TEKTRONIX®

PLOT 10

4010A02

ADVANCED GRAPHING II

SYSTEM MANUAL

TEKTRONIX®

PLOT 10

4010A02

ADVANCED GRAPHING II

SYSTEM MANUAL

Tektronix, Inc.
P.O. Box 500
Beaverton, Oregon 97077
070-2245-00

First Printing FEB 1977

SYSTEM SECTIONS

CONTENTS

Table of Figures

INTRODUCTION

The Advanced Graphing II System Manual is designed to be used with the User Manual. Page and section numbering follows that of the User Manual. The following ten sections of this manual describe the internal subroutines along with other useful information for the system programmer who may need to reduce the size of the package or deal with the fundamentals of the package in any other way.

Sections 5 and 6 contain general system information concerning COMMON Table Reference and flow chart description. Sections 7 through 12 contain descriptions of algorithms and internal subroutines. Section 13 contains information necessary to "prune down" the size of the package. Section 14 contains user written subroutines and functions.

Using the Advanced Graphing II User Sections and System Sections in conjunction, the programmer should have a complete view of the package.

SECTION 5

COMMON TABLE REFERENCE

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, labeled 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 real values, and are initialized by BINITT to reasonable default values.

Basically, the COMMON Table is divided into three major sections which can be accessed by the general user:

1. Curve setting variables which are located with the COMMON Section Pointer IBASEC.

2. X-Axis variables which are located with the COMMON Section Pointer IBASEX.

3. Y-Axis variables which are located with the COMMON Section Pointer IBASEY.

In addition to these sections which are defined in the User's sections a group of internal variables at the beginning of the table define the extent of COMMON and the extent of each major section, and the vector variables define the starting points of the sections.

All access to the global COMMON Table is via a group of intermediate functions and subroutines which are individually described in Section 4.4.

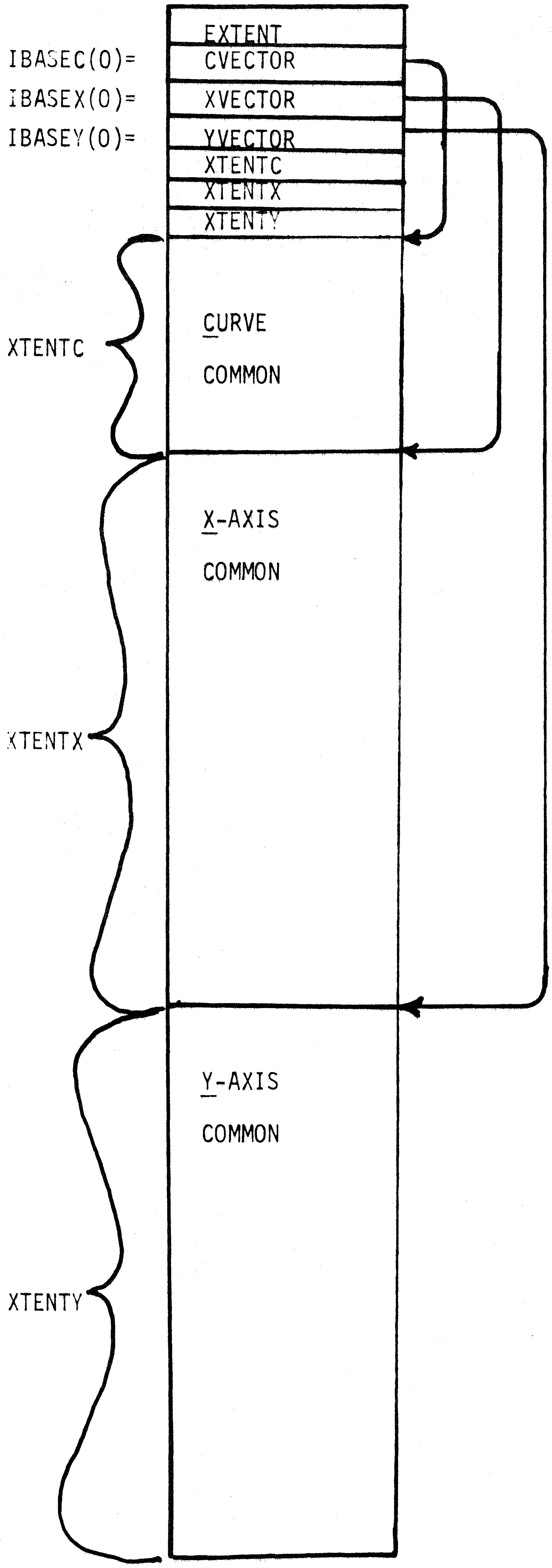
Two advantages are inherent in this structure:

1. The code written to compute axis information may be axis independent since the variables necessary for each axis and their locations within the respective sections are the same. The variable pointer NBASE is used in the code to refer to either IBASEX or IBASEY, depending on which axis is being computed.

2. The Table may be expanded without altering every routine. Only the extent of the expanded section need be changed; the locations of the old variables within the section remain the same.

The structure of the COMMON Table is shown in Figure 1. The variables in COMMON are listed in the chart in Figure 2, followed by codes which refer to the descriptions on pages 5-5 through 5-9.

THE STRUCTURE:



THE REASONS:

1. AXIS-INDEPENDENT CODE -
   Cuts the number of routines nearly in half.

2. Table may be expanded without altering every routine.

Structure of AG-II COMMON

Figure 1

The following chart provides a quick guide to determine which subroutines reference which variables in COMMON.

All COMMON variables are shown vertically and COMMON referencing subroutines horizontally. The code numbers in the grid correspond to numbered comments on the following pages.

The comments are numbered so that all comments referring to a given subroutine appear contiguously.

= GET

= SET

COMMON REFERENCING

LOCATION	COMMON VARIABLE	BAR	CHECK	COPTIM	CPLOT	DINITX/DINITY	DLIMX/DLIMY	EXPOUT	FRAME	GLINE	GRID	HBARST	LABEL	LOGTIX	LOPTIM	LWIDTH	MNMX	NUMSET	OPTIM	PLACE	REMLAB	RGCHEK	SETWIN	SLIMX/SLIMY	SPREAD	TEKSYM	TSET	TYPCK	VBARST	WIDTH
0	CLINE				14							33																	70	
1	CSYMBL	1			15							1																	1	
2	CSTEPS				16																									
3	CINFIN				17								35		43		46					53								
4	CNPTS				18												18											18		
5	CSTEPL				19																									
6	CNUMBR																													
7	CSIZES	2											2													66			2	
8	CSIZEL	3											3																3	
0	CXNEAT/CYNEAT			5											5															
1	CXZERO/CYZERO																					54								
2	CXLOC/CYLOC										28		36														57			
3	CXLAB/CYLAB			6						26			26		6										60					60
4	CXDEN/CYDEN			7											7										61					
5	CXTICS/CYTICS			8 / 8							29		37		8 / 8										62					
6	CXLEN/CYLEN										30		38														30			
7	CXFRM/CYFRM										30	34 / 34															30		34 / 34	
8	CXMTCS/CYMTCS										29			42	44 / 44												29			
9	CXMFRM/CYMFRM																										30			
10	CXDEC/CYDEC					20										20		47												
11	CXDMIN/CYDMIN			9 / 9		21	22								9 / 9	45					51	55 / 55	57							63
12	CXDMAX/CYDMAX			9 / 9		21	22								9 / 9	45					51	55 / 55	57							63
13	CXSMIN/CYSMIN	4		10					25		31		39		10					50			57	59	63		67			
14	CXSMAX/CYSMAX	4		10					25		31		39		10					50			57	59	63		67			
15	CXTYPE/CYTYPE			11				23			11		11		11			48	49			56	58		11			69		11
16	CXLSIG/CYLSIG			12											12	45														
17	CXWDTH/CYWDTH					20							40			45									64 / 64					71
18	CXEPON/CYEPON					20							40			45					52									
19	CXSTEP/CYSTEP												40												65					72
20	CXSTAG/CYSTAG		72										40												65 / 65					72
21	CXETYP/CYETYP							24					40			45														73
22	CXBEG/CYBEG									27	32																68			
23	CXEND/CYEND									27	32																68			
24	CXMBEG/CYMBEG										32																68			
25	CXMEND/CYMEND										32																68			
26	CXAMIN/CYAMIN			13									41		13															
27	CXAMAX/CYAMAX			13									41		13															

Figure 2

1. CSYMBL indicates the type of bar shading for vertical or horizontal Bar charts. See Bar Chart samples in Section 2.8, pages 38, 39,

2. CSIZES is the width of the Bar in raster units.

3. The bar charting routine uses CSIZEL to obtain the distance between shading marks in raster units.

4. BAR uses the screen min and max as limits for drawing bar chart bars.

5. The NEAT Flag indicates whether 'neat' tics are calculated in LOPTIM or COPTIM.

6. COPTIM and LOPTIM require the label type to establish neat tic mark intervals.

7. COPTIM and LOPTIM require the density factor to compute the tic interval.

8. LOPTIM and COPTIM determine the number of major tic mark intervals if they are not already set by the user.

9. LOPTIM and COPTIM require the data min and max to compute the tic mark interval. If neat tics are specified, the min and max may be rounded outward.

10. LOPTIM and COPTIM require the screen min and max to compute the length of the tic mark intervals.

11. COPTIM. If the label type is not specified, the transformation type determines the label type.

12. The routine LOPTIM or COPTIM determines and enters the position of the least significant digit of the labels.

13. LOPTIM or COPTIM sets the min and max values of the tic mark labels for each axis.

14. CLINE specifies the type of line (dashed, solid, bar) which CPLOT is to draw in plotting the curve.

15. CPLOT accesses CSYMBL which indicates the type of plotting symbol for each plotted point.

16. CPLOT accesses CSTEPS which indicates the frequency of symbols on a plot.

17. Machine infinity is required in CPLOT to distinguish "missing data" data points. If a data value equals infinity, the data value is to be skipped and a space left for it.

18. CNPTS indicates the number of points in a non-standard array. This information is required by CPLOT, MNMX, and RGCHEK.

19. CSTEPL indicates the frequency of points plotted. 1 indicates that every point is plotted; 2 indicates every other point, etc.

20. CXDEC or CYDEC indicates the number of decimal places to appear in the tic mark labels. Label width and label scale factor are compiled and set by LWIDTH and may be re-initialized for the appropriate axis by DINITX or DINITY.

21. The data min and max for an axis are reset to zero by DINITX or DINITY.

22. The data min and max may be set by DLIMX or DLIMY.

23. EXPOUT checks TYPE for logarithmic axes if logarithmic tic mark labels are generated without the X (multiplier sign).

24. EXPOUT requires the type of remote exponent to set up and produce the remote labels.

25. FRAME requires the screen min and max to draw a bar around the data window.

26. LABEL and GLINE require the label type in order to select the proper label for the axis.

27. GLINE inserts grid lines or tic marks for month labels where they actually appear on the number line (they vary in width according to the number of days in the month), and so requires the end points of the tic marks.

28. GRID requires the location of each axis for drawing the axis and tic marks.

29. GRID requires the number of tic intervals in order to form the grid.

30. The tic mark length and form are required by GRID to draw the grid and TSET to compute the end points of the grid lines.

31. GRID requires the screen min and max in order to draw the grid lines and axis lines.

32. GRID obtains the calculated end points for the major and minor tic marks from these locations.

33. HBARST sets CLINE to 3 to indicate horizontal bar chart to CPLOT.

34. HBARST and VBARST. The vertical and horizontal bar chart set up involves checking the tic mark form to prevent grid lines from running the length of the bars. If the form is such that this would occur, it is changed.

35. Machine infinity is required in LABEL to set linear and log mod function values.

36. LABEL requires the axis location for label placement.

37. To position the tic mark labels, LABEL requires the number of tic intervals.

38. The length of the tic marks is required in LABEL to space the tic mark labels properly.

39. To position the labels, LABEL requires the screen min and max.

40. LABEL requires the width, scale factor, step frequency, stagger flag and type of remote exponent for building the labels.

41. LABEL obtains the calculated label min and max and calculates the tic labels from these values.

42. If the axis is logarithmic and the number of minor tic marks is set at -1 by LOPTIM, LOGTIX produces minor tic marks on a logarithmic decade. If the number of minor tics is set to -2 , the logarithmic minor tics are drawn along with numbers from 2-9 if there is room. See examples on page 35 of the User's Manual.

43. Machine infinity is required in LOPTIM to establish a minimum log value if the minimum is set to zero.

44. LOPTIM sets the number of minor tic intervals if not set by the user. See #34 above for exception for logarithmic minor tics.

45. LWIDTH determines the maximum width of the tic labels based on the least significant digit (and axis scale factor), and using the data min and max.

46. To recognize and exclude "missing data" points, MNMX needs machine infinity.

47. NUMSET. Numeric label string building requires the number of decimal places to form the tic label string.

48. NUMSET requires the transformation type to determine if exponential labels are to be generated.

49. OPTIM determines which routine to call (LOPTIM or COPTIM) based on the transformation type.

50. PLACE sets the screen min and max.

51. REMLAB, the remote label routine, uses the data min and max to create remote labels for calendar data.

52. REMLAB requires the axis scale factor for generation of the remote label.

53. Machine infinity is required by RGCHEK to establish starting points for the min/max searches.

54. RGCHEK checks for zero suppression before setting the range.

55. RGCHEK checks the values of the data min and max, and if they are equal, assumes they have not been set. It then calls the appropriate min/max routine to find the min and max and sets the min and max in COMMON.

56. TYPE helps RGCHEK to determine which MIN/MAX routine to call.

57. The data min and max and screen min and max are required by SETWIN to establish the data space and screen window.

58. SETWIN. The transformation TYPE is required in setting the data to screen transformation.

59. SLIMX and SLIMY are the primary methods of directly setting the screen window.

60. The label type governs the width of labels for month and day labels in SPREAD and WIDTH.

61. The density factor offsets the space between labels controlled by SPREAD.

62. SPREAD requires the number of tic intervals to compute the label spacing and determine the frequency of labels.

63. SPREAD requires the screen min/max to check inter-label spacing.

64. SPREAD requires the maximum label width in determining inter-label spacing. This width may be changed in SPREAD if the label type is days or months which can be reduced to 3 meaningful characters.

65. SPREAD sets the label step frequency and the stagger flag if the labels are too wide for the space available.

66. CSIZES is a scale factor for the symbol size of software plotting symbols (TEKSYM).

67. TSET sets up the tic mark end points according to the axis location and screen min and max.

68. TSET computes the end point locations of the tic marks and sets these values into the table.

69. TYPCK decodes calendar form arrays and establishes the transformation type.

70. VBARST sets CLINE to 2 to indicate vertical bar charts to CPLOT.

71. WIDTH specifies the maximum tic label width, or calls LWIDTH which determines this value if linear numeric labels are required.

72. If the tic mark labels for log axes are specified as numbers expressed in words such as TEN or ONE HUNDRED, WIDTH sets the step to 1 and the stagger to 2 (2 levels). If this is done, SPREAD is <u>not</u> called by CHECK.

73. WIDTH requires the type of remote exponent to compute the label width if the axis is logarithmic.

SECTION 6

SYSTEM DESCRIPTION

The AG-II System subroutines may be divided into two distinct groups - Table checking and display. The checking subroutines, controlled by CHECK, do not produce output to the screen. This general description of the system should be read with reference to the System Diagram in Appendix C.

6.1 Table Checking

CHECK accesses the COMMON Table with one of the axis pointers (IBASEC, IBASEX, or IBASEY) to see if it has been initialized by BINITT and controls the other Table Checking routines. If BINITT has not been called, CHECK calls it along with ERREC which issues an error message. Any Table setting calls which have been made previously will be nullified by the subsequent call to BINITT.

CHECK next calls TYPCK. If the data type is calendar form, TYPCK decodes the array to determine the actual transformation (days, months, years, etc.) If the type is not calendar, TYPCK takes no action, and control returns to CHECK.

RGCHEK is called next and in turn calls MNMX if the data limits have not been set previously. If a calendar axis has been specified, control is further passed to CMNMX. MNMX is called once for each axis and is passed the array name for that axis. UMNMX is a dummy, user specifiable routine, which will calculate and find a user defined minimum and maximum if the user generates his own data points with UPOINT (see page 281 of this manual). If the user has written his own UPOINT routine, he must either provide his own UMNMX routine to determine the data minimum and maximum or call DLIMX and/or DLIMY prior to calling CHECK.

After RGCHEK has set the data minimum and maximum and entered them into the Table, OPTIM is called. OPTIM determines from the transformation type (CXTYPE and CYTYPE) which of the two grid resolving routines are to be used - COPTIM for calendar axes or LOPTIM for linear or logarithmic axes.

COPTIM calls CALCON which converts the data minimum and maximum from UBGC days to the label type specified. (Calendar data minimum and maximum are always in UBGC days, regardless of whether data is in days, weeks, periods, months, or years.) The label type will be same as the data type if not otherwise specified. For example, if data is monthly, labels will be monthly.

COPTIM uses predefined neat label units to round to neat intervals if specified. It also adjusts the grid so that the screen limits don't greatly exceed the data limits.

CALCON may be called for either of two purposes:

1. If parameter GET is true, CALCON converts from UBGC to label type. (The UBGC, or Universal Business Graphing calendar, is described in Section 4.3.

2. If GET is false, CALCON converts from the label type to UBGC.

Once CALCON converts the UBGC minimum and maximum to the neatly rounded calendar intervals, it is called a second time to convert these updated calendar intervals back into UBGC days. These values are the UBGC min/max of the plotting window.

FINDGE is a function used by both the OPTIM routines (COPTIM and LOPTIM) in the neat algorithm process. See page 200 for a detailed description of the neat algorithm.

Functions ROUNDU and ROUNDD are also used in the neat algorithm.

If the data is not calendar data, LOPTIM is called. LOPTIM is approximately comparable to COPTIM. It computes an ideal tic mark interval size based on the density factor which may be set by the user. (See Section 3.2.5) If neat tic mark intervals are desired, it then expands the range to meet neat tic mark requirements. After execution of LOPTIM or COPTIM, the minimum and maximum values of the axis labels are available and stored in the Table, in a position not normally accessed by the user. This saves the recomputation of the labels during the display sequence.

Next, CHECK calls WIDTH to determine the maximum width of each tic mark label. This process is somewhat more complicated for a linear axis than it is for a calendar axis. LWIDTH is called by WIDTH to determine the maximum width of conventional arithmetic labels. Depending on the scale factor and the axis, LWIDTH determines the number of digits, including the decimal point, to be used for a label.

SPREAD uses the information from WIDTH and OPTIM to determine the amount of space between the labels on the axis line. This is most important for the X axis, but is also important if there is not room on the Y axis for the height of the letters. On the X axis labels may be staggered; on the Y axis alternate labels will be omitted if there is not room.

Alphanumeric calendar labels use the same stagger procedure. If labels are still too long, they are shortened to three character abbreviations.

TSET establishes the end points of the major and minor tic marks. It works in conjunction with TSET2 in determining these values and placing them in the Table.

If only the screen location is altered following the first calls to CHECK and DSPLAY, TSET may be called instead of CHECK to update the Table.

6.2 Display Group

Four primary routines, SETWIN, CPLOT, GRID, and LABEL display the plots. CPLOT and GRID are called once, and LABEL is called twice - once for each axis.

SETWIN

Initially, SETWIN sets the transformation from virtual space to screen coordinates.

CPLOT

CPLOT is the actual curve plotting routine. It decodes all possible combinations of array types with the subroutine KEYSET. Data points are retrieved by DATGET which uses the user defined UPOINT, if necessary, or CALPNT which decodes calendar arrays into UBGC plot points. These routines supply the information for one data point location at a time.

CPLOT calls BAR if bar charts are used. BAR then calls FILBOX which draws the rectangular box and fills it with shading if specified.

BSYMS is a service routine which plots a symbol at a given location. CPLOT moves to a data point and calls BSYMS which draws the symbol at the current screen location. Routine TEKSYM supplies an assortment of software generated symbols. SOFTEK is a dummy which serves as an interface for the PLOT-10 Character Generation System. USYM is a user generated symbol plotting routine which serves as a user hook.

GRID

GRID retrieves the beginning and ending tic mark locations and draws the grid lines. For logarithmic axes with major tic marks of one decade, minor logarithmic tic marks may be inserted and labeled from 2 to 9 if there is room. LOGTIX performs this logarithmic tic mark operation.

LABEL

LABEL generates the major tic mark labels on the axis and calls REMLAB to put out the remote scale factors. The label strings are generated in either NUMSET or ALFSET, or by a user routine USESET.

The subroutines in the NUMSET group replace FORTRAN numeric conversion routines. NUMSET will appropriately convert a number into either a floating point form or an integer form. If a floating point number will not fit the label space, EFORM generates an exponential value.

After an alphanumeric label is produced, JUSTER determines the true length (see String Generation on page 222) and calculates the offset from the point of left, right, or center justification.

LABEL then calls NOTATE to display the string at the position specified by LABEL. The offset from JUSTER is added to the X coordinate. NOTATE uses HLABEL to put out the string, character by character.

After the tic mark labels are displayed, LABEL calls REMLAB for remote labels. Linear labels may require a remote scale factor; calendar data requires beginning year designation.

Monthly labels are positioned by MONPOS, and then GLINE draws the grid line in the correct place. The grid line beginning coordinate is returned to LABEL which uses the string from ALFSET to display the label.

SECTION 7

CHECKING SUBROUTINES

As was described in Section 3, the COMMON Table contains the AG-II variables. Following initialization and setting of variables by the user, internal routines set the remaining Table values. The OPTIM subroutines, which determine the tic mark values are described separately in Section 4 of this manual. Other checking routines follow.

7.1 Error Recovery Subroutine - ERREC

If the user has not called BINITT before CHECK, or INITT before BINITT, this subroutine issues a message notifying him of the error and its correction.

CALL ERREC (I)

Parameter Entered:

I is a value showing which subroutine has not been called.

1 - BINITT was called without INITT.

2 - CHECK was called without BINITT.

Description:

After BINITT or CHECK calls the preceding routine, it calls ERREC which prints the message:

ERROR RECOVERY (Bell) INITT (or BINITT) CALLED.

7.2 Data Type Checking Subroutine - TYPCK

Sets the data type in COMMON to coincide with the form specified in the calendar data array.

Calling Sequence:

```
CALL TYPCK (NBASE,ARRAY)
```

Parameters 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.)

ARRAY — is the data array for the appropriate axis.

Description:

If the calendar short form has been used for the data array, TYPCK determines what data type should be specified in CXTYPE or CYTYPE (years, months, periods, etc.) and sets it.

For any other data form, no action is taken.

7.3 Data Range Checking Subroutine - RGCHEK

Checks to see if the user has set the data minimum and maximum in COMMON, and sets them if the user has not.

Calling Sequence:

```
CALL RGCHEK (NBASE,ARRAY)
```

Parameters 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.)
ARRAY	is the data array for which the minimum and maximum are to be checked.

Description:

RGCHEK determines if the data limits (CXDMIN or CYDMIN and CXDMAX or CYDMAX have been set in COMMON. If not, it sets the minimum to infinity, (CINFIN) and the maximum to negative infinity, and calls MNMX which adjusts them to fit the data. The data limits are then set in the COMMON table.

7.4 Linear Label Computing Subroutine - LWIDTH

Computes the width and form of linear tic mark labels.

Calling Sequence:

```
CALL LWIDTH (NBASE)
```

Parameters 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:

LWIDTH obtains the data minimums and maximums, and the least significant digit from COMMON and determines the width of the labels, the number of decimal places required, and the shifting exponent, and sets the values of CXWDTH or CYWDTH, CXDEC or CYDEC, CXEPON or CYEPON in COMMON.

7.5 Label Postioning Subroutine - SPREAD

Determines how many labels will fit along the X or Y axis, and whether staggering will be necessary.

Calling Sequence:

CALL SPREAD (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:

SPREAD uses the width of label, the number of labels, and the length of the axis to compute how many labels will fit along each axis. With this information SPREAD sets CXSTEP or CYSTEP and CXSTAG or CYSTAG in COMMON. See Section 3.2.17 and 3.2.18.

7.6 Tic Mark Setting Subroutine - TSET

Determines the beginning and ending points of the tic marks.

Calling Sequence:

```
CALL TSET (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:

TSET determines the beginning and ending locations of the major and minor tic marks and sets these values in COMMON. TSET is called by CHECK.

TSET2 is called twice - once for major and once for minor tic marks.

7.7 Tic Mark Setting Subroutine - TSET2

Determines the beginning and ending points of the major and minor tic marks.

Calling Sequence:

```
CALL TSET2 (NEWLOC,NFAR,NLEN,NFRM,KSTART,KEND)
```

Parameters Entered:

NEWLOC	is the location of the axis
NFAR	is the location of the window edge opposite the axis.
NLEN	is the tic mark length from CXLEN or CYLEN in COMMON. This is a negative value if the axis location is on the opposite (top or righthand) side of the screen.
NFRM	is the tic mark form from CXFRM or CYFRM in COMMON.

Parameters Returned:

KSTART	is the starting point for the tic marks (in raster units).
KEND	is the ending point for the tic marks (in raster units).

Description:

TSET calls TSET2 to obtain the beginning and ending points of the major and minor tic marks.

SECTION 8

OPTIMUM TIC MARK SETTING SUBROUTINES

OPTIM and its companion routines LOPTIM (for linear or logarithmic data) and COPTIM (for calendar data) use the data already set in COMMON by BINITT and the user to produce appropriately spaced and labeled tic marks.

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

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 three characters in width or more than 150 raster units (1024 addressing) or 600 raster units (4096 addressing).

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 marks 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 major 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.

8.1 Tic Mark Density

The density algorithm is shown in figure 3. In the equations used, variables are as follows:

IDEN — is the density factor from the variable CXDEN or CYDEN in the COMMON Table.

FAC — is a factor based on the density variable IDEN minus 1 modulo 5.

IDIV — is the number of raster units between adjacent tic marks. Based on this algorithm, the number of raster units will be between the width of three characters and 150 (1024 addressing) raster units.

LEN — is the total length of the axis.

NTICS — is the resulting number of tic mark intervals.

MAXSCR — is the size of the terminal screen as specified in the TCS routine SEETRM.

FDEN1, FDEN2 — are intermediate variables in the calculation.

DENSITY ALGORITHM

MAJOR TICS

```
IFAC   =  MOD(IDEN,5)

FAC    =  IFAC + FRAC

FDEN1  =  MAXSCR/1022 * 150

FDEN2  =  (FDEN1 - LINWDT(3))/4.0

IDIV   =  MAX1(FDEN1-FDEN2*FAC,2.0)
```

MINOR TICS

The number of minor tic mark intervals is chosen from a table based on the tic mark label increment. If the number of intervals (MTCS) is 10, the number is adjusted downward according to the following specification.

If density < 9 MTCS=5

If density < 7 MTCS=2

TIC MARK DENSITY TABLE

	No Minor Tics	With Minor Tics
Sparse	1	6
	2	7
	3	8
	4	9
Dense	5	10

NOTE

If the number of tic marks is set, the density setting will have no effect.

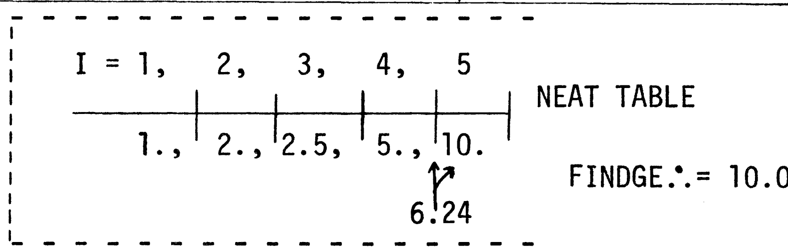
Figure 3

8.2 Neat Tic Mark Intervals

In the neat tic mark algorithm, the following variable terminology is used:

TINT	is the raw tic mark interval.
RANGE	is the range of data covered.
NTICS	is the number of tic marks derived from the variable CXTICS or CYTICS in COMMON.
LSIG	is the location of the least significant digit.
FINDGE	refers to the AG-II function FINDGE which searches a table for the nearest value greater than or equal to a test value.
FACTOR	a power of 10 coefficient which when used as a divisor of TINT yields a number between 1 and 10.

METHOD OF FINDING NEAT TIC INTERVAL

NEAT ALGORITHM		Example based on NTICS = 7
FIND RAW TIC INTERVAL	TINT = RANGE/FLOAT(NTICS)	62.4 = 437.0/7.0
TAKE INTEGER PART OF LOG OF TIC INTERVAL TO FORM FACTOR	LSIG = ALOG10(TINT) FACTOR = 10.0**(LSIG)	1 = ALOG10(62.4) 10.0 = 10.0**(1)
DIVIDE INTERVAL BY FACTOR YIELDS NUMBER BETWEEN 1 AND 10	TINT = FINDGE(TINT/FACTOR,TABLE,I) *FACTOR	6.24 = 62.4/10.0
SELECT VALUE FROM "NEAT" TABLE WHICH IS .GE. THE FACTORED INTERVAL		
NEAT TIC INTERVAL = 100 = 10.0x10.0		

NEAT TIC MARK VALUES ALGORITHM FIGURE 4

8.3 Optimum Calendar Tic Marks Subroutine - COPTIM

Determines the best tic mark values for calendar data.

Calling Sequence:

```
CALL COPTIM (NBASE)
```

Parameters 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:

If the data type is ≥ 3 and ≤ 8, indicating calendar type, OPTIM transfer control to COPTIM which checks the number of tic marks, the density factor if the number of tics is not specified, the neat tic mark flag, and the length of the axis. COPTIM then produces the results necessary for appropriately spaced and labeled tic marks. The description of OPTIM on page 4-16 describes the general sequence of events. Also See the neat and density algorithm descriptions on pages 8-4 and 8-5.

COPTIM

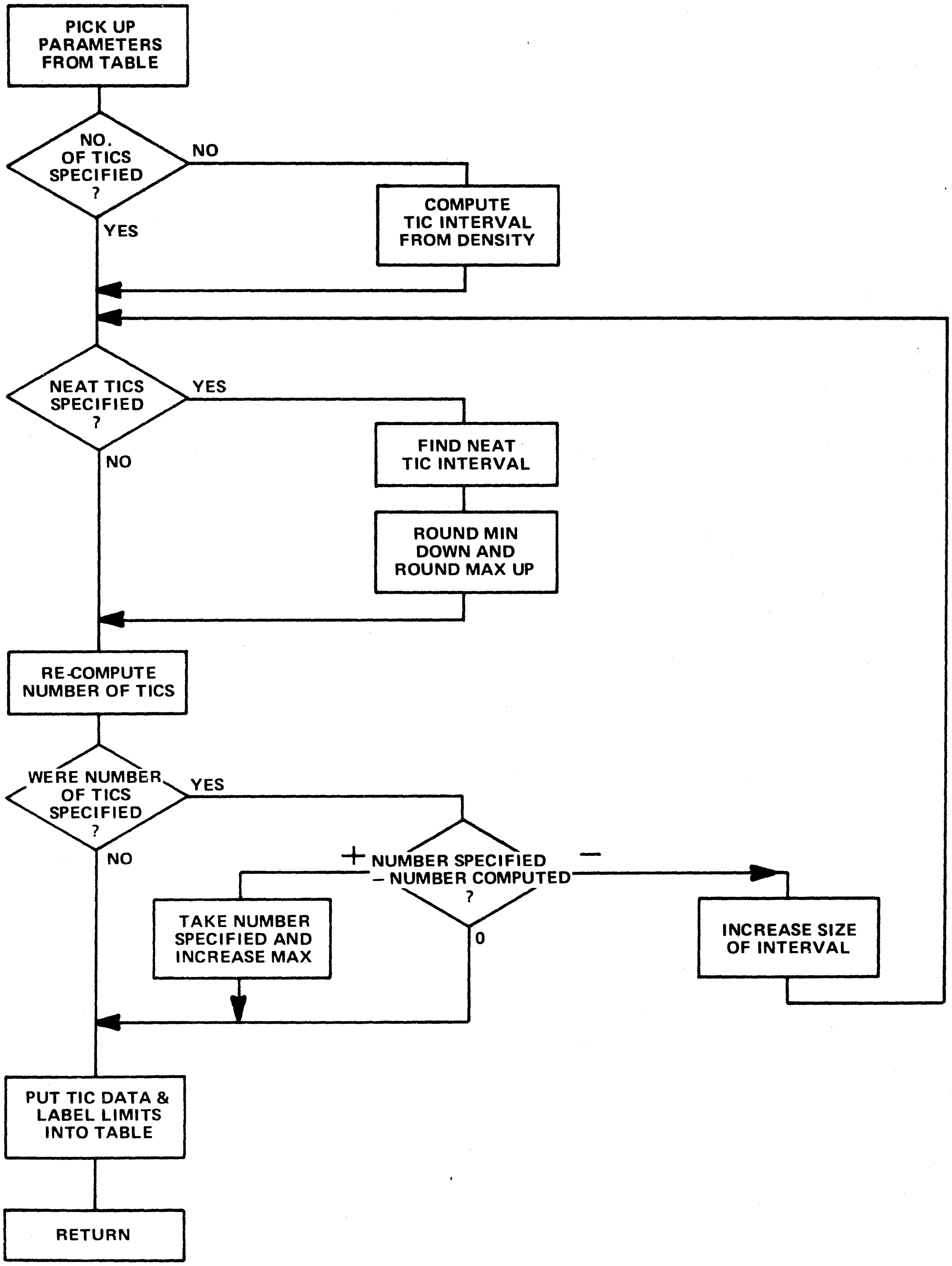


Figure 5

8.4 Optimum Linear or Logarithmic Tic Mark Subroutine - LOPTIM

Determines the appropriate labeling and spacing of the tic marks.

Calling Sequence:

CALL LOPTIM (NBASE)

Parameter Entered:

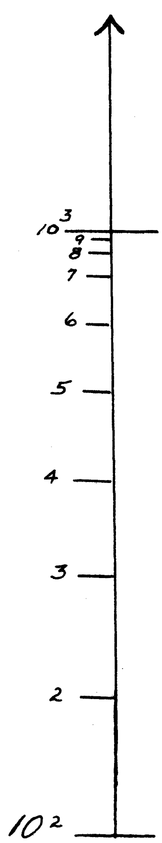
NBASE is the axis pointer, or the location in COMMON of the first item referring to the X or Y axis, whichever is here indicated. (IBASEX or IBASEY provides this location.)

Description:

OPTIM calls LOPTIM if the data type (CXTYPE or CYTYPE) is linear or logarithmic.

LOPTIM checks the label type to determine if the the data is linear or logarithmic, if neat tic marks are desired, and if the number of tics has been specified. If the number of tics is not specified, LOPTIM uses the density factor to determine the number.

If the type of data is logarithmic, and there is room between the major tic marks and the major tic marks are single decades, logarithmic minor tic marks and logarithmic labels are allowed.



Labeled Minor Logarithmic Tic Marks

LOPTIM

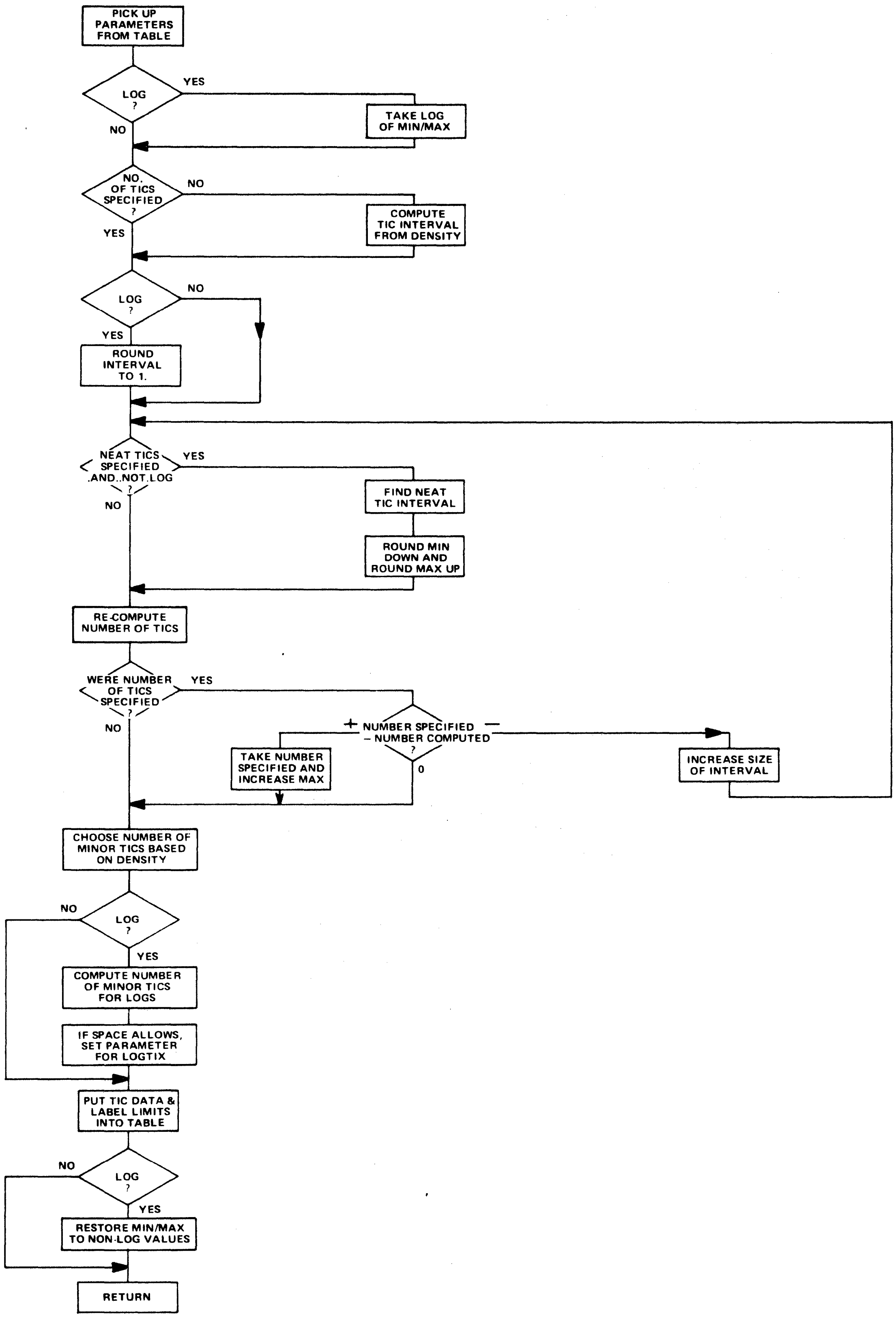


Figure 6

SECTION 9

DISPLAY SUBROUTINES

Once the COMMON Table has been set, the display subroutines draw the plot. CPLOT, the basic plotting routine, calls the other routines necessary to draw the graph. Section 1.2 describes the sequence in which the other subroutines are called. CPLOT, UPOINT, BAR, FILBOX, BSYMS, GRID, LABEL, NUMSET, USESET, JUSTER, and HLABEL are described in the User's Sections.

Flow charts of NUMSET called routines are included in Section 10.

9.1 Curve Plotting Subroutine - CPLOT

CPLOT draws a data curve, using the window location values from COMMON. A flow chart of CPLOT is shown on the following page. The subroutine is described in detail in Section 2.3.4.

CPLOT

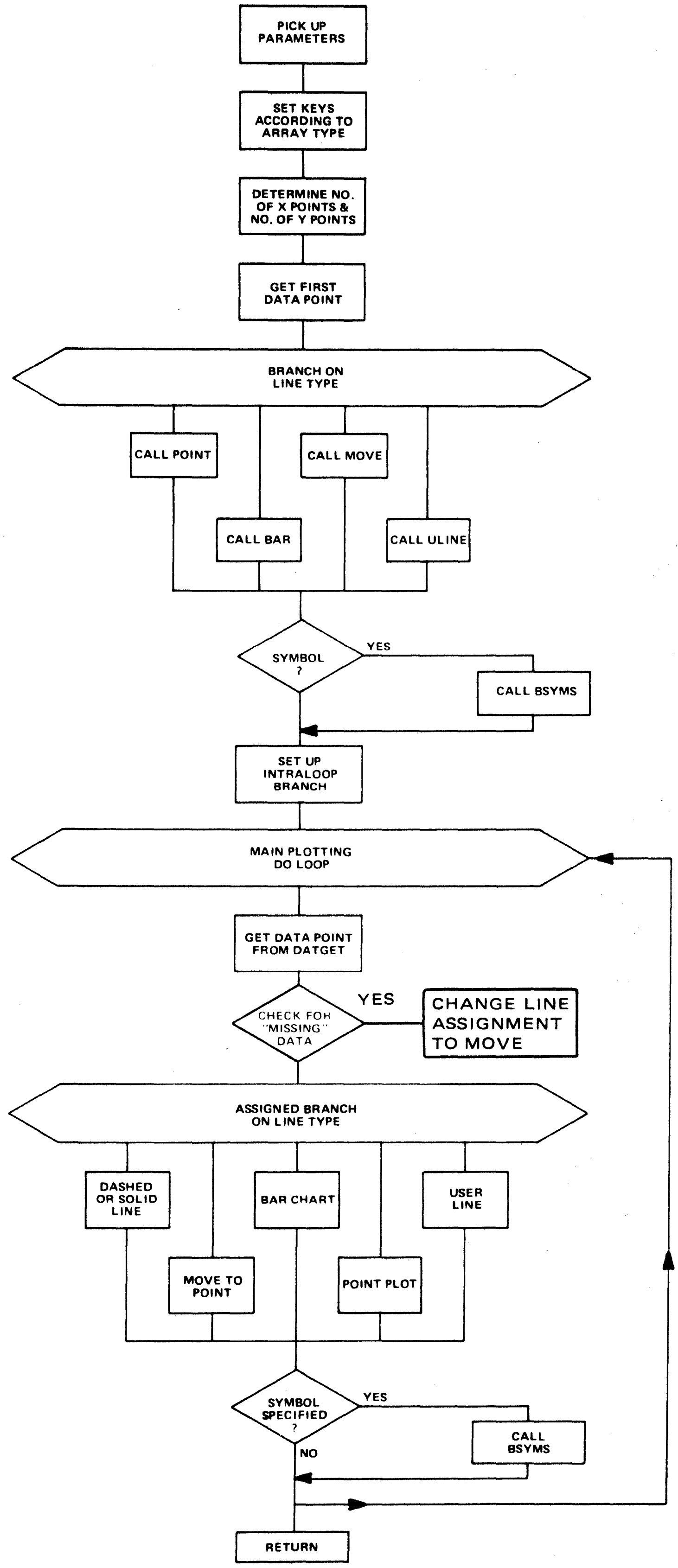


Figure 7

9.2 Label Drawing Subroutine - LABEL

LABEL calls the appropriate routines to construct tick mark labels and compute their positions on the screen, and displays them. The routine is described in Section 4.1.6. A flow chart of LABEL is shown on the following page.

LABEL

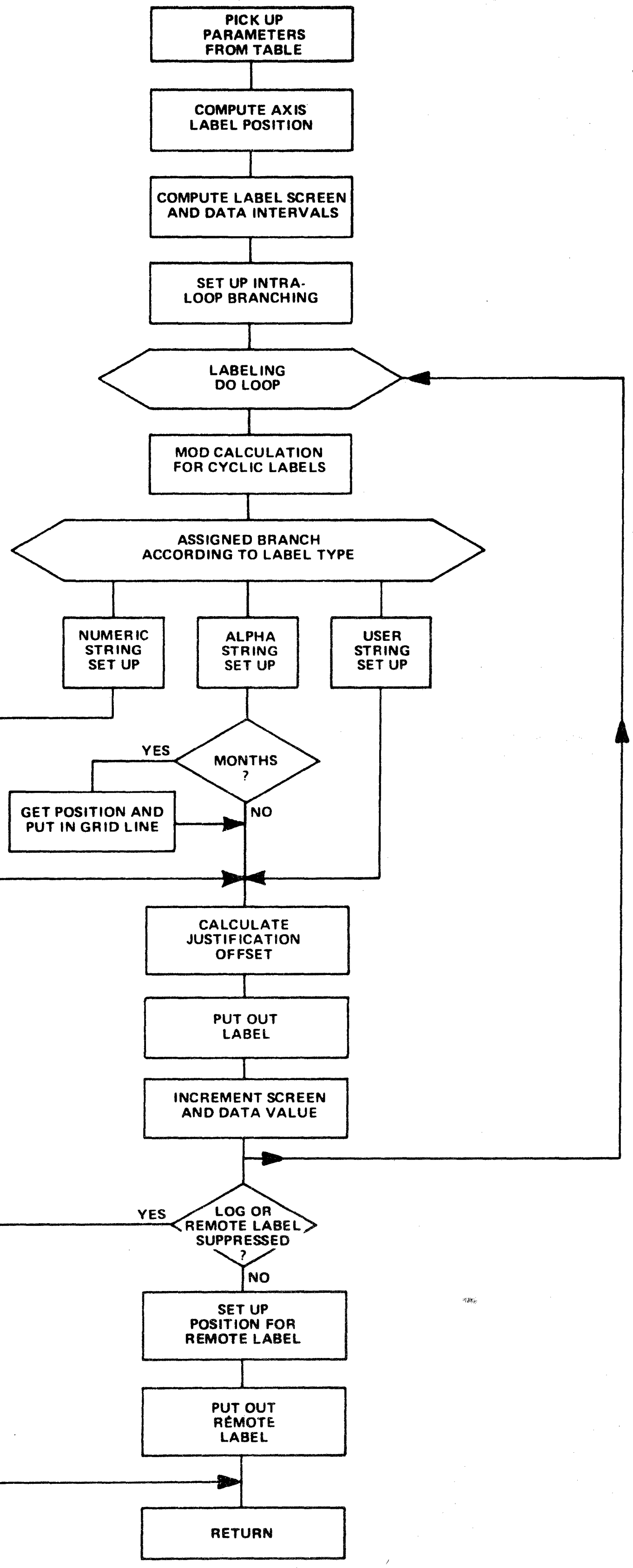


Figure 8

9.3 Key Setting Subroutine for Type of Data Array - KEYSET

Sets the key to indicate the type of data array.

Calling Sequence:

```
CALL KEYSET (ARRAY,KEY)
```

Parameter Entered:

ARRAY	is the data array for which type is to be determined.

Parameter Returned:

KEY	is the key for data array form.

- 1 = standard long form
- 2 = short form
- 3 = short calendar form
- 4 = user defined form
- 5 = non-standard form

9.4 TEK Symbol Drawing Subroutine - TEKSYM

Draws TEK symbols.

Calling Sequence:

```
CALL TEKSYM (ISYM, FACTOR)
```

Parameter Entered:

ISYM	is the value from CSYMBL which specifies the type of symbol to be drawn at data points.
FACTOR	is the floating point scale multiplier for the symbol size.

Description:

This subroutine draws the symbols provided with the AG-II system. See data point symbol chart in Section 3.2.1.

In the code, the following variables retain the first values assigned to them through subsequent executions.

MEMORY
AMULT
ROD
IHALF
IFULL
ITEM
IX
IY arrays dimensioned to 3

This will cause problems on systems where local variables are dynamically stored. Such problems can be solved by forcing these variables to be stored in core memory.

9.5 Data Obtaining Function - DATGET

This function provides the actual data values to be plotted.

Calling Sequence:

VALUE = DATGET (ARR,I,KEY)

Parameters Entered:

ARR	is the data array to be plotted
I	is the data point for which a value is needed.
KEY	is the type of data array being used.

1 is standard data format.

2 is standard short data format.

3 is calendar format

4 is user defined format.

5 is non-standard data format.

Description:

DATGET is called by CPLOT to obtain the data from the data arrays for each point plotted. (Calendar data values are obtained by DATGET calling CALPNT.)

In the code, the variable OLDONE retains the value assigned to it during the previous execution.

9.6 Logarithmic Tic Mark Drawing and Labeling Subroutine - LOGTIX

Draws and labels logarithmic minor tic marks.

Calling Sequence:

```
CALL LOGTIX (NBASE,START,TINTVL,MSTART,MEND)
```

Parameters 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.)
START	is the start of the interval into which logarithmic minor tic marks are to be placed.
TINTVL	is the length of the major tic mark interval.
MSTART	is the minor tic mark starting point.
MEND	is the minor tic mark ending point.

Description:

This subroutine is called by GRID to draw logarithmic tic marks and labels. LOPTIM determines if there is room for logarithmic tic marks or labels. CXMTCS and CYMTCS contain a coded value where -1 specifies logarithmic tic marks and -2 specifies logarithmic tic marks with labels.

9.7 Monthly Tic Mark and Grid Drawing Subroutine - GLINE

Draws monthly tic marks and grid.

Calling Sequence:

CALL GLINE (NBASE,DATAPT,SPOS)

Parameters Entered:

NBASE	is the axis pointer, or the location of the first item referring to the X or Y axis, whichever is here indicated. (IBASEX or IBASEY provides this location.)
DATAPT	is the virtual position of the end of the last day of the month.

Parameters Returned:

SPOS	is the screen position of the last day of the month.

Description:

GLINE determines the monthly tic mark positions in screen coordinates and draws them. The screen position is returned to MONPOS which, in turn, passes it to LABEL for the centering of the labels.

This routine positions grid lines at the end of the month, which means that lines will not be evenly spaced since months are not of equal length.

SECTION 10

STRING HANDLING AND LABELING

Labels are handled through a group of routines which set up character strings to be displayed, a subroutine which left, right, or center justifies the character string, and subroutines which display the character string at the designated place on the screen.

10.1 String Set Up

The following subroutines interact to set up character strings for display. The three primary routines are underlined and followed by subsidiary routines.

NUMSET utilizes IFORM, FFORM or EXPOUT to change floating point numbers into character strings which can later be displayed as tic mark labels. Calling sequence is:

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

as described in full on page 4-19.

IFORM converts a floating point number into a character string without a decimal point. Calling Sequence is:

CALL IFORM (FNUM,IWIDTH,IARRAY,IFILL)

as described in full on page 4-22.

FFORM converts a real value into a character string which provides either a floating point number or a number without a decimal point, depending on which form will best fit the given width. Calling Sequence is:

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

as described in Section 4.2.3.

EFORM converts a real value into an exponential character string to fit a given width. Calling Sequence is:

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

as described in full on page 4-20.

FONLY converts a real value into a character string which provides a number with a decimal point. Calling Sequence is:

CALL FONLY (FNUMBER,IWIDTH,IDECIMAL,IARRAY,IFILL)

As described in detail on page 10-6 of this manual.

FONLY is called by FFORM or EFORM.

ESPLIT determines the exponent required for exponential labels. Calling Sequence is:

CALL ESPLIT (FNUMBER,IWIDTH,IDECIMAL,IEXPON)

as described in detail on page 10-7 of this manual.

ESPLIT is called by EFORM.

EXPOUT constructs the character string for remote scale factors and for logarithmic tic marks. Calling Sequence is:

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

As described in Section 4.2.5.

<u>ALFSET</u> supplies the ASCII values of the characters for writing the days and months for labels. Calling Sequence is:

CALL ALFSET (FNUM,IWIDTH,LABTYP,IARRAY)

as described on page 10-5 of this manual.

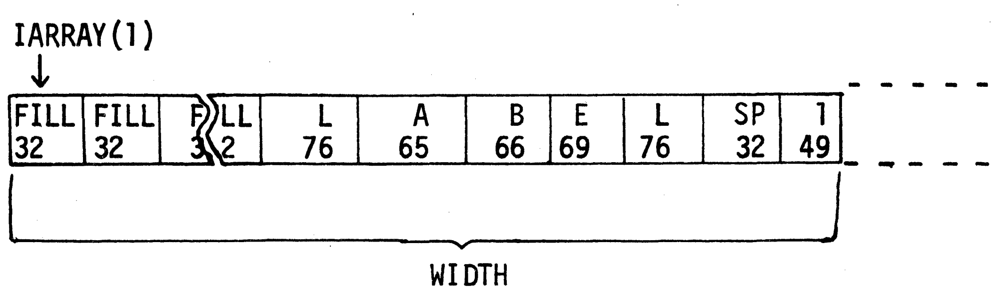
<u>USESET</u> is a user written routine which supplies character strings for labels according to the user's format. Calling Sequence is:

CALL USESET (FNUM,IWIDTH,NBASE,IARRAY)

As described in full in Section 13

Note: This routine must be expanded from its present simple form if it is to be utilized. JUSTER assumes blank fill.

A typical character string produced is shown below.



A Character String

Figure 9

Figure 9 is a flow chart of the interaction of these routines.

In general terms, NUMSET determines the kind of labels needed and calls the appropriate routines. If logarithmic labels are needed, EXPOUT is called to produce exponential labels.

If integer labels are needed, IFORM is called. IFORM forms the string and fills blank spaces at the left with space fill characters (ASCII decimal equivalent 32).

If floating point numbers are needed, NUMSET calls FFORM which determines if the number is too wide for the space available. If it is not too wide, FFORM calls FONLY which separates the fractional part from the whole part. FONLY calls IFORM with the fractional part, inserts a decimal point, calls IFORM with the whole part, and inserts a minus sign if the number is less than zero.

If the floating point number is too wide for the space available, FFORM calls EFORM which calls ESPLIT for the decimal and exponent.

NUMBER CONVERSION

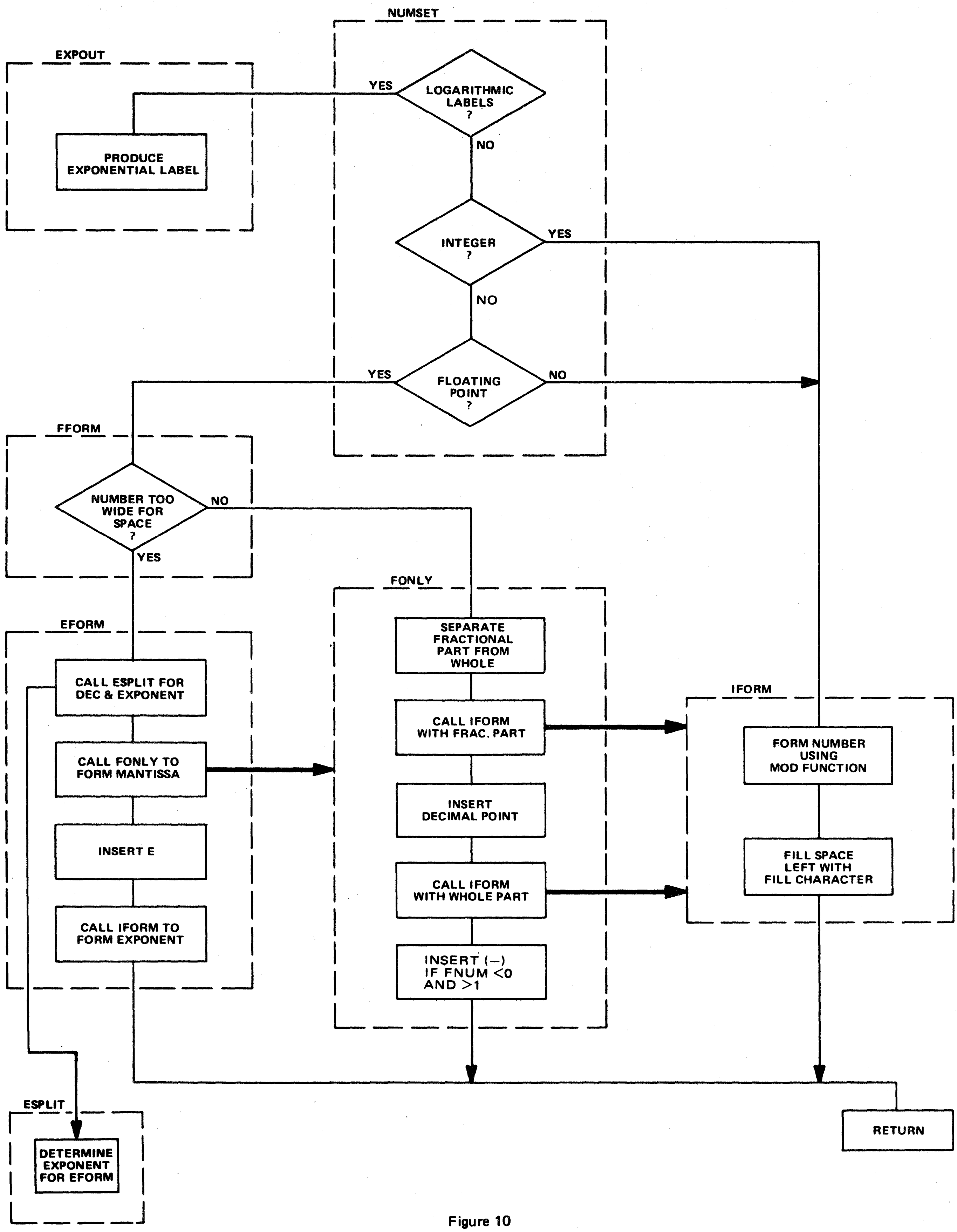


Figure 10

10.1.1 Alphanumeric Labeling Subroutine - ALFSET

Supplies the ASCII values of the characters for writing the days and months for labels.

Calling Sequence:

```
CALL ALFSET (FNUM,IWIDTH,LABTYP,IARRAY)
```

Parameters Entered:

FNUM	is the number of the day or month to be printed.
IWIDTH	is the label width from COMMON.
LABTYP	is the label type from COMMON.

Parameters Returned:

IARRAY	is the string of ASCII values for the label characters.

Description:

ALFSET contains an array of ASCII values for the letters of each day and month. It checks the label type (CXLAB or CYLAB) in COMMON, and if it is 3 (days) or 6 (months) continues to check the label width (CXWDTH). It then supplies as many of the appropriate ASCII values as will fit the width.

The user may increase the array size and add other label types if the need arises.

10.1.2 Floating Point Conversion Subroutine - FONLY

Converts a real value into a character string which provides a number with a decimal point.

Calling Sequence:

```
CALL FONLY (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 used to fill any extra spaces at the beginning of the array (all numbers are right justified).

Parameters Returned:

IARRAY	is the string of characters to compose the label.

Description:

FONLY is called by FFORM and EFORM, as needed, to produce floating point character strings.

10.1.3 Exponent Finding Subroutine - ESPLIT

Determines the exponent required for labels.

Calling Sequence:

```
CALL ESPLIT (FNUMBER,IWIDTH,IDECIMAL,IEXPON)
```

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.

Parameter Returned:

IEXPON	is the exponent value.

Description:

EFORM calls ESPLIT to find the correct exponent value.

10.2 Character String Placement and Printing

After subroutine LABEL determines the position from which tic mark labels or remote labels are to be justified, it calls subroutine JUSTER. JUSTER computes the distance the beginning point of the label is to be offset from the position determined by LABEL. Subroutine REMLAB produces the label string for the remote label or remote scale factor and displays it. NOTATE displays horizontal labels by calling HLABEL. These subroutines are described individually with illustrations on the following pages.

JUSTER right, left, or center justifies a character string. It computes the distance in raster units between the point from which the string is to be justified and the actual starting point for the justified string as well as the length of the character string without fill characters. Calling Sequence is:

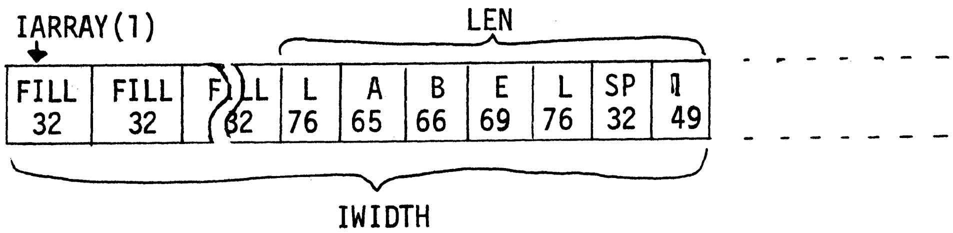
```
CALL JUSTER (IWIDTH,IARRAY,KEY_POSITION,IFILL_CHARACTER,LEN,
             IOFFSET)
```

as described in detail in Section 2.13.4.

The following diagram shows the manner in which a character string is justified.

LEN	represents the length of the actual label to be printed
IWIDTH	the length of the label plus fill characters
KEY_POSITION	the designation of right, left, or center justification
IOFFSET	the distance in raster units between position from LABEL and the actual starting point after justification
IHORZ	the width of an alphanumeric character in raster units (14 raster units for the standard character set on the 4010 family of Display Terminals).
IFILL_CHARACTER	is the character in ASCII code used as filler (usually a space [32]).

Character String:



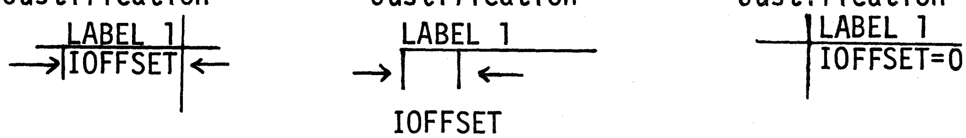
OFFSET Computation

IF KEY_POSITION<0,IOFFSET=0
IF KEY_POSITION>0,IOFFSET=-LEN*IHORZ
IF KEY_POSITION=0,IOFFSET+(-LEN*IHORZ)/2

KEY_POSITION>0 Results in Right Justification

KEY_POSITION=0 Results in Center Justification

KEY_POSITION<0 Results in Left Justification



Label Justification

Figure 11

NOTATE moves the alphanumeric cursor to the position where a label is to begin and prints the label out. It is called by LABEL with the screen coordinates of the beginning point of the label, the length of the character string to be printed, and the string of characters in ASCII decimal equivalents. Calling Sequence is:

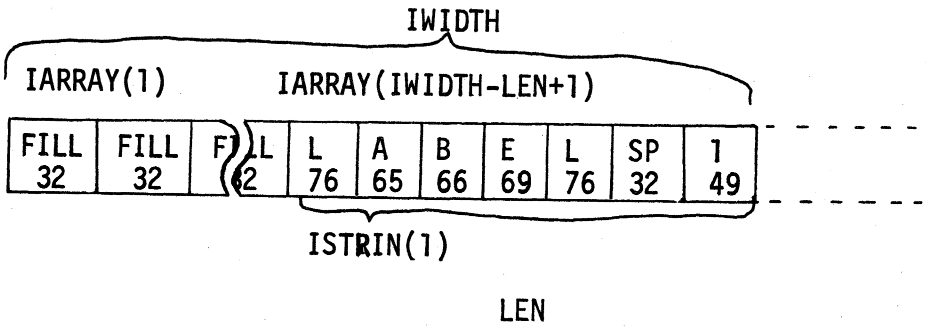
CALL NOTATE (IX,IY,LENCHR,ISTRIN)

as described in detail in Section 2.13.2

As called by LABEL, the computation of the parameters would appear as follows:

CALL NOTATE (IX+IOFF,IY,LENCHR,IARRAY(IWIDTH-LEN+1))

The following figure shows a character string with the parameter designations labeled.



Character String as used with NOTATE

Figure 12

NOTATE then calls the TCS subroutine

MOVABS (IX+IOFF,IY)

which moves the alphanumeric cursor to the beginning point of the label, and the AG-II subroutine.

HLABEL (LEN,ISTRIN)

which produces a label such as:

LABEL 1

HLABEL is a horizontal labeling subroutine described on page 50 of the User's Manual.

REMLAB uses much the same procedure as shown above to print out remote labels or scale factors such as 10^n, M, etc., or the name of a beginning month or year for calendar data. The method of printing superscript characters, such as are often used in remote labels, is described in Section 10.4.

10.3 Remote Labeling Subroutine - REMLAB

Produces the label string for the remote label or remote scale factor (the remote exponent position is calculated in LABEL) and displays it.

Calling Sequence:

```
CALL REMLAB (NBASE,ILOC,LABTYP,IRX,IRY)
```

Parameters 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.)
ILOC	is the location of the axis in relation to the window edge (See Section 3.2.3
LABTYP	is the type of tic label which designates log, linear, days, months, or other. (See Section 3.2.4.)
IRX,IRY	is the location, in raster units, from which the remote exponent is right, left, or center justified.

Description:

REMLAB determines if the remote exponent is left, right, or center justified.

Axes (axis lines plus tic marks and labels) on the lefthand side of the screen have remote exponents that are right justified from a reasonable location. Axes on the righthand side of the screen have labels that are left justified, and X axis remote exponents are centered on the axis.

This routine calls EXPOUT, JUSTER, and NOTATE to compute the value of the exponent and the justification, and displays the exponent.

Calendar data requires a different type of remote label. If the calendar label type is days, the beginning date of the axis is displayed as the remote label. If the calendar label type is other than days, the years spanned will be the remote exponent (1969 if all data is in that year, 1969-1972 if the beginning data was in the year 1969 and the ending data in 1972).

10.4 Labels with Special Characters

Subscript and Superscript characters require special treatment in a character string. A -1 value in a character string indicates that the next character is to be in a superscript position (JUP, or jump up character). A -2 value in a character string indicates that the next character is to be in subscript position (JDN, or jump down character).

For example, to produce a character string for the superscripted label

22^8

the values might be placed in the array as follows:

1. CALL IFORM (FNUM,IWIDTH,IARRAY,IFILL) with the following parameter values.

FNUM equals 8., the value of the superscript.
IWIDTH equals 4, the width of the total array of three numbers plus a superscript designation.
IARRAY is the name of the array or character string being built.
IFILL is the ASCII decimal equivalent of the character to be used as filler in the array (32 is a space).
The resulting character string now contains the following ASCII decimal equivalents:

IARRAY(1)
↓

32	32	-1	56	ASCII Decimal Equivalents
SP	SP		8	Characters Represented

IWIDTH

2. Insert the superscript character designator in IARRAY(3) with the statement

```
IARRAY(3)=JUP
```

where JUP designates a "jump up" or superscript character. The character string now appears as:

IARRAY(3)

32	32	-1	56	ASCII Decimal Equivalents
SP	SP		8	Characters Represented

Available special character designators are:

JUP = -1	denotes that the following characters will be moved up half a character space.
JDN = -2	denotes that the following characters will be moved down half a character space.

3. The remainder of the character string may be placed in the array with another call to IFORM

    CALL IFORM (FNUM,IWIDTH,IARRAY,IFILL)

    with FNUM equal to 22, IWIDTH equal to 2, and the other parameters the same as in step 1.

    The complete array is:

50	50	-1	56	...	ASCII Decimal Equivalents
2	2		8		Characters Represented

$\underbrace{2\quad 2}_{\text{IWIDTH}}$

    Instead of calling IFORM, ASCII decimal equivalents may also be entered directly with statements

```
IARRAY(1)=50
IARRAY(2)=50
```

The computation of left, right, or center justification also requires special consideration by Subroutine JUSTER. In determining the distance from a reference point that a character string is to be offset, the special character designator (JUP or JDN) must be subtracted from the length of the character string.

LEN

50	50	-1	56	...	ASCII Decimal Equivalents
2	2		8		Characters Represented

$\underbrace{2\quad 2\qquad 8}_{\text{IWIDTH}}$

```
ISPECIAL=No of Special Character Designators
IF KEY_POSITION<0,IOFFSET=0
IF KEY_POSITION>0,IOFFSET=-(LEN-ISPECIAL)*IHORZ
IF KEY_POSITION=0,IOFFSET=-(LEN-ISPECIAL)*IHORZ/2
```

Figure 13

NOTATE handles special characters in the following manner. The same character string used in previous examples is also used here.

1. NOTATE first calls MOVABS with the screen coordinates where the label is to begin on the screen.

   MOVABS (IX+IOFF,IY)

2. It computes the length of the string to the first special character designator encountered, and calls HLABEL.

   ANSTR (LENGTH,IARRAY)*

   For our example, LENGTH equals 2

50	50	-1	56	ASCII Decimal Equivalents
2	2		8	Characters Represented

   $\underbrace{2 \quad 2}_{\text{LENGTH}}$

   ANSTR then prints the characters designated

   Characters Printed

   Starting
   Beam Position 22

3. NOTATE checks the type of special character and calls MOVABS once more to move the alphanumeric cursor (or beam) to the end of the string printed thus far and up or down one half character, depending on the type of special character. For our example, MOVE is called in the following manner:

   CALL MOVABS (IX+LENGTH*IHORZ,IY+IVERT/2)

   Where IHORZ is the width of a character in raster units and IVERT is the height of a character in raster units.

   Beam position is now:

   22 □
   Beam Position

*User with AG-II, Release 1.1 or earlier should use the routine HLABEL in place of ANSTR.

4. NOTATE then calls ANSTR with the length and array position of the special character to be printed. For our example,

    CALL HLABEL (LENGTH,IARRAY(2+1+1))

    where LENGTH is the number of words from the array position passed to ANSTR to the end of the string or to the next special character designator (-1 or -2), whichever is encountered first. IARRAY (2+1+1) is the array position of the next character to be printed.

   Note that the beam will remain in the same vertical plane until another special character designator is encountered. Therefore, the character string to print H_2O would be:

72	-2	50	-1	79
H		2		O

   The -2 special character designator (JDN) moves the beam down half a character to print the 2, and the -1 special character designator moves the beam up half a character to the normal position for the O.

SECTION 11

NUMERIC FUNCTION

A group of functions provide various numeric operations. FINDGE and FINDLE find the greatest or the least value in a table of integers. ROUNDD and ROUNDU round values down or up whenever necessary. LOCGE and LOCLE are comparable to FINDGE and FINDLE, but operate on floating point values. IOTHER determines the opposite base (transfers computation from the X axis table of COMMON to the Y axis table or vice versa).

11.1 Maximum Floating Point Value Finding Function - FINDGE

Finds in a table of floating point numbers the smallest entry greater than or equal to a given test value.

Calling Sequence:

RESULT = FINDGE (VALUE,TABLE,IPOINT)

Parameters Entered:

VALUE — is the test value against which TABLE entries are to be checked.

TABLE — is a sequential array of real values to be compared to the test value. The first entry must be smaller than VALUE, and the last entry must be greater than VALUE. (TABLE(1)<VALUE<TABLE(N).) Beginning and ending values such as -1E30 and 1E30 fulfill this requirement.

IPOINT — is a pointer to a word or value in TABLE. Its initial value will be the point in the table at which the search starts, and the ending value will be position of the first number found which is greater than or equal to VALUE.

<u>Restriction</u>: IPOINT must fall within the range of the table (0<IPOINT<N).

Description:

FINDGE checks the initial value of TABLE(IPOINT) to determine if it is greater than or less than VALUE, and thus determines which direction the pointer is to move. For example, if TABLE(IPOINT) is less than VALUE, the pointer will move toward the end of the table; if greater than VALUE, it will move toward the beginning of the table.

```
Example:  Given   TABLE=1E30,1.,3.,3.,4.,1E30
                  IPOINT=2
                  X=FINDGE (3.5,TABLE,IPOINT)
           Then   X=4 and
                  IPOINT=5
```

NOTE: If this routine is used in a loop, execution time will be minimized if IPOINT is not set to a constant inside the loop. This is especially true of sets of VALUE which are ordered sequentially.

11.2 Minimum Floating Point Value Finding Function - FINDLE

Finds in a table of floating point values the largest entry that is less than or equal to a given test value.

Calling Sequence:

```
RESULT = FINDLE (VALUE,TABLE,IPOINT)
```

Parameters Entered:

VALUE	is the test value against which the TABLE entries are to be checked.
TABLE	is a sequential array of real values to be compared to the test value. The first entry must be smaller than VALUE, and the last entry must be greater than VALUE, the same as for FINDGE. (See page 11-2)
IPOINT	is a pointer to a word (or value) in TABLE. Its initial value will be the point in the table at which the search starts and the ending value will be the position of the first number found which is less than or equal to VALUE.

Restriction: IPOINT must fall within the range of the table (0<IPOINT<N).

Description:

FINDLE checks the initial value of TABLE(IPOINT) to determine if it is greater than or less than VALUE, and thus determines which direction the pointer is to move. If TABLE(IPOINT) is less than VALUE, the pointer will move toward the end of the table; if greater, it will move toward the beginning of the table.

Example: Given TABLE = -1E30,1.,3.,3.,4.,1E30

IPOINT=2

X=FINDLE (3.5,TABLE,IPOINT)

Then X=3 and

IPOINT=4

Note: If this routine is used in a loop, execution time will be minimized if IPOINT is not set to a constant inside the loop. This is especially true of sets of VALUE which are ordered sequentially.

11.3 Maximum Integer Value Finding Function - LOCGE

Finds in a table of integer values the smallest entry greater than or equal to a given test value. Basically, this function is the same as FINDGE, but FINDGE uses floating point values.

Calling Sequence:

```
RESULT = LOCGE (IVALUE,ITABLE,IPOINT)
```

Parameters Entered:

IVALUE — is the test value against which table values are to be checked.

ITABLE — is a sequential array of integer values to be compared to the test value. The first entry must be smaller than IVALUE, and the last entry must be larger than IVALUE. (ITABLE(1)<IVALUE<ITABLE(N).) Beginning and ending values such as -999999 and 999999 fulfill this requirement.

IPOINT — is a pointer to a word or value in ITABLE. Its initial value is the point in the table at which the search starts, and the ending value will be the position of the first value found which is greater than or equal to IVALUE.

Restriction: IPOINT must fall within the range of the table (0<IPOINT<N).

Description:

As with FINDGE, the initial value of IPOINT is checked for being greater or less than IVALUE. The pointer then moves in the appropriate direction to find the value it is seeking.

```
Example:  Given  ITABLE=-999999,10,20,30,40,999999

                 IPOINT=2

                 I=LOCGE (35,ITABLE;IPOINT)

          Then   I=40 and

                 IPOINT=5
```

NOTE: If this routine is used in a loop, execution time will be minimized if IPOINT is not set to a constant inside the loop. This is especially true of sets of VALUE which are ordered sequentially.

11.4 Minimum Integer Value Finding Function - LOCLE

Finds in a table of integer values the largest entry that is less than or equal to a given test value.

Calling Sequence:

```
RESULT = LOCLE (IVALUE,ITABLE,IPOINT)
```

Parameters Entered:

IVALUE	is the test value against which table values are to be checked.
ITABLE	is a sequential array of integer values to be compared to the test value. The first entry must be smaller than IVALUE, and the last entry must be larger than IVALUE. (ITABLE(1)<IVALUE<ITABLE(N).) Beginning and ending values such as -999999 and 999999 fulfill this requirement.
IPOINT	is a pointer to a word or value in ITABLE. Its initial value will be the point in the table at which the search starts, and the ending value will be the position of the value it was seeking. <u>Restriction</u>: IPOINT must fall within the range of the table (0<IPOINT<N).

Description:

LOCLE checks the initial value of IPOINT to determine if it is greater or less than IVALUE, and thus determines which direction the pointer is to move. For example, if IPOINT is less than IVALUE, the pointer will move toward the end of the table; if greater, it will move toward the beginning of the table.

Example:	Given	ITABLE=-999999,10,20,30,40,999999
		IPOINT=2
		I=LOCLE (35,ITABLE,IPOINT)
	Then	I=30 and
		IPOINT=4

NOTE: If this routine is used in a loop, execution time will be minimized if IPOINT is not set to a constant inside the loop. This is especially true of sets of VALUE which are ordered sequentially.

11.5 Round Down Function - ROUNDD

Rounds values down any time required.

Calling Sequence:

```
RESULT = ROUNDD (VALUE,FINTERVAL)
```

Parameters Entered:

VALUE	is the number or value to be rounded.
FINTERVAL	defines the interval or position the number is to be rounded to. (Interval is 100 if the value is to be rounded to the next lower hundred).

Description:

Function ROUNDD rounds a value to the greatest multiple of the interval less than the value.

Example:

1. If VALUE is between 201 and 299, and FINTERVAL is 100, the resulting value will be 200.
2. If the VALUE is -130 and the FINTERVAL is 100, the resulting value will be -200.

NOTE: When VALUE is a multiple of FINTERVAL, results are are unpredictable due to floating point roundoff errors. If an extra tic interval occurs at the end of a graph, the rounding constant is too large and should be modified by reducing the number of nines in the constant. In dealing with high precision numbers the tic mark labeling may be irregular; increasing the number of nines in the rounding constant may solve this irregularity.

11.6 Round Up Function - ROUNDU

Rounds values up whenever required.

Calling Sequence:

RESULT = ROUNDU (VALUE,FINTERVAL)

Parameters Entered:

VALUE	is the number or value to be rounded.
FINTERVAL	defines the interval or position which the number is to be rounded to. (Interval is 100 if the value is to be rounded to the next higher hundred.)

Description:

Function ROUNDU rounds the value to the least multiple of the interval greater than the value.

Example:

1. If the VALUE is between 201 and 299, and the FINTERVAL is 100, the resulting value will be 300.

2. If the VALUE is -130, the resulting value will be -100.

NOTE: Due to floating point roundoff errors, results are unpredictable when VALUE is a multiple of FINTERVAL. If an extra tic interval occurs at the end of a graph, the rounding constant is too large and should be modified by reducing the number of nines in the constant. In dealing with high precision numbers the tic mark labeling may be irregular; increasing the number of nines in the rounding constant may solve this irregularity.

11.7 Base Finding Function - IOTHER

Determines the opposite base.

Calling Sequence:

IOTHER (NBASE)

Parameters 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:

Most subroutines are written to function for either the X or the Y axis. IOTHER provides information about the other axis. For example, when working on the Y grid lines, it is necessary to discover the limits of the X axis.

SECTION 12

CALENDAR SUBROUTINES

The Universal Business Graphing Calendar (UBGC) is designed to allow for irregularities in the calendar by providing a continuous numbering of days from January 1, 1901. The basic routines which provide conversion of calendar dates into UBGC days or vice versa, and year-day format to and from year-month-day are described in detail in section 4.3.

The general sequence used in preparing calendar data for plotting is as follows. CMNMX computes the data minimum and maximum in UBGC values and stores them in COMMON. COPTIM obtains these values, uses CALCON to convert the values from UBGC values to calendar values and rounds them to "neat" calendar values. It then stores the label minimum and maximum, converts the new data minimum and maximum into UBGC and replaces the old data minimum and maximum in COMMON with the newly rounded values.

CALPNT and CALCON require history to be kept. That is, they decode the calendar array and holds the information through future executions in variables ISTYR, IWEEK1, ISTPER, NODAYS, and ICLTYP for CALPNT and in variables FNODAY and IWEEK1 for CALCON.

CALENDAR CONVERSION SCHEME

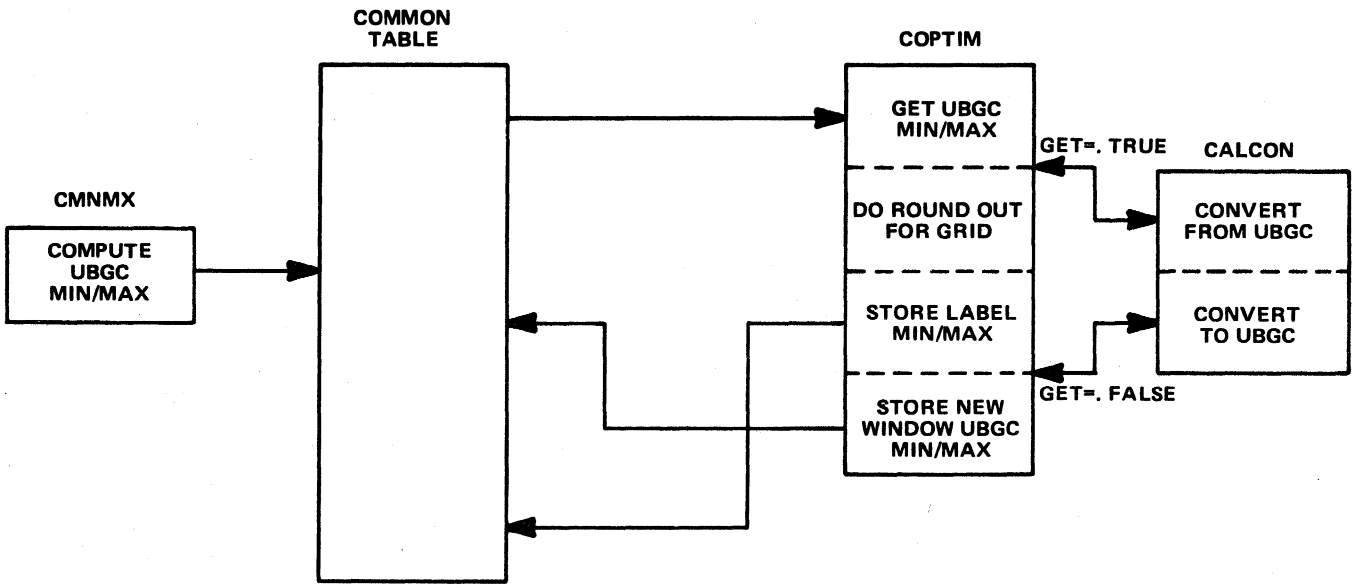


Figure 14

12.1 Calendar Data Obtaining Function - CALPNT

This function provides the actual calendar data values to be plotted.

Calling Sequence:

```
VALUE = CALPNT (ARRAY,I)
```

Parameters Entered:

ARRAY	is the name of the calendar array.
I	is the data point for which a value is needed.

Description:

This function, used by DATGET, is the data array in calendar form. It checks the number of periods per year and makes the necessary calls to provide the correct UBGC (Universal Business Graphing Calendar) value for that point.

Variables ISTYR, IWEEK1, ISTPER, NODAYS, and ICLTYP contain decoded calendar array information and remain unchanged through subsequent executions.

CALENDAR CONVERSION
CALPNT

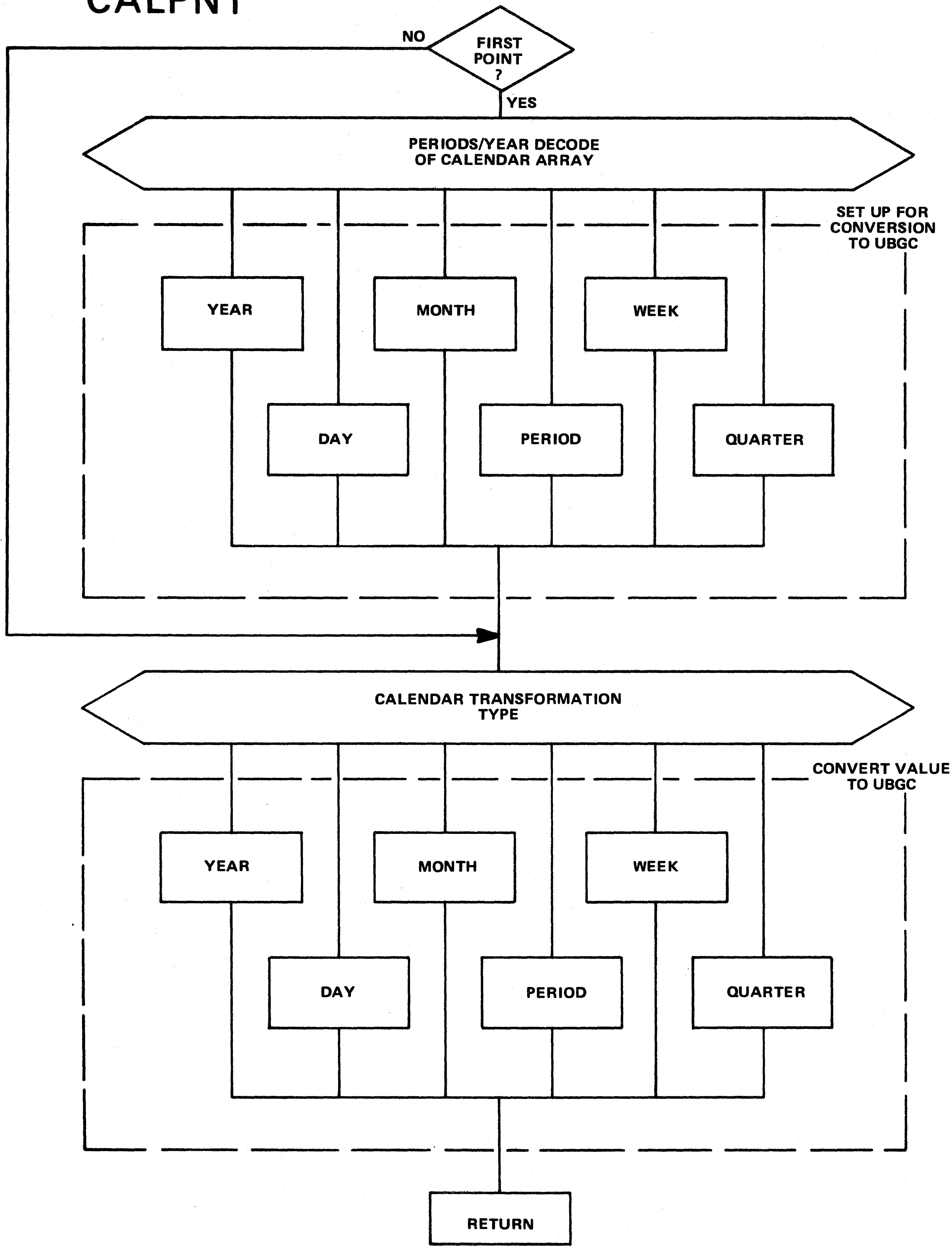


Figure 15

12.2 Calendar Conversion Subroutine - CALCON

Converts calendar label minimums and maximums from virtual day space to the space in which the labels are to be written, or vice versa.

Calling Sequence:

```
CALL CALCON (AMIN,AMAX,LABTYP,GET)
```

Parameters Entered:

AMIN	is the calendar data label minimum in either UBGC day space or space in which to be labeled (years, periods, months, etc.)
AMAX	is the calendar data label maximum in either virtual day space or space in which to be labeled (years, periods, months, etc.)
LABTYP	is the label type (year, day, month, etc.).
GET	is a logical flag where:

TRUE	requires conversion from UBGC day space to label space.
FALSE	requires conversion from label space to UBGC day space.

Description:

CALCON is called by COPTIM to convert the data min and max in UBGC to data min and max in label units. COPTIM then uses the neat algorithm, rounding the min down and the max up if necessary, thus defining the label min/max which are stored in the table. CALCON is then called a second time to convert the label space min/max to UBGC. These values are now the limits of the virtual space window.

Variables FNODAY and IWEEK1 contain decoded calendar array information and remain unchanged through subsequent executions.

This routine must be called with GET = TRUE the first time.

CALENDAR CONVERSION
CALCON

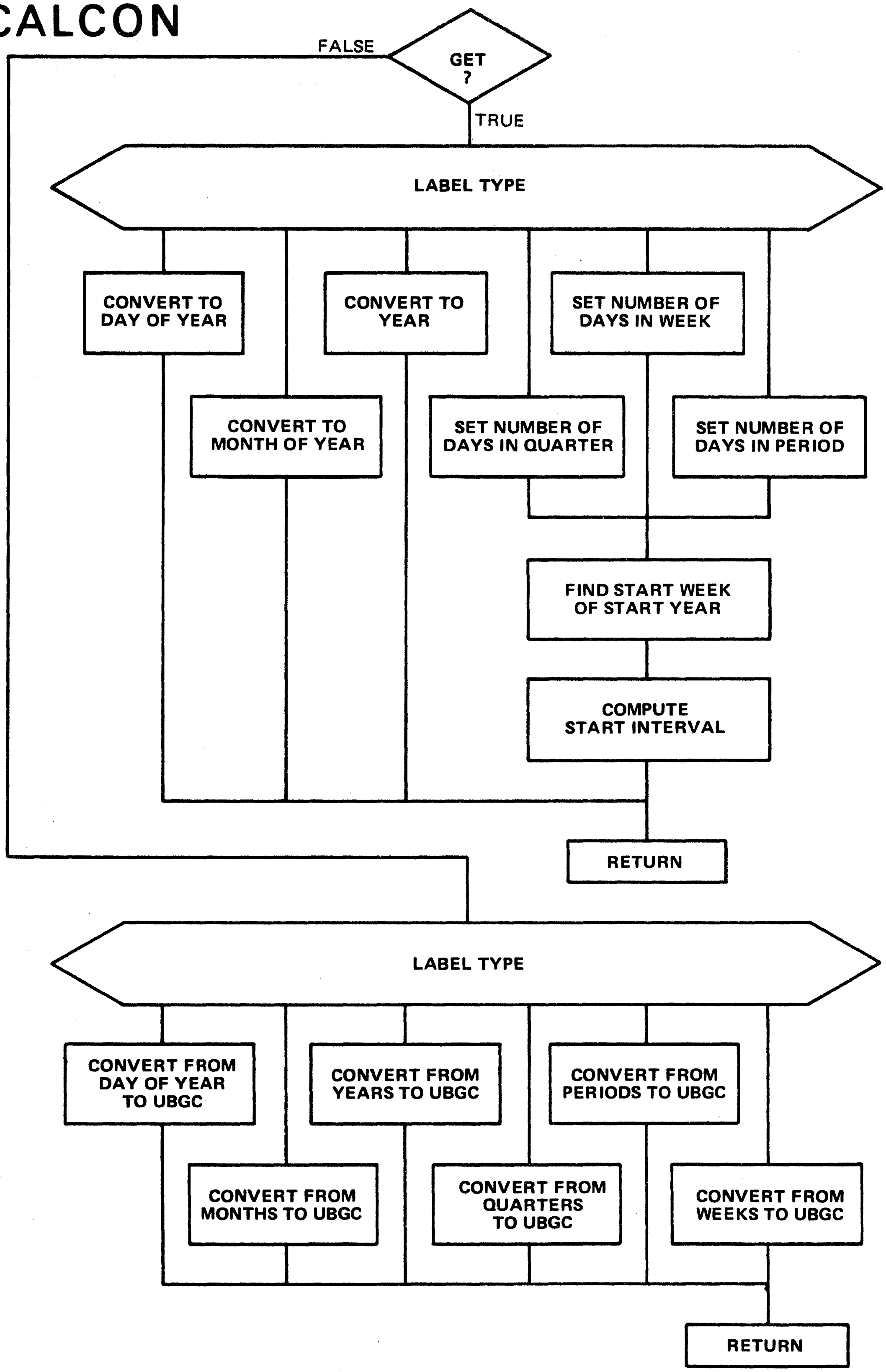


Figure 16

12.3 Month Positioning Subroutine - MONPOS

Determines the positions of monthly tic marks for calendar data. (The lengths of months vary, so the distances between monthly tic marks also vary.)

Calling Sequence:

```
CALL MONPOS (NBASE,IY1,DPOS,SPOS)
```

Parameters 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.)
IY1	the year in which the grid begins.
DPOS	The month in virtual space relative to the beginning year.

Parameters Returned:

SPOS	screen position of the end of the month. (Tic Mark position.)

Description:

This subroutine uses the Universal Business Graphing Calendar (UBGC) to determine the precise length of each month plotted and the position for each monthly tic mark. MONPOS then calls GLINE which plots the tic marks and grid lines. Monthly tic marks are the only ones not drawn with subroutine GRID.

12.4 Calendar Data Minimum and Maximum Subroutine - CMNMX

Determines the minimum and maximum of calendar data in UBGC days.

Calling Sequence:

```
CALL CMNMX (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 in UBGC values.
AMAX	is the maximum array value in UBGC values.

Description:

CMNMX finds the data minimum and maximum for calendar arrays.

CALENDAR CONVERSION
CMNMX

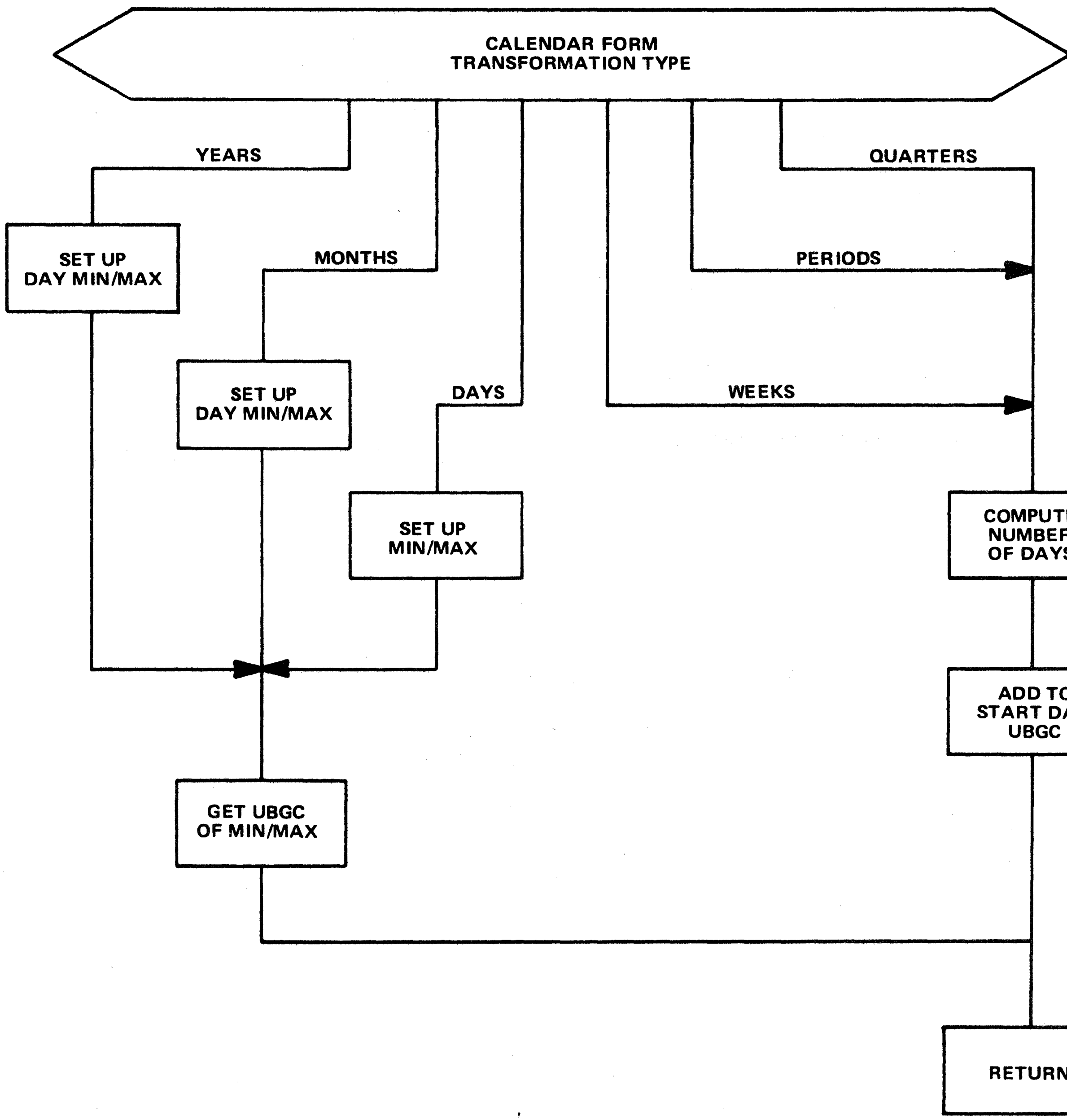


Figure 17

12.5 Leap Year Update Subroutine - LEAP

Updates the leap year variable.

Calling Sequence:

```
CALL LEAP (YEAR)
```

Parameters Entered:

YEAR	is the year to be used in updating the UBGC leap year variable.

Description:

When passed a year in the format YYYY, this routine determines if the year is the same as for the last date processed. If the year is different, the routine determines if it is a leap year and updates the variable in BPP1 COMMON.

SECTION 13

PRUNING (REMOVAL OF UNNEEDED FEATURES)

The procedure for removing unneeded portions of the system to save storage is referred to as "pruning". Pruning has been divided into two levels. At the first level (A level), the package size is reduced without code modification. The second level (B level), provides additional reduction but requires code modification.

13.1 A-Level Pruning

This form of package size reduction involves no code modification. The size is reduced by excluding some subroutines entirely and replacing others with dummy routines. A dummy routine must have the same name and arguments as the subroutine it replaces, and contain RETURN and END lines. A function must also contain an ASSIGNMENT statement, such as

CALPNT = Ø.

for the function CALPNT.

The following list contains the names of the subroutines which may be eliminated. Those which require a dummy replacement are followed by an asterisk. Refer to the system flow chart for the number and type of arguments.

The number of words saved indicated here is based on compilation of the DEC PDP-10. The number of words saved may vary widely depending on the type of computer used.

Features Eliminated	Words Saved	Subroutines Eliminated
Bar Graphs	Approximately 517	BAR * FILBOX
Error Recovery	70	ERREC *
Software Plotting Symbols (TEKSYM)	307	TEKSYM *
Logarithmic Plots	121	LOGTIX *
Calendar Axis	1,764	COPTIM * CALCON CALPNT * MONPOS * GLINE YDYMD YMDYD * LEAP OUBGC * LOCGE IUBGC

* These subroutines require dummy replacements.

13.2 B-Level Pruning (Elimination of Unneeded Code Blocks)

Under each heading, edit the named routines as shown. Code with a solid bar ▌ to the left of column one is to be deleted; code with a shaded bar ▒ to the left of column one is to be modified as shown in the second copy of the code.

Reduction - savings in core size can only be estimated. Savings shown are in addition to savings from A-level pruning mentioned above.

If a feature is to be eliminated with B-level pruning, comparable changes need not be made at A-level.

Each change is described independently. If the same line is changed for several prunings, the changes should be made accumulatively. For example, to eliminate calendar axis, the line

```
GO TO (100, 200, 300, 400, 500), KEY
```

is changed to

```
GO TO (100, 200, 100, 400, 500), KEY
```

To eliminate user defined plots, the same line is changed to

```
GO TO (100, 200, 300, 100, 500), KEY
```

If both calendar axes and user defined points are to be pruned, the resulting line should reflect both changes

```
GO TO (100, 200, 100, 100, 500), KEY
```

CAUTION: Be selective in the elimination of code. Eliminate only that code which is truly unnecessary to your operation.

Feature Eliminated: Bar Charts

Saves : 1 dummy routine (BAR) and
1 line of FORTRAN code

Modified Subroutine: CPLOT

Note: If these changes are made, the A-level changes for bar charts are unnecessary.

Modification to CPLOT

Delete Indicated Lines:

```
      XPOINT=DATGET(X,1,KEYX)
      YPOINT=DATGET(Y,1,KEYY)
C * MOVE TO FIRST DATA LOCATION
      CALL MOVEA(XPOINT,YPOINT)
      IF(LINE .LT.-10)CALL ULINE(XPOINT,YPOINT,1)
█     IF(LINE .EQ.-2 .OR. LINE .EQ.-3)CALL BAR(XPOINT,YPOINT,LINE)
      IF(SYMBOL) CALL BSYMS(XPOINT,YPOINT,ISYM)
C * THE FOLLOWING CODE PREPARES BRANCHES FOR THE
C * TYPE OF LINE TO BE USED IN THIS PLOT
C * IF LINE GREATER THAN 4  DASHED IS ASSUMED
```

Feature Eliminated: Error Recovery

Saves : 1 dummy routine (ERREC) and 6 lines of FORTRAN code

Modified Subroutine: BINITT

Note: If these changes are made, the A-level changes for error recovery are unnecessary.

Modification to BINITT

Delete indicated lines:

```
C * NBASE+26    C(X/Y)AMIN  CALCULATED DATA MINIMUM
C * NBASE+27    C(X/Y)AMAX  CALCULATED DATA MAXIMUM
C * ---IF NBASE = X OR Y PARM VECTOR POINTER
      CALL CSIZE(IH,IV)
      IF(IH .EQ. 0) CALL INITT(120)
      IF(IH .EQ. 0)CALL ERREC(1)
      XTENTC=9
      XTENTX=28
      XTENTY=XTENTX
      POINTR=7
```

Modification to CHECK

Delete indicated lines:

```
      REAL X(2),Y(2)
      COMMON /BPPCOM/ COMGET(80)
C * FOLLOWING CODE INSURES THAT BINITT HAS BEEN CALLED
      IF(COMGET(1) .GE. 50.) GO TO 100
      CALL BINITT
      CALL ERREC(2)
100   NBASE=IBASEX(0)
      DO 300 I=1,2
C * FOLLOWING CODE CALLS TO SET CALENDAR TYPE
      IF(I .EQ. 1)CALL TYPCK(NBASE,X)
```

Feature Eliminated: Calendar Axes

Saves : 5 dummy routines (COPTIM, CALPNT, MONPOS, YMDYD, OUBGC) and 47 lines of FORTRAN code

Modified Subroutines:

1. FUNCTION DATGET
2. LABEL
3. CHECK
4. TYPCK
5. MNMX
6. REMLAB

Note: If these changes are made, the A-level changes for calendar axes are unnecessary.

Modification to FUNCTION DATGET

Change from:

```
C*                                                BEAVERTON,OREGON
C*                    ROUTINE: DATGET
C*****************************************************************
      FUNCTION DATGET(ARR,I,KEY)
      REAL ARR(5)
      GO TO (100,200,300,400,500),KEY
C * STANDARD DATA FORMAT
100   D=ARR(I+1)
      GO TO 600
C * STANDARD SHORT FORM DATA FORMAT
```

To:

```
C*                                                BEAVERTON,OREGON
C*                    ROUTINE: DATGET
C*****************************************************************
      FUNCTION DATGET(ARR,I,KEY)
      REAL ARR(5)
      GO TO (100,200,100,400,500),KEY
C * STANDARD DATA FORMAT
100   D=ARR(I+1)
      GO TO 600
C * STANDARD SHORT FORM DATA FORMAT
```

Delete indicated lines:

```
      GO TO 600
C * STANDARD SHORT FORM DATA FORMAT
200   D=ARR(3)+ARR(4)*FLOAT(I-1)
      GO TO 600
C * SHORT FORM CALENDAR FORMAT
300   D=CALPNT(ARR,I)
      GO TO 600
C * USER FORMAT
400   D=UPOINT(ARR,I,OLDONE)
      GO TO 600
```

Feature Eliminated: Calendar Axes (continued)

Modification to LABEL

Delete indicated lines:

```
C * SET UP INTRA LOOP BRANCH ACCORDING TO LABEL TYPE
      ASSIGN 180 TO LABLE
C * MONTH LABEL=6,DAY LABELS = 3
      IF(LABTYP .EQ. 6 .OR. LABTYP .EQ. 3)ASSIGN 190 TO LABLE
C * NEGATIVE LABEL TYPES FOR USER WRITTEN LABEL ROUTINE
      IF(LABTYP .LT. 0)ASSIGN 170 TO LABLE
      DPOS=DMIN
      SPOS=SMIN
C * IF CALENDAR DATA, FIRST TIC MARK IS NOT LABELED
      IF(.NOT.CAL) GO TO 150
      IYOFF=IYOFF+IYOFF
      CALL OUBGC(IY1,ID,IFIX(COMGET(NBASE+11)))
      DPOS=DPOS+DINT
      SPOS=SPOS+SINTV
      ILIM=ILIM-1
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE
C * USER WRITTEN LABELING OPTION
```

Change From:

```
      ILIM=ILIM-1
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      IF(LABTYP .GT. 0)FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE,(170,180,190)
C * USER WRITTEN LABELING OPTION
170   CALL USESET(DPOS,IWIDTH,NBASE,LABELI)
      GO TO 195
```

To:

```
      ILIM=ILIM-1
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      IF(LABTYP .GT. 0)FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE,(170,180)
C * USER WRITTEN LABELING OPTION
170   CALL USESET(DPOS,IWIDTH,NBASE,LABELI)
      GO TO 195
```

Feature Eliminated: Calendar Axes (continued)

Modification to LABEL

Delete lines indicated:

```
      GO TO LABLE,(170,180,190)
C * USER WRITTEN LABELING OPTION
170   CALL USESET(DPOS,IWIDTH,NBASE,LABELI)
      GO TO 195
C * GENERATE NUMERIC LABEL STRING
180   CALL NUMSET(FNUM*FAC,IWIDTH,NBASE,LABELI,IBLANK)
      GO TO 195
C * GENERATE MONTH OR DAY ALPHA STRING
190   CALL ALFSET(FNUM,IWIDTH,LABTYP,LABELI)
      IF(LABTYP .EQ. 6)CALL MONPOS(NBASE,IY1,DPOS,SPOS)
C * COMPUTE JUSTIFICATION OFFSET
195   CALL JUSTER(IWIDTH,LABELI,IPOSIT,IBLANK,LEN,IOFF)
      IBEGIN = IWIDTH-LEN
      IF (YAXIS) GO TO 200
      ITEM=LEVEL1
C * X AXIS LABELS MAY BE STAGGERED
      IF(STAGER .AND. EVEN) ITEM=LEVEL2
```

Modification to CHECK

Change from:

```
      IF(I .EQ. 1)CALL TYPCK(NBASE,X)
      IF(I .EQ. 2)CALL TYPCK(NBASE,Y)
C * THIS SECTION SETS THE RANGE (MINIMUM AND MAXIMUM) FOR X & Y
      IF(I .EQ. 1)CALL RGCHEK(NBASE,X)
      IF(I .EQ. 2)CALL RGCHEK(NBASE,Y)
      CALL OPTIM(NBASE)
      CALL WIDTH(NBASE)
      STAG=COMGET(NBASE+20)
      IF(STAG .NE. 1.) GO TO 200
      CALL SPREAD(NBASE)
```

To:

```
      IF(I .EQ. 1)CALL TYPCK(NBASE,X)
      IF(I .EQ. 2)CALL TYPCK(NBASE,Y)
C * THIS SECTION SETS THE RANGE (MINIMUM AND MAXIMUM) FOR X & Y
      IF(I .EQ. 1)CALL RGCHEK(NBASE,X)
      IF(I .EQ. 2)CALL RGCHEK(NBASE,Y)
      CALL LOPTIM(NBASE)
      CALL WIDTH(NBASE)
      STAG=COMGET(NBASE+20)
      IF(STAG .NE. 1.) GO TO 200
      CALL SPREAD(NBASE)
```

Delete indicated lines:

```
      CALL BINITT
      CALL ERREC(2)
100   NBASE=IBASEX(0)
      DO 300 I=1,2
C * FOLLOWING CODE CALLS TO SET CALENDAR TYPE
      IF(I .EQ. 1)CALL TYPCK(NBASE,X)
      IF(I .EQ. 2)CALL TYPCK(NBASE,Y)
C * THIS SECTION SETS THE RANGE (MINIMUM AND MAXIMUM) FOR X & Y
      IF(I .EQ. 1)CALL RGCHEK(NBASE,X)
      IF(I .EQ. 2)CALL RGCHEK(NBASE,Y)
```

Modification to TYPCK

Delete the entire subroutine

Feature Eliminated: Calendar Axes (continued)

Modification to MNMX

Delete indicated lines:

```
      IF(NONSTD)NLIM=NPTS
      IF(NONSTD)NSTART=1
      IF(NONSTD .OR. ARRAY(1) .GE. 0.) GO TO 300
      ISET=ABS(ARRAY(1))
      IF(ISET.EQ.1) GO TO 100
      IF(ISET.EQ.2) GO TO 200
C * USER DEFINED MIN/MAX ROUTINE
      CALL UMNMX(ARRAY,AMIN,AMAX)
      GO TO 600
C * SHORT FORM LINEAR DATA ARRAY




100   XMAX=ARRAY(3)+(ARRAY(2)-1.)*ARRAY(4)
      AMIN=AMIN1(ARRAY(3),XMAX,AMIN)
      AMAX=AMAX1(ARRAY(3),XMAX,AMAX)
      GO TO 600
C * CALENDAR SHORT FORM ARRAY
200   CALL CMNMX(ARRAY,AMIN,AMAX)
      GO TO 600
C * STANDARD AND NON-STANDARD LONG FORM ARRAY
300   DO 400 I=NSTART,NLIM
C * IF DATA VALUE EQ TO MACHINE INFINITY-VALUE CONSIDERED MISSING DATA
```

Feature Eliminated: Calendar Axes (continued)

Modification to REMLAB

Delete indicated lines

```
      XAXIS=NBASE .EQ. IBASEX(0)
C * SET JUSTIFICATION ACCORDING TO AXIS AND SIDE
      IPOSIT=0-ISIGN(1,ILOC-1)
      IF(XAXIS)IPOSIT=ICENTR
C * IF NON-CALENDAR - PUT OUT A SCALE FACTOR
      LABTYP=IABS(LABTYP)
      IF(LABTYP .EQ. 1) GO TO 400
C * REMOTE YEAR LABEL FORMED HERE
      ISTART=36
      CALL OUBGC(IY1,ID1,IFIX(COMGET(NBASE+11)))
      CALL OUBGC(IY2,ID2,IFIX(COMGET(NBASE+12)))
      IF(ID2 .LE. 1)IY2=IY2-1
      ID2=ID2-1
C * IF  GRID SPANS A YEAR BOUNDARY
C     BOTH START AND END YEAR ARE WRITTEN
      IF(IY1 .GE. IY2) GO TO 300
C * SET UP TO PRODUCE ENDING YEAR PART OF STRING
      IY=IY2
      ID=ID2
      GO TO 325
C * SET UP TO PRODUCE START YEAR PART OF STRING
300   IY=IY1
      ID=ID1
325   ISTART=ISTART-4
C * PRODUCE YEAR PART OF LABEL
      CALL IFORM(FLOAT(IY),4,LABELI(ISTART),IBLANK)
C * PRODUCE MONTH AND DAY IF LABEL TYPE IS DAYS
      IF(LABTYP .NE. 3) GO TO 350
      ISTART=ISTART-1
      LABELI(ISTART)=IBLANK
      CALL YDYMD(IY,ID,IY3,MON,IDAY)
      IY=IY3
      ISTART=ISTART-3
      CALL ALFSET(FLOAT(MON),3,6,LABELI(ISTART))
      ISTART=ISTART-1
      LABELI(ISTART)=IBLANK
      ISTART=ISTART-2
      CALL IFORM(FLOAT(IDAY),2,LABELI(ISTART),IBLANK)
350   IF(IY .EQ. IY1) GO TO 450
      ISTART=ISTART-1
      LABELI(ISTART)=45
      GO TO 300
C * CODE TO PRODUCE SCALE FACTOR TYPE REMOTE LABEL
400   IEXP=COMGET(NBASE+18)
      IF(IEXP .EQ. 0) GO TO 500
      CALL EXPOUT(NBASE,IEXP,LABELI,25,IBLANK)
      ISTART=1
C * COMPUTE JUSTIFICATION OFFSET
```

Feature Eliminated: Logarithmic Plots

Saves : 1 dummy routine (LOGTIX) and 25 lines of FORTRAN code

Modified Subroutines: GRID
LOPTIM

Note: If these changes are made, the A-level changes for logarithmic plots are unnecessary.

Modification to GRID

Change from:

```
C * THIS IS THE Y AXIS TIC MARKING LOOP
      DO 300 I=1,NLIM
      IF(NOMTIX) GO TO 200
      MLIM=MTICS-1
      BOTM=BOT
100   IF(MLIM)190,200,110
110   BOTM=BOTM+TMNTVL
      CALL MOVABS(MSTART,IFIX(BOTM))
      CALL DRWABS(MEND,IFIX(BOTM))
      MLIM=MLIM-1
```

To:

```
C * THIS IS THE Y AXIS TIC MARKING LOOP
      DO 300 I=1,NLIM
      IF(NOMTIX) GO TO 200
      MLIM=MTICS-1
      BOTM=BOT
100   IF(MLIM)200,200,110
110   BOTM=BOTM+TMNTVL
      CALL MOVABS(MSTART,IFIX(BOTM))
      CALL DRWABS(MEND,IFIX(BOTM))
      MLIM=MLIM-1
```

Delete indicated lines:

```
110   BOTM=BOTM+TMNTVL
      CALL MOVABS(MSTART,IFIX(BOTM))
      CALL DRWABS(MEND,IFIX(BOTM))
      MLIM=MLIM-1
      GO TO 100
190   CALL LOGTIX(NBASE,BOT,TINTVL,MSTART,MEND)
200   BOT=BOT+TINTVL
      CALL MOVABS(ISTART,IFIX(BOT))
300   CALL DRWABS(IEND,IFIX(BOT))
C * THE FOLLOWING CODE DRAWS THE X AXIS
```

Feature Eliminated: Logarithmic Plots (continued)

<u>Modification to GRID</u> (continued)

Change from:

```
C * THIS  IS THE X AXIS TIC MARKING LOOP
      DO 700 I=1,NLIM
      IF(NOMTIX) GO TO 600
      MLIM=MTICS-1
      XLEFTM=XLEFT
500   IF(MLIM)590,600,510
510   XLEFTM=XLEFTM+TMNTVL
      CALL MOVABS(IFIX(XLEFTM),MSTART)
      CALL DRWABS(IFIX(XLEFTM),MEND)
      MLIM=MLIM-1
```

To:

```
C * THIS  IS THE X AXIS TIC MARKING LOOP
      DO 700 I=1,NLIM
      IF(NOMTIX) GO TO 600
      MLIM=MTICS-1
      XLEFTM=XLEFT
500   IF(MLIM)600,600,510
510   XLEFTM=XLEFTM+TMNTVL
      CALL MOVABS(IFIX(XLEFTM),MSTART)
      CALL DRWABS(IFIX(XLEFTM),MEND)
      MLIM=MLIM-1
```

Delete indicated lines:

```
510   XLEFTM=XLEFTM+TMNTVL
      CALL MOVABS(IFIX(XLEFTM),MSTART)
      CALL DRWABS(IFIX(XLEFTM),MEND)
      MLIM=MLIM-1
      GO TO 500
590   CALL LOGTIX(NBASE,XLEFT,TINTVL,MSTART,MEND)
600   XLEFT=XLEFT+TINTVL
      CALL MOVABS(IFIX(XLEFT),ISTART)
700   CALL DRWABS(IFIX(XLEFT),IEND)
800   RETURN
```

Feature Eliminated: Logarithmic Plots (continued)

Modification to LOPTIM

Delete indicated lines:

```
      AMIN=COMGET(NBASE+11)
      AMAX=COMGET(NBASE+12)
      NTICS=NORIG
C * SET DEFAULT MINOR TICS TO FIRST ENTRY IN TABLE
      I=1
200   IF(.NOT.LOG) GO TO 250
C * FOLLOWING CODE PROTECTS AGAINST ATTEMPTING TO FIND LOG OF ZERO
      AMIN=AMAX1(AMIN,1./FINFIN)
      AMAX=ALOG10(AMAX)
      AMIN=ALOG10(AMIN)+0.0000001
C * COMPUTE DATA RANGE
C *   FOLLOWING CODE IS PART OF LOOP TO ADJUST SCALE TO EXISTING TIC MAR
250   RANGE=AMAX-AMIN
      AMINOR=AMIN
      AMAXOR=AMAX
C * CHECK TO SEE IF NUM OF TICS ALREADY SPECIFIED
      IF(NTICS .NE. 0) GO TO 300
```

```
C * REDUCE NO. OF MINOR TICS IF DENSITY NOT HIGHEST
      IF(IDEN .LT. 9)MTCS=5
      IF(IDEN .LT. 7)MTCS=2
      IF(.NOT.LOG) GO TO 500
C * COMPUTE MINOR TICS FOR LOG AXES
      ITINT=TINT
      LTINUL=LEN/NTICS
C * SINGLE DECADES ARE NOT SUBDIVIDED BY MINOR TICS
      IF(ITINT .EQ. 1) GO TO 450
C * START WITH APPROXIMATION EQUAL TO THE NO. OF DECADES PER MAJOR INT.
      MTCS=ITINT
C * COMPUTE MAX NO. OF MINOR TICS BASED ON DENSITY
      MXTCS=10*LTINUL/IDIV
C * FIND NO. OF TICS < MXTCS WHICH DIVIDE THE DATA INTERVAL EVENLY
      LIMTIX=ITINT-1
      DO 445 I=1,LIMTIX
      MTCS=ITINT/I
      IF(MTCS*I .NE. ITINT) GO TO 445
      IF(MTCS .LE. MXTCS) GO TO 500
445   CONTINUE
C * NO SUITABLE NUMBER FOUND - NO MINOR TICS SPECIFIED
      MTCS=0
      GO TO 500
C * IF THE SCREEN INTERVAL LARGE ENOUGH - LOG TIX ARE DRAWN-MTICS=-1
450   IF(LTINUL .GE. 100)MTCS=-1
C * IF SPACE ALLOWS, LOG TICS ARE LABELED-NO. OF MINOR TICS =-2
      IF(LTINUL .GE. 350)MTCS=-2
C *   SET NUMBER OF MINOR TIC INTERVALS
500   CALL COMSET(NBASE+8,FLOAT(MTCS))
C *   GIVE EXTRA DECIMAL DIGIT FOR NON NEAT LABLES
C *   STORE POSITION OF LEAST SIGNIFICANT DIGIT IN LABLE
600   CALL COMSET(NBASE+16,FLOAT(LSIG))
C *   STORE NEW DATA LIMITS (AS WILL APPEAR ON LABLES
```

Feature Eliminated: Logarithmic Plots (continued)

Modification to LOPTIM (continued)

Delete indicated lines:

```
      IF(LTINVL .GE. 350)MTCS=-2
C *   SET NUMBER OF MINOR TIC INTERVALS
500   CALL COMSET(NBASE+8,FLOAT(MTCS))
C *   GIVE EXTRA DECIMAL DIGIT FOR NON NEAT LABLES
C *   STORE POSITION OF LEAST SIGNIFICANT DIGIT IN LABLE
600   CALL COMSET(NBASE+16,FLOAT(LSIG))
C *   STORE NEW DATA LIMITS (AS WILL APPEAR ON LABLES
      CALL COMSET(NBASE+26,AMIN)
      CALL COMSET(NBASE+27,AMAX)
      IF(.NOT.LOG) GO TO 700
C * RESTORE DATA MIN/MAX IF ALTERED FOR LOGS
      AMAX=10.**AMAX
      AMIN=10.**AMIN
C *   STORE NEW DATA LIMITS AS WILL DEFINE GRID
700   CALL COMSET(NBASE+11,AMIN)
      CALL COMSET(NBASE+12,AMAX)
      RETURN
      END
```

Elimination of Exponential Labels

The following changes eliminate various types of exponential labels. These are used not only for logarithmic tic marks, but also for remote labels if tic mark labels are too long for the space available. Therefore, it would not be wise to eliminate all forms of exponential output. Refer to Section 3.2.19, for a description of all exponent types.

Feature Eliminated: Exponential Output
Form M, MM, etc.

Saves : 9 lines of FORTRAN code

Modified Subroutine: EXPOUT

Modification to EXPOUT

Delete indicated lines:

```
      ICHARS=NCHARS
      IF(NEXP .GT. 9) GO TO 300
      IF(NEXP .EQ. 0) GO TO 460
C *   TYPE 4: 1 FOLLOWED BY EXPONENT OF ZEROS
      IF(ITYPE .EQ. 4) GO TO 400
C *   TYPE 3: EXPONENT EQUIVALENT WRITTEN OUT
      IF(ITYPE .EQ. 3) GO TO 100
C *   TYPE 2: M=THOUSANDS     MM=MILLIONS
      IF(ITYPE .EQ. 2) GO TO 200
C *   TYPE 1: (DEFAULT)  X10 TO THE EXPONENT


120   IF(LOG .EQ. 2)I=I-1
      NNN=I-1
      DO 125 III=N,NNN
      IARRAY(ICHARS)=ITEN(I)
      I=I-1
125   ICHARS=ICHARS-1
      GO TO 800
C *   M,MM PROCESSING--NON3 OR NON6 GO TO DEFAULT
200   IF(MOD(NEXP,3) .NE. 0) GO TO 300
      IARRAY(ICHARS)=83
      IARRAY(ICHARS-1)=39
      ICHARS=ICHARS-2
      DO 210 III=3,NEXP,3
      IARRAY(ICHARS)=77
210   ICHARS=ICHARS-1
      GO TO 800
C *   X10 TO EXPONENT PROCESSING
300   IF(IEXP .EQ. 1) GO TO 380
      IF(NEXP .LT. 10) GO TO 350
      IF(NEXP .LT. 100) GO TO 340
```

Feature Eliminated: Exponential Output Form
HUNDREDS, etc.

Saves : 40 lines of FORTRAN and
31 words of storage

Modified Subroutine: EXPOUT

<u>Modification to EXPOUT</u>

Delete indicated lines:

```
C***************************************************
C*                   COPYRIGHT      TEKTRONIX, INC. 1973
C*                                   BEAVERTON, OREGON
C*                ROUTINE: EXPOUT
C***************************************************
      SUBROUTINE EXPOUT(NBASE,IEXP,IARRAY,NCHARS,IFILL)
      INTEGER ITEN(31),IARRAY(2)
      COMMON/BPPCOM/COMGET(80)
C *
C *             00 01 02 03 04 05 06 07 08 09 10 11 12 13
C *                T  E  N  S  H  U  N  D  R  E  D  S
C *             /  /  /  /  /  /  /  /  /  /  /  /  /  /
      DATA ITEN/32,84,69,78,83,72,85,78,68,82,69,68,83,32,
     + 84,72,79,85,83,65,78,68,83,77,73,76,76,73,79,78,83/
C *    /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /  /
C *    T  H  O  U  S  A  N  D  S  M  I  L  L  I  O  N  S
C *    14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
C *
      DATA JUP/-1/
      ITYPE=COMGET(NBASE+21)
      LOG=COMGET(NBASE+15)
      NEXP=IABS(IEXP)
      ICHARS=NCHARS
      IF(NEXP .GT. 9) GO TO 300
      IF(NEXP .EQ. 0) GO TO 460
C *   TYPE 4  1 FOLLOWED BY EXPONENT OF ZEROS
      IF(ITYPE .EQ. 4) GO TO 400
C *   TYPE 3: EXPONENT EQUIVALENT WRITTEN OUT
      IF(ITYPE .EQ. 3) GO TO 100
C *   TYPE 2: M=THOUSANDS     MM=MILLIONS
      IF(ITYPE .EQ. 2) GO TO 200
```

Feature Eliminated: Exponential Output Form
HUNDREDS, etc. (continued)

Modification to EXPOUT
(continued)

Delete indicated lines:

```
C *     TYPE 2: M=THOUSANDS      MM=MILLIONS
        IF(ITYPE .EQ. 2) GO TO 200
C *     TYPE 1: (DEFAULT)  X10 TO THE EXPONENT
        IF(ITYPE .EQ. 1) GO TO 300
        GO TO 300
100     IF(NCHARS .LE. 17) GO TO 300
        GO TO (101,102,103,104,105,106,107,108,109),NEXP
C * OUTPUT TENS
101     N=1
        I=4+N
        GO TO 120
C * OUTPUT HUNDREDS
102     N=5
        I=8+N
        GO TO 120
C * OUTPUT THOUSANDS
103     N=14
        I=9+N
        GO TO 120
C * OUTPUT TEN THOUSANDS
104     NN=13
        II=10+NN
        N=1
        I=3+N
        GO TO 110
C * OUTPUT HUNDRED THOUSANDS
105     NN=13
        II=10+NN
        N=5
        I=7+N
        GO TO 110
C * OUTPUT MILLIONS
106     N=23
        I=8+N
        GO TO 120
107     GO TO 300
108     GO TO 300
109     GO TO 300
C * COMMON CHARACTER SETTING LOOP FOR 2ND WORD
110     NNN=II-1
        DO 115 III=NN,NNN
        IARRAY(ICHARS)=ITEN(II)
        II=II-1
115     ICHARS=ICHARS-1
C * COMMON CHARACTER SETTING LOOP FOR 1ST WORD
120     IF(LOG .EQ. 2)I=I-1
        NNN=I-1
        DO 125 III=N,NNN
        IARRAY(ICHARS)=ITEN(I)
        I=I-1
125     ICHARS=ICHARS-1
        GO TO 800
C *     M,MM PROCESSING--NON3 OR NON6 GO TO DEFAULT
200     IF(MOD(NEXP,3) .NE. 0) GO TO 300
        IARRAY(ICHARS)=83
        IARRAY(ICHARS-1)=39
        ICHARS=ICHARS-2
        DO 210 III=3,NEXP,3
        IARRAY(ICHARS)=77
210     ICHARS=ICHARS-1
        GO TO 800
C *     X10 TO EXPONENT PROCESSING
```

Feature Eliminated: Exponential Output Form 10^n, etc.

Saves : 25 lines of FORTRAN code

Modified Subroutine: EXPOUT

Modification to EXPOUT

Delete indicated lines:

```
      COMMON /BPPCOM/ COMGET(80)
      ITYPE=COMGET(NBASE+21)
      LOG=COMGET(NBASE+15)
      NEXP=IABS(IEXP)
      ICHARS=NCHARS
      IF(NEXP .GT. 9) GO TO 300
      IF(NEXP .EQ. 0) GO TO 400
C *   TYPE 4: 1 FOLLOWED BY EXPONENT OF ZEROS
      IF(ITYPE .EQ. 4) GO TO 400
C *   TYPE 3: EXPONENT EQUIVALENT WRITTEN OUT
      IF(ITYPE .EQ. 3) GO TO 100
C *   TYPE 2: M=THOUSANDS      MM=MILLIONS
      IF(ITYPE .EQ. 2) GO TO 200
C *   TYPE 1: (DEFAULT)  X10 TO THE EXPONENT
      IF(ITYPE .EQ. 1) GO TO 300
      GO TO 300
100   IF(NCHARS .LE. 17) GO TO 300
      GO TO (101,102,103,104,105,106,107,108,109),NEXP
C * OUTPUT TENS
101   N=1
      I=4+N
      GO TO 120
```

Feature Eliminated: Exponential Output Form 10...0

Saves : 4 lines of FORTRAN code

Modified Subroutine: EXPOUT

Modification to EXPOUT

Delete indicated lines:

```
      ICHARS=NCHARS
      IF(NEXP .GT. 9) GO TO 300
      IF(NEXP .EQ. 0) GO TO 460
C *   TYPE 4: 1 FOLLOWED BY EXPONENT OF ZEROS
      IF(ITYPE .EQ. 4) GO TO 400
C *   TYPE 3: EXPONENT EQUIVALENT WRITTEN OUT
      IF(ITYPE .EQ. 3) GO TO 100
C *   TYPE 2: M=THOUSANDS    MM=MILLIONS
      IF(ITYPE .EQ. 2) GO TO 200
C *   TYPE 1: (DEFAULT)  X10 TO THE EXPONENT
```

```
      IF(LOG .EQ. 2) GO TO 800
      IARRAY(ICHARS)=88
      ICHARS=ICHARS-1
      GO TO 800
C *   1 FOLLOWED BY ZEROS PROCESS
400   DO 450 III=1,NEXP
      IARRAY(ICHARS)=48
450   ICHARS=ICHARS-1
460   IARRAY(ICHARS)=49
      ICHARS=ICHARS-1
800   IF(ICHARS .EQ. 0) GO TO 999
      IARRAY(ICHARS)=IFILL
      ICHARS=ICHARS-1
      GO TO 800
999   RETURN
```

The following changes are to eliminate user written options. Refer to Section 14 to determine if any of these features should be eliminated.

In the supplied AG-II package, the user routines ULINE, UPOINT, USERS, USESET, UMNMX, and SOFTEK are already dummy routines.

Feature Eliminated: User Defined Plot Lines

Saves : 1 dummy routine (ULINE) and 4 lines of FORTRAN

Modified Subroutine: CPLOT

Modification to CPLOT

Delete indicated lines:

```
C * GET FIRST DATA POINT
      XPOINT=DATGET(X,1,KEYX)
      YPOINT=DATGET(Y,1,KEYY)
C * MOVE TO FIRST DATA LOCATION
      CALL MOVEA(XPOINT,YPOINT)
▌     IF(LINE .LT.-10)CALL ULINE(XPOINT,YPOINT,1)
      IF(LINE .EQ.-2 .OR. LINE .EQ.-3)CALL BAR(XPOINT,YPOINT,LINE)
      IF(SYMBOL) CALL BSYMS(XPOINT,YPOINT,ISYM)
C * THE FOLLOWING CODE PREPARES BRANCHES FOR THE
C * TYPE OF LINE TO BE USED IN THIS PLOT
```

```
525   ASSIGN 550 TO LINES
      IF(LINE .EQ. 0) ASSIGN 600 TO LINES
      IF(LINE .EQ. 1)LINE=0
      IF(LINE .EQ.-2 .OR. LINE .EQ.-3) ASSIGN 650 TO LINES
      IF(LINE .EQ. 4) ASSIGN 750 TO LINES
▌     IF(LINE .LT.-10) ASSIGN 800 TO LINES
C * THE FOLLOWING IS THE CURVE PLOTTING LOOP
      I1=ISTEPL+1
      ICOUNT=ISTEPS
      DO 900 I=I1,LIMIT,ISTEPL
```

Change from:

```
C * THIS TRAP ALLOWS PLOTS TO SKIP MISSING DATA POINTS WHICH ARE
C     'FILLED' WITH 'MACHINE'S INFINITY' VALUES
      IF(XPOINT .GE. FINFIN) GO TO 900
      IF(YPOINT .GE. FINFIN) GO TO 900
      GO TO LINES,(550,600,650,750,800)
C * DRAW DASHED OR SOLID LINE
550   CALL DASHA(XPOINT,YPOINT,LINE)
      GO TO 850
```

To:

```
C * THIS TRAP ALLOWS PLOTS TO SKIP MISSING DATA POINTS WHICH ARE
C     'FILLED' WITH 'MACHINE'S INFINITY' VALUES
      IF(XPOINT .GE. FINFIN) GO TO 900
      IF(YPOINT .GE. FINFIN) GO TO 900
      GO TO LINES,(550,600,650,750)
C * DRAW DASHED OR SOLID LINE
550   CALL DASHA(XPOINT,YPOINT,LINE)
      GO TO 850
```

Feature Eliminated: User Defined Plot Lines (continued)

Modification to CPLOT
(continued)

Delete indicated lines:

```
      GO TO 850
C * PLOT POINT
750   CALL POINTA(XPOINT,YPOINT)
      GO TO 850
C * USER WRITTEN LINE ROUTINE
800   CALL ULINE(XPOINT,YPOINT,I)
850   IF(.NOT.SYMBOL) GO TO 900
      ICOUNT=ICOUNT-1
      IF(ICOUNT .GT. 0) GO TO 900
      ICOUNT=ISTEPS
```

Feature Eliminated: User Defined Point Plots

Saves : 1 dummy routine (UPOINT) and
2 lines of FORTRAN code

Modified Subroutine: FUNCTION DATGET

Modification to DATGET

Change from:

```
C*                                              BEAVERTON,OREGON
C*                    ROUTINE  DATGET
C*************************************************************
      FUNCTION DATGET(ARR,I,KEY)
      REAL ARR(5)
      GO TO (100,200,300,400,500),KEY
C * STANDARD DATA FORMAT
100   D=ARR(I+1)
      GO TO 600
C * STANDARD SHORT FORM DATA FORMAT
```

To:

```
C*                                              BEAVERTON,OREGON
C*                    ROUTINE  DATGET
C*************************************************************
      FUNCTION DATGET(ARR,I,KEY)
      REAL ARR(5)
      GO TO (100,200,300,100,500),KEY
C * STANDARD DATA FORMAT
100   D=ARR(I+1)
      GO TO 600
C * STANDARD SHORT FORM DATA FORMAT
```

Delete indicated lines:

```
      GO TO 600
C * SHORT FORM CALENDAR FORMAT
300   D=CALPNT(ARR,I)
      GO TO 600
C * USER FORMAT
400   D=UPOINT(ARR,I,OLDONE)
      GO TO 600
C * NON-STANDARD DATA FORMAT
500   D=ARR(I)
600   OLDONE=D
```

Feature Eliminated: User Tic Mark Labels

Saves : 1 dummy routine (USESET) and 3 lines of FORTRAN code

Modified Subroutine: LABEL

Modification to LABEL

Delete indicated lines:

```
C * SET UP INTRA LOOP BRANCH ACCORDING TO LABEL TYPE
      ASSIGN 180 TO LABLE
C * MONTH LABEL=6,DAY LABELS = 3
      IF(LABTYP .EQ. 6 .OR. LABTYP .EQ. 3)ASSIGN 190 TO LABLE
C * NEGATIVE LABEL TYPES FOR USER WRITTEN LABEL ROUTINE
      IF(LABTYP .LT. 0)ASSIGN 170 TO LABLE
      DPOS=DMIN
      SPOS=SMIN
C * IF CALENDAR DATA, FIRST TIC MARK IS NOT LABELED
      IF(.NOT.CAL) GO TO 150
```

```
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE
C * USER WRITTEN LABELING OPTION
170   CALL USESET(FNUM,IWIDTH,NBASE,LABELI)
      GO TO 195
C * GENERATE NUMERIC LABEL STRING
180   CALL NUMSET(FNUM*FAC,IWIDTH,NBASE,LABELI,IBLANK)
      GO TO 195
```

Change from:

```
      ILIM=ILIM-1
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      IF(LABTYP .GT. 0)FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE,(170,180,190)
C * USER WRITTEN LABELING OPTION
170   CALL USESET(DPOS,IWIDTH,NBASE,LABELI)
      GO TO 195
*
```

To:

```
      ILIM=ILIM-1
C * MAIN LABELING LOOP
150   DO 400 I=1,ILIM,ISTEP
      IF(LABTYP .GT. 0)FNUM=AMOD(DPOS-1.,AMODD(LABTYP))+1.
      GO TO LABLE,(180,190)
C * USER WRITTEN LABELING OPTION
170   CALL USESET(DPOS,IWIDTH,NBASE,LABELI)
      GO TO 195
```

Feature Eliminated: User Symbols

Saves : 1 dummy routine (USERS) and 2 lines of FORTRAN code

Modified Subroutine: BSYMS

Modification to BSYMS

Delete indicated lines:

```
C**************************************************************
      SUBROUTINE BSYMS(X,Y,ISYM)
      LOGICAL GENFLG
      CALL PCLIPT(X,Y)
      IF(ISYM .LT. 0) GO TO 200
      IF(GENFLG(1)) GO TO 500
      IF(ISYM .LT. 33) GO TO 100
      IF(ISYM .GT. 127) GO TO 300


      GO TO 400
100   IF(ISYM .GT. 11) GO TO 400
      CALL TEKSYM(ISYM)
      GO TO 400
200   CALL USERS(X,Y,ISYM)
      GO TO 400
300   CALL SOFTEK(ISYM)
400   CALL MOVEA(X,Y)
```

Feature Eliminated: Calculation of Min/Max for User Defined Short Form

Saves : 1 dummy routine (UMNMX) and 2 lines of FORTRAN code

Modified Subroutine: MNMX

Modification to MNMX

Delete indicated lines:

```
      IF(NONSTD .OR. ARRAY(1) .GE. 0.) GO TO 300
      ISET=ABS(ARRAY(1))
      IF(ISET.EQ.1) GO TO 100
      IF(ISET.EQ.2) GO TO 200
C * USER DEFINED MIN/MAX ROUTINE
      CALL UMNMX(ARRAY,AMIN,AMAX)
      GO TO 600
C * SHORT FORM LINEAR DATA ARRAY
100   XMAX=ARRAY(3)+(ARRAY(2)-1.)*ARRAY(4)
      AMIN=AMIN1(ARRAY(3),XMAX,AMIN)
```

Feature Eliminated: Software Plotting Symbols - TEKSYM

Saves : 1 dummy routine (TEKSYM) and
1 line of FORTRAN code

Modified Subroutine: SYMOUT

Note: If these changes are made, the A-level changes for Software Plotting Symbols are unnecessary.

Modification to SYMOUT

Delete indicated lines

```
      CALL TOUTPT(ISYM)
      GO TO 400
100   IF(ISYM .LT. 0) GO TO 200
      IF(ISYM .GT. 11) GO TO 400
      CALL TEKSYM(ISYM,FACTOR)
      GO TO 400
200   CALL USERS(X,Y,ISYM)
      GO TO 400
300   CALL SOFTEK(ISYM)
```

SECTION 14

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 by themselves before merging with the standard package.

This short section of documentation is not intended to be exhaustive but should allow the adventurous user to skip the simple errors and leap directly to the esoteric program bugs that challenge the mind.

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 4010A05 PLOT 10 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 line type with a negative value. If several patterns are required, several negative line types may be choosen 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

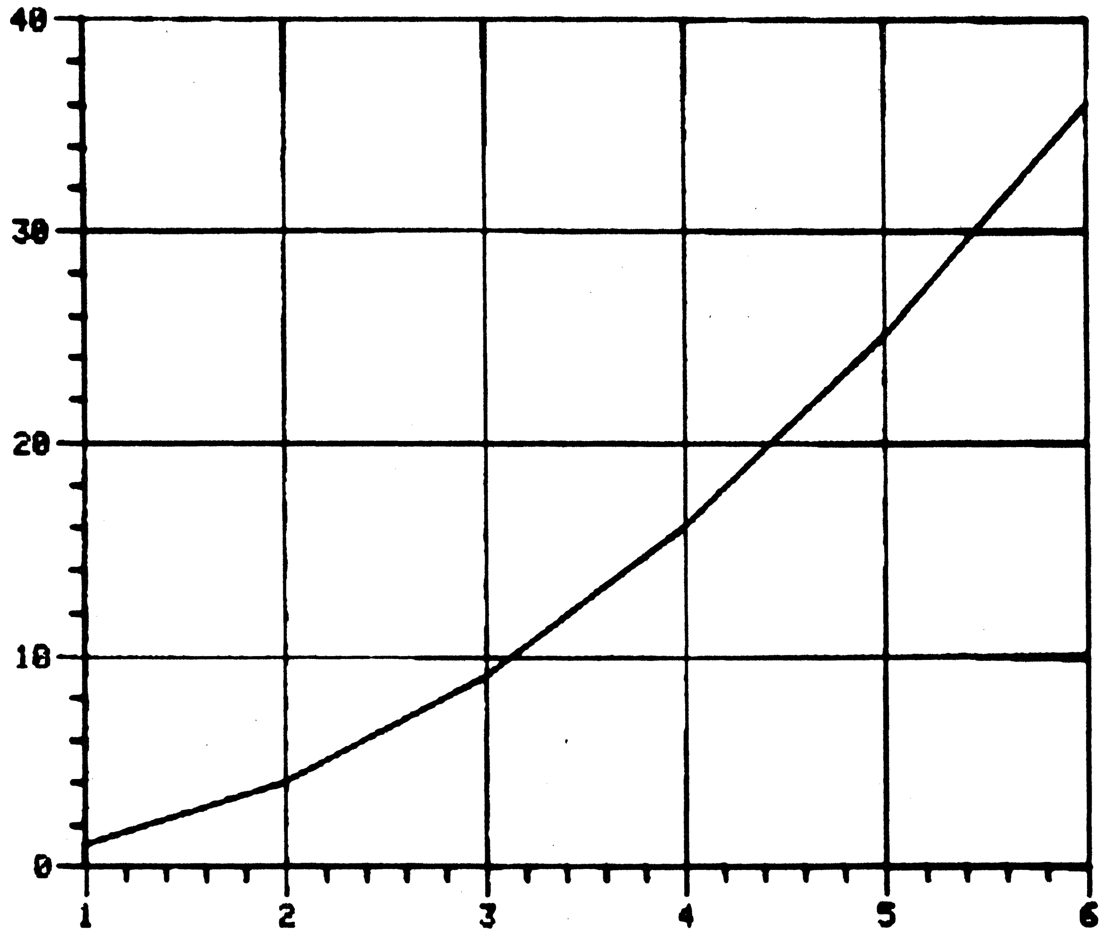
I number (Index) of point being plotted. Runs from 1 to number of points in steps of STEPL

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.

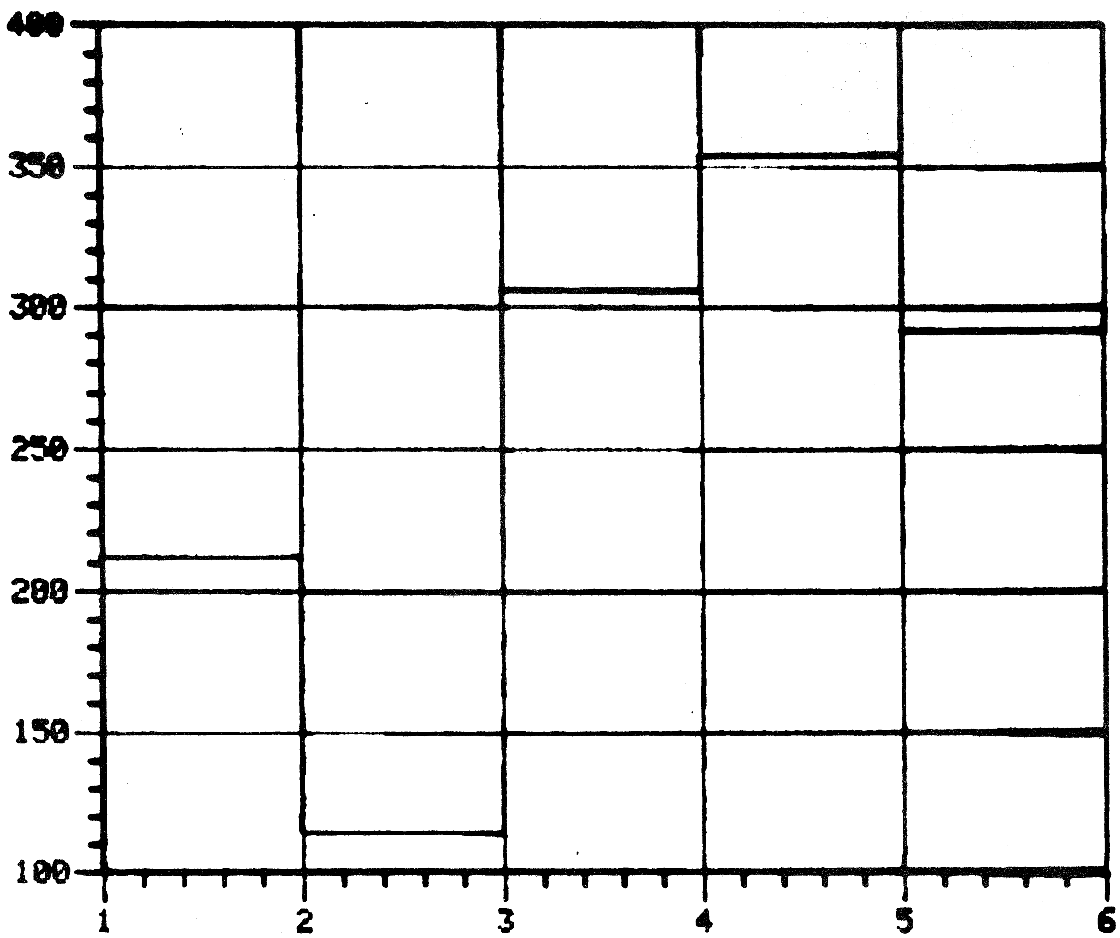
ARGUMENTS:

- 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.

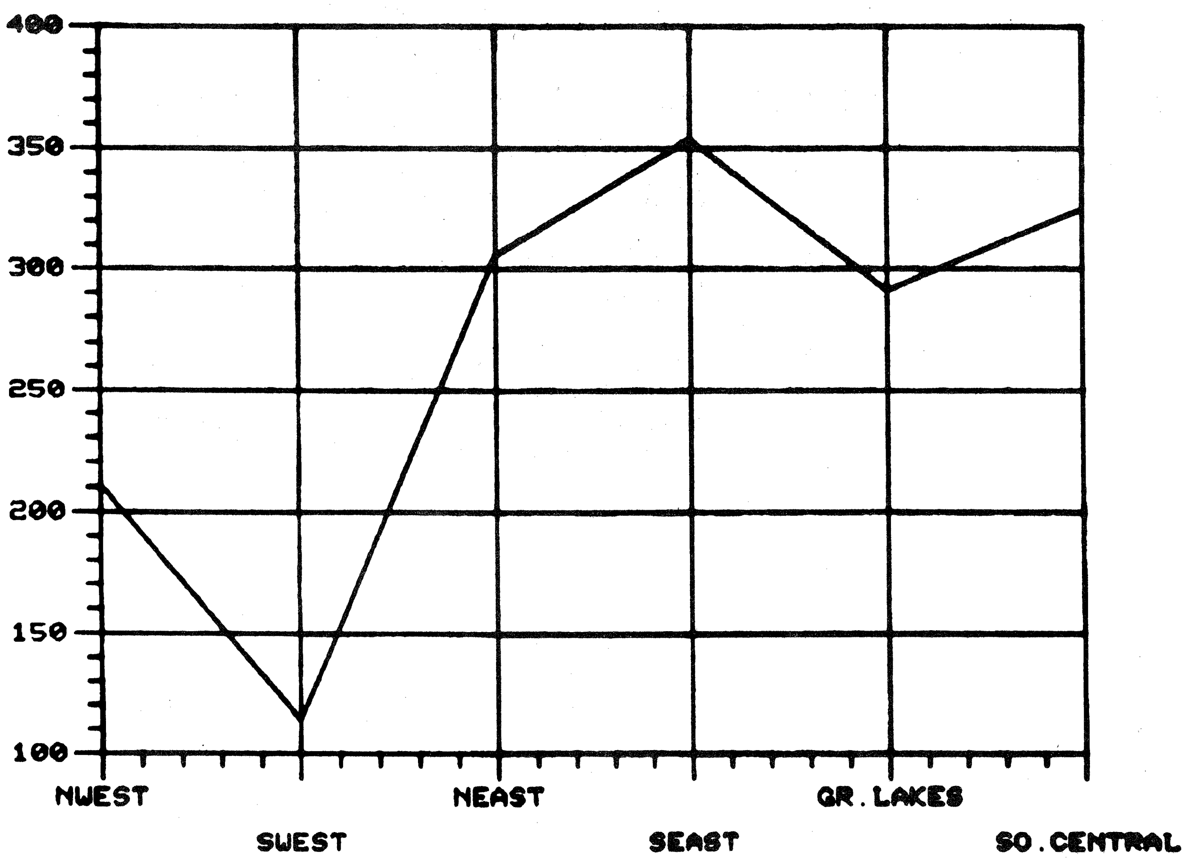
For two years of months beginning in July, FNUM would be 7,8,9,...28,29,30.

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)

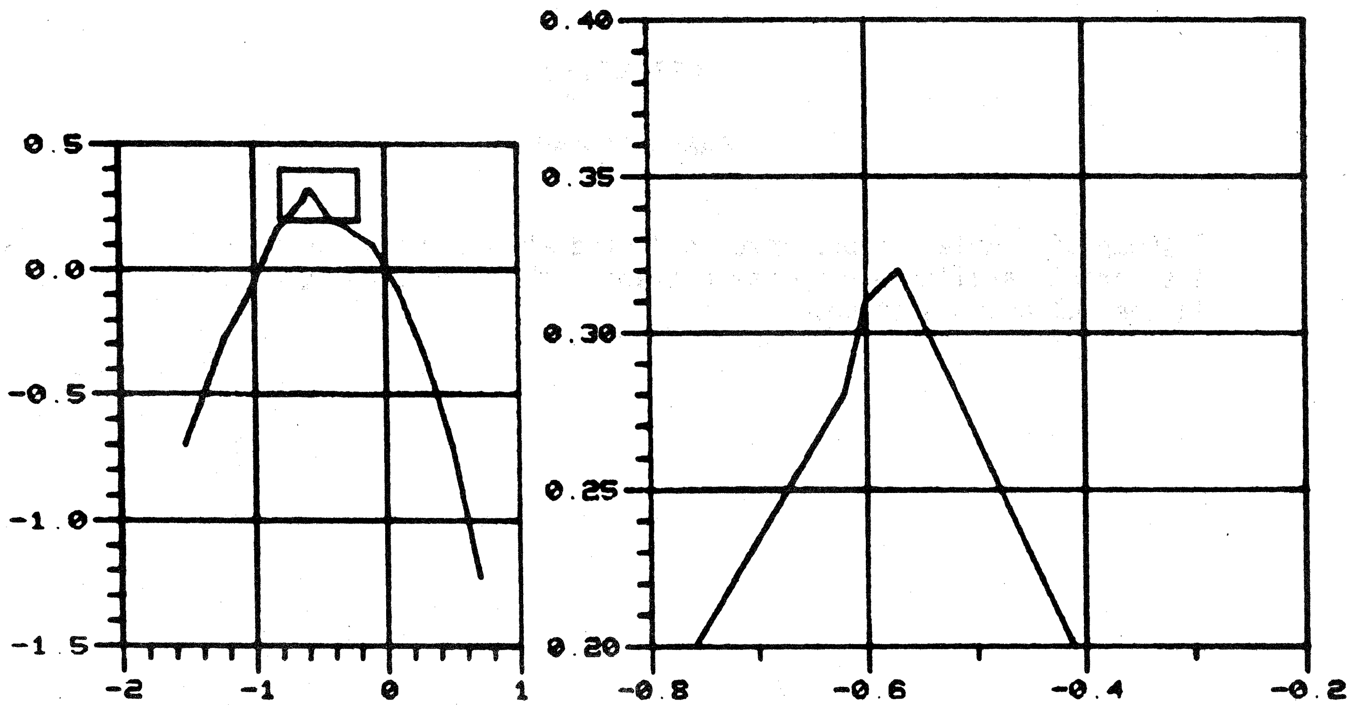
PURPOSE: This routine allows symbols to be used from an independent character generation system.

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

APPENDIX A

SAMPLE GRAPHS

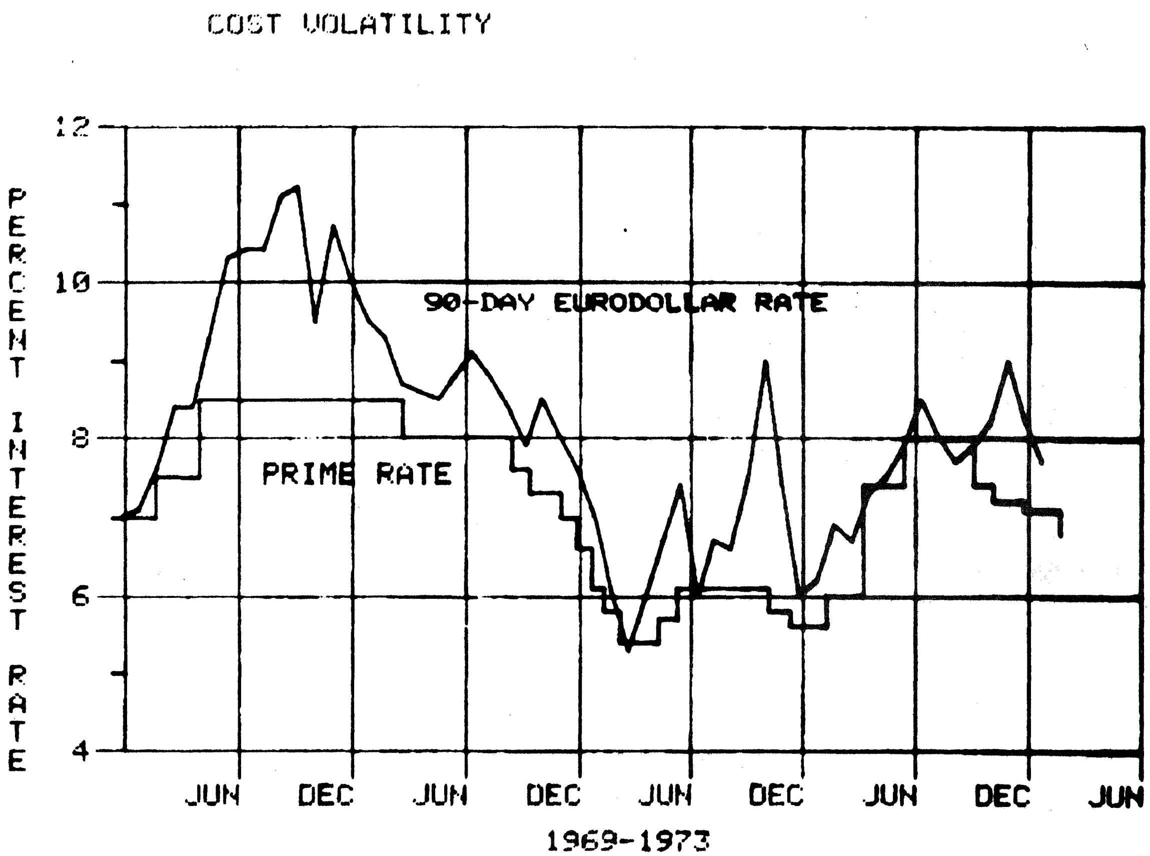
A group of complex graphs were collected at the end of Section 1 to demonstrate the variety available with the package. The code for these examples is contained in the following section.




```
      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(30)
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(500,1000)
      CALL SLIMY(200,700)
      CALL CHECK(XPTS,YPTS)
      CALL DSPLAY(XPTS,YPTS)
      CALL TINPUT(K)
      CALL FINITT(0,700)
      STOP
      END
```

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

Figure A.1

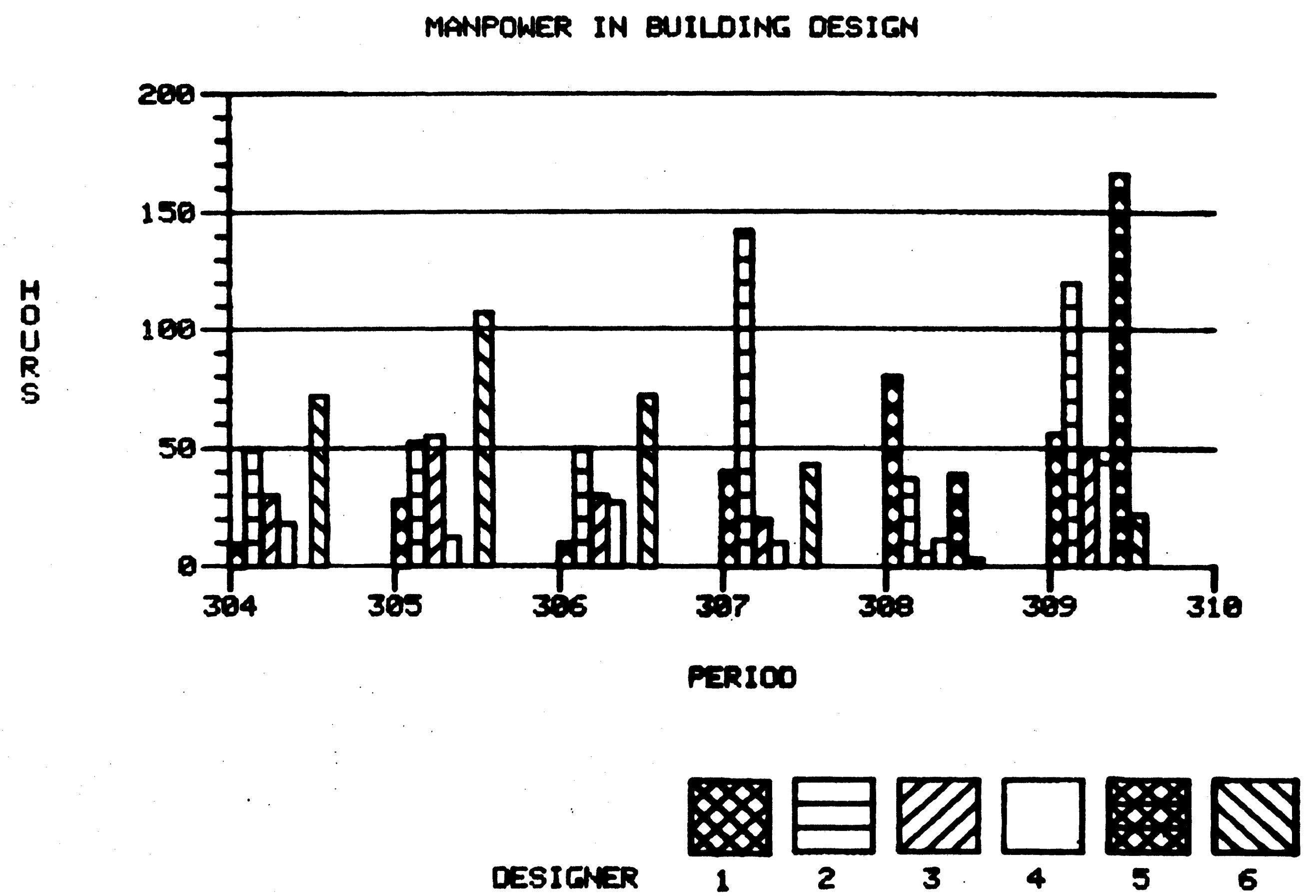


The Cost Volatility graph demonstrates the use of a multiple curve with a user written subroutine ULINE to create a stepped data line

Figure A.2

```
      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.,7.0,7.5,8.5,8.0,7.6,7.3,7.0,6.6,6.1,5.8,5.4,5.7,
     &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 -11
      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
```



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.

Figure A.3

```
C * ROUTINE TO PLOT TIME INVESTED IN BUILDING DESIGN
C *  SINCE PERIOD 304
      REAL PERIOD(11),DESINR(11,6)
      INTEGER 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.,304.,305.,306.,307.,308.,309.,310.,3*0./
      DATA DESINR/6.,10.,28.,10., 40.,80., 56.,4*0.,
     1             6.,49.,52.,49.,142.,37.,120.,4*0.,
     2             6.,30.,55.,30., 20., 6., 49.,4*0.,
     5             6.,18.,12.,27.,10.,11.,44.,4*0.,
     6             6., 4*1.E35,          39.,166.,4*0.,
     7             6.,72.,107.,72.,43., 3., 22.,4*0.
     8   /
      CALL INITT(120)
      CALL BINITT
C * SET THE LINE TYPE IN COMMON
      CALL COMSET(IBASEC(0),-11.)
C * SET THE COMMON FOR THE PROPER BAR TYPE
      CALL SIZEL(12.)
      CALL XFRM(2)
      CALL DLIMX(304.,310.)
      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))
200   CALL TINPUT(K)
      IF(K .NE. 65)GO TO 200
      CALL FINITT(0,30)
      END
C * SUBROUTINE TO DESIDE THE BAR AND SHADING INSIDE

      SUBROUTINE VLINE(X,Y,I)
      DATA XOFF,DEL/-.06,.1/
      IF(I .NE. 1)GO TO 100
      DATA  J/0/
      XOFF=XOFF+DEL + 0.55
      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 AREAEX
      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
*
```

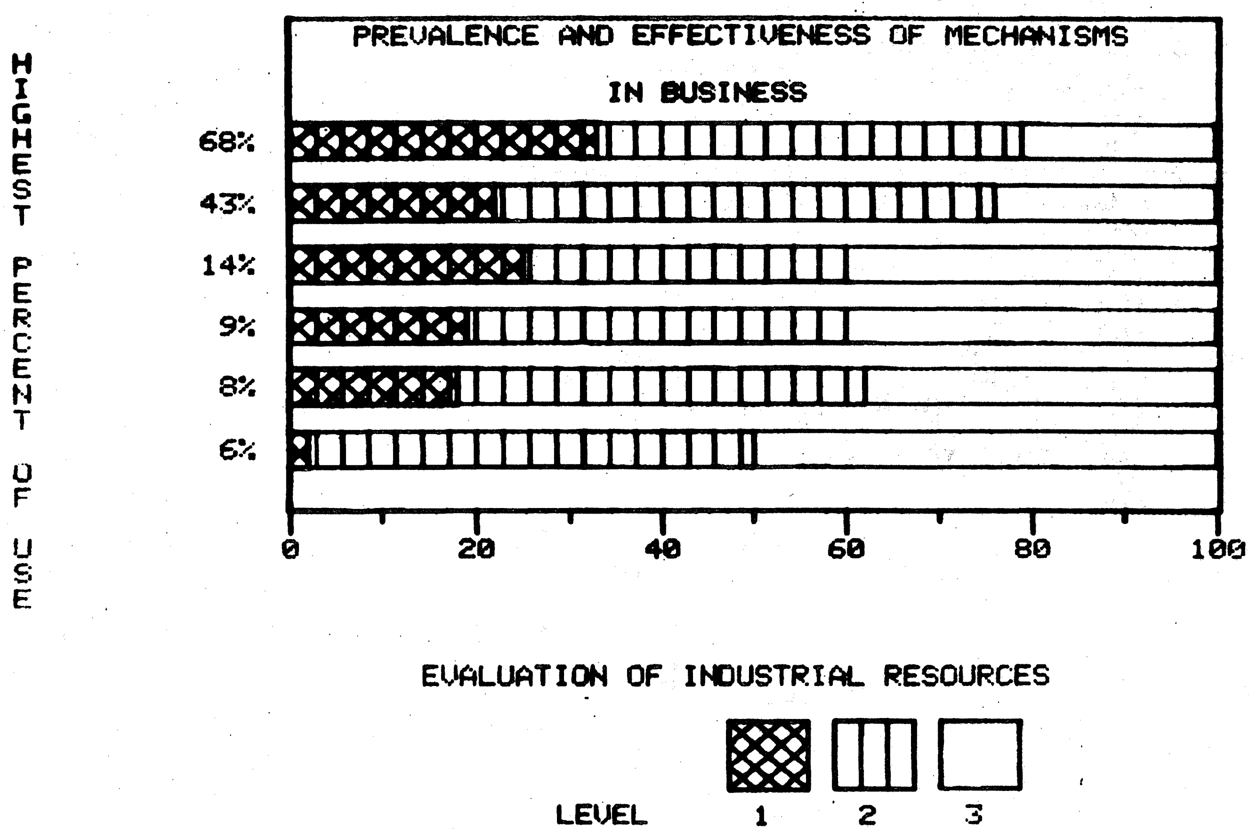
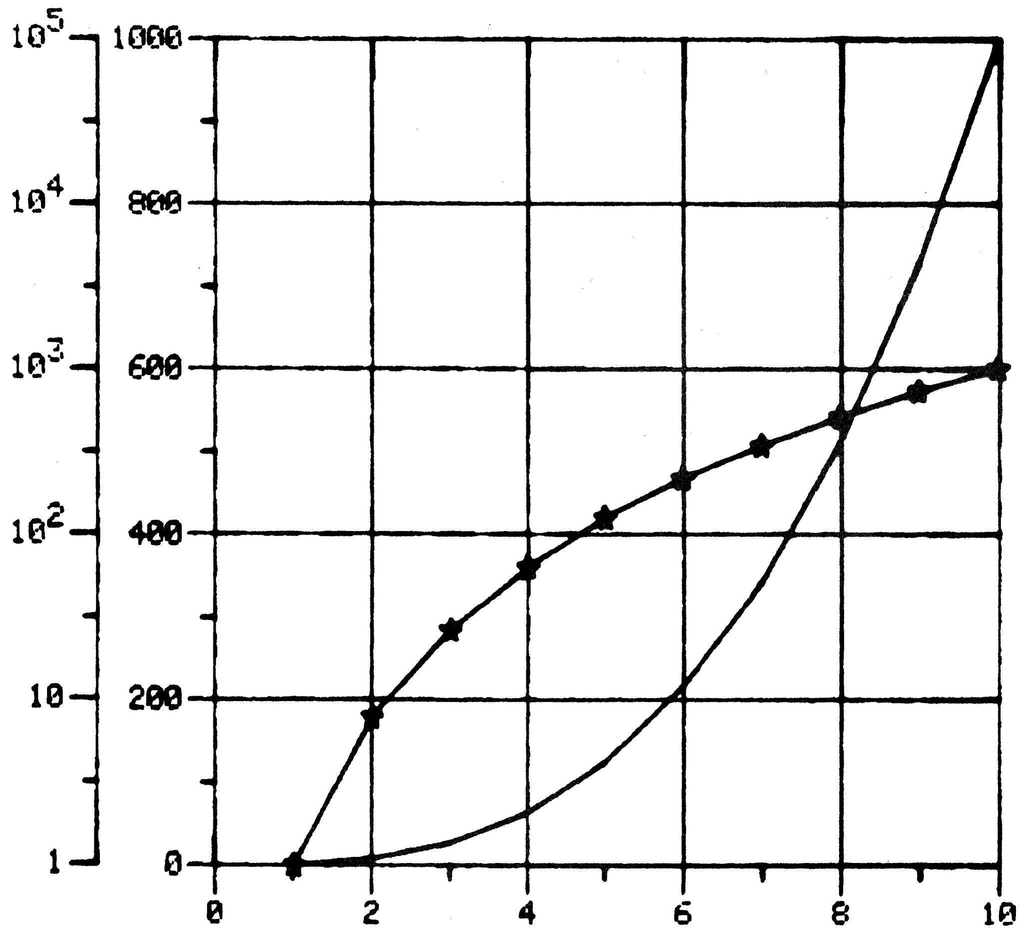

PREVALENCE AND EFFECTIVENESS OF MECHANISMS
IN BUSINESS
HIGHEST PERCENT OF USE
68%
43%
14%
9%
8%
6%
0
20
40
60
80
100
EVALUATION OF INDUSTRIAL RESOURCES
LEVEL
1
2
3

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
```

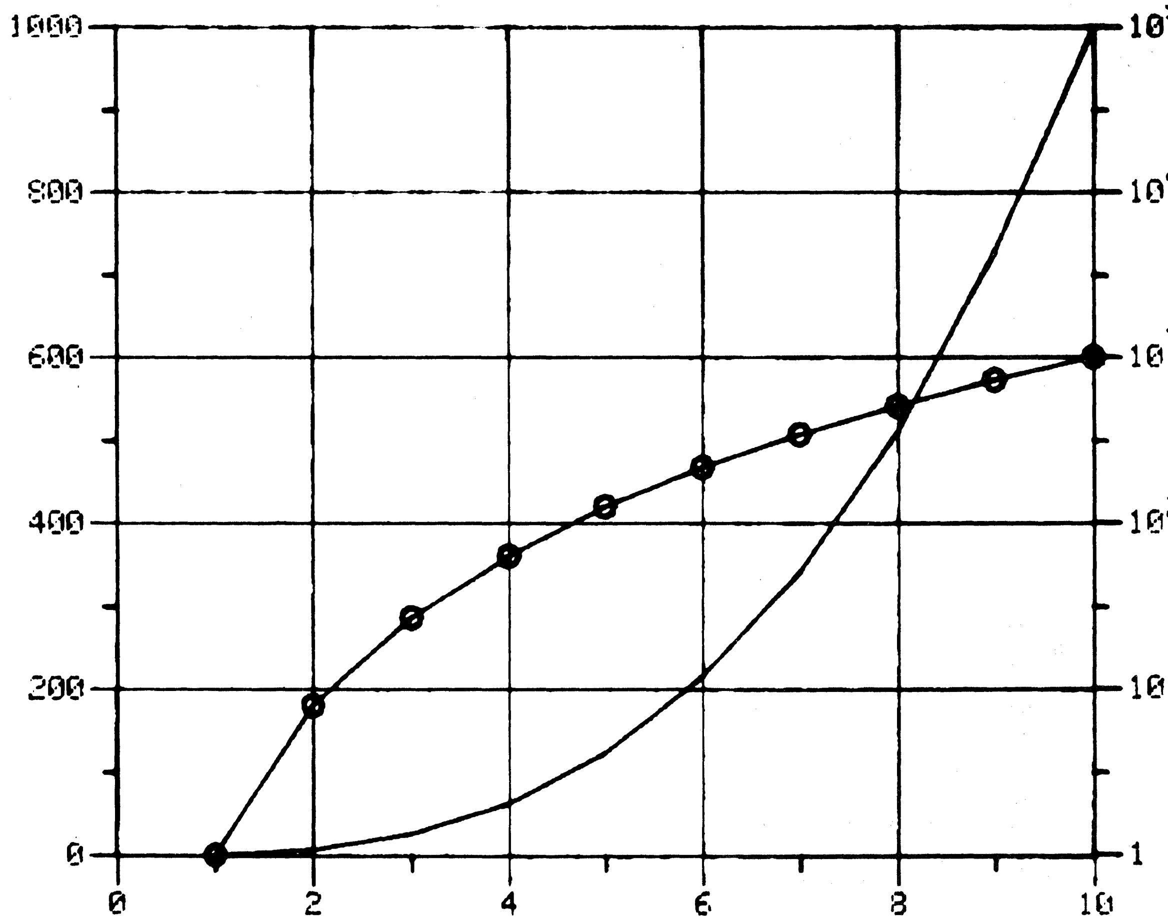
```
C * ROUTINE TO CALCULATE LEGEND
      SUBROUTINE LEDG(IX,IY,I)
      DATA J/0/
      J=J+1
      IF(J .GT. 1)GO TO 100
      ILX=IX-80
      ILY=IY+30
100   ILX=ILX+80
      CALL FILBOX(ILX,ILY,ILX+50,ILY+60,IT(J),20)
      CALL MOVABS(ILX+20,ILY-30)
      CALL ANMODE
      CALL TOUTPT(J+48)
      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
```



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

Figure A.4

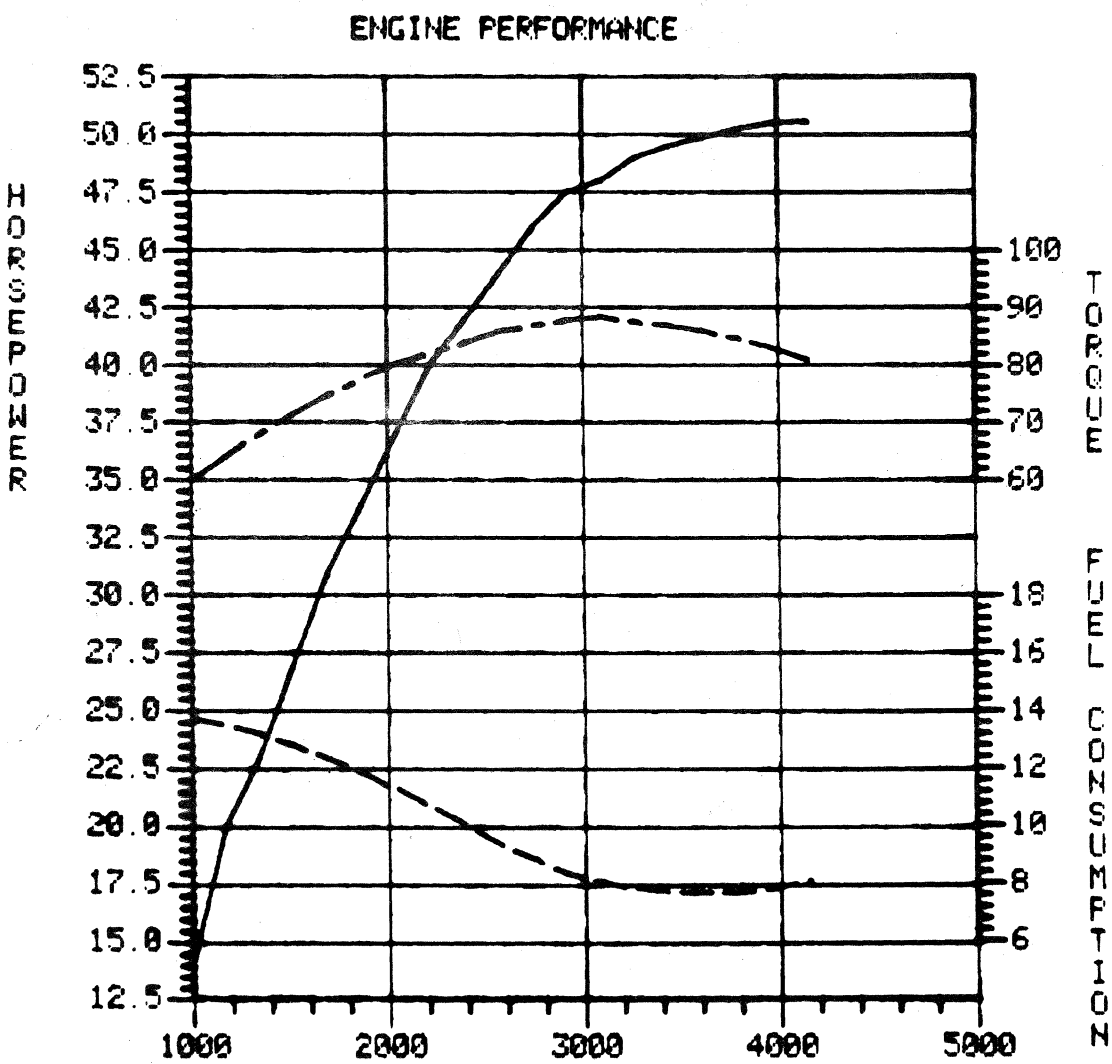
```
C * DEMO PROGRAM USING LOG AND LINEAR PLOTTING
      REAL XPTS(11),YPTS(11)
      XPTS(1)=10
      YPTS(1)=10
C * GENERATE DATA POINTS FOR Y=X**3
      DO 100 I=2,11
      XPTS(I)=FLOAT(I-1)
100   YPTS(I)=FLOAT(I-1)**3
      CALL INITT(120)
      CALL BINITT
C * SET THE HORIZONTAL SCREEN LIMITS
      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 * 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
```

Figure A.5

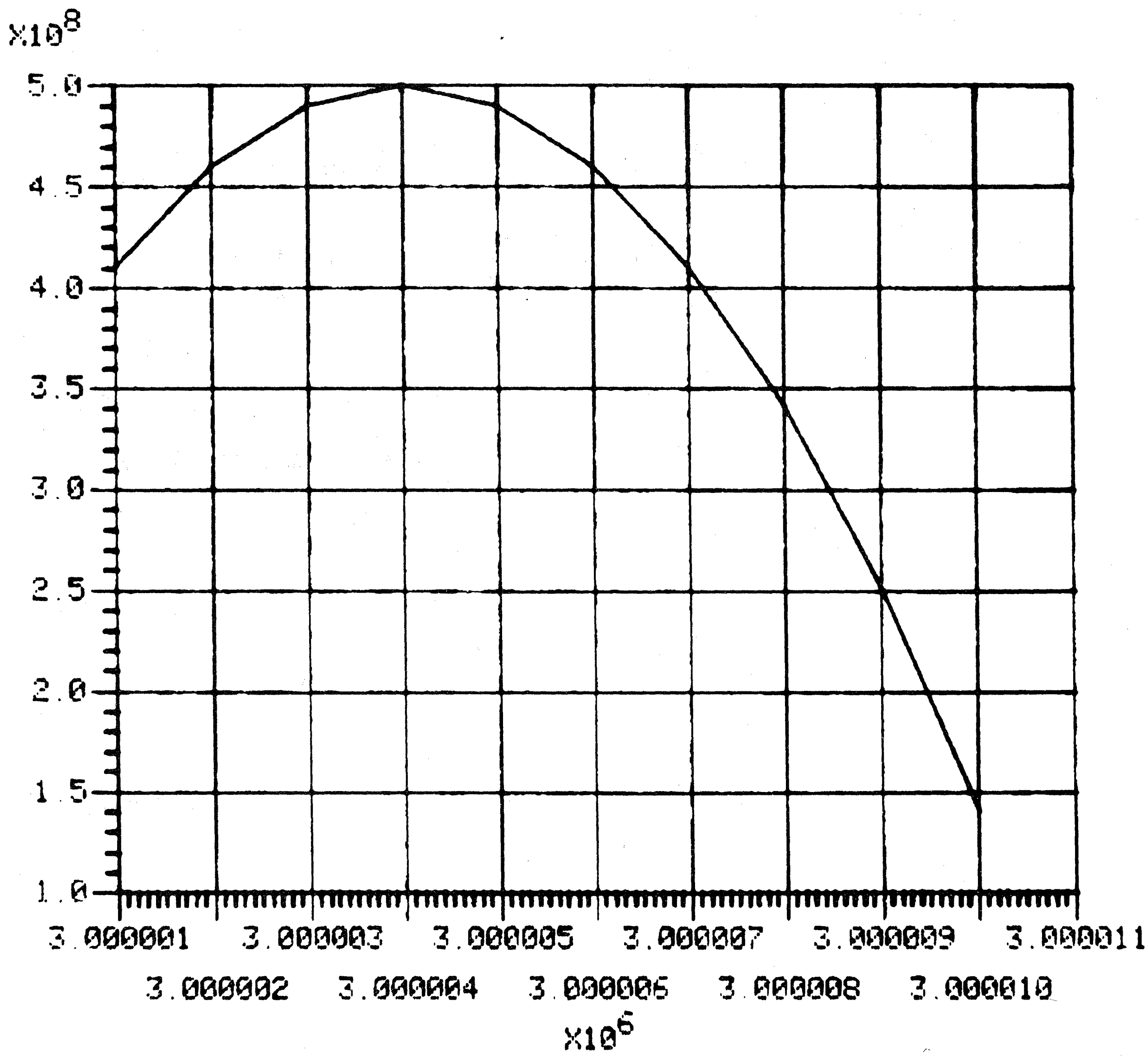
This graph allows comparison of the same data plotted as logarithmic and as linear.



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

Figure A.6

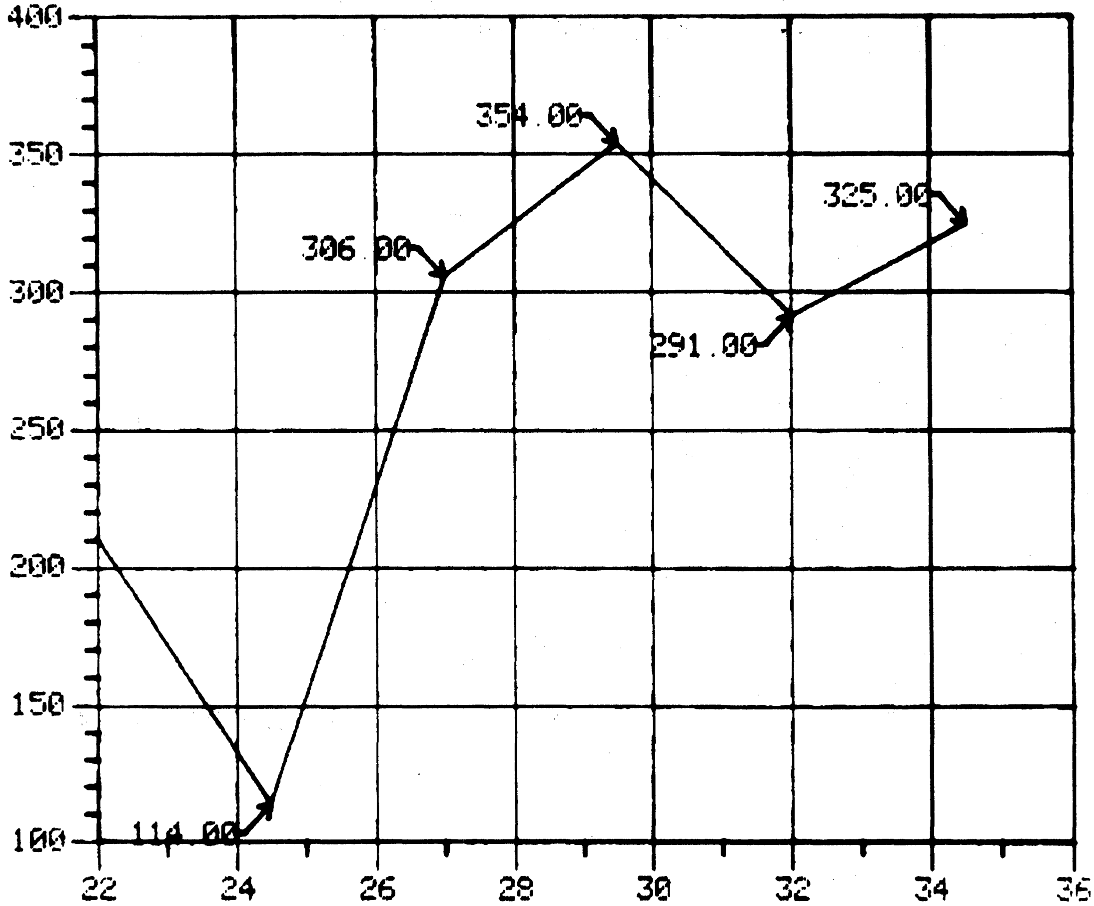
```
      DIMENSION RPM(4),POW(20),SPC(20),TOR(20),LAB1(18),LAB2(10)
     &,LAB3(16),LAB4(6)
C * LABEL LENGTHS
      L1=18
      L2=10
      L3=16
      L4=6
C * GRAPH TITLE: E N G I N E   P E R F O R M A N C E
      DATA LAB1/69,78,71,73,78,69,32,80,69,82,70,79,82,77,65,78,67,69/
C * SCALE LABEL: H 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(1)
      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 * PROGRAM TO DEMONSTRATE 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
```

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

Figure A.7



A user written symbol routine.

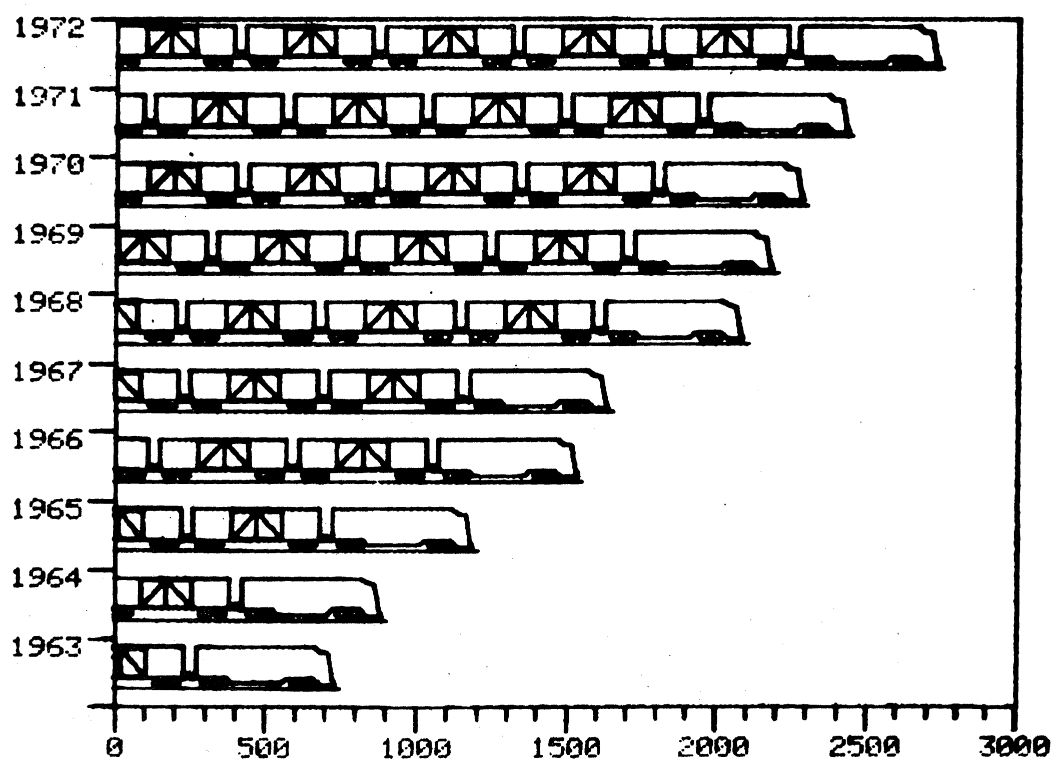
Figure A.8

```
      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
      CALL SYMBL(-1)
      CALL CHECK(XARRAY,YARRAY)
      CALL DSPLAY(XARRAY,YARRAY)
      CALL TINPUT(I)
      CALL FINITT(0,700)
      STOP
      END
C       ROUTINE TO LABLE POINTS WITH Y VALUE
        SUBROUTINE USERS(X,Y,ISYM)
        INTEGER IA(15)
        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 LABLE 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 LABLE
        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 LABLE
        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 LABLE
        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 DRIVER PROGRAM FOR TRAIN GRAPH
      REAL YPTS(5),XPTS(11)
      DATA YPTS/-2.,10.,1.,1963.,1./
      DATA XPTS/10.,750.,900.,1200.,1550.,1650.,2100.,2200.,2300.,
     & 2450.,2750./
      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

C USER LINE ROUTINE TO PLOT TRAIN GRAPHS
      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
```

1963-1972



User written line routine produces train bars

Figure A.9

```
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
        END
```

APPENDIX B

SUBROUTINE CALLING REFERENCES

The following two subroutine reference charts supply quick access to subroutine interaction information.

Reference Chart A

Subroutine	Called By
ALFSET	LABEL, REMLAB
BAR	CPLOT
BINITT	CHECK, COMSET
BSYMS	CPLOT
CALCON	COPTIM
CALPNT	DATGET
CHECK	
CMNMX	MNMX
COMGET	RGCHEK
COMSET	COPTIM, LOPTIM, LWIDTH, PLACE, RGCHEK, SPREAD, TSET, TYPCK, WIDTH
COPTIM	OPTIM
CPLOT	DSPLAY
DATGET	CPLOT
DSPLAY	
EFORM	FFORM
ESPLIT	EFORM
EXPOUT	NUMSET, REMLAB
FFORM	NUMSET
FILBOX	BAR
FINDGE	COPTIM, LOPTIM
FINDLE	
FONLY	EFORM, FFORM
FRAME	
GLINE	MONPOS
GRID	DSPLAY
HLABEL	HSTRIN, NOTATE
HSTRIN	
IBASEC	BAR, CPLOT, LABEL, LOPTIM, MNMX, RGCHEK, TYPCK
IBASEX	BAR, CHECK, DSPLAY, FRAME, GRID, IOTHER, LABEL, PLACE, REMLAB, SETWIN
IBASEY	BAR, CHECK, DSPLAY, FRAME, GLINE, GRID, IOTHER, LOGTIX, PLACE, SETWIN, SPREAD
IFORM	EFORM, FONLY, NUMSET, REMLAB
IOTHER	LABEL, TSET
IUBGC	CALCON, CALPNT, CMNMX, MONPOS
JUSTER	LABEL, REMLAB
KEYSET	CPLOT
LABEL	DSPLAY
LEAP	YDYMD, YMDYD
LOCGE	
LOCLE	OUBGC
LOGTIX	GRID
LOPTIM	OPTIM
LWIDTH	WIDTH
MNMX	RGCHEK
MONPOS	LABEL
NOTATE	LABEL, REMLAB
NUMSET	LABEL
OPTIM	CHECK
OUBGC	CALCON, LABEL, REMLAB
PLACE	
REMLAB	LABEL

Subroutine	Called By
RESCOM	
RGCHEK	CHECK
ROUNDD	CALCON, COPTIM, LOPTIM, LWIDTH
ROUNDU	CALCON, COPTIM, LOPTIM
SAVCOM	
SETWIN	DSPLAY, FILBOX
SOFTEK	SYMOUT
SPREAD	CHECK
SYMOUT	BSYMS
TEKSYM	SYMOUT
TSET	CHECK
TSET2	TSET
TYPCK	CHECK
ULINE	CPLOT
UMNMX	MNMX
UPOINT	DATGET
USERS	BSYMS
USESET	LABEL
WIDTH	CHECK
YDYMD	CALCON, REMLAB
YMDYD	CALCON, CALPNT, CMNMX, MONPOS

NOTE: The routines are arranged in the package in sequential loader order so that called routines follow the routines that call them.

Reference Chart B

Subroutine	Calls
ALFSET	
BAR	FILBOX, IBASEC, IBASEX, IBASEY
BINITT	
BSYMS	SYMOUT, USERS
CALCON	IUBGC, OUBGC, ROUNDD, ROUNDU, YDYMD, YMDYD
CALPNT	IUBGC, YMDYD
CHECK	BINITT, IBASEX, IBASEY, OPTIM, RGCHEK, SPREAD, TSET, TYPCK, WIDTH
CMNMX	IUBGC, YMDYD
COMGET	
COMSET	BINITT
COPTIM	CALCON, COMSET, FINDGE, ROUNDD, ROUNDU
CPLOT	BAR, BSYMS, DATGET, IBASEC, KEYSET, ULINE
DATGET	CALPNT, UPOINT
DSPLAY	CPLOT, GRID, IBASEX, IBASEY, LABEL, SETWIN
EFORM	ESPLIT, FONLY, IFORM
ESPLIT	
EXPOUT	
FFORM	EFORM, FONLY,
FILBOX	SETWIN
FINDGE	
FINDLE	
FONLY	IFORM

Subroutine	Calls
FRAME	IBASEX, IBASEY
GLINE	IBASEY
GRID	IBASEX, IBASEY, LOGTIX
HLABEL	
HSTRIN	HLABEL
IBASEC	
IBASEX	
IBASEY	
IFORM	
IOTHER	IBASEX, IBASEY
IUBGC	
JUSTER	
KEYSET	
LABEL	ALFSET, IBASEC, IBASEX, IOTHER, JUSTER, MONPOS, NOTATE, NUMSET, OUBGC, REMLAB, USESET
LEAP	
LOCGE	
LOCLE	
LOGTIX	IBASEY
LOPTIM	COMSET, FINDGE, IBASEC, ROUNDD, ROUNDU
LWIDTH	COMSET, ROUNDD
MNMX	CMNMX, IBASEC, UMNMX
MONPOS	GLINE, IUBGC, YMDYD
NOTATE	HLABEL
NUMSET	EXPOUT, FFORM, IFORM

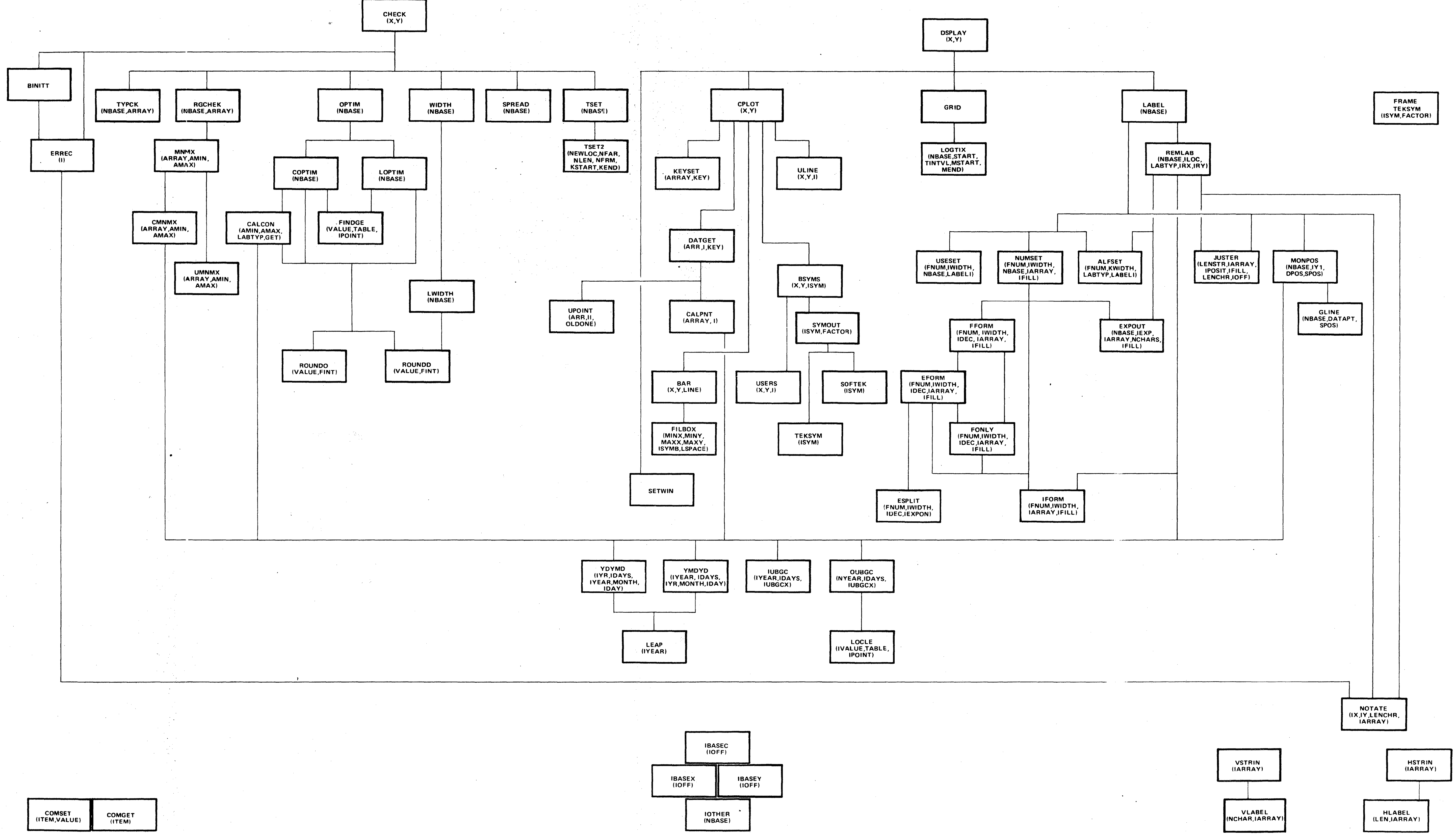
Subroutine	Calls
OPTIM	COPTIM, LOPTIM
OUBGC	LOCLE
PLACE	COMSET, IBASEX, IBASEY
REMLAB	ALFSET, EXPOUT, IBASEX, IFORM, JUSTER, NOTATE, OUBGC, YDYMD
RESCOM	
RGCHEK	COMGET, COMSET, IBASEC, MNMX
ROUNDD	
ROUNDU	
SAVCOM	
SETWIN	IBASEX, IBASEY
SOFTEK	
SPREAD	COMSET, IBASEY
SYMOUT	SOFTEK, TEKSYM
TEKSYM	
TSET	COMSET, IOTHER, TSET2
TSET2	
TYPCK	COMSET, IBASEC
UNLINE	
UMNMX	
UPOINT	
USERS	
USESET	
WIDTH	COMSET, LWIDTH
YDYMD	LEAP
YMDYD	LEAP

Subroutines Which Retain History

The following list shows variables within the code which retain the original values through subsequent executions.

SUBROUTINE	VARIABLES	
BAR	IHALF	
	ISYMB	
	IVRHOR	
	LSPACE	
	MAXX	
	MAXY	
	MINX	
	MINY	
	N	
	NBASE	
CALCON	FNODAY	
	IWEEK1	
CALPNT	ICLTYP	
	ISTPER	
	ISTYR	
	IWEEK1	NO DAYS
DATGET	OLDONE	
TEKSYM	AMULT	
	IFULL	
	IHALF	
	ITEM	
	IX	arrays
	IY	
	MEMORY	
	ROD	

ADVANCED GRAPHING II



USASCII CODE FUNCTIONS

BITS B_4	B_3	B_2	B_1	B_7 Ø B_6 Ø B_5 Ø	Ø Ø 1	Ø 1 Ø	Ø 1 1	1 Ø Ø	1 Ø 1	1 1 Ø	1 1 1
				CONTROL		HIGH X & Y GRAPHIC INPUT		LOW X GRAPHIC INPUT		LOW Y GRAPHIC INPUT	
Ø	Ø	Ø	Ø	NUL Ø	DLE 16	SP 32	Ø 48	@ 64	P 8Ø	` 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 5Ø	B 66	R 82	b 98	r 114
Ø	Ø	1	1	ETX 3	DC3 19	# 35	3 51	C 67	S 83	c 99	s 115
Ø	1	Ø	Ø	EOT 4	DC4 2Ø	$ 36	4 52	D 68	T 84	d 1ØØ	t 116
Ø	1	Ø	1	ENQ 5 STATUS*	NAK 21	% 37	5 53	E 69	U 85	e 1Ø1	u 117
Ø	1	1	Ø	ACK 6	SYN 22	& 38	6 54	F 7Ø	V 86	f 1Ø2	v 118
Ø	1	1	1	BEL 7 BELL	ETB 23 HARDCOPY	' 39	7 55	G 71	W 87	g 1Ø3	w 119
1	Ø	Ø	Ø	BS 8 BACK SPACE	CAN 24	(4Ø	8 56	H 72	X 88	h 1Ø4	x 12Ø
1	Ø	Ø	1	HT 9 SPACE	EM 25	) 41	9 57	I 73	Y 89	i 1Ø5	y 121
1	Ø	1	Ø	LF 1Ø LINE FEED	SUB 26 GRAPH IN*	* 42	: 58	J 74	Z 9Ø	j 1Ø6	z 122
1	Ø	1	1	VT 11 REVERSE LINE FEED	ESC 27	+ 43	; 59	K 75	[91	k 1Ø7	{ 123
1	1	Ø	Ø	FF 12 NEW PAGE*	FS 28	, 44	< 6Ø	L 76	\ 92	l 1Ø8	¦ 124
1	1	Ø	1	CR 13 RETURN	GS 29 VECTOR	- 45	= 61	M 77	] 93	m 1Ø9	} 125
1	1	1	Ø	SO 14	RS 3Ø	. 46	> 62	N 78	^ 94	n 11Ø	~ 126
1	1	1	1	SI 15	US 31 ALPHA	/ 47	? 63	O 79	_ 95	o 111	127 RUBOUT (DEL)
						GRAPHIC INPUT				PRINT IN UPPER CASE	

* CHAR IS PRECEDED BY ESC CHAR TO PERFORM FUNCTION

SUBROUTINE AND FUNCTION INDEX

SUBJECT INDEX