP
200   STOP
      END
C*************************************************************
```

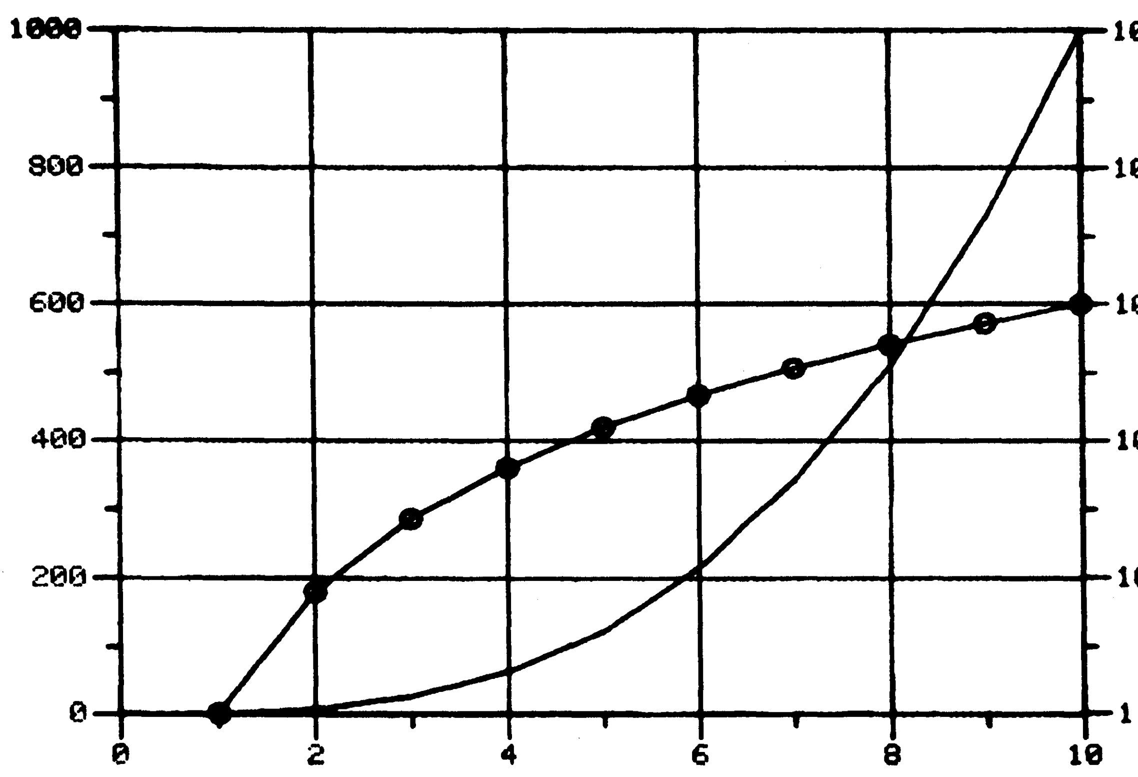


Figure A.5

This graph allows comparison of the same data plotted as logarithmic and as linear.

```
C*****************************************************************
C * DEMO PROGRAM USING LOG AND LINEAR PLOTTING
C * THE LOGARITHMIC AXIS IS TO THE RIGHT OF THE GRAPH
        REAL XPTS(11),YPTS(11)
        XPTS(1)=10
        YPTS(1)=10
        DO 100 I=2,11
        XPTS(I)=FLOAT(I-1)
100     YPTS(I)=FLOAT(I-1)**3
        CALL INITT(120)
        CALL BINITT
        CALL CHECK(XPTS,YPTS)
        CALL DSPLAY(XPTS,YPTS)
        LIMZ=0
        CALL SYMBL(1)
        CALL YLAB(2)
        CALL YTYPE(2)
C * LOCATE THE Y-AXIS ON THE RIGHT OF THE GRAPH
        CALL YLOCRT(LIMZ)
        CALL XLAB(0)
        CALL XFRM(0)
        CALL YFRM(2)
        CALL DLIMY(0,0)
        CALL CHECK(XPTS,YPTS)
        CALL DSPLAY(XPTS,YPTS)
        CALL TINPUT(IVY)
        STOP
        END
C*****************************************************************
```

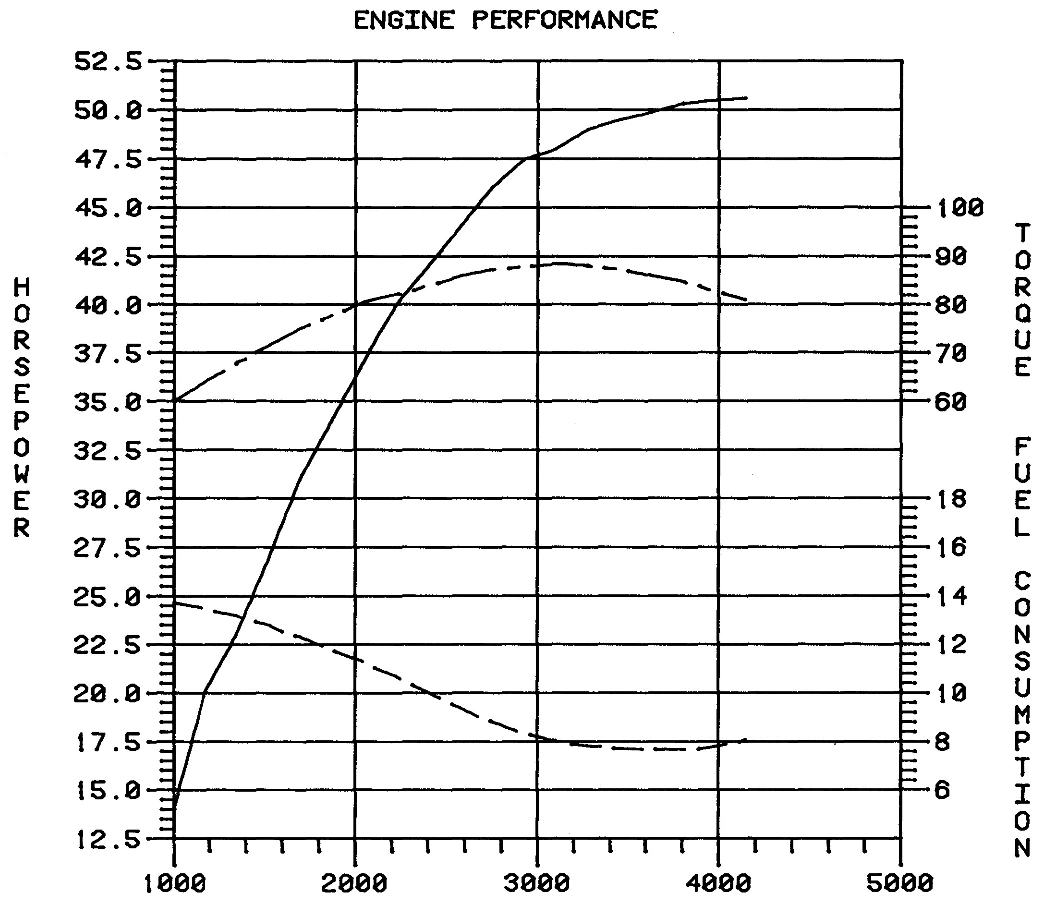
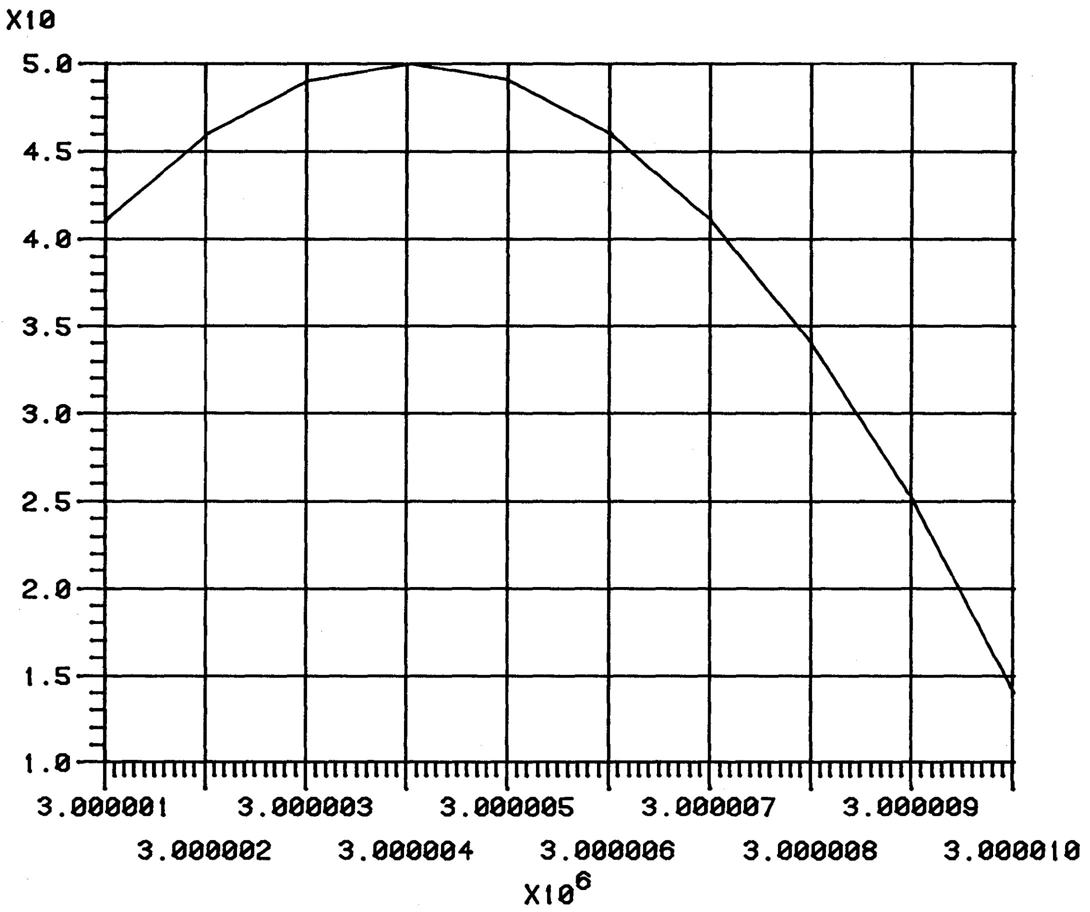


Figure A.6

The engine performance graph uses DINITY to reinitialize tic mark label values.

```
C*****************************************************************
      DIMENSION RPM(4),POW(20),SPC(20),TOR(20),LAB1(18),LAB2(10)
     &,LAB3(16),LAB4(6)
C * LABEL LENGTHS
      L1=18
      L= 10
      L3=16
      L4=6
C * GRAPH TITLE: E  N  G  I  N  E     P  E  R  F  O  R  M  A  N  C  E
      DATA LAB1/69,78,71,73,78,69,32,80,69,82,70,79,82,77,65,78,67,69/
C * SCALE LABEL: H  O  R  S  E  P  O  W  E  R
      DATA LAB2/72,79,82,83,69,80,79,87,69,82/
C * SCALE LABEL: F  U  E  L     C  O  N  S  U  M  P  T  I  O  N
      DATA LAB3/70,85,69,76,32,67,79,78,83,85,77,80,84,73,79,78/
```

```
C * SCALE LABEL: T  O  R  Q  U  E
      DATA LAB4/84,79,82,81,85,69/
      DATA RPM/-1.,19.,1000.,175./
      DATA POW/19.,14.,20.,23.,27.,31.,34.,37.,40.,42.,44.,46.,47.5,
     &48.,49.,49.5,49.9,50.3,50.5,50.6/
      DATA SPC/19.,13.7,13.5,13.17,12.77,12.30,11.8,11.25,10.65,10.0,
     &9.35,8.8,8.35,8.0,7.8,7.7,7.65,7.66,7.8,8.1/
      DATA TOR/19.,60.,64.,68.,71.5,74.9,77.9,80.4,82.3,
     &84.,86.,87.2,88.,88.6,87.9,87.2,86.,84.8,83.,81./
      CALL INITT(120)
      CALL BINITT
      CALL SLIMX(200,800)
C * EACH Y-AXIS TIC INTERVAL WILL SPAN 40 RASTER UNITS
      CALL SLIMY(100,100+16*40)
      CALL YTICS(16)
      CALL CHECK(RPM,POW)
      CALL DSPLAY(RPM,POW)
C * Y-DATA DIFFERS IN THE SECOND CURVE, SO REINITIALIZATION IS NEEDED
      CALL DINITY
C * SECOND CURVE SPANS 6 TIC INTERVALS
      CALL SLIMY(140,140+6*40)
      CALL YTICS(6)
      CALL LINE(54)
      CALL YLOCRT(0)
      CALL XFRM(0)
      CALL XLAB(0)
      CALL YFRM(2)
      CALL CHECK(RPM,SPC)
      CALL DSPLAY(RPM,SPC)
C * THIRD CURVE IS HANDLED SIMILARLY TO SECOND CURVE
      CALL DINITY
      CALL LINE(7434)
      CALL YTICS(4)
      CALL SLIMY(460,460+4*40)
      CALL CHECK(RPM,TOR)
      CALL DSPLAY(RPM,TOR)
C * PRINT LABELS INTERACTIVELY
      CALL DCURSR(IC,IX,IY)
      CALL MOVABS(IX,IY)
      CALL HLABEL(L1,LAB1)
      CALL DCURSR(IC,IX,IY)
      CALL MOVABS(IX,IY)
      CALL VLABEL(L2,LAB2)
      CALL DCURSR(IC,IX,IY)
      CALL MOVABS(IX,IY)
      CALL VLABEL(L3,LAB3)
      CALL DCURSR(IC,IX,IY)
      CALL MOVABS(IX,IY)
      CALL VLABEL(L4,LAB4)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
C**************************************************************
```



```
C * PROGRAM TO DEMOMSTRATE THE REMOTE EXPONENT AND LABEL STAGGERING
      REAL XPTS(11),YPTS(11)
      XPTS(1)=10
      YPTS(1)=10
      BASE=3000000
      DO 100 I=2,11
      XPTS(I)=BASE+1*FLOAT(I-1)
100   YPTS(I)=10000000*(50-FLOAT(I-5)**2)
      CALL INITT(30)
      CALL BINITT
      CALL XDEN(10)
      CALL YDEN(10)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
      CALL TINPUT(IVY)
      CALL FINITT(0,700)
      END
```

Figure A.7

This graph demonstrates use of a remote exponent and staggered labeling.

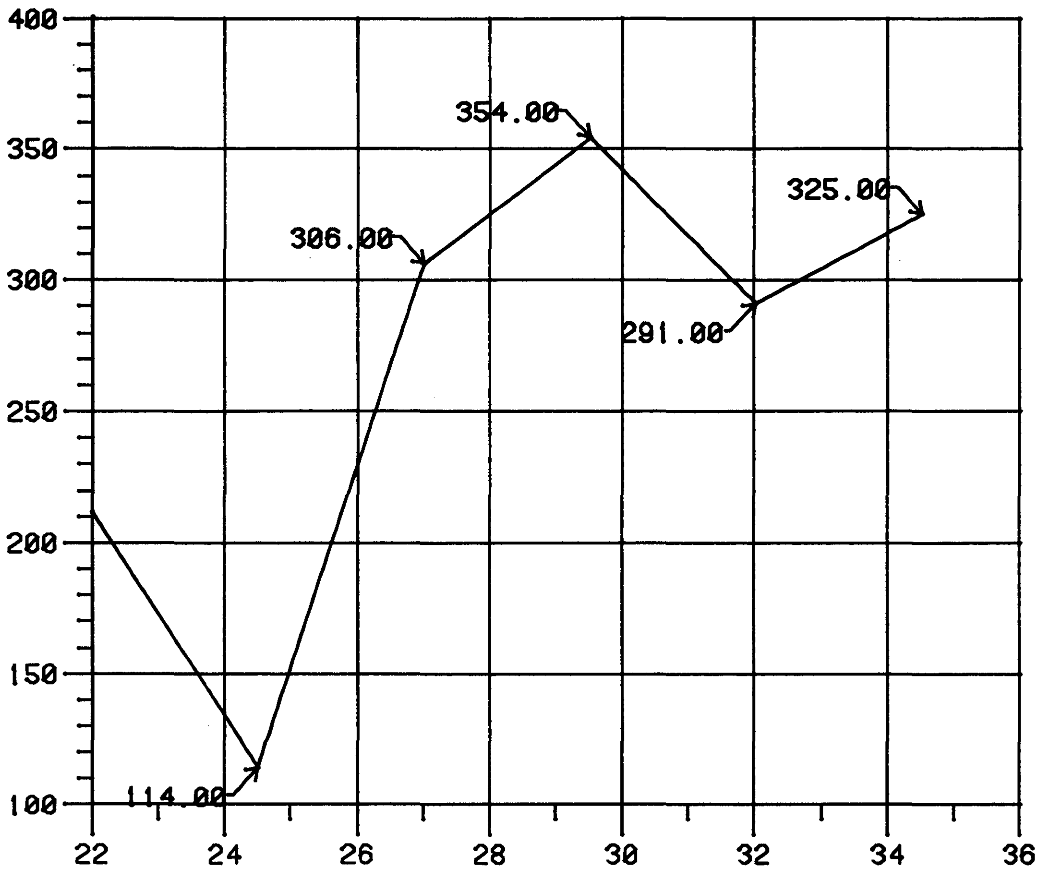


Figure A.8

A user written symbol routine.

```
C***************************************************************
      DIMENSION XARRAY(7),YARRAY(7)
      DATA XARRAY/6.,22.0,24.5,27.0,29.5,32.0,34.5/
      DATA YARRAY/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(120)
      CALL BINITT
      CALL SYMBL(-1)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
C * ROUTINE TO LABEL POINTS WITH Y VALUE
      SUBROUTINE USERS(X,Y,ISYM)
      INTEGER IA(15)
      LOGICAL GENFLG
      DATA IYOLD,ISIZ/98987,20/
C * CONVERT TO SCREEN COORDINATES
      CALL WINCOT(X,Y,IX,IY)
C * THIS CODE IS A FIRST ENTRY BRANCH
      IF(IYOLD .EQ. 98987)GO TO 100
C * STORE SLOPE OF LINE -DOWN OR UP- FOR LABEL LOC
      ISIG=IY-IYOLD
C * BRANCH IF POINT IS OUTSIDE OF WINDOW
      IF(GENFLG(1))GO TO 100
C * GET PARAMETERS FROM AXIS COMMON TO DEFINE LENGTH OF LABEL
      NBASE=IBASEY(0)
      NDEC=COMGET(NBASE+10)+2
      IWIDTH=COMGET(NBASE+17)+3
      IEXP=COMGET(NBASE+18)
C * CONVERT Y VALUE TO ALPHA STRING
      CALL FFORM(Y/10.**IEXP,IWIDTH,NDEC,IA,32)
C * RIGHT JUSTIFY LABEL
      CALL JUSTER(IWIDTH,IA,1,32,LEN,IOFF)
C * DRAW SMALL ARROW ABOVE OR BELOW LINE
      CALL MOVABS(IX-10,IY+ISIGN(2,ISIG))
      CALL DRWABS(IX,IY)
      CALL DRWABS(IX-2,IY+ISIGN(10,ISIG))
      CALL MOVABS(IX,IY)
      CALL DRWABS(IX-ISIZ,IY+ISIGN(ISIZ,ISIG))
      CALL DRWABS(IX-ISIZ-5,IY+ISIGN(ISIZ,ISIG))
C * WRITE OUT LABEL
      IYPOS=IY+ISIGN(ISIZ,ISIG)-8
      CALL NOTATE(IX-ISIZ-7+IOFF,IYPOS,LEN,IA(IWIDTH-LEN+1))
C * PREPARE FOR NEXT SLOPE CALCULATION
100   IYOLD=IY
      RETURN
      END
C***************************************************************
```

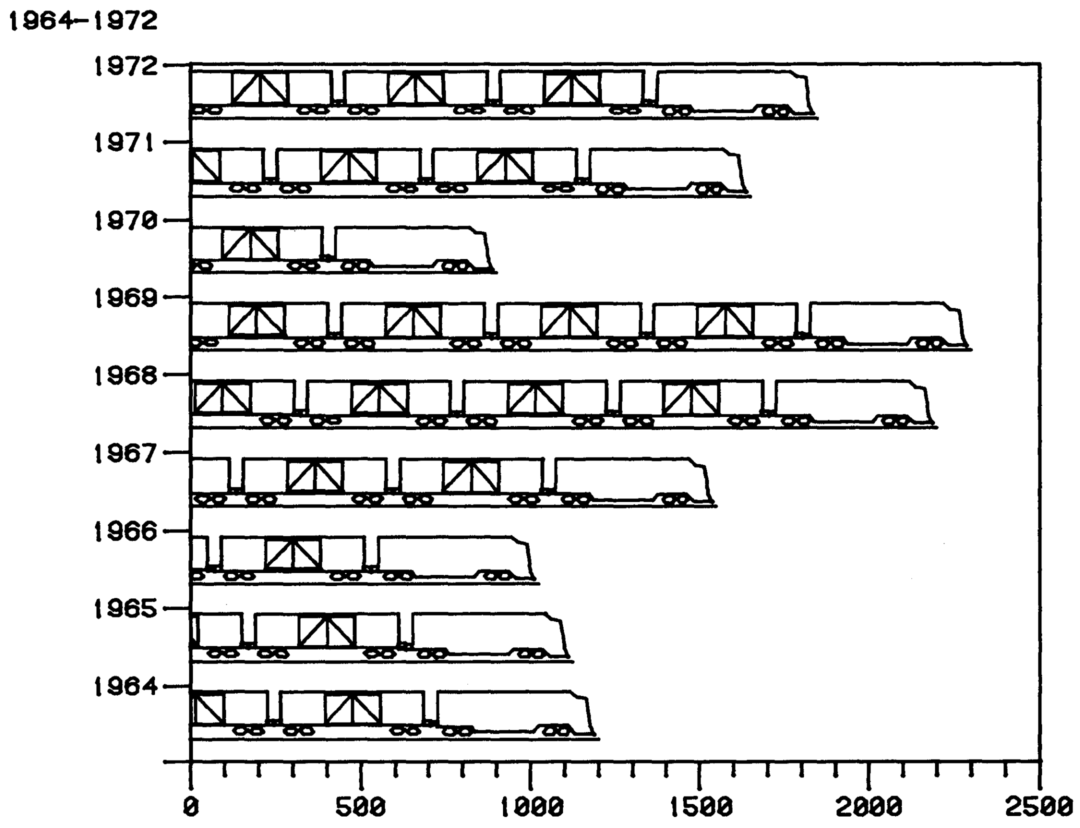


Figure A.9

User written line routine produces train bars

```
C*****************************************************************
C DRIVER PROGRAM FOR TRAIN GRAPH
      REAL YPTS(5),XPTS(11)
      DATA YPTS/-2.,10.,1.,1963.,1./
      DATA XPTS/10.,750.,1200.,1125.,1025.,1550.,2200.,2300.,900.,
     & 1650.,1850./
      CALL INITT(120)
      CALL BINITT
      CALL XZERO(0)
      CALL YDEN(10)
      CALL SLIMX(200,900)
      CALL YFRM(2)
      CALL XFRM(2)
      CALL LINE(-11)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
      CALL FRAME
      CALL TINPUT(JAKE)
      CALL FINITT(0,0)
      END
      SUBROUTINE ULINE(X,Y,I)
      CALL MOVEA(0.,Y-255.)
      CALL DRAWA(X,Y-255.)
      IF(I .NE. 1)GO TO 100
      TICINT=COMGET(IBASEY(5))
      AMIN=COMGET(IBASEY(11))
      AMAX=COMGET(IBASEY(12))
      SCALE=(AMAX-AMIN)/TICINT
      CALL RSCALE(0.8*SCALE)
100   NUM=X/500.+1.
      CALL TRAIN(NUM)
      RETURN
      END
C*****************************************************************

C ROUTINE TO DRAW TRAIN
      SUBROUTINE TRAIN(NUM)
      REAL ARRAY1(3,48),ARRAY2(3,49)
      DATA ARRAY2/32.,-.0700,0.,32.,.0021,.0806,32.,.0679,.0,32.,0.,
     & -.0785,32.,-1.5690,0.,32.,-.0021,
     & .0806,32.,.0658,0.,32.,0.,-.0785,32.,-.0021,0.5541,32.,1.4416,
     & -.0021,32.,0.,-.5520,77.,-.0403,-.0955,32.,-.0339,-.0467,32.,
     & -.0636,.0,32.,-.0276,.0530,32.,.0276,.0594,32.,.0679,0.,32.,
     & .027,-.0594,77.,-.1528,.0063,32.,-.0403,-.0573,32.,-.0700,0.,32.,
     & -.0297,.0594,32.,.0297,.0530,32.,.0721,0.,32.,.0318,-.0531,77.,
     & -1.1974,-.0064,32.,.0254,-.0509,32.,.0743,0.,32.,.0297,.0552,32.,
     & -.0360,.0552,32.,-.0679,0.,32.,-.0233,-.0530,77.,.1528,0.,32.,
     & .0361,-.0552,32.,.0785,0.,32.,.0233,.0509,32.,-.0318,.0594,32.,
     & -.0679,0.,32.,-.0361,-.0531,77.,.8110,.0998,32.,-.0021,.4968,
     & 32.,-.5690,.0021,32.,0.,-.4989,32.,.5690,0.,32.,-.2824,.4947,32.,
     & 0.,-.4925,77.,0.,.4968,32.,-.2866,-.4968,77.,-.5074,-.0254/
      DATA ARRAY1/77.,-.0517,.0742,32.,-.0292,.1281,32.,-.0449,.4472,
     & 32.,-.0921,.0269,32.,-.0629,.0831,32.,-1.3460,0.,32.,0.,-.4921,
     & 32.,-.0651,-.0022,32.,-.0022,-.0674,32.,.0651,0.,32.,-.0022,
     & .0651,32.,0.,-.0674,32.,.3438,0.,32.,.0449,-.0898,32.,.5932,0.,
     & 32.,.0517,.0921,32.,.2943,0.,32.,.0719,-.1213,32.,.1798,0.,77.,
     & -.2382,.0449,32.,-.0224,-.0696,32.,-.0719,0.,32.,-.0202,.0651,
     & 32.,.0224,.0606,32.,.0719,0.,32.,.0202,-.0607,77.,-.1416,0.,32.,
     & -.0247,-.0651,32.,-.0674,.0023,32.,-.0247,.0607,32.,.0292,.0652,
     & 32.,.0674,0.,32.,.0179,-.0607,77.,-.8517,-.0023,32.,-.0270,
     & -.0629,32.,-.0674,0.,32.,-.0269,.0607,32.,.0269,.0629,32.,.0651,
     & 0.,32.,.0292,-.0607, 77.,-.1483,0.,32.,-.0292,-.0629,32.,-.0764,
     & 0.,32.,-.0224,.0607, 32.,.0337,.0651,32.,.0629,0.,32.,.0269,
     & -.0629,77., -.2607,.0831/
C   PUT OUT THE LOCOMOTIVE
100   DO 400 J=1,48
      IF(ARRAY1(1,J).EQ.77.)GO TO 350
      CALL DRAWR(ARRAY1(2,J),ARRAY1(3,J))
      GO TO 400
350   CALL MOVER(ARRAY1(2,J),ARRAY1(3,J))
400   CONTINUE
C PUT OUT NUM BOXCARS
      DO 800 K=1,NUM
      DO 600 J=1,49
      IF(ARRAY2(1,J).EQ.77.)GO TO 550
      CALL DRAWR(ARRAY2(2,J),ARRAY2(3,J))
      GO TO 600
550   CALL MOVER(ARRAY2(2,J),ARRAY2(3,J))
600   CONTINUE
800   CONTINUE
900   RETURN
```

APPENDIX B

User Routines

Most users will never need the routines described in this section and need not read this section. They are included so that the 5% of our customers who require special features not supported by the standard routines may conveniently add them to the package. With such flexibility there are many opportunities to make errors. We therefore strongly suggest that the programmer become familiar with the standard features of the package before experimenting with the user routines. It is recommended that a standard plot close to the desired special plot be generated and debugged first before linking in any special routines. It is also suggested that the user routines be tested as much as is practical alone before they are merged with the standard package.

There are currently six user hooks supplied in the AG-II package:

> Two routines, UMNMX and UPOINT, are used to expand a user defined short form.
>
> Two routines, ULINE and USERS, are used to implement special line types and special symbols.
>
> One routine, USESET, is used to pass user defined labels to the axis labeling routines.
>
> One routine, SOFTEK, is used in conjunction with the Tektronix Character Generation System.

These are described briefly below.

If the user wishes to have an unsupported short form, he must replace both UMNMX and UPOINT. UMNMX will return a minimum and maximum when given a short form, and UPOINT will return X, Y coordinates when given the same short form. UPOINT may also be written as a function of the X array in the form Y=f(x). UMNMX and UPOINT are invoked if the first entry of the data is less than -2.

If the user needs a different type of line such as extra heavy or a stepped line, ULINE can be rewritten to support this function. The routine is invoked once for each point in the plot to draw the desired pattern. The routine is controlled by specifying a negative line type less than -10. If several patterns are required, several negative line types may be chosen with ULINE executing a case branch.

A special symbol may be added to the package by recoding USERS. This routine is called once for each point plotted if the symbol type is set to a negative value. This routine may then perform any function desired by the user. During the development period, USERS was used to implement bar charts to check concepts connected to bar charts. This routine may be used to make point dependent symbols such as a routine to print out the value of a point on the plot.

The fifth routine is designed to help the user use the standard label routine to put non-standard labels on an axis. At each labeled tic mark, USESET is called with an array in which can be placed suitable character codes right justified. These are processed by LABEL to provide justification, and are printed on the labels. LABEL calls USESET if the label type is negative.

Subroutine UMNMX

USED: Called by MNMX to resolve user (non-standard) short forms. A non-standard short form data array is defined to be any array whose first entry is less than -2.

```
CALL UMNMX(ARRAY,AMIN,AMAX)
```

PURPOSE: This routine calculates the minimum and maximum of the given short form and returns the lesser of that minimum and AMIN, and the greater of that maximum and AMAX. This routine will only be invoked for short forms not supported by standard code.

ARGUMENTS:

ARRAY	Input array (short form)	
	ARRAY(1)	Negative number representing type. Must not be one currently supported.
	ARRAY(2)	Number of points represented in this array.
AMIN	Input: current minimum data value. Output: The minimum of all data values or AMIN	
AMAX	Input: current maximum Data value. Output; The maximum of all data values or AMAX.	

UMNMX is always used with the function UPOINT.

Function: UPOINT

USED: Called by DATGET to resolve user short forms

```
V = UPOINT(ARRAY,I,OLDONE)
```

PURPOSE: To provide a data value, on demand, to be plotted.

ARGUMENTS: ARRAY Input array (short form)

ARRAY(1)	negative number representing type.
ARRAY(2)	number of points

I number (Index) of point being plotted. Runs from 1 to number of points in steps of STEPL

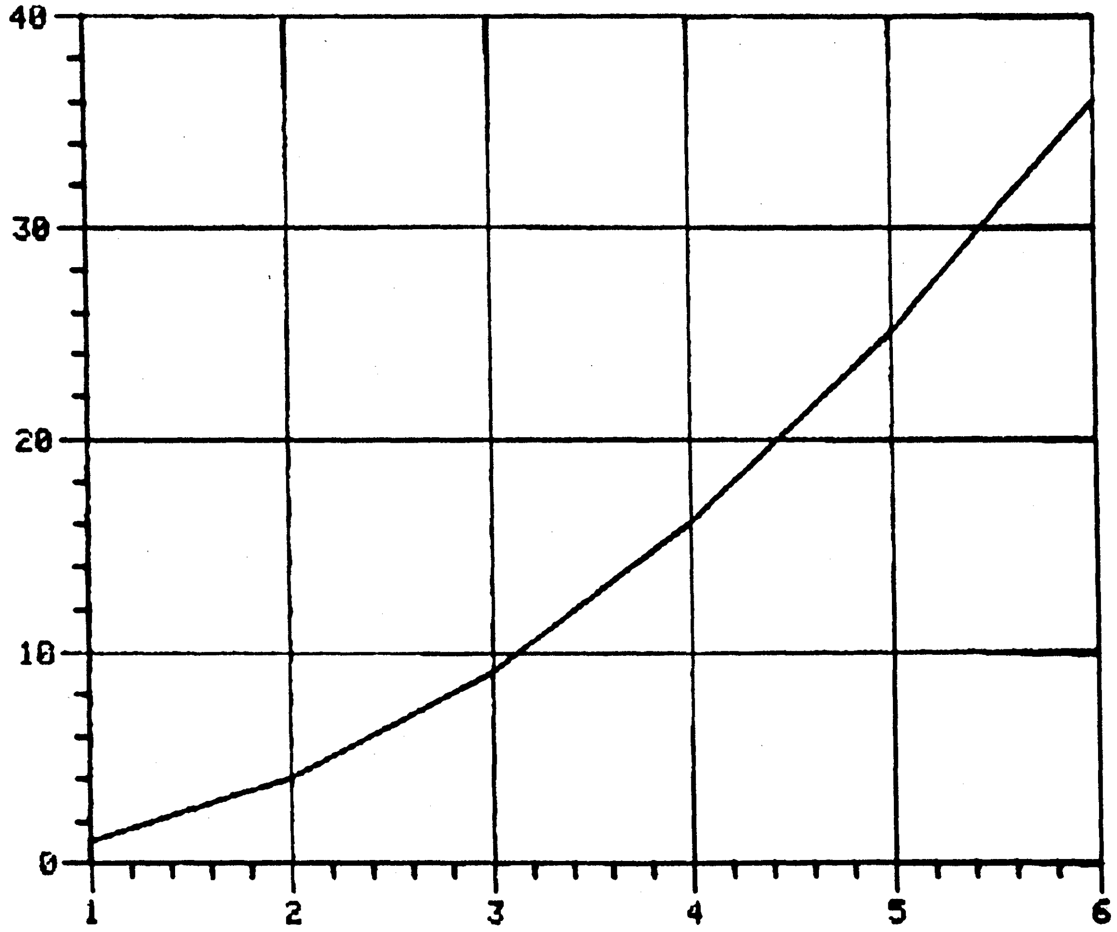
OLDONE This is the previous value calculated by DATGET.

```
     Y = f(x) may be coded as Y = f(OLDONE)
     X = f(OLDONE) is really equivelent to:
         X = f(Y(I-1))
```

```
      DIMENSION XDATA(7),YDATA(2)
      DATA XDATA/6.,1.,2.,3.,4.,5.,6./
      DATA YDATA/-10.,6./
      CALL INITT(120)
      CALL BINITT
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
      CALL FINITT(0,700)
      STOP
      END

      FUNCTION UPOINT(ARRAY,I,OLDONE)
C     Y=X SQUARED
      UPOINT=OLDONE*OLDONE
      RETURN
      END

      SUBROUTINE UMNMX(ARRAY,AMIN,AMAX)
      DIMENSION ARRAY(2)
      AMIN=1.
      AMAX=36.
      RETURN
      END
*
```



Subroutine ULINE

USED:	Called by CPLOT once for each point if line type is less than -10.

```
CALL ULINE(X,Y,I)
```

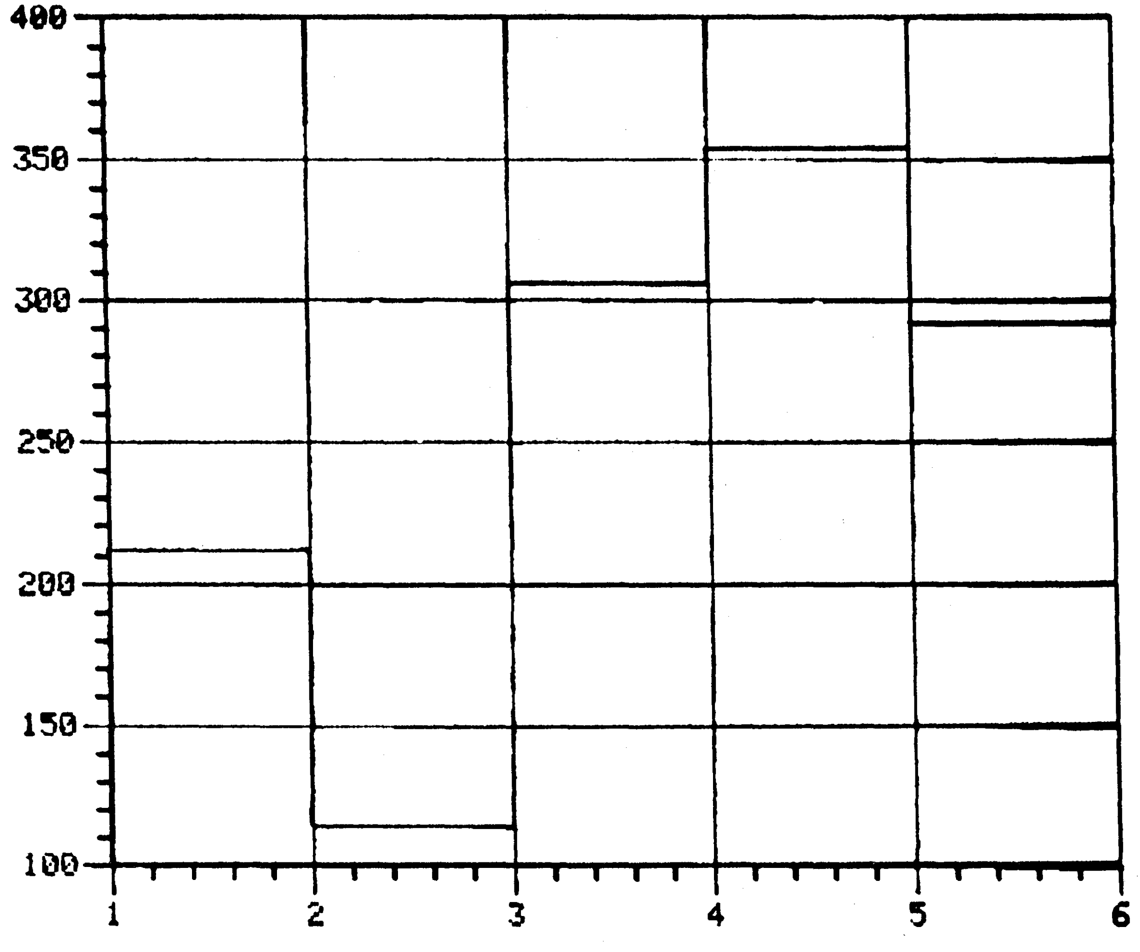
PURPOSE	This routine permits the user to do special moves and draws between points. It should be used for features occurring between points. A similar routine is provided for features occurring at points.

ARGUMENTS:	X	X (Horizontal) coordinate in data space
	Y	Y (Vertical) coordinate in data space
	I	Number (Index) of the data point in question. I varies from 1 to number of points in increments of STEPL.

HT

```
        DIMENSION XDATA(7),YDATA(7)
        DATA XDATA/6.,1.,2.,3.,4.,5.,6./
        DATA YDATA/6.,211.,114.,306.,354.,291.,325./
        CALL INITT(120)
        CALL BINITT
        CALL LINE(-1)
        CALL CHECK(XDATA,YDATA)
        CALL DSPLAY(XDATA,YDATA)
        CALL FINITT(0,700)
        STOP
        END

        SUBROUTINE ULINE(X,Y,I)
        IF(I.EQ.1)GO TO 100
        CALL DRAWA(X,YOLD)
        CALL DRAWA(X,Y)
 100    YOLD=Y
        RETURN
        END
```



Subroutine: USERS

USED: Called by BSYM once for each point if symbol type (CSYMBL) is less than zero.

```
CALL USERS(X,Y,ISYM)
```

PURPOSE: This routine permits the user to construct his own symbols at each data point. It has more flexibility than the standard TEKSYM in that the actual data value is available.

ARGUMENTS:

X	X (horizontal) data value of point
Y	Y (Vertical) data value of point
ISYM	The desired symbol code (will always be less than zero)

PROGRAMMING CONSIDERATIONS:

Normally a move will be made to the point represented by X,Y before entry into this subroutine. This subroutine is invoked even if the point is outside of the visible window. It is therefore the users responsibility to use virtual moves and draws ,or check for visibility. A GENFLG value of D indicates symbol will be within the window.

Example: See Figure A.2.

Subroutine USESET

USED: This routine is called by label for each labeled tic mark if a label type is less than zero.

```
CALL USESET(FNUM,IWIDTH,NBASE,LABELI)
```

PURPOSE: This routine is provided for people who wish to provide their own labels on the tic marks.

ARGUMENTS:

FNUM data value represented by tic mark. For linear data FNUM is the unscaled value of the tic mark.

For logarithmic data FNUM is $\log_{10}$ of the tic mark value.

For calendar data FNUM is the number of label intervals numbered from the first period of the initial year and incremented continuously for the entire axis.

Examples: For two years of months starting in January, FNUM would be 1, 2, 3, ...12, 13,14, ...23,24, incrementing by months.

For a two year bi-monthly series, FNUM would be 2,4,6,...20,22,24.

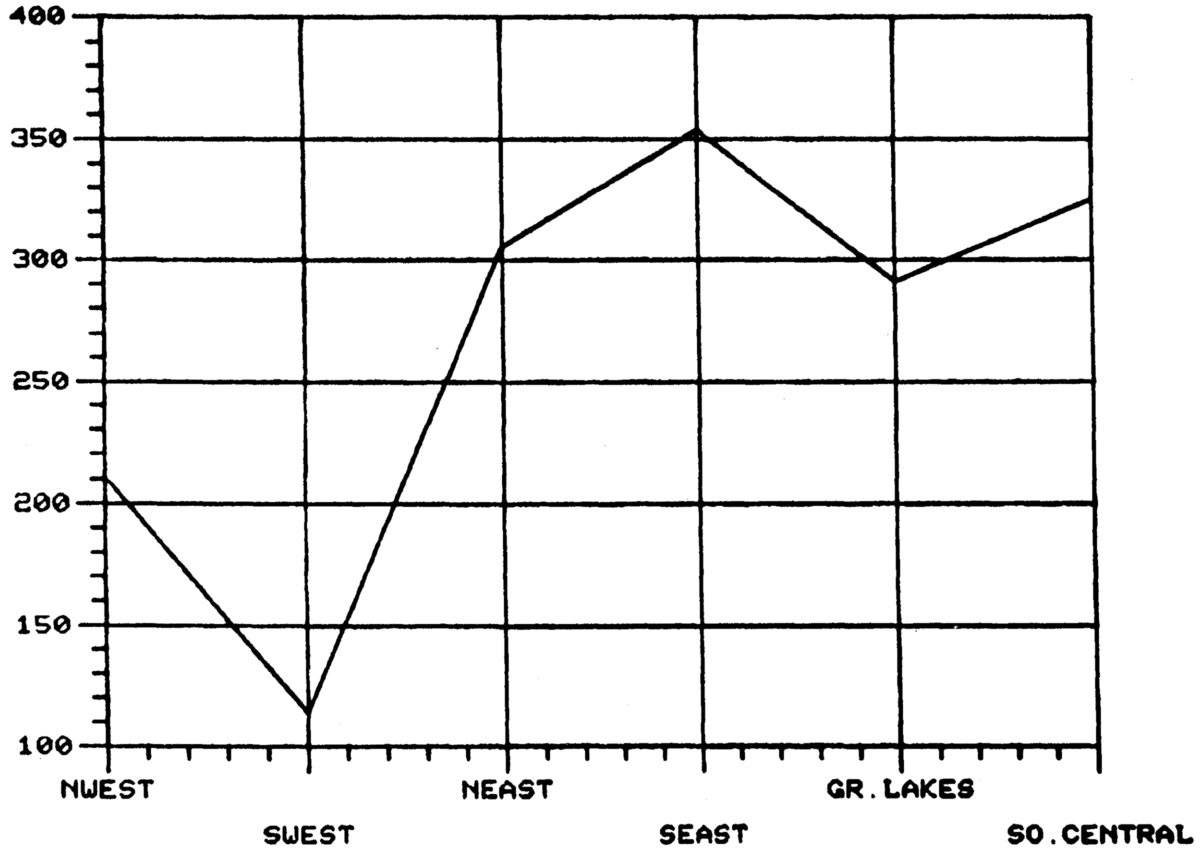
For two years of months beginning in July, FNUM would be 7,8,9,...28,29,30.

IWIDTH width of label. IWIDTH is specified by the AG-II call to USESET. The user must pass out a label of this width. (IWIDTH is 20 in distributed version of AG-II.)

NBASE Pointer to COMMON section of related axis. Will be equal to either IBASEX(0) or IBASEY(0).

LABELI Integer array of length IWIDTH to be filled with character codes. Actual label should be right justified, and leading unused characters should be filled with blanks (code 32).

```
C * EXAMPLE OF A USER DEFINED LABEL
      DIMENSION XDATA(7),YDATA(7)
      DATA XDATA/6.,1.,2.,3.,4.,5.,6./
      DATA YDATA/6.,211.,114.,306.,354.,291.,325./
      CALL INITT(30)
      CALL BINITT
      CALL XLAB(-1)
      CALL XWDTH(10)
      CALL CHECK(XDATA,YDATA)
      CALL DSPLAY(XDATA,YDATA)
      CALL FINITT(0,700)
      STOP
      END
C ** SUBROUTINE TO DEFINE USER DEFINED LABEL
      SUBROUTINE USESET (FNUM,IWIDTH,NBASE, LABELI)
      DIMENSION LABELS(10,6),LABELI(20)
C                                    N  W  E  S  T
C                                    S  W  E  S  T
C                                    N  E  A  S  T
C                                    S  E  A, S  T
C                           G  R     L  A  K  E  S
C                     S  O     C  E  N  T  R  A  L
      DATA LABELS/32,32,32,32,32,78,87,69,83,84,
     &            32,32,32,32,32,83,87,69,83,84,
     &            32,32,32,32,32,78,69,65,83,84,
     &            32,32,32,32,32,83,69,65,83,84,
     &            32,32,71,82,46,76,65,75,69,83,
     &            83,79,46,67,69,78,84,82,65,76/
      I=FNUM+0.5
      DO 100 K=1,IWIDTH
100   LABELI(K)=LABELS(K,I)
      RETURN
      END
```



Subroutine: SOFTEK(ISYM)

USED: Called by BSYMS to supply non-standard symbols from an independent Character Generation system such as the Tektronix 4010A05 PLOT 10 Character Generation System. A value greater than 127 in CSYMBL causes SOFTEK to be called.

```
CALL SOFTEK(ISYM)
```

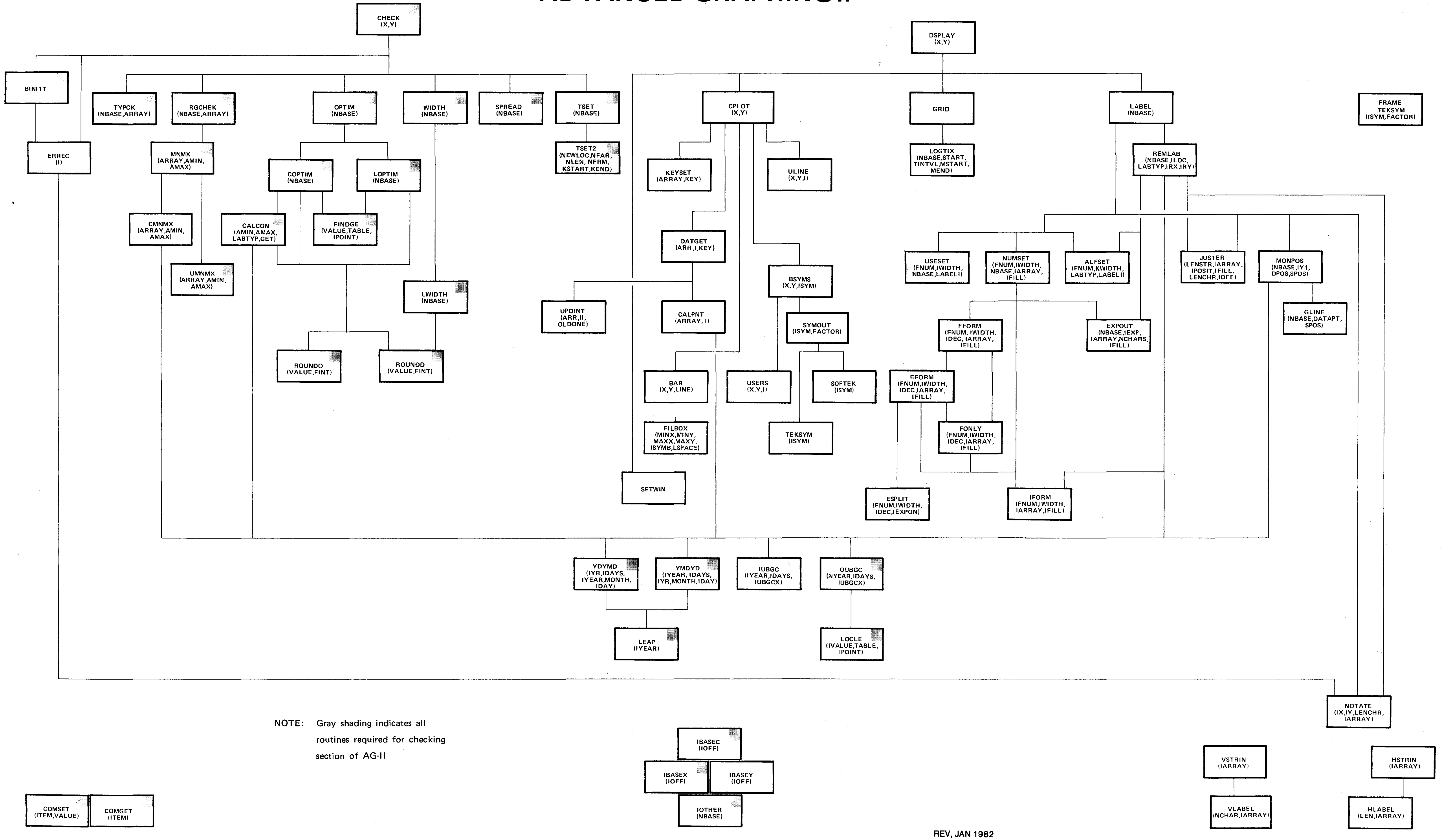
PURPOSE: This routine allows symbols to be used from an independent character generation system.

ARGUMENTS: ISYM is the symbol code (will always be greater than 127).

ASCII CODE CHART

BITS			B7 B6 B5	Ø Ø Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
B4	B3	B2	B1	CONTROL		HIGH X & Y GRAPHIC INPUT		LOW X		LOW Y	
Ø	Ø	Ø	Ø	NUL 0	DLE 16	SP 32	0 48	@ 64	P 80	` 96	p 112
Ø	Ø	Ø	1	SOH 1	DC1 17	! 33	1 49	A 65	Q 81	a 97	q 113
Ø	Ø	1	Ø	STX 2	DC2 18	" 34	2 50	B 66	R 82	b 98	r 114
Ø	Ø	1	1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø	1	Ø	Ø	EOT 4	DC4 20	$ 36	4 52	D 68	T 84	d 100	t 116
Ø	1	Ø	1	ENQ 5	NAK 21	% 37	5 53	E 69	U 85	e 101	u 117
Ø	1	1	Ø	ACK 6	SYN 22	& 38	6 54	F 70	V 86	f 102	v 118
Ø	1	1	1	BEL 7	ETB 23	' 39	7 55	G 71	W 87	g 103	w 119
1	Ø	Ø	Ø	BS 8	CAN 24	(40	8 56	H 72	X 88	h 104	x 120
1	Ø	Ø	1	HT 9	EM 25	) 41	9 57	I 73	Y 89	i 105	y 121
1	Ø	1	Ø	LF 10	SUB 26	* 42	: 58	J 74	Z 90	j 106	z 122
1	Ø	1	1	VT 11	ESC 27	+ 43	; 59	K 75	[91	k 107	{ 123
1	1	Ø	Ø	FF 12	FS 28	, 44	< 60	L 76	\ 92	l 108	¦ 124
1	1	Ø	1	CR 13	GS 29	– 45	= 61	M 77	] 93	m 109	} 125
1	1	1	Ø	SO 14	RS 30	. 46	> 62	N 78	^ 94	n 110	~ 126
1	1	1	1	SI 15	US 31	/ 47	? 63	O 79	_ 95	o 111	RUBOUT (DEL) 127

ADVANCED GRAPHING II


CHECK (X,Y)
DSPLAY (X,Y)
BINITT
ERREC (I)
TYPCK (NBASE,ARRAY)
RGCHEK (NBASE,ARRAY)
OPTIM (NBASE)
WIDTH (NBASE)
SPREAD (NBASE)
TSET (NBASE)
TSET2 (NEWLOC,NFAR, NLEN, NFRM, KSTART,KEND)
CPLOT (X,Y)
GRID
LABEL (NBASE)
FRAME TEKSYM (ISYM,FACTOR)
MNMX (ARRAY,AMIN, AMAX)
CMNMX (ARRAY,AMIN, AMAX)
UMNMX (ARRAY,AMIN, AMAX)
COPTIM (NBASE)
LOPTIM (NBASE)
CALCON (AMIN,AMAX, LABTYP,GET)
FINDGE (VALUE,TABLE, IPOINT)
LWIDTH (NBASE)
ROUNDO (VALUE,FINT)
ROUNDD (VALUE,FINT)
KEYSET (ARRAY,KEY)
ULINE (X,Y,I)
DATGET (ARR,I,KEY)
BSYMS (X,Y,ISYM)
UPOINT (ARR,II, OLDONE)
CALPNT (ARRAY, I)
SYMOUT (ISYM,FACTOR)
BAR (X,Y,LINE)
USERS (X,Y,I)
SOFTEK (ISYM)
FILBOX (MINX,MINY, MAXX,MAXY, ISYMB,LSPACE)
TEKSYM (ISYM)
SETWIN
LOGTIX (NBASE,START, TINTVL,MSTART, MEND)
REMLAB (NBASE,ILOC, LABTYP,IRX,IRY)
USESET (FNUM,IWIDTH, NBASE,LABELI)
NUMSET (FNUM,IWIDTH, NBASE,IARRAY, IFILL)
ALFSET (FNUM,KWIDTH, LABTYP,LABELI)
JUSTER (LENSTR,IARRAY, IPOSIT,IFILL, LENCHR,IOFF)
MONPOS (NBASE,IY1, DPOS,SPOS)
GLINE (NBASE,DATAPT, SPOS)
FFORM (FNUM, IWIDTH, IDEC, IARRAY, IFILL)
EXPOUT (NBASE,IEXP, IARRAY,NCHARS, IFILL)
EFORM (FNUM,IWIDTH, IDEC,IARRAY, IFILL)
FONLY (FNUM,IWIDTH, IDEC,IARRAY, IFILL)
ESPLIT (FNUM,IWIDTH, IDEC,IEXPON)
IFORM (FNUM,IWIDTH, IARRAY,IFILL)
YDYMD (IYR,IDAYS, IYEAR,MONTH, IDAY)
YMDYD (IYEAR, IDAYS, IYR,MONTH,IDAY)
IUBGC (IYEAR,IDAYS, IUBGCX)
OUBGC (NYEAR,IDAYS, IUBGCX)
LEAP (IYEAR)
LOCLE (IVALUE,TABLE, IPOINT)
NOTATE (IX,IY,LENCHR, IARRAY)
IBASEC (IOFF)
IBASEX (IOFF)
IBASEY (IOFF)
IOTHER (NBASE)
VSTRIN (IARRAY)
HSTRIN (IARRAY)
VLABEL (NCHAR,IARRAY)
HLABEL (LEN,IARRAY)
COMSET (ITEM,VALUE)
COMGET (ITEM)

NOTE: Gray shading indicates all routines required for checking section of AG-II

REV, JAN 1982

SUBROUTINE AND FUNCTION INDEX

*Terminal Control System Routines, See Tektronix User Manual.

SUBJECT INDEX