NASA LANGLEY RESEARCH CENTER

Central Scientific
Computing Complex
Document G-13

Common Graphics Library (CGL)
Volume II: Low-Level User's Guide

Nancy L. Taylor

Dana P. Hammond
Pauline M. Theophilos

April 1989

Acknowledgements

The following people have contributed significantly in the development, implementation, verification, and installation of the Common Graphics Library (CGL): David L. Miner, Alicia S. Hofler, Michelle S. Hollis, Scott Nolf, and Debbie M. Garrett. Also contributing in these regards are: Bradford D. Bingel, Eric L. Everton, Mary M. Johnson, Erma L. Wilson, and Helen O. Yue.

We thank the following individuals for their careful reading and constructive comments: Connie Curran, Wesley L. Goodman, Christopher J. Harris, Mary K. McCaskill, and Eleanor C. Wynne.

Common Graphics Library (CGL)
Volume II: Low-Level User's Guide
April 1989

List of Figures

List of Figures (continued)

How To Use This Document

Volume II documents the use of the Common Graphics Library's (CGL) Low-Level routines. This manual is intended to serve both as a user's guide and reference manual.

Section 1 provides an introduction to the CGL by describing its purpose and capabilities. This section acquaints the user with an overview of the New Graphics System (NGS) and reveals the significance of the Common Graphics Library.

Section 2 consists of basic terms and definitions which are used throughout the remainder of the document.

Section 3 consists of a detailed description of how to use the Low-Level routines. This section is primarily intended to provide a basic understanding of the necessary attributes, primitives, graphic components, coordinate systems, and graphical environment necessary to use the Low-Level routines.

Section 4 gives an overview of the error detection and debugging capabilities available in the Low-Level routines.

Section 5 describes how to interface the CGL Low-Level routines with other graphics packages. This section will discuss the requirements necessary to interface the Low-Level routines with any graphics package, then use the LEZ routines as a specific example.

Section 6 gives a general overview of postprocessing considerations.

Appendix A supplies a complete description of each Low-Level routine, providing a description of the arguments, the routine's restrictions, and any relevant notes.

Appendix B is a collection of commented test cases accompanied with the corresponding graphics output. The examples in this section are intended to reflect some of the typical charts generally desired by various groups at Langley.

Appendix C provides a list of DI-3000 routines. Although the user must know DI-3000 to use the Low-Level routines, this list is included for completeness, and provides the functionality of the DI-3000 calls.

Appendix D shows the user how to access and execute the CGL on various computers.

Glossary

ACD	Analysis and Computation Division
attribute	An inherent characteristic generally used in describing an external appearance of an entity (e.g., color, height, etc.)
bar chart	A chart consisting of rectangles with heights representing a dependent value at an associated independent position.
CDC	Control Data Corporation
character axis	An axis which has evenly spaced tick marks with text tick mark labels.
chart	A graphical representation giving information in tabular form. Plots which are "bars" are referred to as "bar charts", while plots which use "pie segments" are referred to as "pie charts".
Common Library	A subroutine library which satisfies LaRC local requirements and generates features not supported by other graphics libraries.
composite chart	A collection of two or more charts on a single display area (terminal screen, graphics frame, etc.).
CORE	A proposed graphics standard developed by the Association of Computing Machinery's Special Interest Group on Graphics (SIGGRAPH).
data coordinates	A user-defined set of coordinates in which the data is to be plotted. This term is used interchangeably with data space. See section 2 for a description and example.
defaults	A set of initial values for a set of attributes.

device driver	A device-dependent program that supports a specific graphics device. The device driver generates device-dependent output from device-independent input and handles device-dependent interaction.
device independence	The ability to control all graphics devices uniformly.
frame	A complete unit of plotting. A final figure. A page.
frame eject	A frame eject clears the display area of the selected graphics device in preparation for the next frame of picture(s).
FSGB	Flight Software and Graphics Branch of ACD
GKS	Graphics Kernel System - a graphics standard approved by ISO.
graphics session	time in which interaction with, or use of a graphics package takes place.
High-Level routines	A set of routines providing an interface to generate specific chart types.
interval units	The increment between numeric major tick labels.
ISO	International Standards Organization
kernel	A subroutine library which contains all required functions for performing interactive and passive graphics tasks.
key	A list of words or phrases giving an explanation of symbols or abbreviations.
LaRC	Langley Research Center
LARCGOS	A library of locally written graphics subroutines and a set of postprocessors that drive the ACD Production Devices. LARCGOS is part of the Old Graphics System (OGS).

line graph	A graph in which points represent values of a variable for suitable values of an independent variable and are connected by line segments. Line graphs available in the Common Graphics Library include linear charts (those with linear axes) and logarithmic charts (those with logarithmic axes).
linear axis	An axis which has a linear progression of numeric major tick mark labels.
linear chart	A chart in which both axes are linear (see section 2 for a description and an example).
logarithmic axis	An axis which has a logarithmic progression of numeric major tick mark labels.
logarithmic chart	A chart in which both axes are logarithmic (see section 2 for a description and an example).
Low-Level routines	A set of routines in the Common Graphics Library which provide graphical support in the generation of plots unique to the graphics requirements of LaRC.
metafile	A sequential file which contains the device-independent picture information necessary to produce the desired graphical output.
metafile translator	The program which interprets the device-independent metafile commands for specific physical devices
MOVIE.BYU	A graphics display system written at Brigham Young University.
NASA line patterns	A set of LaRC preferred line patterns used to represent data in line charts.
NASA Logo	An emblem officially recognized by NASA.
NASA symbols	A set of LaRC preferred symbols used to represent data in line graphs.

NCAR	A library of graphics subroutines written at the National Center for Atmospheric Research.
NGS	New Graphics System at LaRC
NOS	Network Operating System running on Control Data computers.
OCO	Operations Control Office in Building 1268 at LaRC
OGS	Old Graphics System at LaRC
page coordinates	A user-defined set of coordinates in which all charts and their components can be described.
passive graphics	Graphics requiring no dynamic interaction with the display.
PC	Personal computer (e.g., IBM PC XT)
pie chart	A diagram consisting of a circle which is divided by radii, called pie segments, each of which represent a part of a whole entity.
pie segment	Section of a circle consisting of an area swept out by a radius.
postprocessor	A device-dependent program that drives an ACD production device.
PRIMOS	PRIME Operating System
production devices	Batch graphics output facilities at the Central site; includes Calcomp, Varian, and Versatec.
publication standards	A set of requirements describing the appearance and contents of plots which must be satisfied for use in technical documents published through the NASA publication process.
PVI	Precision Visuals, Inc. of Boulder, Colorado
scale factor	The change in value per scale increment (between major tick marks).

scale origin	The beginning lower scale value. This is usually the minimum value or an adjusted minimum value.
selected graphics device	A graphics device which is enabled to receive graphics output or to send graphics input.
semi-log chart	A chart with one linear and one logarithmic axis.
tick mark	Short lines at specified intervals denoting grid.
viewgraph standards	A set of requirements describing the appearance and contents of plots which must be satisfied for use in audience presentations.
workstation	An abstract logical unit consisting of zero or one display surface and zero or more input devices.
2D	Two-dimensional
3D	Three-dimensional

1. INTRODUCTION

The Common Graphics Library (CGL) is a simple-to-use, yet powerful graphics package which enables the user to view data quickly and easily, provides a means of generating publication and/or viewgraph quality charts, and satisfies LaRC's unique graphics requirements.

The use of the Low-Level routines of the CGL, documented in this manual, requires a working knowledge of the DI-3000 graphics package. In developing a graphics program, the Low-Level routines work in conjunction with the DI-3000 routines. The LEZ routines of the CGL should be examined to determine if the user's application can be accomplished at the highest level (i.e., LEZ routines).

The graphics requirements at LaRC are very diverse in nature, reflecting the variety of applications (i.e., scientific, engineering, business, etc.) present at NASA Langley. The CGL and this manual concentrate on 2D chart generation. The three basic chart types generated by the CGL include: line/logarithmic charts, bar charts, and pie charts (see Figure 1-1). A few of the library's most sought after features include:

- multiple plots per frame;
- multiple data sets per plot;
- the ability to access multiple fonts (i.e., Greek, math symbols, etc.) and text mnemonics (i.e., subscripts, superscripts, etc.);
- color capability;
- the creation of metafiles (for hard copy postprocessing); and
- the ability to automatically scale and determine increments for axes suitable for publication quality reports.

1.1 Capabilities and Structure of the Common Graphics Library

The CGL has two user interface levels, a set of generalized High-Level (Langley Easy, or LEZ) routines [G-12], and a set of specific Low-Level routines [G-13] (see Figure 1-2). These two levels reflect the degree of sophistication both in terms of difficulty of use and provided capabilities.

The LEZ routines, intended for users with no graphics experience, provide a stand-alone means to generate complete report-quality charts. At this level, the user controls aspects of an entire chart, that is conceptually designs and composes complete charts without being involved in the complexity of the underlying graphics package. While this level of separation frees the user from learning the underlying graphics package, it also ensures that the application program will require no or little change even if the underlying graphics package is modified or replaced (i.e., possibly by a graphics standard like the Graphics Kernel System - GKS).

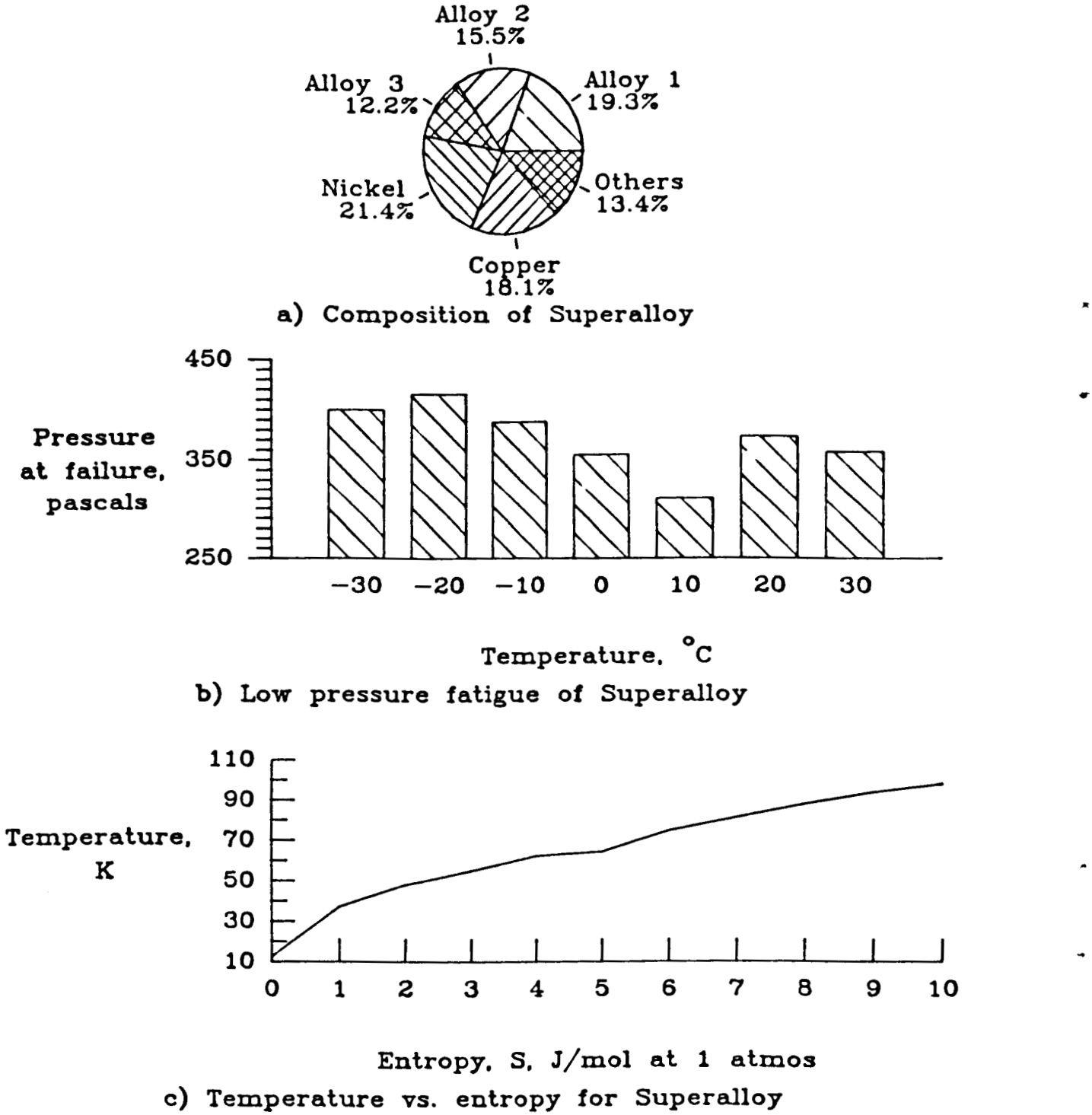


Figure 1-1. Three basic chart types.
Pie chart (top); bar chart (middle); line chart (bottom).

The Low-Level routines, which require the user to have an applied knowledge of DI-3000, allow the user to manipulate chart components, and thereby design specific report-quality charts for unique or unusual purposes. The features at this level can be used alone with the base graphics package, or in conjunction with other packages (e.g., LEZ, etc.) to enhance or augment their capabilities. Since the user is constructing a chart from components, the user has a greater range in the diversity of charts to be generated. However, this diversity is offset by a greater programming effort. It is recommended that the LEZ routines be examined first to determine if the user's application can be accomplished at the highest level possible.

The capabilities of both levels are presented as an application-independent graphics package written in ANSI FORTRAN 77, and currently uses DI-3000 as its underlying graphics package. The library is therefore machine-independent, providing support for centralized and/or distributed computer systems.

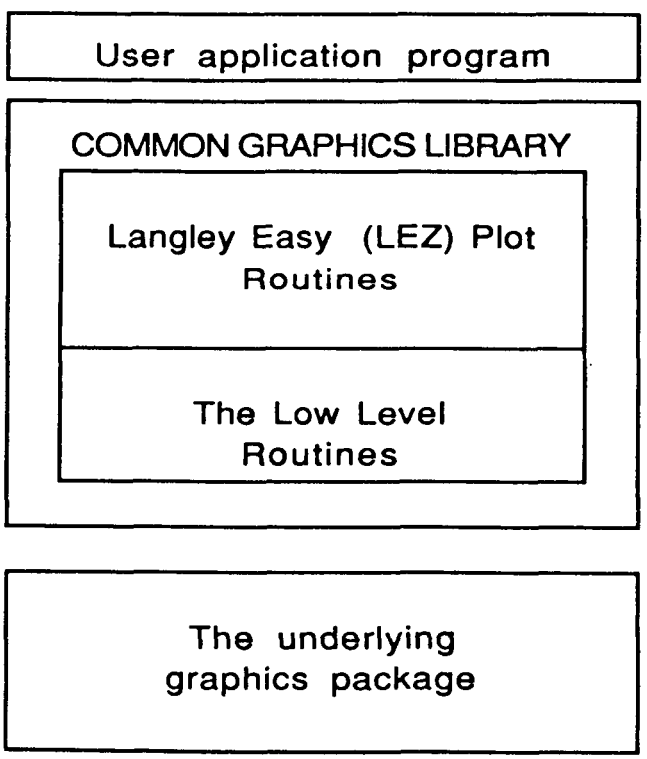


Figure 1-2. Structure of the Common Graphics Library from an application viewpoint.

1.2 Common Graphics Library's Prominence in the New Graphics System (NGS)

The Common Graphics Library is a FORTRAN user-callable library which can be used independently or in conjunction with other graphics packages. This subsection is intended to briefly describe how the Common Graphics Library is related to NGS. The Graphics MINI Manual [G-1] identifies and describes the various graphics software and hardware components, details the interfaces between these components, and provides information concerning the use of these components at LaRC. Figure 1-3, the NGS hierarchy structure, shows the major graphics software packages and hardware devices currently supported.

As previously described, the Low-Level routines of the Common Graphics Library interface with the underlying graphics package directly. The level of sophistication of the Low-Level routines enable the software packages which are functionally higher in the hierarchy to utilize these capabilities. To interface the CGL with other graphics packages, the user must either directly modify the package calling the CGL or use the CGL in conjunction with the calling package adhering to the constraints of both packages.

1.2.1 Underlying Graphics Package

Currently, DI-3000 is the application independent, underlying graphics package serving as a primary component of the CGL. The LEZ routines provide an abstract interface which is "commercial graphics package independent". This implies that the calling sequence and arguments supplied to the LEZ routines do not rely on a specific underlying graphics package. This independence is conducive to future transitions of the LEZ routines from DI-3000 to another underlying graphics package (e.g., GKS).

The Low-Level routines interface with DI-3000 directly, and are therefore "context sensitive" (i.e., must conform to the requirements of "where" and "how" the graphical components can be created and generated). Additionally, the capabilities of the Low-Level routines depend heavily upon the primitives and attributes present in the underlying graphics packages (e.g., color, character, fonts, interim line patterns, etc.).

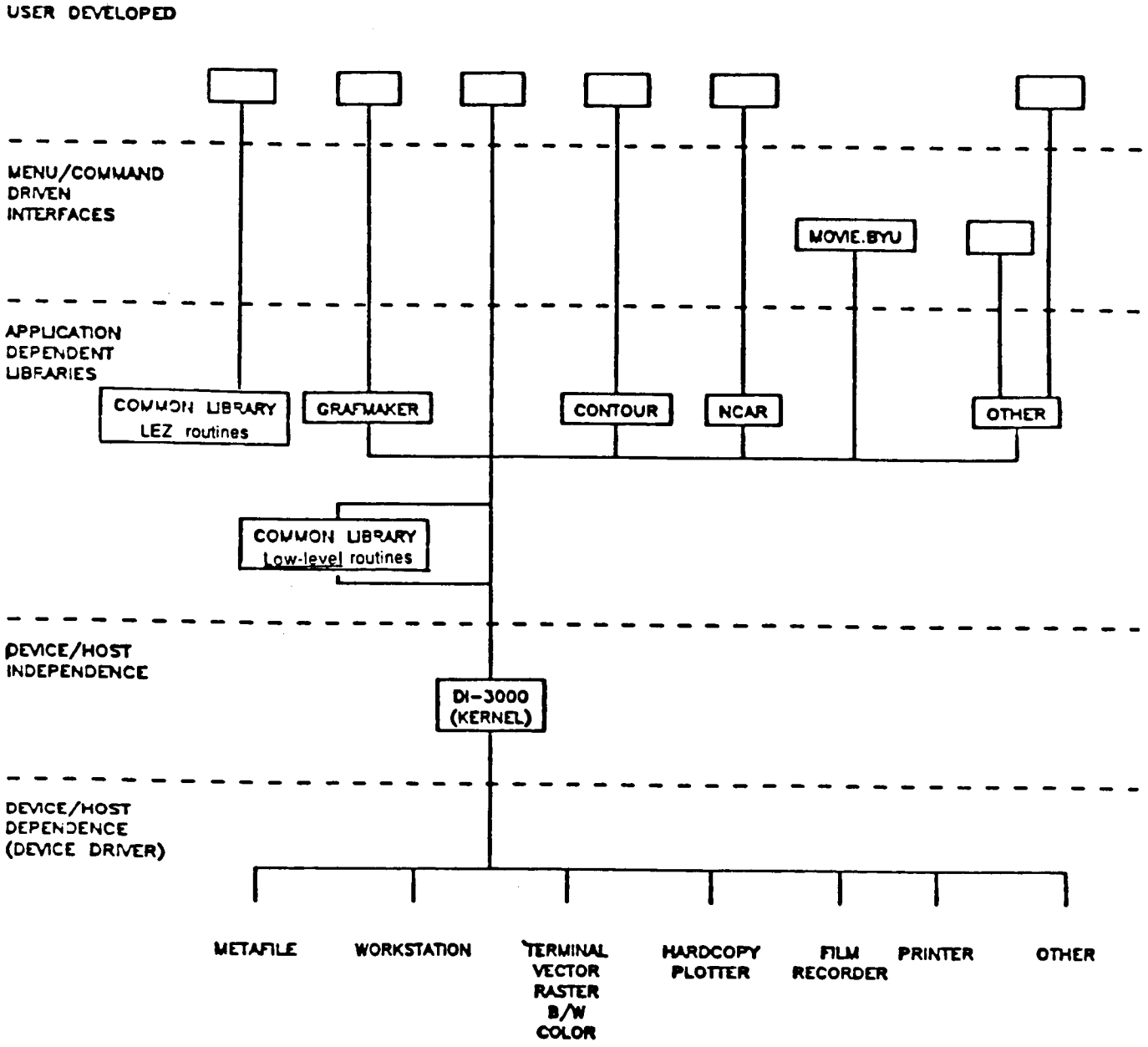


Figure 1-3. NGS hierarchy structure.

1.2.2 Device Drivers

The following is an overview of the NGS device drivers, and how the CGL interfaces with them. The user is referred to the Graphics Mini Manual [G-1] for a more detailed description of the device drivers.

Each of the NGS software components interface to a hardware device through a device driver. A DI-3000 device driver is a collection of device specific subroutines that drive a particular display device. A separate DI-3000 device driver is associated with each graphics output device. A special device driver, the metafile driver, produces a file of device independent graphics information that may be processed by the METAFILE TRANSLATOR [G-6]. The features of each device driver are documented in the Device Driver Guide [G-10]. Although the device driver is linked at load-time, the user must initialize, select, and perform other device action from within the application program. Each device is assigned as an integer number. For load-time device selection, the metafile driver is assigned "0" and any other device is assigned "1".

Since the LEZ routines are not restricted to a specific general purpose graphics package, but provides an abstract interface, the following call selects and initializes the identified device(s):

```
CALL LEZINI (IDEV)
```

where IDEV is the assigned device name (0 - metafile, 1 - Interactive device, 2 - both metafile and interactive device). Note, this routine can be called again to select and deselect specific devices. A complete description of selecting and initializing the underlying graphics package from the LEZ routines can be found in Appendix A [G-12] in the description of LEZINI.

When the Low-Level routines are used solely in conjunction with the underlying graphics package, then the underlying graphics package will provide the means by which the device drivers are initialized and selected. In the case of the Low-Level routines of the CGL, and the underlying graphics package DI-3000, the device driver must be selected and initialized with the following DI-3000 calls

```
CALL JDINIT (IDEV)
CALL JDEVON (IDEV)
```

where IDEV is the assigned device number (0 - metafile, 1 - interactive device). A complete description of selecting and initializing DI-3000 can be found in the DI-3000 User's Guide [G-5].

1.3 LaRC's Unique Plotting Requirements

The graphics requirements at LaRC are very diverse in nature, reflecting the variety of applications (business, CAD/CAM, engineering, and scientific, etc.) present at NASA Langley. The CGL and this manual concentrate on 2D chart generation. These types of charts include: line/logarithmic charts, bar charts, and pie charts. See Section 2 for a description of each chart type.

Unlike a commercially developed general purpose graphics package which is typically designed for the graphics user community at large, the CGL is tailored toward a specific set of restrictions proposed by LaRC. This set of requirements necessitated the development of a Langley specific graphics library, the Common Graphics Library. One of the chief considerations in the design of the CGL was the capability of generating publication quality charts requiring a minimal of effort.

The format of computer-generated charts must conform to various criteria depending on where and how the charts are to be used. Several types of journals and technical papers require that the charts are acceptable to the Langley publication process. The remainder of this subsection will provide only the basic characteristics of this criteria, whereas a complete description can be found in "Guidelines in Preparing Computer-Generated Plots for NASA Technical Reports with the LaRC Graphics Output System" [G-2].

A fundamental intent of the CGL is to provide a set of primitive routines unique to LaRC. These requirements partially consist of: the NASA line patterns, the NASA symbols, and the NASA logo.

The NASA line patterns consist of the following eight line patterns in the following sequence:

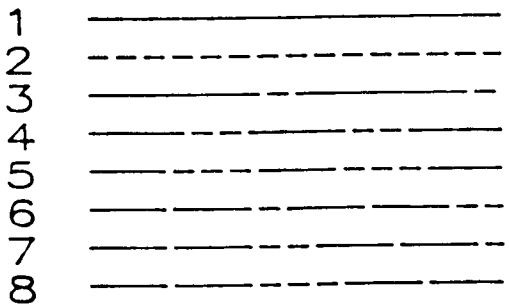


Figure 1-4. NASA standard line patterns.

The NASA symbols consist of the following symbols in the following sequence:

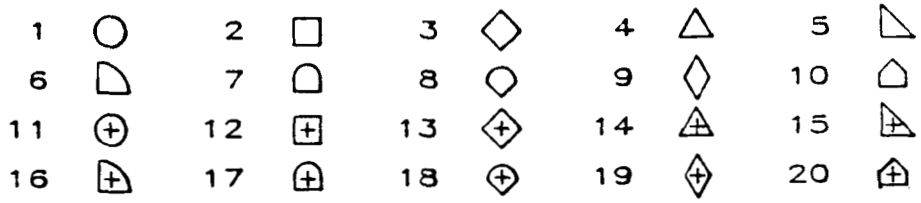


Figure 1-5. NASA standard symbols.

If a flag is desired, add 100, 200, 300, ..., 800 to the base symbol number. Note, symbols one through ten are recommended for publication quality plots.

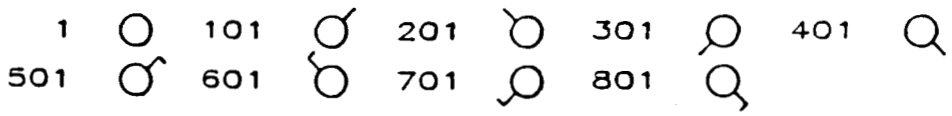


Figure 1-6. NASA standard symbols with flags.

If a solid filled symbol is desired, add 900 to the base symbol number. Note, however, the use of solid filled symbols is not desirable for publication quality charts.

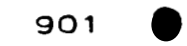


Figure 1-7. NASA standard symbols - solid filled.

The NASA logo is an emblem officially recognized by NASA. The policy governing the use of the NASA logo is documented in the NASA Management Manual (NMI 1020.1G Paragraph 13b).

Figure 1-8. NASA logo.

1.4 Use of the CGL for Publication Purposes

In addition to these primitives, the page layout and the contents of the charts must conform to the criteria determined by the chart's intended audience. For example, the sizes and distances between various chart elements must be sufficient to provide a clear and legible chart.

The intent of the publication quality standard is to ensure that charts will consistently be published legibly with no loss of integrity. As such, a set of publication parameters are defined that control the size of the characters used for labeling, the distances to the labels, the distances between lines and columns, the length of tick marks, and the symbol sizes. Whereas, the use of viewgraph standards ensures that charts being presented to a group will show the data clearly and legibly.

The Common Graphics Library provides a method to obtain publication and viewgraph quality charts. By default, the CGL will conform to publication quality standards by providing:

- appropriate sizes and distances between various chart components;
- data curves using lines and symbols, such that the line does not pass through the symbols;
- text attributes suitable to acceptable publication standards; and
- appropriate distancing between tick mark labels and the axes.

Because the CGL defaults to the use of these NASA standards, the user is required to have a minimum knowledge of the actual specifications needed to obtain these charts. However, the user is responsible for using the library in such a way as to stay within the criteria of the publication process. The user should ensure that:

- user defined text heights and widths are acceptable;
- appropriate capitalization of text conforms to standards;
- numeric tick mark labels have suitable scale factors;
- text does not overwrite other components (and/or text); and
- content is appropriate for the audience's clarity.

1.5 Planning Figures for Publication

It is a common practice that most of the figures in NASA reports are grouped together at the end of the report instead of inserted in the text; therefore, all figures should be as uniform as possible. However, authors are instructed to discuss requirements with the editors if they want figures integrated in the text. The basic rules to follow in planning uniform figures are:

1. The chart perspective should be read from one point; the person reading the charts should not have to rotate the paper to read the charts. The one exception to this rule is a long vertical label.

2. The plot areas should be kept the same size and the charts should be oriented the same throughout the report.

3. The same scale per unit value (scale increment) should be used on similar data, even if the data are in different ranges. For example, a scale increment, for an angle should remain the same even if the range is 0 to 90 for one chart and 90 to 180 for another chart.

4. The vertical figure orientation (see Figure 9) is preferred. The vertical measurement is 9 3/16 inches (233.4 mm.) and the horizontal measurement is 7 1/8 inches (181.0 mm.).

5. The horizontal figure orientation (see Figure 10) is not preferred because the report must be physically rotated to view the figure, but there are times when the data necessitates using the horizontal figure. The horizontal measurement is 9 3/16 inches (233.4 mm.) and the vertical measurement is 7 1/8 inches (181.0 mm.).

6. All labeling within the plot area should be the same size even if the figure is divided into multiple parts. If a figure is divided into multiple parts, the height of the full-size character for labeling is based on the size of the whole figure, not each separate part.

7. Horizontal axis should represent independent data, vertical axis should represent dependent data.

8. Character heights should be 1/4 greater than widths (or conversely width should be 0.8 of height).

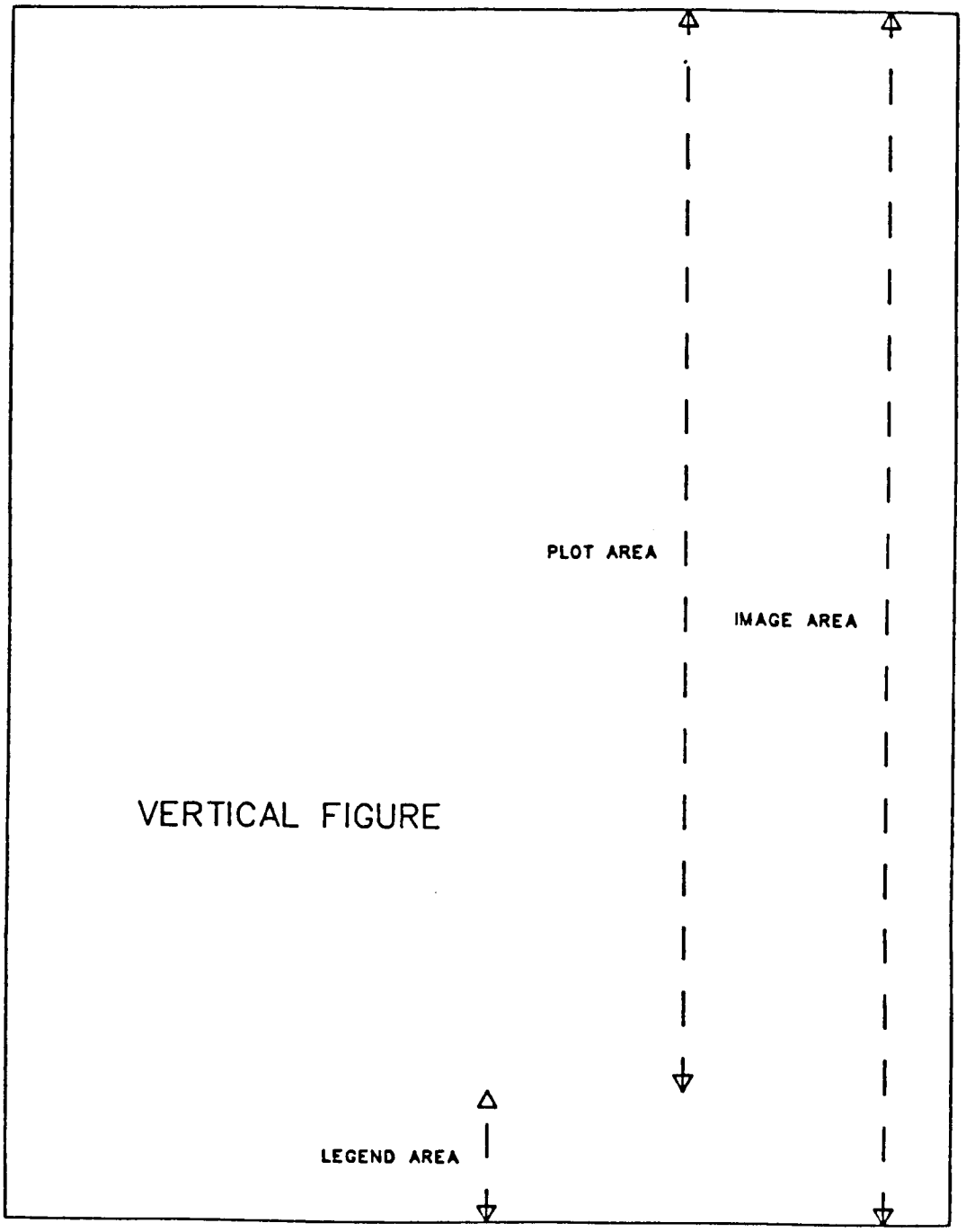


Figure 1-9. Vertical figure orientation.

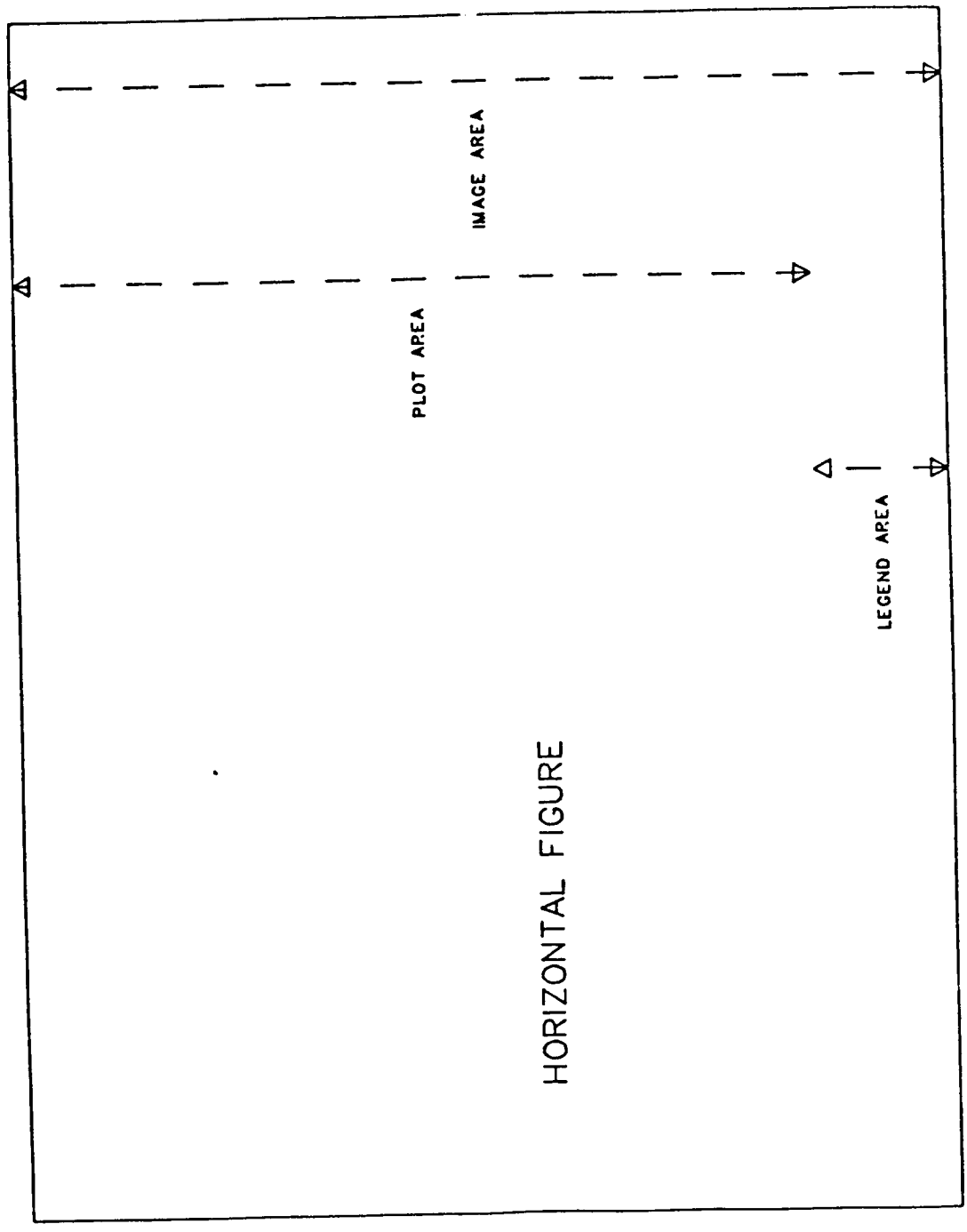


Figure 1-10. Horizontal figure orientation.

2. BASIC CONCEPTS AND TERMINOLOGY

This section provides an overview of the various types of charts and identifies and illustrates their various components. This section is intended as a reference only. Conceptually, this section begins with the more abstract concepts and breaks them into smaller, more refined elements.

2.1 Terms 'Chart' and 'Plot'

The terms chart and plot are often used interchangeably, but have subtle differences depending on the context in which they are used.

chart - a diagram (a series of one or more symbols, lines, line segments, curves, tabular data, or areas) that represents the variation of a variable in comparison with that of one or more other variables. Charts which depict linear data are referred to as "line charts" or "linear charts". Charts which depict logarithmic data are referred to as "log charts" or "logarithmic charts". Charts which use "bars" are referred to as "bar charts", while plots which use "pie segments" are referred to as "pie charts". There are three basic chart types: linear/logarithmic charts, bar charts, and pie charts (see Figure 1-1). Each type is explained in a following subsection, and concepts specific to each are detailed.

plot - one or more charts on a single display area (terminal screen, graphics frame, etc.). Thus, "plot" is used interchangeably in this document as a generic term for "chart".

composite chart - a collection of two or more charts on a single display area (terminal screen, graphics frame, etc.).

2.2 Coordinate Systems

A fundamental approach in chart generation using the CGL routines is to conceptually distinguish between entities framing the data (e.g., axes, text, etc.), and the data itself. To facilitate chart generation, two independent windows are typically established, and are referred to as page coordinates and data coordinates.

page coordinates - The CGL routines allow the user to define a coordinate system in which to describe all charts and their components. This system provides a unit of measure allowing the user to establish lengths and distances (e.g., axis length, axis positioning, etc.). This coordinate system is referred to as page coordinates (or page space). The units used to describe the page boundaries can be determined by the user, and can represent any unit of measure the user wishes. If the user is going to send the charts to a postprocessing device, a compatible unit of measure is desirable (e.g., inches or millimeters).

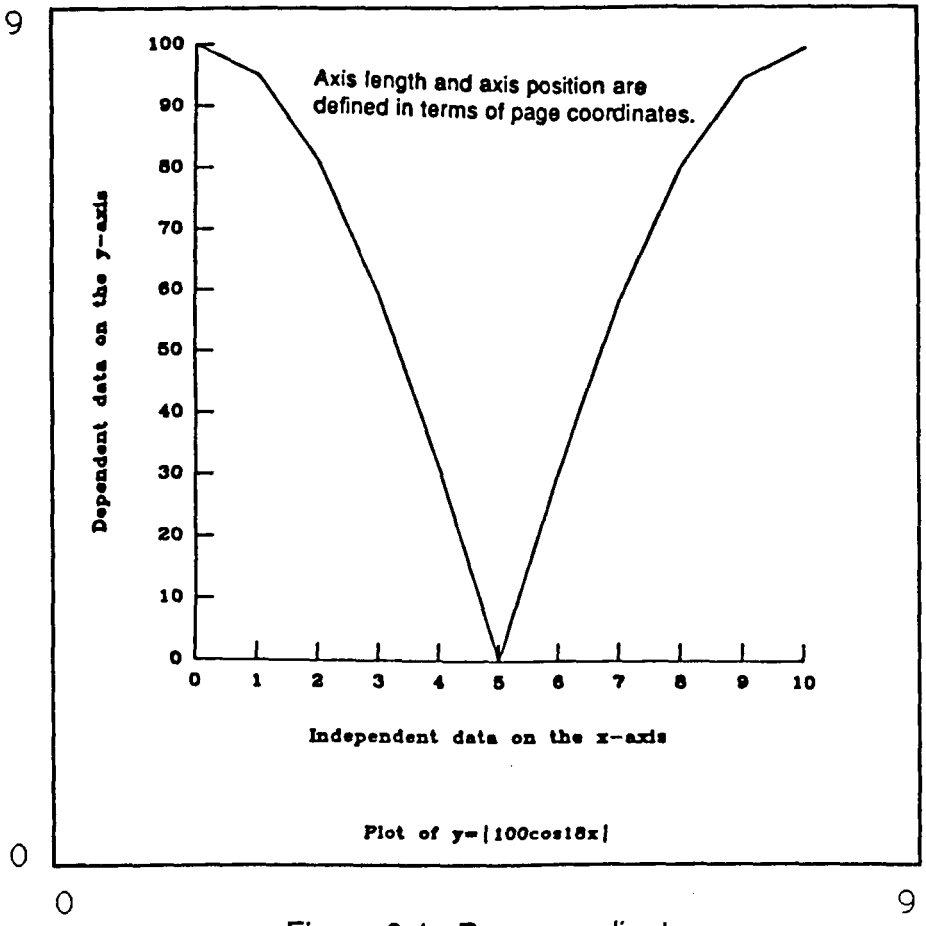
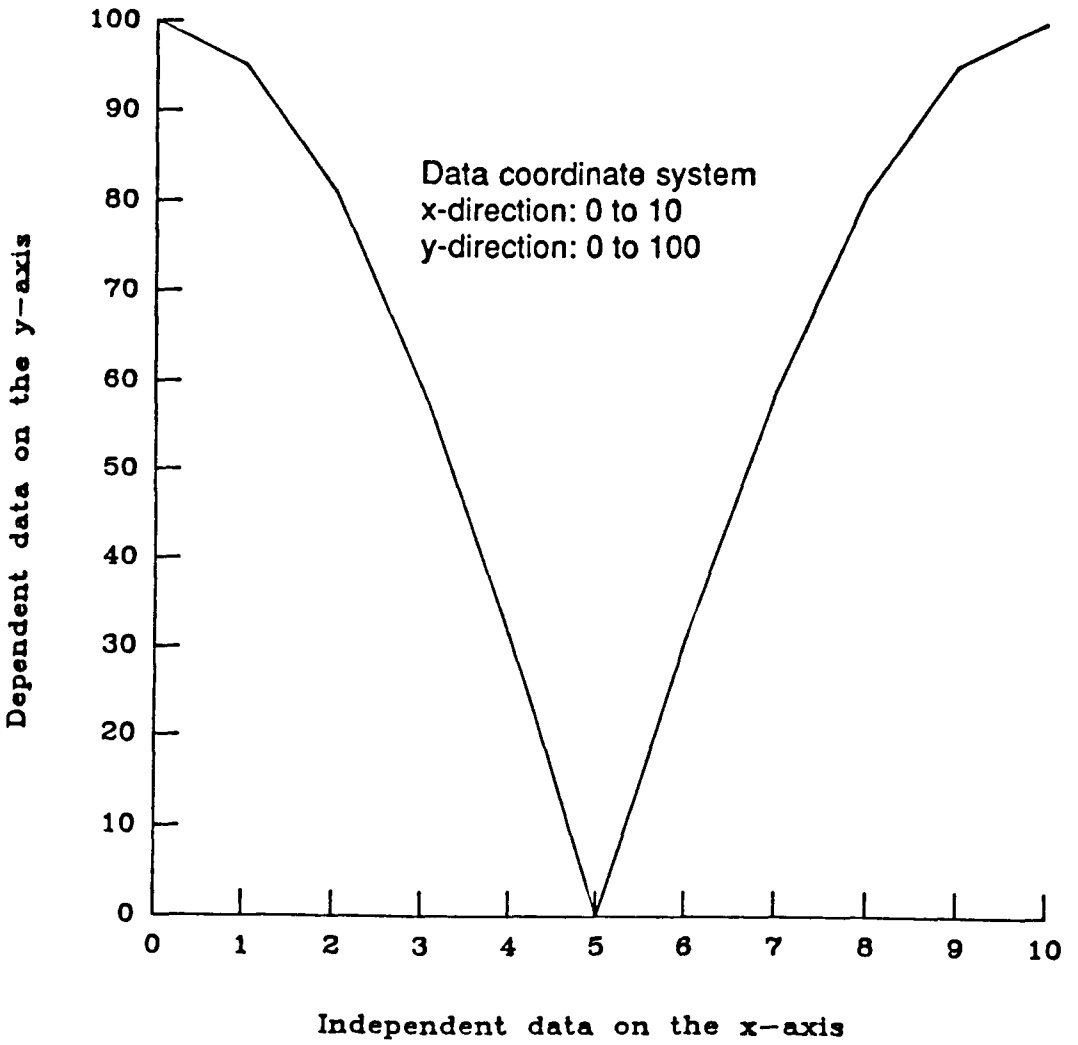


Figure 2-1. Page coordinates.

data coordinates (or data space) - The CGL routines allow the user to define a coordinate system in which the data is to be plotted. The user controls the range and values of the data coordinate system through calls to the appropriate routines. The data coordinates only affect how and where the data is to be plotted on the page. If data to be plotted lies outside the data space, then those data values may optionally be clipped (i.e., not plotted). The location of the data coordinate system is typically determined by the boundaries formed by a set of axes (two or more).



Plot of y=|100cos18x|

Figure 2-2. Data coordinates.

2.3 Linear/Logarithmic Charts

A linear/logarithmic chart is a diagram which uses a series of one or more symbols, lines, line segments, and/or a combination of these to represent data. For a linear chart, the scale factor (i.e., the change in values between major tick marks) progresses in a linear fashion. For a logarithmic chart, the scale factor progresses in a logarithmic fashion.

Charts are further classified based on the scale factors for the dependent and independent axes:

linear chart	-	both axes are linear
logarithmic chart	-	both axes are logarithmic
semi-log chart	-	one axis is linear and logarithmic.

The CGL line plotting routines will generate publication and/or viewgraph line charts with the NASA line patterns and symbols. When a data curve is represented with both lines and symbols, the CGL routines will display the data with the line not passing through the symbol. The user may plot linear, logarithmic, or semi-log charts. The CGL line routines offer the capability of generating a "key", which provides additional information about the data in the chart. By the use of unique line patterns and symbols, it is possible to plot a family of curves which can clearly be seen either on a color viewgraph, or on a black and white printed report.

1) horizontal axis -
the axis parallel to the bottom of the figure orientation when viewing the page from the intended perspective. The horizontal axis usually represents the independent data.

2) vertical axis -
the axis parallel to the side of the figure orientation when viewing the page from the intended perspective. The vertical axis usually represents the dependent data.

3) major tick mark -
short lines at specified intervals perpendicular to the axis being denoted.

4) major tick mark labels -
a character or numeric description associated with a major tick mark. Often referred to as "tick mark labels".

5) minor tick marks -
short lines (preferably shorter than major tick marks) between major tick marks perpendicular to the axis being denoted.

6) scale factor -
the change in value between numeric major tick marks.

7) axis label -
a text string describing the properties associated with an axis.

8) key -
a block of information placed within the page boundaries, explaining any codes or symbols used on the chart.

9) key entry -
a line consisting of a description associated with a symbol and/or line.

10) caption -
the figure number and title. Often referred to as a chart title.

The following figure illustrates the various components of linear/logarithmic charts.

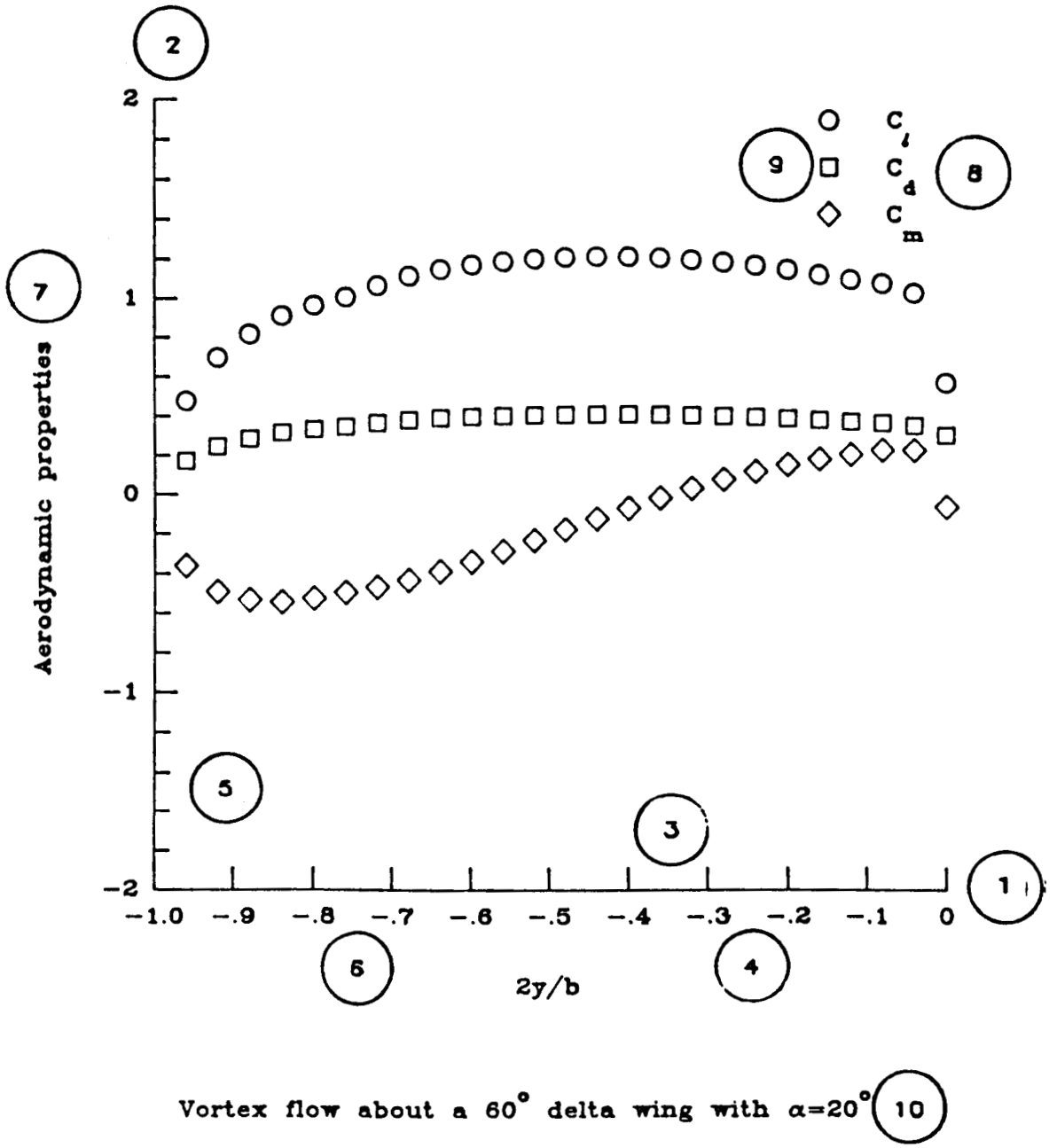


Figure 2-3. Linear/logarithmic chart components.

2.4 Bar Charts

A bar chart is a diagram which uses a series of rectangles. The height of each rectangle represents a dependent value at an associated independent position.

The charts may be represented in an absolute manner (side by side) contrasting the data distinctly. The charts may be represented in an additive manner (stacked) showing their cumulative properties. A key option can be used to provide additional information concerning multiple data sets.

1) horizontal axis -
   the axis parallel to the bottom of the figure orientation when viewing the page from the intended perspective. The horizontal axis usually represents the independent data.

2) vertical axis -
   the axis parallel to the side of the figure orientation when viewing the page from the intended perspective. The vertical axis usually represents the dependent data.

3) major tick mark -
   short lines (preferably shorter than major tick marks) between major tick marks perpendicular to the axis being denoted.

4) major tick mark labels -
   a character or numeric description associated with a major tick mark. Often referred to as "tick mark labels".

5) minor tick marks -
   short lines between major tick marks perpendicular to the axis being denoted.

6) scale factor -
   the change in value between numeric major tick marks.

7) axis label -
   a text string describing the properties associated with an axis.

8) key -
   a block of information placed within the page boundaries, explaining any codes or symbols used on the chart.

9) key entry -
a line consisting of a description associated with a symbol and/or line.

10) caption -
the figure number and title. Often referred to as a chart title.

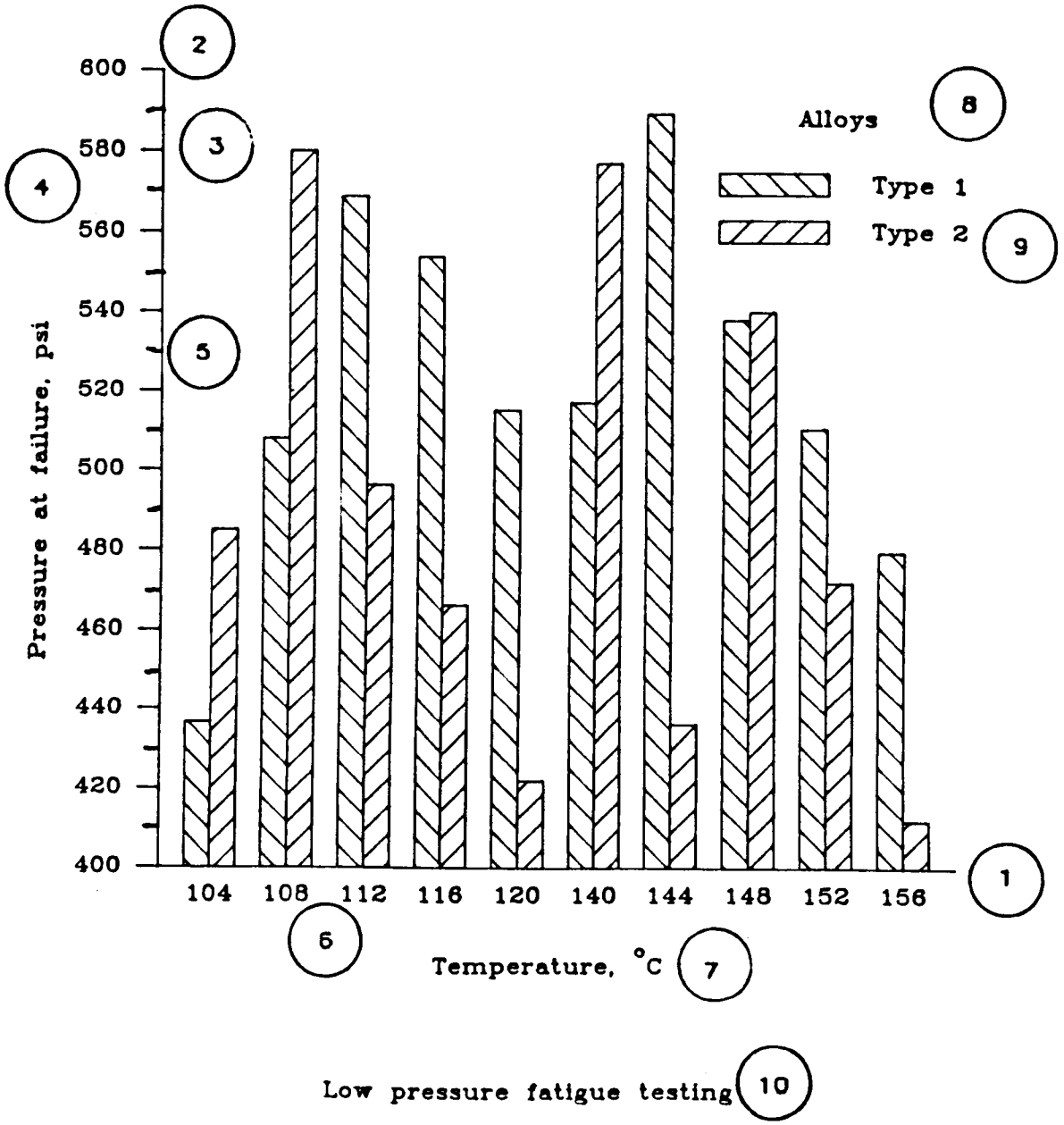


Figure 2-4. Bar chart components.

2.5 Pie Charts

A pie chart is a diagram consisting of a circle which is divided by radii, called pie segments, each of which represent a part of a whole entity. The following figure illustrates the various components of a pie chart. Pies are most often used in showing the "percent of" or "composition of" a particular item.

The LEZ routines generates complete pie charts. Although there are no Low-Level pie routines, this type of chart is presented in the Low-Level document because it is often requested, and can easily be generated from the underlying graphics package and the Low-Level routines.

1) radius -
   the length of a line segment extending from the center of a pie segment to the pie segment circumference.

2) origin -
   the center of the pie.

3) pie segment -
   a wedge-shaped polygon representing a percentage of the entire pie.

4) pie segment text label -
   a description associated with a pie segment.

5) pie segment quantity label -
   the actual value of which the segment represents.

6) pie segment percentage label -
   the percentage of the pie which the segment denotes.

7) key -
   a block of information placed within the page boundaries, explaining any codes or symbols used on the chart.

8) key entry -
   a line consisting of a description associated with a symbol and/or line.

9) caption -
   the figure number and title. Often referred to as a chart title.

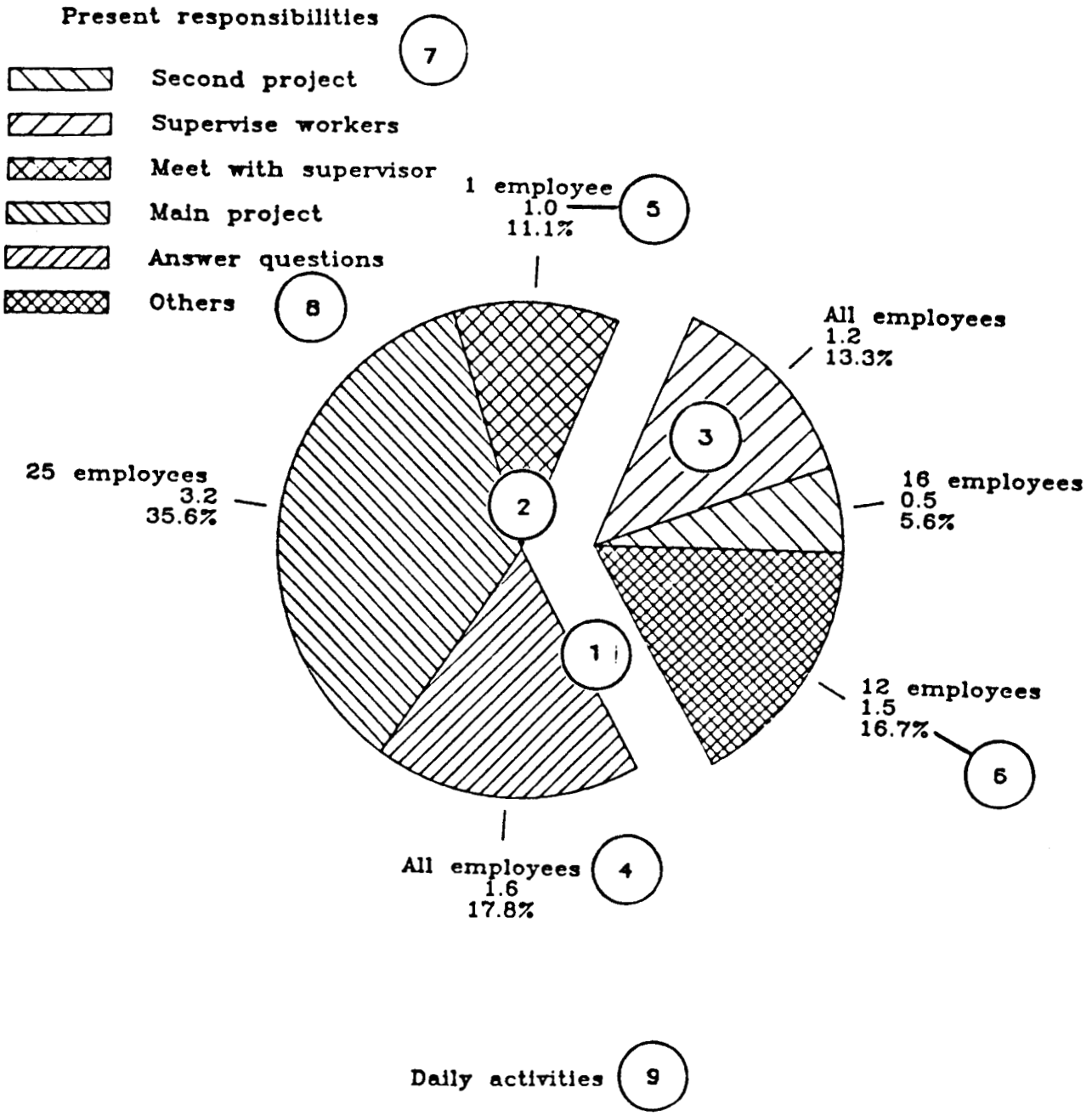


Figure 2-5. Pie chart components.

2.6 Composite Charts

The CGL routines can easily generate multiple charts per page, hereafter referred to as **composite charts.** The number and type of charts that can be placed on a page is limited only by the user-definable page size.

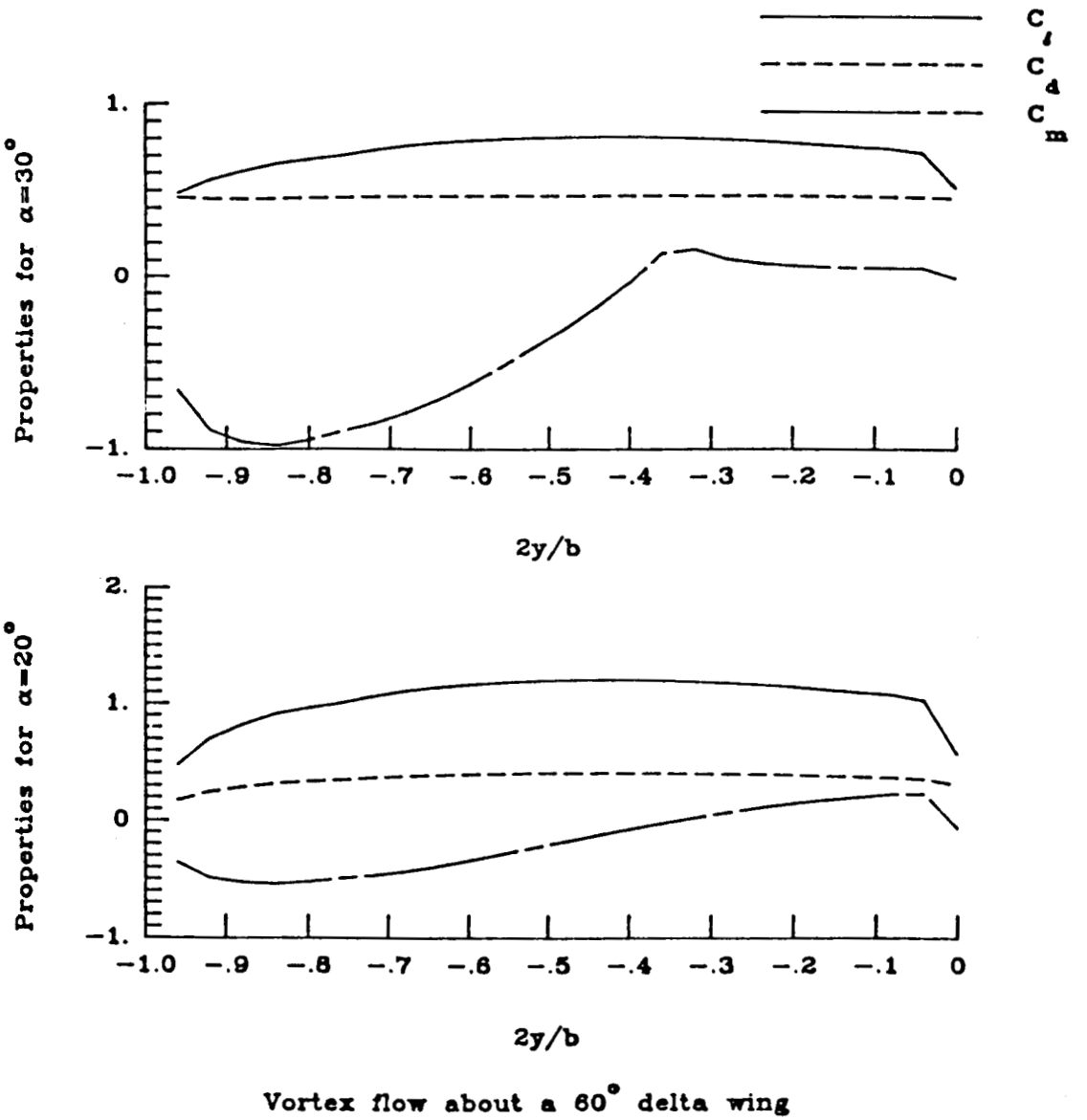


Figure 2-6. Example of a composite chart.

3. THE LOW-LEVEL ROUTINES

This section focuses on the prerequisites and criteria needed to use the Low-Level routines. Since the CGL is written in ANSI FORTRAN 77, resides on top of the general purpose graphics package, and has no machine dependencies, the following section applies to all machines. However, the method of linking this library during execution time, is system dependent, and can be found in Appendix D.

3.1 Introduction

The Low-Level routines are a group of ANSI FORTRAN 77 routines, which utilize the graphics primitives of the underlying graphics package. A graphics routine call will either cause graphical operations to be performed (i.e., move, draw, etc), or will modify graphical attributes (i.e., color, text size, etc.). The CGL includes both the manipulation of graphical output, and the setting of environmental attributes, in addition to its own graphical operations.

The generation of a chart typically consists of displaying data in one or more regions associated by a set of scales (i.e., axes, grids, etc.). Subsection 3.3 will describe several techniques of establishing several regions, and a method to conveniently display the associated data sets. But, in order to use the Low-Level routines it is necessary to understand how these routines interface with the underlying graphics package. Note, the Section 3.2 will provide an overview of the characteristics of the graphical interface, Section 3.3 provides a step by step basis demonstrates these points while generating a line chart.

3.2 Underlying Graphics Package Characteristics used by the Low-Level Routines (using DI-3000 as an Example)

The Low-Level routines assume the user is familiar with DI-3000 or a similar general purpose graphics package (e.g., GKS). This section will describe how the Low-Level routines interface with the underlying graphics package. Appendix C contains a list of all DI-3000 routines with a brief description of their purpose.

The basic skeleton of a graphics program is outlined in Figure 3-1, and is described in the DI-3000 User's Guide [G-5]. The three major components consist of a graphical environment, coordinate systems, and graphical primitives and attributes. The Low-Level routines provide an additional set of graphical output capabilities with associated attributes. These graphical

capabilities (e.g., axes, grids, etc.), although not themselves primitives (i.e., move, draw, etc.) are built using the output primitives of the underlying graphics package.

Most DI-3000 programs have the same basic program structure as shown below. The following figure comes from the DI-3000 User's Guide [G-5], and is described there in more detail.

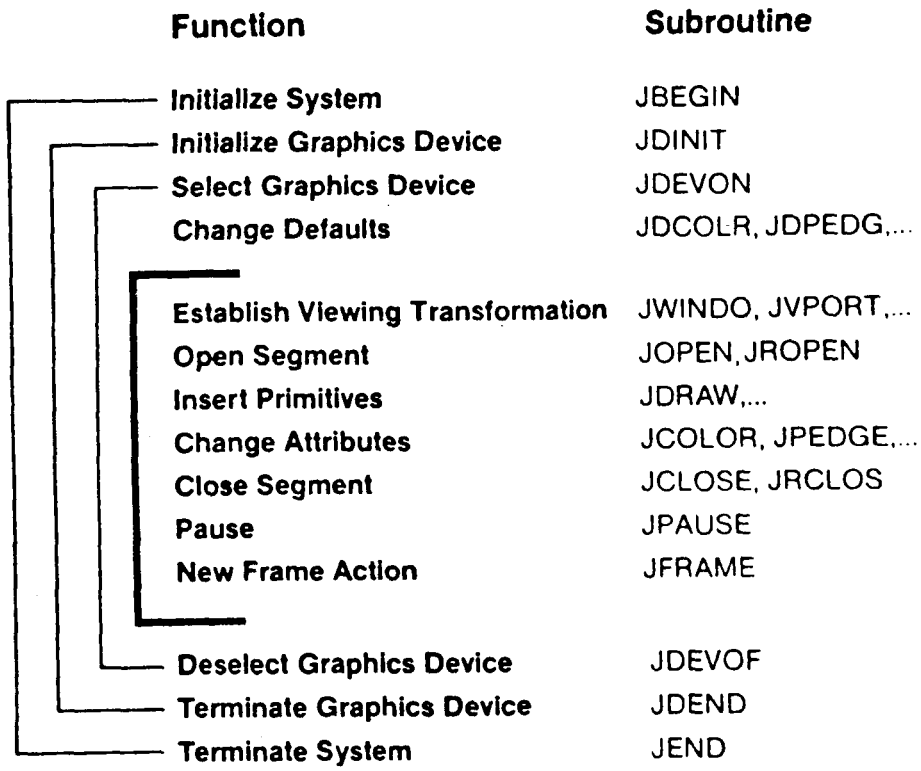


Figure 3-1. DI-3000 basic skeleton program.

3.2.1 Graphical Environment

The graphical environment consists of the establishment (i.e., initialization and termination) of the graphics package and the graphics devices, in addition to providing an appropriate graphical context necessary to use the Low-Level routines.

Unlike the CGL LEZ routines which provide a high-level abstract user interface incorporating the establishment of the graphical environment, the Low-Level routines must be called after the user has ensured the appropriate graphical environment has been established. The Low-Level routines only augment the existing graphics package capabilities.

The first Low-Level routine to be called must be CBEGIN. This routine initializes all CGL related variables. Each variable has an associated default which is assumes when CBEGIN is called. The invocation of CBEGIN is functionally independent of DI-3000, and can be called anytime at any place. However, subsequent invocations will reset any user defined CGL attributes.

```
CALL CBEGIN
```

The majority of the Low-Level routines must be called at a point which is appropriate to their function. CBEGIN initializes CGL variables and can be called anytime; whereas Low-Level routines which perform graphical operations (i.e., output, attribute setting, etc.) must be called in context when graphical output can be generated. In DI-3000, this context is referred to as an opened segment. Figure 3-2, described in the DI-3000 User's Guide[G-5], shows a DI-3000 skeleton program necessary to establish the graphical environment.

Within this framework a combination of DI-3000 based packages may be used. For example, DI-3000 primitives and CGL primitives could be invoked independently, or in combination. Other DI-3000 graphics packages, such as GRAFEASY or GRAFMAKER, may already incorporate parts of the graphical environment into its structure, such as initialization/termination, or the opening/closing of segments. Thus, each package's environment may vary. A more detailed discussion of how to intermix other graphics packages with the Low-Level routines can be found in Section 5.

```
CALL JBEGIN                 • Initialize DI-3000.
CALL JDINIT   (1)           • Initialize device number 1.
CALL JDEVON   (1)           • Activate device number 1.
      •                     •
      •                     •   (Change default values.)
      •                     •
CALL JWINDO   (...)         • Set the window, bordering the objects to be viewed.
CALL JVPORT   (...)         • Set the viewport on the display device.
CALL JOPEN or JROPEN        • Open a segment.
      •                     •
      •                     •   (Graphics output primitives.)
      •                     •
CALL JCLOSE or JRCLOS       • Close the segment.
                            •
CALL JPAUSE   (1)           • Wait for user response.
                            •
CALL JFRAME                 • Cause a new frame action.
                            •
CALL JDEVOF   (1)           • Turn off device number 1.
CALL JDEND    (1)           • Terminate device number 1.
CALL JEND                   • Terminate DI-3000.
```

Figure 3-2. DI-3000 skeleton program.

3.2.2 Coordinate Systems

Once the appropriate graphical environment and associated primitive attributes have been established, then the graphical output routines can be invoked. Note, the graphical output will be generated to the current segment which must be opened prior by the user.

Additionally, the positioning and sizing of the graphical output generated will be in accordance with the current coordinate system established (i.e., world and virtual coordinate systems). The routine CVSPAC determines the largest possible viewspace, based on the width and height passed, and calls JVSPAC accordingly. Most notably, this routine sets an internal variable (VUINCH), which is the relationship between virtual coordinates and page coordinates.

Each CGL graphical routine has a unique set of routine arguments and attributes. For example, Figure 3-3 illustrates the invocation of the routine CNASA. This routine will use the current position (CP) of the present world coordinate system and the current text justification (JJUST) to determine the current point's relationship within the resultant logo. Figure 3-3 shows a partial program using the current coordinate system and the CGL.

```
      PROGRAM ...
C-----------------------------------------------------------------------
C SIMPLE PROGRAM TO DEMONSTRATE THE USE OF A COORDINATE SYSTEM
C-----------------------------------------------------------------------
C INITIALIZE DI-3000
      CALL JBEGIN
C-----------------------------------------------------------------------
C THIS MUST BE THE FIRST CGL CALL TO INITIALIZE THE CGL
      CALL CBEGIN
C-----------------------------------------------------------------------
C WRITE TO THE IDEV DEVICE
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET UP THE VIEWING WINDOW
      CALL JWINDO(0.,11.,0.,11.)
C-----------------------------------------------------------------------
C OPEN CURRENT SEGMENT
      CALL JOPEN
C-----------------------------------------------------------------------
C A LOOP WHICH PRINTS NASA LOGO USING TEXT JUSTIFICATION (JJUST), AND
C WORLD COORINDATES (JMOVE).
      DO 1 I=1,10
        CALL JJUST(I,1)
        CALL JMOVE(REAL(I),1.)
 1    CALL CNASA(0.,0.3,0)
C-----------------------------------------------------------------------
C ENDS ALL GRAPGICS AND CLOSES CURRENT SEGMENT
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

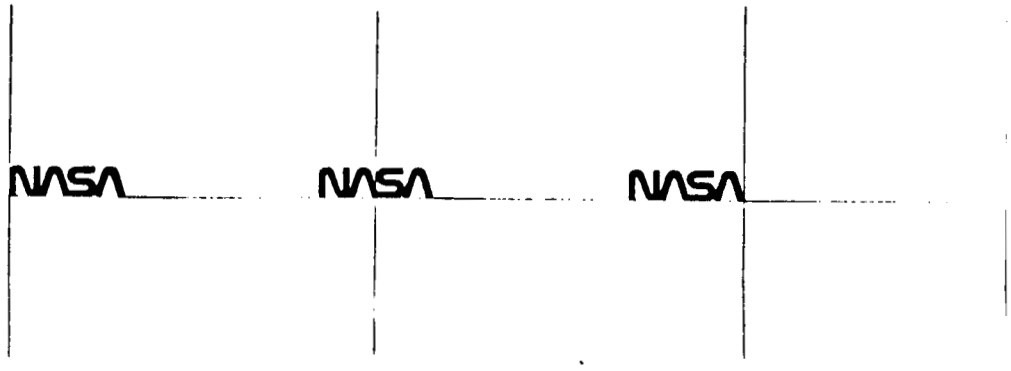


Figure 3-3. Example of the use of a coordinate system.

3.2.3 Page Coordinates and Describing Components

A convenient approach to aid the user in designing a chart is to establish a coordinate system, referred to here as page coordinates, which easily enables the user to arrange and describe the chart components. The page coordinates can be of any dimension and represent any unit. Since publication-quality charts are often requested, the page coordinates described in the document are typically in terms of inches to coincide with the postprocessors (plotters).

A call to CVSPAC will set up the viewspace and viewport, and set appropriate internal CGL attributes. For example, the following calls will establish a square viewspace, and internally determine how virtual coordinates relate to the associated page coordinates.

```
CALL CVSPAC(9.,9.)
```

In this example, the viewspace and viewport will be set to the largest possible square on the device. The internal relationship between page coordinates and virtual coordinates (determined by CVSPAC) is the extent of the viewspace to a 9 by 9 region. In DI-3000, the viewspace boundaries are -1 to 1, thus the relationship is 2/9.

```
CALL JWINDO(0.,9.,0.,9.)
```

A call to JWINDO(0.,9.,0.,9.) will establish the current window to match the virtual coordination system, and the internal relationship. Note, it is important to have the same aspect ratio for the window (JWINDO) and viewport (CVSPAC) to avoid distortion.

Once the page coordinates have been recognized, the user can position and describe objects in terms of the page coordinates. For example, the following code will output a graphics string as a title, and draw a set of axes to illustrate positioning and sizing (as shown in Figure 3-6).

```
C SET VIEWSPACE AND WINDOW
    CALL CVSPAC(9.,9.)
    CALL JWINDO(0.,9.,0.,9.)
C OPEN A SEGMENT
    CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
    CALL JSIZE(.2,.2*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,CENTER)
    CALL JMOVE(4.5,YORG+YLEN+.5)
    CALL JJUST(2,2)
C OUTPUT THE STRING
    CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
    CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
    CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
    CALL CHLAB('HORIZONTAL AXIS',1)
    CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
    CALL CVLAB('VERTICAL AXIS',1)
    CALL CVPREC(3)
    CALL CVAXIS(.0,.01,.001,YLEN)
```

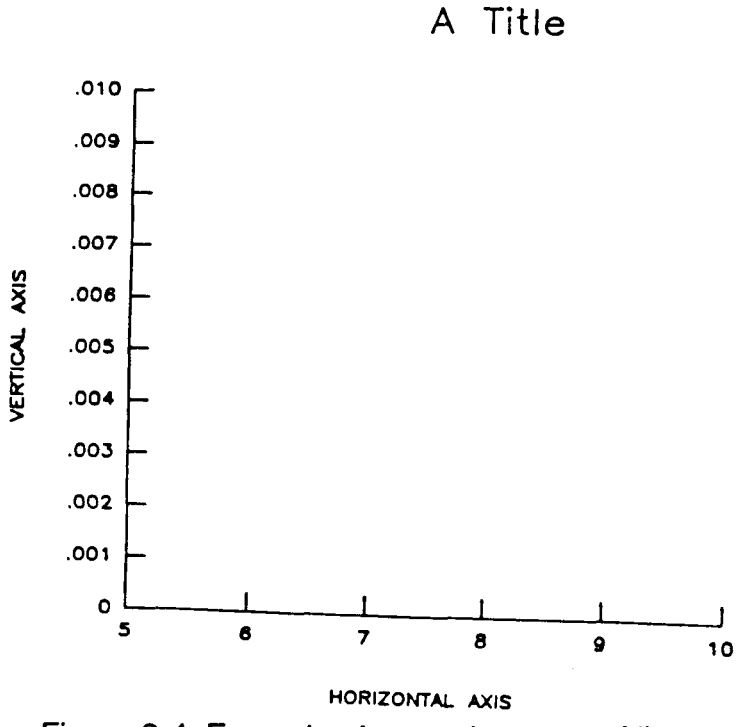


Figure 3-4. Example of outputting a set of linear axes.

3.2.4 Graphics Primitives and Attributes

Each type of output primitive has an associated set of primitive attributes. Attributes determine the appearance and characteristics of the graphics output. For example, drawing attributes involve such factors as line attributes (i.e., line color, line width, etc.), marker attributes (i.e., symbol number, symbol size, etc.), polygon attributes (i.e., interior color, interior pattern, etc.), and text attributes (i.e., text font, size, and justification).

The CGL uses a combination of underlying attributes and CGL attributes associated with a particular output primitive. The specific attributes required for each CGL routine are identified in Appendix A. Thus the user must coordinate and keep track of both sets of attributes which affect the graphical output. Figure 3-4 shows a partial program using DI-3000 and the CGL primitives and attributes.

```
                    •
                    •
C Set the world coordinates to describe page components
          CALL JWINDO(0.,11.,0.,11.)
C Open the window
          CALL JOPEN
C Set DI-3000 attributes
          CALL JPINTR(1)
C Set CGL attributes
          CALL CSYMSZ(0.5)
          DO 1 I=1,10
C Set the symbol number
              CALL CSYMNO(I)
C Call the CGL routine to draw a symbol at coorindates (I,1.)
    1     CALL CPNTPT(REAL(I),1.)
                    •
                    •
```

Figure 3-5. Example of the use of graphics primitives and attributes.

Each CGL attribute is set to a default when CBEGIN is called. To assign another value to a CGL attribute, the user should call the specific routine associated with the attribute. Most CGL attributes begin with "N" (e.g., NHBTIC, NHLABJ, NSYMNO, NSYMSZ, NLNPAT, etc.), and can be assigned through a corresponding "C" routine (e.g., CHBTIC, CHLABJ, CSYMNO, CSYMSZ, CLNPAT, etc.).

The CGL attribute values will retain their initial values until overwritten by the user. All attribute values will then retain these new values unless subsequently changed. Note, reinvoking CBEGIN will set all CGL attributes to their default values. Figure 3-5 shows a partial program setting and resetting CGL attributes.

```
      .
      .
C INITIALIZE CGL ATTRIBUTES TO DEFAULTS AND SET CURRENT WINDOW
      CALL CBEGIN
      CALL JWINDO(0.,11.,0.,11.)
C OPEN CURRENT SEGMENT
      CALL JOPEN
C OUTPUT SYMBOL (WITH DEFAULT SYMBOL NUMBER AND SYMBOL SIZE)
      CALL CPNTPT(1.,1.)
C CHANGE SYMBOL NUMBER TO 3, SYMBOL SIZE TO .5, AND DISPLAY
      CALL CSYMNO(3)
      CALL CSYMSZ(.5)
      CALL CPNTPT(2.,1.)
C RESET CGL ATTRIBUTES BACK TO DEFAULTS
      CALL CBEGIN
      CALL CPNTPT(3.,1.)
C END ALL GRAPHICS AND CLOSE CURRENT SEGEMENT
      CALL JPAUSE(IDEV)
      .
      .
```

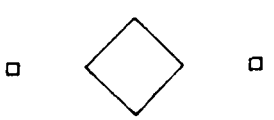


Figure 3-6. Setting and resetting CGL attributes.

3.3 Steps in the Generation of Linear and Logarithmic Line Charts

This section enumerates and describes the steps needed to generate an XY line chart. These steps are explained separately, then combined and illustrated as a complete program generating a complete chart. As previously mentioned, the three primary ingredients are:

- establishing a graphical environment;
- setting the world coordinate systems; and
- calling the graphical primitives after setting their corresponding attributes.

This section describes each of these ingredients, and walks through the major steps of generating an XY line chart. This section naturally progresses from a simple XY line chart with a single data set by adding multiple data sets, including a key, and showing how to generate publication-quality axes. Subsequent sections address the topics of logarithmic and semi-log axes, and grids.

3.3.1 Plotting Linear Data

Typically, the range of the data coordinates will not coincide with the page coordinates. In the previous example, the page coordinates were defined to be 9 by 9-units, whereas the data ranged from 5 to 10 in the x-direction, and 0.0 to 0.01 in the y-direction.

In order to plot the linear data within the area bounded by the axes, the data coordinates will need to be mapped onto the page bounded by the axes. A convenient approach is to save off the virtual coordinates of the axes boundaries described in terms of page coordinates. This is accomplished by two calls to JCONWV, one for the lower-left intersection of the axes (i.e., the point denoted XORG,YORG), the other for the upper-right boundary of the axes (i.e., the point denoted XORG+XLEN, YORG+YLEN).

```
C XORG,YORG    - THE INTERSECTION OF THE AXES
C VX1,VY1      - THE VIRTUAL COORDINATES OF XORG,YORG
     CALL JCONWV(XORG,YORG,0.,VX1,VY1)
C XLEN, YLEN   -  THE LENGTHS OF THE AXES
C VX2,VY2      -  THE VIRTUAL COORDINATES OF THE OPPOSITE
C                 DIAGONAL OF THE AXES
     CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
```

Next, in order to plot the linear data, which has different scales than the page coordinates (i.e., data versus page coordinates), the user must close the segment, set the new world coordinates to match the data extremes, then set the viewport to match the virtual coordinates corresponding to axes boundaries inquired above. In essence, we are mapping the data coordinate system onto the virtual coordinates bounded by the axes. Since the data and world coordinates are linear, no additional conversion is needed to plot the data.

```
                    •
                    •
                    •
            CALL JCLOSE
            CALL JWINDO(5.,10.,0.,.01)
C 5.,10. - X-DIRECTION DATA COORDINATES
C 0.,.01 - Y-DIRECTION DATA COORDINATES
            CALL JVPORT(VX1,VX2,VY1,VY2)
                    •
                    •
                    •
```

In case the data exceeds the current world coordinates, set clipping "on" (.TRUE.) to suppress unwanted data.

```
            CALL JWCLIP(.TRUE.)
```

Finally, open the segment, set the desired plotting attributes, and plot the data. The routine CLNPLT will plot an array of values in the current window. Thus, the window must be set to match or coincide with the data. The appearance of the points to be plotted will be determined by the attributes associated with CLNPLT. A few notable attributes are: NSYMNO (symbol number), NSYMSZ (symbol size), and NLNPAT (line pattern). After plotting the data, the window for the data can be closed.

```
            CALL JOPEN
            CALL CSYMNO(1)
            CALL CLNPLT(X,Y,NPTS)
            CALL JCLOSE
```

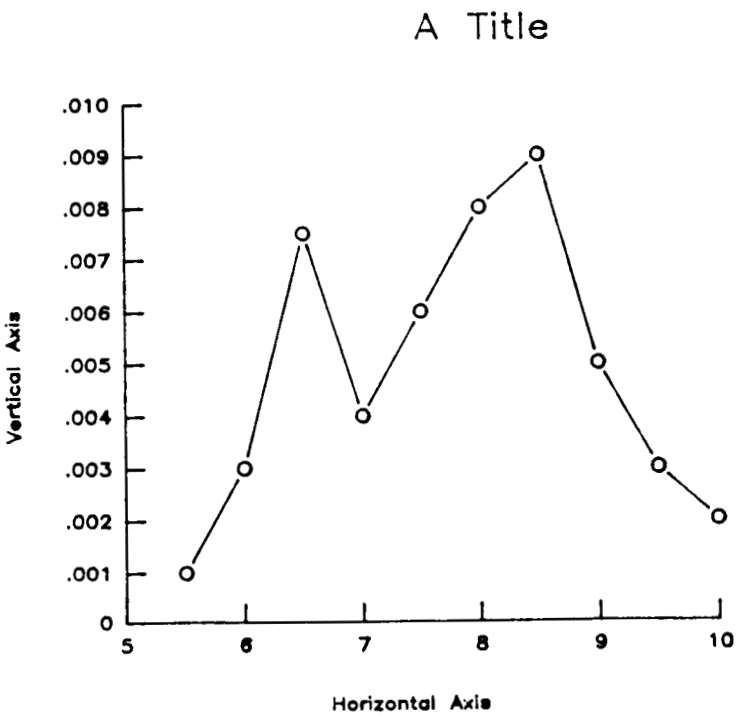


Figure 3-7. Example of plotting data in a linear chart.

3.3.2 Complete Program Plotting Multiple Linear Data Sets

The routine CLNPLT must be invoked for every data set. Thus for multiple data sets, CLNPLT will be called more than once. To distinguish between the data sets, the attributes controlling the data's appearance should be altered.

```
    CALL JOPEN
C Plot the first data set with circles and solid lines
    CALL CSYMNO(1)
    CALL CLNPAT(1)
    CALL CLNPLT(X,Y1,NPTS)
C Plot the second data set with squares and dashed lines
    CALL CSYMNO(2)
    CALL CLNPAT(2)
    CALL CLNPLT(X,Y2,NPTS)
```

The following is a complete program combining the method previously described in context. The corresponding graphics output can be found in Figure 3-8.

```
      PROGRAM CDR5
C-----------------------------------------------------------------------
C  COMPLETE LINE CHART
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
```

```
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

3.3.3 Adding a Key

The previous subsections have shown how to establish the graphical environment, manipulate coordinate systems, use primitives, and set attributes. In context, a simple XY line chart with multiple data sets has been demonstrated. The next evolutionary step is the addition of a key. A key is a list of words or phrases giving an explanation of symbols or abbreviations (Figure 2-4 indicates a key in a XY line chart).

In order to add a key, four steps are necessary:

1) declare and allocate key related variables;
2) initialize the key related variables by calling CKEYIN;
3) for each data set to be represented in the key, call CKEYLB. This call should be performed when the data is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active; and
4) finally to plot the key, call CKEYPL. This call should be performed in page coordinates. Thus, the window and viewport may have to be reset.

Step 1 - Declare and allocate variables

A key is a series of entries (lines), where each entry is stored internally as a collection of attributes. The user is required to declare and allocate storage for these attributes. Specifically, two variables are needed, ISTORE (storage array for attributes) and KEYCHR (storage array for line labels). The dimensions of these arrays should be large enough to accommodate the largest number of data sets plotted at one time. See the description of the routine CKEYPL in Appendix A for a more detailed description.

```
                    •
                    •
      PARAMETER(NLINS=2,NCOLS=1)
C NLINS  - the maximum number of lines in the key
C NCOLS  - the number of columns for an entry
C *20    - the largest character string per entry
C          (including mnemonics)
C (Note, in this example 20 is used, could be any value.)

      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER   ISTORE(15,NLINS)
                    •
                    •
```

Step 2 - Initialize the key related variables by calling CKEYIN. This initialization can be performed anywhere, at anytime (i.e., the variables passed to CKEYIN will be reset).

```
                    •
                    •
                    •
C Set window to match data coordinates, and plot within
C boundaries of the axes

    CALL JWINDO(5.,10.,0.,0.01)

C 5,10     - the data coordinate boundaries in the X-direction
C 0.,0.01 - the data coordinate boundaries in the Y-direction

    CALL JVPORT(VX1,VX2,VY1,VY2)
    CALL JOPEN
    CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
                    •
                    •
                    •
```

Step 3 - For each data set to be represented in the key, call CKEYLB. This call should be performed when the data is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active. The key related variables will be assigned values to be plotted when all the key entries have been obtained. This typically requires the plotting of the data (in terms of data coordinates).

```
                    •
                    •
                    •
C Plot the first data curve using CGL defaults
    CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 1')
    CALL CLNPLT(X,Y(1,1),MAXPTS)
C Set symbol number to 2, line pattern to 2, and plot the
C second data curve
    CALL CSYMNO(2)
    CALL CLNPAT(2)
C Note, the data attributes must be set prior to CKEYLB
    CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 2')
    CALL CLNPLT(X,Y(1,2),MAXPTS)
    CALL JCLOSE
                    •
                    •
                    •
```

Step 4 - Finally to plot the key, call CKEYPL. When all the key entries have been made, a call to CKEYPL will plot the key based on the attributes stored in ISTORE and KEYCHR (and a few internal variables). The call to CKEYPL should be performed in page coordinates, and since most data is plotted in data coordinates (where the attributes are saved by CKEYLB), the window and viewport will probably have to be reset.

```
                                    •
                                    •
                                    •
C Now output the key
C Set the window and viewport for page coordinates
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(0.1,0.1*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE',
     +     'C[BLC]OLUMN TITLE',BPOS,5,JCOL,1)
      CALL JCLOSE
                                    •
                                    •
                                    •
C Terminate graphics
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
                                    •
                                    •
                                    •
```

3.3.4 Complete Program Plotting Multiple Linear Data Sets with a Key

The following program is used to demonstrate how to embed these four steps into a complete program. The key related statements have been additionally commented for clarity. Figure 3-9 shows the graphics output from the following program.

```
      PROGRAM CDR6
C-----------------------------------------------------------------------
C  COMPLETE LINE CHART WITH A LEGEND
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
```

```
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 1')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 2')
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(.1,.1*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE','C[BLC]OLUMN TITLE',
     +            BPOS,5,JCOL,1)
      CALL JCLOSE
```

```
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

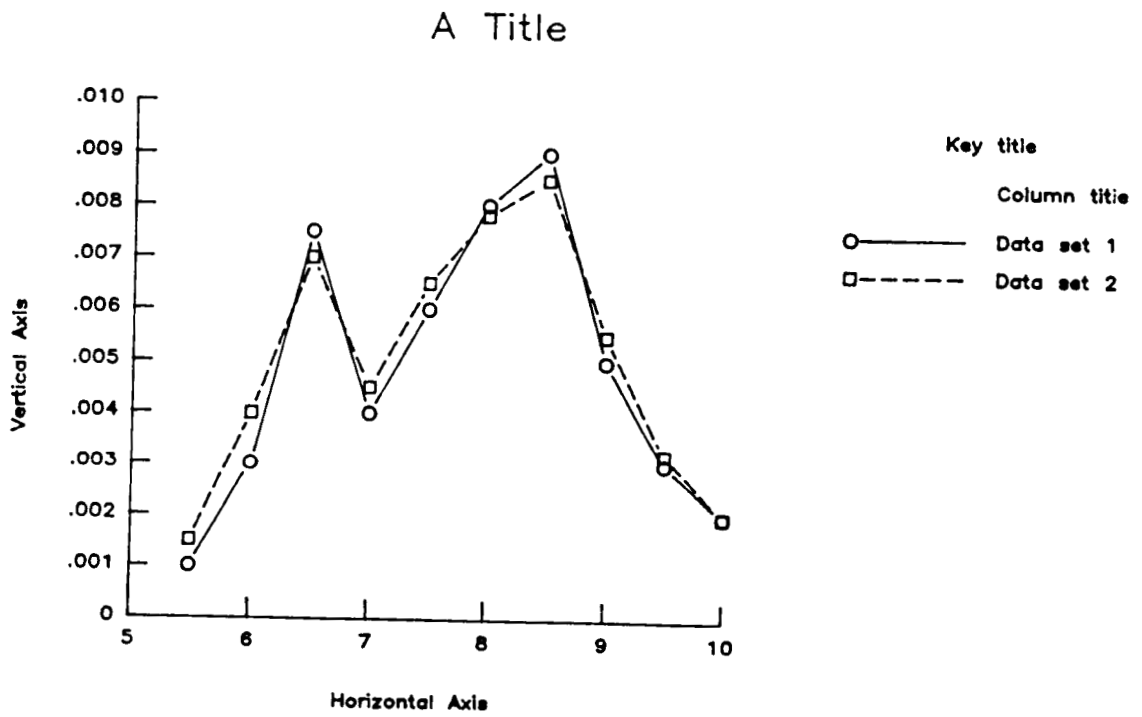


Figure 3-9. Complete XY line chart with a key.

3.3.5 Determining Publication Quality Axis Scale Factors

When the range of the data is predetermined, the user can generate a chart with appropriate axes scales suitable for publication. When the range of the data is unknown, the user can call subroutine CSCALE to aid in determining suitable axis characteristics (i.e., scale factors, number of major and minor tick marks, etc.).

First, the user must establish the data's minimum and maximum values:

```
      CALL CMNMX(X,NPTS,XMIN,XMAX)
```

Next, the user can call CSCALE with the minimum and maximum:

```
      CALL CSCALE(XMIN,XMAX,AMIN,AMAX,NMAJOR,NMINOR,
     +            XINCR,NLEFT,NRIGHT)
```

where:

input

- XMIN - array minimum passed into CSCALE
- XMAX - array maximum passed into CSCALE

output

- AMIN - adjusted minimum
- AMAX - adjusted maximum
- NMAJOR - number of major tick marks
- NMINOR - number of minor tick marks
- XINCR - increments between major tick marks
- NLEFT - number of digits to the left of the decimal place to represent number on axis
- NRIGHT - number of digits to the right of the decimal place to represent number on axis

These values can be used to create an axis suitable for publication purposes:

```
C CHMTIC set the # of minor tick marks on the horizontal axis
      CALL CHMTIC(NMINOR)
C CHPREC set the precision for a numeric horizontal axis
      CALL CHPREC(NRIGHT)
C CHAXIS will draw a horizontal axis of length XLEN, with
C tick mark labels starting at AMIN to AMAX in increments of
C XINCR.  Axes attributes (e.g., NHMTIC, NHPREC, etc.) are
C determined by the attributes set when CHAXIS is called.
      CALL CHAXIS(AMIN,AMAX,XINCR,XLEN)
```

The following is a partial program demonstrating the use of CSCALE to generate publication acceptable axis. The graphics output generated from this example is found in Figure 3-10.

```
      PROGRAM CDR7
C-----------------------------------------------------------------------
C  SEMI-LOGARITHMIC LINE CHART (SINGLE DATA SET)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(NPTS=6)
      REAL X(NPTS),Y(NPTS),YTEMP(NPTS)
      DATA X/2.2,3.0,4.3,5.1,5.5,6.7/
      DATA Y/3.,50.,100.,500.,750.,985./
      DATA XORG/2.5/,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C---------------------  ----------------------------------------------
                       .
                       .
                       .
C---------------------  ----------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LINEAR)
      CALL CMNMX(X,NPTS,XMIN,XMAX)
C DETERMINE PUBLICATION QUALITY SCALE FACTORS
      CALL CSCALE(XMIN,XMAX,AMIN,AMAX,NMAJOR,NMINOR,XINCR,NLEFT,NRIGHT)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHMTIC(NMINOR)
      CALL CHPREC(NRIGHT)
      CALL CHLAB('F[BLC]REQUENCY, [BUC]H[BLC]Z',1)
      CALL CHAXIS(AMIN,AMAX,XINCR,XLEN)
C---------------------------  ----------------------------------------
                             .
                             .
                             .
```

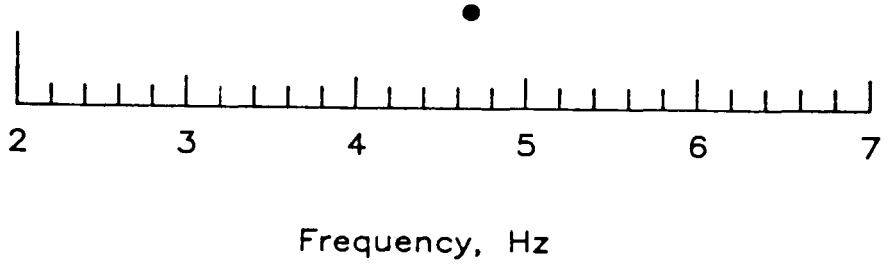


Figure 3-10. Example of axis generation with CSCALE.

3.3.6 Plotting Logarithmic Data

In Subsection 3.3.2, Plotting Linear Data, both the data and the data space progressed in a linear fashion (i.e., the scales increase in a proportional manner in relationship to the axis). A call to JWINDO with the data extremes established a data space encompassing the data to be plotted. Since the underlying graphics package only supplies linear windows, logarithmic data must be transformed onto a linear coordinate system. This is accomplished by converting the logarithmic data by "taking the log of the data", and plotting the resultant array onto the linear data space which has the appropriate range.

A couple of CGL routines are available to aid the user in plotting logarithmic data. First, to determine the minimum and maximum of the logarithmic data, the user can call CMNMX:

```
CALL CMNMX(YARRAY,NPTS,YMIN,YMAX)
```

where:

input

- YARRAY - represents the array containing the logarithmic data
- NPTS - represents the number of points in YARRAY

output

- YMIN - the returned minimum from YARRAY
- YMAX - the retuned maximum from YARRAY

Next, to determine the "powers of ten" to use as the axis boundaries, the routine CEXP can be used.

```
CALL CEXP(YMIN,YMAX,MINEXP,MAXEXP)
```

where:

input

- YMIN - the minimum from YARRAY
- YMAX - the maximum from YARRAY

output

- MINEXP - the returned minimum exponent
- MAXEXP - the returned maximum exponent

The returned values, MINEXP and MAXEXP, represent the exponents (i.e., the powers of ten) to be used as the axis extremes which will ensure the data will be completely encompassed. Also, from these two values, the number of logarithmic cycles (intervals) can be determined by MAXEXP-MINEXP, whereas the number of major tick marks is determined by (MAXEXP-MINEXP)+1.

For example, if CMNMX returns 3.0 and 997.0 as the minimum and maximum of an array, then CEXP would return MINEXP as 0, and MAXEXP as 3 (i.e., 10**0 to 10**3, the axis extremes based on the powers of ten to encompass the data).

To draw a logarithmic axis, the routine CHLOG will draw horizontal axis, while CVLOG will draw a vertical axis.

```
C Set the initial power of ten used to label the axis
    CALL CHLOGS(MXEXP1)
C Generate a horizontal log axis of length XLEN, with NTICX
C major tick marks, with the minor tick marks labeled as
C denoted by NHLOGI (see CHLOG description in Appendix A).
    CALL CHLOG(XLEN,NTICX,NHLOGI)
```

We now can establish a data space corresponding to these extremes. First, a set of logarithmic axes will be positioned and drawn. The following partial program illustrates how to draw logarithmic axes.

```
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LOG)
      CALL CMNMX(X,MAXPTS,XMIN,XMAX)
      CALL CEXP(XMIN,XMAX,MXEXP1,MXEXP2)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CSET('NHLOGF',1)
      CALL CSET('NHLOGS',MXEXP1)
      NTICX=(MXEXP2-MXEXP1)+1
      CALL CHLOG(XLEN,NTICX,1)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y(1,1),MAXPTS,YMIN,YMAX)
      DO 10 I=2,MAXSET
        CALL CMNMX(Y(1,I),MAXPTS,TMIN,TMAX)
        IF(TMIN.LT.YMIN)YMIN=TMIN
 10     IF(TMAX.GT.YMAX)YMAX=TMAX
      CALL CEXP(YMIN,YMAX,MYEXP1,MYEXP2)
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MYEXP1)
      NTICY=(MYEXP2-MYEXP1)+1
      CALL CVLOG(YLEN,NTICY,1)
C-----------------------------------------------------------------------
```

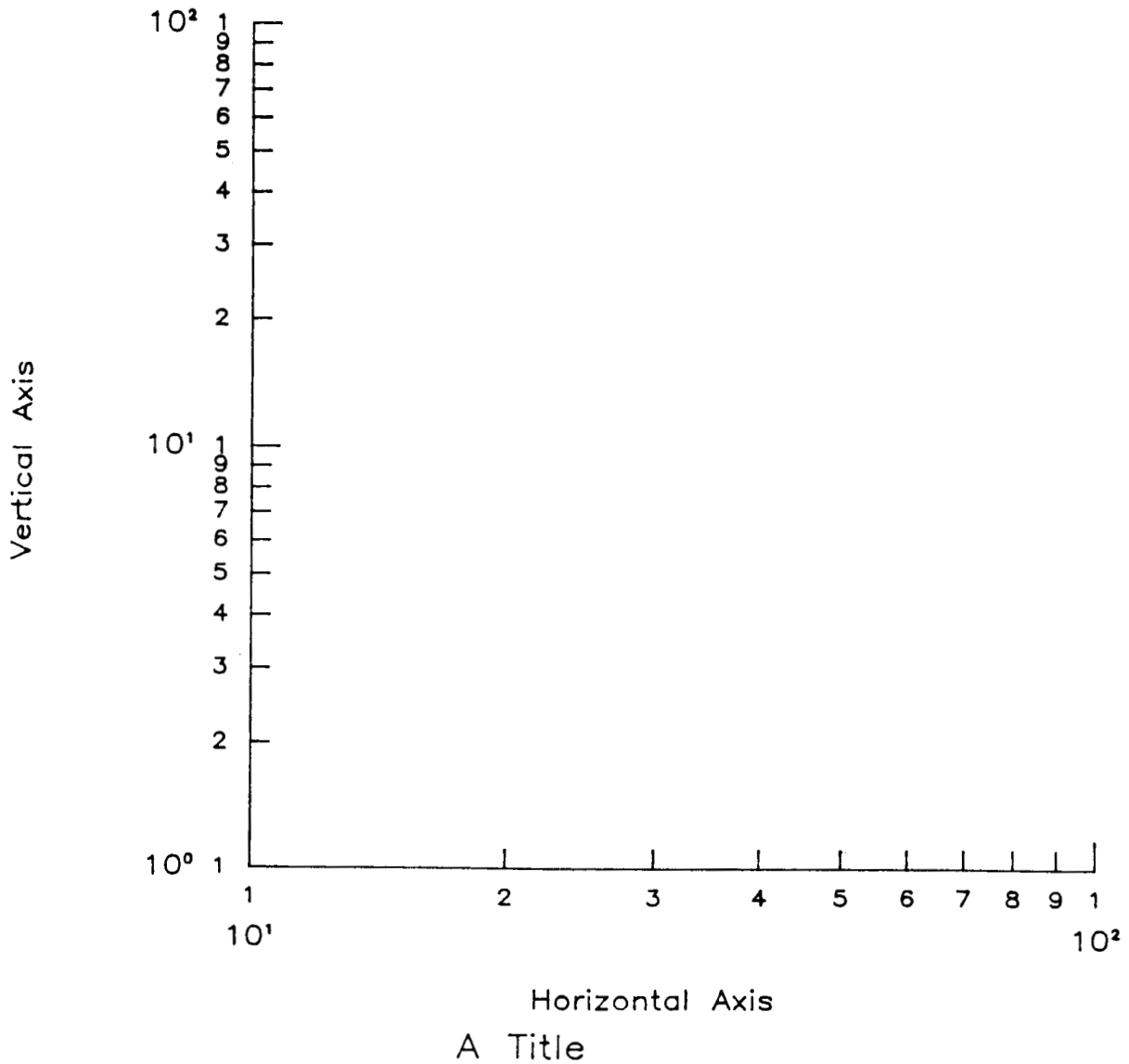


Figure 3-11. Example of a set of logarithmic axes.

Next, we must determine what the current virtual coordinates are which bound the axes to be used to set the data space and the corresponding virtual coordinate system (see Subsection 3.3.2 for a detailed description)

```
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)

C XORG,YORG -  The intersection of the axes
C VX1,VY1   -  The virtual coordinates of XORG,YORG

      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)

C XLEN,YLEN -  The lengths of the axes
C VX2,VY2   -  The virtual coordinates of the opposite
C              diagonal of the axes
```

Next, we set up the data space and viewport:

```
C Set viewport to data region virtual coordinates
    CALL JVPORT(VX1,VX2,VY1,VY2)
C Set up data region window
C Since X-axis is log, minimum and maximum must be scaled
    CALL JWINDO(REAL(MXEXP1),REAL(MXEXP2),
   +            REAL(MYEXP1),REAL(MYEXP2))
C Enable clipping to exclude extraneous data
    CALL JWCLIP(.TRUE.)
```

To complete the chart we open the segment, set the desired plotting attributes, and plot the data. But, we must convert the logarithmic data to correspond to the linear data space [i.e., XTEMP(I)=LOG(X(I))].

```
            DO 1 I=1,NPTS
            XTEMP(I)=LOG(X(I))
    1       YTEMP(I)=LOG(Y(I))
C NOTE: THE ABOVE LOOP ASSUMES THE VALUES IN THE X AND Y
C       ARRAYS ARE GREATER THAN 0.
            CALL JOPEN
            CALL CSYMNO(1)
            CALL CLNPLT(XTEMP,YTEMP,NPTS)
            CALL JCLOSE
```

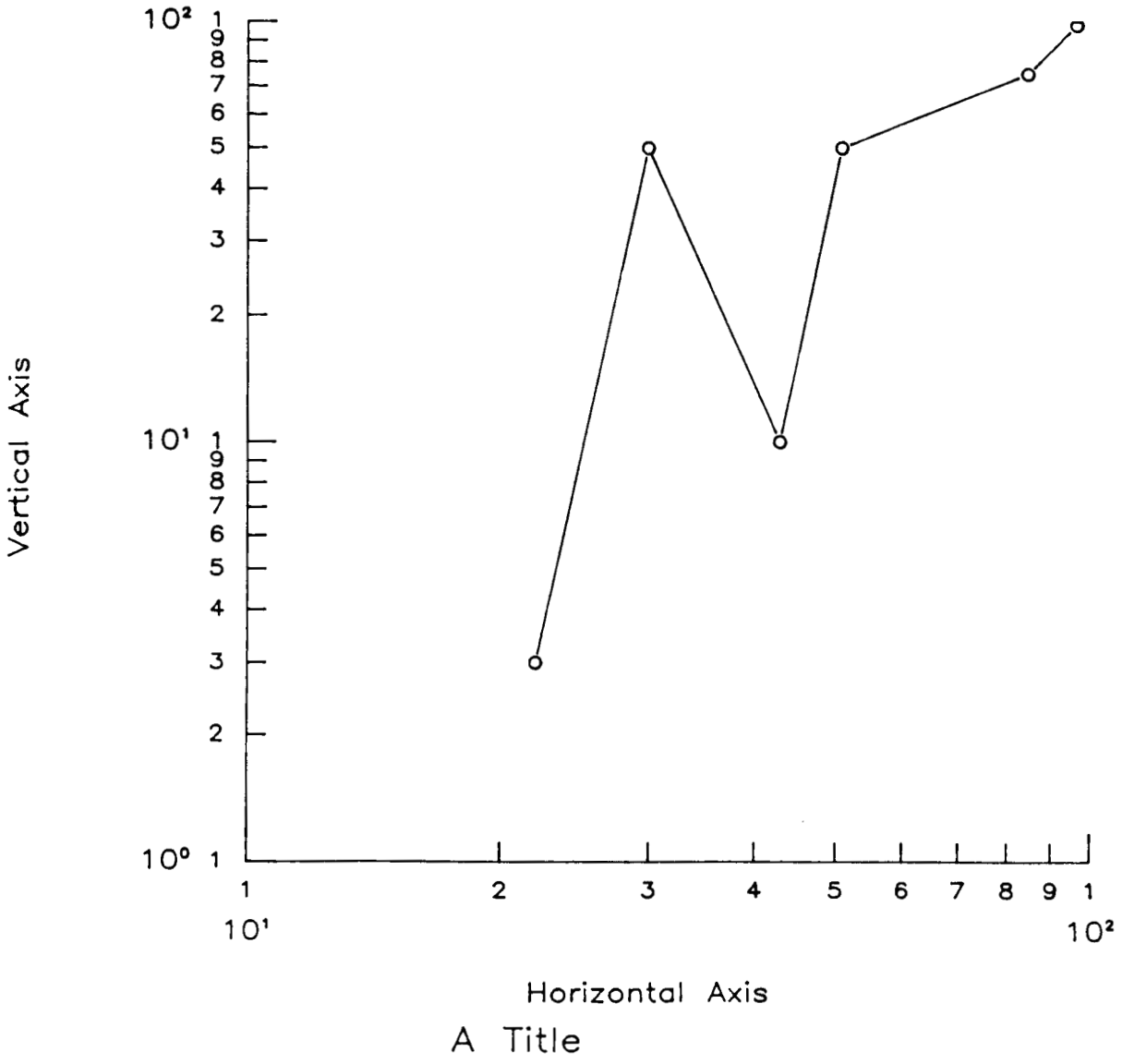


Figure 3-12. Example of plotting logarithmic data.

3.3.7 Complete Program Plotting Multiple Logarithmic Data Sets

The following is a complete program showing the generation of a logarithmic chart. This program illustrates the use of CMNMX, CEXP, JCONWV, and the mapping of logarithmic data onto a linear scale. The corresponding graphics output can be found in Figure 3-13.

```
      PROGRAM CDR8
C-----------------------------------------------------------------------
C  LOG-LOG LINE CHART (MULTIPLE DATA SETS)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(MAXPTS=6,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),YTEMP(MAXPTS),XTEMP(MAXPTS)
      DATA X/22.,30.,43. ,51. ,85. ,97. /
      DATA (Y(KI,1),KI=1,MAXPTS)/3.0,50.,10. ,50. ,75. ,98. /
      DATA (Y(KI,2),KI=1,MAXPTS)/7.0,90.,10.,25. ,55. ,90. /
      DATA XORG/2./,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C-----------------------------------------------------------------------
C SET UP GRAPHICS AREA
      IDEV=0
      CALL JBEGIN
      CALL CBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET PAGE COORDINATES, AND OPEN A TEMPORARY SEGMENT
      CALL JWINDO(0.,XPAGE,0.,YPAGE)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET CHARACTER SIZE, AND OUTPUT TITLE
C SET THE CHARACTER SIZE (IN TERMS OF PAGE CORRDINATES)
      XSIZE=.25
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,BOTTOM)
      CALL JMOVE(XPAGE/2.,0.+XSIZE)
      CALL JJUST(2,1)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
```

```
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LOG)
      CALL CMNMX(X,MAXPTS,XMIN,XMAX)
      CALL CEXP(XMIN,XMAX,MXEXP1,MXEXP2)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CSET('NHLOGF',1)
      CALL CSET('NHLOGS',MXEXP1)
      NTICX=(MXEXP2-MXEXP1)+1
      CALL CHLOG(XLEN,NTICX,1)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y(1,1),MAXPTS,YMIN,YMAX)
      DO 10 I=2,MAXSET
        CALL CMNMX(Y(1,I),MAXPTS,TMIN,TMAX)
        IF(TMIN.LT.YMIN)YMIN=TMIN
 10     IF(TMAX.GT.YMAX)YMAX=TMAX
      CALL CEXP(YMIN,YMAX,MYEXP1,MYEXP2)
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MYEXP1)
      NTICY=(MYEXP2-MYEXP1)+1
      CALL CVLOG(YLEN,NTICY,1)
C-----------------------------------------------------------------------
C SAVE DATA REGION VIRTUAL COORDINATES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET VIEWPORT TO DATA REGION VIRTUAL COORDINATES
      CALL JVPORT(VX1,VX2,VY1,VY2)
C-----------------------------------------------------------------------
C SET UP DATA REGION WINDOW
C X-AXIS IS LOG, MIN AND MAX MUST BE SCALED.
      CALL JWINDO(REAL(MXEXP1),REAL(MXEXP2),REAL(MYEXP1),REAL(MYEXP2))
C ENABLE CLIPPING TO EXCLUDE EXTRANEOUS DATA
      CALL JWCLIP(.TRUE.)
      CALL JOPEN
C PLOT DATA
      DO 1 KI=1,MAXPTS
 1    XTEMP(KI)=LOG10(X(KI))
      DO 2 KI1=1,MAXSET
        CALL CSYMNO(KI1)
      CALL CLNPAT(KI1)
        DO 3 KI2=1,MAXPTS
 3      YTEMP(KI2)=LOG10(Y(KI2,KI1))
 2    CALL CLNPLT(XTEMP,YTEMP,MAXPTS)
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
```

```
C-----------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
C-----------------------------------------------------------------
      STOP
      END
```

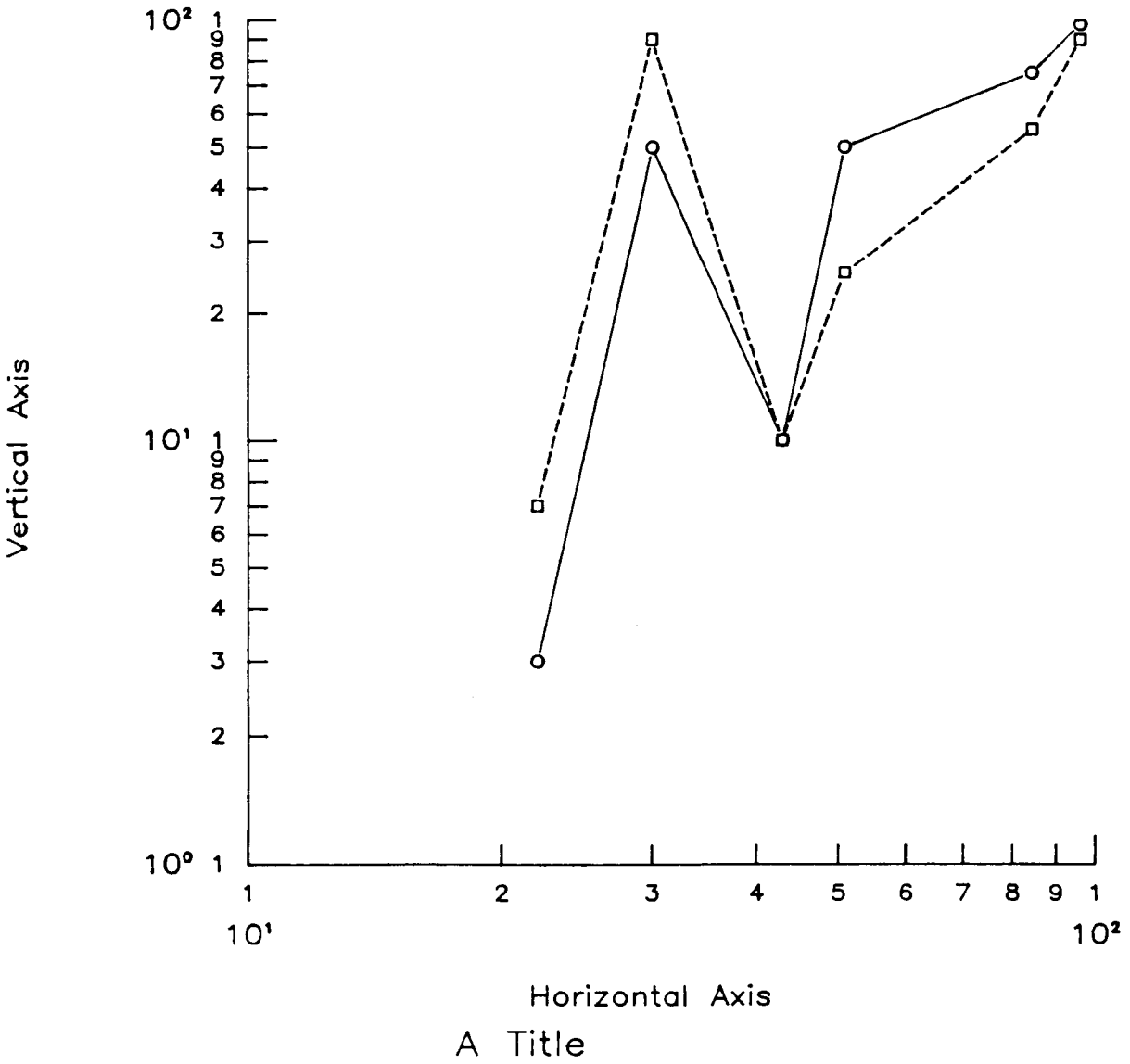


Figure 3-13. Complete logarithmic chart with multiple data sets.

3.3.8 Complete Program Plotting Multiple Semi-Logarithmic Data Sets

Throughout the discussion in Section 3.3.2 through 3.3.7, the reader has been shown how to generate linear and logarithmic charts. The same techniques and principles apply in the generation of semi-logarithmic chart. The following program combines these techniques using the linear method on the horizontal axis, and the logarithmic method on the vertical axis.

```
      PROGRAM CDR7
C-----------------------------------------------------------------------
C  SEMI-LOGARITHMIC LINE CHART (SINGLE DATA SET)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(NPTS=6)
      REAL X(NPTS),Y(NPTS),YTEMP(NPTS)
      DATA X/2.2,3.0,4.3,5.1,5.5,6.7/
      DATA Y/3.,50.,100.,500.,750.,985./
      DATA XORG/2.5/,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C-----------------------------------------------------------------------
C SET UP GRAPHICS AREA
      IDEV=0
      CALL JBEGIN
      CALL CBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET PAGE COORDINATES, AND OPEN A TEMPORARY SEGMENT
      CALL JWINDO(0.,XPAGE,0.,YPAGE)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET CHARACTER SIZE, AND OUTPUT TITLE
C SET THE CHARACTER SIZE (IN TERMS OF PAGE CORRDINATES)
      XSIZE=.25
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,BOTTOM)
      CALL JMOVE(XPAGE/2.,0.+XSIZE)
      CALL JJUST(2,1)
C OUTPUT THE STRING
      CALL JHSTRG('P[BLC]OWER LEVEL VS. FREQUENCY')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
```

```
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LINEAR)
      CALL CMNMX(X,NPTS,XMIN,XMAX)
C DETERMINE PUBLICATION QUALITY SCALE FACTORS
      CALL CSCALE(XMIN,XMAX,AMIN,AMAX,NMAJOR,NMINOR,XINCR,NLEFT,NRIGHT)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHMTIC(NMINOR)
      CALL CHPREC(NRIGHT)
      CALL CHLAB('F[BLC]REQUENCY, [BUC]H[BLC]Z',1)
      CALL CHAXIS(AMIN,AMAX,XINCR,XLEN)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y,NPTS,YMIN,YMAX)
      CALL CEXP(YMIN,YMAX,MINEXP,MAXEXP)
      CALL CVLAB('P[BLC]OWER LEVEL',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MINEXP)
      NTIC=(MAXEXP-MINEXP)+1
      CALL CVLOG(YLEN,NTIC,4)
C-----------------------------------------------------------------------
C SAVE DATA REGION VIRTUAL COORDINATES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET VIEWPORT TO DATA REGION VIRTUAL COORDINATES
      CALL JVPORT(VX1,VX2,VY1,VY2)
C-----------------------------------------------------------------------
C SET UP DATA REGION WINDOW
      DO 1 KI=1,NPTS
 1    YTEMP(KI)=LOG10(Y(KI))
C X-AXIS IS LOG, MIN AND MAX MUST BE SCALED.
      CALL JWINDO(AMIN,AMAX,REAL(MINEXP),REAL(MAXEXP))
C ENABLE CLIPPING TO EXCLUDE EXTRANEOUS DATA
      CALL JWCLIP(.TRUE.)
      CALL JOPEN
C PLOT DATA
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,YTEMP,NPTS)
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
C-----------------------------------------------------------------------
      STOP
      END
```

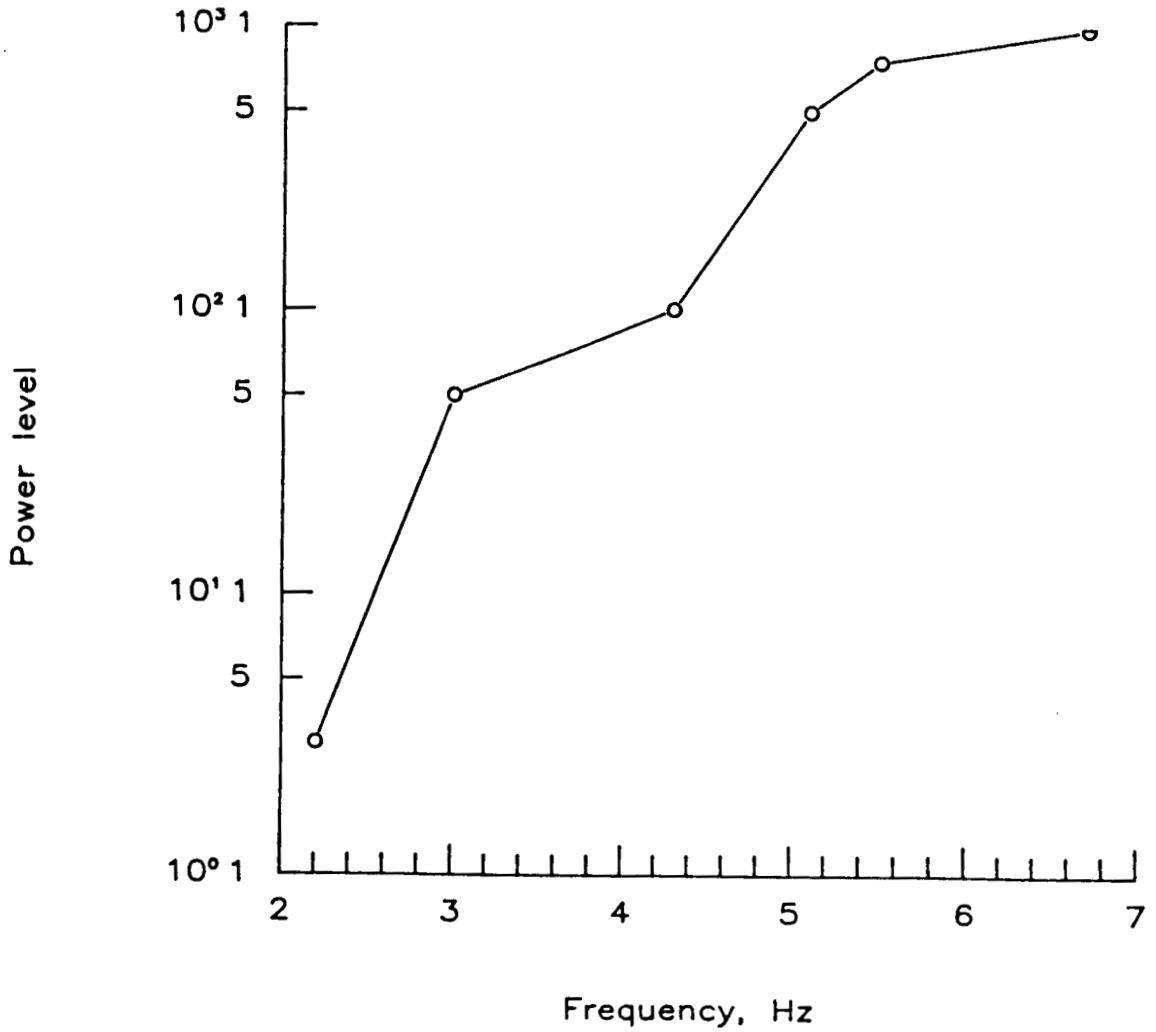


Figure 3-14. Complete semi-logarithmic chart.

3.3.9 Plotting Data with Grids without Keys

The Low-Level routines offer a grid capability for both linear and logarithmic scales. This section describes how to generate linear-linear grids, semi-logarithmic grids, and log-log grids. Subsequently, this section will illustrate how to generate any of these grid types without rectangular areas remaining blank (i.e., usually intended for keys). Section 3.3.11 describes grid generation with blank areas.

Two routines are currently available, CGRID and CLGRID. The CGRID routine draws a linear grid over a specified plotting area with optional rectangular or square areas remaining blank. CLGRID enables the user to draw log-log, semi-log, or linear grids with optional blanks for keys.

Three major steps are needed to generate a grid (either linear or logarithmic). The first step is to generate the axes as described in the previous subsections. The grid routine will not label the axes, this must be performed (if desired) by the axis routines. The second step consists of calling the appropriate grid routine (i.e., CGRID or CLGRID) to superimpose a grid onto the axes. The user must ensure that these grid intervals match and align with the axes intervals. The third step is to close the window (representing page coordinates), open a window for the data coordinates, and plot the data.

The only new step is the generation of grids. A grid extends both in the X and Y directions, and may be required to leave a space (hole or area) for a key. Unlike, the axis routines which are handled independently, the call to CGRID (or CLGRID) will draw the complete grid. The user must therefore supply both axes information to the single grid routine.

To generate a linear grid, a call to CGRID is required. This call must be performed in an opened segment, with the window representing page coordinates.

The following is a partial program showing the interaction between axes and linear grids.

```
                                    •
                                    •
                                    •
    CALL JWINDO(0.,9.,0.,9.)
    CALL JOPEN
    CALL JMOVE(XORG,YORG)
    CALL CHAXIS(5.,10.,1.,XLEN)
    CALL CVAXIS(0.,0.01,0.001,YLEN)
C Based on the axes, determine # of intervals
C X intervals =        (XMAX-XMIN)/XINCR
    NOINCX     =  NINT((10.0- 5.0)/1.0  )
C X increments= X-axis length/X intervals
    XS         = XLEN          /REAL(NOINCX)
C Y intervals =        (YMAX-YMIN)/YINCR
    NOINCY     =  NINT((10.0- 5.0)/1.0  )
C Y increments= Y-axis length/Y intervals
    YS         = YLEN          /REAL(NOINCY)
C Since no blank areas, then set NBLANK TO 0
    NBLANK=0
C Call linear grid routine
    CALL CGRID(XORG,YORG,XS,YS,NOINCX,NOINCY,BLANK,NBLANK)
                                    •
                                    •
                                    •
```

The resultant figure is shown in Figure 3-15.

Similarly, to generate a semi-log or log-log grid, the same approach applies where a call to CLGRID replaces CGRID. Refer to Appendix A for a complete description of CLGRID, and Appendix B for an example program showing a semi-log grid.

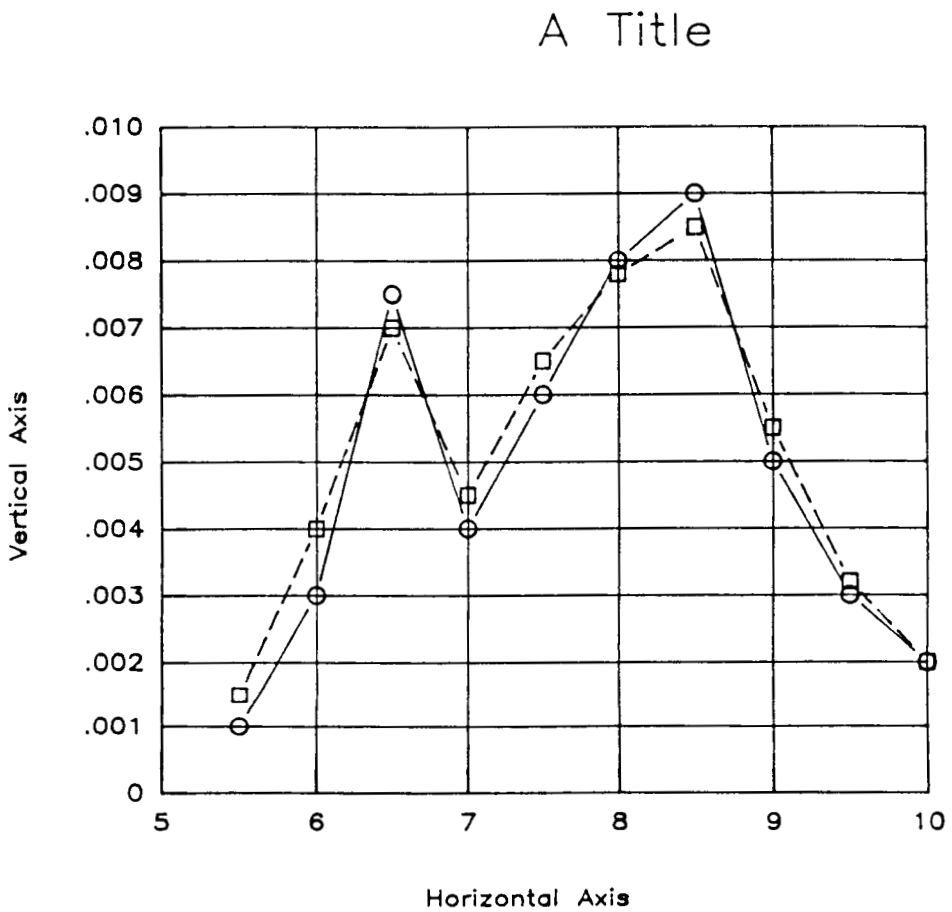


Figure 3-15. Linear grid generated with CGRID.

3.3.10 Plotting Data with Grids and Keys

In order to leave a space in a grid for a key, the key extents (height and width) must be determined. To do so, the attributes of all data sets must be saved. Since saving attributes (i.e., call to CKEYLB) is performed in the data coordinate system, and the grid is usually described in page coordinates, the window must be reset.

The following are the major steps in setting up a grid with a space for the key:

1) generate the axes (page coordinates)
2) draw the data (data coordinates), while saving key related attributes
3) determine key extents, and call the grid routine with key extents (page coordinates)
4) plot the key

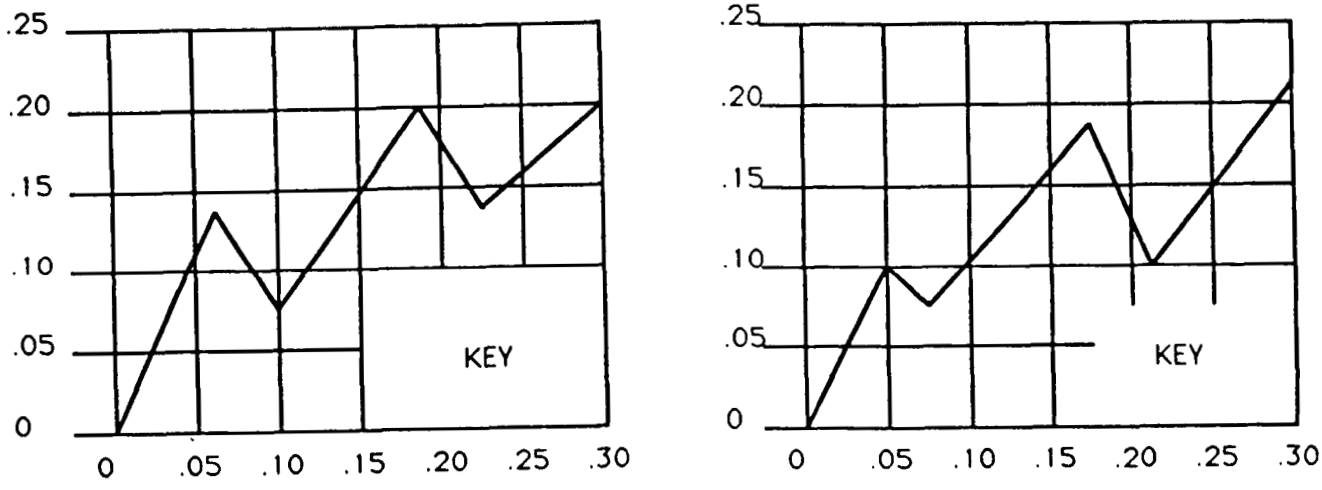
Steps 1 and 2 are explained in previous subsections.

Step 3 - Once all the data sets have been plotted (i.e., all of the key related attributes have been saved), then the key extents can be determined. A call to CKEYPL with the variable KPT set to 1, will return the lower-left X and Y coordinates in BPOS(1),BPOS(2), and the upper-right X and Y coordinate in BPOS(3), BPOS(4).

```
      .
      .
      .
      KPT=1
      CALL CKEYPL(ISTORE,KEYCHR,CTTL,BPOS,KPT,ICOL,NHPR)
C BPOS(1),BPOS(2) = the lower-left  corner (XY coordinate)
C BPOS(3),BPOS(4) = the upper-right corner (XY coordinate)
C Calculate the X and Y extents
      XXTENT=BPOS(3)-BPOS(1)
      YXTENT=BPOS(4)-BPOS(2)
      .
      .
      .
```

These extents can then be passed to the grid routine to suppress the grid lines through this area.

Step 4 - Now that the user knows the dimensions of the key, the user must determine where to position the key in relationship to the chart. It is preferred for publications that the grid lines stop on grid lines, as opposed to grid lines which dangle (see Figure 3-16). Thus, even though the user determines the extents, the user must further determine the dimensions of the grid lines to suppress. These dimensions are then passed to the grid routine (i.e., either CGRID or CLGRID), and the key should be centered in the grid space requested to be blank.



preferred non-preferred

Figure 3-16. Sample grids with keys.

To request blank space in a grid, the variables NBLANK and BLANK are used to inform the grid routine. NBLANK indicates the number of blank areas. BLANK is an array the coordinates (representing the diagonals) of the rectangle to be left.

```
                                    •
                                    •
                                    •
C The number of blank regions
    NBLANK=2
C Coordinates of the blank regions
    BLANK(1)= {lower-left  X coordinate of region 1}
    BLANK(2)= {lower-left  Y coordinate of region 1}
    BLANK(3)= {upper-right X coordinate of region 1}
    BLANK(4)= {upper-right Y coordinate of region 1}
C
    BLANK(5)= {lower-left  X coordinate of region 2}
    BLANK(6)= {lower-left  Y coordinate of region 2}
    BLANK(7)= {upper-right X coordinate of region 2}
    BLANK(8)= {upper-right Y coordinate of region 2}
                                    •
                                    •
                                    •
```

Program CDR11 in Appendix B, is a complete program demonstrating these steps.

3.3.11 Line Charts with Variations of Axes

This subsection describes how to generate line charts with axes other than the standard linear/logarithmic vertical and horizontal axes. Examples of non-standard axes include (see Figure 3-17):

- decreasing scales: axes which have a negative increment
- multiple scales: two or more vertical or horizontal axes
- interior axes: axes which are interior to the data space
- disjoint axes: axes which do not intersect

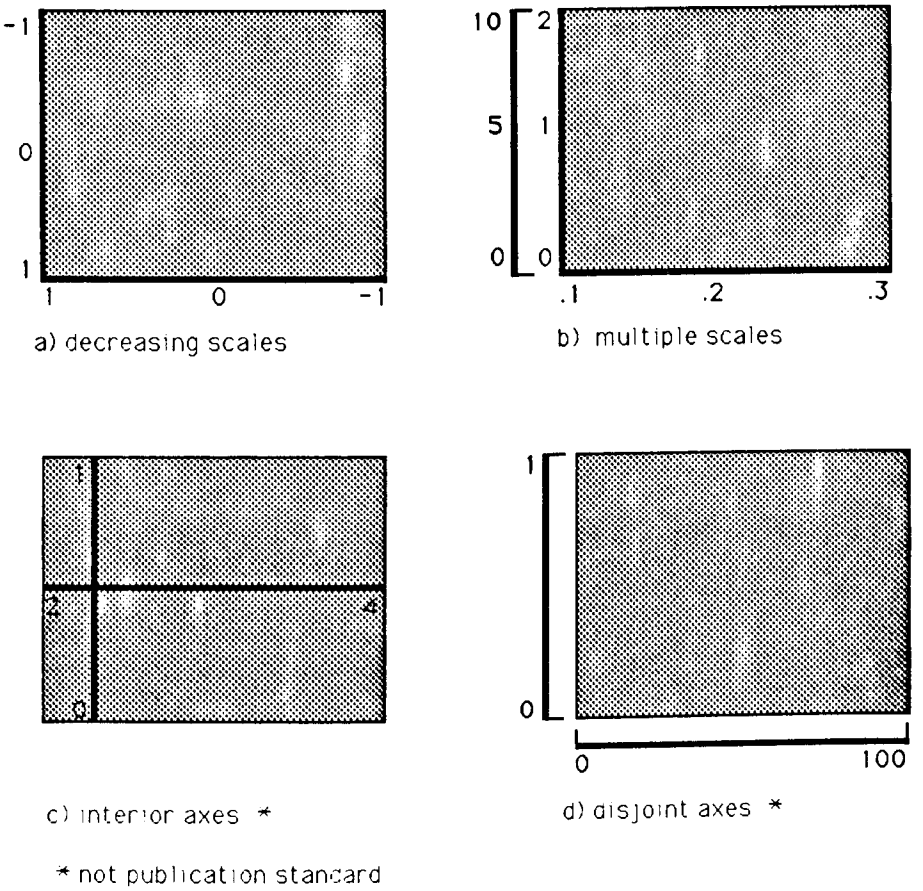


Figure 3-17. Examples of non-standard axes.
Note: shaded areas represent the data space denoted by the axes.

3.3.11.1 Plotting Data with a Decreasing Axis

A decreasing axis is a numeric axis with negative increments. Thus, the values on a horizontal axis decrease from left to right, while the values on a vertical axis decrease from bottom to top (see Figure 3-18).

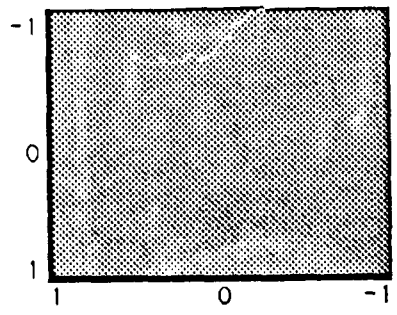


Figure 3-18. Example of decreasing axes.

In order to plot data with decreasing axes (or axis), two major steps must be performed:

1) the generation of the axes; and
2) the plotting of the data to coincide with the axes.

The generation of a decreasing numeric axis is obtained using a call to CHAXIS (for horizontal axes) or CVAXIS (for vertical axes). To generate axes with decreasing values the appropriate arguments must be passed (i.e., the starting value less than the ending value, and a negative increment). Note, a decreasing axis can also be generated using CHAXIC or CVAXIC, with the axis labels having character strings representing the decreasing values.

Since windows (data spaces) in underlying graphics packages are usually increasing in nature, the user must manipulate the decreasing data to conform the increasing window. This entails the usual process of drawing the axes, and setting the window and viewport to match the axes. Then because axis is decreasing to match the data, and the window must be increasing, the data has to be mapped to match the world. For example, if the horizontal axis decreases from 10 to 5, the axis should reflect this decrease, while the corresponding window should be set to 5 to 10. Next, the data set (or sets) coinciding with the decreasing axis must be adjusted to reflect the window in which the data is to be plotted. To aid in this process, the routine CFLIP has been written. The user

passes the data set, the number of points, and the reflection point about which the data is to be mapped. CFLIP will return the array of points offset from the reflection point on the reflection point's opposite side. For example, given a value 8 and a reflection point 5, CFLIP will return a value of 2 (i.e., an equal distance from the reflection point on the other side). Note, the original value passed through CFLIP can be obtained by passing the adjusted value with the same reflection point (i.e., given a value 2 and a reflection point 5, CFLIP will return a value of 8).

The following is a complete program illustrating a decreasing axis, and the use of CFLIP to adjust the data. See Figure 3-19 for the resultant chart.

```
      PROGRAM CDR12
C-----------------------------------------------------------------------
C A LINE CHART WITH A DECREASING AXIS.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
```

```
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]RIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(10.,5.,-1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
C NOTE: THE WINDOW OF THE UNDERLYING GRAPHICS PACKAGE MUST BE
C       INCREASING IN NATURE.  THE RANGE MUST MATCH THE DATA.
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (XMAX-XMIN)/2. + XMIN
      REFLECT=(10. -  5.)/2. + 5.
      CALL CFLIP(X,MAXPTS,REFLECT)
      CALL JOPEN
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C IF THE X ARRAY IS TO BE USED AGAIN, THE USER MUST APPLY CFLIP AGAIN.
C IN THIS CASE, THE X ARRAY IS NOT USED AGAIN.
C-----------------------------------------------------------------------
```

```
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

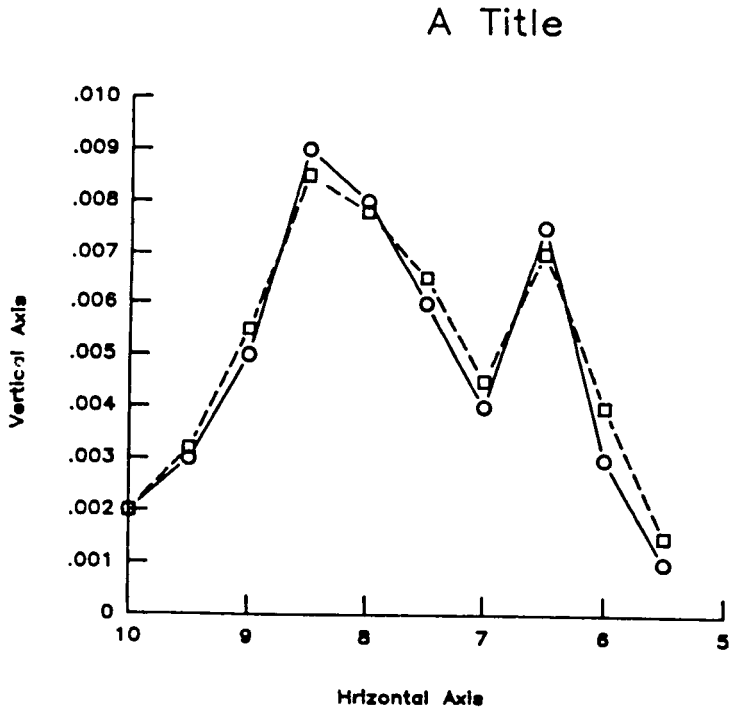


Figure 3-19. Complete program with a decreasing axis.

3.3.11.2 Line Charts with Multiple Scales on an Axis

This subsection describes how to generate line charts with two or more scales on the horizontal or vertical axis. In order to have multiple scales on the same axis, the user must map two windows (with different data extents) onto the same viewport. Thus, the user must set a viewport and window, and plot the appropriate data. Next, the user closes the current segment, resets the window to match the next scale, and plots the data.

The user is referred to previous sections for instructions on generating axes. The following example, however, illustrates the use of mapping two scales (of an axis) onto the same viewport. See Figure 3-20 for the resultant output.

```
      PROGRAM CDR13
C-----------------------------------------------------------------------
C A LINE CHART WITH MULTIPLE VERTICAL SCALES ON THE SAME AXIS.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
```

```
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,.5)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-------------------------------------------------------------------------
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('W[BLC]ING LENGTH',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C-------------------------------------------------------------------------
C DESCRIBE THE VERTICAL AXIS
C NO AXIS LABEL ON INNERMOST VERTICAL AXIS
      CALL CVLAB(' ',0)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
C PLOT FIRST AXIS (DECREASING)
      CALL CVAXIS(.01,.0,-.001,YLEN)
      CALL JMOVE(XORG,YORG+YLEN+.75)
      CALL JHSTRG('[BLC]CM')
C PLOT SECOND AXIS (INCREASING)
C POSITION AND DESCRIBE THE OUTMOST VERTICAL AXIS
      CALL JMOVE(XORG-CSIZE*7.,YORG)
      CALL CVLAB('W[BLC]ING WIDTH',1)
      CALL CVAXIS(.0,.01,.001,YLEN)
C LABEL FOR OUTMOST VERTICAL AXIS
      CALL JMOVE(XORG-CSIZE*7.,YORG+YLEN+.75)
      CALL JHSTRG('[BLC]IN.')
C-------------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-------------------------------------------------------------------------
```

```
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (YMAX-YMIN)/2. + YMIN
      REFLECT=(.01 -  0.)/2. + 0.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C PLOT FIRST DATA CURVE WITH DECREASING AXIS (USING CGL DEFAULTS)
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ECREASING AXIS DATA SET 1')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C REVERSE DATA AGAIN, TO RETURN DATA TO ORIGINAL VALUES.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'I[BLC]NCREASING AXIS DATA SET 2')
C PLOT SECOND DATA CURVE WITH INCREASING AXIS.
      CALL CLNPLT(X,Y(1,1),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'C[BLC]OLUMN TITLE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

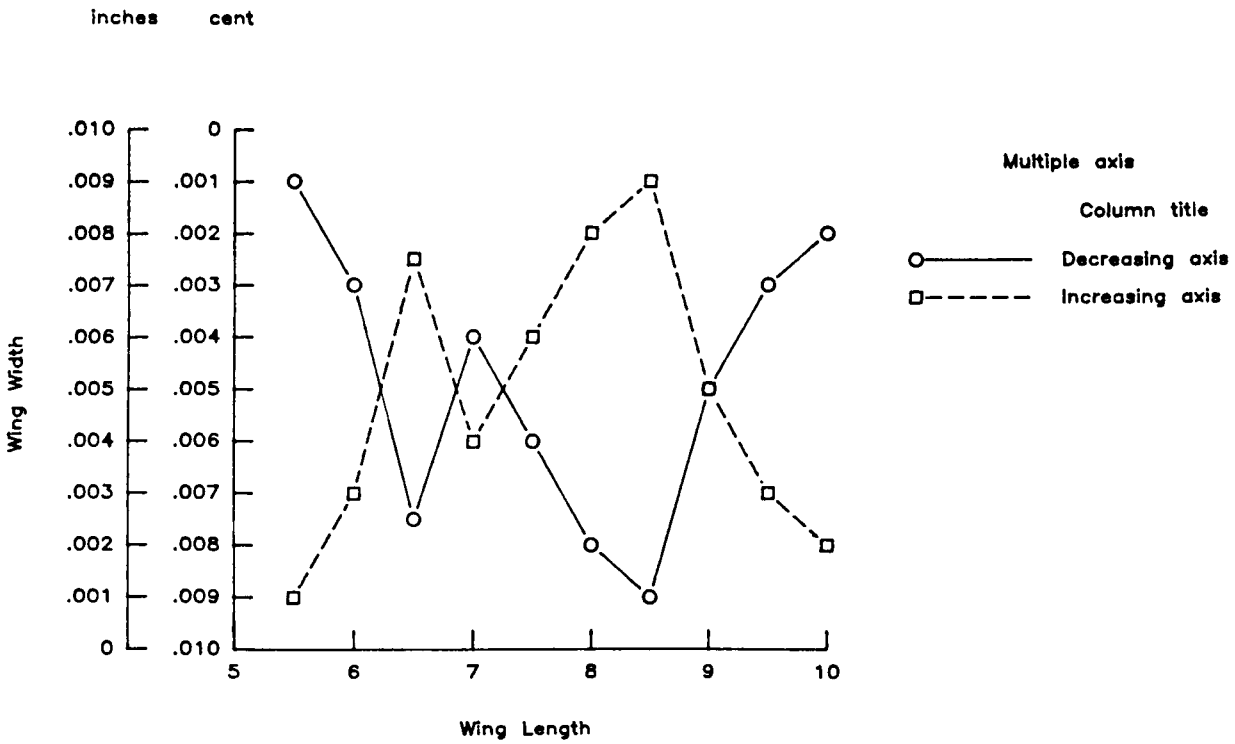


Figure 3-20. Example of multiple vertical scales on the same axis.

The following example illustrates the use of mapping of two scales (of an axis) onto the opposite side of the chart. See Figure 3-21 for the resultant output.

```
      PROGRAM CDR14
C-----------------------------------------------------------------------
C  LINE CHART WITH MULTIPLE VERTICAL SCALES (OPPOSITE AXIS)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,.5)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-----------------------------------------------------------------------
```

```
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=1.25
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C-----------------------------------------------------------------------
C DESCRIBE THE VERTICAL AXIS
C NO AXIS LABEL ON INNERMOST VERTICAL AXIS
      CALL CVLAB('D[BLC]ECREASING AXIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
C PLOT FIRST AXIS (DECREASING)
      CALL CVAXIS(.01,.0,-.001,YLEN)
C PLOT SECOND AXIS (INCREASING)
C POSITION AND DESCRIBE THE OPPOSITE VERTICAL AXIS
      CALL JMOVE(XORG+XLEN,YORG)
C REQUEST VERTICAL AXIS TICK MARKS TO THE RIGHT OF THE AXIS.
C     CALL CVTICJ(2)
C REQUEST VERTICAL AXIS TICK MARK LABELS TO THE RIGHT OF THE AXIS.
      CALL CVLABJ(1)
      CALL CVTICJ(2)
      CALL CVLAB('I[BLC]NCREASING AXIS',1)
      CALL CVAXIS(.0,.01,.001,YLEN)
C-----------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (YMAX-YMIN)/2. + YMIN
      REFLECT=(.01 -  0.)/2. + 0.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
```

```
C PLOT FIRST DATA CURVE WITH DECREASING AXIS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ECREASING DATA')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C REVERSE DATA AGAIN, TO RETURN DATA TO ORIGINAL VALUES.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'I[BLC]NCREASING DATA')
C PLOT SECOND DATA CURVE WITH INCREASING AXIS.
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'D[BLC]ATA TYPE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

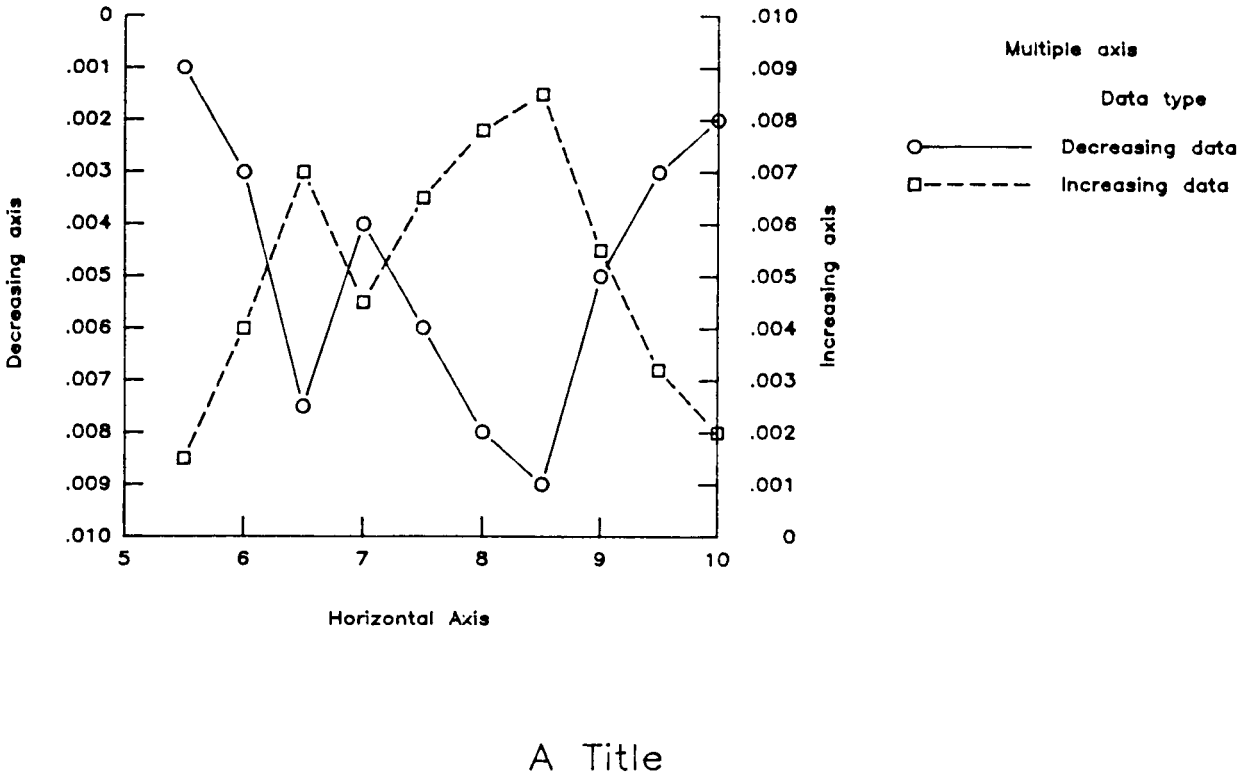


Figure 3-21. Example of multiple vertical scales on opposite axis.

3.3.11.3 Line Charts with Interior Axes

In order to have interior axes, the user must establish the window and viewport to encompass the axes.

The user is referred to previous sections for instructions of generating axes. The following example, however, illustrates the use of interior axes. See Figure 3-22 for the resultant output.

```
      PROGRAM CDR15
C-----------------------------------------------------------------------
C  SIMPLE, YET COMPLETE LOW-LEVEL PROGRAM (WITH A KEY)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(NUMPT1=13,NUMPT2=3,MAXLAB=20)
      REAL X1(NUMPT1),Y1(NUMPT1),BPOS(4)
      REAL X2(NUMPT2),Y2(NUMPT2)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20,CLABS(MAXLAB)*10
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X1/-8.,-6.,-4.,-2.,0.,2.,4.,6.,8.,10.,12.,14.,16./
      DATA JCOL/1/
      DATA (Y1(KI),KI=1,NUMPT1)
     +  /.079,.068,.035,.021,.0,-.021,-.035,-.065,-.08,-.11,
     +   -.118,-.12,-.139/
      DATA X2/-4.,8.,16./,Y2/.039,-.071,-.145/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,TOP)
      CALL JMOVE(9./2.,9.-.5)
      CALL JJUST(2,3)
C OUTPUT THE STRING
      CALL JHSTRG('C[BLC]OMPUTATIONS VS[ELC] E[BLC]XPERIMENTS')
```

```
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-----------------------------------------------------------------------
C BASED AXES DATA ATTRIBUTES (MINIMUM, MAXIMUM, AND INCREMENTS).
      XMIN=-8.
      XMAX=20.
      XINC=4.
      YMIN=-.16
      YMAX=0.12
      YINC=.04
C-----------------------------------------------------------------------
C SET X AND Y AXIS LENGTHS (I.E., XLEN AND YLEN RESPECTIVELY).
      XLEN=4.0
      YLEN=5.5
C POSITION LOWER-LEFT CORNER OF DATA SPACE ON THE PAGE.
      XORG=((9./2.)-1.)-.5*XLEN
      YORG=(9./2.)-.5*YLEN
C NOW, SET XZERO AND YZERO PAGE COORDINATES WHERE AXES 0,0 EXISTS.
C (THIS PROGRAM ASSUMES THAT DATA ENCOMPASSES 0,0 IN X AND Y).
      XZERO=XORG+XLEN*(-XMIN/(XMAX-XMIN))
      YZERO=YORG+YLEN*(-YMIN/(YMAX-YMIN))
C-----------------------------------------------------------------------
C DESCRIBE X-AXIS. MOVE TO X ORIGIN, Y ZERO.
      CALL JMOVE(XORG,YZERO)
C GENERATE X AXIS LABELS.
      CALL CILAB(XMIN,XMAX,XINC,0,CLABS)
      ICOUNT=NINT((XMAX-XMIN)/XINC)+1
C BLANK OUT INTERSECTION LABEL (I.E., 0).
      KI=NINT(ABS(XMIN)/XINC)+1
      CLABS(KI)=' '
C DRAW AXIS
      CALL CHAXIC(XLEN,ICOUNT,CLABS,1,ICOUNT)
C POSITION AND OUTPUT AXIS LABEL
      CALL JMOVE(XORG+(5./7.)*XLEN,YZERO-.5)
      CALL JHSTRG('[FONT=9][BLC]A[FONT=3],DEG')
C-----------------------------------------------------------------------
C DESCRIBE Y-AXIS. MOVE TO X ZERO, Y ORIGIN.
      CALL JMOVE(XZERO,YORG)
C GENERATE Y AXIS LABELS.
      CALL CILAB(YMIN,YMAX,YINC,2,CLABS)
      ICOUNT=NINT((YMAX-YMIN)/YINC)+1
C BLANK OUT INTERSECTION LABEL (I.E., 0).
      KI=NINT(ABS(YMIN)/YINC)+1
      CLABS(KI)=' '
C DRAW AXIS.
      CALL CVAXIC(YLEN,ICOUNT,CLABS,1,ICOUNT)
C POSITION AND OUTPUT AXIS LABEL
      CALL JMOVE(XZERO-1.,YORG+(2.5/7.)*YLEN)
      CALL JHSTRG('C[BLC][BSUB]M (0.25C)')
C-----------------------------------------------------------------------
```

```
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(XMIN,XMAX,YMIN,YMAX)
C XMIN,XMAX - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C YMIN,YMAX - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
C SET SYMBOL TO 1 (A CIRCLE) FOR EXPERIMENTAL DATA.
      CALL CSYMNO(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'E[BLC]XPERIMENTAL')
      CALL CLNPLT(X1,Y1,NUMPT1)
C PLOT SECOND DATA CURVE.
C SET SYMBOL TO SOLID SQUARE (ACTUAL), AND PLOT SYMBOLS ONLY.
      CALL CSYMNO(902)
      CALL CPLTYP(-1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'A[BLC]CTUAL')
      CALL CLNPLT(X2,Y2,NUMPT2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'D[BLC]ATA TYPE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

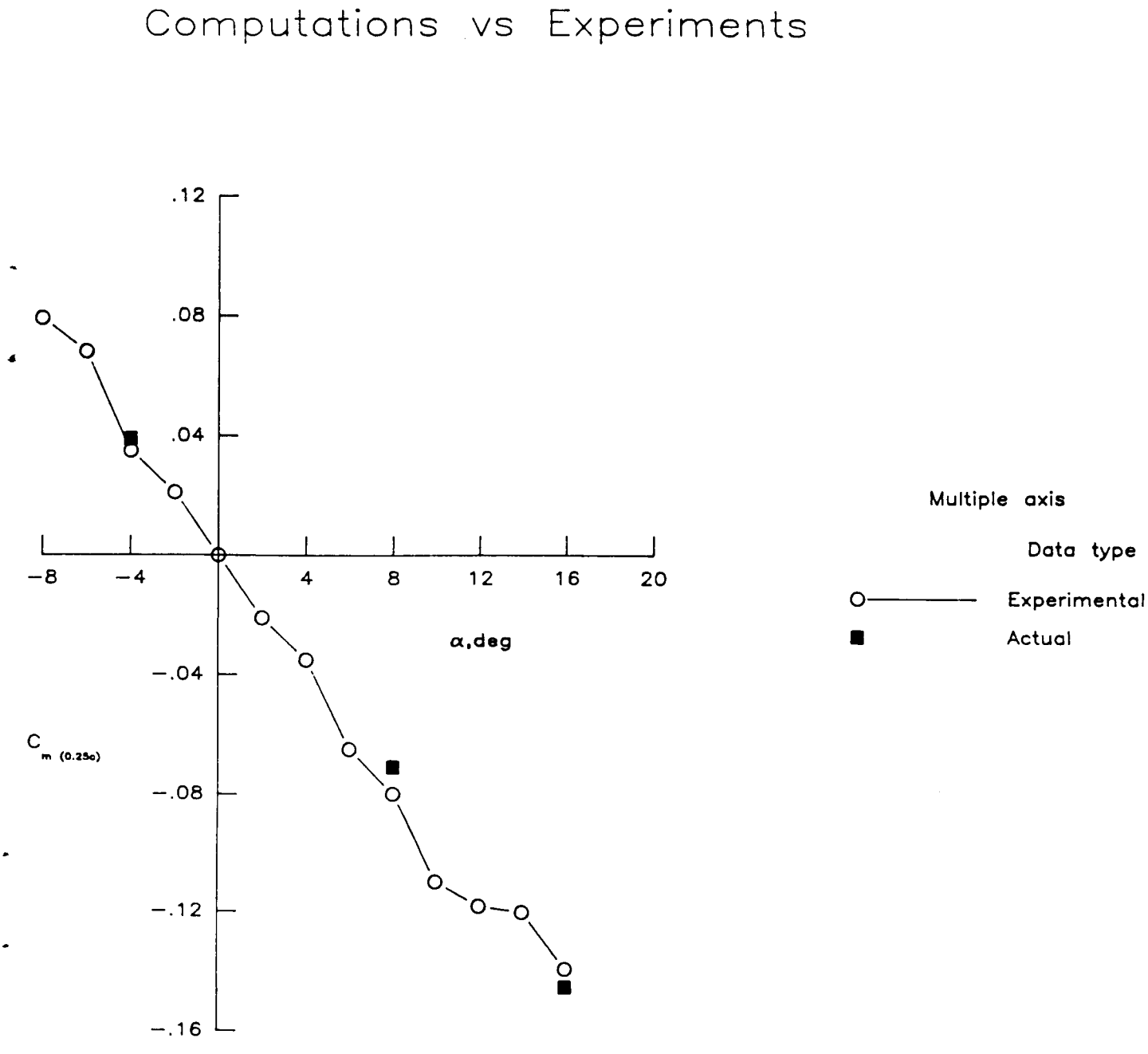


Figure 3-22. Complete program with interior axes.

3.3.11.4 Line Charts with Disjoint Axes

This subsection describes how to generate line charts with a disjoint set of axes. Note, disjoint axes are not publication quality. The crucial element consists of the placement of the axes in relationship to one another and the data space (i.e., placement of the viewport). First the user draws the axes, then offsets the viewport from the axes. The data will be plotted in the window, which in turn is mapped onto the viewport.

The user is referred to previous sections for instructions on generating axes. The following example, however, illustrates the use of interior axes. See Figure 3-23 for the resultant output.

```
      PROGRAM CDR16
C-----------------------------------------------------------------------
C LINE CHART WITH DISJOINT AXES (BASED ON CDR6)
C-----------------------------------------------------------------------
C THE KEY ELEMENTS INVOLVING DISJOINT AXES INCLUDE:
C - THE POSITIONING OF THE AXES (SEE XOFF AND YOFF),
C - THE SETTING OF THE VIEWPORT TO MATCH THE DATA SPACE
C   (WHICH DIFFERS FROM THE AXES BOUNDARIES).
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
```

```
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
C XOFF - THE DISTANCE TO SHIFT THE X-AXIS DOWN.
      XOFF=.2
      CALL JMOVE(2.,2.-(XOFF))
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHAXIS(5.,10.,1.,4.)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YOFF - THE DISTANCE TO SHIFT THE Y-AXIS LEFT.
      YOFF=.2
      CALL JMOVE(2.-YOFF,2.)
      CALL CVAXIS(.0,.01,.001,3.5)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
C NOTE: THE VIRTUAL COORDINATES ARE BASED ON DIMENSIONS OF DATA
C       SPACE, NOT ON AXIS BOUNDARIES.
      CALL JCONWV(2.,2.,0.,VX1,VY1)
      CALL JCONWV(2.+4.,2.+3.5,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

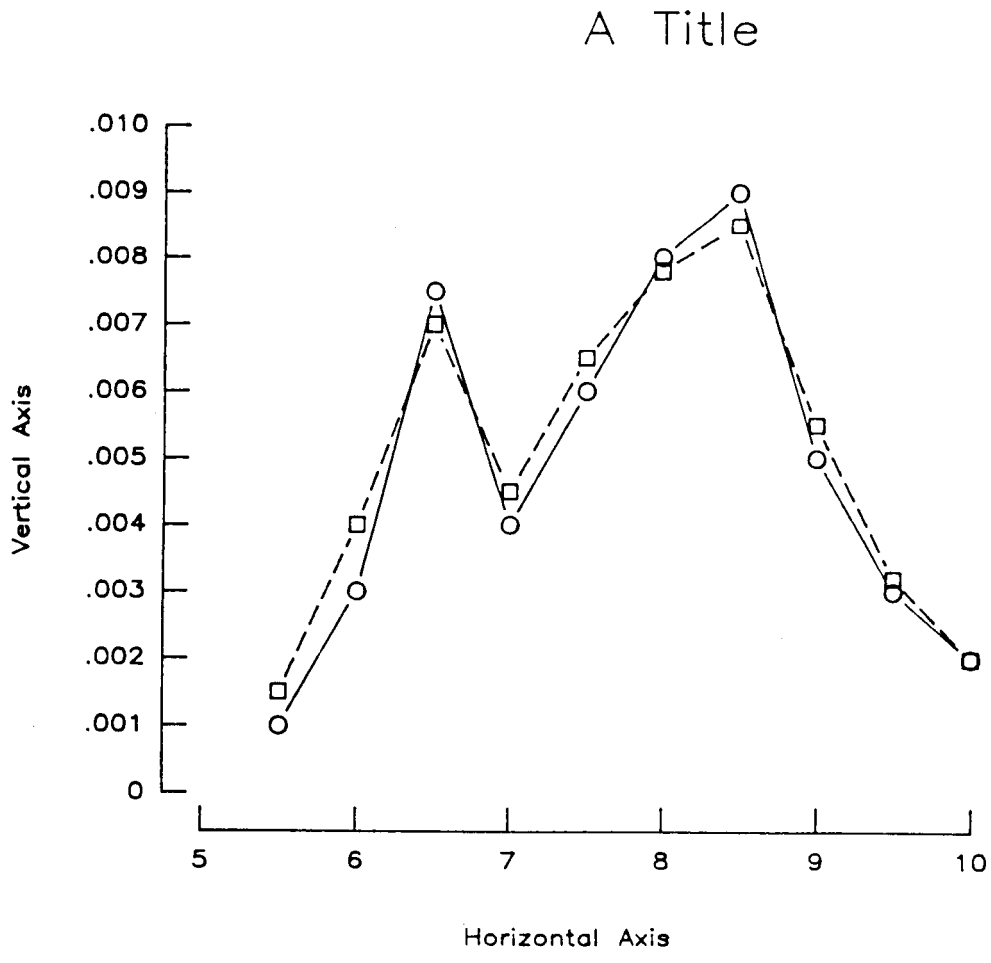


Figure 3-23. Complete program with disjoint axes.

3.4 Steps in the Generation of Bar Charts

This section describes how to generate bar charts, both absolute and additive. In order to generate bar charts, the user must:

- establish page coordinates
- position, describe and draw components (i.e., axes, keys, etc.)
- plot the data (using bars)

3.4.1 Page Coordinates and Describing Components

Since the setting up of page coordinates and the generation of chart components is identical to line charts, the user is referred to Section 3.2 for a complete explanation. After the page coordinates have been established, the components need to be positioned and described. As an example, the following partial code positions and describes a set of axes. Instead of a numeric axis, this chart could require a character horizontal axis with tick mark labels rotated at a 45 degree angle.

```
                    •
                    •
                    •
C NEXT, POSITION THE AXIS AND DESCRIBE ITS CONTENTS.
        CALL JMOVE(XORG,YORG)
C NO HORIZONTAL AXIS LABEL
        CALL CHLAB(' ',0)
C SET HORIZONTAL TICK MARK LABELS ROTATION ANGLE
        CALL CHTROT(45)
        CALL CHAXIC(XLEN,NTICK+2,HTLAB,1,NTICK+2)
C HTLAB IS A CHARACTER ARRAY REPRESENTING MONTHS.
C THE FIRST AND LAST ELEMENTS ARE BLANK TO ENSURE THE
C DATA IS NOT PLOTTED ON THE AXES.
                    •
                    •
                    •
```

Now, define the vertical axes (in this example, a vertical axis on the left and right of the data area).

```
                    .
                    .
                    .
CALL CVLAB('CRUS (THOUSANDS)',1)
CALL CVAXIS(0.,10.,5.,YLEN)
CALL JMOVE(XORG+XLEN,YORG)
CALL CVLAB(' ',0)
CALL CVJUST(0)
CALL CVLABJ(1)
CALL CVAXIS(0.,10.,5.,YLEN)
                    .
                    .
                    .
```

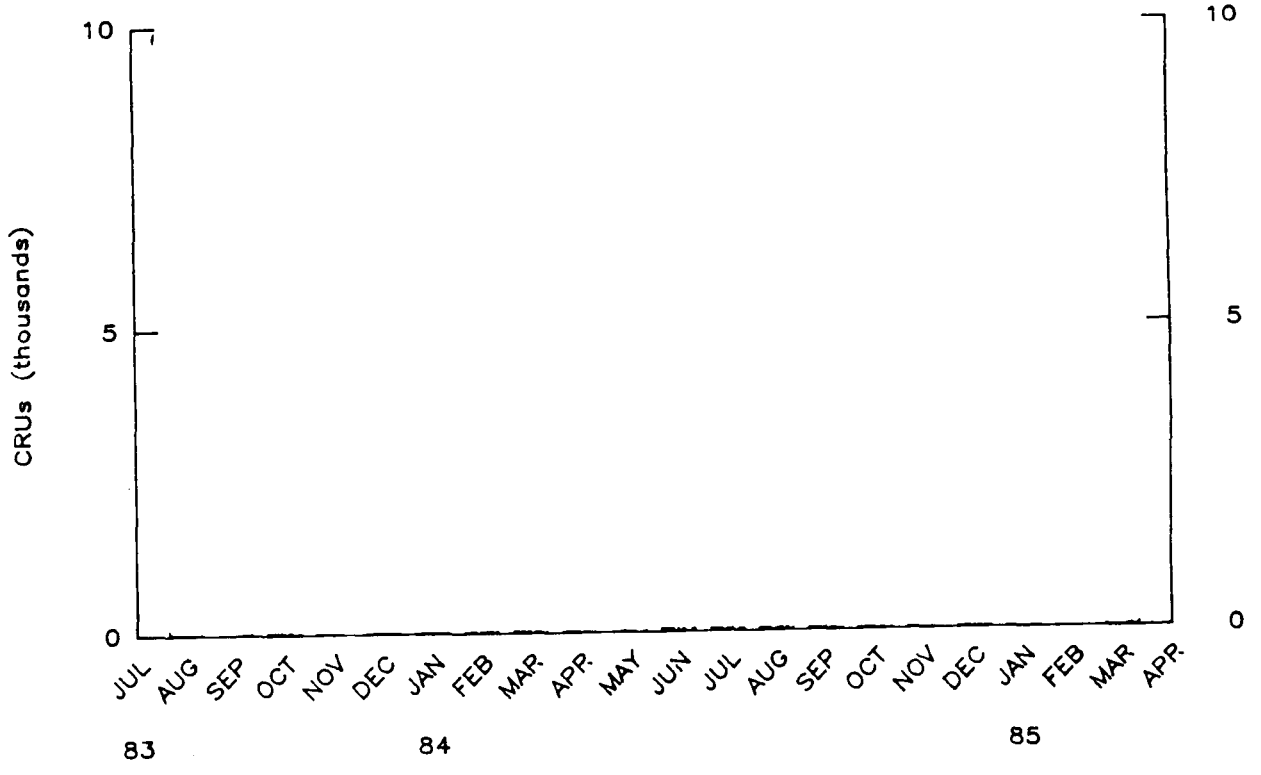


Figure 3-24. Horizontal character axis with rotated tick mark labels.

C-2

3.4.2 Plotting Absolute Bar Chart Data

When the data is absolute, each data set will have a bar depicting data at each horizontal tick mark. The user must ensure that the width of each bar must be small enough, such that data sets can be clearly distinguished between tick marks. Thus, the cumulative width of each tick mark's data set is

(XMAX-XMIN)/REAL(NTICK-1)

where

XMAX-XMIN - represents the length of the axis in terms of the current world coordinate system;

NTICK-1 - represents the number of intervals on the axis.

This would leave a gap before the first tick mark, and after the last tick mark. However, the bars of each tick mark will abut against the bars of another. Thus, we must reduce the width by a reasonable factor (e.g., 2./3.).

```
((XMAX-XMIN)/REAL(NTICK-1))* (2./3.)
```

The width of each data set's bars on a single tick mark would be

```
XWID=((XMAX-XMIN)/REAL(NTICK-1))*(2./3.)
```

divided by the number of data sets (i.e., number of bars per tick mark)

```
XBARW=XWID/REAL(NUMSET)
```

Finally, to plot the data sets we merely setup a loop and perform the appropriate moves and rectangle calls.

```
                    .
                    .
                    .
      DO 2 I=1,NUMSET
            CALL JPIDEX(I,48-I))
C XOFF - OFFSET OF THE DATA SET PER TICK MARK ON THE AXIS
            XOFF=-XWID/2. + REAL(I-1)*XBARW
            DO 2 J=1,NUMPTS
2     CALL JRECT(REAL(J)+XOFF,0.0,
     +           REAL(J)+XOFF+XBARW,Y(I,J))
                    .
                    .
                    .
```

3.4.3 Complete Program Plotting Multiple Absolute Bar Chart Data Sets

This section combines the steps presented in Section 3.4.2 into a complete program. This program demonstrates a bar chart with multiple absolute data sets.

```
      PROGRAM CDB2
C-----------------------------------------------------------------------
C ABSOLUTE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS)
      CHARACTER MONTHS(0:37)*3,VALUES(3)*2
      DATA MONTHS/'   ','JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC','   '/
      DATA VALUES/' 0',' 5','10'/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
```

```
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-------------------------------------------------------------------------
C VERTICAL AXIS - ON LEFT
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIC(YLEN,2,VALUES,1,1)
C VERTICAL AXIS - ON RIGHT
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVAXIC(YLEN,2,VALUES,1,0)
C-------------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-------------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-------------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,10.)
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
```

```
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                  2./3.
      XBARW=XWIDTH/REAL(NUMSET)
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        XOFF=-XWIDTH/2. + REAL(I-1)*XBARW
        DO 2 J=1,NUMPTS
 2    CALL JRECT(REAL(J)+XOFF,0.,REAL(J)+XOFF+XBARW,REVCRU(I,J))
C-------------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

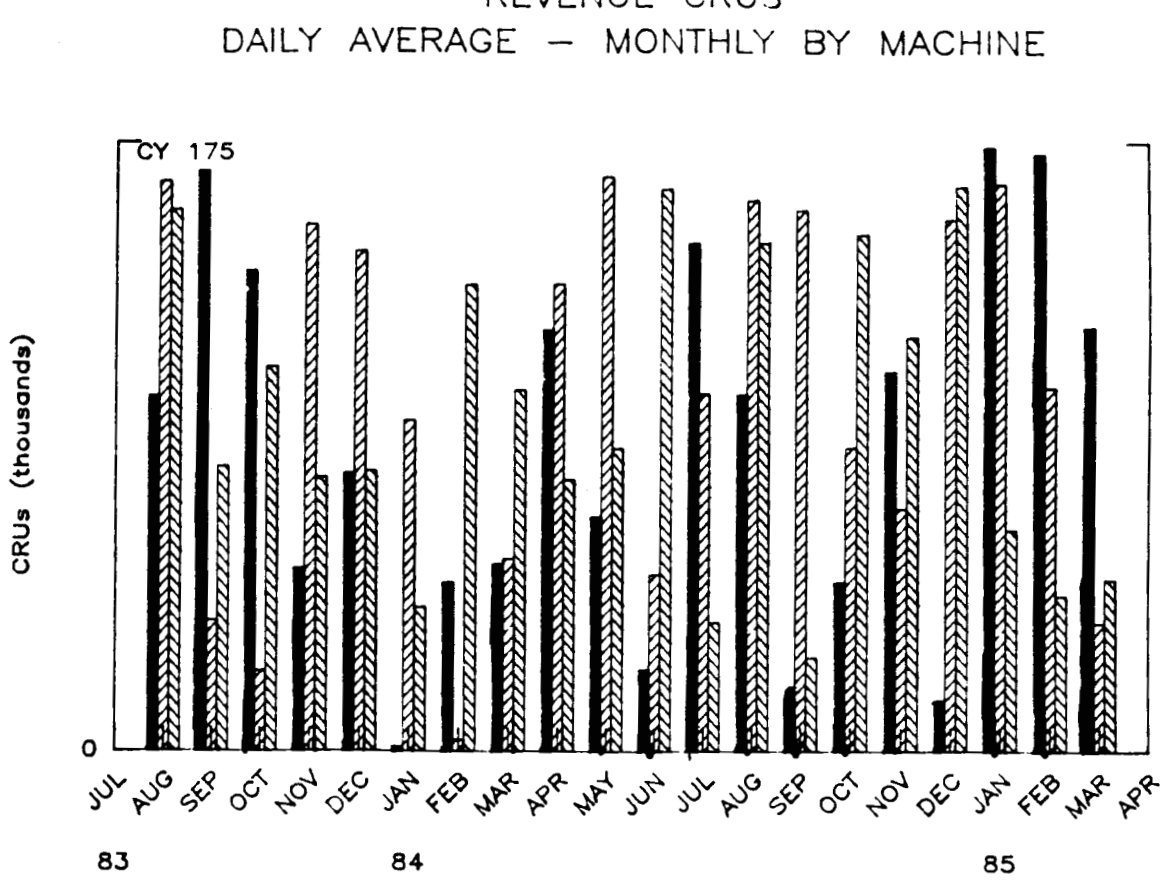


Figure 3-25. Bar chart with multiple absolute data sets.

3.4.4 Plotting Additive Bar Chart Data

With additive data sets, the data from subsequent data sets are stacked (or accumulated) per tick mark. For example, Figure 3-26 shows three data sets represented in an additive fashion. The last two data sets are stacked on each other, on top of the first data set.

The establishment of the axis is the same as in Subsection 3.4.1, noting that the vertical axis bounds must represent the largest additive tick mark values. The following code will determine the bar widths, and plot the data in additive fashion. The accumulative width is computed as in the previous section as

```
XWID=((XMAX-XMIN)/REAL(NTICK-1))*(2./3.)
```

and the midpoint of XWID is calculated as

```
XHALF=XWID/2.
```

Thus, the remaining difference between plotting additive versus absolute is illustrated by the following partial program.

```
                    •
                    •
                    •
C KEEP A RUNNING ACCUMULATION FOR EACH TICK MARK
        DO 1 I=1,NUMPTS
      1 TOTAL(I)=0.0
C FOR THE NUMBER OF DATA SETS (NUMSET) DO
        DO 2 I=1,NUMSET
C SET THE INTERIOR FILL
          CALL JPIDEX(I,48-I)
C FOR THE NUMBER OF DATA VALUES (NUMPTS) DO
          DO 2 J=1,NUMPTS
C PLOT THE DATA AS BARS
            CALL JRECT(REAL(J)-XHALF,TOTAL(J),
     *                 REAL(J)+XHALF,TOTAL(J)+Y(I,J))
C ACCUMULATE THE DATA
      2   TOTAL(J)=TOTAL(J)+Y(I,J)
                    •
                    •
                    •
```

3.4.5 Complete Program Plotting Multiple Additive Bar Chart Data Sets

The following is a complete program using the method described Section 3.4.4 to generate bar charts with multiple additive data sets. Figure 3-26 shows the graphics output generated by the following program.

```
      PROGRAM CDB3
C-----------------------------------------------------------------------
C ADDITIVE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      CHARACTER MONTHS(0:37)*3
      DATA MONTHS/'   ','JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC','   '/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
```

```
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-------------------------------------------------------------------------
C VERTICAL AXIS - LABEL ON LEFT OF AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIS(0.,30.,5.,YLEN)
C VERTICAL AXIS - LABEL ON RIGHT OF AXIS
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVAXIS(0.,30.,5.,YLEN)
C-------------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-------------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-------------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,30.)
C-------------------------------------------------------------------------
```

```
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                    2./3.
      XHALF=XWIDTH/2.
      DO 99 I=1,NUMPTS
 99   TOTAL(I)=0.0
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        DO 2 J=1,NUMPTS
          CALL JRECT(REAL(J)-XHALF,TOTAL(J),
     +               REAL(J)+XHALF,TOTAL(J)+REVCRU(I,J))
 2    TOTAL(J)=TOTAL(J)+REVCRU(I,J)
C-------------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

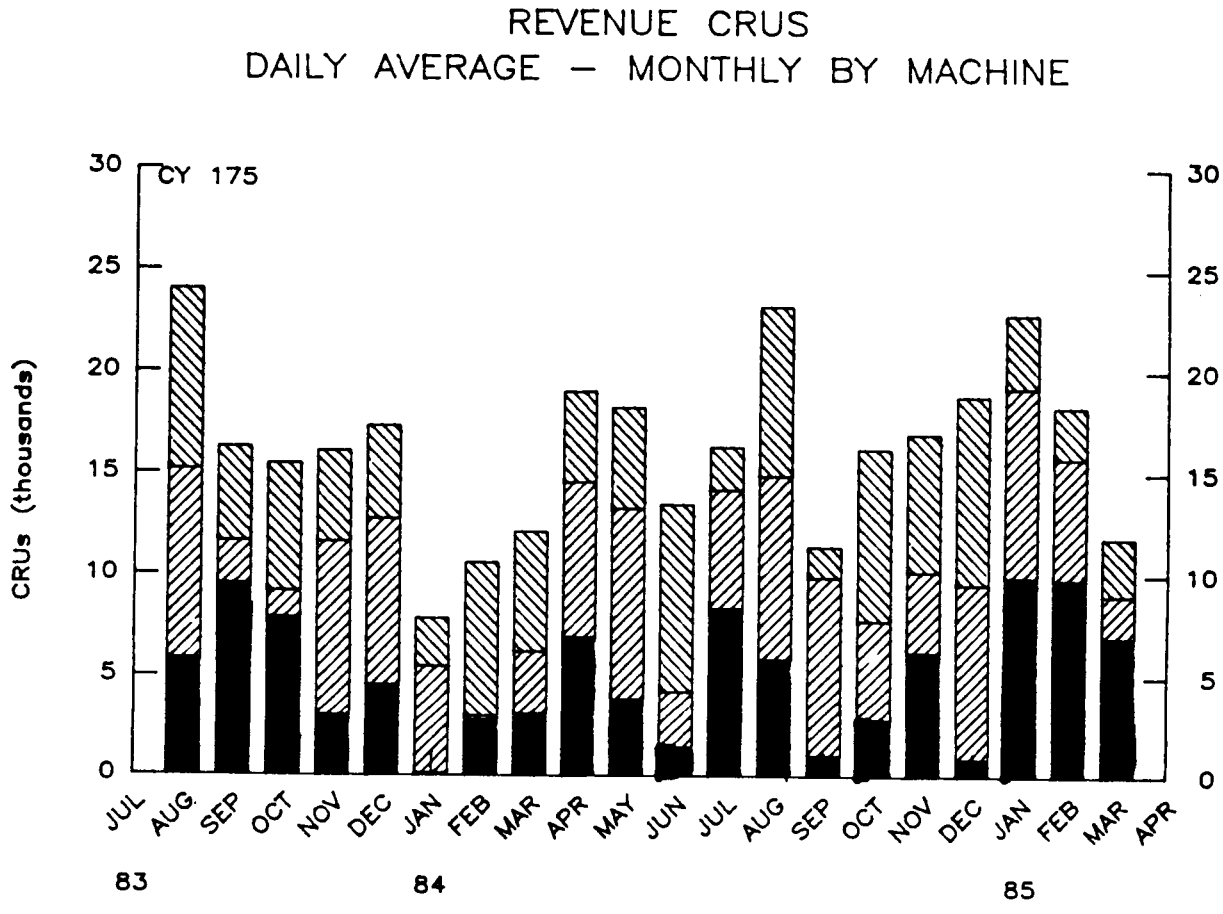


Figure 3-26. Complete additive bar chart.

3.4.6 Plotting Both Absolute and Additive Bar Chart Data

Before attempting this section, it is recommended that Sections 3.4.2 and 3.4.4 are read on plotting absolute and additive data sets onto separate charts. The following section explains how to combine the two types onto one chart. There are two steps that should be taken before writing a program.

Step 1: Know the number of data sets. The number of absolute data sets is very important because it directly affects the bar width and the spacing between each tick mark. In this example, there are a total of three data sets, two of which are absolute data sets.

Step 2: Know how to combine the data sets. The first data set will always be plotted as an absolute bar graph because there is no preceding data. The type of the subsequent data sets to be plotted must be indicated by the user as being additive or absolute (i.e., see ID below). In this example, the second data set will be plotted as an additive bar, while the third will be plotted as an absolute bar.

```
C Allocate storage and initialize variables.

      PARAMETER(NUMSET=3,NUMPTS=20)
      INTEGER ID(NUMSET)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      DATA ID/0,1,0/

C ID[1] = 0      means an absolute data set
C ID[2] = 1      means an additive data set
C ID[3] = 0      means an absolute data set


C Determine the number of additive data sets.

      ICT=0
      DO 101 I=2,NUMSET
101          IF(ID(I).EQ.1)ICT=ICT+1

C Note: NUMSET-ICT equals the number of absolute data sets.
C Open segment and plot bars.

      CALL JOPEN
      CALL JPINTR(1)
```

```
C The width of the bars at a single tick mark is determined
C by the length of the axis (in data coordinates) divided by
C the number of absolute data sets, times (2./3.).
C          XWIDTH = ((NUMPTS+1) / REAL(NUMPTS+2-1)) * (2./3.)
C NOTE:If the world is from 0 to a value (with an increment
C of 1, as is typical for bar charts), then XWIDTH = 2./3.
C Each bar width (XBARW) is the XWIDTH divided by the number
C of absolute data sets.

     XWIDTH=2./3.
     XBARW=XWIDTH/REAL(NUMSET-ICT)
     DO 99 I=1,NUMPTS
99          TOTAL(I)=0.0


C ICT is the current absolute data set counter, and
C determines the offset (based on which absolute data set).

     ICT=1
     DO 2 I=1,NUMSET
          CALL JPIDEX(I,48-I)
          XOFF=-XWIDTH/2. + REAL(ICT-1)*XBARW

C If ID[i]=1 (absolute data set), then increment ICT counter.

     IF(ID(I).EQ.1)ICT=ICT+1
     DO 2 J=1,NUMPTS
          DO JRECT(REAL(J)+XOFF,TOTAL(J),REAL(J)+XOFF+XBARW,
    +               TOTAL(J)+REVCRU(I,J))


C Check to see if ID[i+1] = 1 (i.e., an additive data set),
C then accumulate TOTAL.

          IF(ID(I+1).EQ.1)THEN
              TOTAL(J)=TOTAL(J)+REVCRU(I,J)
          ELSE
              TOTAL(J)=0.
          ENDIF
2    CONTINUE
     CALL JCLOSE
```

Figure 3-27. Partial program with absolute and additive data sets.

3.4.7 Complete Program Plotting Multiple Absolute and Additive Bar Chart Data Sets

The following program shows how to generate a bar chart with multiple absolute and additive data sets. This program combines the steps presented in the last section in context. The resultant graphical output is found in Figure 3-27.

```
      PROGRAM CDB1
C-----------------------------------------------------------------------
C ADDITIVE AND ABSOLUTE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      INTEGER ID(NUMSET)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      CHARACTER MONTHS(0:37)*3
      DATA ID/0,1,0/
      DATA MONTHS/'   ','JAN','FEB','MAR','APR','MAY','JUN',
     +            'JUL','AUG','SEP','OCT','NOV','DEC',
     +            'JAN','FEB','MAR','APR','MAY','JUN',
     +            'JUL','AUG','SEP','OCT','NOV','DEC',
     +            'JAN','FEB','MAR','APR','MAY','JUN',
     +            'JUL','AUG','SEP','OCT','NOV','DEC','   '/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
```

```
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-------------------------------------------------------------------------
C VERTICAL AXIS - ON LEFT OF PLOT
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      ICT=0
      DO 101 I=2,NUMSET
101     IF (ID(I).EQ.1) ICT=ICT+1
      VMAX=10.*REAL(NUMSET-ICT)
      CALL CVAXIS(0.,VMAX,5.,YLEN)
C VERTICAL AXIS - ON RIGHT OF PLOT
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVAXIS(0.,VMAX,5.,YLEN)
C-------------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-------------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-------------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,VMAX)
```

```
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                    2./3.
      XBARW=XWIDTH/REAL(NUMSET-ICT)
      DO 99 I=1,NUMPTS
99      TOTAL(I)=0.0
      ICT=1
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        XOFF=-XWIDTH/2. + REAL(ICT-1)*XBARW
        IF(ID(I).EQ.1)ICT=ICT+1
        DO 2 J=1,NUMPTS
          CALL JRECT(REAL(J)+XOFF,TOTAL(J),REAL(J)+XOFF+XBARW,
     +     TOTAL(J)+REVCRU(I,J))
        IF (ID(I+1).EQ.1) THEN
          TOTAL(J)=TOTAL(J)+REVCRU(I,J)
        ELSE
          TOTAL(J)=0.
        END IF
2     CONTINUE
C-------------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

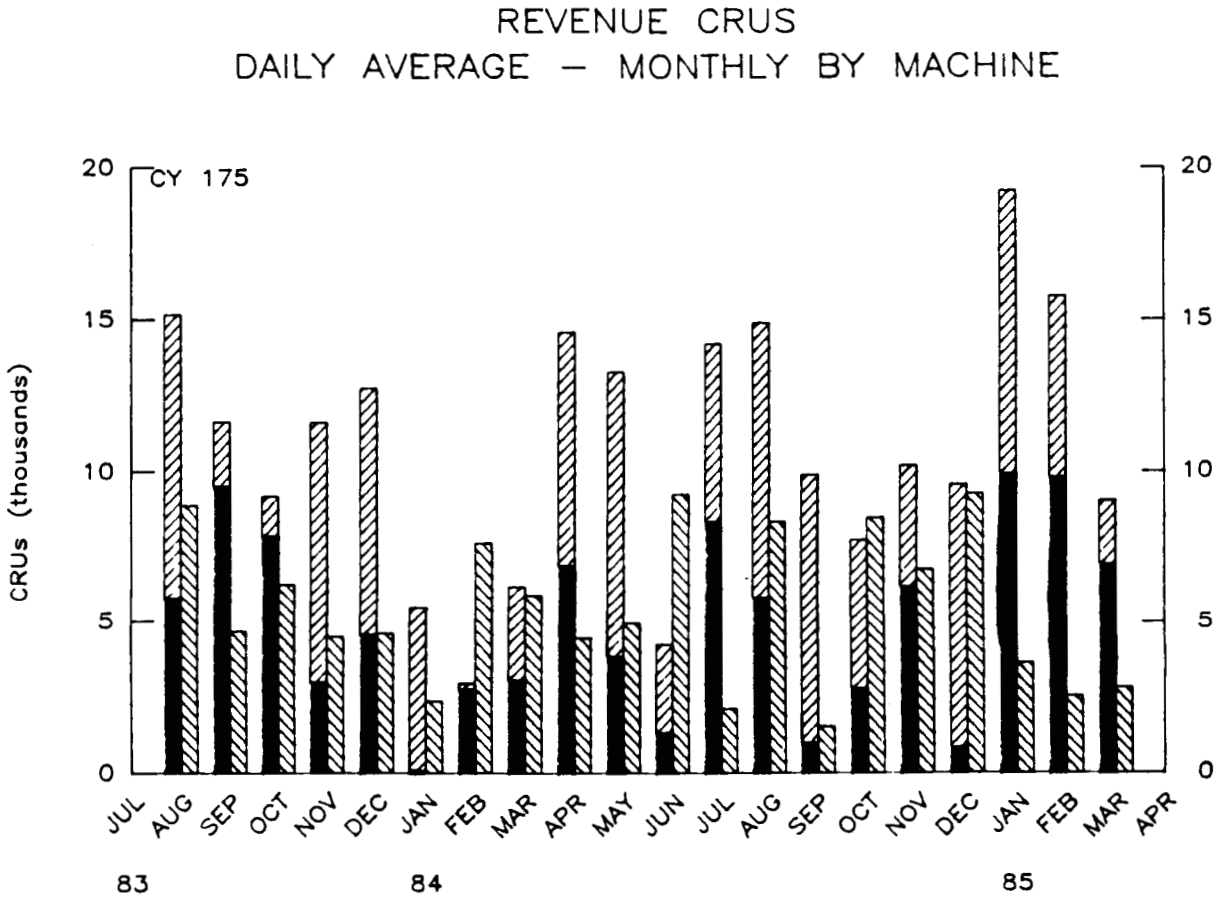


Figure 3-28. Complete program with absolute and additive data sets.

3.4.8 Adding a Key to Bar Charts

The necessary steps to generate a bar chart with multiple data sets has been demonstrated. The next evolutionary step is the addition of a key. A key is a list of words or phrases giving an explanation of symbols or abbreviations (Figure 2-4 indicates a key in a XY line chart).

In order to add a key, the following four steps are necessary:

1) declare and allocate key related variables;
2) initialize the key related variables by calling CKEYIN;
3) for each data set to be represented in the key, call CKEYLB. This call should be performed when the data is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active; and
4) finally to plot the key, call CKEYPL. This call should be performed in page coordinates. Thus, the window and viewport may have to be reset.

Step 1 - Declare and allocate variables

A key is a series of entries (lines), where each entry is stored internally as a collection of attributes. The user is required to declare and allocate storage for these attributes. Specifically, two variables are needed, ISTORE (storage array for attributes) and KEYCHR (storage array for line labels). The dimensions of these arrays should be large enough to accommodate the largest number of data sets plotted at one time. See the description of the routine CKEYPL in Appendix A for a more detailed description.

```
                    •
                    •
                    •
      PARAMETER(NLINS=4,NCOLS=1)
C NLINS - the maximum number of lines in the key
C NCOLS  - the number of columns for an entry
C *20    - the largest character string per entry
C          (including mnemonics)
C (Note, in this example 20 is used, could be any value.)

      CHARACTER KEYCHR(NCOLS,NLINS)*9
      INTEGER   ISTORE(15,NLINS)
                    •
                    •
                    •
```

Step 2 - Initialize the key related variables by calling CKEYIN. This initialization can be performed anywhere, at anytime (i.e., the variables passed to CKEYIN will be reset).

```
                                    •
                                    •
                                    •
C Set window to match data coordinates, and plot within
C boundaries of the axes

    CALL JWINDO(0.,REAL(NUMPTS+1),0.,10.)

C 0.,REAL(NUMPTS+1) - the data coordinate boundaries in the
C                     X-direction.  Set 1 to NUMPTS, plus an
C                     extra point on each end of the scale.
C 0.,10.            - the data coordinate boundaries in the
C                     Y-direction

    CALL JVPORT(VX1,VX2,VY1,VY2)
    CALL JOPEN
    CALL CKEYIN(ISTORE, KEYCHR,NLINS,NCOLS,NTLINS)
                                    •
                                    •
                                    •
```

Step 3 - For each data set to be represented in the key, call CKEYLB. This call should be performed when the data is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active. The key related variables will be assigned values to be plotted when all the key entries have been obtained. This typically requires the plotting of the data (in terms of data coordinates). Note, a "-4" is passed to CKEYLB to indicate a rectangle (instead of a line) is to be used in the key.

```
C Plot the first data set
    DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        CALL CKEYLB(ISTORE,KEYCHR,-4,KEYLAB(I))
        DO 2 J=1,NUMPTS
            CALL JRECT(REAL(J)-XHALF,TOTAL(J),
   +                   REAL(J)+XHALF,TOTAL(J)+REVCRU(I,J))
2   TOTAL(J)=TOTAL(J)+REVCRU(I,J)
    CALL JCLOSE
                                    •
                                    •
                                    •
```

Step 4 - Finally to plot the key, call CKEYPL. When all the key entries have been made, a call to CKEYPL will plot the key based on the attributes stored in ISTORE and KEYCHR (and a few internal variables). The call to CKEYPL should be performed in page coordinates, and since most data is plotted in data coordinates (where the attributes are saved by CKEYLB), the window and viewport will probably have to be reset.

```
                                    •
                                    •
                                    •
C Now output the key
C Set the window and viewport for page coordinates
     CALL JVPORT(...)
     CALL JWINDO(...)
     BPOS(1)=XORG+XLEN
     BPOS(2)=YORG+YLEN
     CALL JOPEN
     CALL JSIZE(0.1,0.1*1.25)
     CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE',
    +     'C[BLC]OLUMN TITLE',BPOS,5,JCOL,1)
     CALL JCLOSE
                                    •
                                    •
                                    •
C Terminate graphics
     CALL JPAUSE(IDEV)
     CALL JFRAME
     CALL JDEVOF(IDEV)
     CALL JDEND(IDEV)
     CALL JEND
                                    •
                                    •
                                    •
```

3.4.9 Complete Bar Chart Program with a Key

The following program is used to demonstrate how to embed these four steps into a complete program. The key related statements have been additionally commented for clarity. Figure 3-29 shows the graphics output from the following program.

```
      PROGRAM CDB4
C-----------------------------------------------------------------------
C ADDITIVE BARS - MULTIPLE DATA SETS, WITH A LEGEND.
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0)
      PARAMETER(XBEG=1.5,YBEG=1.5)
      PARAMETER(XLEN=8.5,YLEN=5.0)
      PARAMETER(CSIZE=.15)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      PARAMETER(NLINES=4,NCOLS=1,NTLINES=1)
      CHARACTER KEYCHR(NCOLS,NLINES)*9,KEYLAB(NLINES)*9
      INTEGER ISTORE(15,NLINES),JCOL(1)
      REAL BPOS(4)
      CHARACTER MONTHS(12)*3,VALUES(3)*2
      DATA MONTHS/'JAN','FEB' 'MAR','APR','MAY','JUN',
     +            'JUL','AUG','SEP','OCT','NOV','DEC'/
      DATA VALUES/' 0',' 5','10'/
      DATA KEYLAB(1)/'-AVAIL+PM'/,KEYLAB(2)/'-AVAIL    '/
      DATA KEYLAB(3)/'-USE      '/,KEYLAB(4)/'-PROD     '/
      DATA JCOL/1/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*3.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      CALL CDEBUG(1)
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
```

```
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHLABJ(1)
      CALL CHAXIC(XLEN,2,' ',0,NUMPTS)
C MONTHS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-2.*CSIZE*1.25)
      CALL JJUST(3,2)
      CALL CSETBP(90.)
      IMONTH1=8
      DO 1 KI=1,NUMPTS
        IMONTH=KI+IMONTH1-1
        IMONTH=MOD(IMONTH,12)
        IF(IMONTH.EQ.0)IMONTH=12
        CALL JRMOVE(XINCR,0.)
 1    CALL JHSTRG(MONTHS(IMONTH))
      CALL CSETBP(0.)
      CALL JSETDB(0)
C YEARS
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL CSETBP(0.)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVTICJ(2)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
C-----------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
```

```
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-----------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,X1,Y1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,X2,Y2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(X1,X2,Y1,Y2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,10.)
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINES,NCOLS,NTLINES)
       CALL JPINTR(1)
       XWIDTH=2./3.
       XHALF=XWIDTH/2.
       DO 99 I=1,NUMPTS
 99    TOTAL(I)=0.0
       CALL JSIZE(CSIZE,CSIZE*1.25)
       DO 2 I=1,NUMSET
         CALL JPIDEX(I,47-(I-1))
         CALL CKEYLB(ISTORE,KEYCHR,-4,KEYLAB(I))
         DO 2 J=1,NUMPTS
           CALL JRECT(REAL(J)-XHALF,TOTAL(J),
     +                REAL(J)+XHALF,TOTAL(J)+REVCRU(I,J))
 2     TOTAL(J)=TOTAL(J)+REVCRU(I,J)
       CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE AREA
       CALL JSETDB(0)
       CALL JVPORT(-1.,1.,-.773,.773)
       CALL JWINDO(0.,10.,0.,7.73)
       BPOS(1)=XBEG+XLEN
       BPOS(2)=YBEG+YLEN
       CALL JOPEN
       CALL JSIZE(CSIZE,CSIZE*1.25)
C      CALL CKLPLN(.08)
       CALL CKEYPL(ISTORE,KEYCHR,' ',' ',BPOS,3,JCOL,0)
       CALL JCLOSE
C---------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

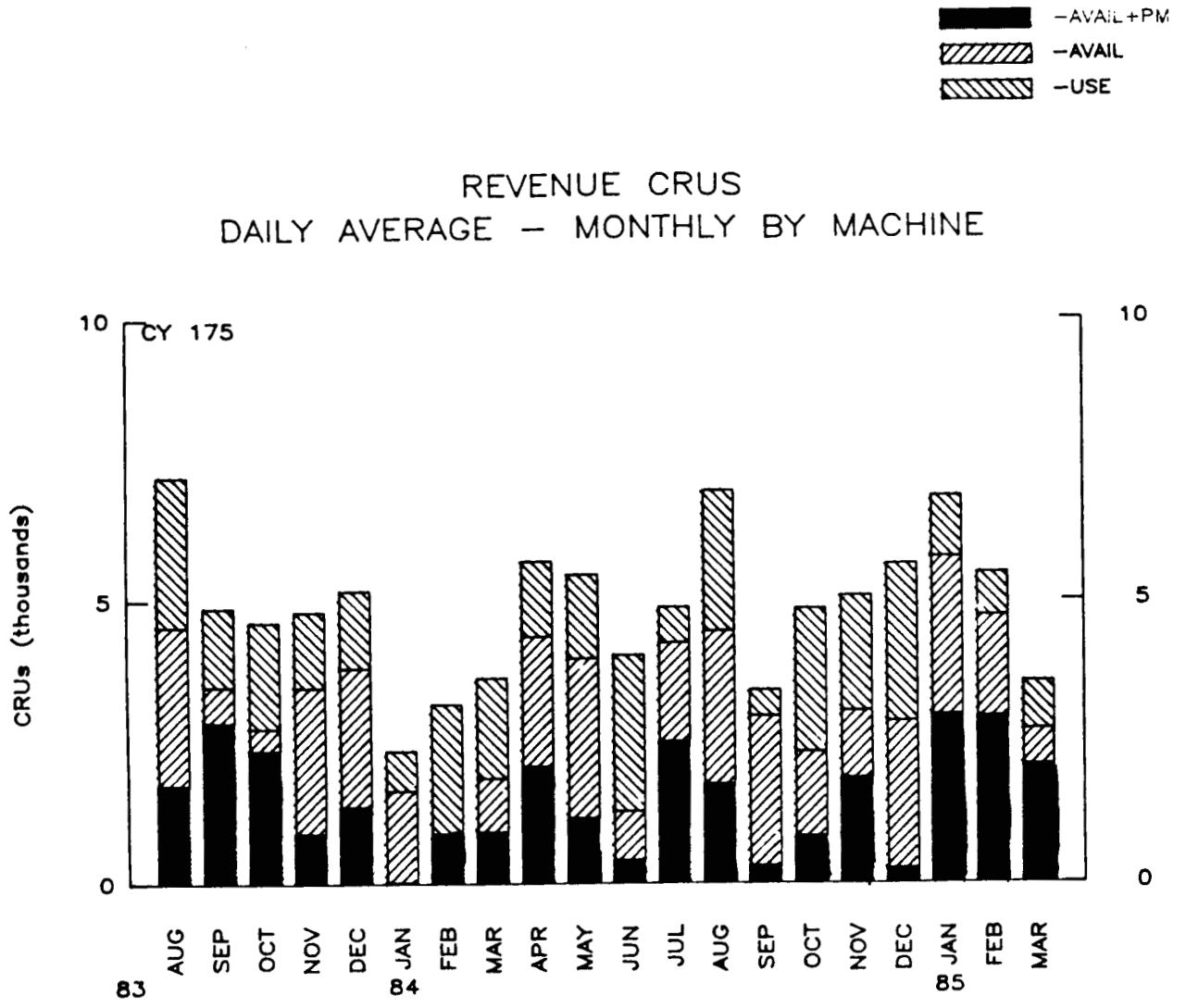


Figure 3-29. Complete bar chart with a key.

3.5 Steps in the Generation of Pie Charts

This section discusses how to generate pie charts. The Low-Level routines of the CGL contain no routines for the generation of pie charts. This is because a pie chart consists of a set of sectors (possibly with associated text). Thus, the components of pie charts are so primitive, that little advantage can be obtained by calling any routines other than those supplied by the underlying graphics package.

This section will describe a method of calling the underlying graphics package, and take advantage of some CGL routines to perform supplementary functions (i.e., keys). The major steps in the generation of pie charts consists of:

- setting page coordinates
- determining the placement of the center of the pie
- drawing the pie segments (possibly with segment labels)

3.5.1 Page Coordinates and Describing Components

A convenient approach to aid the user in designing a chart is to establish a coordinate system, referred to here as the page coordinate system, which easily enables the user to arrange and describe the chart components. The page coordinates can be of any dimension and represent any unit. Since publication-quality charts are often requested, the page coordinates described in the document are typically in terms of inches to coincide with the postprocessors (plotters).

A call to JVSPAC will setup the viewspace and viewport. A call to JWINDO defines the viewing window.

```
CALL JVSPAC(-1.,1.,-1.,1.)
CALL JWINDO(0.,10.,0.,10.)
```

Once the page coordinates have been set, the user can position and describe objects in terms of the page coordinates. For example, the following code will output a graphics string as a title, output a graphics string in the bottom left hand corner of the page as well as the bottom right hand corner of the page (as shown in Figure 3-30).

```
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.4,.4)
      CALL JFONT(3)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.2)
      CALL JHSTRG('MATERIALS')
      CALL JMOVE(.1,.8)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.8)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
```

Figure 3-30. Example of page layout for a pie chart.

3.5.2 Plotting Data for Pie Charts

Once the page coordinates have been established, the center of the pie must be decided and the pie segments drawn. In order to get a true circle for the pie, the program should ensure consistent aspect ratios for both the current viewport and current window in which the pie is drawn.

In order to plot the data, the sectors of the pie must be plotted individually. To draw pie segments, the following pie characteristics (see Figure 3-31) must be supplied:

- pie center (i.e., X, Y, 0.0)
- pie radius (RAD1)
- number of pie segments (NUMSEG)
- segment attributes (see JPIDEX)
- segment angles (see variables A0 and A1 of JSECTR)
- segment text attributes (i.e., JSIZE, JFONT, JJUST, etc.)
- segment text position (based on RADIUS and quadrant)

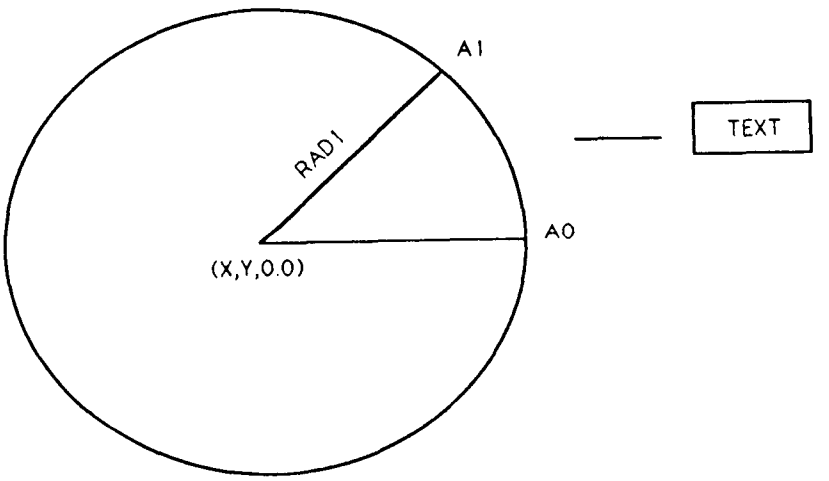


Figure 3-31. Variables associated with pie chart generation.

The following is a partial program explaining the steps involved in plotting pie chart data.

```
C-------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
C THE FIRST PIE SEGMENT STARTS AT 0.0 DEGREES.
      START=0.
C SET THE TEXT ATTRIBUTES
      CALL JSIZE(.2,.2)
      CALL JFONT(1)
C LOOP FOR ALL PIE SEGMENTS (NUMSEG IS 8)
      DO 5000 I=1,8
C COMPUTE BEGINNING AND ENDING ANGLES
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        START=A1
C SET PIE SEGMENT ATTRIBUTES, AND DRAW SEGMENTS
        CALL JPIDEX(I,I+30)
        CALL JSECTR(5.,5.,0.,RAD1,0,A0,A1)
C CALCULATE SEGMENT MIDPOINT, AND ASSOCIATED POSITION OF SEGMENT TEXT
        AMID=(A1-A0)/2.+A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
C MOVE AND DRAW A LINE FROM THE PIE SEGMENT TO THE TEXT
        CALL JMOVE(5.+RAD2*C1,5.+RAD2*S1)
        CALL JDRAW(5.+RAD3*C1,5.+RAD3*S1)
C DETERMINE THE PIE SEGMENT TEXT JUSTIFICATION (BASED ON ANGLE)
        IF (AMID .LT.  45. .OR.  AMID .GE. 315.) CALL JJUST(1,2)
        IF (AMID .GT.  45. .AND. AMID .LT. 135.) CALL JJUST(2,1)
        IF (AMID .GE. 135. .AND. AMID .LT. 225.) CALL JJUST(3,2)
        IF (AMID .GE. 225. .AND. AMID .LT. 315.) CALL JJUST(2,3)
C POSITION TEXT AND DRAW TEXT
        CALL JMOVE(5.+RAD4*C1,5.+RAD4*S1)
 5000 CALL JHSTRG(LABEL(I))
```

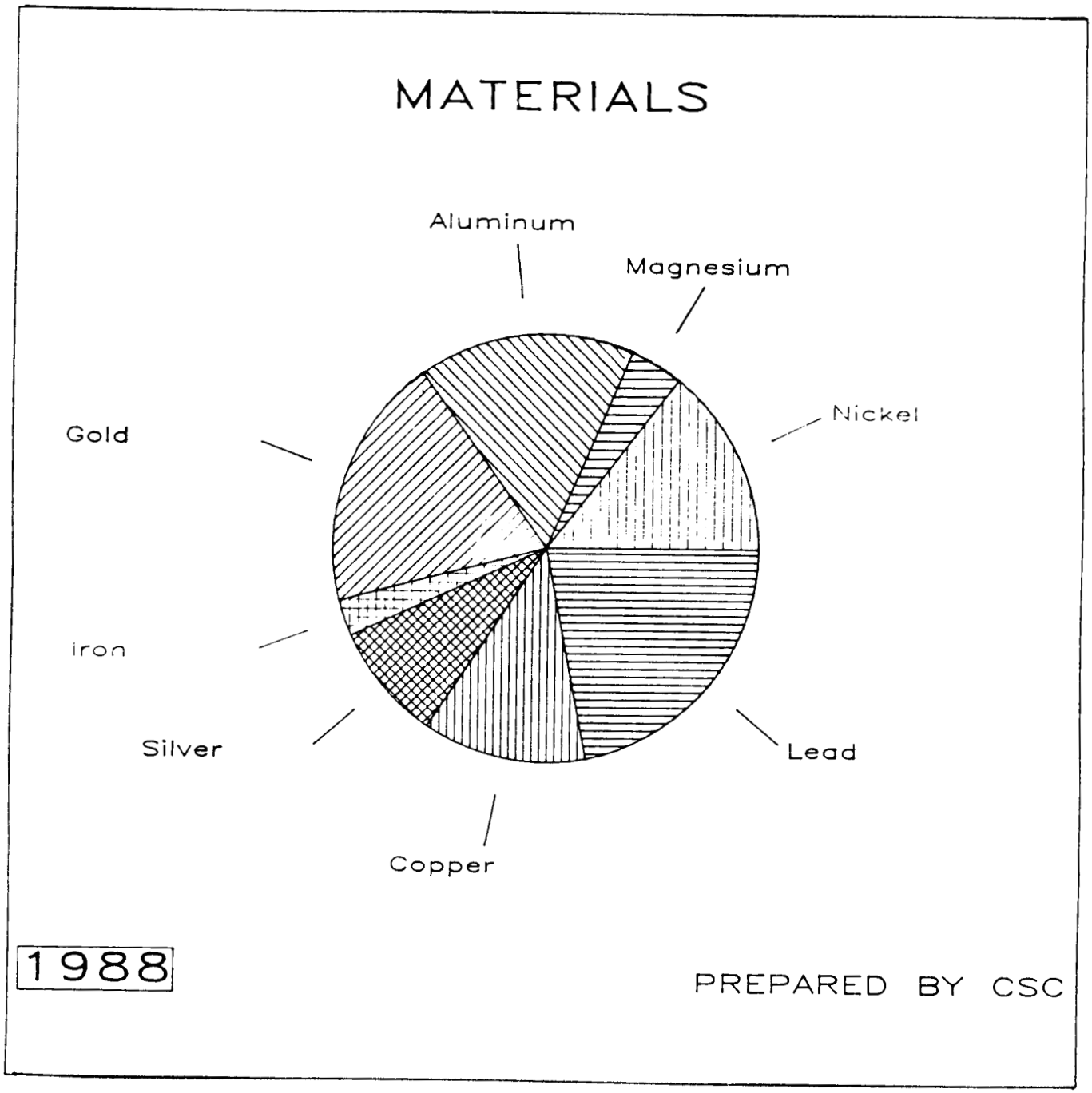


Figure 3-32. Plotting pie data.

3.5.3 Complete Program Plotting a Pie Chart

The previous subsections walked through the steps and described a method of generating a pie chart. The following is a complete program using the method previously described in context. The corresponding graphics output can be found in Figure 3-33.

```
      PROGRAM CDP1
C-------------------------------------------------------------
C BASIC PIE CHART
C-------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C-------------------------------------------------------------
C INITIALIZE DATA
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.25,2.,2.35,2.85,2.95/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/21.,6.,24.,28./
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/4.,13.,18.,32./
C
      DATA YEAR /'1988'/
C
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'G[BLC]OLD'/
      DATA LABEL(5)/'I[BLC]RON'/
      DATA LABEL(6)/'S[BLC]ILVER'/
      DATA LABEL(7)/'C[BLC]OPPER'/
      DATA LABEL(8)/'L[BLC]EAD'/
C-------------------------------------------------------------
C INITIALIZE DI-3000
       CALL JBEGIN
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-------------------------------------------------------------
C FILL POLYGONS BY DEFAULT.
      CALL JDPINT(1)
```

```
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.4,.4)
      CALL JFONT(3)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.2)
      CALL JHSTRG('MATERIALS')
      CALL JMOVE(.1,.8)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.8)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      START=0.
      CALL JSIZE(.2,.2)
      CALL JFONT(1)
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        START=A1
        CALL JPIDEX(I,I+30)
        CALL JSECTR(5.,5.,0.,RAD1,0,A0,A1)
C
```

```
         AMID=(A1-A0)/2.+A0
         ARAD=AMID*DTORAD
         C1=COS(ARAD)
         S1=SIN(ARAD)
         CALL JMOVE(5.+RAD2*C1,5.+RAD2*S1)
         CALL JDRAW(5.+RAD3*C1,5.+RAD3*S1)
C
         IF (AMID.LT.45. .OR. AMID .GE. 315.) CALL JJUST(1,2)
         IF (AMID .GT. 45. .AND. AMID .LT. 135.) CALL JJUST(2,1)
         IF (AMID .GE. 135. .AND. AMID .LT. 225.) CALL JJUST(3,2)
         IF (AMID .GE. 225. .AND. AMID .LT. 315.) CALL JJUST(2,3)
C
         CALL JMOVE(5.+RAD4*C1,5.+RAD4*S1)
 5000 CALL JHSTRG(LABEL(I))
C-------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-------------------------------------------------------------
C PAUSE TO VIEW CHART.
      CALL JPAUSE(IDEV)
C-------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
      STOP
      END
```

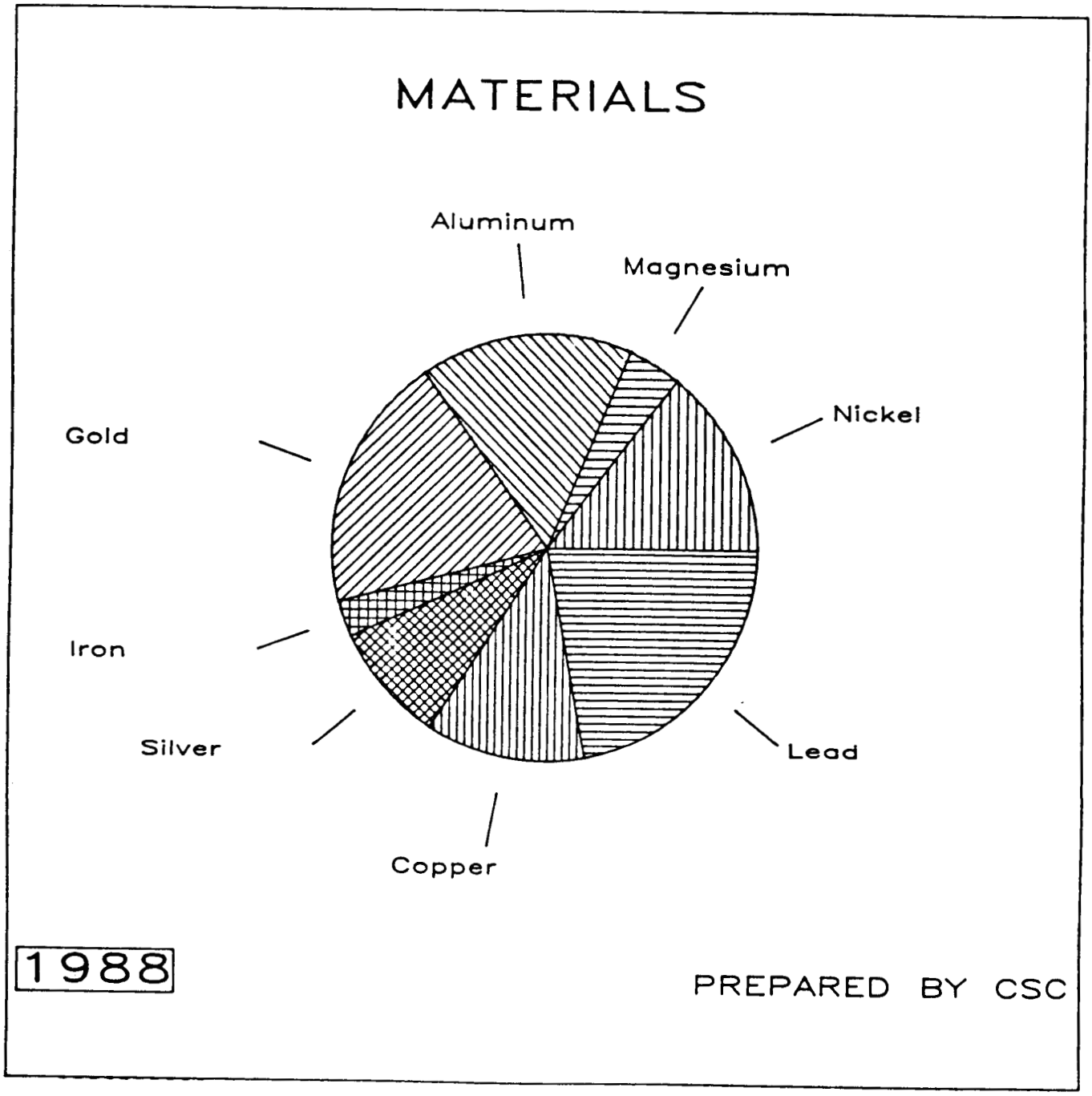


Figure 3-33. Complete pie chart program.

3.5.4 Steps in Plotting a Pie Chart with a Key

In context, a simple pie chart has been demonstrated. The next evolutionary step is the addition of a key. A key is a list of words or phrases giving an explanation of symbols or abbreviations (Figure 2-6 indicates a key in a pie chart).

In order to add a key, four steps are necessary:

1) declare and allocate key related variables;
2) initialize the key related variables by calling CKEYIN;
3) for each data set to be represented in the key, call CKEYLB. This call should be performed when the pie segment is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active;
4) finally to plot the key, call CKEYPL. This call should be performed in page coordinates.

Step 1 - Declare and allocate variables

A key is a series of entries (lines), where each entry is stored internally as a collection of attributes. The user is required to declare and allocate storage for these attributes. Specifically, two variables are needed, ISTORE (storage array for attributes) and KEYCHR (storage array for line labels). The dimensions of these arrays should be large enough to accommodate the largest number of data sets plotted at one time. See the description of the routine CKEYPL in Appendix A for a more detailed description.

```
                    •
                    •
                    •
      PARAMETER(NLINS=8)
C NLINS  - the maximum number of lines in the key
C *9     - the largest character string per entry
C          (including mnemonics)
C (Note, in this example 20 is used, could be any value.)

      CHARACTER KEYCHR(1,NLINS)*9
      INTEGER   ISTORE(15,NLINS)
                    •
                    •
                    •
```

Step 2 - Initialize the key related variables by calling CKEYIN. This initialization can be performed anywhere, at anytime (i.e., the variables passed to CKEYIN will be reset).

```
                                    •
                                    •
                                    •
C Set window to match page coordinates

    CALL JWINDO(0.,10.,0.,10.)
    CALL JOPEN
    CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
                                    •
                                    •
                                    •
```

Step 3 - For each data set to be represented in the key, call CKEYLB. This call should be performed when the data is to be drawn, since this routine will save off various DI-3000 and CGL attributes currently active. The key related variables will be assigned values to be plotted when all the key entries have been obtained. This typically requires the plotting of the data (in terms of data coordinates).

```
                                    •
                                    •
                                    •
C Plot the pie segments
    DO 1 I=1,NUMSEG
          .
          .
          .
C Set pie segment attributes
       CALL JPIDEX(I,I+30)
       CALL JSECTR(...)
          .
          .
       CALL CKEYLB(ISTORE,KEYCHR,-4,LABEL(I))
1   CONTINUE
                                    •
                                    •
                                    •
```

Step 4 - Finally to plot the key, call CKEYPL. When all the key entries have been made, a call to CKEYPL will plot the key based on the attributes stored in ISTORE and KEYCHR (and a few internal variables).

```
                                    •
                                    •
                                    •
C Now output the key
C Set the key position.
     BPOS(1)=...
     BPOS(2)=...
C Plot the key
     CALL CKEYPL(ISTORE,KEYCHR,' ',' ',BPOS,5,1,0)
     CALL JCLOSE
                                    •
                                    •
                                    •
C Terminate graphics
     CALL JPAUSE(IDEV)
     CALL JFRAME
     CALL JDEVOF(IDEV)
     CALL JDEND(IDEV)
     CALL JEND
                                    •
                                    •
                                    •
```

3.5.5 Complete Program Plotting a Pie Chart with a Key

The following program is used to demonstrate how to embed these four steps into a complete program. Figure 3-34 shows the graphics output from the following program.

```
      PROGRAM CDP3
C-------------------------------------------------------------
C BASIC PIE CHART WITH A LEGEND
C-------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL BPOS(4)
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C
      CHARACTER KEYCHR(1,8)*15
      INTEGER ISTORE (15,8)
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.5,4.,4.7,5.7,5.9/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/2.1,.6,2.4,2.8/
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/.4,1.3,1.8,3.2/
C
      DATA YEAR /'1988'/
C
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'I[BLC]RON'/
      DATA LABEL(5)/'T[BLC]IN'/
      DATA LABEL(6)/'C[BLC]OPPER'/
      DATA LABEL(7)/'G[BLC]OLD'/
      DATA LABEL(8)/'Z[BLC]INC'/
C-------------------------------------------------------------
C INITIALIZE DI-3000
      CALL JBEGIN
      CALL CBEGIN
C-------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-------------------------------------------------------------
```

```
C FILL POLYGONS BY DEFAULT.
      CALL JDPINT(1)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.6,.6)
      CALL JFONT(5)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.1)
      CALL JHSTRG('CGL PIE GRAPH')
      CALL JMOVE(.1,.5)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.5)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-----------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-----------------------------------------------------------------
C DESIGN THE PIE TO BE 50% SMALLER
       RAD1=RAD1-(RAD1*.50)
       RAD2=RAD2-(RAD2*.50)
       RAD3=RAD3-(RAD3*.50)
       RAD4=RAD4-(RAD4*.50)
C-----------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-----------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,8,1,1)
      START=5.
      CALL JSIZE(.3,.3)
      CALL JFONT(1)
```

```
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        CALL JPIDEX(I,I+30)
        CALL JSECTR(3.,5.,3.,RAD1,0,A0,A1)
C
        START=A1
        AMID=(A1-A0)/2.+A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
 5000   CALL CKEYLB(ISTORE,KEYCHR,-4,LABEL(I))
      BPOS(1)=9.5
      BPOS(2)=8.
      CALL CKEYPL(ISTORE,KEYCHR,' ',' ',BPOS,5,1,0)
C-------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-------------------------------------------------------------
C PAUSE FOR OPERATOR ACTION.
      CALL JPAUSE(IDEV)
C-------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
      STOP
      END
```

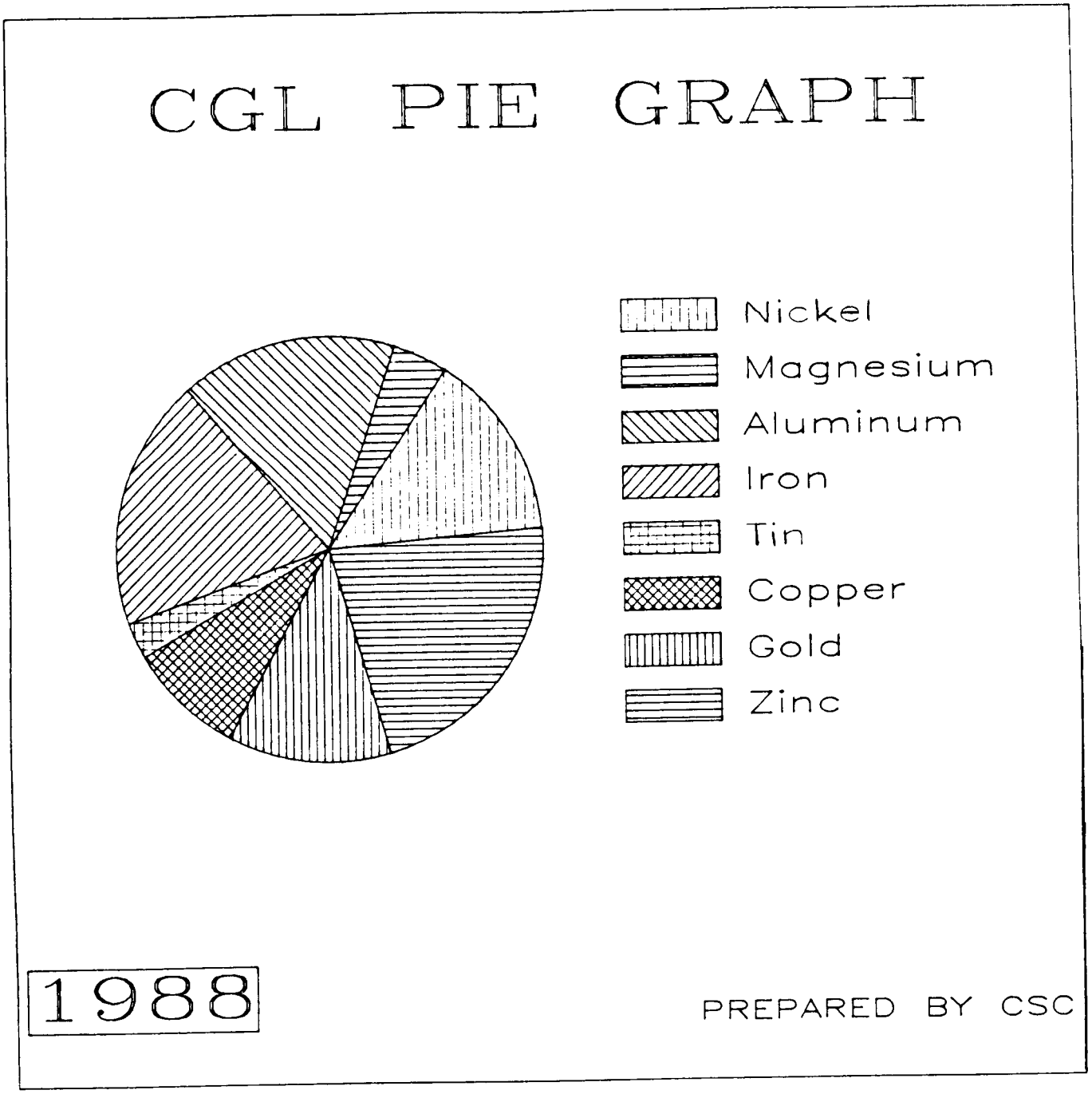


Figure 3-34. Pie chart with a key.

3.5.6 Plotting a Pie Chart with Exploded Segments

In order to plot an exploded pie chart, all steps are the same as in the previous examples, except for the positioning of each segment. For pie charts with non-exploded segments, the origin of the pie segments are fixed. The only change needed to generate pie charts with exploded pie segments, is to vary the origin of the pie segments.

For non-exploded segments

```
      X0= {a fixed value}
      Y0= {a fixed value}
      DO 1 I=1,NUMSEG
            CALL JSECTR(X0,Y0,...)
1     CONTINUE
```

For exploded segments

```
      DO 1 I=1,NUMSEG
           X0= {a changing value}
           Y0= {a changing value}
           CALL JSECTR(X0,Y0,...)
1     CONTINUE
```

A complete pie chart program follows, illustrating a pie chart with exploded segments. The corresponding graphics output can be found in Figure 3-35.

```
      PROGRAM CDP2
C-------------------------------------------------------------
C EXPLODED PIE CHART
C-------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.25,2.,2.35,2.85,2.95/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/21.,6.,24.,28./
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/4.,13.,18.,32./
C
      DATA YEAR /'1988'/
C
```

```
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'G[BLC]OLD'/
      DATA LABEL(5)/'I[BLC]RON'/
      DATA LABEL(6)/'S[BLC]ILVER'/
      DATA LABEL(7)/'C[BLC]OPPER'/
      DATA LABEL(8)/'L[BLC]EAD'/
C-----------------------------------------------------------------
C INITIALIZE DI-3000
      CALL CBEGIN
      CALL JBEGIN
C-----------------------------------------------------------------
C WRITE TO A DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-----------------------------------------------------------------
C FILL POLYGONS BY DEFAULT
      CALL JDPINT(1)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.5,.5)
      CALL JFONT(3)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.2)
      CALL JHSTRG('MATERIALS')
      CALL JMOVE(.1,.8)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.8)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-----------------------------------------------------------------
C CLOSE TEMPORARY SEGMENT
      CALL JCLOSE
C-----------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
```

```
C-------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      START=5.
      CALL JSIZE(.15,.15)
      CALL JFONT(1)
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        START=A1
        AMID=(A1-A0)/2. +A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
        CALL JPIDEX(I,I+30)
C
        X0=OFFSET*C1+5.
        Y0=OFFSET*S1+5.
C
        CALL JSECTR(X0,Y0,0.,RAD1,0,A0,A1)
C
        CALL JMOVE(RAD2*C1+5.,RAD2*S1+5.)
C
        IF (AMID.LT.45. .OR. AMID .GE. 315.) CALL JJUST(1,2)
        IF (AMID .GT. 45. .AND. AMID .LT. 135.) CALL JJUST(2,1)
        IF (AMID .GE. 135. .AND. AMID .LT. 225.) CALL JJUST(3,2)
        IF (AMID .GE. 225. .AND. AMID .LT. 315.) CALL JJUST(2,3)
C
        CALL JMOVE(RAD4*C1+5.,RAD4*S1+5.)
 5000   CALL JHSTRG(LABEL(I))
C-------------------------------------------------------------
C CLOSE TE TEMPORARY SEGMENT.
      CALL JCLOSE
C
C-------------------------------------------------------------
C PAUSE TO VIEW CHART.
      CALL JPAUSE(IDEV)
C-------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
      STOP
      END
```

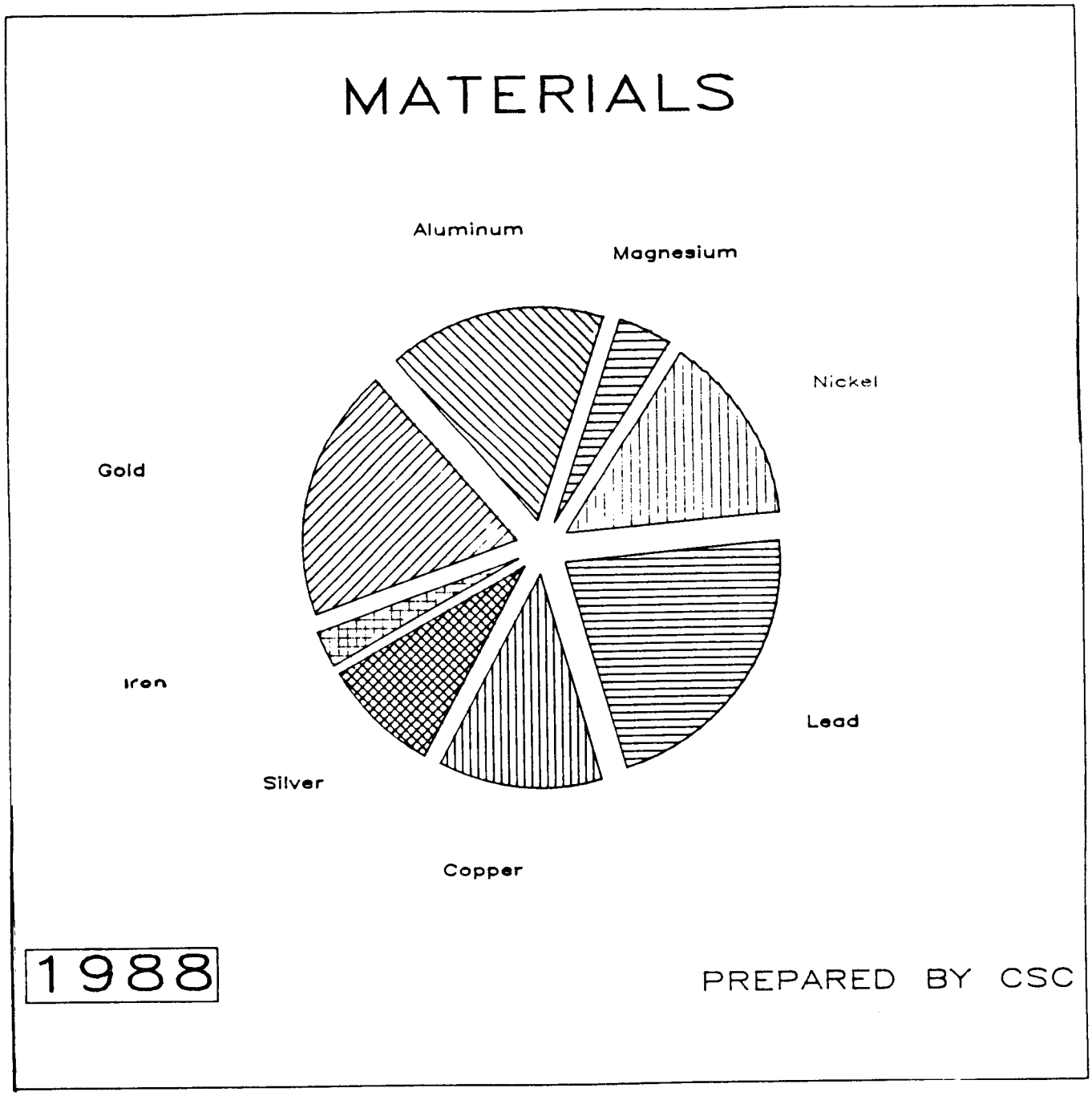


Figure 3-35. Pie chart with exploded segments.

3.6 Steps in the Generation of Composite Charts

The previous subsections described how to generate the various chart types offered by the Common Graphics Library (CGL). The following sections describe how to create a composite chart (i.e., multiple charts on a single display area).

3.6.1 Arranging Composite Charts

Multiple charts can be displayed on a single frame by opening and closing several windows and viewports, and plotting the axes and data corresponding to each chart. Although many charts can be displayed on a page, the user should keep in mind the readability of the resultant chart. When plotting the composite chart, it is recommended one chart be done at a time, and each be done in its own segment. The following is a skeleton program showing the major elements involved in the generation of composite charts.

```
              PROGRAM ...
              . . .
              CALL CBEGIN
              CALL CVSPAC(...)
              . . .
-------------------------------------------------------------
{Repeat this section of code for every chart.}
              CALL JWINDO(...) {page coordinates}
              CALL JOPEN
              CALL CHAXIS(...)
              CALL CVAXIS(...)
                   {save the axes' diagonals virtual values}
              CALL JCONWV(...)
              CALL JCLOSE

              . . .

              CALL JWINDO(...) {data coordinates}
              CALL JVPORT(...) {set viewport based on JCONWV}
              CALL JOPEN
              CALL CLNPLT(..)  {plot data}
              CALL JCLOSE
-------------------------------------------------------------
              . . .
              CALL JDEVOF(...)
              CALL JDEND(...)
              CALL JEND
              STOP
              END
```

In the preceding skeleton program, a window is opened to draw the axes (i.e., page related components), and then a window is opened to draw the data. These steps (indicated above between the dashed lines) should be repeated for each chart.

3.6.2 Complete Program Plotting a Composite Chart

The following program shows how to generate a composite chart. The resultant graphical output is found in Figure 3-36.

```
      PROGRAM CDC1
C-----------------------------------------------------------------------
C COMPOSITE CHART (IE, TWO CHARTS ON ONE PAGE)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=10,MAXSET=4)
      REAL X(MAXPTS,MAXSET),Y(MAXPTS,MAXSET)
      DATA (X(KI,1),KI=1,10)
     + /0.,.1,.2,.3,.4,.5,.6,.7,.8,.9/
      DATA(X(KI,2),KI=1,10)
     + /.1,.15,.3,.35,.45,.55,.65,.75,.85,1.2/
      DATA(X(KI,3),KI=1,10)
     + /.0,.1,.2,.3,.4,.5,.6,.7,.8,.9/
      DATA(X(KI,4),KI=1,10)
     + /.1,.2,.4,.6,.7,.8,.9,1.,1.1,1.2/
      DATA (Y(KI,1),KI=1,10)
     + /0.,2.,4.,7.,10.,12.,15.,18.,21.,24./
      DATA (Y(KI,2),KI=1,10)
     + /1.,3.,5.,8.,11.,13.,16.5,20.,22.,23./
      DATA (Y(KI,3),KI=1,10)
     + /0.,.005,.01,.015,.02,.025,.029,.03,.035,.04/
      DATA (Y(KI,4),KI=1,10)
     + /-.005,-.01,-.015,-.02,-.025,-.03,-.04,-.05,-.065,-.08/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      XS=.2
      YS=6.0
      CALL CVSPAC(10.,10.)
      CALL J4RGET(2,VX3,VX4,VY3,VY4)
      CALL JWINDO(0.,10.,0.,10.)
      CALL JOPEN
```

```
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(XS,XS*1.25)
C POSITION TEXT, AND SET TTEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(5.0,10.0)
      CALL JJUST(2,3)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(XS,XS*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.0
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('C[BSUB]L',1)
      CALL CHTICJ(1)
C XLEN-REPRESENTS THE X-AXIS LENGTH
      XLEN=6.0
      CALL CHPREC(1)
      CALL CHAXIS(-.2,1.2,.2,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('[FONT=9][BLC]A[FONT=3],DEG',1)
      CALL CVPREC(3)
      CALL CVTICJ(1)
C YLEN-REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVPREC(-1)
      CALL CVJUST(1)
      CALL CVAXIS(-4.,24.,4.,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(-.2,1.2,-4.,24.)
C -.2,1.2 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C -4.,24. - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 902, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
```

```
C-----------------------------------------------------------------------
C WORK ON SECOND GRAPH
      CALL JWINDO(0.,10.,0.,10.)
      CALL JVPORT(VX3,VX4,VY3,VY4)
      CALL JOPEN
      CALL JSIZE(XS,XS*1.25)
      CALL JMOVE(XORG,YS)
C YLEN-REPRESENTS THE Y-AXIS LENGTH
      CALL CVLAB('C[BSUB][BLC]M',1)
      CALL CVPREC(2)
      CALL CVAXIS(-.08,.04,.04,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YS,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YS+3.5,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(-.2,1.2,-.08,.04)
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT THIRD DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,3),MAXPTS)
C PLOT FOURTH SET OF DATA POINTS
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,4),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

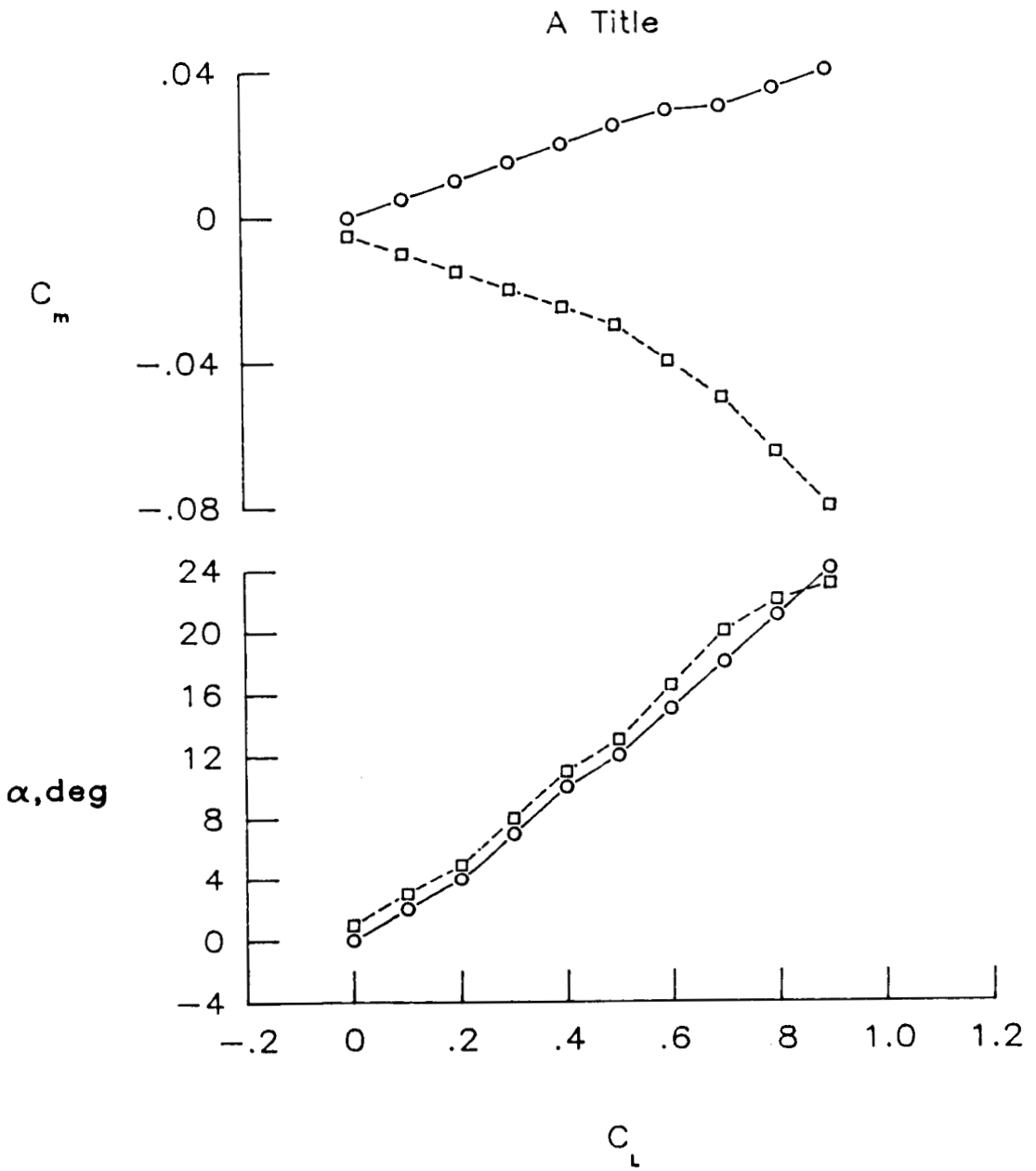


Figure 3-36. Example of a composite chart.

4. ADDITIONAL CAPABILITIES

This section describes capabilities and methodologies useful in generating and debugging programs using the Low-Level graphics programs. Currently, this section contains error detection and debugging capabilities. This section will be expanded to include additional capabilities as they are developed.

4.1 Error Detection and Debugging Capabilities

A fundamental premise of the Low-Level routines states that correct usage of the routines will generate a chart as described by this document. Correct usage consists of a proper sequence of calls with a proper set of routine arguments.

4.1.1 Error Detection

The error detection mechanism implemented in the Low-Level routines is passive; that is as errors are detected, error messages are written to a specified destination. The error mechanism requires no calls to initiate, as error detection is automatically performed. However, there is no means of error inquiry, nor any means of user-definable error recovery.

The errors detected by the Low-Level routines are classified as non-fatal and fatal. Non-fatal errors are considered tolerable in that the Low-Level routines can make suitable adjustments and continue processing. Fatal errors are those which are so severe that error recovery cannot occur, and the program terminates. All errors and their types are included in Appendix A under the routine which can generate them).

An example of a non-fatal error is to pass a value out of range for a subroutine argument. An appropriate error message will be written indicating that the information cannot be used. Typically, errors due to invalid values will result in the arguments being set to their defaults (see Appendix A for defaults).

Figure 4-1 shows an example of the error detection and reporting features, along with the corresponding graphics output. Note, as shown by the following example, the default destination of the error messages is the output device (often overwriting the graphics output). Refer to Section 4.1.3 for changing the destination of output.

The following program illustrates the Low-Level error detection mechanism.

```
      PROGRAM CDD1
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A SIMPLE CGL ERROR.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHTICJ(-1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.01,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
```

```
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDIATES, AND PLOT WITHIN
C BOUNDARIES OF THE AXES (BY SAVE VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10  - REPRESENTS THE DATA BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

```
CHTICJ:    IMPROPER ENTRY GIVEN FOR POSITIONING
           OF TICKMARKS.  VALUE MUST BE 0, 1,
           2, OR 3.  DEFAULT VALUE OF 1 IS USED.
```

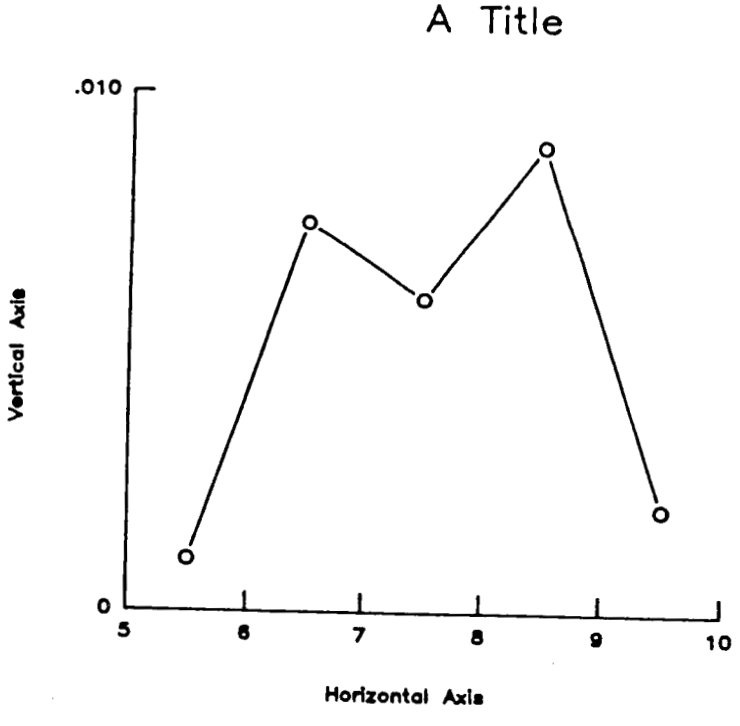


Figure 4-1. Low-Level error detection mechanism.

4.1.2 Debugging Capabilities

The Low-Level routine CDEBUG provides a method for recording the calls to and the arguments of the Low-Level routines. This enables the user to validate the sequence of Low-Level calls, along with their arguments. There are two debug levels: 0 (OFF) or 1 (ON). Initially, the debug level is OFF (i.e., the default), but can be turned ON or OFF at any time. The user can control the amount and content of debugging output by setting the level to include or exclude sections of code to be monitored. To enable Low-Level debugging, the user can enter

```
CALL CDEBUG(1)
```

To disable Low-Level debugging, the user can enter

```
CALL CDEBUG(0)
```

When the debugging level in ON and a Low-Level routine is entered, the following message(s) will be written:

```
routine name:  ENTER
argument_1       : argument type    VALUE= value
argument_2(1)    : argument type    VALUE= value
argument_3(EQ)   : argument type    VALUE= value
```

where the lower-case strings will be replaced by actual values (see the following example). Note, when an argument is an array, then a one in parentheses will follow the argument (see argument_2 above), and the first value of the array will be displayed. If an argument may vary in type depending on the context (e.g., INTEGER in one case, and REAL in another), then an "EQ" in parentheses will follow the argument (see argument_3 above), and the corresponding value of all applicable types will be displayed (see the following example).

Similarly, when the debugging level is ON and a Low-Level routine is exited, the following message(s) will be written:

```
routine name:  EXIT
```

where routine name will be replaced with an actual routine name (see the following example). This information validates the sequence of Low-Level calls, along with the values and types of the corresponding arguments.

The following partial program illustrates the debugging capability along with some sample output. This program is similar to the previous program; a call to CDEBUG should be noted.

```
      PROGRAM CDD2
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A DEBUGGING CAPABILITIES.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C DEBUGGING TURNED ON FOR USER-CALLABLE ROUTINES.
      CALL CDEBUG(1)
C-----------------------------------------------------------------------
```

```
      CALL JWINDO(5.,10.,.0,.01)
C 5,10  - REPRESENTS THE DATA BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

The following output reflects the debug messages written to the default destination TAPE77 (logical unit 77).

```
CDEBUG: ENTER
   IDBLVL     : INTEGER  VALUE = 1
CDEBUG: EXIT
      IDBLVL     : REAL     VALUE = 1
CVSPAC: ENTER
   WIDTH      : REAL     VALUE = 9.
   HEIGHT     : REAL     VALUE = 9.
CVSPAC: EXIT
CHLAB: ENTER
   CHARA(1)   : CHAR*(*) VALUE = H[BLC]ORIZONTAL [BUC]A[BLC]XIS
   NLINE      : INTEGER  VALUE = 1
CSTRIP: ENTER
   STRING     : CHAR*(*) VALUE = H[BLC]ORIZONTAL [BUC]A[BLC]XIS
CSTRIP: EXIT
CHLAB: EXIT
CHAXIS: ENTER
   HZMIN      : REAL     VALUE = 5.
   HZMAX      : REAL     VALUE = 10.
   TINCR      : REAL     VALUE = 1.
   HLONG      : REAL     VALUE = 4.
CGTSTR: ENTER
   VALUE(EQ)  : REAL     VALUE = 0.
   NDEC       : INTEGER  VALUE = -1
CSTRIP: ENTER
   STRING     : CHAR*(*) VALUE =             5
CSTRIP: EXIT
...
CHAXIS: EXIT
CVLAB: ENTER
   CHARA(1)   : CHAR*(*) VALUE = V[BLC]ERTICAL [BUC]A[BLC]XIS
   NLINE      : INTEGER  VALUE = 1
CSTRIP: ENTER
   STRING     : CHAR*(*) VALUE = V[BLC]ERTICAL [BUC]A[BLC]XIS
CSTRIP: EXIT
CVLAB: EXIT
```

```
CLNPLT: ENTER
   X(1)       : REAL     VALUE = 5.5
   Y(1)       : REAL     VALUE = .001
   N          : INTEGER  VALUE = 5
CPNTPT: ENTER
   A          : REAL     VALUE = 5.5
   B          : REAL     VALUE = .001
CPNTPT: EXIT
```

```
CPNTPT: ENTER
   A          : REAL      VALUE = 9.5
   B          : REAL      VALUE = .002
CPNTPT: EXIT
CLNPLT: EXIT
```

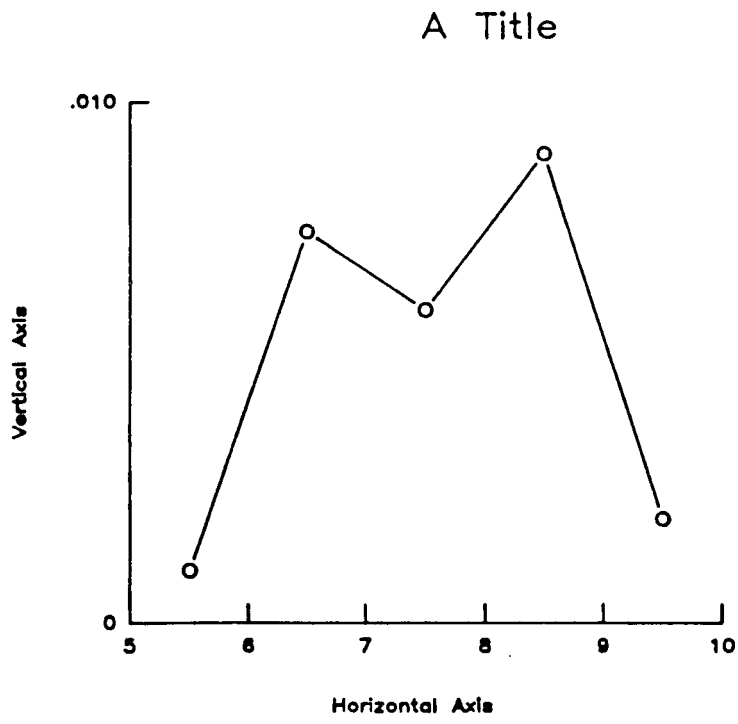


Figure 4-2. Low-Level debug capability.

4.1.3 Destination of Output

The routine CFILES enables the user to control the output destination of error and debug processing. Error messages, both fatal and non-fatal, are written by default to output device (i.e., the current display device). This is often undesirable as it will overwrite graphics output. To redirect Low-Level error messages the user can call

```
ICODE=1
CALL CFILES(ICODE,logical unit,filename)
```

The above call will inform the Low-Level routines to redirect the subsequent Low-Level error messages, indicated by the first argument, to the logical unit with the associated filename. If the file is not already open, CFILES will open the file with the appropriate filename and logical unit number as indicated.

Debug messages, when turned on by CDEBUG, have the default destination of file TAPE77 associated with the logical unit 77. To redirect Low-Level debug messages the user can call

```
ICODE=2
CALL CFILES(ICODE,logical unit,filename)
```

The above call will inform the Low-Level routines to redirect the subsequent Low-Level debug messages, indicated by the first argument, to the logical unit with the associated filename. If the file is not already open, CFILES will open the file with the appropriate filename and logical unit number as indicated.

It is often advantageous to redirect output from several processes into a single output destination. This will provide the user with a single collection of information in the order in which the appropriate processes were executed (i.e., Low-Level error messages, Low-Level debug messages, underlying graphics package messages, and user generated messages).

The following partial program illustrates the changing of the error and debugging destination. This program is similar to the previous program; two calls to CFILES should be noted.

```
      PROGRAM CDD3
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A ERROR AND DEBUGGING REDIRECTION.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C WRITE CGL ERROR MESSAGES TO UNIT 80, FILE 'TAPE80'.
      CALL CFILES(1,80,'TAPE80')
C-----------------------------------------------------------------------
C WRITE CGL DEBUG MESSAGES TO UNIT 80, FILE 'TAPE80'.
      CALL CFILES(2,80,'TAPE80')
C-----------------------------------------------------------------------
C DEBUGGING TURNED ON FOR USER-CALLABLE ROUTINES.
      CALL CDEBUG(1)
C-----------------------------------------------------------------------
```

```
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

Note, this example redirects both error and debugging messages and combines them in the same file for ease of debugging. The user could also write to the same unit, possibly delimiting sections of this file to examine carefully.

```
 CDEBUG: ENTER
    IDBLVL     : INTEGER  VALUE = 1
 CDEBUG: EXIT
       IDBLVL     : REAL     VALUE = 1
 CVSPAC: ENTER
    WIDTH      : REAL     VALUE = 9.
    HEIGHT     : REAL     VALUE = 9.
 CVSPAC: EXIT
 CHLAB: ENTER
    CHARA(1)   : CHAR*(*) VALUE = H[BLC]ORIZONTAL [BUC]A[BLC]XIS
    NLINE      : INTEGER  VALUE = 1
 CSTRIP: ENTER
    STRING     : CHAR*(*) VALUE = H[BLC]ORIZONTAL [BUC]A[BLC]XIS
 CSTRIP: EXIT
 CHLAB: EXIT
 CHTICJ: ENTER
    IV         : INTEGER  VALUE = -1
CHTICJ:   IMPROPER ENTRY GIVEN FOR POSITIONING
          OF TICK MARKS.  VALUE MUST BE 0, 1,
          2, OR 3.  DEFAULT VALUE OF 1 IS USED.
 CHTICJ: EXIT
 CHAXIS: ENTER
    HZMIN      : REAL     VALUE = 5.
    HZMAX      : REAL     VALUE = 10.
    TINCR      : REAL     VALUE = 1.
    HLONG      : REAL     VALUE = 4.
 CGTSTR: ENTER
    VALUE(EQ)  : REAL     VALUE = 0.
    NDEC       : INTEGER  VALUE = -1
 CSTRIP: ENTER
    STRING     : CHAR*(*) VALUE =             5
 CSTRIP: EXIT
                                        .
                                        .
                                        .
 CLNPLT: ENTER
    X(1)       : REAL     VALUE = 5.5
    Y(1)       : REAL     VALUE = .001
    N          : INTEGER  VALUE = 5
 CPNTPT: ENTER
    A          : REAL     VALUE = 5.5
    B          : REAL     VALUE = .001
 CPNTPT: EXIT
                                        .
                                        .
                                        .
```

5. INTERFACING WITH OTHER GRAPHICS PACKAGES

This section describes how to interface the CGL (focusing on the Low-Level routines) with other graphical packages. This section will identify what requirements are necessary to interface with the CGL, and how to interface. The description of interfacing will consist of a generalized method, followed by a specific example (i.e., interfacing with the CGL LEZ routines).

Why interface with the CGL? The Low-Level routines of the CGL provide primitive graphical entities (i.e., NASA symbols, NASA line patterns), and compound graphical entities (i.e., axes, grids, plotting an array, etc.). Because these entities are part of a chart and are basic in nature, there are fewer graphical and computational inherent limitations than with a higher-level graphics package.

These limitations may be graphical or computational. For example, the LEZ routines have a limited number of points which can comprise a data set representing a curve. This limit exists because the LEZ routines must store this information internally. The Low-Level routines display a set of data directly supplied by the user, thus the maximum is limited by the size of the array the user can allocate. Similarly, the Low-Level routines may be used to augment the graphical capabilities of a limited package. For example, the LEZ routines only produce a single set of X and Y axes, whereas the Low-Level routines can be called repeatedly to generate several additional axes.

5.1 Requirements for Interfacing with the CGL

In order to interface with the CGL, it is assumed that a program (i.e., possibly FORTRAN, C, etc.) will invoke the CGL and generate graphics output (i.e., either interactively, or to a metafile). Thus, the appropriate software tools (i.e., device drivers, metafile translators, etc.) are assumed prerequisites when needed.

All computer programs require system resources (i.e., I/O units, field length, device interaction, etc.), and thus must be coordinated. Since the CGL resides on top of a general purpose graphics package, that package must be available and be able to be invoked. If an additional graphics package is to be invoked, then it will either:

- use the same underlying graphics package; or
- require the use of another primitive graphics package.

When using a different underlying graphics package (e.g., using CGL with DI-3000, and another package, possibly GKS), the graphical resources (e.g., graphics devices, metafiles, etc.) are in contention. Typically, these resources are non-sharable simultaneously. However, it may be possible to perform the graphical operations from one package, display the results, terminate that package, then subsequently use the second package to generate graphics on the same display device. This is only possible if both packages can access the same device, and the transition between the two graphics packages can be coordinated. This is typically not the case, as termination and initialization of graphics packages often clear the display device. As an alternative approach, generate the plots separately, then if the metafile translator recognizes the file representation (i.e., metafile, SAVPLT, etc.) combine the chart appropriately.

When using the same underlying graphics package (e.g., using CGL with DI-3000, and the LEZ with DI-3000) the graphics resources are also in contention, but are typically sharable. The following subsections will work with this premise.

5.2 Interfacing with a Generalized High-Level Graphics Package

Sections 3.1 through 3.2 describe the characteristics of the underlying graphics package, and how to interface the CGL with such fundamental packages. This section will describe the general method of interfacing the Low-Level routines with high-level graphics packages.

As indicated in Section 3, most underlying graphics packages operate on similar concepts. These concepts, as illustrated in Figure 5-1, consist of the establishment (i.e., initialization and termination) of the graphics package and the graphics devices, in addition to providing an appropriate graphical context necessary to use the Low-Level routines.

The majority of the Low-Level routines must be called at a point when graphical output can be generated. When interfacing with the underlying graphics package such as DI-3000, this context is referred to as an opened segment. However, high-level graphics packages, such as LEZ and CONTOUR, already incorporate parts of the graphical environment into its structure, such as initialization/termination, or the opening/closing of segments.

To coordinate the interface of the Low-Level routines with a high-level graphics package, the user must ensure that an appropriate graphical environment has been established. Since the Low-Level routines are designed to operate with an

underlying graphics package, it is convenient to allow the high-level package to establish the graphical environment. This may consist of initializing the graphics package and appropriate devices, and generating graphical output (possibly pausing and clearing the display devices). Then, the user can inquire about attributes in the current graphical environment, and invoke the Low-Level routines. Note, the user should ensure that any attribute changed by the CGL will not affect subsequent use of the high-level package.

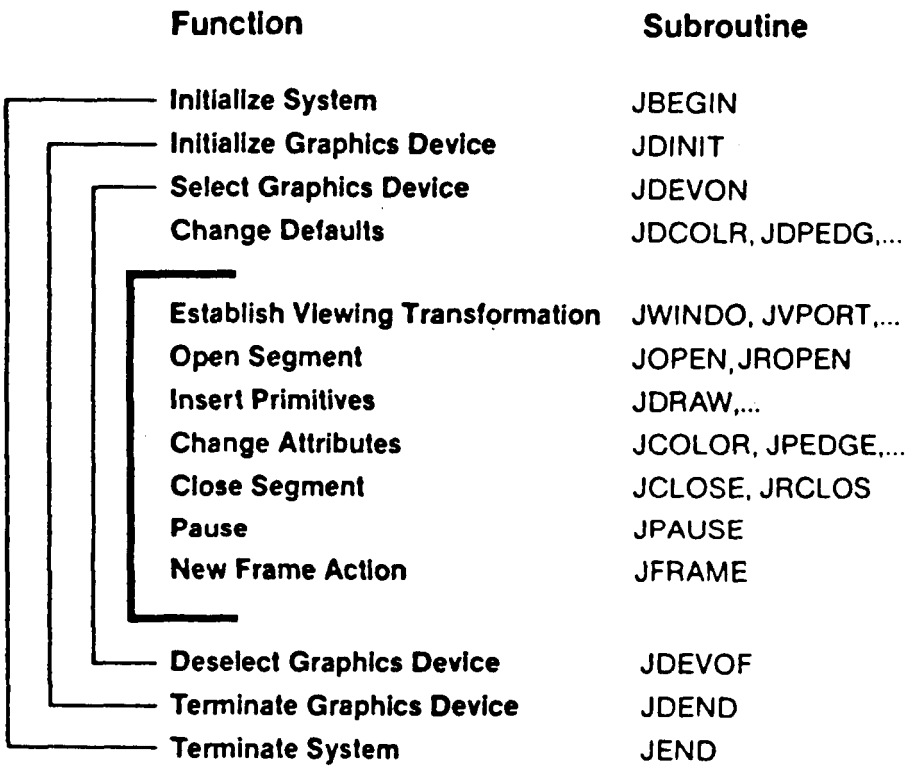


Figure 5-1. DI-3000 skeleton program.

5.3 Interfacing with the CGL LEZ Routines

The CGL LEZ routines provide a high-level abstract user interface incorporating the establishment of the graphical environment. The Low-Level routines must be called after the user has ensured the graphical environment has been established. Thus the Low-Level routines are used to augment the existing graphics package capabilities by being called in an appropriate context.

The LEZ routines are described in the LEZ User's Guide [G-12]; Figure 5-2 shows an LEZ skeleton program. Based on this skeleton, the user should interface the Low-Level routines near LEZSHW (i.e., roughly where the high-level graphics package generates graphics).

The critical environmental attributes to be considered include the window and viewport extents, primitive attributes (i.e., color, character sizes, etc.), and whether the segment is opened.

Function	**Subroutine**
Initialize system Initialize graphics device(s) Select graphics device(s) Set defaults (LEZ/DI-3000)	LEZINI
Establish chart as linear/logarithmic	LEZLIN/LEZLOG
Display current chart	LEZSHW
Deselect graphics device(s) Terminate graphics device(s) Terminate system	LEZTRM

Figure 5-2. LEZ skeleton program.

The values of the window can be set to match the page coordinate area, in which the user can invoke the Low-Level routines to draw a chart component (i.e., axis, etc.). Or, the window can be set to match the data coordinate area, in which the user can display data. In either case, the window will be mapped onto the current viewport, and in turn the viewport will determine where the graphical output will be displayed in the viewspace (i.e., on the display device). For a complete description of mapping see the DI-3000 User's Guide [G-5].

In the following example program, the Low-Level routines are used to draw additional axes (i.e., a top and right axis - see Figure 5-3). Since the axes are components described in terms of page coordinates, the corresponding window and viewport are initially set to the LEZ default values as documented in the LEZ User's Guide [G-12].

Next, the length and origin of the axes are obtained by calling the LEZ inquiry routine LEZGET. These values are in terms of the page coordinate system common to both LEZ and the Low-Level by the previous calls to the window and viewport.

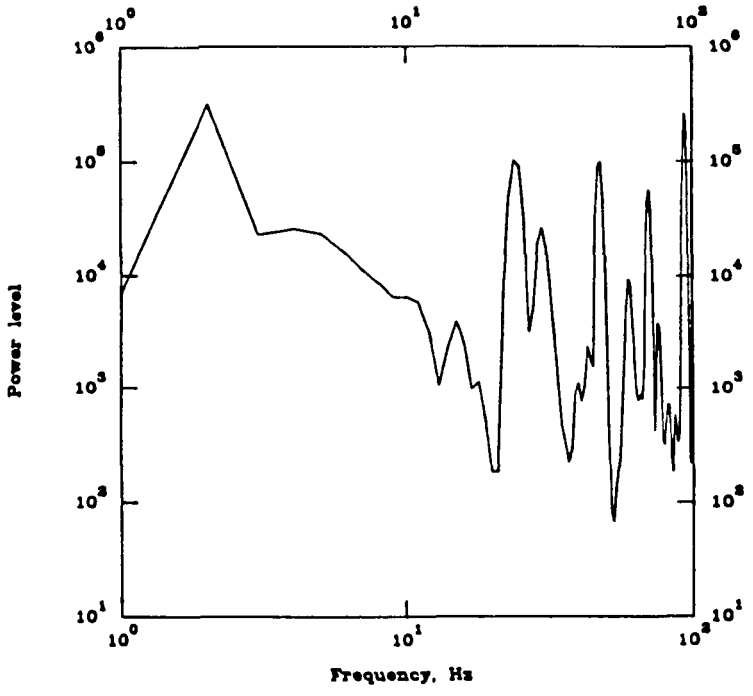
The graphical segment is then opened, and the Low-Level routines are called to generate the axes. Note, the axes are displayed prior to calling LEZSHW, and thus before the chart generated by LEZSHW. Since LEZSHW is called next, any environmental attributes altered by the Low-Level routines (either DI-3000 or CGL) must be restored. Next, the segment is closed, completing the augmentation using the Low-Level routines. Finally, LEZSHW is called to complete the chart generation. The LEZTRM routine is called to deselect and terminate the graphics device(s), and terminate the system.

The following program illustrates the interfacing of the Low-Level routines with a high-level package (i.e., the LEZ routines).

```
      PROGRAM CDLL1
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
```

```
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
      IAXTYP=3
      IHLOGI=0
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDLL1',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
```

```
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
      CALL JCLOSE
C-----------------------------------------------------------------------
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```



Power level vs. frequency

CDLL1

Figure 5-3. Interfacing the Low-Level routines with a high-level package.

In the previous example, the Low-Level routines are called prior to LEZSHW. The Low-Level routines could have been called after LEZSHW with a few minor adjustments. A call to LEZSHW displays the LEZ graphics output, and by default pauses for the viewer. Subsequently, the user hits any key to continue, causing the display device to be cleared (i.e., a frame advance).

Thus, if the Low-Level routines are called after LEZSHW, the use must suppress both the pause and the clear performed by LEZSHW, and invoke these operations appropriately after performing the desired Low-Level operations.

The following program is identical to the previous, except now the calls to the Low-Level routines are made after LEZSHW. Note, the calls to LEZOPT prior to LEZSHW suppress the pause and clear operations. These operations are now performed after the calls to the Low-Level routines by calls to the underlying graphics package (i.e., JPAUSE and JFRAME).

```
      PROGRAM CDLL2
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
     + 23314.47811626, 16222.72126064, 11123.46224798, 8573.673551723,
     + 6533.280733426, 6567.484820724, 5931.92717933,  3100.185764845,
     + 1075.542233512, 2304.552081512, 3938.878376374, 2372.668800947,
     + 990.5827153465, 1139.692923046, 539.1035104119, 186.2399260743,
     + 185.9121866205, 6922.275564162, 47871.80671708, 101610.8109471,
     + 90151.05656181, 32565.81919631, 3126.25985372,  5378.883275207,
     + 19139.27829711, 26311.51279586, 18051.10302768, 7595.925085295,
     + 2769.064915013, 1114.906853713, 472.3435990609, 336.5553643497,
     + 224.6112529072, 292.8078714839, 913.1650006148, 1111.290988571,
     + 763.2538952391, 1036.757022708, 2318.840862193, 1815.115702427,
     + 1546.468077257, 30998.79018191, 90446.39960755, 99542.65318885,
     + 46517.56984878, 8264.407149583, 477.4873134733, 82.183079214,
     + 67.7501808438,  136.6466583528, 197.6147132091, 229.3505457034,
     + 643.2165902202, 1824.921600301, 5246.018888617, 9355.942325835,
     + 8240.202957789, 4042.516486596, 1615.454279283, 876.7523693048,
     + 780.321551955,  869.2380712535, 806.9102907816, 1417.485863451,
     + 13314.08558278, 42705.82833591, 56933.22987158, 34522.64829539,
     + 8369.70034301,  423.4410260173, 1721.308093883, 3815.021552281/
```

```
      DATA (POWER(I),I=77,100)/
     + 3096.896818539, 1130.33337685,  351.6629052997, 323.9146852672,
     + 496.8216394068, 716.3789266813, 707.1830537755, 395.452882063,
     + 185.9595078555, 349.7278623283, 579.1256929328, 427.7957796537,
     + 339.4866401359, 404.7726592585, 3303.494231505, 44732.3344563,
     + 168132.2764905, 259620.2781873, 179647.069922,  51634.19154093,
     + 4701.338454798, 220.74733211,   377.2257021375, 552.6383507909/
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
      IAXTYP=3
      IHLOGI=0
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDLL2',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C SUPPRESS THE LEZ DEFAULTS OF PAUSE AND CLEAR
      CALL LEZOPT('PAUSE',.FALSE.)
      CALL LEZOPT('CLEAR',.FALSE.)
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
```

```
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW PAUSE AND CLEAR
      IF(IDEV.GE.1)CALL JPAUSE(1)
      CALL JFRAME
C-----------------------------------------------------------------------
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```

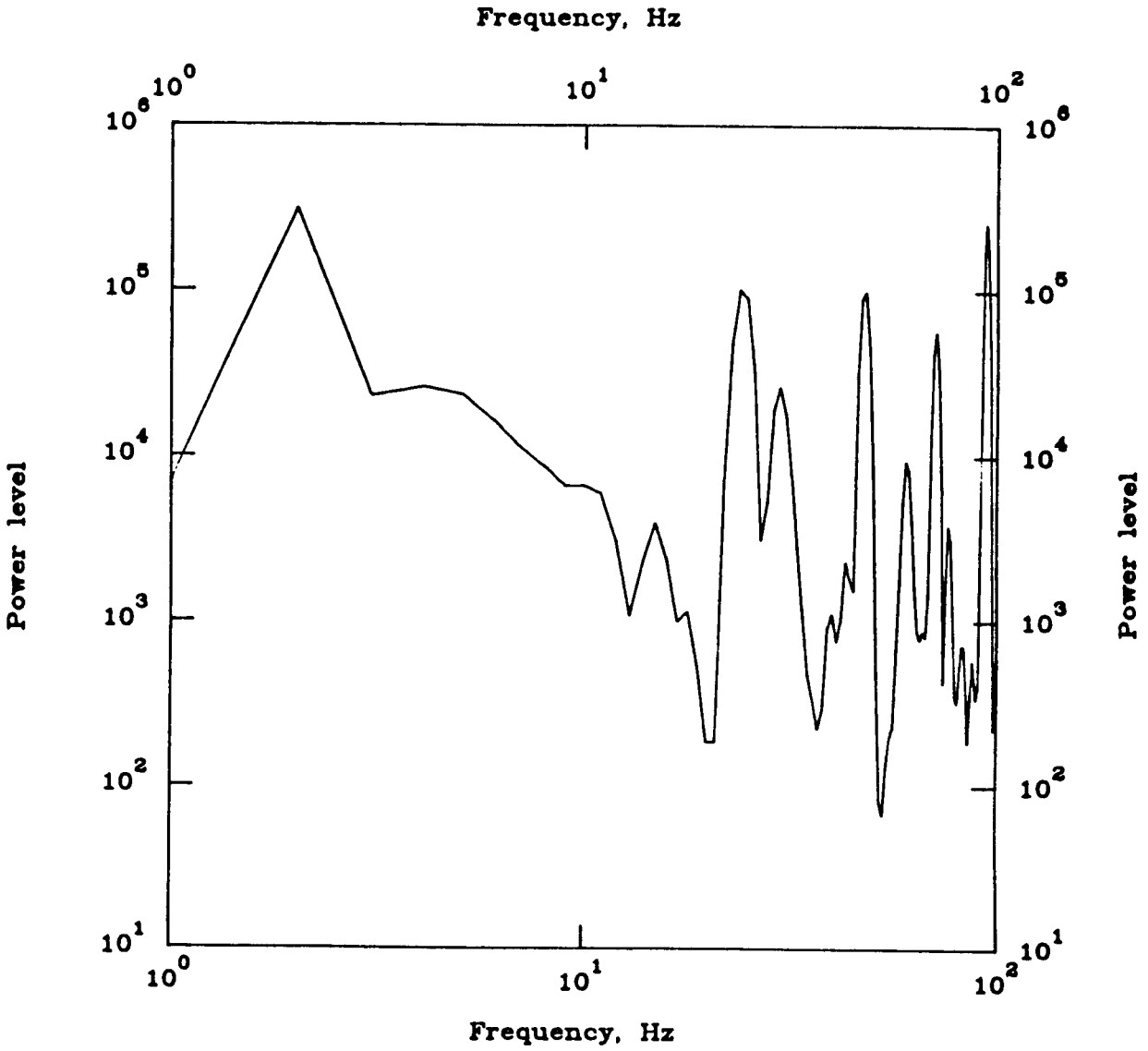


Figure 5-4. Interfacing the Low-Level routines with a high-level package.

6. POSTPROCESSING CONSIDERATIONS

This section gives basic information needed to postprocess plots generated by the Low-Level routines. This section also describes what postprocessing capabilities are available, and what information is necessary in order to interface with the postprocessors.

Most large-scale, general-purpose graphics packages provide the ability to generate and manipulate graphical information in an external file. Thus the user can generate interactive plots, and save the equivalent graphical representation on a file for later use. However, the ability to postprocess depends on a translator to convert the external file information into appropriate instructions to drive the physical device. Thus the translator can take the same external file and plot (or display) it on several different display devices.

For example, the Low-Level routines use DI-3000 as its underlying graphics package, and thus generates external files, called metafiles, which can be postprocessed using the Metafile Translator. Thus, DI-3000 provides a device independent method of generating plots which can be displayed on several different devices (interactive or hardcopy) without the need of major code modifications.

6.1 Postprocessors

A postprocessor is a separate program that reads a metafile, performs various operations on the file, and formats the data to a file that can be used to drive the specified graphics plotting device.

The Metafile Translator is a plot postprocessor which is useful for previewing and editing of metafiles. The Metafile Translator is documented in the Metafile User's Guide [6]. The Metafile Translator instructions allow more flexibility in how the metafile may be displayed.

A special form of the Metafile Translator is the PLOT control statement which is available on the ACD central computers (CDC/NOS). The PLOT statement is documented in the ACD static plotter device driver manual [7].

6.2 DI-3000 Concepts as Related to Static Plotters

Certain postprocessing limitations exist and may need to be addressed. First, the availability of the devices is installation dependent, and the supported device drivers may further restrict this accessibility. Second, each device has inherent limitations (such as color, selective erase, and so forth), and these device dependent characteristics can have a major impact on the resultant output. Most notably, special considerations are needed to ensure that proper sizing or scaling is consistent between the constructed image and the plotted image. Specific information for devices are given in the device driver manuals [7].

The plotters vary in the quality and types of plots that are generated. The pen plotter provides smooth lines but limits the available colors. The pen plotter will also allow for varying the thickness of a line. The electrostatic plotters offer limited line thickness and also may not generate smooth curves because of the pixel resolution. This non-smooth problem is usually eliminated by generating larger plots and then having them reduced.

6.3 Distortion and Scaling

The user can request that a metafile be generated by calling the appropriate underlying graphics package routines (as mentioned in Section 1.2.2, and as shown in several examples in Section 3). In order to postprocess this metafile, the user should know the dimensions used to draw the image. The user is responsible for requesting the appropriate dimensions of the plotter, typically by matching the metafile's aspect ratio. All of the plotters regard a metafile's aspect ratio as its height to width (determined by the viewspace). Proper information will ensure:

- a compatible aspect ratio between the page size and the device coordinate system. For example, a plot generated in a 7 by 7-inch page size should be plotted in an area of equal (square) aspect ratio to avoid distortion.

- the scaling to be performed from the world coordinate to device is consistent (e.g., a 7 by 7-inch plot is sent to a plotting device and will be plotted on a 7 by 7-inch region).

The page size is set by a call to CVSPAC, which ensures that the Low-Level routines of the CGL will establish the largest viewspace corresponding to the desired page size, and thereby a consistent relationship between the page size and the underlying virtual coordinates. However, the user must ensure that the postprocessor aspect ratio coincides with the viewspace. The NGS software defaults to plot in the largest possible square on a given device. This ratio may be changed to meet specific needs. Since each plotter uses a different frame size, the physical dimensions of the square varies between devices. The COMMON GRAPHICS LIBRARY offers routines which support commonly used aspect ratios, such as page size, and also allows the user the capability of setting an aspect ratio to meet specific requirements.

Since there are no default world coordinates in the Low-Level routines, the user must establish them, preferably through CVSPAC. For example, a call to CVSPAC with the values (7.,7.), would establish a world coordinate system of a 7 by 7-unit region (e.g., possibly inches). This implies the aspect ratio is 1. To avoid distortion, the user should select a plotter aspect ratio of 1. For example:

```
PLOT.CAL,11(HEIGHT=7.,WIDTH=7.)
 or
PLOT.CAL,11(HEIGHT=11.,WIDTH=11.)
```

The first plot instruction will plot the image on the CalComp 11" plotter in a 7 by 7-inch region (matching identically with the image area). The second plot instruction will plot the image in an 11 by 11-inch region (proportionally enlarging all objects).

6.4 Visual Characteristics

Because of the variations in plotter characteristics, the user should try to use only enough colors to specifically address the plots needs. If monochrome copies are to be made, then line patterns and fill patterns should be clear enough so that the contents of the chart are distinguished easily by the reader. Care should also be taken to ensure that text is properly positioned on the page and is of sufficient size that it is legible. The text should be **bold** enough so that it may be copied by a photocopy device without losing any portion of a character. Viewgraphs should contain characters large and bold enough to be seen across a dark room. All of these items require planning in the design of a plot. Fortunately for the user, there are usually consistent requirements. Once the appropriate technique is selected, most plots will conform to standards, thus resulting in little or no change to the user's plot program.

APPENDIX A

Description of the Low-Level Routines

The Low-Level routines are described in detail on the following pages. The following information is provided for each routine:

Subroutine Name: The subroutines are listed in alphabetical order.

Purpose: A brief description of what the routine is used for.

Use: The structure of how to call the routine. Also a list of the parameters for that routine with a description of what they are and the types of values they accept.

Restrictions: A list of restrictions on the values passed to the parameters or on the use of that routine.

Notes: Any further information that the user might have need to know.

Errors: A list of errors which may be produced by the routine. The error message will indicate the routine which reported the error, and a brief description of the cause of the error. Typically, the message will be followed by the method of recovery the routine used (i.e., routine not executed, the default value used, etc.).

Description of the Low-Level Routines

CARROW - Draw a Line between 2 Pts (with an Arrow Head at an End)

LANGUAGE: FORTRAN 77

PURPOSE: To draw a line between two points starting at the point (AX,AY) and ending at the point (BX,BY). An arrow head of ITYPE is drawn at point (AX,AY). The final pen position is set to point (BX,BY).

USE: CALL CARROW(AX,AY,BX,BY,ALNGTH,ITYPE)

AX,AY - Starting coordinates of the line.
type:REAL
BX,BY - Ending coordinates of the line.
type:REAL
ALNGTH - Length of the arrow head.
range:greater than 0.
type:REAL
ITYPE - Type of arrow head.
range: 0 to 5.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, polygon attributes.
2) CGL attributes: None.

RESTRICTIONS:
1) AX,AY cannot equal BX,BY.
2) Segment must be open.

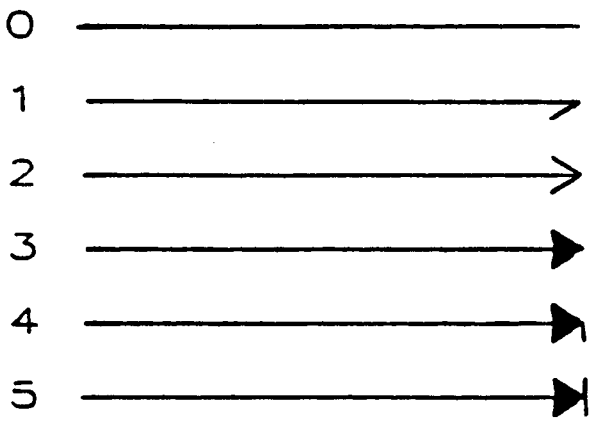
ERRORS:

CARROW: THE RANGE OF ALNGTH IS .GT. 0
ATTEMPTED TO PASS IN ALNGTH = alngth
THIS ROUTINE IS NOT EXECUTED.

CARROW: THE RANGE OF ITYPE IS 0 TO 5 (INTEGER).
ATTEMPTED TO PASS IN ITYPE = itype
THIS ROUTINE IS NOT EXECUTED.

CARROW: BOTH (BX=AX) AND (BY=AY) CAN"T BE TRUE.
THE ROUTINE IS NOT EXECUTED.

NOTE: Arrowheads set to ITYPE of 3,4,5 are drawn as polygons, and therefore can be solid filled.


0
1
2
3
4
5

CBEGIN - Initialize Values used by the CGL

LANGUAGE: FORTRAN 77

PURPOSE: Initialize values used by the CGL.

USE: CALL CBEGIN

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: None.
2) CGL attributes: None.

RESTRICTIONS: 1) Must be called prior to using other CGL routines. If this routine is not called, uninitialized variables may cause unpredictable results. The invocation of this routine is functionally independent of DI-3000, and can thus be called anywhere, anytime. However, subsequent invocations will reset any user defined CGL attributes.

ERRORS: none.

EXAMPLE:

```
      PROGRAM MAIN
      .
      .
C Initialize DI-3000
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C Initialize COMMON GRAPHICS LIBRARY (CGL)
      CALL CBEGIN
```

CDEBUG - Change The Low-Level Debug Level.

LANGUAGE: FORTRAN 77

PURPOSE: Change the Low-Level debug level.

USE: CALL CDEBUG(IDBLVL)

WHERE: IDBLVL - The debug level.
0 - Debugging turned OFF. (default)
1 - Debugging turned ON for user-callable routines.
2 - Debugging turned ON for user-callable and internal routines.
type: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

```
CDEBUG: INVALID ARGUMENT IN CDEBUG.
        IDBLVL SHOULD BE AN INTEGER 0,1 OR 2.
        IDBLVL NOT CHANGED, CURRENT VALUE = idblvl
```

NOTES: See CFILES for the file name and associated logical unit where the debugging information is written.

CDREXP - Convert FORTRAN to Standard Scientific Notation

LANGUAGE: FORTRAN 77

PURPOSE: This routine converts the FORTRAN exponent notation to standard scientific notation. If a REAL zero value is input as '.0' it will be returned as '0.0'. Thus, the routine only reformats a value, as opposed to recomputing it.

This routine also puts in the DI-3000 graphics art precision text notation for the superscripts of an exponent (if one is needed). If no exponent is needed, the number is returned as specified by NDIGIT on the call statement.

USE: CALL CDREXP(CNMIN,NUMIN,NDIGIT,CNMOUT,NUMOUT)

where: CNMIN - Character array containing FORTRAN notation exponent.
type:CHARACTER

NUMIN - Number of characters in CNMIN
type:INTEGER

NDIGIT - Number of digits to precede (to the left of) the decimal point.
type:INTEGER

CNMOUT - Character array containing scientific notation exponent.
type:CHARACTER (returned)

NUMOUT - Number of characters in CNMOUT.
type:INTEGER (returned)

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes:

NRLINT - REAL to INTEGER conversion flag
.TRUE. - REAL value is plotted as an INTEGER
.FALSE. - REAL value is plotted as a REAL (default)
Type:logical

RESTRICTIONS: 1) The maximum number of characters for CNMIN is 18. The maximum number of characters for CNMOUT is 34. CNMOUT will contain DI-3000 text mnemonics to represent superscripts.

ERRORS:

CDREXP: NUMIN IS NOT OF INTEGER TYPE.
ATTEMPTED TO PASS IN NUMIN = numin
THIS ROUTINE IS NOT EXECUTED.

CDREXP: NDIGIT IS NOT OF INTEGER TYPE.
ATTEMPTED TO PASS IN NDIGIT = ndigit

THIS ROUTINE IS NOT EXECUTED.

EXAMPLE:

```
CALL CDREXP('??',2,2,CNMOUT,NUMOUT)
(* CNMOUT WILL = '???????', AND NUMOUT WILL = ????. *)
```

CELIPS - Plot an Arc of an Ellipse Centered at Current Point

LANGUAGE: FORTRAN 77

PURPOSE: To plot an arc of an ellipse centered at current point. Since the arc may not be a closed line segment, the DI-3000 polygon is not used (i.e., not solid filled).

USE: CALL CELIPS(A,B,ALPHA,THETA0,THETAF)

A - Length of the semi-major axis (world coordinates).
type:REAL

B - Length of the semi-minor axis (world coordinates).
type:REAL

ALPHA - Angle of orientation of the major axis. This angle is measured from the horizontal to the major axis. Counter-clockwise is defined as positive direction.
type:REAL

THETA0- Starting angle between radius vector and major axis.
type:REAL

THETAE- Final angle between radius vector and major axis.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes.
2) CGL attributes: None.

RESTRICTIONS: 1) If THETA0=THETAF, then no ellipse is generated.
2) Segment must be open.

NOTES: Since the arc may not be a closed line segment, the DI-3000 polygon is not used (i.e., not solid filled).

ERRORS: none.

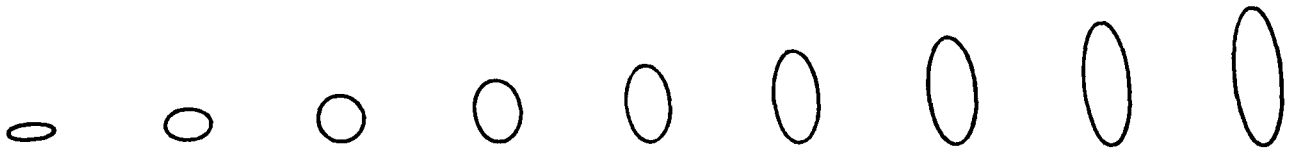


Figure illustrating ellipses.

CERROR - To Set the Error Level for the Low-Level Routines

LANGUAGE: FORTRAN 77

PURPOSE: To set the error level for the Low-Level routines.

USE: CALL CERROR(IERLVL)

WHERE: IERLVL - The error level.
0 - Erroring checking disabled.
1 - Erroring checking enabled for Low-Level ("C"-level) routines. (default)
2 - Erroring checking enabled for level 1 and all internal routines.
type: INTEGER

ERRORS:

CERROR:INVALID ARGUMENT IN CERROR.
IERLVL SHOULD BE AN INTEGER 0, 1, OR 2.
IERLVL NOT CHANGED, CURRENT VALUE = ierlvl

NOTES: Error level 2 (internal routines) is not intended for users, but for internal debugging of the CGL.

CEXP - Determine the Exponent for Log Base 10 Min/Max Values

LANGUAGE: FORTRAN 77

PURPOSE: Determine the exponent for log base 10 of a minimum and maximum value. The resultant exponent for the minimum value will be such that: VMIN >= 10.**MINEXP. The resultant exponent for the maximum value will such that: VMAX <= 10.**MAXEXP.

EXAMPLE:

VALUE	MINEXP	MAXEXP
.01	-2	-2
.05	-2	-1
.1	-1	-1
.5	-1	0
1.0	0	0
1.5	0	1
9.5	0	1
10.5	1	1
15.0	1	2
100.0	2	2
105.0	2	3

USE: CALL CEXP(VMIN,VMAX,MINEXP,MAXEXP)

where:

VMIN - Minimum value
type:REAL

VMAX - Maximum value
type:REAL

MINEXP - Log 10 exponent for VMIN (returned)
type:INTEGER

MAXEXP - Log 10 exponent for VMAX (returned)
type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) VMIN and VMAX must be greater than zero.

ERRORS:

CEXP: INVALID VALUE FOR VMIN OR VMAX.
VMIN,VMAX = vmin,vmax
VALID RANGE IS REAL GREATER THAN ZERO.
MINEXP,MAXEXP RETURNED AS ZERO.

CFILES - Change The Output Destination Of Various Processes.

LANGUAGE: FORTRAN 77

PURPOSE: Change the output destination of various processes. This routine opens a file, denoted by the logical unit LUN and file name of CFILE.

USE: CALL CFILES(ICODE,LUN,CFILE)

WHERE: ICODE - Specifies the process to be changed.
1 - Error processing.
2 - Debug processing.
3 - Internal processing (LEZLIN/LEZLOG).
type: INTEGER

LUN - Output logical unit.
Must be an integer greater than 0.
Logical units are MACHINE DEPENDENT.
type: INTEGER

CFILE - Output file descriptor.
CFILE declared internally as size of 20.
Valid file names are MACHINE DEPENDENT.
type: CHARACTER

NOTES: The files are opened with:
"OPEN(UNIT=LUN,FILE=CFILE,STATUS=UNKNOWN").
All other file attributes use MACHINE defaults.

Both LUN and CFILE must be valid before files are processed.

ERRORS:

CFILES: INVALID ARGUMENT IN CFILES.
ICODE MUST BE AN INTEGER 1,2, OR 3.
NO CHANGE HAS BEEN MADE.

CFILES: INVALID ARGUMENT IN CFILES.
LUN SHOULD BE AN INTEGER GREATER THAN 0.
NO CHANGE HAS BEEN MADE.

CFILES: INVALID ARGUMENT IN CFILES.
CFILE IS INVALID.
NO CHANGE HAS BEEN MADE.

CFILES: LUN=lun WAS ASSIGNED TO "fn".
REASSIGNED TO "jfile".

CFILES: ERROR DURING OPEN.
ICODE,LUN,CFILE= icode,lun,"cfile".
FILE NOT PROCESSED.

```
CFILES: ERROR DURING INQUIRE.
        ICODE,LUN,CFILE= icode,lun,"cfile".
        FILE NOT PROCESSED.
```

CFLIP - Offset an Array about a Reflection Point (2D)

LANGUAGE: FORTRAN 77

PURPOSE: Subroutine to offset an array about a reflection point. Specifically, offset each element of array A, by an equal distance (of that element), on the opposite side of a point of reflection (REFLEC).

USE: CALL CFLIP(A,N,REFLEC)

Where: A - The array to be examined and transformed.
Type:REAL
N - Number of elements in array A.
Type:INTEGER
REFLEC- Point about which to invert data.
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) N must be 2 or greater.

Discussion:

This routine was written to map elements of an array into the corresponding mirror image. One purpose of this would be to plot data in decreasing order.

Example: Given a minimum and maximum of 100 to 200, the elements of an array must be adjusted in order to plot them on an axis decreasing from 200 to 100.

Plot data in a world represented by 200...100.

The input required would consist of the array to be adjusted, the size of the array, and in this example, the point of reflection (150). More specifically, 100+((200-100)/2.), or more generally, MINIMUM+((MAXIMUM-MINIMUM)/2.).

This presupposes that the world coordinate system will be inverted by mapping MAXIMUM...MINIMUM, to MINIMUM...MAXIMUM (as shown above).

Thus, given an array A to be plotted from 200 to 100,

```
CALL JWINDO(100,200,YMIN,YMAX)
CALL CFLIP(A,N,REFLEC)
<plot data>
```

Note: to return data to original values, apply
CFLIP to the adjusted data ("unflip").
(i.e., flip(flip(A)) == A).

A detailed example can be found in Appendix B of the CGL Low-Level User's Guide.

ERRORS:

CFLIP: N IS NOT OF INTEGER TYPE.
ATTEMPTED TO PASS N = n
THIS ROUTINE IS NOT EXECUTED.

CGRID - Draw a Linear Grid (with or without Blank Areas)

LANGUAGE: FORTRAN 77

PURPOSE: Draw a linear grid over a specified plotting area with a rectangular or square area(s) remaining blank.

USE: CALL CGRID (X,Y,XS,YS,NOINCX,NOINCY,BLANK,NBLANK)
where

X,Y - The lower left corner of grid in world coordinate units.
type:REAL

XS - The increment in world coordinate units between vertical grid lines.
range: greater than 0, but less than 4000.
type:REAL

YS - The increment in world coordinate units between horizontal grid lines.
range: greater than 0, but less than 4000.
type:REAL

NOINCX - The number of increments along the X-axis.
range: greater than 0.
type:INTEGER

NOINCY The number of increments along the Y-axis.
range: greater than 0.
type:INTEGER

BLANK - An array containing the coordinates of the rectangular or square blank area(s). The user is responsible for supplying an array containing the corner points of the blank areas, and the corresponding number of blank areas.
type:REAL

The first blank area:
BLANK(1) contains X lower left coordinate
BLANK(2) contains Y lower left coordinate
BLANK(3) contains X upper right coordinate
BLANK(4) contains Y upper right coordinate
For the second blank area:
BLANK(5) contains X lower left coordinate
BLANK(6) contains Y lower left coordinate
BLANK(7) contains X upper right coordinate
BLANK(8) contains Y upper right coordinate

NBLANK - An INTEGER specifying the number of blank areas to be drawn. A maximum of 10 blank areas are allowed.
range: 0 to 10. If NBLANK is 0, then a dummy variable should be passed for BLANK.
type:INTEGER

NOTES:

To draw a box around the areas, the user
is required to make the following calls
immediately before or after the call to "CGRID" :
CALL JRECT(BLANK(1,1),BLANK(2,1),BLANK(3,1),BLANK(4,1))
For the 2nd box -
CALL JRECT(BLANK(1,2),BLANK(2,2),BLANK(3,2),BLANK(4,2))
For the N'th box -
CALL JRECT(BLANK(1,n),BLANK(2,n),BLANK(3,n),BLANK(4,n))

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: line attributes.
2) CGL attributes: none.

RESTRICTIONS:

1) The maximum number of boxes is limited to 10.
2) NOINCX and NOINCY are limited to 4000.
3) Segment must be open.

ERRORS:

CGRID: XS MUST BE .GT. 0, BUT LESS THAN maxinc.
ATTEMPTED TO PASS IN XS = xs
THIS ROUTINE IS NOT EXECUTED.'

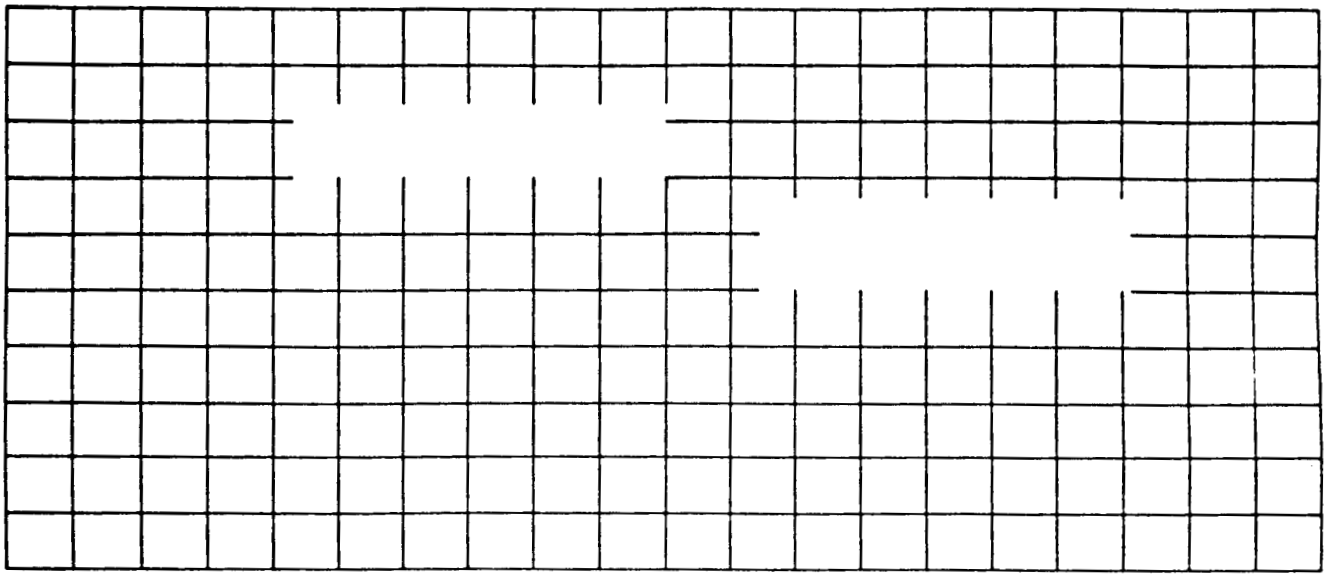
CGRID: YS MUST BE .GT. 0, BUT LESS THAN maxinc.
ATTEMPTED TO PASS IN YS = ys
THIS ROUTINE IS NOT EXECUTED.

CGRID: NOINCX EXPECTED INTEGER TYPE.
ATTEMPTED TO PASS IN NOINCX = noincx
THIS ROUTINE IS NOT EXECUTED.

CGRID: NOINCY EXPECTED INTEGER TYPE.
ATTEMPTED TO PASS IN NOINCY = noincy
THIS ROUTINE IS NOT EXECUTED.

CGRID: NBLANK EXPECTED INTEGER BETWEEN 0 AND maxbox
ATTEMPTED TO PASS IN NBLANK = nblank
THIS ROUTINE IS NOT EXECUTED.

LINEAR GRID



```
CALL CGRID(X,Y,XS,YS,NOINCX,NOINCY,BLANK,NBLNK)
  X,Y      WORLD COORDINATES OF LOWER LEFT PT
  XS       INCREMENT BETWEEN X-AXIS GRID LINES
  YS       INCREMENT BETWEEN Y-AXIS GRID LINES
  NOINCX   NUMBER OF INCREMENTS ALONG X-AXIS
  NOINCY   NUMBER OF INCREMENTS ALONG Y-AXIS
  BLANK    COORDINATE ARRAY OF THE BLANK AREAS
  NBLNK    NUMBER OF BLANK AREAS
```

CGTEXP - Convert Exponential Value to a Character String

LANGUAGE: FORTRAN 77

PURPOSE: This routine converts an exponential value to a character string and returns the number of characters. This is needed when plotting exponential values, as the actual value to be plotted is converted to display code using a FORTRAN 77 internal write.

USE: CALL CGTEXP(VALUE,NDEC,CHRNUM,NUM)

where: VALUE - Numeric value to be converted.
type:REAL
NDEC - Number of significant digits.
type:INTEGER
CHRNUM- Character string containing the numeric value (returned).
type:CHARACTER
NUM - Number of characters in CHRNUM. up to 18. (returned).
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes:
NRLINT - REAL to INTEGER conversion flag
.TRUE. - REAL value is plotted as an INTEGER
.FALSE. - REAL value is plotted as a REAL (default)
Type:logical

RESTRICTIONS: 1) Up to 13 significant digits are allowed including the decimal point.

ERRORS:
CGTEXP: NDEC EXPECTED INTEGER TYPE.
ATTEMPTED TO PASS IN NDEC= ndec
THIS ROUTINE IS NOT EXECUTED.

CGTSTR - Convert a Numeric Value to a Character Value

LANGUAGE: FORTRAN 77

PURPOSE: This routine converts a numeric value to a character value and returns the number of numeric characters or digits.

USE: CALL CGTSTR(VALUE,NDEC,CHRNUM,NUM)

WHERE: VALUE - Numeric value to be converted.
Type may be either INTEGER or REAL based on NDEC.
type:REAL/INTEGER

NDEC - If the sign of NDEC is positive (i.e., the value of NDEC is 0 or more), then the value NDEC specifies the number of digits to the right of the decimal point. If the sign is negative, then the value NDEC specifies the minimum number of digits.
type:INTEGER

CHRNUM- Character string containing the numeric value (returned).
type:CHARACTER (returned)

NUM - Number of characters in CHRNUM.
Maximum of 13. (returned).
type:INTEGER (returned)

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes:
NRLINT - REAL to INTEGER conversion flag
.TRUE. - REAL value is plotted as an INTEGER
.FALSE. - REAL value is plotted as a REAL (default)
Type:logical

RESTRICTIONS: 1) Up to 13 significant digits are allowed, including the decimal point.

ERRORS:
CGTSTR: NDEC EXPECTED INTEGER TYPE.
ATTEMPTED TO PASS IN NDEC= ndec
THIS ROUTINE IS NOT EXECUTED.

NOTES: A value of 0.0 will be represented as "0" regardless of NDEC (i.e., regardless of type, or precision).

CHAXIC - Draw a Horizontal Axis with Character Labels

LANGUAGE: FORTRAN 77

PURPOSE: To draw a horizontal axis with character labels. This routine retrieves attributes (DI-3000 and CGL) when the subroutine is invoked. These axis characteristics remain in force until SUBROUTINE CHAXIC is reinvoked. If a horizontal axis label is desired, SUBROUTINE CHLAB must be invoked prior to this routine.

USE: CALL CHAXIC(XLEN,NTIC,CHARA,NLINES,NLABS)

XLEN - Length of axis (in world coordinates).
range: greater than 0.
type:REAL

NTIC - Number of tick marks (end tick marks inclusive). If NTIC is negative, the major and minor tick marks will be suppressed, but the tick mark labels will be retained.
range: IABS(NTIC) must be greater than 1.
type:INTEGER

CHARA - Tick mark labels, dimensioned: CHARA(NLINES, NLABS).
type:CHARACTER

NLINES - Number of lines in the tick mark labels.
Range: must be greater than or equal to 0.
type:INTEGER

NLABS - Number of labels for the tick marks.
Range: must be greater than or equal to 0.
type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: line attributes, polygon attributes.
2) CGL attributes:

NHBTIC - Position of horizontal axis label in relation to horizontal tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
type:INTEGER

NHJUST - Determines if horizontal axis label is written horizontally or vertically.
0 - vertical (default).
1 - horizontal.
Type:INTEGER

NHLABJ - Position of axis label in relation to horizontal axis. (above or below)
0 - horizontal axis label is below axis.(default)
1 - horizontal axis label is above axis.

type: INTEGER

NHMTIC - Number of minor tick marks between major tick marks for the horizontal axis. (default=0)
Range : INTEGERS greater or equal to 0.
TYPE: INTEGER

NHTICJ - A value to indicate positioning of tick marks for horizontal axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
Default: 1
Type: INTEGER

NHTROT - Angle of rotation for horizontal axis tick mark labels.
Type: INTEGER

RESTRICTIONS:
1) XLEN must be positive.
2) Segment must be open.
3) An absolute minimum of two major tick marks.
4) Axis label can only be horizontal.

ERRORS:

CHAXIC: XLEN EXPECTED VALUE .GT. 0
ATTEMPTED TO PASS XLEN= xlen
THIS ROUTINE IS NOT EXECUTED.

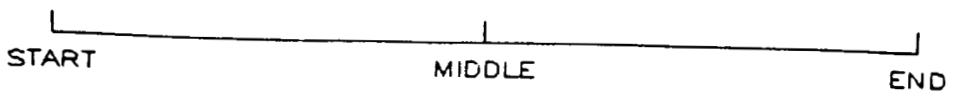
CHAXIC: NTIC EXPECTED INTEGER .GT. 1
ATTEMPTED TO PASS NTIC= ntic
THIS ROUTINE IS NOT EXECUTED.

CHAXIC: NLINES EXPECTED INTEGER .GE. 0
ATTEMPTED TO PASS NLINES= nlines
THIS ROUTINE IS NOT EXECUTED.

CHAXIC: NLABS EXPECTED INTEGER .GE. 0
ATTEMPTED TO PASS NLABS= nlabs
THIS ROUTINE IS NOT EXECUTED.

CHAXIC: TICK MARK ROTATION (NHTROT) ONLY APPLY TO ONE TICK MARK LINE (NLINES).
NO ROTATION PERFORMED.
NH=0

HORIZONTAL CHARACTER AXIS



HORIZONTAL AXIS LABEL

CHAXIS - Draw a Horizontal Axis with Numeric Labels

LANGUAGE: FORTRAN 77

PURPOSE: This routine plots a horizontal numeric axis. The numeric values input are calculated, converted to characters, then a call to the CHAXIC routine is made to do the actual plotting. The attributes used by CHAXIC and CHLAB should be considered prior to calling this routine.

USE: CALL CHAXIS(HZMIN,HZMAX,TINCR,HLONG)

where: HZMIN - Minimum value used for tick marks
type:REAL
HZMAX - Maximum value used for tick marks
type:REAL
TINCR - Increment value of tick marks
type:REAL
HLONG - Length of axis in current world coordinates
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NHBTIC - Position of horizontal axis label in relation to horizontal tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
type:INTEGER

NHJUST - Determines if horizontal axis label is written horizontally or vertically.
0 - vertical (default).
1 - horizontal.
Type:INTEGER

NHLABJ - Position of axis label in relation to horizontal axis. (above or below)
0 - horizontal axis label is below axis.(default)
1 - horizontal axis label is above axis.
type:INTEGER

NHMTIC - Number of minor tick marks between major tick marks for the horizontal axis. (default=0)
Range : INTEGERS greater or equal to 0.
TYPE:INTEGER

NHPREC - Number of digits to the right of the decimal point in the tick mark labels of the horizontal axis.
< 0 - plot as an INTEGER (no decimal point)

= 0 - decimal point only
> 0 - plot as a REAL. Value represents the number of digits to the right of the decimal point

Default: -1
Type: INTEGER

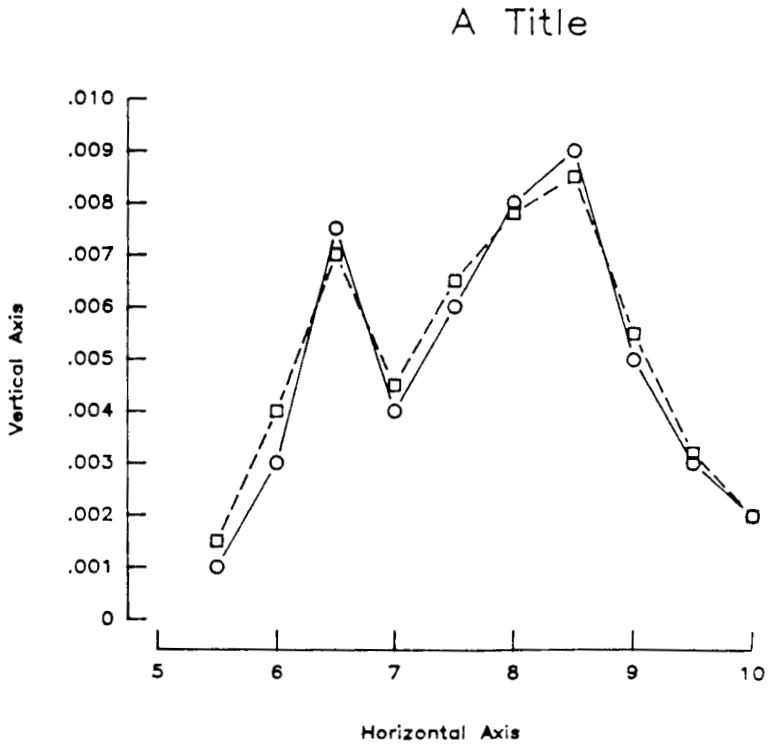
NHTICJ - A value to indicate positioning of tick marks for horizontal axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
Default: 1
Type: INTEGER

NHTROT - Angle of rotation for horizontal axis tick mark labels.
Type: INTEGER

RESTRICTIONS: 1) Tick mark labels always occur at the major tick marks (regardless of attribute settings).
2) If HZMIN is greater than HZMAX, then TINCR must be negative. HZMIN and HZMAX cannot be equal.
3) Segment must be open.

ERRORS: none.

NOTES: If HZMIN is greater than HZMAX, then axis labels will be output in decreasing order.



CHBTIC - Set Position of Horizontal Axis Label Tick Marks

LANGUAGE: FORTRAN 77

PURPOSE: Set position of horizontal axis label tick marks.

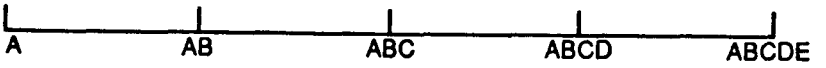
USE: CALL CHBTIC(IV)

where: IV
Position of horizontal axis label in relation to the horizontal tick marks.
0 - labels centered at tick marks.(default)
1 - labels centered between tick marks.
Type:INTEGER

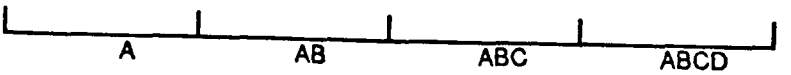
ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CHBTIC: INVALID VALUE FOR IV.
THE RANGE OF nhbtic IS 0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



HORIZONTAL LINE 1



HORIZONTAL LINE 1

CHEXP - Compute Minimum and Maximum Exponents

LANGUAGE: FORTRAN 77

PURPOSE: To determine the minimum and maximum value of the input data for computing the horizontal axis labels.

USE: CALL CHEXP(XMIN,XMAX,XBEG,XEND)

XMIN - Minimum data value.
XMAX - Maximum data value.
XBEG - First tick mark label (returned).
XEND - Last tick mark label (returned).

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CHLAB - Describe Text and Attributes of Horizontal Axis Label

---

LANGUAGE: FORTRAN 77

PURPOSE: To describe the text and associated attributes of the horizontal axis label. This routine retrieves attributes (DI-3000 and CGL) when the subroutine is invoked. These label characteristics remain in force until SUBROUTINE CHLAB is reinvoked.

USE: CALL CHLAB(CHARA,NLINE)

CHARA(NLINE) - Array of characters for axis.
type:CHARACTER
NLINE - Number of lines in label.
range: integer 0 to 5.
type:INTEGER

Associated Attributes:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes: none.

RESTRICTIONS:
1) The number of lines per axis label is limited to 0.
2) The number of characters per line is limited to 256.
3) All lines in axis label are single spaced.
4) Segment must be open.
5) Horizontal axis label can only be horizontal text.

NOTES: if NLINE is less than or equal to 0, then all horizontal axis label attributes are reinitialized. If NLINE is greater than 5, then only the first 5 array elements are used.

ERRORS:
CHLAB: NLINE MUST BE .GE. 0
ATTEMPTED TO PASS NLINE= nline
THIS ROUTINE IS NOT EXECUTED.

CHLABJ - Set Position of Axis Label Relative to Horizontal Axis

---

LANGUAGE: FORTRAN 77

PURPOSE: Set position of horizontal axis label relative to horizontal axis.

USE: CALL CHLABJ(IV)

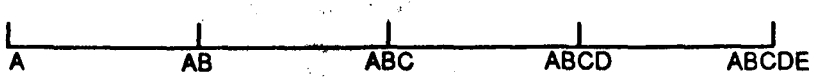
where: IV
Position of axis label in relation to horizontal axis. (above or below)
0 = horizontal axis label is below axis. (default)
1 = horizontal axis label is above axis.
Type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

NOTES: Attribute CHTICJ controls the horizontal axis tick mark positioning (i.e., above, below,etc.). The attribute CHLABJ controls the positioning of the tick mark labels.

RESTRICTIONS: none.

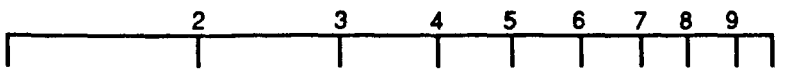
ERRORS:
CHLABJ: INVALID ARGUEMENT.
THE RANGE OF "NHLABJ" IS 0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



HORIZONTAL LINE 1

CALL CHLABJ(0)

HORIZONTAL LINE 1



CALL CHLABJ(1)

CHLOG - Draw a Horizontal Log Axis

LANGUAGE: FORTRAN 77

PURPOSE: To draw a horizontal log axis.
This routine retrieves attributes (DI-3000 and CGL) when this subroutine is invoked.
These characteristics remain in force until subroutine CHLOG is reinvoked.
If a horizontal axis label is desired, subroutine CHLAB must be invoked prior to this routine.

USE: CALL CHLOG(XLEN,NTIC,NHLOGI)

XLEN - Length of axis (in world coordinates).
range: greater than 0.
type: REAL

NTIC - Number of tick marks (end tick marks inclusive) (i.e., number intervals over XLEN axis).
If ntic is negative, the major tick marks will be suppressed, but the numeric tick mark label will be retained.
range: IABS(NTIC) must be greater than 1.
type: INTEGER

NHLOGI- Tick mark labels type.
The following are the various types :
0 - No small tick marks.
1 - 1,2,3,4,5,6,7,8,9,1
2 - 1,2,4,6,8,1
3 - 1,3,5,7,9,1
4 - 1,5,1
5 - 1,1
6 - 2,3,4,5,6,7,8,9
7 - 2,4,6,8
8 - 3,5,7,9
9 - 5

If NHLOGI is positive then labels and small log tick marks are drawn.
If NHLOGI is negative then only small log tick marks are drawn.
type: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NHLABJ - Position of axis label in relation to horizontal axis. (above or below)
0 - horizontal axis label is below axis.(default)
1 - horizontal axis label is above axis.
type:INTEGER

NHLOGF - Determine if horizontal axis will have

major tick marks as 10 raised to a power.
-1 - major tick mark labels decrease
0 - no major tick mark labels (default)
1 - major tick mark labels increase
TYPE:INTEGER

NHLOGS - The initial power for the horizontal log axis labels. (i.e., 10 raised to this power) (default=0)
Range : any valid INTEGER.
Type:INTEGER

NLABP - The size of the minor tick marks as a percentage of the size of the major tick marks. (Default=2/3)
Range : REAL values greater than 0.
Type:REAL

RESTRICTIONS: 1) XLEN must be positive.
2) Only whole log cycles starting at minor tick mark one are generated.
3) An absolute minimum of two major tick marks.
4) Segment must be open.

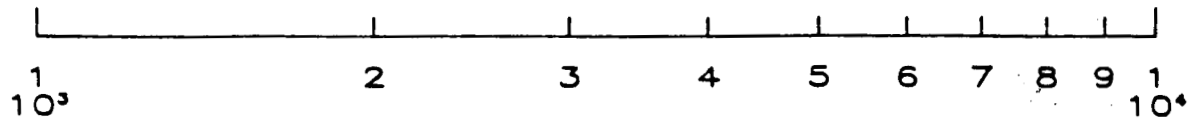
ERRORS:

CHLOG: XLEN EXPECTED TO BE .GT. 0
ATTEMPTED TO PASS XLEN= xlen
THIS ROUTINE IS NOT EXECUTED.

CHLOG: NTIC EXPECTED TO BE AN INTEGER.
ATTEMPTED TO PASS NTIC= ntic
THIS ROUTINE IS NOT EXECUTED.

CHLOG: NHLOGI EXPECTED TO BE AN INTEGER.
ATTEMPTED TO PASS NHLOGI= nhlogi
THIS ROUTINE IS NOT EXECUTED.

HORIZONTAL LOG AXIS



HORIZONTAL LOG AXIS LABEL

1) TO SET AXIS LABEL
   CALL CHLAB(CHARA,NLINE)
   CHARA ARRAY OF CHARACTERS FOR THE AXIS
   NLINE NUMBER OF LINES IN THE AXIS LABEL

2) TO DRAW AXIS AND LABEL
   CALL CHLOG(XLEN,NTIC,NLOGI)
   XLEN   LENGTH OF THE AXIS
   NTIC   NUMBER OF MAJOR TIC MARKS
   NLOGI TIC MARK LABEL TYPE
   0 - NO MINOR TIC MARKS
   1 - 1.2.3.4.5.6.7.8.9.1
   2 - 1.2.4.6.8.1
   3 - 1.3.5.7.9.1
   4 - 1.5.1
   5 - 1.1
   6 - 2.3.4.5.6.7.8.9
   7 - 2.4.6.8
   8 - 3.5.7.9
   9 - 5

CHLOGF - Set Horizontal Axis Power of 10

LANGUAGE: FORTRAN 77

PURPOSE: Set horizontal axis power of 10

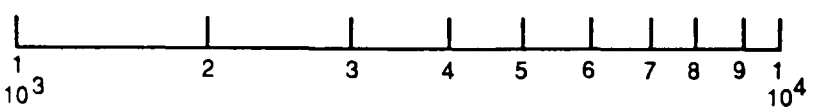
USE: CALL CHLOGF(IV)

where: IV -
Determine if horizontal axis will have major tick marks as 10 raised to a power.
-1 - Major tick mark labels decrease
0 - No major tick mark labels (default)
1 - Major tick mark labels increase
Type: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CHLOGF: INVALID VALUE FOR IV .
THE RANGE OF "NHLOGF" IS -1,0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



CALL CHLOGF(1)

CHLOGS - Set the Initial Power for the Horizontal Log Axis Labels

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the initial power for the horizontal log axis labels.

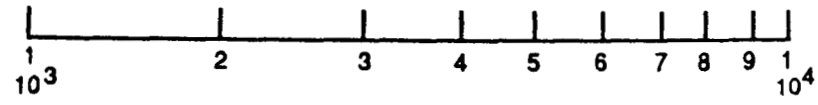
USE: CALL CHLOGS(IV)

where: IV
The initial power for the horizontal log axis labels (i.e., 10 raised to this power) (default = 0)
Range: any valid integer
Type: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: None.

ERRORS:
CHLOGS: INVALID VALUE FOR IV.
THE RANGE OF "NHLOGS" IS ANY INTEGER.
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



CALL CHLOGS(3)

CHMTIC - Set the # of Horizontal Minor Tick Marks between Major

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the number of horizontal minor tick marks between major tick marks.

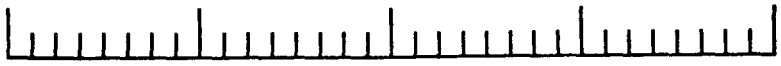
USE: CALL CHMTIC(IV)

where: IV -
Number of minor tick marks between major tick marks for the horizontal axis. (default = 0)
Range: Integers greater or equal to zero.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: None.

ERRORS:
CHMTIC: IV MUST BE AN INTEGER .GE. 0
ATTEMPTED TO PASS IV= iv
THE DEFAULT VALUE IS USED.



CALL CHMTIC(7)

CHNMBR - Draw a Numeric Value using DI-3000 Text Attributes

LANGUAGE: FORTRAN 77

PURPOSE: This routine draws a numeric value using current text attributes.

USE: CALL CHNMBR(VALUE,NDEC)

where: VALUE - Value to be plotted real or integer.
type:REAL
NDEC - Integer specifying number of digits to the right of the decimal point.
type:INTEGER
NDEC = 0 - Decimal point, number digits To the right.
NDEC < 0 - No decimal point, minimum # of digits in integer value.

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: text attributes.
2) CGL attributes: none.

RESTRICTIONS: 1) The number is restricted to a maximum of 12 significant digits.
2) Segment must be open.

ERRORS:
CHNMBR: NDEC MUST BE AN INTEGER .GT. 0
ATTEMPTED TO PASS NDEC= ndec
THIS ROUTINE IS NOT EXECUTED.

CHPREC - Set Horizontal Numeric Axis Tick Mark Label Precision

---

LANGUAGE: FORTRAN 77

PURPOSE: Set horizontal numeric axis tick mark label precision

USE: CALL CHPREC(IV)

Where:

IV - Number of digits to the right of the decimal point in the tick mark labels of the horizontal axis
- > 0 - Plot as a real. IV represents the number of digits to the right of the decimal point
- < 0 - Display as integer (no decimal point)
- = 0 - No trailing digits.

Default: -1

type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: None.

ERRORS:

CHPREC: IV MUST BE AN INTEGER.
ATTEMPTED TO PASS IV= iv
THE DEFAULT VALUE IS USED.

CHTICJ - Set the Tick Mark Position Relative to the Horizontal Axis

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the tick mark positioning relative to the horizontal axis.

USE: CALL CHTICJ(IV)

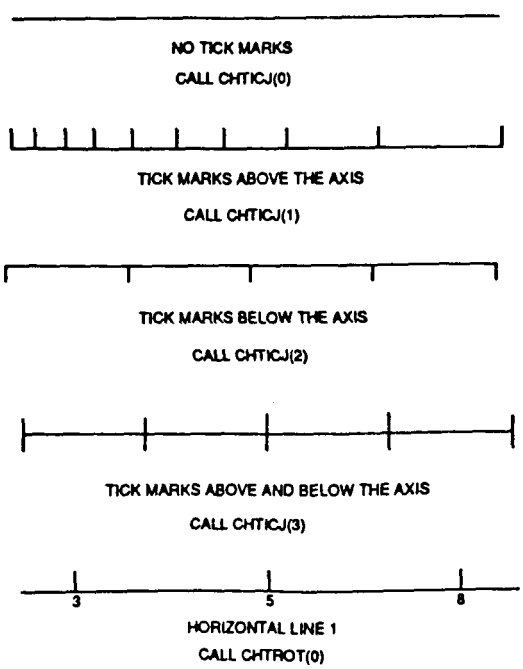
WHERE: IV -
is a value to indicate positioning of tick marks
0 -- No tick marks.
1 -- Tick marks positioned above the axis.
2 -- Tick marks positioned below the axis.
3 -- Tick marks positioned above and below the axis.
TYPE: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

NOTES: Attribute CHTICJ controls the horizontal axis tick mark positioning (i.e., above, below,etc.). The attribute CHLABJ controls the positioning of the tick mark labels.

RESTRICTIONS: none.

ERRORS:
CHTICJ: IMPROPER ENTRY GIVEN FOR POSITIONING OF TICK MARKS. VALUE MUST BE 0, 1, 2, OR 3. DEFAULT VALUE IS USED.



CHTROT - Set Angle of Rotation of Horizontal Axis Tick Mark Labels

---

LANGUAGE: FORTRAN 77

PURPOSE: Set angle of rotation of horizontal axis tick mark label

USE: CALL CHTROT(IV)

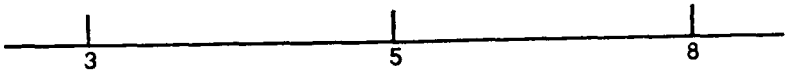
WHERE: IV - Rotation angle
TYPE:Integer

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

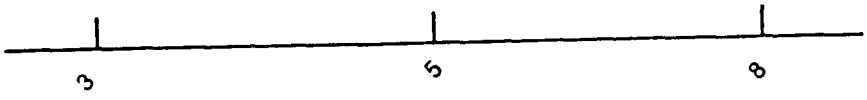
RESTRICTIONS: 1) Angles are entered in as integer values.

ERRORS:

```
CHTROT:  IV MUST BE AN INTEGER.
         ATTEMPTED TO PASS IV= iv
         THE DEFAULT VALUE IS USED
```



HORIZONTAL LINE 1
CALL CHTROT(0)



HORIZONTAL LINE 1
CALL CHTROT(45)

CKEYIN - Initialize Key Related Variables

LANGUAGE FORTRAN 77

PURPOSE: Initialize key related variables (i.e., ISTORE and KEYCHR). This routine will reset all key related variables to their defaults. Thus existing keys using these variables will be overwritten. To plot a key, use the following sequence of calls to the CGL:

- Initialization - CALL CKEYIN(....) before constructing the label for the first line in the first key.
- Label - CALL CKEYLB(....) for each line to save off current line information to be plotted by a call to CKEYPL.
- Plot key - CALL CKEYPL(.....) once all lines have been created, send the key box coordinates and plot the key.

Note: If the same key variables are used for more than one plot, all values are valid until another call is made to CKEYIN.

USE: CALL CKEYIN(ISTORE,KEYCHR,LINES,NCOL,NTLINS)

where:

ISTORE - Storage array for attributes used by the CGL. Must be dimensioned: INTEGER(15,LINES) type:INTEGER (returned)

KEYCHR - Storage array for line labels. Line labels may have more than one column. If so each column must be specified using a delimiter. The storage array must be able to handle the longest label(s) for any line. Must be dimensioned: CHARACTER*?? KEYCHR(NCOL,LINE). (where ?? is the largest expected character length of all the labels). type:CHARACTER (returned)

LINES - Number of lines to be keyed. range:greater than 0. type:INTEGER

NCOL - Number of columns in the key. range: greater than 0. type: INTEGER

NTLINS - Number of lines to key column titles (in CCTLS) to be stored

in COMMON CKEYDT as NLINS.
range:greater than or equal to 0.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) Only one key may be built at a time per ISTORE and KEYCHR.

ERRORS:

CKEYIN: LINES MUST BE AN INTEGER .GT. 0.
ATTEMPTED TO PASS LINES = lines
THIS ROUTINE IS NOT EXECUTED.

CKEYIN: NCOL MUST BE AN INTEGER .GT. 0.
ATTEMPTED TO PASS NCOL= ncol
THIS ROUTINE IS NOT EXECUTED.

CKEYIN: NTLINS MUST BE AN INTEGER .GE. 0.
ATTEMPTED TO PASS NTLINS= ntlins
THIS ROUTINE IS NOT EXECUTED.

CKEYLB - Label the Key

LANGUAGE: FORTRAN 77

PURPOSE: This routine labels the key for a given line using current attributes.

USE: CALL CKEYLB(ISTORE,KEYCHR,ICODE,LABEL)

where: ISTORE - Storage array for line and symbol attributes. (See CKEYIN)
type:INTEGER (returned)

KEYCHR - Storage array for line labels.
type:CHARACTER(returned) (See CKEYIN)

ICODE - Code deciding what is to be used in key.
type:INTEGER
= 1 - CGL line only
= 2 - CGL marker only
= 3 - CGL marker and line
= -1 - DI-3000 line only
= -2 - DI-3000 marker only
= -3 - DI-3000 marker and line
= -4 - Rectangle

LABEL - Text string for this line.
type:CHARACTER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) Storage area arrays should be initialized using subroutine CKEYIN. Only one key may be built at a time per ISTORE and KEYCHR.
2) DI-3000 markers are device dependent, therefore the height and width are not selectable by the user. Any key using DI-3000 markers are only approximate. CGL markers and line patterns are preferred. approximate (i.e., CGL markers are recommended).
3) CKEYLB should be invoked just before or after the call to have the line and/or symbol plotted, in order to retain the current attributes.

ERRORS:
CKEYLB: ICODE MUST BE AN INTEGER .GE. -4 .AND. .LE. 3.
ATTEMPTED TO PASS ICODE= icode
THIS ROUTINE IS NOT EXECUTED.

CKEYPL - Plot the Key

LANGUAGE: FORTRAN 77

PURPOSE: To plot the key within the user's box dimensions for the CGL routines. This routine uses the variables ISTORE and KEYCHR initialized by CKEYIN and set by CKEYLB.

USE: CALL CKEYPL(ISTORE,KEYCHR,CTTL,CCTLS,BPOS,KPT,JCOL,NHDR)

where:

ISTORE - Storage array for line and symbol attributes (see CKEYIN).
type:INTEGER

A description of ISTORE's content:

location: description

01-10	: DI-3000 current internal environment
1	: the current color (J1IGET(5)).
2	: the current intensity.
3	: the current line style.
4	: the current line width.
5	: the current pen.
6	: the current polygon edge style.
7	: the current polygon interior style.
8	: the current polygon color index.
9	: the current polygon intensity index.
10	: the current marker symbol.
11-15	: additional attributes used by CGL.
11	: CGL marker symbol number.
12	: marker symbol size.
13	: plot type.
14	: line pattern type.
15	: magnification factor.

KEYCHR - Character array (See CKEYIN) key lines.
type:CHARACTER

CTTL - Character variable contains key title.
type:CHARACTER

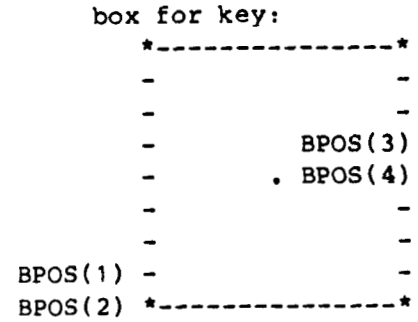
CCTLS - Character array (NCOL,NTLINS) for column titles. See CKEYIN for description of NCOL and NTLINS.
type:CHARACTER

BPOS - Storage array for key box coordinates.
REAL BPOS(4) (user supplied)
type:REAL (returned)

positions for key box:
BPOS(1): X coordinate for specified box corner.
BPOS(2): Y coordinate for specified box corner.

BPOS(3): X coordinate for box center point.
BPOS(4): Y coordinate for box center point.

KPT - Type of box coordinates input.
type:INTEGER (returned)
1 - Center point given in BPOS(3),BPOS(4)
2 - Lower left corner in BPOS(1),BPOS(2)
3 - Lower right corner in BPOS(1),BPOS(2)
4 - Upper left corner in BPOS(1),BPOS(2)
5 - Upper right corner in BPOS(1),BPOS(2)



NOTE: **** If the center point is supplied (KPT=1) the box's two diagonal points will be determined by the key's dimensions, and output in BPOS(1),BPOS(2) and BPOS(3),BPOS(4) without plotting the key.

JCOL - Array containing the justification codes
INTEGER JCOL(LCOL)
type:INTEGER
for each column.
1 = left justification
2 = center justification
3 = right justification

NHDR - Code specifying if a key title is given
type:INTEGER
0 = no header
1 = header given in CTTL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NKLPLN - Key line pattern length (in inches). (Default=1.)
Range : REAL values greater then 0.
Type:REAL

NPBCOL - Distance (in inches) between columns and between title and the identification column in the key. (Default=.25)
Range : REAL values greater than 0.
Type:REAL

NPBLIN - Distance between lines in the key (in inches).

(Default=.081)
Range : REAL values greater than 0.
Type:REAL

NPDKEY - Margins for the 4 sides of the key (in inches).
(Default=.188)
Range : REAL values greater than or equal to 0.
Type:REAL

NSYMSZ - Size of symbol.
Range : REAL values greater than zero.
Type:REAL

RESTRICTIONS:

1) Associated attributes must be described in terms of the current world coordinates. The attributes associated with height, width, and length, are converted and stored in virtual coordinates.
2) The DI-3000 marker is device dependent, therefore only an approximation of its size is used in CKEYPL's calculations. Since the DI-3000 marker cannot be scaled, then if CKEYPL is required to rescale, the resultant key output will probably be in error. Thus, the use of DI-3000 marker in conjunction with CKEYPL is left to the user's discretion.
3) Seqment must be open.

ERRORS:

CKEYPL: NHDR MUST BE 0 OR 1.
ATTEMPTED TO PASS NHDR= nhdr
THIS ROUTINE IS NOT EXECUTED.

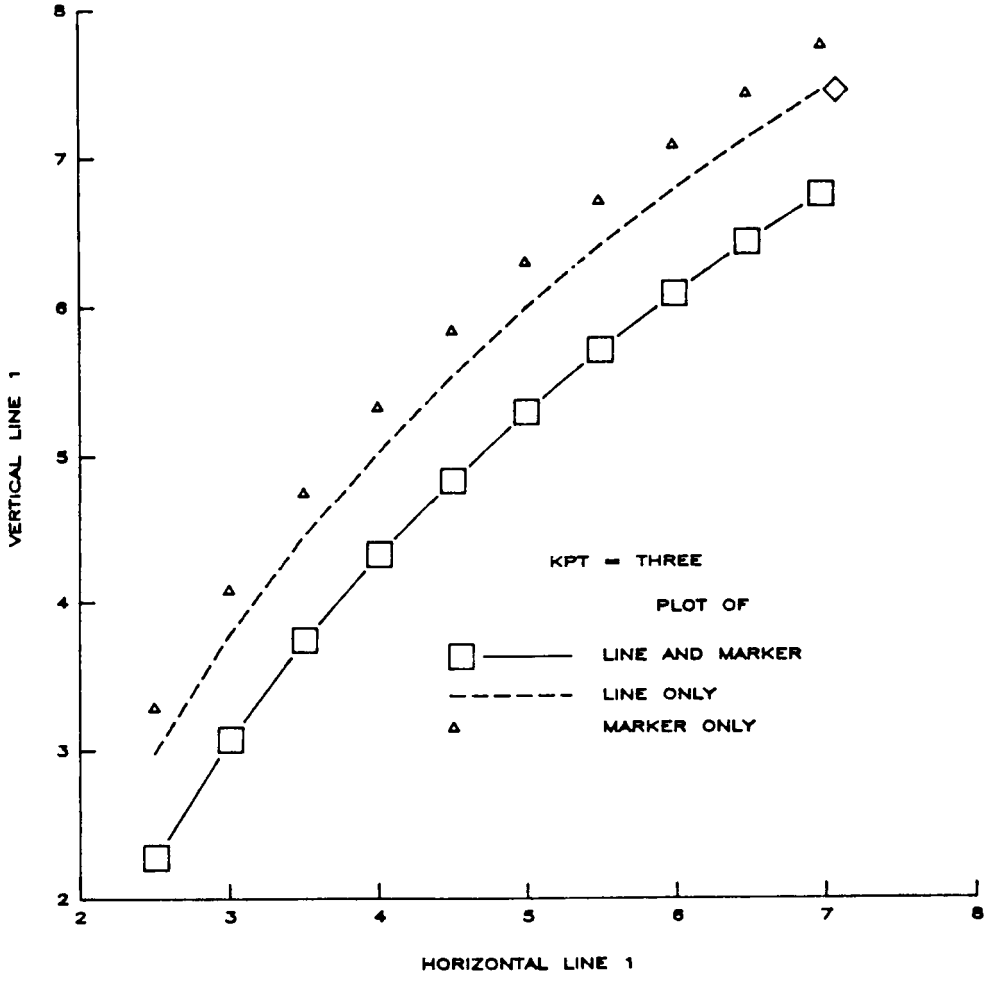
CKEYPL: INVALID VALUE FOR KPT.'
THE RANGE OF "KPT" [1,5] (INTEGER).'
ILLEGAL VALUE IS ',KPT
NO DEFAULT VALUE CAN BE ASSIGNED.'
THE ROUTINE IS NOT EXECUTED.'

NOTE: **** If the center point is supplied (KPT=1) the box's two diagonal points will be determined by the key's dimensions, and output in BPOS(1),BPOS(2) and BPOS(3),BPOS(4) without plotting the key.

EXAMPLE: See appendix B for an example of generating a key in context. The partial program below shows the dimensions of variables, and the interrelationship among routines, in key generation.

```
      PROGRAM CKEYPL1
      PARAMETER(NCOLS=1,NTLINS=1,NLINS=3,NPTS=10)
      INTEGER ISTORE(15,NLINS)
      CHARACTER KEYCHR(NCOLS,NLINS)*15,KEYLAB(NLINS)*15
      REAL BPOS(4)
      DATA ICODE/3,1,2/,BPOS/3.,6.,0.,0./
      DATA KEYLAB/'LINE AND MARKER','LINE ONLY','MARKER ONLY'/

      CALL JBEGIN
      CALL CBEGIN
      CALL CVSPAC(...)
      CALL JWINDO(...)
      CALL JOPEN
       .......
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
      DO 10 I=1,NLINS
        CALL CSET(...)
         .......
        CALL CKEYLB(ISTORE,KEYCHR,ICODE,KEYLAB)
10      CALL CLNPLT(...)
     CALL CKEYPL(ISTORE,KEYCHR,CTTL,CCTLS,BPOS,KPT,JCOL,NHDR)
```



CKEYPL1: EMBEDDED KEY INVOCATIONS.

CKEYTC - Set Key Text to Normal or Color of Key Symbol

LANGUAGE: FORTRAN 77

PURPOSE: Set key text to normal or color of key symbol.

USE: CALL CKEYTC(LV)

WHERE: LV -
Flag to set key text to normal or color
.TRUE. - Key text is normal (default).
.FALSE.- Key text is same color as symbol
TYPE:Logical

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CKLPLN - Set Key Line Pattern Length

LANGUAGE: FORTRAN 77

PURPOSE: Set key line pattern length

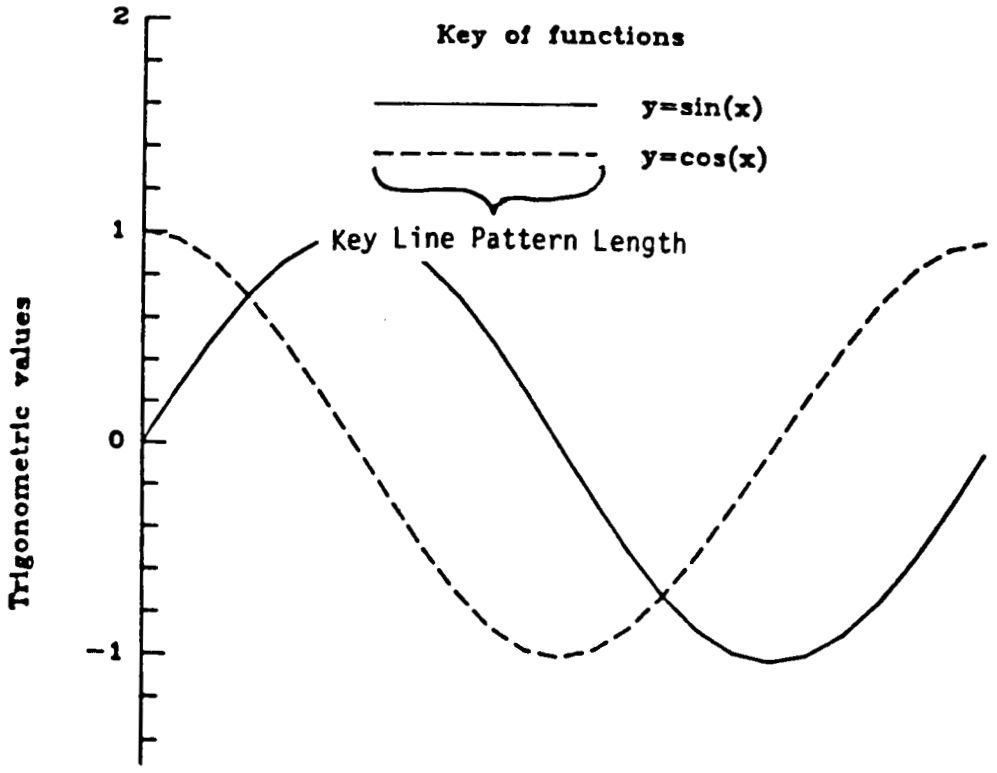
USE: CALL CKLPLN(RV)

where: RV -
Key line pattern length (world coordinates).
(default = 1.)
Range: Real values greater than zero.
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CKLPLN: RV MUST BE .GT. 0.
ATTEMPTED TO PASS RV= rv.
THE DEFAULT VALUE IS USED.



CLABP - Set the Size of the Minor Tick Marks as a % of Major

LANGUAGE: FORTRAN 77

PURPOSE: Set the size of the minor tick marks as a % of major.

USE: Call CLABP(RV)

Where: RV -
The size of the minor tick marks as a percentage of the size of the major tick marks. (Default=2/3)
Range: Real values greater than zero.
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

```
CLABP:  INVALID VALUE FOR RV.
        THE RANGE OF "NLABP" IS .GT. ZERO (REAL).
        ILLEGAL VALUE IS RV
        THE DEFAULT VALUE IS USED.
```

CLEAF - Set Interleave Factor

LANGUAGE: FORTRAN 77

PURPOSE: Set interleave factor. Currently, this attribute is only used by CLNPLT.

USE: CALL CLEAF(IV)

where: IV
Interleaf Factor
1 - values are stored sequentially(default)
2 - values stored in every other location
N - values stored in every N'th location
RANGE: INTEGER values greater than zero
Type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CLEAF: INVALID VALUE FOR IV.
THE RANGE OF "NLEAF" IS .GT. ZERO (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.

EXAMPLE: given:

```
REAL A(10)
DATA A/1.,2.,3.,4.,5.,6.,7.,8.,9.,10./
```

then
CALL CLEAF(1) will allow CLNPLT to access {1.,..,10.}
CALL CLEAF(2) ==> {1.,3.,5.,7.,9.}
CALL CLEAF(3) ==> {1.,4.,7.,10.}
CALL CLEAF(IV)==> {1st, 1+iv position, 1+2*iv, etc.}

CLGRID - Draw Log-Log, Semi-log or Linear Grid

LANGUAGE: FORTRAN 77

PURPOSE: To draw a log-log, semi-log (in either axis) or linear grid with rectangular or square area(s) remaining blank.

USE: CALL CLGRID (XSC,YSC,IXC,IYC,XGS,YGS,XGI,YGI, BLANK,NBLANK)

XSC
- log - Number of units per log cycle for X.
- lin - Total X axis length.
- type:REAL

YSC
- log - Number of units per log cycle for Y.
- lin - Total Y axis length.
- type:REAL

IXC
- log - Number of log cycles along X axis.
- lin - -1 must be loaded.
- type:INTEGER

IYC
- log - Number of log cycles along Y axis.
- lin - -1 must be loaded.
- type:INTEGER

XGS
- log - Starting grid line for X (1.0 To 9.0). This also controls ending grid line.
- lin - 0. must be loaded.
- type:REAL

YGS
- log - Starting grid line for Y (1.0 to 9.0). This also controls ending grid line.
- lin - 0. must be loaded.
- type:REAL

XGI
- log - Interval between secondary grid lines in each cycle. Interval is doubled above 6 primary grid lines for each cycle. Each integer grid line must also be an interval line. If not, interval is modified so this condition will be satisfied.
  - 1.0 no secondary grid lines
  - .5 one secondary grid line
  - .25 three secondary grid lines
  - -XGI use tick marks not full grid lines
  - +XGI use full grid lines
  - type:REAL
- lin - Interval between grid lines for entire axis.
  - -XGI use tick marks not full grid lines
  - +XGI use full grid lines
  - type:REAL

YGI
- log - Interval between secondary grid lines in each cycle. This interval is doubled above 6 primary grid lines per cycle. Each integer grid line must also be an

interval line. If not, interval is modified so this condition will be met.
1.0 no secondary grid lines
.5 one secondary grid line
.25 three secondary grid lines
-YGI use tick marks not full grid lines
+YGI use full grid lines

lin - Interval between grid lines for entire axis.
-YGI use tick marks not full grid lines
+YGI use full grid lines
type:REAL

BLANK An array containing the coordinates of the rectangular or square blank area(s).
For the first blank area
BLANK(1) - X lower left coordinate
BLANK(2) - Y lower left coordinate
BLANK(3) - X upper right coordinate
BLANK(4) - Y upper right coordinate
For the second blank area
BLANK(5) - X lower left coordinate
BLANK(6) - Y lower left coordinate
BLANK(7) - X upper right coordinate
BLANK(8) - Y upper right coordinate
type:REAL

NBLANK An integer specifying the number of blank areas. A maximum of 10 blank areas are allowed.
range: 0 to 10. If NBLANK is 0, then a dummy variable should be passed for BLANK.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:
NLABP - The size of the minor tick marks as a percentage of the size of the major tick marks. (Default=2/3)
Range : REAL values greater than 0.
Type:REAL

NOTES:
(1) Major tick mark lengths -
Horizontal axis : current character height.
Vertical axis : current character width.

(2) Minor tick mark lengths -
Major tick mark length*NLABP
(where NLABP is desired percentage of major tick mark).

RESTRICTIONS:
1) Segment must be opened.
2) The routine will first move the pen to

X=0 and Y=0. Only perpendicular axes
are allowed (no skewed grids).

METHOD: For log-grids the cycle size, number of cycles
and choice of grid lines within each cycle are
specified independently for each axis.
For a log cycle, starting grid lines and
interval between lines are specified.
This interval is used until primary grid lines over
six are encountered. Then the interval is doubled.

For a linear grid, the axis length and
interval between lines are specified.
For both linear and log grids, user may choose
between full grid lines or 0.25-unit
abbreviated grid lines independently for each axis.

OTHER CODING INFORMATION

If repeat interval .GT. one after doubled, it
will be set to one. If it is one, the
repeat interval lines will coincide with
the integer grid lines.

ERRORS:

CLGRID: IXC MUST BE AN INTEGER.
ATTEMPTED TO PASS IXC= ixc
THIS ROUTINE IS NOT EXECUTED.

CLGRID: IYC MUST BE AN INTEGER.
ATTEMPTED TO PASS IYC= iyc
THIS ROUTINE IS NOT EXECUTED.

CLGRID: NBLANK MUST BE AN INTEGER .ge. 0 .AND. .LE. 10.
ATTEMPTED TO PASS NBLANK= nblank
THIS ROUTINE IS NOT EXECUTED.

CLNPAT - Set Line Pattern for NASA Standard Line Patterns

LANGUAGE: FORTRAN 77

PURPOSE: Set line pattern for NASA standard line patterns.

USE: CALL CLNPAT(NLNPAT)

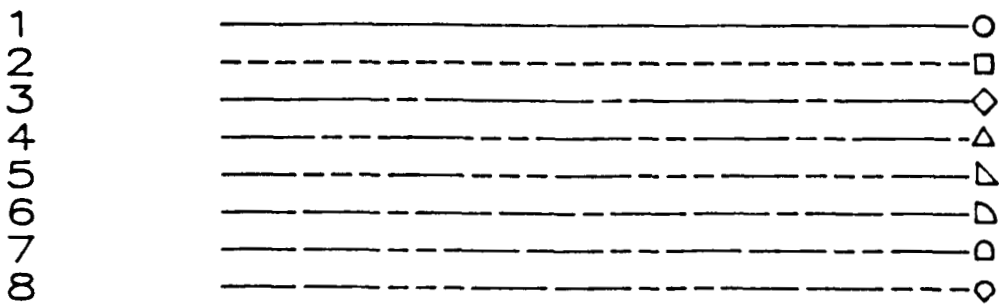
NLNPAT Line pattern for NASA standard line patterns
1 Solid line (default)
2 - 8 NASA dashed patterns
range: Integer values from 1 to 8.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CLNPAT: INVALID VALUE FOR IV.
THE RANGE OF "NLNPAT" IS 1 TO 8 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.

LINE PATTERNS



CLNPLT - Draw a Line between and/or Draw NASA Standard Symbol

---

LANGUAGE: FORTRAN 77

PURPOSE: To draw a line between and/or draw NASA standard symbol or to draw continuous dashed line with a symbol at the end of a line.

The characteristics of the line and/or symbols are set by defining the appropriate CGL attributes.

USE: CALL CLNPLT(X,Y,N)

X - X array to plotted (X world coordinates).
type:REAL
Y - Y array to plotted (Y world coordinates).
type:REAL
N - Number of points to be plotted.
type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: line and polygon attributes.
2) CGL attributes:

NLEAF - Interleave factor for line plots (CLNPLT).
1 values are stored sequentially (Default)
2 values stored in every other location.
Range : INTEGER values greater than 0.
Type:INTEGER

NPLTYP - Code for line plot type (CLNPLT).
Magnitude specifies the alternate points for symbol
positive for line and symbol plot
negative for symbol only
0 for line plot with specified symbol at end
1 for symbol for each data point and line between (default = 1)
-2 for symbol for every other data point
-n for symbol for every other n'th data point
Range : any INTEGER value.
Type:INTEGER

NSYMNO - The Type of symbol to be drawn:

NSYMNO =	0	no symbol
	1	circle
	2	square (default)
	3	diamond
	4	triangle
	5	right triangle
	6	quadrant
	7	dog house
	8	fan

```
 9   long diamond
10   house
11   circle          with plus
12   square          with plus
13   diamond         with plus
14   triangle        with plus
15   right triangle  with plus
16   quadrant        with plus
17   dog house       with plus
18   fan             with plus
19   long diamond    with plus
20   house           with plus
21   dot
22   plus sign
```

If a flag is desired, add 100, 200,..., 900 to NSYMNO, such that:

```
  1 - symbol
101 - symbol + upper right line
201 - symbol + upper left line
301 - symbol + lower left line
401 - symbol + lower right line
501 - symbol + upper right flag
601 - symbol + upper left flag
701 - symbol + lower left flag
801 - symbol + lower right flag
901 - symbol is solid filled
```

Range : 0 to 22,100 to 122,...,900 to 922.
Type:INTEGER

NSYMSZ - Size of symbol.
Range : REAL values greater than zero.
Type:REAL

NLNPAT - Line pattern for NASA standard line patterns.
1 solid line (Default)
2 - 8 NASA dashed patterns
Range : INTEGER values between 1 and 8 (inclusive).
Type:INTEGER

NMAGFL - Magnification factor of line pattern (CLNPLT). Each element of the selected line pattern is multiplied by the value NMAGFL. (Default = 1.0)
Range : REAL values greater then 0.
Type:REAL

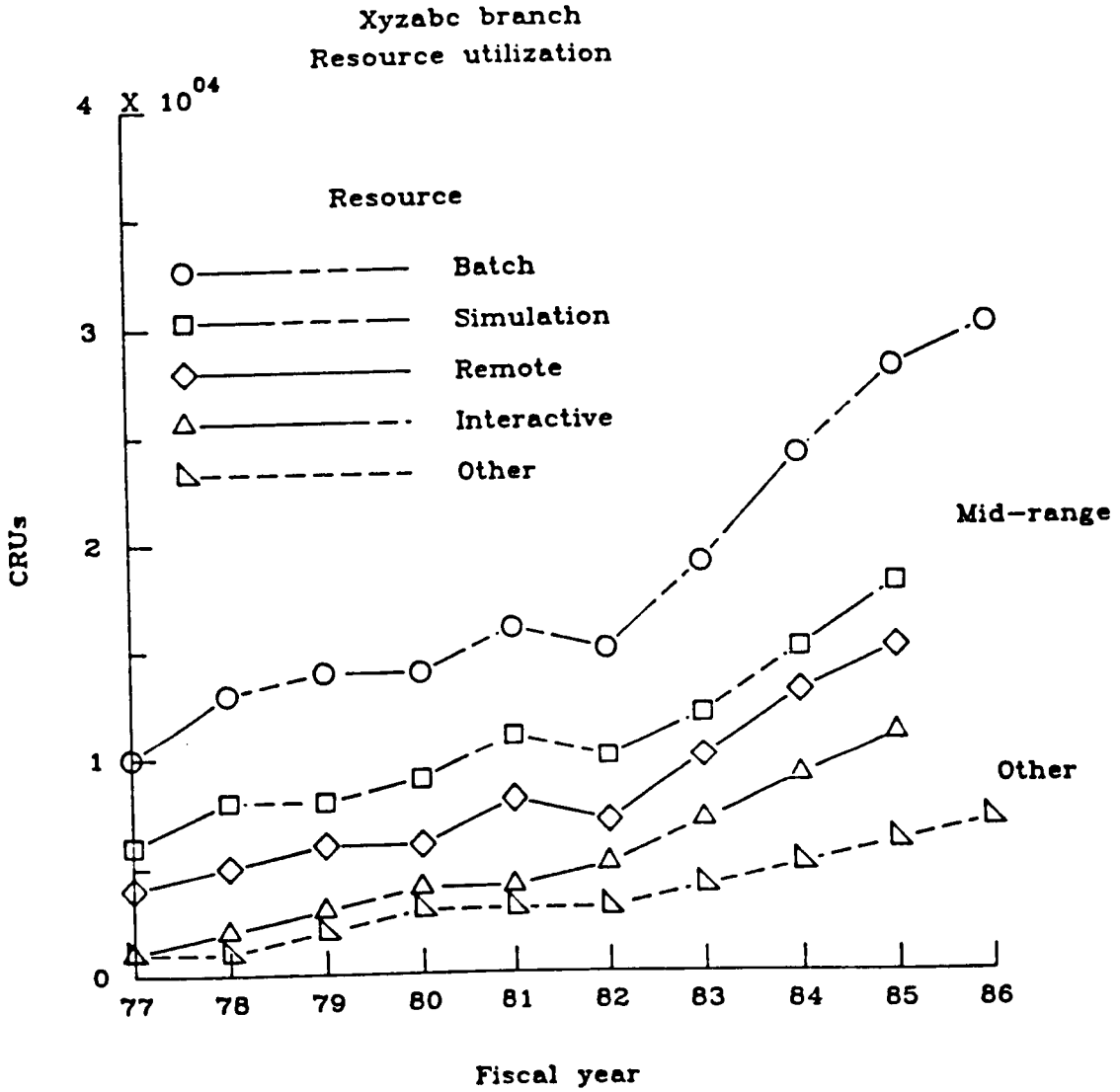
NOTE: To draw a line only (i.e., no symbols, nor spaces for symbols) set NSYMNO to 0.

RESTRICTIONS: 1) Segment must be opened.
2) The latest world coordinate system (See DI-3000's JWINDO) should be related to the values in the X and Y arrays.

ERRORS:

CLNPLT: N MUST BE AN INTEGER .GT. 0.
ATTEMPTED TO PASS N= n
THIS ROUTINE IS NOT EXECUTED.

CLNPLT: NLEAF MUST BE AN INTEGER .GT. 0.
ATTEMPTED TO PASS NLEAF= nleaf
THIS ROUTINE IS NOT EXECUTED.



CMAGFL - Set Magnification Factor of Line Pattern

LANGUAGE: FORTRAN 77

PURPOSE: Set magnification factor of line pattern.

USE: CALL CMAGFL(NMAGFL)

where: NMAGFL
Magnification factor of line pattern. Each element of the selected line pattern is multiplied by the value of NMAGFL. (Def = 1.)
Range: Real values greater than zero.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

```
CMAGFL:  INVALID VALUE FOR NMAGFL.
         THE RANGE OF NMAGFL IS .GT. 0 (REAL).
         ILLEGAL VALUE IS nmagfl
         THE DEFAULT VALUE IS USED.
```

CMNMX - Find the Minimum and Maximum of an Array

LANGUAGE: FORTRAN 77

PURPOSE: Subroutine to find the minimum and maximum of an array.

USE: CALL CMNMX(A,N,XMIN,XMAX)

Where: A - The array to be examined.
Type:REAL
N - Number of elements in array A.
Type:INTEGER
XMIN,
XMAX - Minimum and maximum values found.
(returned)
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) N must be 2 or greater.

ERRORS:
CMNMX: N MUST BE INTEGER .GT. 0.
ATTEMPTED TO PASS N= n
THIS ROUTINE IS NOT EXECUTED.

CNASA - Draw NASA LOGO

LANGUAGE: FORTRAN 77 (ANSI)

PURPOSE: To draw the NASA LOGO.
The policy governing the use of the NASA LOGO is documented in the NASA LANGLEY RESEARCH CENTER Management Manual (NMI 1020.1e paragraph 13.b).

USE: CALL CNASA (THETA,HEIGHT,ICOLOR)

where -

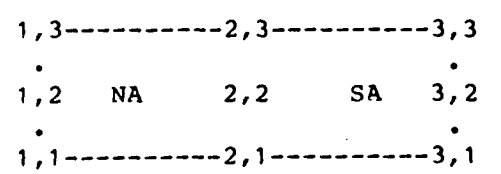
THETA : Angle (degrees) of inclination of the logo about the current position. See the note below for a further explanation.
type:REAL

HEIGHT : Height of the logo (world coordinates).
range: greater than 0.
type:REAL

ICOLOR : Integer value showing color of NASA LOGO polygon fill.
0 - black
1 - red
2 - normal
3 - complement of normal
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

Notes: The angle of inclination (THETA) is applied relative to the NASA logo as indicated by the current text justification (JJUST). For example, a current text justification of 2,2 (center,center) will rotate the NASA logo about the logo's center-center location. The following figure provides a summary of the text justification.

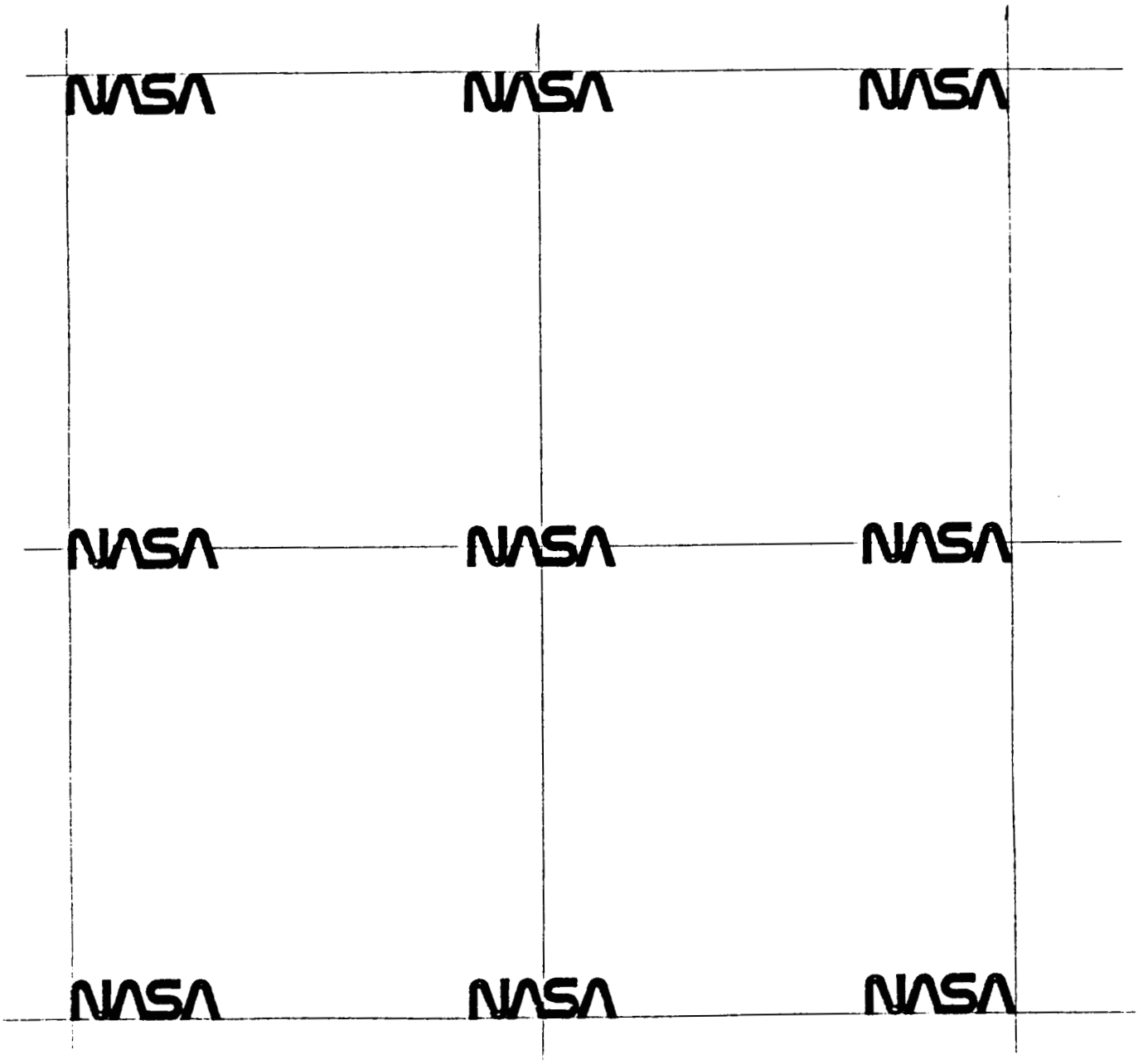


RESTRICTIONS: 1) Segment must be opened.
2) HEIGHT must be greater than zero.
3) The ratio of HEIGHT to width is 1:3.76.
4) Each logo letter is a polygon with about 244 points. Therefore, the user must ensure that the device driver can support polygons with that many points.

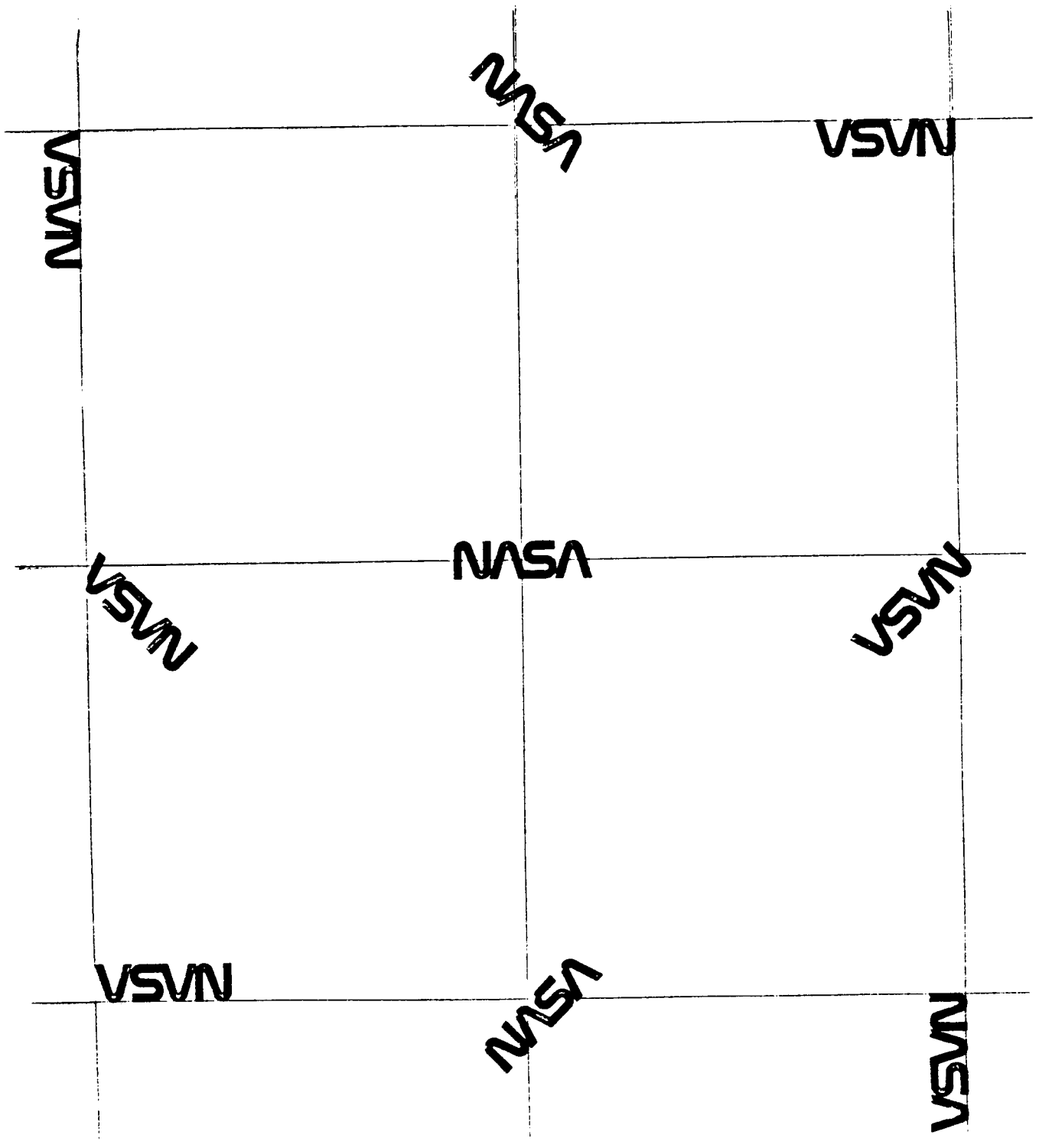
ERRORS:

CNASA: INVALID VALUE FOR ICOLOR.
THE RANGE OF ICOLOR IS 0,1,2,3.
THE DEFAULT OF 1 (RED) IS USED.

CNASA: INVALID VALUE FOR HEIGHT.
THE RANGE OF "HEIGHT" IS .GT. ZERO.
THE ROUTINE IS NOT EXECUTED.



This figure illustrates justification of the NASA Logo about a point.



This figure illustrates rotation of the NASA Logo about a point

CNOTE - Output a Graphics String Rotated by an Angle

LANGUAGE: FORTRAN 77

PURPOSE: To output a graphics string STR, rotated about the X-axis by ANGLE degrees. The string is output in the precision determined by PREC.

USE: CALL CNOTE(STR,ANGLE,PREC)

where:

- STR - Graphics string to be output.
  type:CHARACTER(*)
- ANGLE - Angle of rotation (degrees).
  type:REAL
- PREC - Type of graphics precision.
  Range : 1 - string precision text.
  2 - character precision text.
  3 - stroke precision text.
  4 - graphics art precision text.
  5 - di-textpro precision text.
  Type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes:
   - NTCOL - Text color.
     Range : 0 or greater (see underlying package).
     Type:INTEGER

RESTRICTIONS: Segment must be open.

ERRORS:

CNOTE: PREC MUST BE AN INTEGER 1 TO 5.
ATTEMPTED TO PASS PREC= prec
THIS ROUTINE IS NOT EXECUTED.

NOTE: 1) The current character attributes are used (except JBASE and JPLANE).
2) The vectors for JBASE and JPLANE may be altered.

CPBCOL - Set Distance Between Columns in the Key

LANGUAGE: FORTRAN 77

PURPOSE: Set distance between columns.

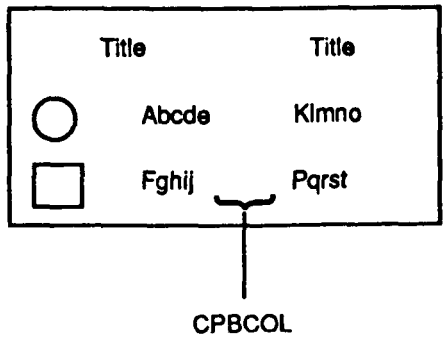
USE: CALL CPBCOL(RV)

where: RV -
Distance (in inches) between columns and between title and the identification column in the key.(Default=.25)
Range: REAL values greater than zero.
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CPBCOL: INVALID VALUE FOR RV.
THE RANGE OF "NPBCOL" IS .GT. ZERO (REAL).
ILLEGAL VALUE IS rv
THE DEFAULT VALUE IS USED.



CPBLIN - Set Horizontal Distance Between Lines in the Key

LANGUAGE: FORTRAN 77

PURPOSE: Set horizontal distance between lines in the key.

USE: CALL CPBLIN(RV)

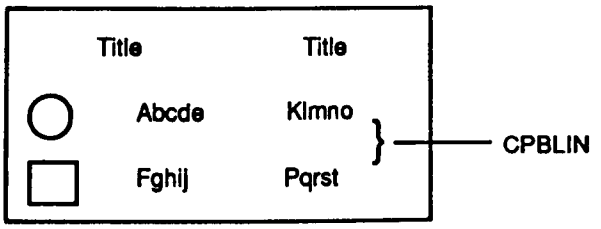
where: RV -
Distance between lines (in inches).
(Default = .081)
Range: REAL values greater than zero
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

CPBLIN: INVALID VALUE FOR RV.
THE RANGE OF "NPBLIN" IS .GT. 0. (REAL).
ILLEGAL VAULE IS rv
THE DEFAULT VALUE IS USED.



CPBXVL - Return X World Coordinates for X Inches

LANGUAGE: FORTRAN 77

PURPOSE: Determine amount of DI-3000 X world coordinates for a given length in X inches.

USE: VAL = CPBXVL(XINCH)

where: XINCH - Value in inches.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: CVSPAC or CVPORT must be called prior to this routine.

ERRORS: none.

CPBYVL - Return Y World Coordinates for Y Inches

LANGUAGE: FORTRAN 77

PURPOSE: Determine amount of DI-3000 Y world coordinates for a given length in Y inches.

USE: VAL = CPBYVL(YINCH)

where: YINCH - Value in inches.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: CVSPAC or CVPORT must be called prior to this routine.

ERRORS: none.

CPDKEY - Set Margins for the Four Sides of the Key

LANGUAGE: FORTRAN 77

PURPOSE: Set margins for the four sides of the key.

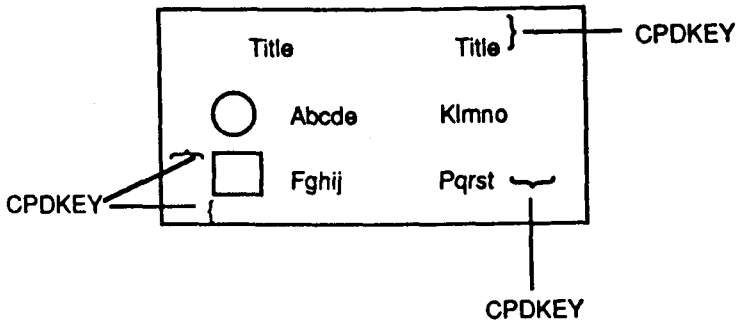
USE: CALL CPDKEY(RV)

where: RV -
Margins for the four sides of the key.
(Default = .188)
Range: Real values greater than or equal to 0.
Type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CPDKEY: INVALID VALUE FOR RV.
THE RANGE OF "NPDKEY" IS .GT. 0. (REAL).
ILLEGAL VALUE IS rv
THE DEFAULT VALUE IS USED.



CPLGRD - Draw a Polar Grid

LANGUAGE: FORTRAN

PURPOSE: To draw a polar grid, which consists of a pair of intersecting axes, concentric circles, and tick marks on the outer circles. The tick marks can also be extended to the origin, thus forming radii.

USE: CALL CPLGRD(X,Y,R,TMAJ,TICDEG)
where:

X,Y	- The coordinates of the center of the polar grid. type:REAL
R	- Length of the radius of the grid. range: greater than 0. type:REAL
TMAJ	- Length along the axis separating concentric circles. + label axis - axis not labeled range: not equal to 0. type:REAL
TICDEG	- The number of degrees between tick marks. + tick mark only - tick mark extended to grid origin type:REAL

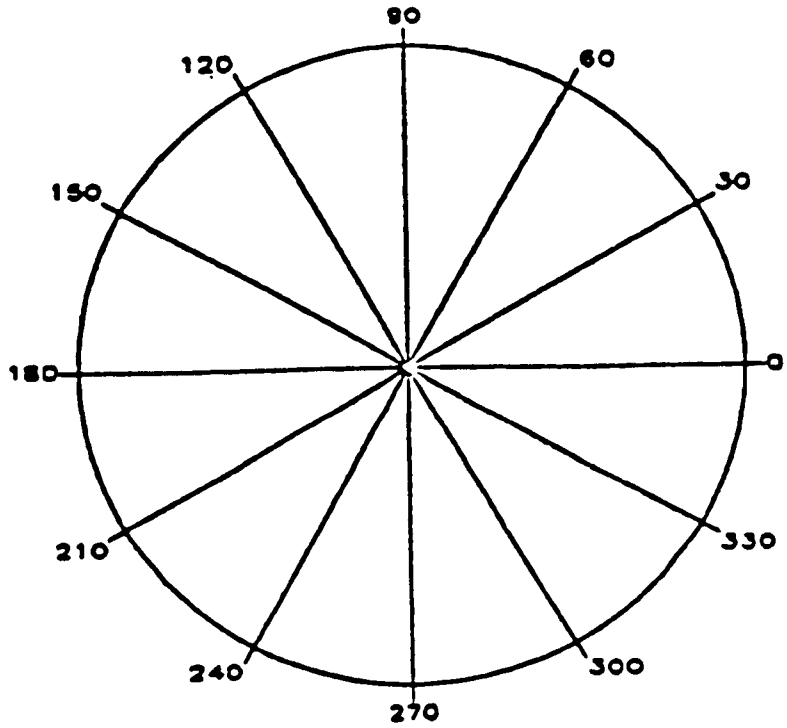
ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes.
2) CGL attributes: none.

RESTRICTIONS: - A segment must be open.
- X, Y, and R are in world coordinates.

ERRORS:

CPLGRD: R MUST BE .GT. 0.
ATTEMPTED TO PASS R= r
THIS ROUTINE IS NOT EXECUTED.

CPLGRD: TMAJ CANNOT .EQ. 0.
ATTEMPTED TO PASS TMAJ= tmaj
THIS ROUTINE IS NOT EXECUTED.


90
120
60
150
30
180
0
210
330
240
300
270

CPLTST - Send a Message to the Plotter Operator

LANGUAGE: FORTRAN 77

PURPOSE: This routine is used to send a message to the plot operator. It is intended for use with static plotters operated by the ACD support personnel.

USE: CALL CPLTST (MESSAGE)

MESSAGE - is the message for the operator.
Type: character*(*)

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: Only the first 80 characters are used.

ERRORS: none.

CPLTYP - Set Code for Plot Type

LANGUAGE: FORTRAN 77

PURPOSE: Set code for plot type.

USE: CALL CPLTYP(IV)

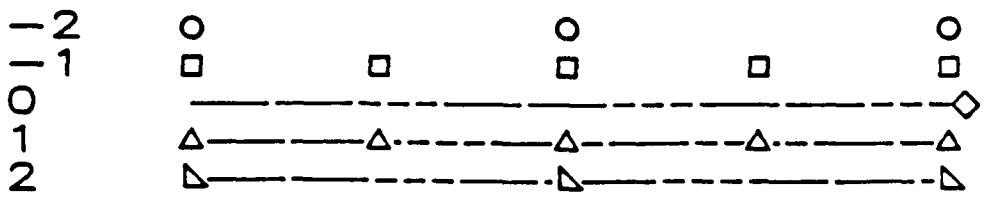
WHERE: IV -
Code for plot type.
Magnitude specifies the alternate points for symbol:
Positive for line and symbol;
Negative for symbol only.
0 For line plot with the specified symbol at end;
1 For symbol for each data point and line between;
(default = 1)
-2 For symbol for every other data point;
-N For symbol for every other N'th data point.
RANGE : Any INTEGER value.
TYPE:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

NOTES: Attribute CLEAF determines which elements frlements will be selected, CPLTYP determines which selected elements will be represented by a symbol.

ERRORS:
CPLTYP: INVALID VALUE FOR IV.
THE RANGE OF "NPLTYP" IS ANY INTEGER.
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



CPNTPT - Draw a NASA Symbol at the Specified Location

LANGUAGE: FORTRAN 77

PURPOSE: Draws a NASA point symbol at the specified location. The type and size of the symbol are specified by setting the associated attributes listed below.

USE: CALL CPNTPT(XO,YO)
where: XO - Center of symbol X world coordinates.
type:REAL
YO - Center of symbol Y world coordinates.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: polygon attributes.
2) CGL attributes:
NSYMNO - The Type of symbol to be drawn:

NSYMNO =	0	no symbol	
	1	circle	
	2	square (default)	
	3	diamond	
	4	triangle	
	5	right triangle	
	6	quadrant	
	7	dog house	
	8	fan	
	9	long diamond	
	10	house	
	11	circle	with plus
	12	square	with plus
	13	diamond	with plus
	14	triangle	with plus
	15	right triangle	with plus
	16	quadrant	with plus
	17	dog house	with plus
	18	fan	with plus
	19	long diamond	with plus
	20	house	with plus
	21	dot	
	22	plus sign	

If a flag is desired, add 100, 200,..., 900 to NSYMNO, such that:

1	- symbol
101	- symbol + upper right line
201	- symbol + upper left line
301	- symbol + lower left line
401	- symbol + lower right line
501	- symbol + upper right flag
601	- symbol + upper left flag

701 - symbol + lower left flag
801 - symbol + lower right flag
901 - symbol is solid filled
Range : 0 to 22,100 to 122,...,900 to 922.
Type:INTEGER

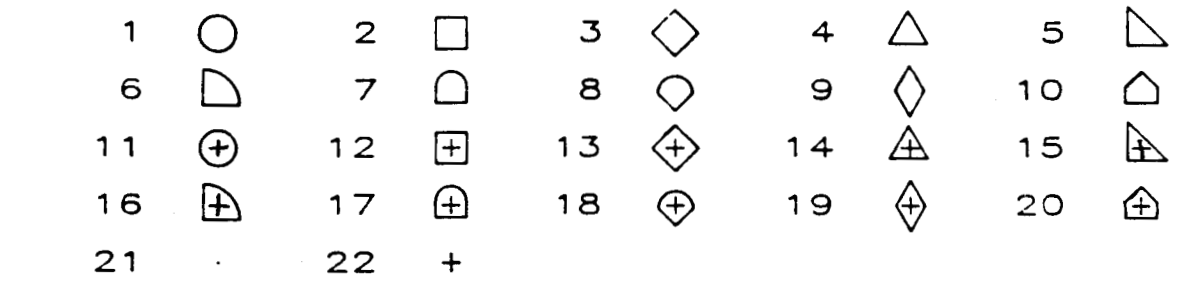
NSYMSZ - Size of symbol.
Range : REAL values greater than zero.
Type:REAL

RESTRICTIONS: 1) Segment must be open.
2) CPNTPT will set the current position (CP) to XO,YO.

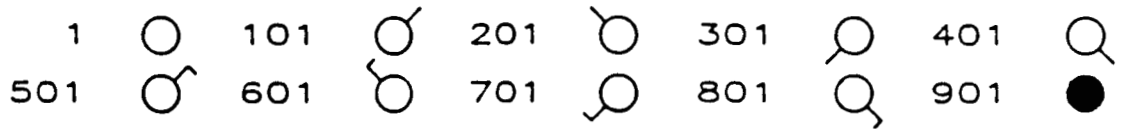
ERRORS: none.

NASA SYMBOLS

ISYM =



IF A FLAG IS WANTED, ADD 100, 200, ..., 900 TO ISYM



CREXP - Return the Integer Value between '[BSUP]' And '[ESUP]'.

LANGUAGE: FORTRAN 77

PURPOSE: To read a string "CEXP" and return an integer "IVAL" between "[BSUP]" and "[ESUP]".

USE: CALL CREXP(CEXP,IVAL)

where: CEXP - Character string to be scanned.
type:CHARACTER(*)
IVAL - Integer value returned.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: The string CEXP must contain "[BSUP]IVAL[ESUP]". IVAL is limited to two digits.

ERRORS: none.

CRLINT - Set Real to Integer Conversion Flag

LANGUAGE: FORTRAN 77

PURPOSE: Set real to integer conversion flag for any numeric text to be displayed via the CGL.

USE: CALL CRLINT(LV)

where: LV -
Real to integer conversion flag
.TRUE. - Real is plotted as integer
.FALSE. - Real is plotted as a real (default)
The numeric value is not changed. Conversion is done to the character string containing the printed number. Hence, the number is not rounded. If rounding is preferred, it should be done by the user prior to calling the conversion routines.
type:LOGICAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CSCALE - To Determine Publication Quality Axis Attributes

LANGUAGE: FORTRAN 77

PURPOSE: Determine publication quality axis attributes. Specifically, the data min and max are passed to this routine, which returns an adjusted minimum and maximum, the number of major and minor tick marks, and the appropriate number of decimal places to the left and right of the decimal point. These values will provide publication-quality labels when used on an axis.

USE:
```
CALL CSCALE(DMIN,DMAX,AMIN,AMAX,NMAJOR,NMINOR,XINCR,
            NLEFT,NRIGHT)
```

where:
- DMIN - Data minimum. (input) type:REAL
- DMAX - Data maximum. type:REAL (input)
- AMIN - Adjusted data minimum. type:REAL (returned)
- AMAX - Adjusted data maximum. type:REAL (returned)
- NMAJOR - Number of major tick marks (including the endpoints). type:INTEGER (returned)
- NMINOR - Number of minor tick marks type:INTEGER (returned)
- XINCR - The step value between min and max. XINCR set to (AMAX-AMIN)/NMAJOR-1. type:REAL (returned)
- NLEFT - Number of digits to the left of the decimal point. type:INTEGER (returned)
- NRIGHT - Number of digits to the right of the decimal point. type:INTEGER (returned)

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CSET - Set Common Graphics Library (CGL) Attributes

LANGUAGE: FORTRAN 77

PURPOSE: This routine allows the user to set CGL attributes from a single routine. The use of a single routine frees the user from finding and calling different routines, but at the expense of additional field length. If the user desires to call the specific routine, thus reducing the field length, then the routine is generally denoted as C#### for the attribute N####.

Attribute	Routine
NHBTIC	CHBTIC
N.....	C.....
NVMTIC	CVMTIC

USE: CALL CSET (ATTRIBUTE,VALUE)

where: ATTRIBUTE - option selection.
type: character
VALUE - option value.
type: determined by option

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes:

NHBTIC - Position of horizontal axis label in relation to horizontal tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
type:INTEGER

NHLABJ - Position of axis label in relation to horizontal axis. (above or below)
0 - horizontal axis label is below axis.(default)
1 - horizontal axis label is above axis.
type:INTEGER

NHLOGF - Determine if horizontal axis will have major tick marks as 10 raised to a power.
-1 - major tick mark labels decrease
0 - no major tick mark labels (default)
1 - major tick mark labels increase
TYPE:INTEGER

NHLOGS - The initial power for the horizontal log axis labels. (i.e., 10 raised to this power)
(default=0)
Range : any valid INTEGER.
Type:INTEGER

NHMTIC - Number of minor tick marks between major tick marks for the horizontal axis. (default=0)
Range : INTEGERS greater or equal to 0.
TYPE:INTEGER

NHPREC - Number of digits to the right of the decimal point in the tick mark labels of the horizontal axis.
< 0 - plot as an INTEGER (no decimal point)
= 0 - decimal point only
> 0 - plot as a REAL. Value represents the number of digits to the right of the decimal point
Default: -1
Type:INTEGER

NHTICJ - A value to indicate positioning of tick marks for horizontal axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
Default: 1
Type: INTEGER

NKLPLN - Key line pattern length (in inches). (Default=1.)
Range : REAL values greater then 0.
Type:REAL

NLABP - The size of the minor tick marks as a percentage of the size of the major tick marks. (Default=2/3)
Range : REAL values greater than 0.
Type:REAL

NLEAF - Interleave factor for line plots (CLNPLT).
1 values are stored sequentially (Default)
2 values stored in every other location.
Range : INTEGER values greater than 0.
Type:INTEGER

NLNPAT - Line pattern for NASA standard line patterns.
1 solid line (Default)
2 - 8 NASA dashed patterns
Range : INTEGER values between 1 and 8 (inclusive).
Type:INTEGER

NMAGFL - Magnification factor of line pattern (CLNPLT). Each element of the selected line pattern is multiplied by the value NMAGFL. (Default = 1.0)
Range : REAL values greater then 0.
Type:REAL

NPBCOL - Distance (in inches) between columns and between title and the identification column in the key. (Default=.25)

Range : REAL values greater than 0.
Type:REAL

NPBLIN - Distance between lines in the key (in inches).
(Default=.081)
Range : REAL values greater than 0.
Type:REAL

NPDKEY - Margins for the 4 sides of the key (in inches).
(Default=.188)
Range : REAL values greater than or equal to 0.
Type:REAL

NPLTYP - Code for line plot type (CLNPLT).
Magnitude specifies the alternate points for symbol
positive for line and symbol plot
negative for symbol only
0 for line plot with specified symbol at end
1 for symbol for each data point and line between
(default = 1)
-2 for symbol for every other data point
-n for symbol for every other n'th data point
Range : any INTEGER value.
Type:INTEGER

NRLINT - REAL to INTEGER conversion flag
.TRUE. - REAL value is plotted as an INTEGER
.FALSE. - REAL value is plotted as a REAL (default)
Type:logical

NSYMSZ - Size of symbol.
Range : REAL values greater than zero.
Type:REAL

NTPREC - Text precision.
Range : 1 - string precision text.
2 - character precision text.
3 - stroke precision text.
4 - graphics art precision text.
5 - DI-TEXTPRO precision text.
Type:INTEGER

NTCOL - Text color.
Range : 0 or greater (see underlying package).
Type:INTEGER

NVBTIC - Position of vertical axis label in relation to
to vertical tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
Type:INTEGER

NVJUST - Determines if vertical axis label is written
horizontally or vertically.

0 - vertical (default).
1 - horizontal.
Type:INTEGER

NVLABJ - Position of axis label in relation to vertical axis. (left or right)
0 - vertical axis label is left of axis.(default)
1 - vertical axis label is right of axis.
type:INTEGER

NVLOGF - Determine if vertical log axis will have major marks as 10 raised to a power.
-1 - no major tick mark labels
0 - major tick mark labels (default)
1 - major tick mark labels increase
Type:INTEGER

NVLOGS - The initial power for the vertical log axis labels.
Range: any valid INTEGER(10 raised to this power). (default 0)
Type:INTEGER

NVMTIC - Number of minor tick marks between major tick marks for the vertical axis. (default=0)
Range : INTEGER values greater than or equal to 0
Type:INTEGER

NVPREC - Number of digits to the right of the decimal point in the tick mark labels of the vertical axis.
< 0 - display as a REAL. value represents the number of digits to the right of the decimal point.
= 0 - decimal point only
> 0 - display as INTEGER (no decimal point)
default: -1
Type:INTEGER

NVTICJ - A value to indicate positioning of tick marks for vertical axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
default: 1
Type: INTEGER

RESTRICTIONS: 1) Segment must be open.

ERRORS: Errors from the routines which set the attributes.

EXAMPLE:

```
CALL CSET('NLEAF',nleaf_val)   or CALL CLEAF(nleaf-real)
CALL CSET('NLNPAT',nlnpat_val) or CALL CLNPAT(nlnpat-val)
```

```
CALL CSET('NSYMSZ',nsymsz_val) or CALL CSYMSZ(nysmsz-val)
CALL CSET('NPLTYP',npltyp_val) or CALL CPLTYP(npltyp-val)
```

CSETBP - Set the Base/Plane Vectors for an Angle of Rotation

LANGUAGE: FORTRAN 77

PURPOSE: Set the base/plane vectors for an angle of rotation.

USE: CALL CSETBP(ANGLE)

where
ANGLE - Angle of rotation (degrees).
type:REAL

NOTE: This routine will set JBASE and JPLANE, and not restore these values.

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: Segment must be open.

ERRORS: none.

CSTRG - Output Graphics Text

LANGUAGE: FORTRAN 77

PURPOSE: Output graphics text.

USE: CALL CSTRG(STR)

where: STR -
Graphics text string to be displayed.
Type:CHARACTER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: text attributes.
2) CGL attributes:
NTCOL - Text color.
Range : 0 or greater (see underlying package).
Type:INTEGER

NTPREC - Text precision.
Range : 1 - string precision text.
2 - character precision text.
3 - stroke precision text.
4 - graphics art precision text.
5 - DI-TEXTPRO precision text.
Type:INTEGER

RESTRICTIONS: Segment must be open.
DI-TEXTPRO must be part of the load process if NTPREC equals 5.

ERRORS: none.

CSTRIP - Find indices for first and last non-blank character

LANGUAGE: FORTRAN 77

PURPOSE: Find indices for the first and last non-blank character (i.e., IBEG and IEND respectively).

USE: CALL CSTRIP(STRING,IBEG,IEND)

STRING	- Character string. Type:CHARACTER
IBEG	- First non-blank character position. (returned) Type:CHARACTER
IEND	- Last non-blank character position. (returned) Type:CHARACTER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: A blank character string will result in IBEG and IEND set to one.

ERRORS: none.

EXAMPLE: given

```
CHARACTER*5 C1,C2,C3
DATA C1/'1 3 5'/,C2/' 234 '/,C3/'     '/
```

then

```
CALL CSTRIP(C1,IBEG,IEND) ==> IBEG=1, IEND=5
CALL CSTRIP(C2,IBEG,IEND) ==> IBEG=2, IEND=4
CALL CSTRIP(C3,IBEG,IEND) ==> IBEG=1, IEND=1
```

CSYMNO - Set the Type of Symbol

LANGUAGE: FORTRAN 77

PURPOSE: Set the type of symbol

USE: CALL CSYMNO(NSYMNO)

where: NSYMNO -

0	no symbol	
1	circle	
2	square (default)	
3	diamond	
4	triangle	
5	right triangle	
6	quadrant	
7	dog house	
8	fan	
9	long diamond	
10	house	
11	circle	with plus
12	square	with plus
13	diamond	with plus
14	triangle	with plus
15	right triangle	with plus
16	quadrant	with plus
17	dog house	with plus
18	fan	with plus
19	long diamond	with plus
20	house	with plus
21	dot	
22	plus sign	

If a flag is desired, add 100, 200,...,900 to NSYMNO, such that:

1	- Symbol
101	- Symbol + upper right line
201	- Symbol + upper left line
301	- Symbol + lower left line
401	- Symbol + lower right line
501	- Symbol + upper right flag
601	- Symbol + upper left flag
701	- Symbol + lower left flag
801	- Symbol + lower right flag
901	- Symbol is solid filled

range : 0 to 22,100 to 122,...,900 to 922.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

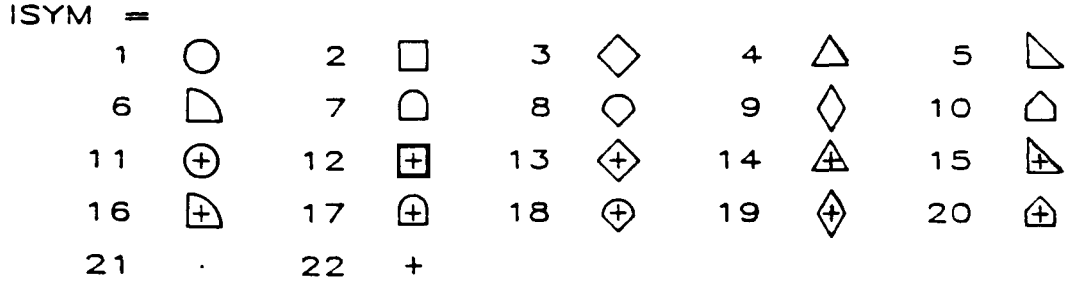
RESTRICTIONS: none.

ERRORS:

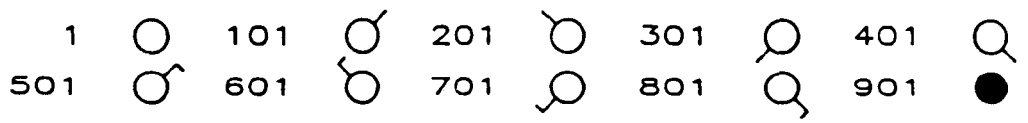
CSYMNO: IV MUST BE AN INTEGER .GE.0.
ATTEMPTED TO PASS IV= iv
THE DEFAULT VALUE IS USED.

NASA SYMBOLS

ISYM =



IF A FLAG IS WANTED, ADD 100, 200, ..., 900 TO ISYM



CSYMSZ - Set Size of Symbol

LANGUAGE: FORTRAN 77

PURPOSE: Set size of symbol

USE: CALL CSYMSZ(NSYMSZ)

where: NSYMSZ - Size of symbol
To set the symbol size in terms of current page coordinates.
range: Real values greater than zero
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTION: Segment must be open.

ERRORS:
CSYMSZ: INVALID VALUE FOR RV.
THE RANGE OF "NSYMSZ" IS .GT. ZERO (REAL).
ILLEGAL VALUE IS rv
THE DEFAULT VALUE IS USED.

CTCOL - Set Text Color for CGL Graphics Text

LANGUAGE: FORTRAN 77

PURPOSE: Set text color for all subsequent CGL graphical text.

USE: CALL CTCOL(IV)

WHERE: IV -
Text color (for all subsequent text).
range : 0 or greater (see underlying graphics package).
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CTCOL: INVALID VALUE FOR IV.
THE RANGE OF "NTCOL" IS ANY INTEGER.
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.

CTODX - Convert Degrees to a Vector Related Value

LANGUAGE: FORTRAN 77

PURPOSE: To convert degrees to a vector related value.
Conversion formula: value = TAN(XDEG in radians)
Example: JBASE(DX,1.,0.), where DX=CTODX(XDEG).

USE: VALUE=CTODX(XDEG)

where:
XDEG - Angle(in degrees) to be converted.
type:REAL

NOTE: The sign of the resultant value of CTODX is the same as the sign of XDEG.
If the TAN result is 0, then CTODX is 0.

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CTPREC - Set Text Precision for CGL Graphical Text

LANGUAGE: FORTRAN 77

PURPOSE: Set text precision for all subsequent CGL graphical text.

USE: CALL CTPREC(IV)

WHERE: IV -
Text precision.
range : 1 - String precision text.
2 - Character precision text.
3 - Stroke precision text.
4 - Graphics art precision text.
5 - DI-TEXTPRO precision text.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) DI-TEXTPRO must be loaded prior to execution if type 5 is (to be) selected.

NOTES: Call CSETBP(ANGLE) for text to be written at an angle of rotation (see CSETBP description).

ERRORS:
CTPREC: INVALID VALUE FOR IV.
THE RANGE OF "NTPREC" IS ANY INTEGER.
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.

CVAXIC - Draw a Vertical Axis with Character Labels

LANGUAGE: FORTRAN 77

PURPOSE: To draw a vertical axis with character labels.
This routine retrieves attributes (DI-3000 and CGL) when the subroutine is invoked. These axis characteristics remain in force until SUBROUTINE CVAXIC is reinvoked.
If a vertical axis label is desired, SUBROUTINE CVLAB must be invoked prior to this routine.

USE: CALL CVAXIC(XLEN,NTIC,CHARA,NLINES,NLABS)

XLEN - Length of axis (in world coordinates).
range: greater than 0.
type:REAL

NTIC - Number of tick marks (end tick marks inclusive).
If NTIC is negative, the major and minor tick marks will be suppressed, but the tick mark labels will be retained.
range: IABS(NTIC) must be greater than 1.
type:INTEGER

CHARA - Tick mark labels, dimensioned CHARA(NLINES,NLABS).
type:CHARACTER

NLINES - Number of lines in the tick mark labels.
Range: must be greater than or equal to 0.
type:INTEGER

NLABS - Number of labels for the tick marks.
Range: must be greater than or equal to 0.
type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NVBTIC - Position of vertical axis label in relation to to vertical tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
Type:INTEGER

NVJUST - Determines if vertical axis label is written horizontally or vertically.
0 - vertical (default).
1 - horizontal.
Type:INTEGER

NVLABJ - Position of axis label in relation to vertical axis. (left or right)

0 - vertical axis label is left of axis.(default)
1 - vertical axis label is right of axis.
Type:INTEGER

NVMTIC - Number of minor tick marks between major tick marks for the vertical axis. (default=0)
Range : INTEGER values greater than or equal to 0
Type:INTEGER

NVTICJ - A value to indicate positioning of tick marks for vertical axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
default: 1
Type: INTEGER

NVTROT - Angle of rotation for vertical axis tick mark labels.
Type: INTEGER

RESTRICTIONS: 1) XLEN must be positive.
2) An absolute minimum of two major tick marks.
3) Segment must be open.

ERRORS:

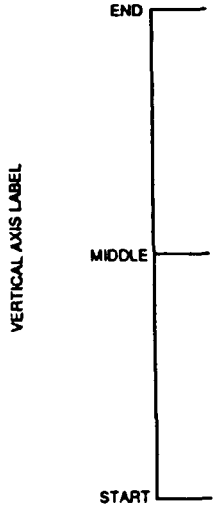
CVAXIC: NTIC MUST BE AN INTEGER .GT. 1.
ATTEMPTED TO PASS NTIC= ntic
THIS ROUTINE IS NOT EXECUTED.

CVAXIC: NLINES MUST BE AN INTEGER .GE.0.
ATTEMPTED TO PASS NLINES= nlines
THIS ROUTINE IS NOT EXECUTED.

CVAXIC: NLABS MUST BE AN INTEGER .GE. 0
ATTEMPTED TO PASS NLABS= nlabs
THIS ROUTINE IS NOT EXECUTED.

CVAXIC: INVALID VALUE FOR XLEN.
THE RANGE OF "XLEN" IS .GT. ZERO.
THE ROUTINE IS NOT EXECUTED.

CVAXIC: TICK MARK ROTATION (NVTROT) ONLY APPLY TO ONE TICK MARK LINE (NLINES).
NO ROTATION PERFORMED.



CVAXIS - Draw a Vertical Axis with Numeric Labels

LANGUAGE: FORTRAN 77

PURPOSE: This routine plots a vertical axis. The numeric values are calculated, converted to labels, then CVAXIC routine is called to do the actual plotting. The attributes for CVAXIC are the same and actually do not affect this routine, but will affect the actual plot.

USE: CALL CVAXIS(VTMIN,VTMAX,TINCR,VLONG)

where:
- VTMIN - Minimum value used for tick marks (Type:REAL)
- VTMAX - Maximum value used for tick marks (Type:REAL)
- TINCR - Increment value of tick marks (Type:REAL)
- VLONG - Length of axis in current world coordinates (Type:REAL)

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NVBTIC - Position of vertical axis label in relation to to vertical tick marks.
0 - labels centered at tick marks. (default)
1 - labels centered between tick marks.
Type:INTEGER

NVJUST - Determines if vertical axis label is written horizontally or vertically.
0 - vertical (default).
1 - horizontal.
Type:INTEGER

NVLABJ - Position of axis label in relation to vertical axis. (left or right)
0 - vertical axis label is left of axis.(default)
1 - vertical axis label is right of axis.
Type:INTEGER

NVMTIC - Number of minor tick marks between major tick marks for the vertical axis. (default=0)
Range : INTEGER values greater than or equal to 0
Type:INTEGER

NVPREC - Number of digits to the right of the decimal point in the tick mark labels of the vertical axis.
< 0 - display as a REAL. value represents the

number of digits to the right of the decimal point.
= 0 - decimal point only
> 0 - display as INTEGER (no decimal point)
default: -1
Type:INTEGER

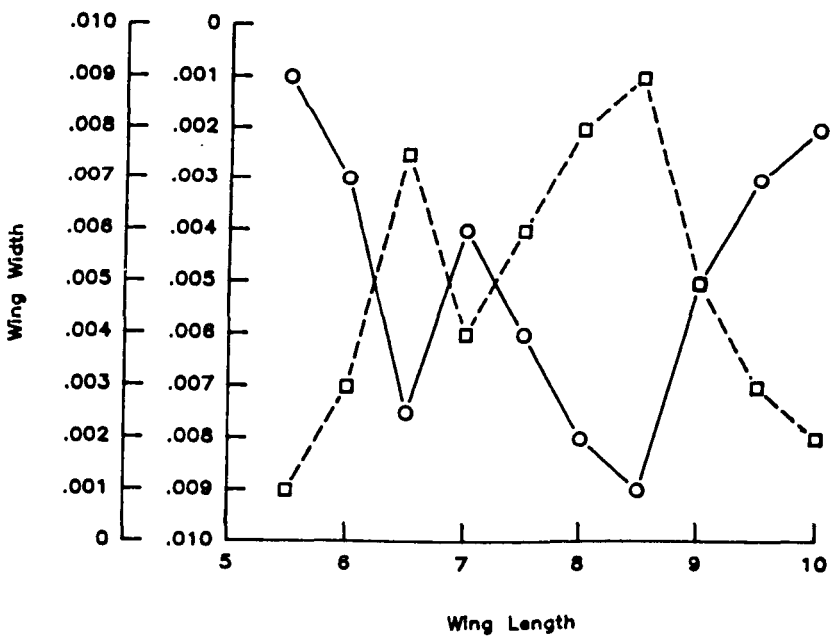
NVTICJ - A value to indicate positioning of tick marks for vertical axis.
0 - tick marks are not positioned about the axis.
1 - tick marks positioned right of the axis.
2 - tick marks positioned to the left of axis.
3 - tick marks positioned on both sides of axis.
default: 1
Type: INTEGER

NVTROT - Angle of rotation for vertical axis tick mark labels.
Type: INTEGER

RESTRICTIONS: 1) Tick mark labels always occur at the major tick marks (regardless of attribute settings).
2) If VTMIN is greater than VTMAX, then TINCR must be negative. VTMIN and VTMAX cannot be equal.
3) Segment must be open.

ERRORS: none.

NOTES: If VTMIN is greater than VTMAX, then axis labels will be output in decreasing order.



A Title

CVBTIC - Set Position of Vertical Axis Label Tick Marks

LANGUAGE: FORTRAN 77

PURPOSE: Set position of vertical axis label tick marks.

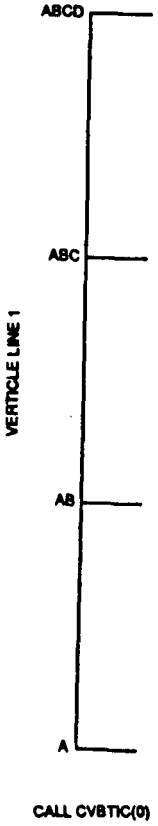
USE: CALL CVBTIC(IV)

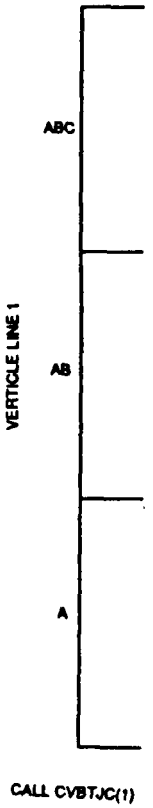
WHERE: IV -
Position of vertical axis label in relation to vertical tick marks.
0 - Labels centered at tick marks. (default)
1 - Labels centered between tick marks.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CVBTIC: INVALID VALUE FOR IV.
THE RANGE OF "NVBTIC" IS 0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.





CVJUST - Set Position of Vertical Axis Label (Horizontal/Vertical)

LANGUAGE: FORTRAN 77

PURPOSE: Set position of vertical axis label (horizontal/vertical)

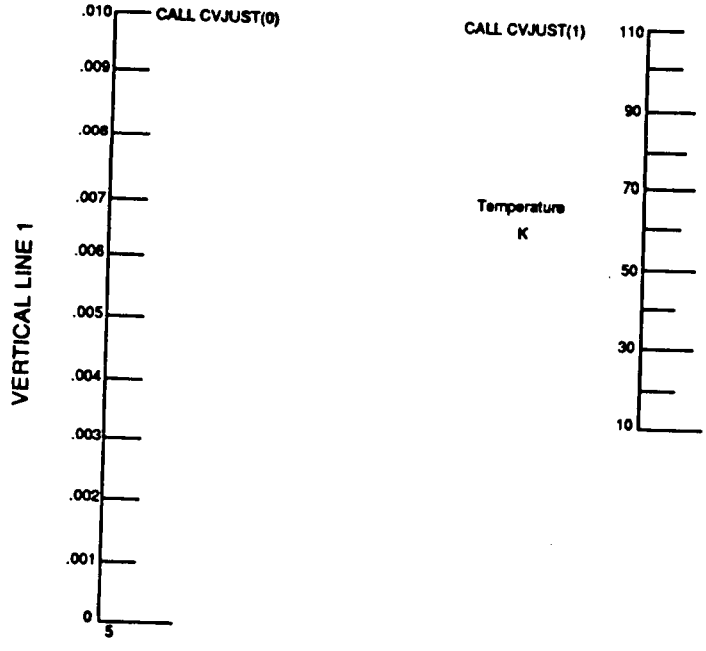
USE: CALL CVJUST(IV)

where: IV -
Determines if vertical axis label is written horizontally or vertically.
0 - vertical (default)
1 - horizontal.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CVJUST: INVALID VALUE FOR IV.
THE RANGE OF "NVJUST" IS 0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



CVLAB - Describe the Text and Attributes of Vertical Axis Label

---

LANGUAGE: FORTRAN 77

PURPOSE: To describe the text and associated attributes of the vertical axis label.
This routine retrieves attributes (DI-3000 and CGL) when the subroutine is invoked. These label characteristics remain in force until SUBROUTINE CVAXIC is reinvoked.

USE: CALL CVLAB(CHARA,NLINE)

CHARA(NLINE) - Array of characters for axis.
type:CHARACTER
NLINE - Number of lines in label.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:
NVJUST - Determines if vertical axis label is written horizontally or vertically.
0 - vertical (default).
1 - horizontal.
Type:INTEGER

RESTRICTIONS:
1) Maximum number of lines per axis label is limited to 5.
2) The number of characters per line is limited to 256.
3) All lines in axis label are single spaced.
4) Segment must be open.

ERRORS:
CVLAB: NLINE MUST BE AN INTEGER .GE. 0.
ATTEMPTED TO PASS NLINE= nline
THIS ROUTINE IS NOT EXECUTED.

Note: If NLINE is less than or equal to 0, then all vertical axis label attributes are reinitialized.

CVLABJ - Set Position of Axis Label Relative to Vertical Axis

---

LANGUAGE: FORTRAN 77

PURPOSE: Set position of axis label relative to vertical axis.

USE: CALL CVLABJ(IV)

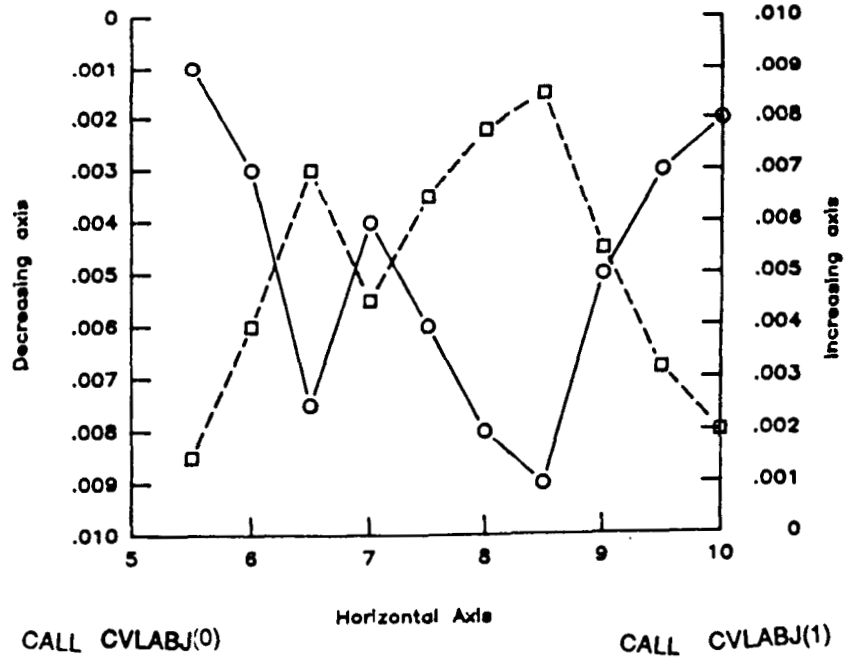
where: IV -
Position of axis label in relation to vertical axis. (left or right)
0 - Vertical axis label is left of axis. (default)
1 - Vertical axis label is right of axis.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

NOTES: Attribute CVTICJ controls the vertical axis tick mark positioning (i.e., above, below,etc.). The attribute CVLABJ controls the positioning of the tick mark labels.

ERRORS:
CVLABJ: INVALID VALUE FOR IV.
THE RANGE OF "NVLABJ" IS 0,1 (INTEGER).
ILLEGAL VALUE IS iv
THE DEFAULT VALUE IS USED.



CVLOG - Draw a Vertical Log Axis

LANGUAGE: FORTRAN 77

PURPOSE: To draw a vertical log axis.
This routine retrieves attributes (DI-3000 and CGL) when the subroutine is invoked. These characteristics remain in force until subroutine CVLOG is reinvoked.
If a vertical axis label is desired, subroutine CVLAB must be invoked prior to this routine.

USE: CALL CVLOG(XLEN,NTIC,NVLOGI)

XLEN - Length of axis (in world coordinates).
range: greater than 0.
type:REAL

NTIC - Number of tick marks (end tick marks inclusive; i.e., number of cycles-1 on XLEN).
If NTIC is negative, the major tick marks will be suppressed, but the numeric tick mark labels will be retained.
range: IABS(NTIC) must be greater than 1.
type:INTEGER

NVLOGI - Tick mark label type.
The following are the various types :
0 - No small tick marks.
1 - 1,2,3,4,5,6,7,8,9,1
2 - 1,2,4,6,8,1
3 - 1,3,5,7,9,1
4 - 1,5,1
5 - 1,1
6 - 2,3,4,5,6,7,8,9
7 - 2,4,6,8
8 - 3,5,7,9
9 - 5

If NVLOGI is positive then labels and small log tick marks are drawn.
if NVLOGI is negative then only small log tick marks are drawn.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: line attributes, text attributes.
2) CGL attributes:

NLABP - The size of the minor tick marks as a percentage of the size of the major tick marks. (Default=2/3)
Range : REAL values greater than 0.
Type:REAL

NVLABJ - Position of axis label in relation to vertical axis. (left or right)

0 - vertical axis label is left of axis.(default)
1 - vertical axis label is right of axis.
Type:INTEGER

NVLOGF - Determine if vertical log axis will have major marks as 10 raised to a power.
-1 - no major tick mark labels
0 - major tick mark labels (default)
1 - major tick mark labels increase
Type:INTEGER

NVLOGS - The initial power for the vertical log axis labels.
Range: any valid INTEGER(10 raised to this power).
(default 0)
Type:INTEGER

RESTRICTIONS: 1) XLEN must be positive.
2) An absolute minimum of two major tick marks.
3) Segment must be open.

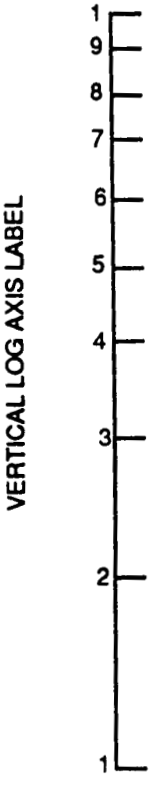
ERRORS:

CVLOG: XLEN MUST BE .GT. 0.
ATTEMPTED TO PASS XLEN= xlen.
THIS ROUTINE IS NOT EXECUTED.

CVLOG: NTIC MUST BE AN INTEGER .GT. 1.
ATTEMPTED TO PASS NTIC= ntic.
THIS ROUTINE IS NOT EXECUTED.

CVLOG: INVALID VALUE FOR XLEN.
THE RANGE OF "XLEN" IS .GT. ZERO.
THE ROUTINE IS NOT EXECUTED.

CVLOG: INVALID VALUE FOR NVLOGI.
THE RANGE OF "NVLOGI" IS -9 TO 9.
THE ROUTINE IS NOT EXECUTED.



VERTICAL LOG AXIS

1) TO SET AXIS LABEL

CALL CVLABS(CHARA, NLINE)

CHARA ARRAY OF CHARACTERS FOR THE AXIS

NLINE NUMBER OF LINES IN THE AXIS LABEL

2) TO DRAW AXIS AND LABEL

CALL CVLOG(XLEN, NTIC, NLOGI)

XLEN LENGTH OF THE AXIS

NTIC NUMBER OF MAJOR TIC MARKS

NLOGI TIC MARK LABEL TYPE

0 - NO MINOR TIC MARKS

1 - 1, 2, 3, 4, 5, 6, 7, 8, 9, 1

2 - 1, 2, 4, 6, 8, 1

3 - 1, 3, 5, 7, 9, 1

4 - 1, 5, 1

5 - 1, 1

6 - 2, 3, 4, 5, 6, 7, 8, 9

7 - 2, 4, 6, 8

8 - 3, 5, 7, 9

9 - 5

CVLOGF - Set Vertical Log Axis Raised to a Power of 10

LANGUAGE: FORTRAN 77

PURPOSE: Set vertical log axis raised to a power of 10

USE: CALL CVLOGF(IV)

where: IV -
Determine if vertical log axis will have major marks as 10 raised to a power.
- -1 - Major tick mark labels decrease
- 0 - No major tick mark labels (default)
- 1 - Major tick mark labels increase

type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

CVLOGF: INVALID VALUE FOR IV.
THE RANGE OF "NVLOGF" IS -1,0,1 (INTEGER).
ILLEGAL VALUE IS iv.
THE DEFAULT VALUE IS USED.

CVLOGS - Set the Initial Power of the Vertical Log Axis Label

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the initial power of the vertical LOG axis label

USE: CALL CVLOGS(IV)

where: IV -
The initial power for the vertical log axis labels (i.e., 10 raised to this power).
range: Any valid integer
(default 0)
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:
CVLOGS: INVALID VALUE FOR IV.
THE RANGE OF "NVLOGS" IS ANY INTEGER.
ILLEGAL VALUE IS iv.
THE DEFAULT VALUE IS USED.

CVMTIC - Set the Number of Vertical Minor Tick Marks between Major

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the number of vertical minor tick marks between major tick marks.

USE: CALL CVMTIC(IV)

where: IV -
Number of minor tick marks between major tick marks for the vertical axis. (default = 0)
range: Integer values greater or equal to 0.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

```
CVMTIC:  INVALID VALUE FOR IV.
         THE RANGE OF "NVMTIC" IS .GE. ZERO (INTEGER).
         ILLEGAL VALUE IS iv.
         THE DEFAULT VALUE IS USED.
```

CVPORT - Define a Viewport into the Virtual Coordinates by DI-3000

LANGUAGE: FORTRAN 77

PURPOSE: This subroutine uses "relative" inches to define virtual coordinates used by DI-3000 based upon the values contained in the COMMON BLOCK CFRSIZ. Routine CVSPAC defines the maximum relative inches (minimum is always zero). This is used to map the virtual coordinates to device coordinates.

USE: CALL CVPORT(XLCOR,YLCOR,XUCOR,YUCOR)

XLCOR - X value of the lower boundary of the viewport (inches). (default = 0 inches)
range: 0 or greater.
type:REAL

XUCOR - X value of the right boundary of the viewport (inches). (default = height)
range: cannot exceed width in CVSPAC.
type:REAL

YLCOR - Y value of the left boundary of the viewport (inches). (default = 0 inches)
range: 0 or greater.
type:REAL

YUCOR - Y value of the upper boundary of the viewport (inches). (default = width)
range: cannot exceed height in CVSPAC.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) Subroutine CVSPAC must be called prior to CVPORT.
2) Segment must NOT be open.

ERRORS:

CVPORT: XLCOR MUST BE .GE. 0.
ATTEMPTED TO PASS XLCOR= xlcor.
THIS ROUTINE IS NOT EXECUTED.

CVPORT: YLCOR MUST BE .GE. 0.
ATTEMPTED TO PASS YLCOR= ylcor.
THIS ROUTINE IS NOT EXECUTED.

CVPREC - Set Vertical Numeric Axis Tick Mark Label Precision

LANGUAGE: FORTRAN 77

PURPOSE: Set vertical numeric axis tick mark label precision.

USE: CALL CVPREC(IV)

Where: IV - Number of digits to the right of the decimal point in the tick mark labels of the vertical axis.
< 0 - Plot as an integer
= 0 - Decimal point only
> 0 - Plot as a real. IV represents the number of digits to the right of the decimal point.
default: -1
type: INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS:

```
CVPREC:  INVALID VALUE FOR IV.
         ATTEMPTED TO PASS IV= iv.
         THE DEFAULT VALUE IS USED.
```

CVSPAC - Establish DI-3000 Boundaries for an Entire Frame

LANGUAGE: FORTRAN 77

PURPOSE: Establishes DI-3000 boundaries for an entire frame which correspond to the values used on the plot card. This routine should be called for plotting regions that are rectangular rather than square. If this routine is not invoked then the default frame will be an 11" by 11" square. This routine can only be called once, and must be called prior to opening the first segment.

This routine will establish a coordinate system based on the values in the X-direction of 0 to width, and values in the Y-direction of 0 to height. These coordinates are referred to as "inches" or "relative inches".

USE: CALL CVSPAC(WIDTH,HEIGHT)

WIDTH - Width of the frame. Should be same as or proportional to the value used on the plot card used to call the device driver for the static plotters. (default = 11.)
range: greater than 0.
type:REAL

HEIGHT - Height of the frame. Should be same as or proportional to the value used on the plot card used to call the device driver for the static plotters.(default = 11.)
range: greater than 0.
type: REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) WIDTH and HEIGHT must be greater than 0.
2) Must be called prior to opening the first segment.

ERRORS:

CVSPAC: HEIGHT MUST BE .GE. 0.
ATTEMPTED TO PASS HEIGHT= height.
THIS ROUTINE IS NOT EXECUTED.

CVSPAC: WIDTH MUST BE .GE. 0.
ATTEMPTED TO PASS WIDTH= width.
THIS ROUTINE IS NOT EXECUTED.

CVTICJ - Set the Tick Mark Position Relative to the Vertical Axis

---

LANGUAGE: FORTRAN 77

PURPOSE: Set the tick mark position relative to the vertical axis.

USE: CALL CVTICJ(IV)

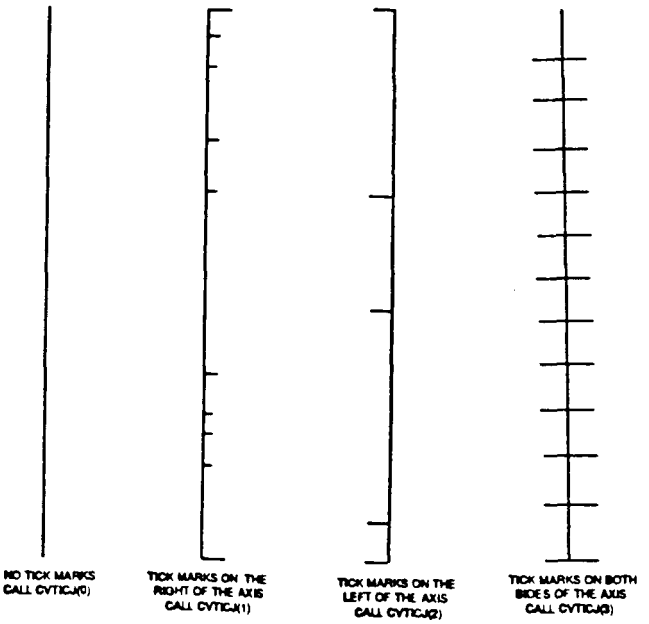
WHERE: IV -
is a value to indicate positioning of tick marks.
0 - No tick marks on the axis.
1 - Tick marks positioned right of the axis. (default)
2 - Tick marks positioned to the left of axis.
3 - Tick marks positioned on both sides of axis.
Type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

NOTES: Attribute CVTICJ controls the vertical axis tick mark positioning (i.e., above, below,etc.). The attribute CVLABJ controls the positioning of the tick mark labels.

ERRORS:
CVTICJ: IV MUST BE A 0, 1, 2, 3.
ATTEMPTED TO PASS IV= iv.
THE DEFAULT VALUE IS USED.



CVTROT - Set Angle of Rotation for Vertical Axis Tick Mark Labels

---

LANGUAGE: FORTRAN 77

PURPOSE: Set angle of rotation for vertical axis tick mark labels.

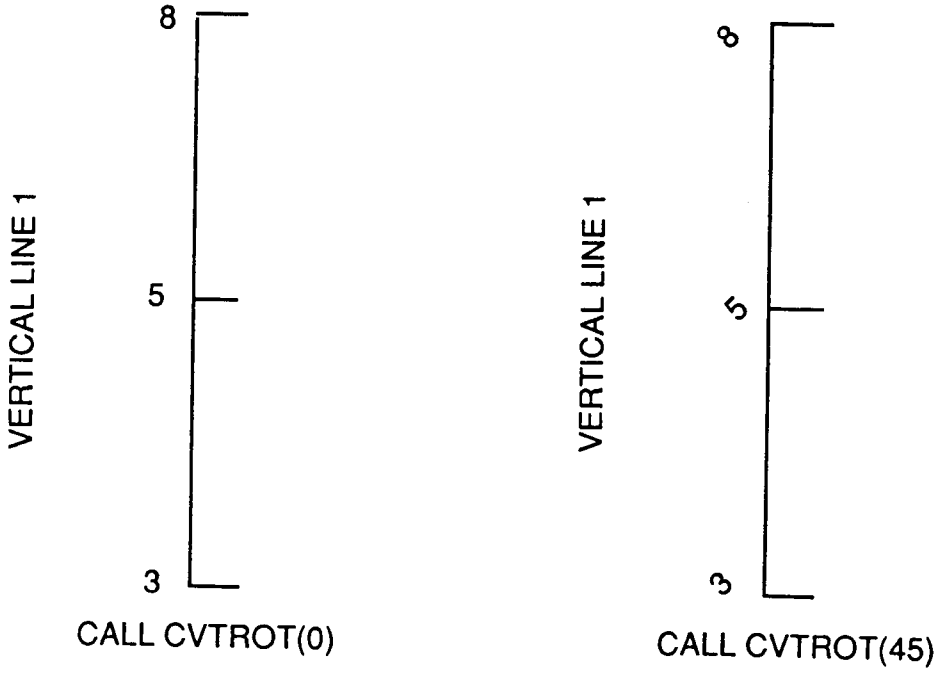
USE: CALL CVTROT(IV)

WHERE: IV - Rotation angle.
type: INTEGER.

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: 1) Angles are entered in as integer values.

ERRORS:
CVTROT: INVALID VALUE FOR IV.
ATTEMPTED TO PASS IV= iv.
THE DEFAULT VALUE IS USED.



CXCALC - Return Equivalent Delta X Value for a Given Delta Y

LANGUAGE: FORTRAN 77

PURPOSE: Return the equivalent delta X value for a given delta Y (in world coordinates).

USE: x_value = CXCALC(YVAL)

where : YVAL - Y distance in world coordinates.
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

CYCALC - Return Equivalent Delta Y Value for a Given Delta X

LANGUAGE: FORTRAN 77

PURPOSE: Return the equivalent delta Y value for a given delta X (in world coordinates).

USE: y_value = CYCALC(XVAL)

where : XVAL - X distance in world coordinates
type:REAL

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS: none.

ERRORS: none.

C1NMBR - Draw a Numeric Value Using Text Precision

LANGUAGE: FORTRAN 77

PURPOSE: This routine draws a numeric value using current text attributes. Low quality text precision is used.

USE: CALL C1NMBR(VALUE,NDEC)

where: VALUE - Value to be plotted real or integer.
type:REAL

NDEC - Integer specifying number of digits to the right of the decimal point.

NDEC = 0 - Decimal point, number digits to the right.

NDEC < 0 - No decimal point, min number of digits in integer value.

type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS:

1) The number is restricted to a maximum of 12 significant digits.
2) Segment must be open.

ERRORS:

C1NMBR: NDEC MUST BE AN INTEGER.
ATTEMPTED TO PASS NDEC= ndec
THIS ROUTINE IS NOT EXECUTED.

C2NMBR - Draw a Numeric Value Using Character Precision

LANGUAGE: FORTRAN 77

PURPOSE: This routine draws a numeric value using current text attributes. Character precision is used.

USE: CALL C2NMBR(VALUE,NDEC)

where: VALUE - Value to be plotted real or integer.
type:REAL
NDEC - Integer specifying number of digits to the right of the decimal point.

NDEC = 0 - Decimal point, number digits to the right.
NDEC < 0 - No decimal point, min number of digits in integer value.
type:INTEGER

ASSOCIATED ATTRIBUTES:
1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS:
1) The number is restricted to a maximum of 12 significant digits.
2) Segment must be open.

ERRORS:
C2NMBR: NDEC MUST BE AN INTEGER.
ATTEMPTED TO PASS NDEC= ndec
THIS ROUTINE IS NOT EXECUTED.

C3NMBR - Draw a Numeric Value Using Stroke Precision

LANGUAGE: FORTRAN 77

PURPOSE: This routine draws a numeric value using current text attributes. Stroke precision is used.

USE: CALL C3NMBR(VALUE,NDEC)

where: VALUE - Value to be plotted real or integer.
type:REAL

NDEC - Integer specifying number of digits to the right of the decimal point.

NDEC = 0 - Decimal point, number digits to the right.

NDEC < 0 - No decimal point, min number of digits in integer value.

type:INTEGER

ASSOCIATED ATTRIBUTES:

1) DI-3000 attributes: none.
2) CGL attributes: none.

RESTRICTIONS:

1) The number is restricted to a maximum of 12 significant digits.
2) Segment must be open.

ERRORS:

C3NMBR: NDEC MUST BE AN INTEGER.
ATTEMPTED TO PASS NDEC= NDEC.
THIS ROUTINE IS NOT EXECUTED.

APPENDIX B

CHARTS AND TEST CASES OF THE LOW-LEVEL ROUTINES

In this appendix, the output generated by the sample programs will appear before the programs. All output will be labeled with the program's title and a short description of each figure.

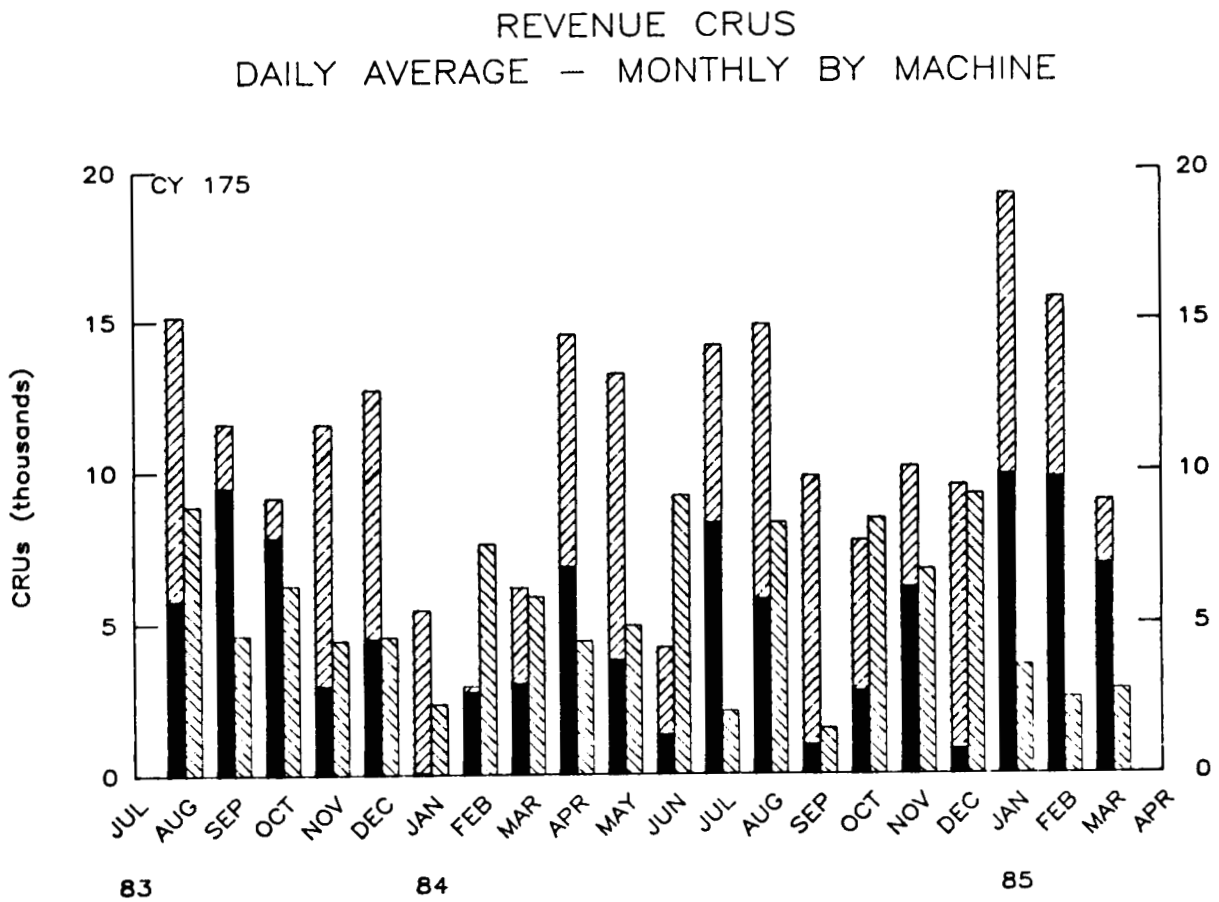


Figure CDB1. Additive and absolute bars - multiple data sets.

```
      PROGRAM CDB1
C-----------------------------------------------------------------------
C ADDITIVE AND ABSOLUTE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      INTEGER ID(NUMSET)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      CHARACTER MONTHS(0:37)*3
      DATA ID/0,1,0/
      DATA MONTHS/'   ','JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC','   '/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
```

```
C-----------------------------------------------------------------------
C VERTICAL AXIS - ON LEFT OF PLOT
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      ICT=0
      DO 101 I=2,NUMSET
101     IF (ID(I).EQ.1) ICT=ICT+1
      VMAX=10.*REAL(NUMSET-ICT)
      CALL CVAXIS(0.,VMAX,5.,YLEN)
C VERTICAL AXIS - ON RIGHT OF PLOT
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVAXIS(0.,VMAX,5.,YLEN)
C-----------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-----------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,VMAX)
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                    2./3.
      XBARW=XWIDTH/REAL(NUMSET-ICT)
      DO 99 I=1,NUMPTS
99      TOTAL(I)=0.0
      ICT=1
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        XOFF=-XWIDTH/2. + REAL(ICT-1)*XBARW
        IF(ID(I).EQ.1)ICT=ICT+1
```

```
         DO 2 J=1,NUMPTS
           CALL JRECT(REAL(J)+XOFF,TOTAL(J),REAL(J)+XOFF+XBARW,
     +      TOTAL(J)+REVCRU(I,J))
         IF (ID(I+1).EQ.1) THEN
           TOTAL(J)=TOTAL(J)+REVCRU(I,J)
         ELSE
           TOTAL(J)=0.
         END IF
2      CONTINUE
C-------------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
       CALL JCLOSE
       CALL JFRAME
       CALL JDEVOF(IDEV)
       CALL JDEND(IDEV)
       CALL JEND
       STOP
       END
```

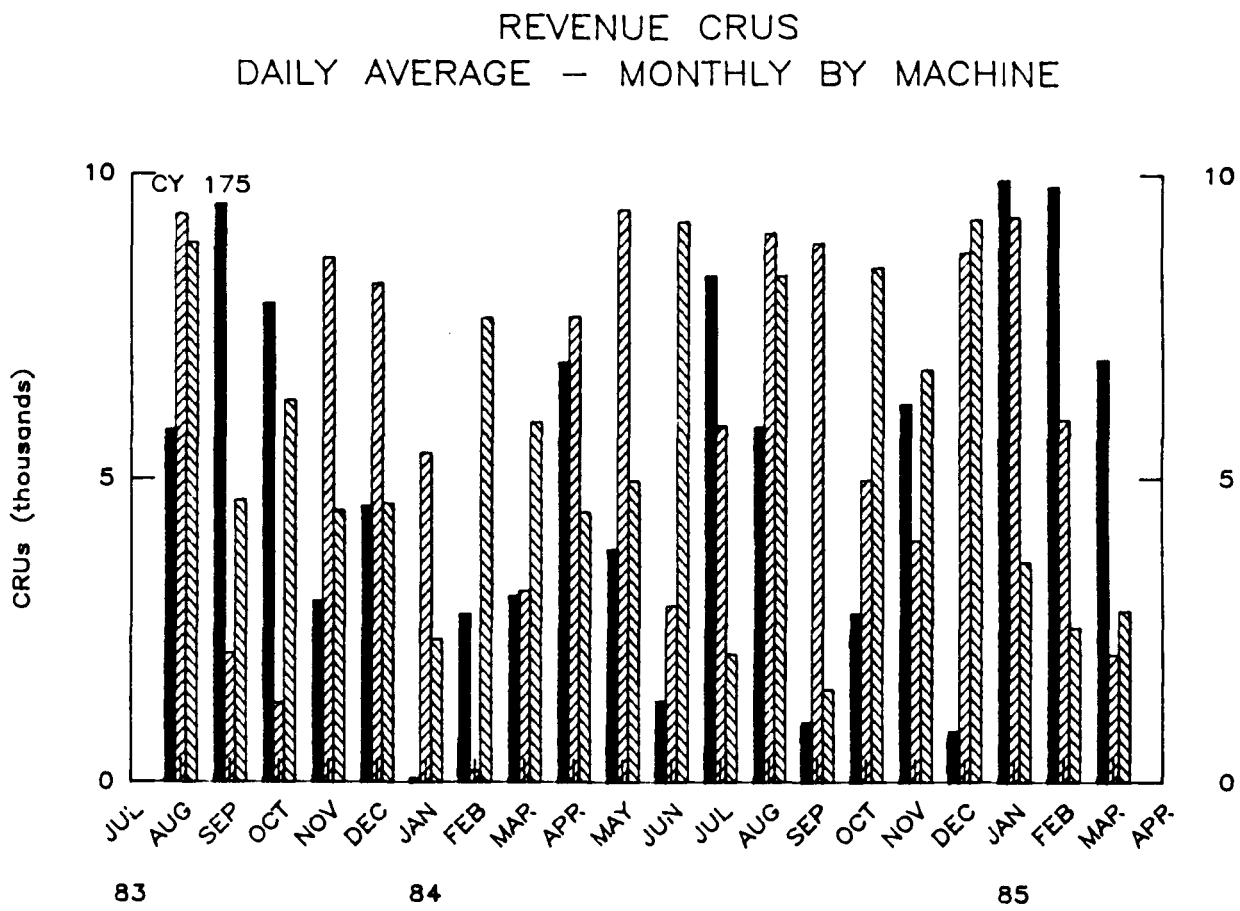


Figure CDB2. Absolute bars - multiple data sets.

```
      PROGRAM CDB2
C-----------------------------------------------------------------------
C ABSOLUTE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS)
      CHARACTER MONTHS(0:37)*3,VALUES(3)*2
      DATA MONTHS/'   ','JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC','   '/
      DATA VALUES/' 0',' 5','10'/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-----------------------------------------------------------------------
```

```
C VERTICAL AXIS - ON LEFT
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
C VERTICAL AXIS - ON RIGHT
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVTICJ(2)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
C-----------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-----------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,10.)
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                  2./3.
      XBARW=XWIDTH/REAL(NUMSET)
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        XOFF=-XWIDTH/2. + REAL(I-1)*XBARW
        DO 2 J=1,NUMPTS
 2    CALL JRECT(REAL(J)+XOFF,0.,REAL(J)+XOFF+XBARW,REVCRU(I,J))
C-----------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
```

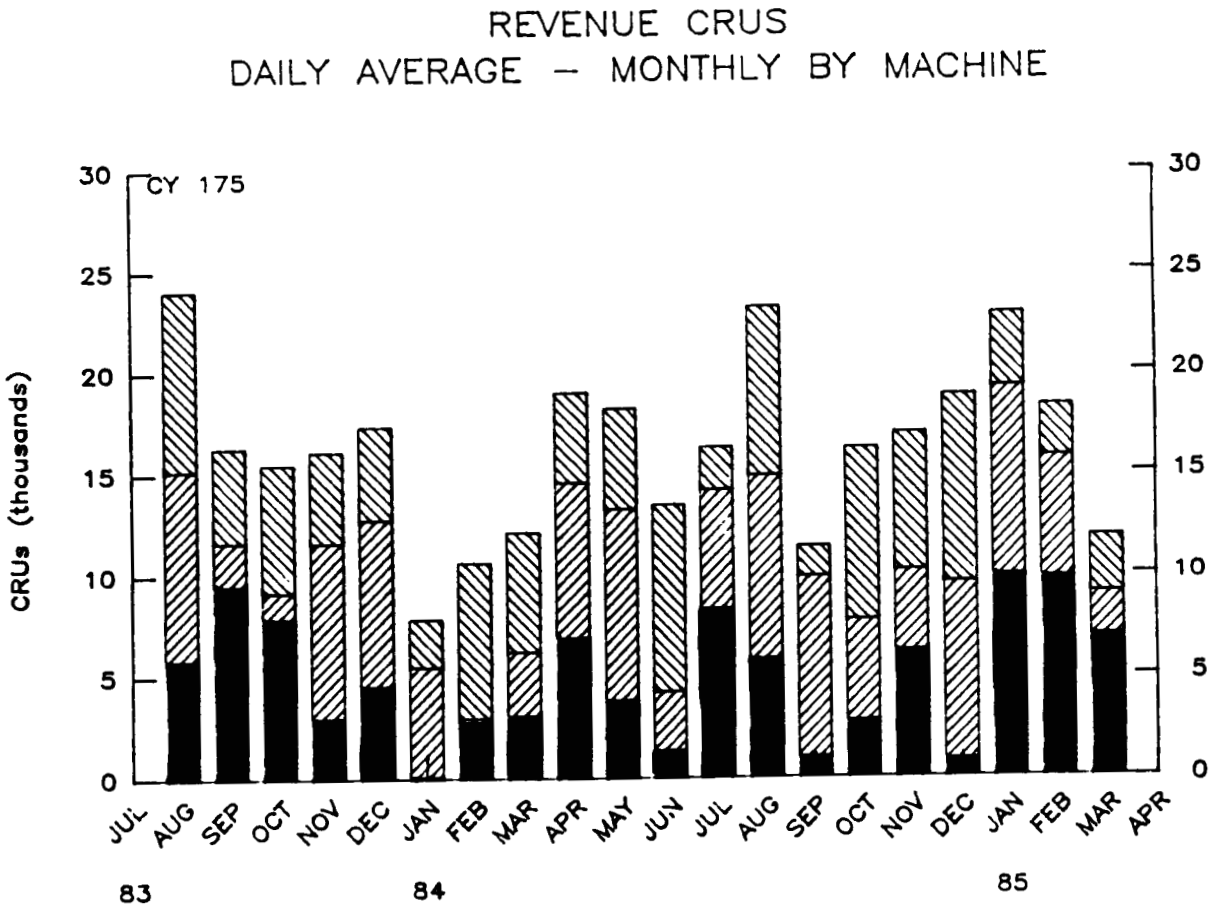


Figure CDB3. Additive bars - multiple data sets.

```
      PROGRAM CDB3
C-----------------------------------------------------------------------
C ADDITIVE BARS - MULTIPLE DATA SETS
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0,CSIZE=.15)
      PARAMETER(XBEG=1.5,YBEG=1.5,XLEN=8.5,YLEN=5.0)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      CHARACTER MONTHS(0:37)*3
      DATA MONTHS/'    ','JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC',
     +                  'JAN','FEB','MAR','APR','MAY','JUN',
     +                  'JUL','AUG','SEP','OCT','NOV','DEC','    '/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*10.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL CVSPAC(XWMAX,YWMAX)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHTROT(45)
      IMONTH1=8
      CALL CHAXIC(XLEN,NUMPTS+2,MONTHS(IMONTH1-1),1,NUMPTS+2)
C YEARS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-----------------------------------------------------------------------
C VERTICAL AXIS - LABEL ON LEFT OF AXIS
```

```
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIS(0.,30.,5.,YLEN)
C VERTICAL AXIS - LABEL ON RIGHT OF AXIS
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVAXIS(0.,30.,5.,YLEN)
C-----------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-----------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,XV1,YV1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,XV2,YV2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(XV1,XV2,YV1,YV2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,30.)
C-----------------------------------------------------------------------
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL JPINTR(1)
C THE WIDTH OF BARS AT A SINGLE TICK MARK IS DETERMINED
C BY THE LENGTH OF THE AXIS (IN DATA COORDINATES) DIVIDED
C BY THE NUMBER OF INTERVALS TIMES A FRACTION (2/3).
C     XWIDTH=((NUMPTS+1)/REAL(NUMPTS+2-1))*(2./3.)
C NOTE IF THE WORLD IS FROM 0 TO A VALUE, THEN XWIDTH=2/3.
      XWIDTH=                                  2./3.
      XHALF=XWIDTH/2.
      DO 99 I=1,NUMPTS
 99   TOTAL(I)=0.0
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,48-I)
        DO 2 J=1,NUMPTS
          CALL JRECT(REAL(J)-XHALF,TOTAL(J),
     +               REAL(J)+XHALF,TOTAL(J)+REVCRU(I,J))
 2    TOTAL(J)=TOTAL(J)+REVCRU(I,J)
C-----------------------------------------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JCLOSE
      CALL JFRAME
```

```
CALL JDEVOF(IDEV)
CALL JDEND(IDEV)
CALL JEND
STOP
END
```

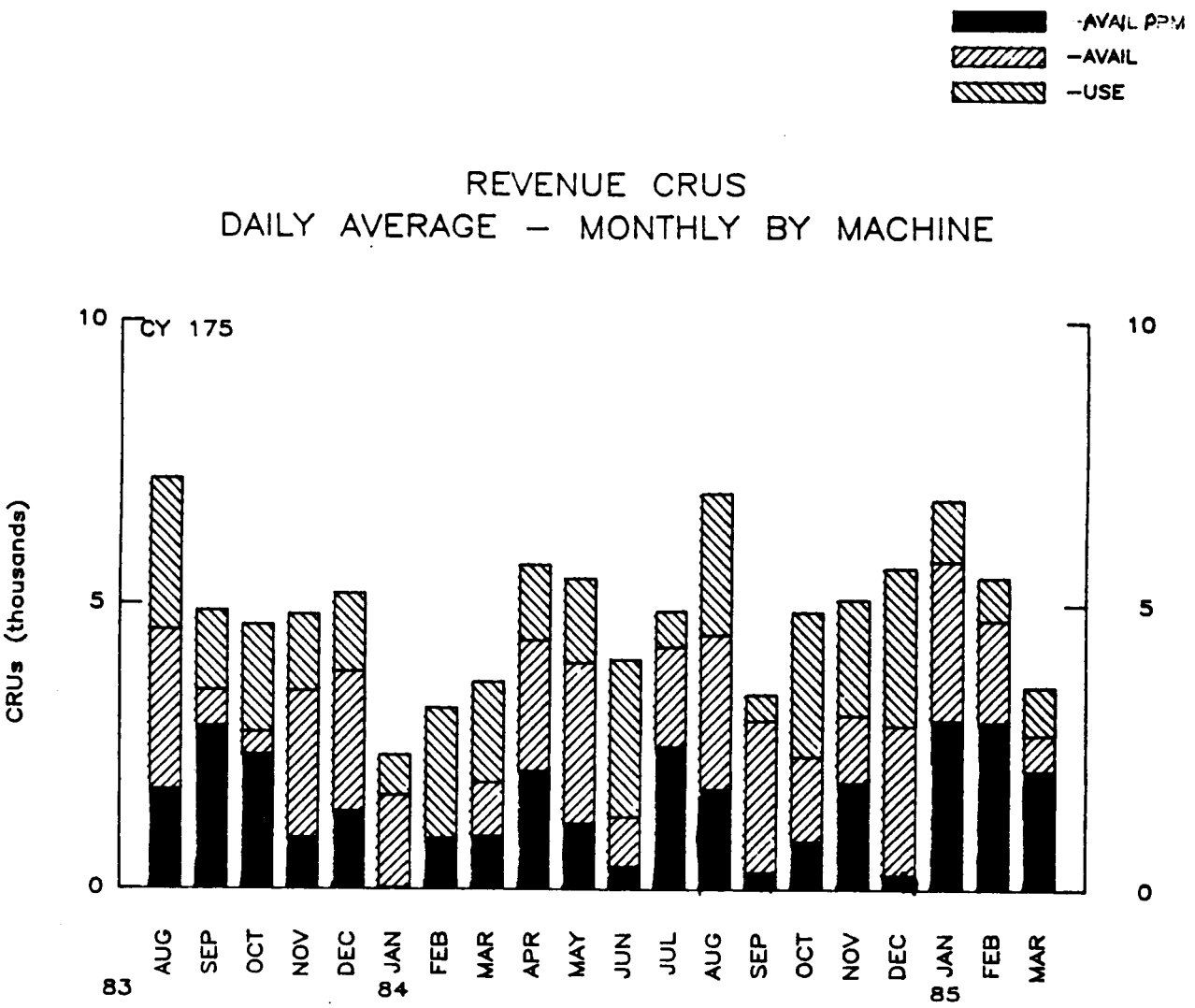


Figure CDB4. Additive bars - multiple data sets, with a legend.

```
      PROGRAM CDB4
C-----------------------------------------------------------------------
C ADDITIVE BARS - MULTIPLE DATA SETS, WITH A LEGEND.
C-----------------------------------------------------------------------
C SET UP PLOT CONSTANTS
      PARAMETER(XWMAX=11.0,YWMAX=11.0)
      PARAMETER(XBEG=1.5,YBEG=1.5)
      PARAMETER(XLEN=8.5,YLEN=5.0)
      PARAMETER(CSIZE=.15)
C-----------------------------------------------------------------------
C ALLOCATE STORAGE AND INITIALIZE VARIABLES
      PARAMETER(NUMSET=3,NUMPTS=20)
      REAL REVCRU(NUMSET,NUMPTS),TOTAL(NUMPTS)
      PARAMETER(NLINES=4,NCOLS=1,NTLINES=1)
      CHARACTER KEYCHR(NCOLS,NLINES)*9,KEYLAB(NLINES)*9
      INTEGER ISTORE(15,NLINES),JCOL(1)
      REAL BPOS(4)
      CHARACTER MONTHS(12)*3,VALUES(3)*2
      DATA MONTHS/'JAN','FEB' 'MAR','APR','MAY','JUN',
     +            'JUL','AUG','SEP','OCT','NOV','DEC'/
      DATA VALUES/' 0',' 5','10'/
      DATA KEYLAB(1)/'-AVAIL+PM'/,KEYLAB(2)/'-AVAIL    '/
      DATA KEYLAB(3)/'-USE      '/,KEYLAB(4)/'-PROD     '/
      DATA JCOL/1/
      DO 9 I=1,NUMSET
      DO 9 J=1,NUMPTS
 9    REVCRU(I,J)=RANF()*3.
C-----------------------------------------------------------------------
C INITIALIZE DI3000 AND CGL
      CALL JBEGIN
      CALL CBEGIN
      CALL CDEBUG(1)
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR PAGE AREA
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,XWMAX,0.,YWMAX)
      CALL JOPEN
C-----------------------------------------------------------------------
      CALL JSIZE(CSIZE,CSIZE*1.25)
C HORIZONTAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CHLABJ(1)
      CALL CHAXIC(XLEN,2,' ',0,NUMPTS)
C MONTHS
      XINCR=XLEN/REAL(NUMPTS+2-1)
      CALL JMOVE(XBEG,YBEG-2.*CSIZE*1.25)
      CALL JJUST(3,2)
      CALL CSETBP(90.)
      IMONTH1=8
      DO 1 KI=1,NUMPTS
        IMONTH=KI+IMONTH1-1
```

```
        IMONTH=MOD(IMONTH,12)
        IF(IMONTH.EQ.0)IMONTH=12
        CALL JRMOVE(XINCR,0.)
 1    CALL JHSTRG(MONTHS(IMONTH))
      CALL CSETBP(0.)
      CALL JSETDB(0)
C YEARS
      CALL JMOVE(XBEG,YBEG-((CSIZE*1.25)+(3.*CSIZE)+(CSIZE*1.25)))
      CALL JJUST(2,3)
      CALL CSETBP(0.)
      CALL JHSTRG('83')
      IF(IMONTH1.NE.1)THEN
        DIST=REAL((12-IMONTH1)+2)
        CALL JRMOVE(DIST*XINCR,0.)
        CALL JHSTRG('84')
      ENDIF
      CALL JRMOVE(12.*XINCR,0.)
      CALL JHSTRG('85')
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XBEG,YBEG)
      CALL CVLAB('CRU[BLC]S ([BLC]THOUSANDS)',1)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
      CALL JMOVE(XBEG+XLEN,YBEG)
      CALL CVLAB(' ',0)
      CALL CVJUST(0)
      CALL CVLABJ(1)
      CALL CVTICJ(2)
      CALL CVAXIC(YLEN,3,VALUES,1,3)
C-----------------------------------------------------------------------
C DRAW MISC. TEXT OVER PLOT
      YOFF=(2.*CSIZE+5.*CSIZE)
      CALL JMOVE(XBEG+XLEN/2.,YBEG+YLEN+YOFF)
      CALL JJUST(2,1)
      CALL JSIZE(1.5*CSIZE,1.5*CSIZE*1.25)
      CALL JHSTRG('REVENUE CRUS')
      CALL JRMOVE(0.,-2.*CSIZE*1.25)
      CALL JHSTRG('DAILY AVERAGE - MONTHLY BY MACHINE')
      CALL JMOVE(XBEG+CSIZE,YBEG+YLEN)
      CALL JJUST(1,3)
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL JHSTRG('CY 175')
C-----------------------------------------------------------------------
C SAVE OFF VIRTUAL COORDINATES OF DATA AREA (FROM AXES)
      CALL JCONWV(XBEG,YBEG,0.,X1,Y1)
      CALL JCONWV(XBEG+XLEN,YBEG+YLEN,0.,X2,Y2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW AND VIEWPORT FOR DATA AREA
      CALL JVPORT(X1,X2,Y1,Y2)
      CALL JWINDO(0.,REAL(NUMPTS+1),0.,10.)
C OPEN SEGMENT, AND PLOT BARS
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINES,NCOLS,NTLINES)
```

```
      CALL JPINTR(1)
      XWIDTH=2./3.
      XHALF=XWIDTH/2.
      DO 99 I=1,NUMPTS
 99   TOTAL(I)=0.0
      CALL JSIZE(CSIZE,CSIZE*1.25)
      DO 2 I=1,NUMSET
        CALL JPIDEX(I,47-(I-1))
        CALL CKEYLB(ISTORE,KEYCHR,-4,KEYLAB(I))
        DO 2 J=1,NUMPTS
          CALL JRECT(REAL(J)-XHALF,TOTAL(J),
     +               REAL(J)+XHALF,TOTAL(J)+REVCRU(I,J))
 2    TOTAL(J)=TOTAL(J)+REVCRU(I,J)
      CALL JCLOSE
C-------------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE AREA
       CALL JSETDB(0)
       CALL JVPORT(-1.,1.,-.773,.773)
       CALL JWINDO(0.,10.,0.,7.73)
       BPOS(1)=XBEG+XLEN
       BPOS(2)=YBEG+YLEN
       CALL JOPEN
       CALL JSIZE(CSIZE,CSIZE*1.25)
C      CALL CKLPLN(.08)
       CALL CKEYPL(ISTORE,KEYCHR,' ',' ',BPOS,3,JCOL,0)
       CALL JCLOSE
C---------------------------------------
C TERMINATE DI3000 AND CGL
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

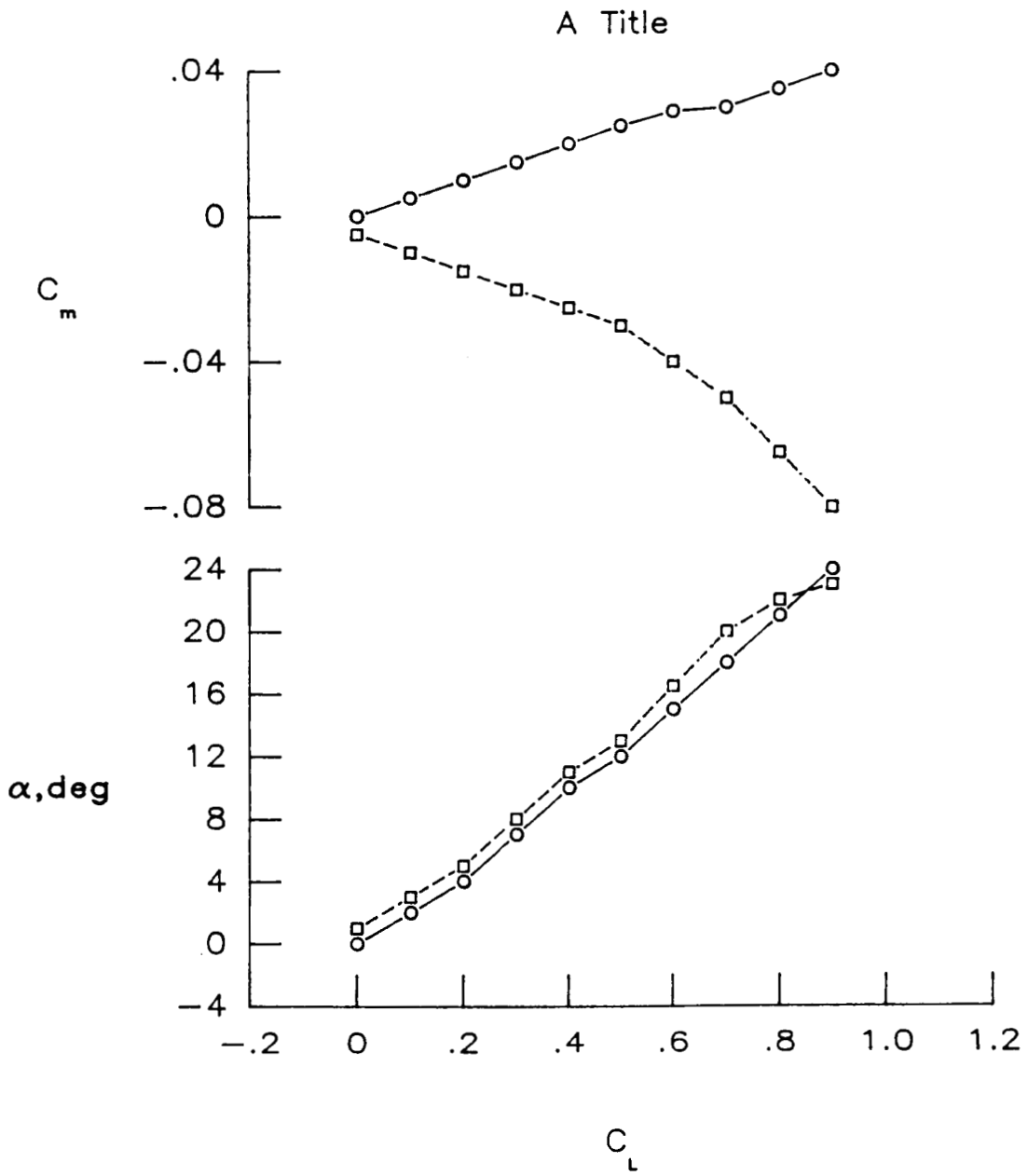


Figure CDC1. Composite chart (i.e., two charts on one page).

```
      PROGRAM CDC1
C-----------------------------------------------------------------------
C COMPOSITE CHART (IE, TWO CHARTS ON ONE PAGE)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=10,MAXSET=4)
      REAL X(MAXPTS,MAXSET),Y(MAXPTS,MAXSET)
      DATA (X(KI,1),KI=1,10)
     + /0.,.1,.2,.3,.4,.5,.6,.7,.8,.9/
      DATA(X(KI,2),KI=1,10)
     + /.1,.15,.3,.35,.45,.55,.65,.75,.85,1.2/
      DATA(X(KI,3),KI=1,10)
     + /.0,.1,.2,.3,.4,.5,.6,.7,.8,.9/
      DATA(X(KI,4),KI=1,10)
     + /.1,.2,.4,.6,.7,.8,.9,1.,1.1,1.2/
      DATA (Y(KI,1),KI=1,10)
     + /0.,2.,4.,7.,10.,12.,15.,18.,21.,24./
      DATA (Y(KI,2),KI=1,10)
     + /1.,3.,5.,8.,11.,13.,16.5,20.,22.,23./
      DATA (Y(KI,3),KI=1,10)
     + /0.,.005,.01,.015,.02,.025,.029,.03,.035,.04/
      DATA (Y(KI,4),KI=1,10)
     + /-.005,-.01,-.015,-.02,-.025,-.03,-.04,-.05,-.065,-.08/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      XS=.2
      YS=6.0
      CALL CVSPAC(10.,10.)
      CALL J4RGET(2,VX3,VX4,VY3,VY4)
      CALL JWINDO(0.,10.,0.,10.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(XS,XS*1.25)
C POSITION TEXT, AND SET TTEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(5.0,10.0)
      CALL JJUST(2,3)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(XS,XS*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.0
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('C[BSUB]L',1)
      CALL CHTICJ(1)
```

```
C XLEN-REPRESENTS THE X-AXIS LENGTH
      XLEN=6.0
      CALL CHPREC(1)
      CALL CHAXIS(-.2,1.2,.2,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('[FONT=9][BLC]A[FONT=3],DEG',1)
      CALL CVPREC(3)
      CALL CVTICJ(1)
C YLEN-REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVPREC(-1)
      CALL CVJUST(1)
      CALL CVAXIS(-4.,24.,4.,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C---------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(-.2,1.2,-4.,24.)
C -.2,1.2 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C -4.,24. - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 902, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C---------------------------------------------------------------------
C WORK ON SECOND GRAPH
      CALL JWINDO(0.,10.,0.,10.)
      CALL JVPORT(VX3,VX4,VY3,VY4)
      CALL JOPEN
      CALL JSIZE(XS,XS*1.25)
      CALL JMOVE(XORG,YS)
C YLEN-REPRESENTS THE Y-AXIS LENGTH
      CALL CVLAB('C[BSUB][BLC]M',1)
      CALL CVPREC(2)
      CALL CVAXIS(-.08,.04,.04,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YS,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YS+3.5,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C---------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
```

```
      CALL JWINDO(-.2,1.2,-.08,.04)
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT THIRD DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,3),MAXPTS)
C PLOT FOURTH SET OF DATA POINTS
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,4),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

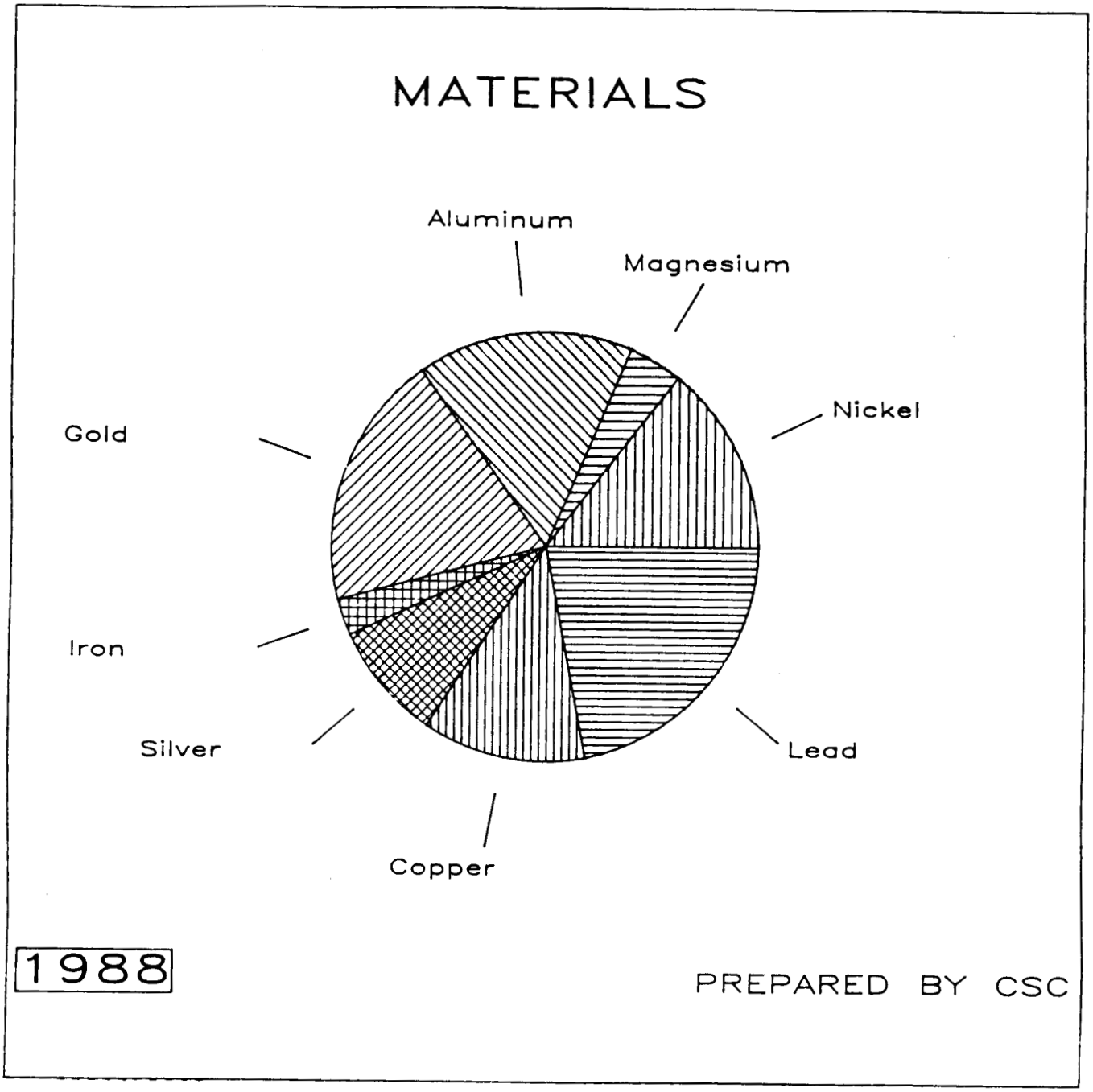


Figure CDP1. Basic pie chart.

```
      PROGRAM CDP1
C-------------------------------------------------------------
C BASIC PIE CHART
C-------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C-------------------------------------------------------------
C INITIALIZE DATA
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.25,2.,2.35,2.85,2.95/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/21.,6.,24.,28./
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/4.,13.,18.,32./
C
      DATA YEAR /'1988'/
C
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'G[BLC]OLD'/
      DATA LABEL(5)/'I[BLC]RON'/
      DATA LABEL(6)/'S[BLC]ILVER'/
      DATA LABEL(7)/'C[BLC]OPPER'/
      DATA LABEL(8)/'L[BLC]EAD'/
C-------------------------------------------------------------
C INITIALIZE DI-3000
       CALL JBEGIN
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-------------------------------------------------------------
C FILL POLYGONS BY DEFAULT.
      CALL JDPINT(1)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.4,.4)
      CALL JFONT(3)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.2)
      CALL JHSTRG('MATERIALS')
      CALL JMOVE(.1,.8)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
```

```
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.8)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-----------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-----------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-----------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      START=0.
      CALL JSIZE(.2,.2)
      CALL JFONT(1)
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        START=A1
        CALL JPIDEX(I,I+30)
        CALL JSECTR(5.,5.,0.,RAD1,0,A0,A1)
C
        AMID=(A1-A0)/2.+A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
        CALL JMOVE(5.+RAD2*C1,5.+RAD2*S1)
        CALL JDRAW(5.+RAD3*C1,5.+RAD3*S1)
C
        IF (AMID.LT.45. .OR. AMID .GE. 315.) CALL JJUST(1,2)
        IF (AMID .GT. 45. .AND. AMID .LT. 135.) CALL JJUST(2,1)
        IF (AMID .GE. 135. .AND. AMID .LT. 225.) CALL JJUST(3,2)
        IF (AMID .GE. 225. .AND. AMID .LT. 315.) CALL JJUST(2,3)
C
        CALL JMOVE(5.+RAD4*C1,5.+RAD4*S1)
 5000 CALL JHSTRG(LABEL(I))
C-----------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-----------------------------------------------------------------
C PAUSE TO VIEW CHART.
      CALL JPAUSE(IDEV)
C-----------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
```

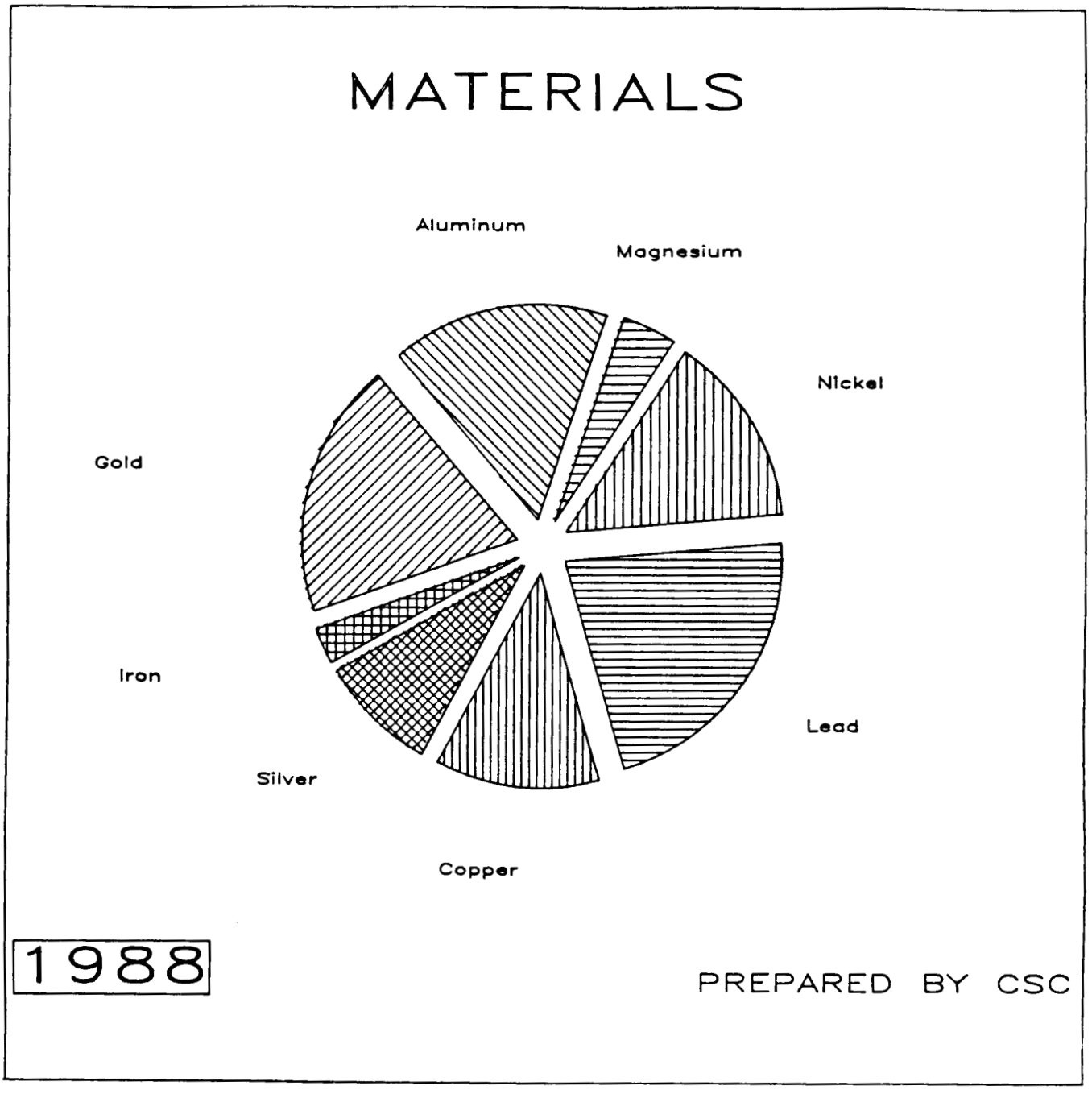


Figure CDP2. Exploded pie chart.

PRECEDING PAGE BLANK NOT FILMED

```
      PROGRAM CDP2
C-----------------------------------------------------------------
C EXPLODED PIE CHART
C-----------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.25,2.,2.35,2.85,2.95/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/21.,6.,24.,28./
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/4.,13.,18.,32./
C
      DATA YEAR /'1988'/
C
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'G[BLC]OLD'/
      DATA LABEL(5)/'I[BLC]RON'/
      DATA LABEL(6)/'S[BLC]ILVER'/
      DATA LABEL(7)/'C[BLC]OPPER'/
      DATA LABEL(8)/'L[BLC]EAD'/
C-----------------------------------------------------------------
C INITIALIZE DI-3000
      CALL CBEGIN
      CALL JBEGIN
C-----------------------------------------------------------------
C WRITE TO A DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-----------------------------------------------------------------
C FILL POLYGONS BY DEFAULT
      CALL JDPINT(1)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.5,.5)
      CALL JFONT(3)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.2)
      CALL JHSTRG('MATERIALS')
      CALL JMOVE(.1,.8)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
```

```
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.8)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-----------------------------------------------------------
C CLOSE TEMPORARY SEGMENT
      CALL JCLOSE
C-----------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-----------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      START=5.
      CALL JSIZE(.15,.15)
      CALL JFONT(1)
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        START=A1
        AMID=(A1-A0)/2. +A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
        CALL JPIDEX(I,I+30)
C
        X0=OFFSET*C1+5.
        Y0=OFFSET*S1+5.
C
        CALL JSECTR(X0,Y0,0.,RAD1,0,A0,A1)
C
        CALL JMOVE(RAD2*C1+5.,RAD2*S1+5.)
C
        IF (AMID.LT.45. .OR. AMID .GE. 315.) CALL JJUST(1,2)
        IF (AMID .GT. 45. .AND. AMID .LT. 135.) CALL JJUST(2,1)
        IF (AMID .GE. 135. .AND. AMID .LT. 225.) CALL JJUST(3,2)
        IF (AMID .GE. 225. .AND. AMID .LT. 315.) CALL JJUST(2,3)
C
        CALL JMOVE(RAD4*C1+5.,RAD4*S1+5.)
 5000   CALL JHSTRG(LABEL(I))
C-----------------------------------------------------------
C CLOSE TE TEMPORARY SEGMENT.
      CALL JCLOSE
C
C-----------------------------------------------------------
```

```
C PAUSE TO VIEW CHART.
      CALL JPAUSE(IDEV)
C-----------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
      STOP
      END
```

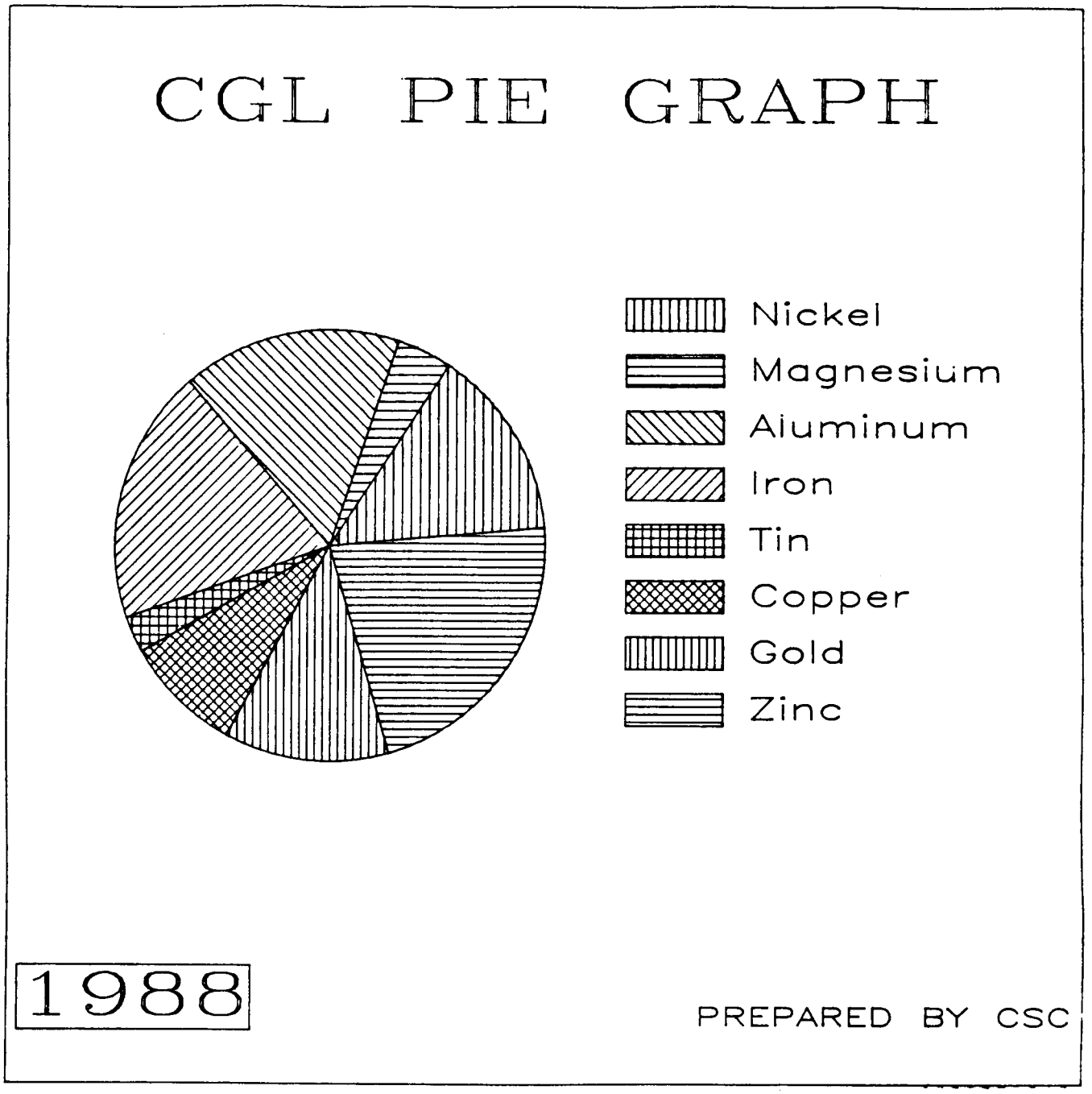


Figure CDP3. Basic pie chart with a legend.

```
      PROGRAM CDP3
C-------------------------------------------------------------
C BASIC PIE CHART WITH A LEGEND
C-------------------------------------------------------------
C INITIALIZE DATA
      REAL OFFSET,RAD1,RAD2,RAD3,RAD4
      REAL BPOS(4)
      REAL XV(4),YV(4),ZV(4),REGION(8)
      CHARACTER*14 LABEL(8)
      CHARACTER*4 YEAR
C
      CHARACTER KEYCHR(1,8)*15
      INTEGER ISTORE (15,8)
      DATA OFFSET,RAD1,RAD2,RAD3,RAD4 /.5,4.,4.7,5.7,5.9/
      DATA REGION(1),REGION(2),REGION(3),REGION(4)/2.1,.6,2.4,2.8/
      DATA REGION(5),REGION(6),REGION(7),REGION(8)/.4,1.3,1.8,3.2/
C
      DATA YEAR /'1988'/
C
      DATA LABEL(1)/'N[BLC]ICKEL'/
      DATA LABEL(2)/'M[BLC]AGNESIUM'/
      DATA LABEL(3)/'A[BLC]LUMINUM'/
      DATA LABEL(4)/'I[BLC]RON'/
      DATA LABEL(5)/'T[BLC]IN'/
      DATA LABEL(6)/'C[BLC]OPPER'/
      DATA LABEL(7)/'G[BLC]OLD'/
      DATA LABEL(8)/'Z[BLC]INC'/
C-------------------------------------------------------------
C INITIALIZE DI-3000
      CALL JBEGIN
      CALL CBEGIN
C-------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-------------------------------------------------------------
C DEFINE THE VIEWSPACE AND WINDOW OF SAME ASPECT RATIO
      CALL JVSPAC(-1.,1.,-1.,1.)
      CALL JWINDO(0.,10.,0.,10.)
C-------------------------------------------------------------
C FILL POLYGONS BY DEFAULT.
      CALL JDPINT(1)
C-------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR BORDERS.
      CALL JOPEN
      CALL JIWIND(XV,YV,ZV)
      CALL JPIDEX(4,0)
      CALL JPOLGN(XV,YV,4)
      CALL JSIZE(.6,.6)
      CALL JFONT(5)
      CALL JJUST(2,2)
      CALL JMOVE(5.,9.1)
      CALL JHSTRG('CGL PIE GRAPH')
```

```
      CALL JMOVE(.1,.5)
      CALL JHSXTN(YEAR,DX,DY)
      CALL JPIDEX(6,0)
      CALL JRRECT(DX,DY)
      CALL JJUST(1,1)
      CALL JHSTRG(YEAR)
      CALL JMOVE(9.9,.5)
      CALL JSIZE(.25,.25)
      CALL JFONT(1)
      CALL JJUST(3,1)
      CALL JHSTRG('PREPARED BY CSC')
C-----------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-----------------------------------------------------------------
C DESIGN THE PIE TO BE 50% SMALLER
       RAD1=RAD1-(RAD1*.50)
       RAD2=RAD2-(RAD2*.50)
       RAD3=RAD3-(RAD3*.50)
       RAD4=RAD4-(RAD4*.50)
C-----------------------------------------------------------------
C COMPUTE THE FACTOR FOR CONVERTING DEGREES TO RADIANS.
      DTORAD=ATAN(1.0)/45.
C-----------------------------------------------------------------
C TOTAL THE REGIONAL DATA.
      TOTAL=0.0
      DO 3000 I=1,8
 3000   TOTAL = TOTAL+REGION(I)
C-----------------------------------------------------------------
C OPEN A TEMPORARY SEGMENT FOR THE PIE CHART SECTORS AND LABELS.
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,8,1,1)
      START=5.
      CALL JSIZE(.3,.3)
      CALL JFONT(1)
      DO 5000 I=1,8
        A0=START
        A1=REGION(I)/TOTAL*360.+START
        CALL JPIDEX(I,I+30)
        CALL JSECTR(3.,5.,3.,RAD1,0,A0,A1)
C
        START=A1
        AMID=(A1-A0)/2.+A0
        ARAD=AMID*DTORAD
        C1=COS(ARAD)
        S1=SIN(ARAD)
 5000   CALL CKEYLB(ISTORE,KEYCHR,-4,LABEL(I))
      BPOS(1)=9.5
      BPOS(2)=8.
      CALL CKEYPL(ISTORE,KEYCHR,' ',' ',BPOS,5,1,0)
C-----------------------------------------------------------------
C CLOSE THE TEMPORARY SEGMENT.
      CALL JCLOSE
C-----------------------------------------------------------------
```

```
C PAUSE FOR OPERATOR ACTION.
      CALL JPAUSE(IDEV)
C-----------------------------------------------------------------
C TERMINATE DI-3000; ALL DEVICES ARE DESELECTED AND TERMINATED.
      CALL JEND
      STOP
      END
```

Figure CDR1. Simple program to demonstrate symbols and polygon fill patterns.

```
      PROGRAM CDR1
C-----------------------------------------------------------------------
C SIMPLE PROGRAM TO DEMONSTRATE SYMBOLS AND POLYGON FILL PATTERNS
C-----------------------------------------------------------------------
C  INITIALIZE DI-3000
      CALL JBEGIN
C-----------------------------------------------------------------------
C  THIS MUST BE THE FIRST CGL CALL TO INITIALIZE THE CGL
      CALL CBEGIN
C-----------------------------------------------------------------------
C  WRITE TO THE IDEV DEVICE
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C  SET UP THE VIEWING WINDOW
      CALL JWINDO(0.,11.,0.,11.)
C-----------------------------------------------------------------------
C  OPEN CURRENT SEGMENT
      CALL JOPEN
C-----------------------------------------------------------------------
C  SET THE CURRENT POLYGON AND SET THE CURRENT SIZE OF SYMBOLS
      CALL JPINTR(1)
      CALL CSYMSZ(.5)
C-----------------------------------------------------------------------
C  A LOOP WHICH PRINTS NASA POINT SYMBOLS WITHIN CURRENT SEGMENT
      DO 1 I=1,10
        CALL JPIDEX(1,48-I)
        CALL CSYMNO(I)
 1    CALL CPNTPT(REAL(I),1.)
C-----------------------------------------------------------------------
C  ENDS ALL GRAPGICS AND CLOSES CURRENT SEGMENT
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

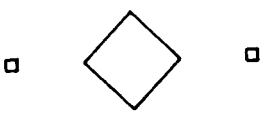


Figure CDR2. Simple program to demonstrate symbols and ploygon symbol sizes.

```
      PROGRAM CDR2
C-----------------------------------------------------------------------
C SIMPLE PROGRAM TO DEMONSTRATE SYMBOLS AND POLYGON SYMBOL SIZES
C-----------------------------------------------------------------------
C  INITIALIZE DI-3000 AND SET UP CURRENT IDEV DEVICE
      IDEV=0
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C  INITIALIZE CGL ATTRIBUTES TO DEFAULTS AND SET CURRENT WINDOW
      CALL CBEGIN
      CALL JWINDO(0.,11.,0.,11.)
C-----------------------------------------------------------------------
C  OPEN CURRENT SEGMENT
      CALL JOPEN
C-----------------------------------------------------------------------
C OUTPUT SYMBOL (WITH DEFAULT SYMBOL NUMBER AND SYMBOL SIZE)
      CALL CPNTPT(1.,1.)
C-----------------------------------------------------------------------
C CHANGE SYMBOL NUMBER TO 3, SYMBOL SIZE TO .5, AND DISPLAY
      CALL CSYMNO(3)
      CALL CSYMSZ(.5)
      CALL CPNTPT(2.,1.)
C-----------------------------------------------------------------------
C RESET CGL ATTRIBUTES BACK TO DEFAULTS
      CALL CBEGIN
      CALL CPNTPT(3.,1.)
C-----------------------------------------------------------------------
C  END ALL GRAPHICS AND CLOSE CURRENT SEGEMENT
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

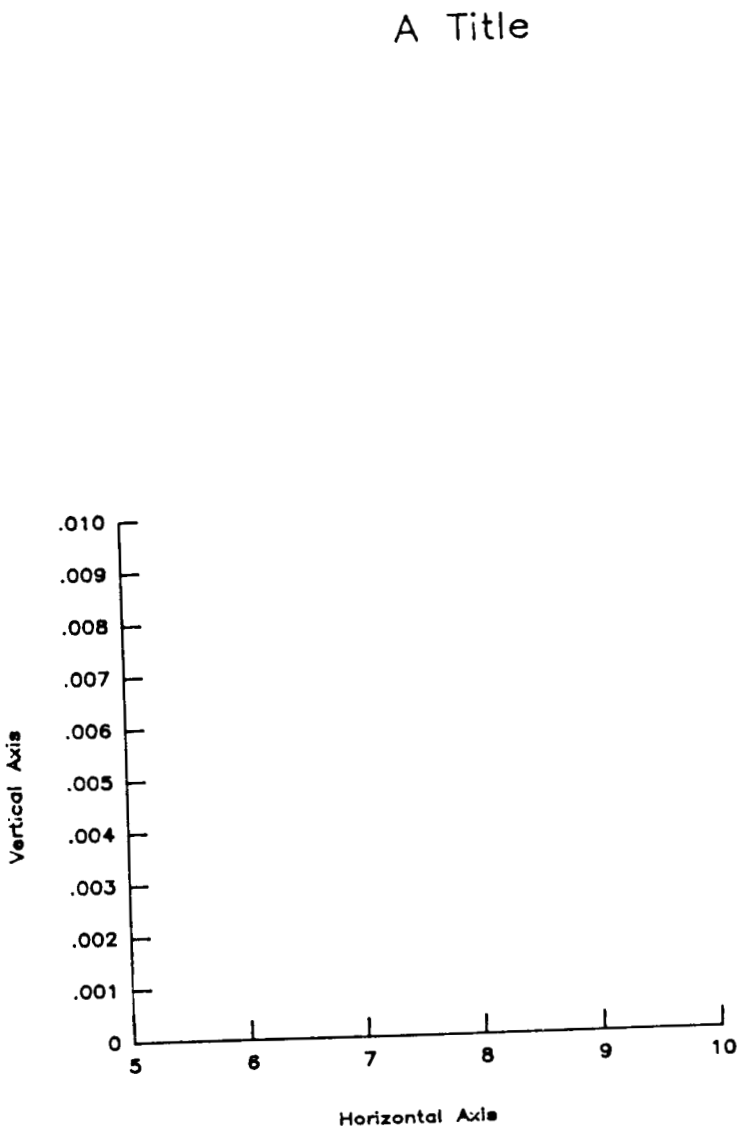


Figure CDR3. Simple program to demonstrate how to generate a set of axes.

```
      PROGRAM CDR3
C-----------------------------------------------------------------------
C SIMPLE PROGRAM TO DEMONSTRATE HOW TO GENERATE A SET OF AXES
C-----------------------------------------------------------------------
C  INITIALIZE CGL AND DI-3000
      CALL CBEGIN
      CALL JBEGIN
C-----------------------------------------------------------------------
C  WRITE TO THE IDEV DEVICE
      IDEV=0
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C  SET UP WINDOW AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
C-----------------------------------------------------------------------
C  OPEN CURRENT SEGMENT
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C-----------------------------------------------------------------------
C POSITION STRING, AND SET JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,8.75)
      CALL JJUST(2,2)
C-----------------------------------------------------------------------
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C-----------------------------------------------------------------------
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(2.,2.)
C-----------------------------------------------------------------------
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHAXIS(5.,10.,1.,4.)
C-----------------------------------------------------------------------
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
      CALL CVAXIS(.0,.01,.001,3.5)
C-----------------------------------------------------------------------
C  END ALL GRAPHICS AND CLOSE CURRENT SEGMENT
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

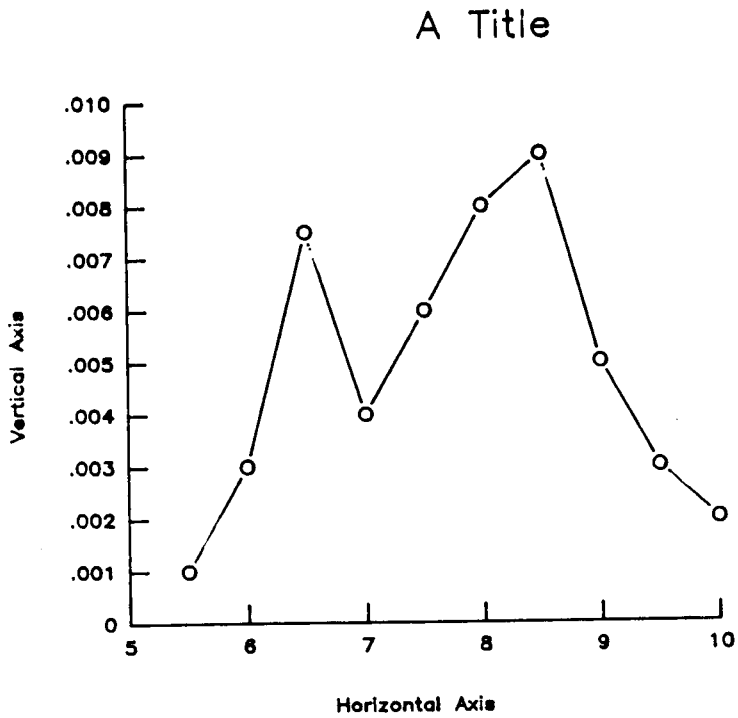


Figure CDR4. Basic line chart.

```
      PROGRAM CDR4
C-----------------------------------------------------------------------
C  BASIC LINE CHART
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=1)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(2.,2.)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHAXIS(5.,10.,1.,4.)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
      CALL CVAXIS(.0,.01,.001,3.5)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(2.,2.,0.,VX1,VY1)
      CALL JCONWV(2.+4.,2.+3.5,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
```

```
      CALL CLNPLT(X,Y(1,1),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

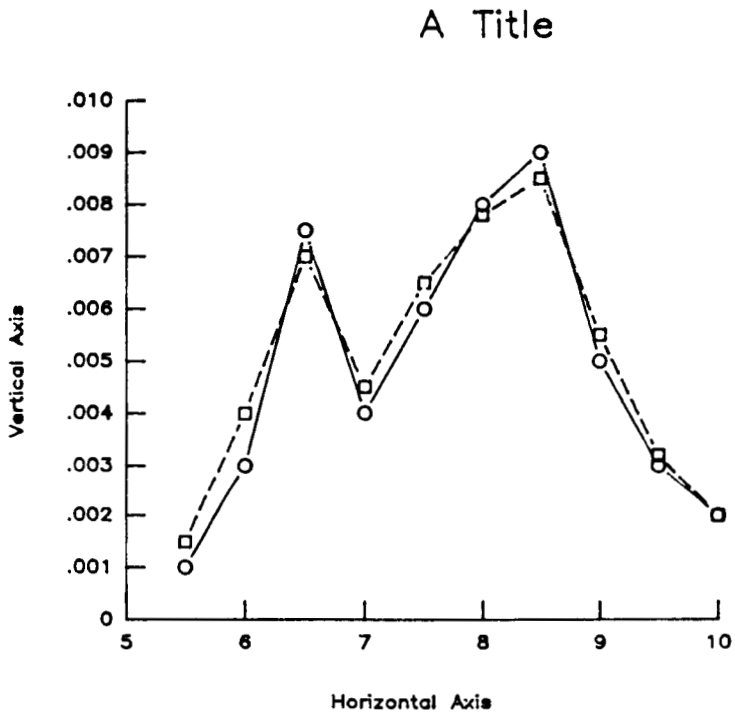


Figure CDR5. Complete line chart.

```
      PROGRAM CDR5
C-----------------------------------------------------------------------
C  COMPLETE LINE CHART
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
```

```
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

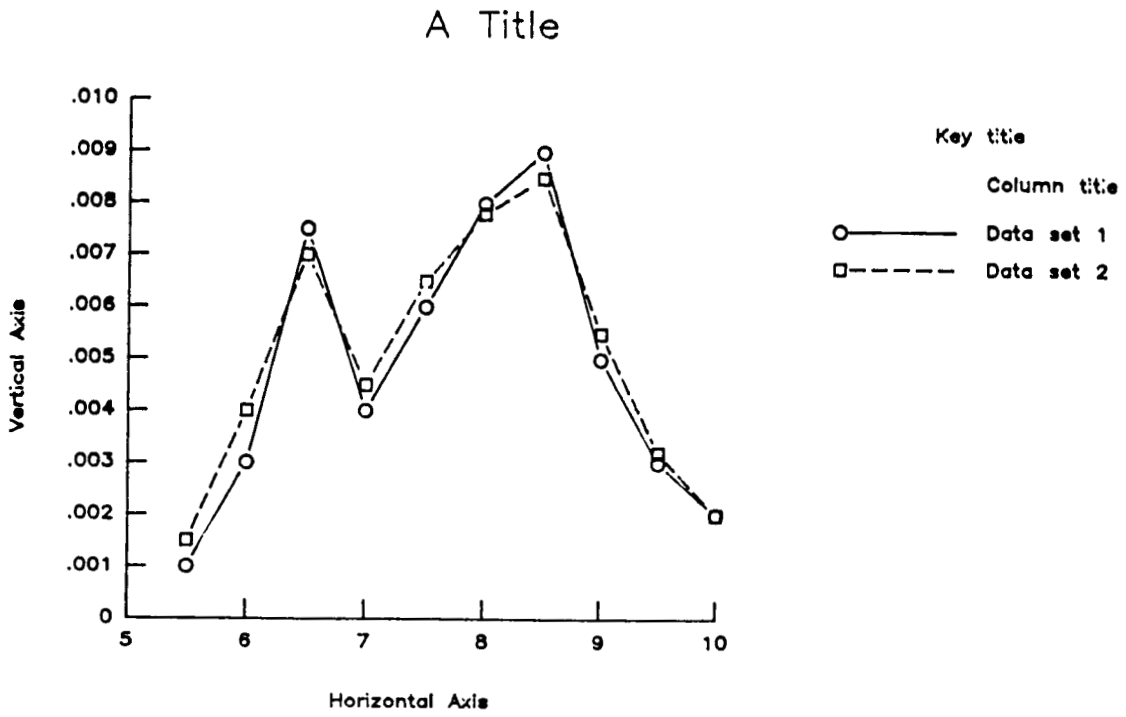


Figure CDR6. Complete line chart with a legend.

```
      PROGRAM CDR6
C-----------------------------------------------------------------------
C  COMPLETE LINE CHART WITH A LEGEND
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
```

```
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C PLOT FIRST DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 1')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 2')
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(.1,.1*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE','C[BLC]OLUMN TITLE',
     +            BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

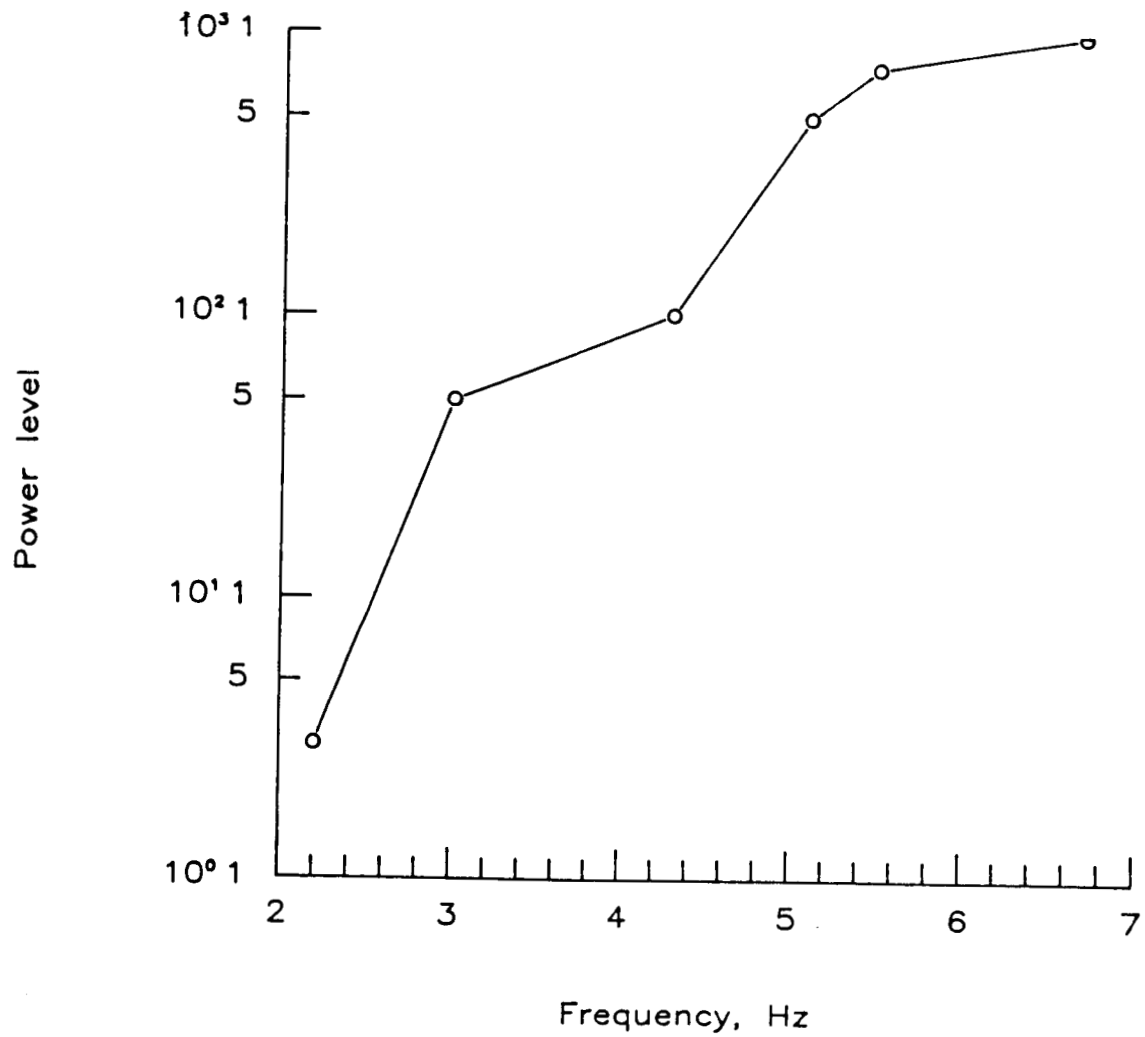


Figure CDR7. Semi-logarithmic line chart (single data set).

```
      PROGRAM CDR7
C-----------------------------------------------------------------------
C  SEMI-LOGARITHMIC LINE CHART (SINGLE DATA SET)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(NPTS=6)
      REAL X(NPTS),Y(NPTS),YTEMP(NPTS)
      DATA X/2.2,3.0,4.3,5.1,5.5,6.7/
      DATA Y/3.,50.,100.,500.,750.,985./
      DATA XORG/2.5/,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C-----------------------------------------------------------------------
C SET UP GRAPHICS AREA
      IDEV=0
      CALL JBEGIN
      CALL CBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET PAGE COORDINATES, AND OPEN A TEMPORARY SEGMENT
      CALL JWINDO(0.,XPAGE,0.,YPAGE)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET CHARACTER SIZE, AND OUTPUT TITLE
C SET THE CHARACTER SIZE (IN TERMS OF PAGE CORRDINATES)
      XSIZE=.25
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,BOTTOM)
      CALL JMOVE(XPAGE/2.,0.+XSIZE)
      CALL JJUST(2,1)
C OUTPUT THE STRING
      CALL JHSTRG('P[BLC]OWER LEVEL VS. FREQUENCY')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LINEAR)
      CALL CMNMX(X,NPTS,XMIN,XMAX)
C DETERMINE PUBLICATION QUALITY SCALE FACTORS
      CALL CSCALE(XMIN,XMAX,AMIN,AMAX,NMAJOR,NMINOR,XINCR,NLEFT,NRIGHT)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHMTIC(NMINOR)
      CALL CHPREC(NRIGHT)
      CALL CHLAB('F[BLC]REQUENCY, [BUC]H[BLC]Z',1)
      CALL CHAXIS(AMIN,AMAX,XINCR,XLEN)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y,NPTS,YMIN,YMAX)
      CALL CEXP(YMIN,YMAX,MINEXP,MAXEXP)
      CALL CVLAB('P[BLC]OWER LEVEL',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MINEXP)
```

```
      NTIC=(MAXEXP-MINEXP)+1
      CALL CVLOG(YLEN,NTIC,4)
C-----------------------------------------------------------------------
C SAVE DATA REGION VIRTUAL COORDINATES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET VIEWPORT TO DATA REGION VIRTUAL COORDINATES
      CALL JVPORT(VX1,VX2,VY1,VY2)
C-----------------------------------------------------------------------
C SET UP DATA REGION WINDOW
      DO 1 KI=1,NPTS
 1    YTEMP(KI)=LOG10(Y(KI))
C X-AXIS IS LOG, MIN AND MAX MUST BE SCALED.
      CALL JWINDO(AMIN,AMAX,REAL(MINEXP),REAL(MAXEXP))
C ENABLE CLIPPING TO EXCLUDE EXTRANEOUS DATA
      CALL JWCLIP(.TRUE.)
      CALL JOPEN
C PLOT DATA
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,YTEMP,NPTS)
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
C-----------------------------------------------------------------------
      STOP
      END
```

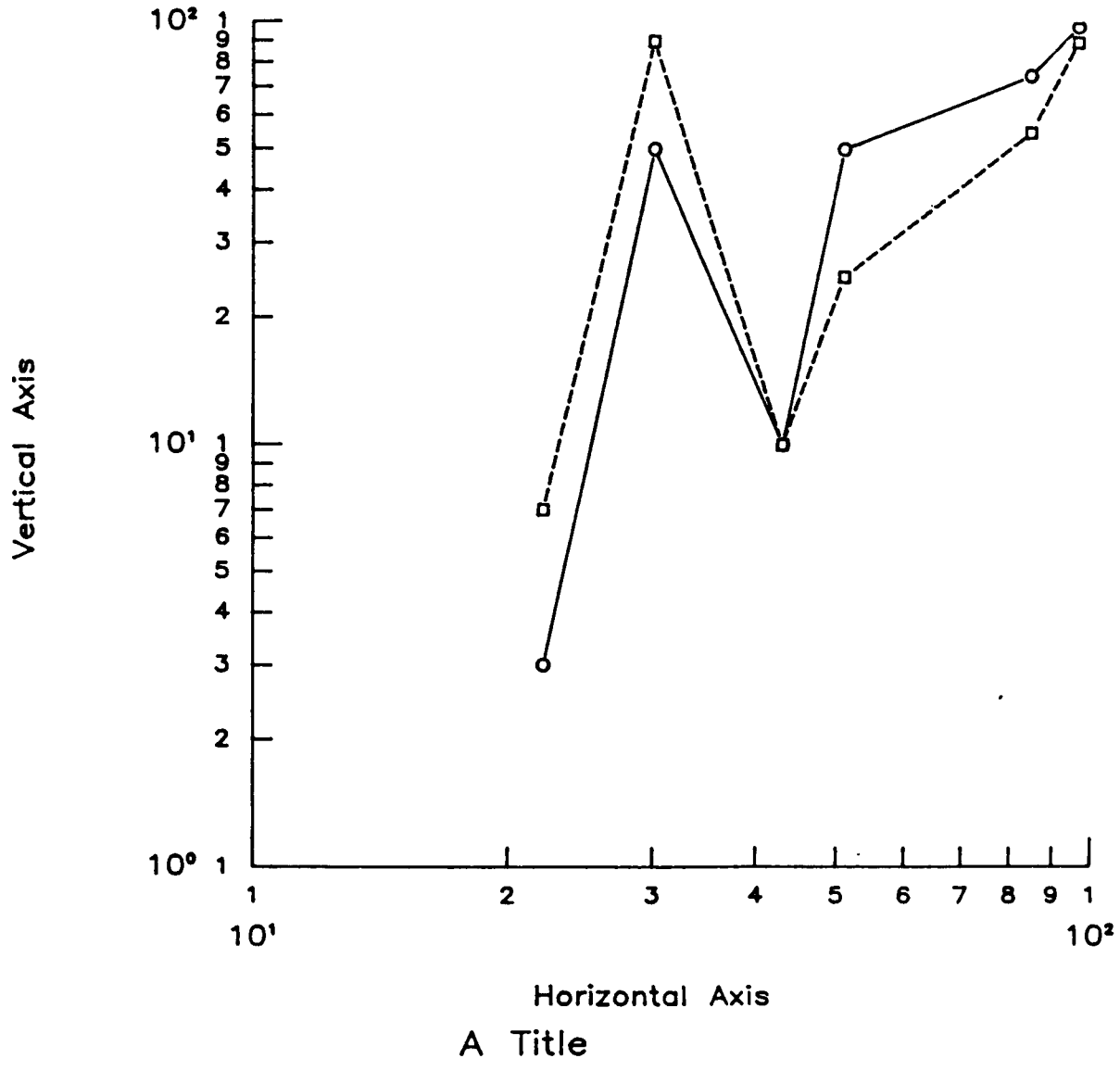


Figure CDR8. Log-log line chart (multiple data sets).

```
      PROGRAM CDR8
C-----------------------------------------------------------------------
C  LOG-LOG LINE CHART (MULTIPLE DATA SETS)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(MAXPTS=6,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),YTEMP(MAXPTS),XTEMP(MAXPTS)
      DATA X/22.,30.,43. ,51. ,85. ,97. /
      DATA (Y(KI,1),KI=1,MAXPTS)/3.0,50.,10. ,50. ,75. ,98. /
      DATA (Y(KI,2),KI=1,MAXPTS)/7.0,90.,10.,25. ,55. ,90. /
      DATA XORG/2./,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C-----------------------------------------------------------------------
C SET UP GRAPHICS AREA
      IDEV=0
      CALL JBEGIN
      CALL CBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET PAGE COORDINATES, AND OPEN A TEMPORARY SEGMENT
      CALL JWINDO(0.,XPAGE,0.,YPAGE)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET CHARACTER SIZE, AND OUTPUT TITLE
C SET THE CHARACTER SIZE (IN TERMS OF PAGE CORRDINATES)
      XSIZE=.25
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,BOTTOM)
      CALL JMOVE(XPAGE/2.,0.+XSIZE)
      CALL JJUST(2,1)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LOG)
      CALL CMNMX(X,MAXPTS,XMIN,XMAX)
      CALL CEXP(XMIN,XMAX,MXEXP1,MXEXP2)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CSET('NHLOGF',1)
      CALL CSET('NHLOGS',MXEXP1)
      NTICX=(MXEXP2-MXEXP1)+1
      CALL CHLOG(XLEN,NTICX,1)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y(1,1),MAXPTS,YMIN,YMAX)
      DO 10 I=2,MAXSET
        CALL CMNMX(Y(1,I),MAXPTS,TMIN,TMAX)
        IF(TMIN.LT.YMIN)YMIN=TMIN
```

```
 10     IF(TMAX.GT.YMAX)YMAX=TMAX
      CALL CEXP(YMIN,YMAX,MYEXP1,MYEXP2)
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MYEXP1)
      NTICY=(MYEXP2-MYEXP1)+1
      CALL CVLOG(YLEN,NTICY,1)
C-----------------------------------------------------------------------
C SAVE DATA REGION VIRTUAL COORDINATES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET VIEWPORT TO DATA REGION VIRTUAL COORDINATES
      CALL JVPORT(VX1,VX2,VY1,VY2)
C-----------------------------------------------------------------------
C SET UP DATA REGION WINDOW
C X-AXIS IS LOG, MIN AND MAX MUST BE SCALED.
      CALL JWINDO(REAL(MXEXP1),REAL(MXEXP2),REAL(MYEXP1),REAL(MYEXP2))
C ENABLE CLIPPING TO EXCLUDE EXTRANEOUS DATA
      CALL JWCLIP(.TRUE.)
      CALL JOPEN
C PLOT DATA
      DO 1 KI=1,MAXPTS
 1    XTEMP(KI)=LOG10(X(KI))
      DO 2 KI1=1,MAXSET
        CALL CSYMNO(KI1)
      CALL CLNPAT(KI1)
        DO 3 KI2=1,MAXPTS
 3      YTEMP(KI2)=LOG10(Y(KI2,KI1))
 2    CALL CLNPLT(XTEMP,YTEMP,MAXPTS)
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
C-----------------------------------------------------------------------
      STOP
      END
```

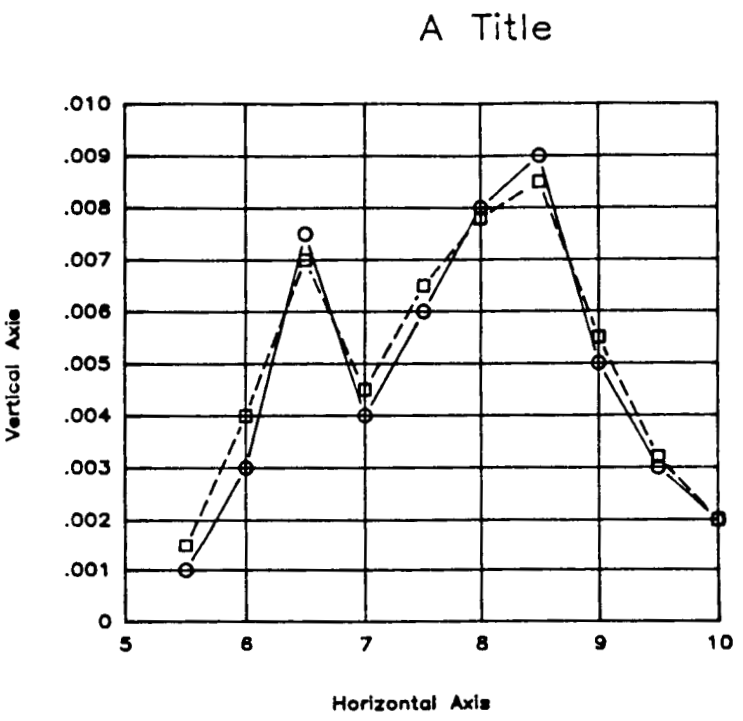


Figure CDR9. Line chart with a grid (without a legend).

```
      PROGRAM CDR9
C-----------------------------------------------------------------------
C  LINE CHART WITH A GRID (WITHOUT A LEGEND)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C CALL GRID ROUTINE TO REQUEST GRID LINES TO BE OUTPUT OVER DATA AREA.
      NOINCX=NINT((10.-5.)/1.)
      NOINCY=NINT((.01-.0)/.001)
      XS=XLEN/REAL(NOINCX)
      YS=YLEN/REAL(NOINCY)
      NBLANK=0
```

```
      CALL CGRID(XORG,YORG,XS,YS,NOINCX,NOINCY,BLANK,NBLANK)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C---------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C---------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

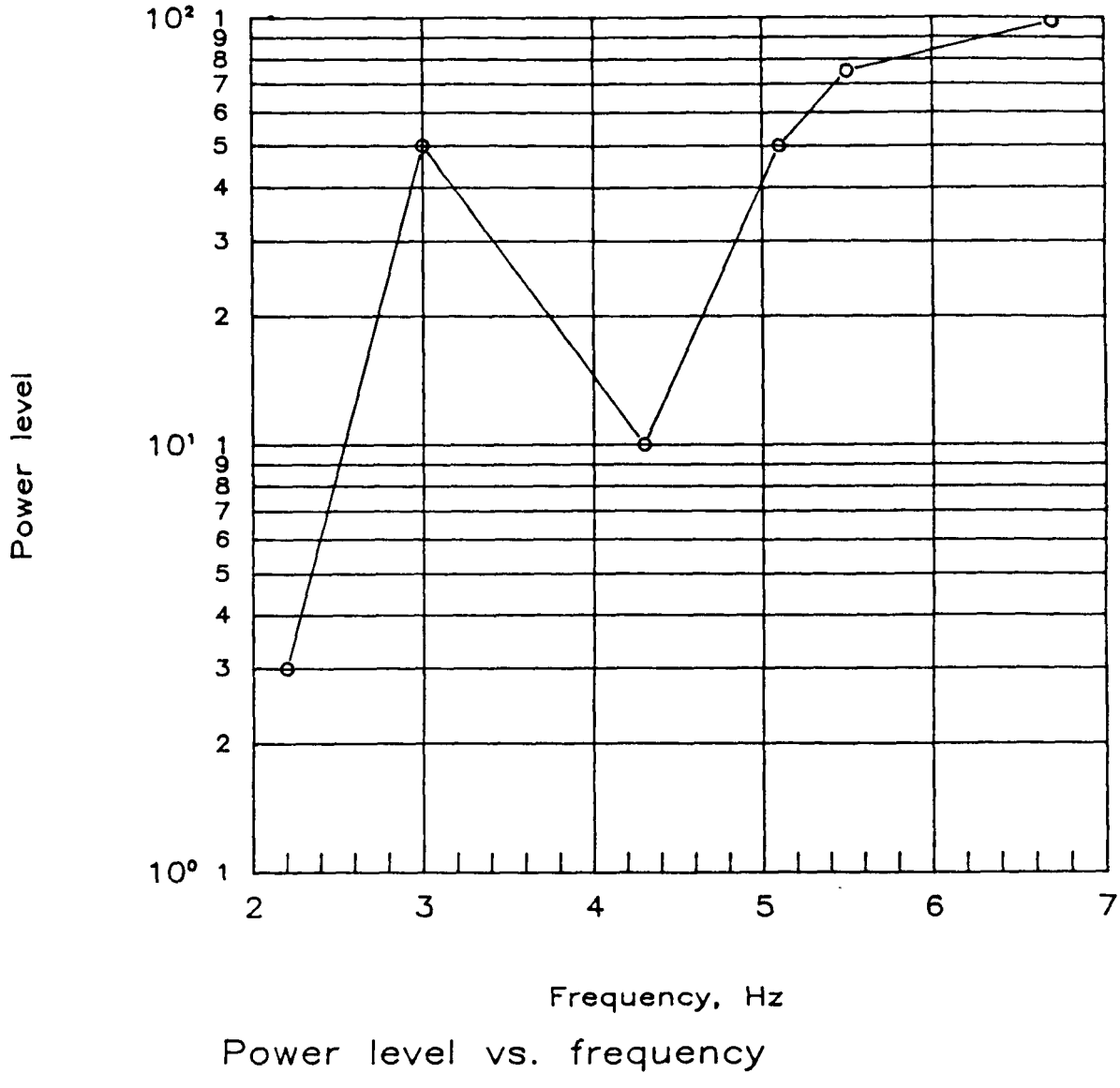


Figure CDR10. Semi-log line chart with a grid (single data set).

```
      PROGRAM CDR10
C-----------------------------------------------------------------------
C  SEMI-LOG LINE CHART WITH A GRID (SINGLE DATA SET)
C-----------------------------------------------------------------------
C SET UP DATA
      PARAMETER(NPTS=6)
      REAL X(NPTS),Y(NPTS),YTEMP(NPTS),BLANK(4)
      DATA X/2.2,3.0,4.3,5.1,5.5,6.7/,BLANK/0.,0.,0.,0./
      DATA Y/3.,50.,10. ,50. ,75. ,98. /
      DATA XORG/2.5/,YORG/2./,XLEN/7./,YLEN/7./,XPAGE/11./,YPAGE/11./
C-----------------------------------------------------------------------
C SET UP GRAPHICS AREA
      IDEV=0
      CALL JBEGIN
      CALL CBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C SET PAGE COORDINATES, AND OPEN A TEMPORARY SEGMENT
      CALL JWINDO(0.,XPAGE,0.,YPAGE)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET CHARACTER SIZE, AND OUTPUT TITLE
C SET THE CHARACTER SIZE (IN TERMS OF PAGE CORRDINATES)
      XSIZE=.25
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,BOTTOM)
      CALL JMOVE(XPAGE/2.,0.+XSIZE)
      CALL JJUST(2,1)
C OUTPUT THE STRING
      CALL JHSTRG('P[BLC]OWER LEVEL VS. FREQUENCY')
C-----------------------------------------------------------------------
C RESET THE CHARACTER SIZE FOR THE AXES
      XSIZE=.2
      CALL JSIZE(XSIZE,XSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      CALL JMOVE(XORG,YORG)
C-----------------------------------------------------------------------
C HORIZONTAL AXIS (LINEAR)
      CALL CMNMX(X,NPTS,XMIN,XMAX)
C DETERMINE PUBLICATION QUALITY SCALE FACTORS
      CALL CSCALE(XMIN,XMAX,AMIN,AMAX,NMAJOR,NMINOR,XINCR,NLEFT,NRIGHT)
C SET HORIZONTAL AXIS ATTRIBUTES
      CALL CHMTIC(NMINOR)
      CALL CHPREC(NRIGHT)
      CALL CHLAB('F[BLC]REQUENCY, [BUC]H[BLC]Z',1)
      CALL CHAXIS(AMIN,AMAX,XINCR,XLEN)
C-----------------------------------------------------------------------
C VERTICAL AXIS (LOGARITHMIC)
      CALL CMNMX(Y,NPTS,YMIN,YMAX)
      CALL CEXP(YMIN,YMAX,MINEXP,MAXEXP)
      CALL CVLAB('P[BLC]OWER LEVEL',1)
      CALL CSET('NVLOGF',1)
      CALL CSET('NVLOGS',MINEXP)
```

```
      NTIC=(MAXEXP-MINEXP)+1
      CALL CVLOG(YLEN,NTIC,1)
C-----------------------------------------------------------------------
C GENERATE A GRID
      XSC=XLEN
      YSC=YLEN/REAL(NTIC-1)
      IXC=-1
      IYC=NTIC-1
      XGS=0.
      YGS=1.
      XGI=XLEN/REAL(NMAJOR-1)
      YGI=1.
      NBLANK=1
      CALL CLGRID(XSC,YSC,IXC,IYC,XGS,YGS,XGI,YGI,BLANK,NBLANK)
C-----------------------------------------------------------------------
C SAVE DATA REGION VIRTUAL COORDINATES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET VIEWPORT TO DATA REGION VIRTUAL COORDINATES
      CALL JVPORT(VX1,VX2,VY1,VY2)
C-----------------------------------------------------------------------
C SET UP DATA REGION WINDOW
      DO 1 KI=1,NPTS
 1    YTEMP(KI)=LOG10(Y(KI))
C X-AXIS IS LOG, MIN AND MAX MUST BE SCALED.
      CALL JWINDO(AMIN,AMAX,REAL(MINEXP),REAL(MAXEXP))
C ENABLE CLIPPING TO EXCLUDE EXTRANEOUS DATA
      CALL JWCLIP(.TRUE.)
      CALL JOPEN
C PLOT DATA
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,YTEMP,NPTS)
      CALL JPAUSE(IDEV)
      CALL JCLOSE
      CALL JFRAME
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
C-----------------------------------------------------------------------
      STOP
      END
```

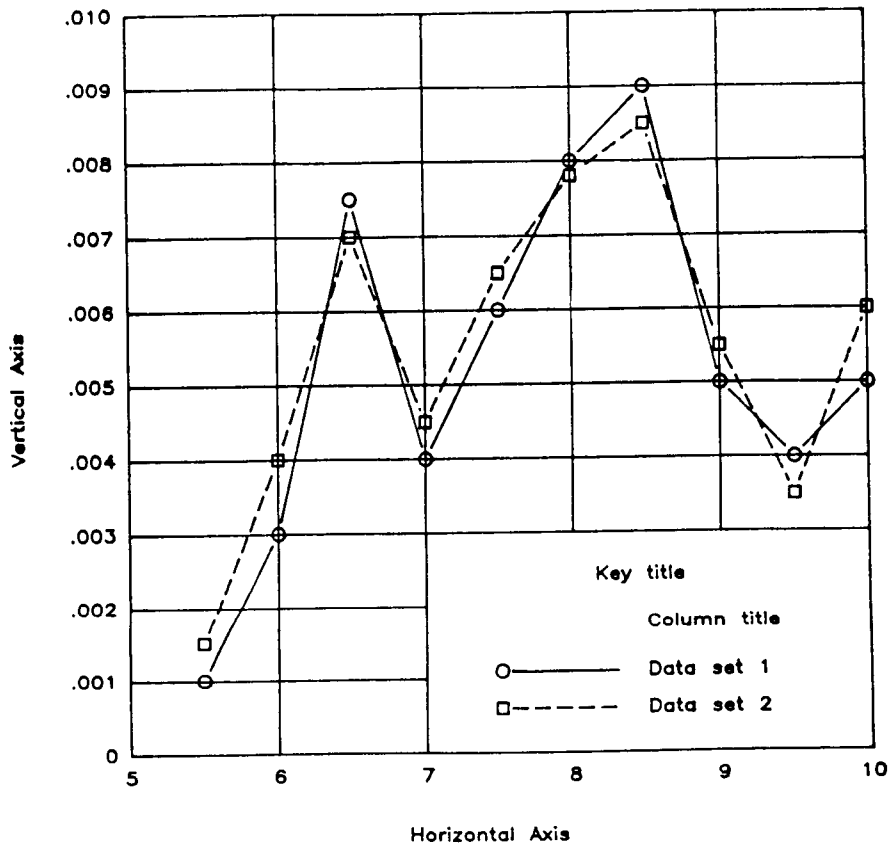


Figure CDR11. Complete low-level program (with a key) and a grid.

```
      PROGRAM CDR11
C-----------------------------------------------------------------------
C  COMPLETE LOW-LEVEL PROGRAM (WITH A KEY) AND A GRID
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4),BLANK(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.004,.005/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0035,.006/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,0.5)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=5.0
      CALL CHTICJ(0)
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=5.0
```

```
      CALL CVAXIS(.0,.01,.001,YLEN)
C-----------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 1')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ATA SET 2')
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND AND GRID WITH A HOLE.
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
C FIRST CALL CKEYPL WITH KPT SET TO 3 TO OBTAIN KEY EXTENTS.
      KPT=1
      BPOS(1)=2.0
      BPOS(2)=2.0
      BPOS(3)=3.0
      BPOS(4)=3.0
      CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE','C[BLC]OLUMN TITLE',
     +            BPOS,KPT,JCOL,1)
      XXTENT=BPOS(3)-BPOS(1)
      YXTENT=BPOS(4)-BPOS(2)
      WRITE(77,*)'XXTENT,YXTENT=',XXTENT,YXTENT
C NOW BASED ON EXTENTS AND AXES, DETERMINE WHERE TO PLACE HOLE FOR KEY.
C SET BPOS AND KPT ACCORDINGLY.
      XL=XLEN/5.
      YL=YLEN/10.
      BPOS(1)=(XORG+XLEN)-((XL*3.0-XXTENT)/2.)
      BPOS(2)=YORG+((YL*3.-YXTENT)/2.)
       KPT=3
      CALL CKEYPL(ISTORE,KEYCHR,'K[BLC]EY TITLE','C[BLC]OLUMN TITLE',
     +            BPOS,KPT,JCOL,1)
```

```
C NOW GENERATE GRID WITH APPROPRIATE HOLE.
      NOINCX=NINT((10.-5.)/1.)
      NOINCY=NINT((.01-.0)/.001)
      XS=XLEN/REAL(NOINCX)
      YS=YLEN/REAL(NOINCY)
      NBLANK=1
      BLANK(1)=XORG+XL*2.
      BLANK(3)=XORG+XLEN
      BLANK(2)=YORG
      BLANK(4)=YORG+YL*3.
      CALL CGRID(XORG,YORG,XS,YS,NOINCX,NOINCY,BLANK,NBLANK)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

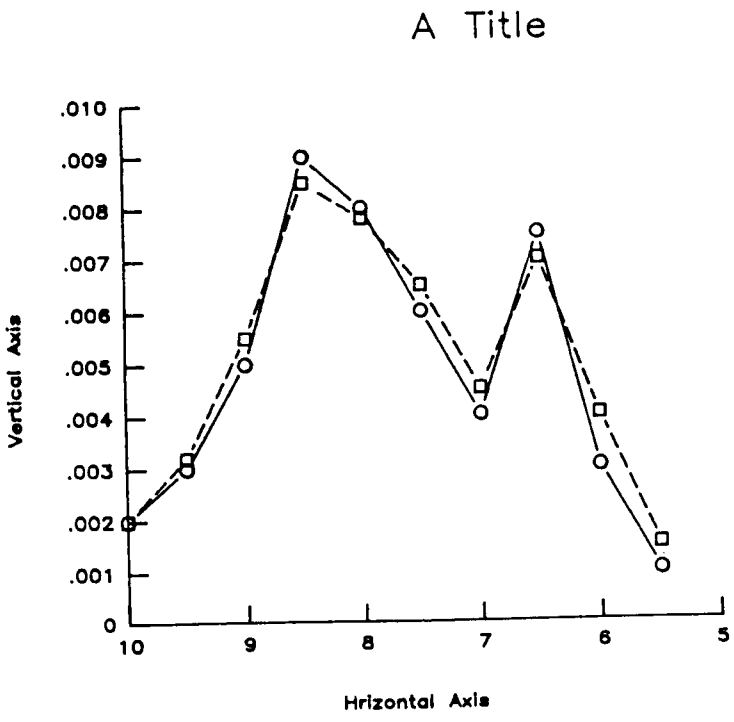


Figure CDR12. Line chart with a decreasing axis.

```
      PROGRAM CDR12
C-----------------------------------------------------------------------
C A LINE CHART WITH A DECREASING AXIS.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]RIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(10.,5.,-1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.001,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
```

```
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
C NOTE: THE WINDOW OF THE UNDERLYING GRAPHICS PACKAGE MUST BE
C       INCREASING IN NATURE.  THE RANGE MUST MATCH THE DATA.
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (XMAX-XMIN)/2. + XMIN
      REFLECT=(10. -  5.)/2. + 5.
      CALL CFLIP(X,MAXPTS,REFLECT)
      CALL JOPEN
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C IF THE X ARRAY IS TO BE USED AGAIN, THE USER MUST APPLY CFLIP AGAIN.
C IN THIS CASE, THE X ARRAY IS NOT USED AGAIN.
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

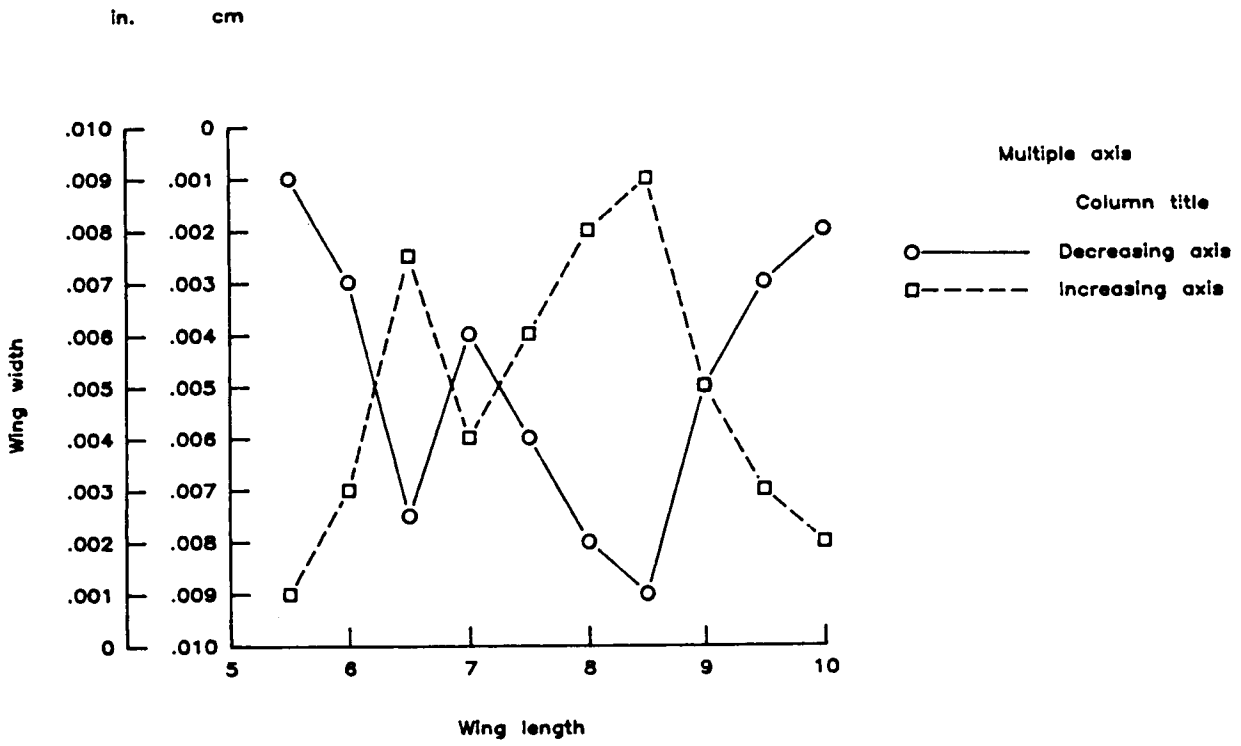
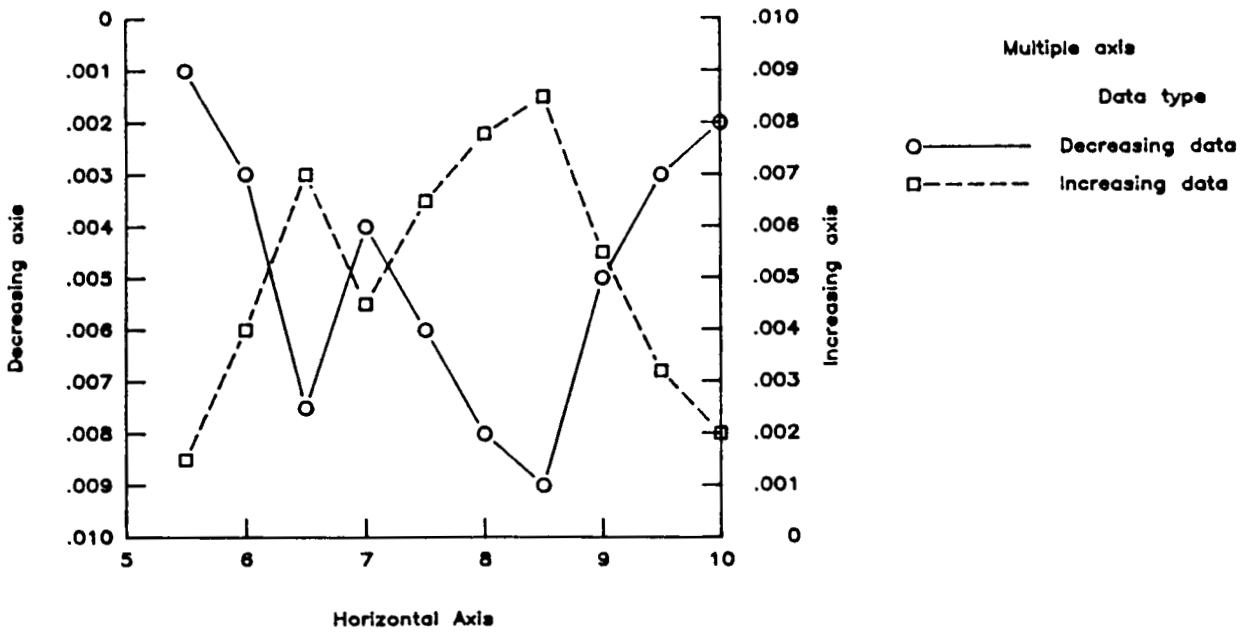


Figure CDR13. Line chart with multiple vertical scales on the same axis.

```
      PROGRAM CDR13
C-----------------------------------------------------------------------
C A LINE CHART WITH MULTIPLE VERTICAL SCALES ON THE SAME AXIS.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,.5)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-----------------------------------------------------------------------
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('W[BLC]ING LENGTH',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C-----------------------------------------------------------------------
C DESCRIBE THE VERTICAL AXIS
C NO AXIS LABEL ON INNERMOST VERTICAL AXIS
      CALL CVLAB(' ',0)
```

```
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
C PLOT FIRST AXIS (DECREASING)
      CALL CVAXIS(.01,.0,-.001,YLEN)
      CALL JMOVE(XORG,YORG+YLEN+.75)
      CALL JHSTRG('[BLC]CM')
C PLOT SECOND AXIS (INCREASING)
C POSITION AND DESCRIBE THE OUTMOST VERTICAL AXIS
      CALL JMOVE(XORG-CSIZE*7.,YORG)
      CALL CVLAB('W[BLC]ING WIDTH',1)
      CALL CVAXIS(.0,.01,.001,YLEN)
C LABEL FOR OUTMOST VERTICAL AXIS
      CALL JMOVE(XORG-CSIZE*7.,YORG+YLEN+.75)
      CALL JHSTRG('[BLC]IN.')
C-----------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (YMAX-YMIN)/2. + YMIN
      REFLECT=(.01 -  0.)/2. + 0.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C PLOT FIRST DATA CURVE WITH DECREASING AXIS (USING CGL DEFAULTS)
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ECREASING AXIS DATA SET 1')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C REVERSE DATA AGAIN, TO RETURN DATA TO ORIGINAL VALUES.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'I[BLC]NCREASING AXIS DATA SET 2')
C PLOT SECOND DATA CURVE WITH INCREASING AXIS.
      CALL CLNPLT(X,Y(1,1),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
```

```
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'C[BLC]OLUMN TITLE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```



A Title

Figure CDR14. Line chart with multiple vertical scales (opposite axis).

```
      PROGRAM CDR14
C-----------------------------------------------------------------------
C  LINE CHART WITH MULTIPLE VERTICAL SCALES (OPPOSITE AXIS)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET),BPOS(4)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA JCOL/1/
      DATA (Y(KI,1),KI=1,10)
     +  /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     +  /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,.5)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-----------------------------------------------------------------------
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=1.25
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRSENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C-----------------------------------------------------------------------
C DESCRIBE THE VERTICAL AXIS
C NO AXIS LABEL ON INNERMOST VERTICAL AXIS
      CALL CVLAB('D[BLC]ECREASING AXIS',1)
```

```
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
C PLOT FIRST AXIS (DECREASING)
      CALL CVAXIS(.01,.0,-.001,YLEN)
C PLOT SECOND AXIS (INCREASING)
C POSITION AND DESCRIBE THE OPPOSITE VERTICAL AXIS
      CALL JMOVE(XORG+XLEN,YORG)
C REQUEST VERTICAL AXIS TICK MARKS TO THE RIGHT OF THE AXIS.
C     CALL CVTICJ(2)
C REQUEST VERTICAL AXIS TICK MARK LABELS TO THE RIGHT OF THE AXIS.
      CALL CVLABJ(1)
      CALL CVTICJ(2)
      CALL CVLAB('I[BLC]NCREASING AXIS',1)
      CALL CVAXIS(.0,.01,.001,YLEN)
C-----------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10 -  REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C FLIP THE DATA ABOUT A REFLECTION POINT (I.E., MIRROR IMAGE).
C REFLECT =   (YMAX-YMIN)/2. + YMIN
      REFLECT=(.01 -  0.)/2. + 0.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C PLOT FIRST DATA CURVE WITH DECREASING AXIS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'D[BLC]ECREASING DATA')
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C REVERSE DATA AGAIN, TO RETURN DATA TO ORIGINAL VALUES.
      CALL CFLIP(Y(1,1),MAXPTS,REFLECT)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CKEYLB(ISTORE,KEYCHR,3,'I[BLC]NCREASING DATA')
C PLOT SECOND DATA CURVE WITH INCREASING AXIS.
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
```

```
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'D[BLC]ATA TYPE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

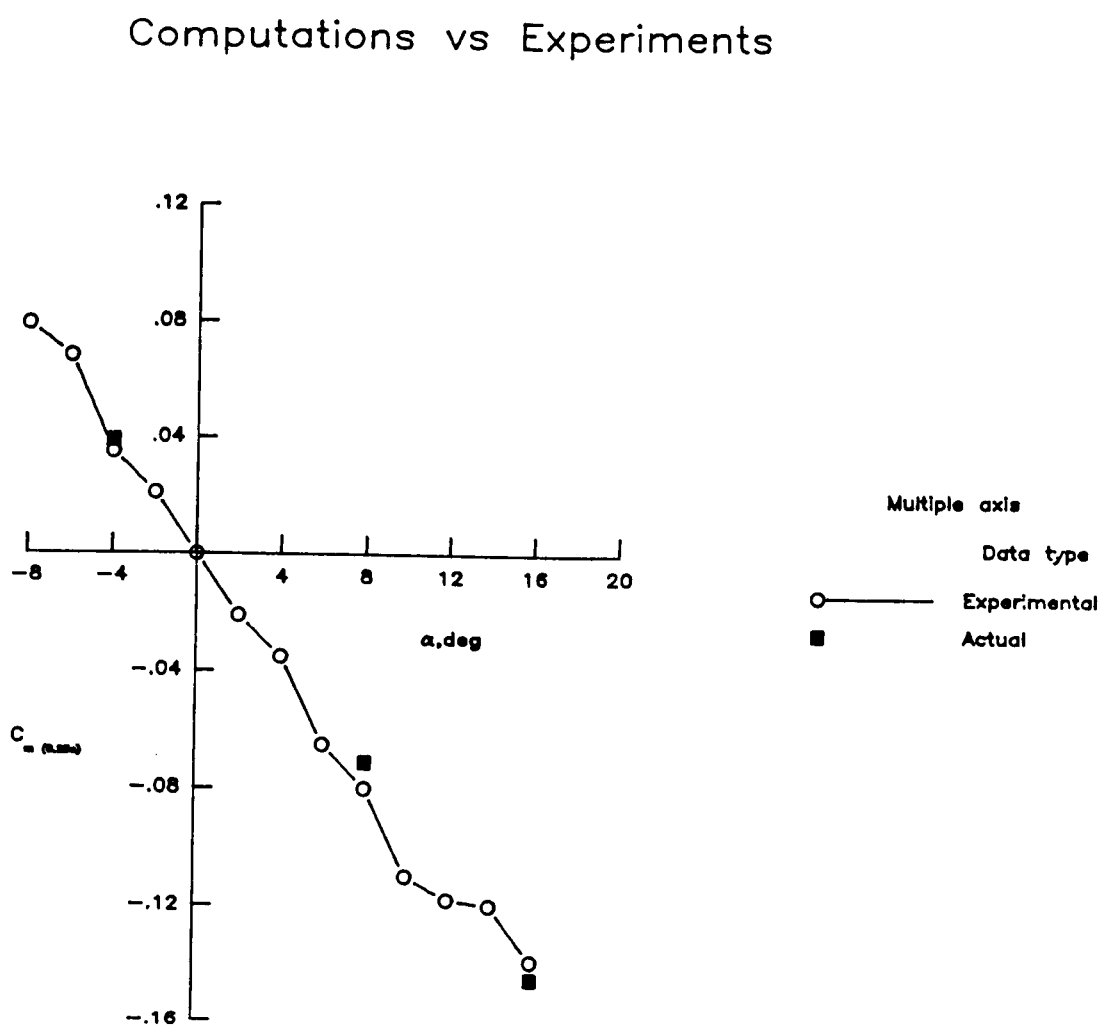


Figure CDR15. Simple, yet complete low-level program (with a key).

```
      PROGRAM CDR15
C-----------------------------------------------------------------------
C  SIMPLE, YET COMPLETE LOW-LEVEL PROGRAM (WITH A KEY)
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(NUMPT1=13,NUMPT2=3,MAXLAB=20)
      REAL X1(NUMPT1),Y1(NUMPT1),BPOS(4)
      REAL X2(NUMPT2),Y2(NUMPT2)
      PARAMETER (NLINS=2,NCOLS=1,NTLINS=1)
      CHARACTER KEYCHR(NCOLS,NLINS)*20,CLABS(MAXLAB)*10
      INTEGER ISTORE(15,NLINS),JCOL(NCOLS)
      DATA X1/-8.,-6.,-4.,-2.,0.,2.,4.,6.,8.,10.,12.,14.,16./
      DATA JCOL/1/
      DATA (Y1(KI),KI=1,NUMPT1)
     +  /.079,.068,.035,.021,.0,-.021,-.035,-.065,-.08,-.11,
     +   -.118,-.12,-.139/
      DATA X2/-4.,8.,16./,Y2/.039,-.071,-.145/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C-----------------------------------------------------------------------
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CSIZE=.2
      CALL JSIZE(CSIZE,CSIZE*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,TOP)
      CALL JMOVE(9./2.,9.-.5)
      CALL JJUST(2,3)
C OUTPUT THE STRING
      CALL JHSTRG('C[BLC]OMPUTATIONS VS[ELC] E[BLC]XPERIMENTS')
C RESET THE CHARACTER SIZE FOR THE AXES
      CSIZE=.1
      CALL JSIZE(CSIZE,CSIZE*1.25)
C-----------------------------------------------------------------------
C BASED AXES DATA ATTRIBUTES (MINIMUM, MAXIMUM, AND INCREMENTS).
      XMIN=-8.
      XMAX=20.
      XINC=4.
      YMIN=-.16
      YMAX=0.12
      YINC=.04
C-----------------------------------------------------------------------
C SET X AND Y AXIS LENGTHS (I.E., XLEN AND YLEN RESPECTIVELY).
      XLEN=4.0
      YLEN=5.5
C POSITION LOWER-LEFT CORNER OF DATA SPACE ON THE PAGE.
```

```
      XORG=((9./2.)-1.)-.5*XLEN
      YORG=(9./2.)-.5*YLEN
C NOW, SET XZERO AND YZERO PAGE COORDINATES WHERE AXES 0,0 EXISTS.
C (THIS PROGRAM ASSUMES THAT DATA ENCOMPASSES 0,0 IN X AND Y).
      XZERO=XORG+XLEN*(-XMIN/(XMAX-XMIN))
      YZERO=YORG+YLEN*(-YMIN/(YMAX-YMIN))
C-----------------------------------------------------------------------
C DESCRIBE X-AXIS. MOVE TO X ORIGIN, Y ZERO.
      CALL JMOVE(XORG,YZERO)
C GENERATE X AXIS LABELS.
      CALL CILAB(XMIN,XMAX,XINC,0,CLABS)
      ICOUNT=NINT((XMAX-XMIN)/XINC)+1
C BLANK OUT INTERSECTION LABEL (I.E., 0).
      KI=NINT(ABS(XMIN)/XINC)+1
      CLABS(KI)=' '
C DRAW AXIS
      CALL CHAXIC(XLEN,ICOUNT,CLABS,1,ICOUNT)
C POSITION AND OUTPUT AXIS LABEL
      CALL JMOVE(XORG+(5./7.)*XLEN,YZERO-.5)
      CALL JHSTRG('[FONT=9][BLC]A[FONT=3],DEG')
C-----------------------------------------------------------------------
C DESCRIBE Y-AXIS. MOVE TO X ZERO, Y ORIGIN.
      CALL JMOVE(XZERO,YORG)
C GENERATE Y AXIS LABELS.
      CALL CILAB(YMIN,YMAX,YINC,2,CLABS)
      ICOUNT=NINT((YMAX-YMIN)/YINC)+1
C BLANK OUT INTERSECTION LABEL (I.E., 0).
      KI=NINT(ABS(YMIN)/YINC)+1
      CLABS(KI)=' '
C DRAW AXIS.
      CALL CVAXIC(YLEN,ICOUNT,CLABS,1,ICOUNT)
C POSITION AND OUTPUT AXIS LABEL
      CALL JMOVE(XZERO-1.,YORG+(2.5/7.)*YLEN)
      CALL JHSTRG('C[BLC][BSUB]M (0.25C)')
C-----------------------------------------------------------------------
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(XMIN,XMAX,YMIN,YMAX)
C XMIN,XMAX - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE X-DIRECTION
C YMIN,YMAX - REPRESENTS THE DATA COORDINATE BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
      CALL CKEYIN(ISTORE,KEYCHR,NLINS,NCOLS,NTLINS)
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
C SET SYMBOL TO 1 (A CIRCLE) FOR EXPERIMENTAL DATA.
      CALL CSYMNO(1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'E[BLC]XPERIMENTAL')
      CALL CLNPLT(X1,Y1,NUMPT1)
```

```
C PLOT SECOND DATA CURVE.
C SET SYMBOL TO SOLID SQUARE (ACTUAL), AND PLOT SYMBOLS ONLY.
      CALL CSYMNO(902)
      CALL CPLTYP(-1)
      CALL CKEYLB(ISTORE,KEYCHR,3,'A[BLC]CTUAL')
      CALL CLNPLT(X2,Y2,NUMPT2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW OUTPUT LEGEND
C SET WINDOW AND VIEWPORT FOR PAGE COORDINATES
      CALL JVPORT(-1.,1.,-1.,1.)
      CALL JWINDO(0.,9.,0.,9.)
      BPOS(1)=9.
      BPOS(2)=2.+3.5
      CALL JOPEN
      CALL JSIZE(CSIZE,CSIZE*1.25)
      CALL CKEYPL(ISTORE,KEYCHR,'M[BLC]ULTIPLE AXIS',
     +            'D[BLC]ATA TYPE',BPOS,5,JCOL,1)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

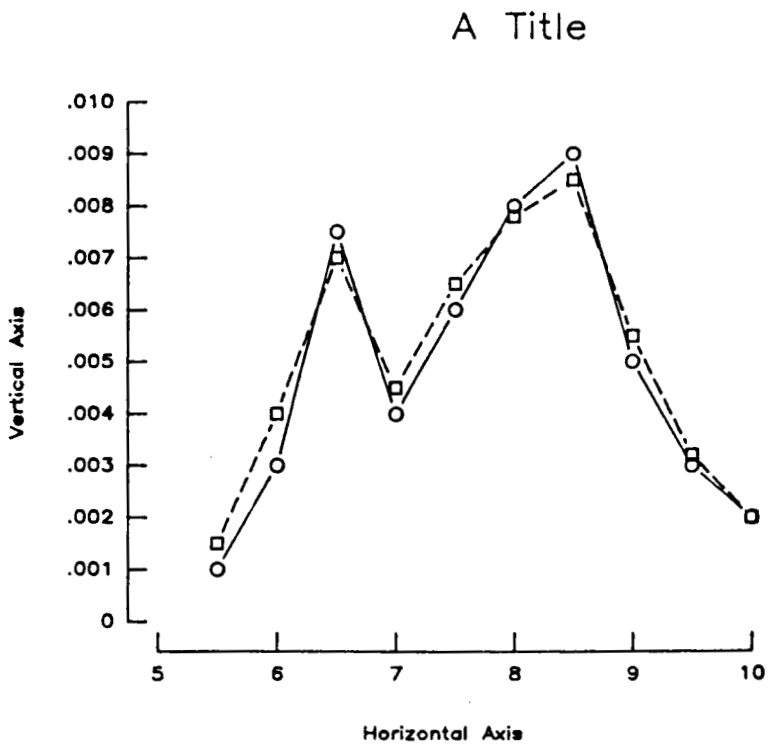


Figure CDR16. Line chart with disjoint axes (based on CDR4).

```
      PROGRAM CDR16
C-----------------------------------------------------------------------
C LINE CHART WITH DISJOINT AXES (BASED ON CDR6)
C-----------------------------------------------------------------------
C THE KEY ELEMENTS INVOLVING DISJOINT AXES INCLUDE:
C - THE POSITIONING OF THE AXES (SEE XOFF AND YOFF),
C - THE SETTING OF THE VIEWPORT TO MATCH THE DATA SPACE
C   (WHICH DIFFERS FROM THE AXES BOUNDARIES).
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER (MAXPTS=10,MAXSET=2)
      REAL X(MAXPTS),Y(MAXPTS,MAXSET)
      DATA X/5.5,6.0,6.5,7.0,7.5,8.0,8.5,9.0,9.5,10./
      DATA (Y(KI,1),KI=1,10)
     + /.001,.003,.0075,.004,.006,.008,.009,.005,.003,.002/
      DATA (Y(KI,2),KI=1,10)
     + /.0015,.004,.007,.0045,.0065,.0078,.0085,.0055,.0032,.002/
C-----------------------------------------------------------------------
C WRITE TO THE DEVICE IDEV
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION STRING, AND SET JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
C XOFF - THE DISTANCE TO SHIFT THE X-AXIS DOWN.
      XOFF=.2
      CALL JMOVE(2.,2.-(XOFF))
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHAXIS(5.,10.,1.,4.)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YOFF - THE DISTANCE TO SHIFT THE Y-AXIS LEFT.
      YOFF=.2
      CALL JMOVE(2.-YOFF,2.)
      CALL CVAXIS(.0,.01,.001,3.5)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
C NOTE: THE VIRTUAL COORDINATES ARE BASED ON DIMENSIONS OF DATA
```

```
C       SPACE, NOT ON AXIS BOUNDARIES.
      CALL JCONWV(2.,2.,0.,VX1,VY1)
      CALL JCONWV(2.+4.,2.+3.5,0.,VX2,VY2)
C CLOSE CURRENT WORLD COORDINATES (PAGE COORDINATES)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDINATES, AND PLOT WITHIN BOUNDARIES
C OF THE AXES (BY SAVED VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT FIRST DATA CURVE USING CGL DEFAULTS
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y(1,1),MAXPTS)
C SET SYMBOL NUMBER TO 2, AND PLOT SECOND DATA CURVE
      CALL CSYMNO(2)
      CALL CLNPAT(2)
      CALL CLNPLT(X,Y(1,2),MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
C TERMINATE GRAPHICS
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

CHTICJ: IMPROPER ENTRY GIVEN FOR POSITIONING OF TICKMARKS. VALUE MUST BE 0, 1, 2, OR 3. DEFAULT VALUE OF 1 IS USED.

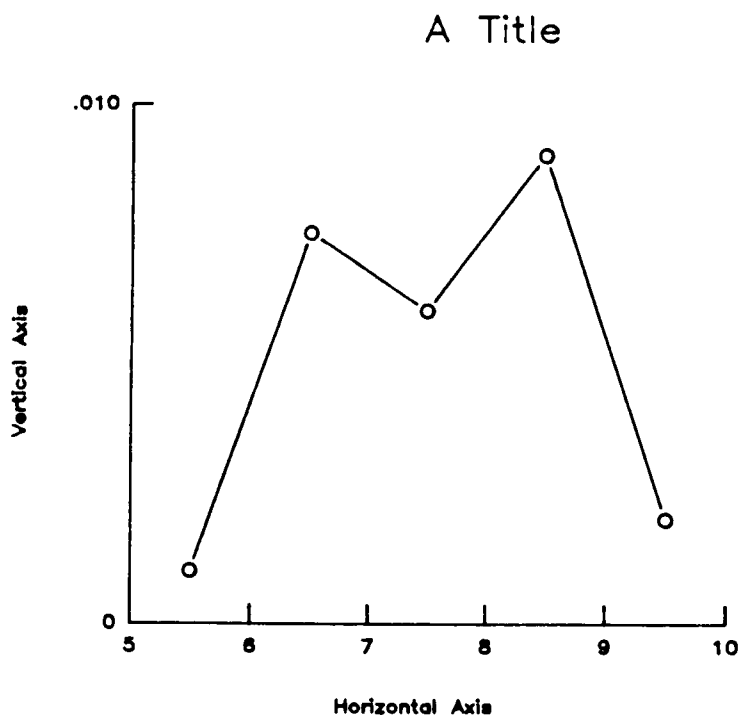


Figure CDD1. Example showing a simple CGL error.

```
      PROGRAM CDD1
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A SIMPLE CGL ERROR.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHTICJ(-1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.01,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDIATES, AND PLOT WITHIN
C BOUNDARIES OF THE AXES (BY SAVE VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10  - REPRESENTS THE DATA BOUNDARIES IN THE X-DIRECTION
```

```
C 0,.01 - REPRESENTS THE DATA BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

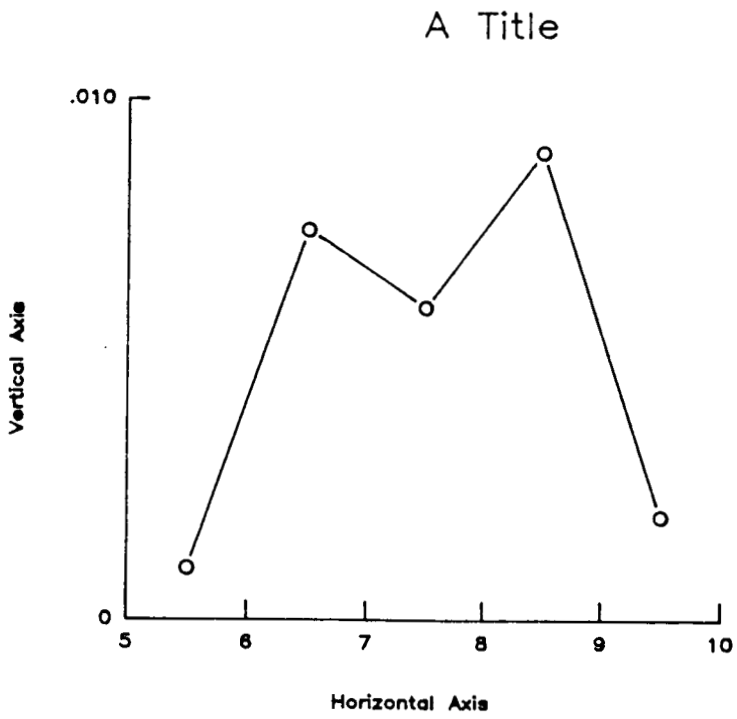


Figure CDD2. Example showing debugging capabilities.

```
      PROGRAM CDD2
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A DEBUGGING CAPABILITIES.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C DEBUGGING TURNED ON FOR USER-CALLABLE ROUTINES.
      CALL CDEBUG(1)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.01,YLEN)
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDIATES, AND PLOT WITHIN
C BOUNDARIES OF THE AXES (BY SAVE VIRTUAL COORDINATES)
```

```
      CALL JWINDO(5.,10.,.0,.01)
C 5,10  - REPRESENTS THE DATA BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

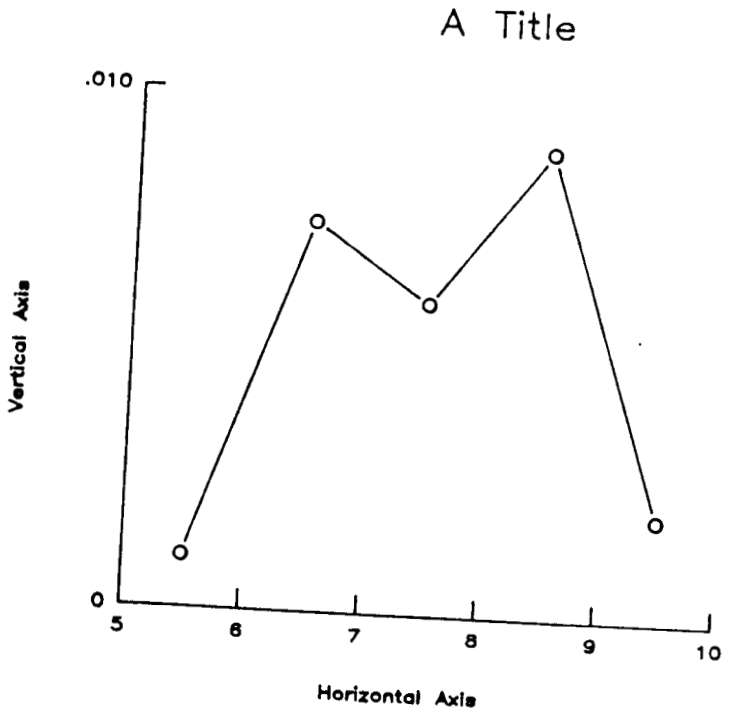


Figure CDD3. Example showing an error and debugging redirection.

```
      PROGRAM CDD3
C-----------------------------------------------------------------------
C AN EXAMPLE SHOWING A ERROR AND DEBUGGING REDIRECTION.
C-----------------------------------------------------------------------
C ALLOCATE AND INITIALIZE DATA
      PARAMETER(MAXPTS=5)
      REAL X(MAXPTS),Y(MAXPTS)
      DATA X/5.5,6.5,7.5,8.5,9.5/,Y/.001,.0075,.006,.009,.002/
C-----------------------------------------------------------------------
C WRITE TO THE METAFILE
      IDEV=0
      CALL CBEGIN
      CALL JBEGIN
      CALL JDINIT(IDEV)
      CALL JDEVON(IDEV)
C-----------------------------------------------------------------------
C WRITE CGL ERROR MESSAGES TO UNIT 80, FILE 'TAPE80'.
      CALL CFILES(1,80,'TAPE80')
C-----------------------------------------------------------------------
C WRITE CGL DEBUG MESSAGES TO UNIT 80, FILE 'TAPE80'.
      CALL CFILES(2,80,'TAPE80')
C-----------------------------------------------------------------------
C DEBUGGING TURNED ON FOR USER-CALLABLE ROUTINES.
      CALL CDEBUG(1)
C-----------------------------------------------------------------------
C ESTABLISH THE PAGE COORDINATES AND VIEWSPACE
      CALL CVSPAC(9.,9.)
      CALL JWINDO(0.,9.,0.,9.)
      CALL JOPEN
C SET THE CHARACTER SIZE (IN TERMS OF PAGE COORDINATES)
      CALL JSIZE(.2,.2*1.25)
C POSITION TEXT, AND SET TEXT JUSTIFICATION (CENTER,CENTER)
      CALL JMOVE(4.5,6.0)
      CALL JJUST(2,2)
C OUTPUT THE STRING
      CALL JHSTRG('A T[BLC]ITLE')
C RESET THE CHARACTER SIZE FOR THE AXES
      CALL JSIZE(.1,.1*1.25)
C POSITION THE LOWER-LEFT INTERSECTION OF THE AXES
      XORG=2.
      YORG=2.
      CALL JMOVE(XORG,YORG)
C DESCRIBE THE HORIZONTAL AXIS
      CALL CHLAB('H[BLC]ORIZONTAL [BUC]A[BLC]XIS',1)
      CALL CHTICJ(-1)
C XLEN - REPRESENTS THE X-AXIS LENGTH
      XLEN=4.0
      CALL CHAXIS(5.,10.,1.,XLEN)
C DESCRIBE THE VERTICAL AXIS
      CALL CVLAB('V[BLC]ERTICAL [BUC]A[BLC]XIS',1)
      CALL CVPREC(3)
C YLEN - REPRESENTS THE Y-AXIS LENGTH
      YLEN=3.5
      CALL CVAXIS(.0,.01,.01,YLEN)
```

```
C SAVE VIRTUAL COORDINATES OF AXES BOUNDARIES
      CALL JCONWV(XORG,YORG,0.,VX1,VY1)
      CALL JCONWV(XORG+XLEN,YORG+YLEN,0.,VX2,VY2)
      CALL JCLOSE
C-----------------------------------------------------------------------
C SET WINDOW TO MATCH DATA COORDIATES, AND PLOT WITHIN
C BOUNDARIES OF THE AXES (BY SAVE VIRTUAL COORDINATES)
      CALL JWINDO(5.,10.,.0,.01)
C 5,10  - REPRESENTS THE DATA BOUNDARIES IN THE X-DIRECTION
C 0,.01 - REPRESENTS THE DATA BOUNDARIES IN THE Y-DIRECTION
      CALL JVPORT(VX1,VX2,VY1,VY2)
      CALL JOPEN
C PLOT DATA CURVE
      CALL CSYMNO(1)
      CALL CLNPAT(1)
      CALL CLNPLT(X,Y,MAXPTS)
      CALL JCLOSE
C-----------------------------------------------------------------------
      CALL JPAUSE(IDEV)
      CALL JFRAME
      CALL JDEVOF(IDEV)
      CALL JDEND(IDEV)
      CALL JEND
      STOP
      END
```

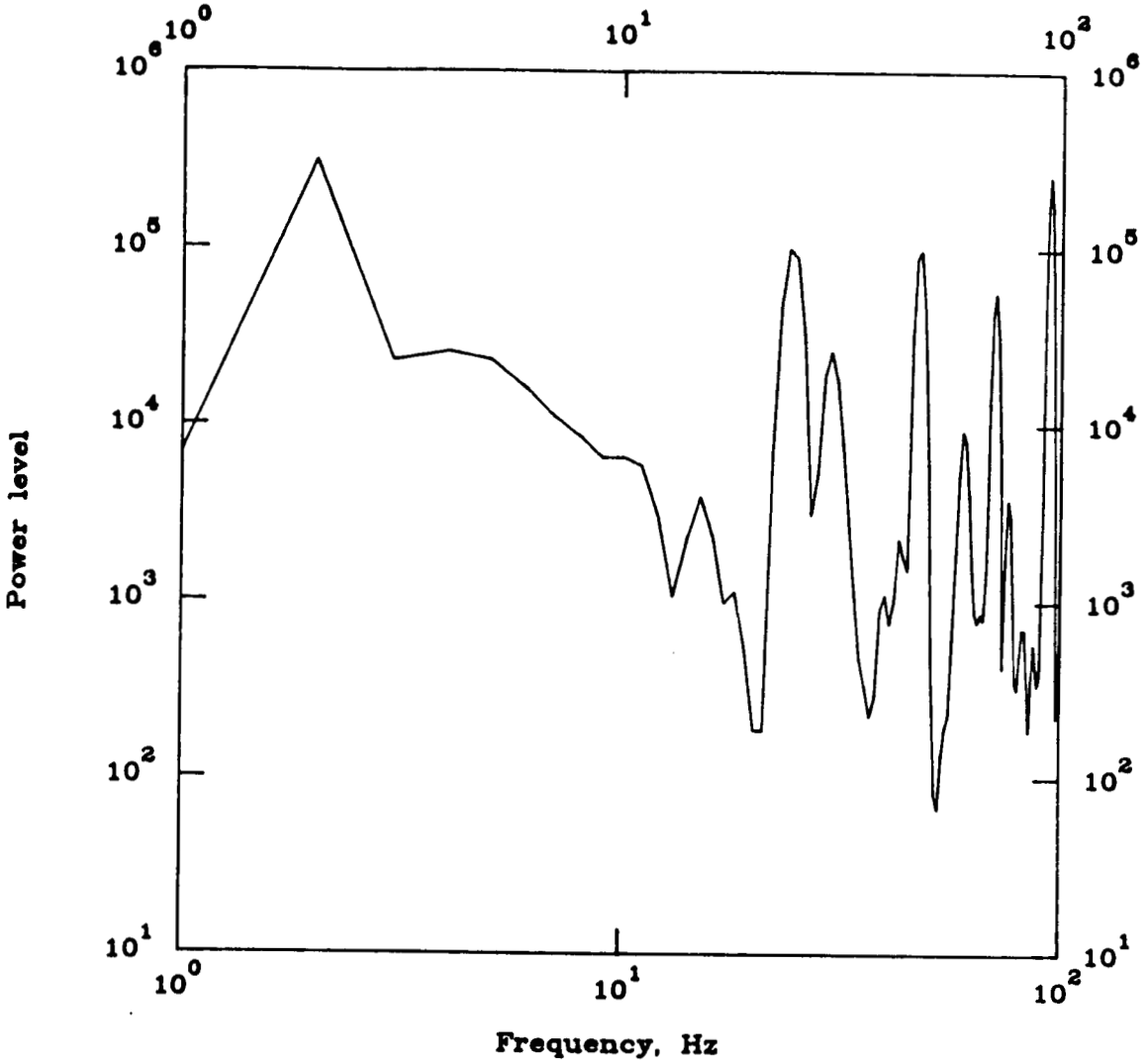


Figure CDLL1. Program interfacing the low-level routines with the LEZ Routines.

```
      PROGRAM CDLL1
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
     + 23314.47811626, 16222.72126064, 11123.46224798, 8573.673551723,
     + 6533.280733426, 6567.484820724, 5931.92717933,  3100.185764845,
     + 1075.542233512, 2304.552081512, 3938.878376374, 2372.668800947,
     + 990.5827153465, 1139.692923046, 539.1035104119, 186.2399260743,
     + 185.9121866205, 6922.275564162, 47871.80671708, 101610.8109471,
     + 90151.05656181, 32565.81919631, 3126.25985372,  5378.883275207,
     + 19139.27829711, 26311.51279586, 18051.10302768, 7595.925085295,
     + 2769.064915013, 1114.906853713, 472.3435990609, 336.5553643497,
     + 224.6112529072, 292.8078714839, 913.1650006148, 1111.290988571,
     + 763.2538952391, 1036.757022708, 2318.840862193, 1815.115702427,
     + 1546.468077257, 30998.79018191, 90446.39960755, 99542.65318885,
     + 46517.56984878, 8264.407149583, 477.4873134733, 82.183079214,
     + 67.7501808438,  136.6466583528, 197.6147132091, 229.3505457034,
     + 643.2165902202, 1824.921600301, 5246.018888617, 9355.942325835,
     + 8240.202957789, 4042.516486596, 1615.454279283, 876.7523693048,
     + 780.321551955,  869.2380712535, 806.9102907816, 1417.485863451,
     + 13314.08558278, 42705.82833591, 56933.22987158, 34522.64829539,
     + 8369.70034301,  423.4410260173, 1721.308093883, 3815.021552281/
      DATA (POWER(I),I=77,100)/
     + 3096.896818539, 1130.33337685,  351.6629052997, 323.9146852672,
     + 496.8216394068, 716.3789266813, 707.1830537755, 395.452882063,
     + 185.9595078555, 349.7278623283, 579.1256929328, 427.7957796537,
     + 339.4866401359, 404.7726592585, 3303.494231505, 44732.3344563,
     + 168132.2764905, 259620.2781873, 179647.069922,  51634.19154093,
     + 4701.338454798, 220.74733211,   377.2257021375, 552.6383507909/
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
      IAXTYP=3
      IHLOGI=0
```

```
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDLL1',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
      CALL JCLOSE
C-----------------------------------------------------------------------
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```

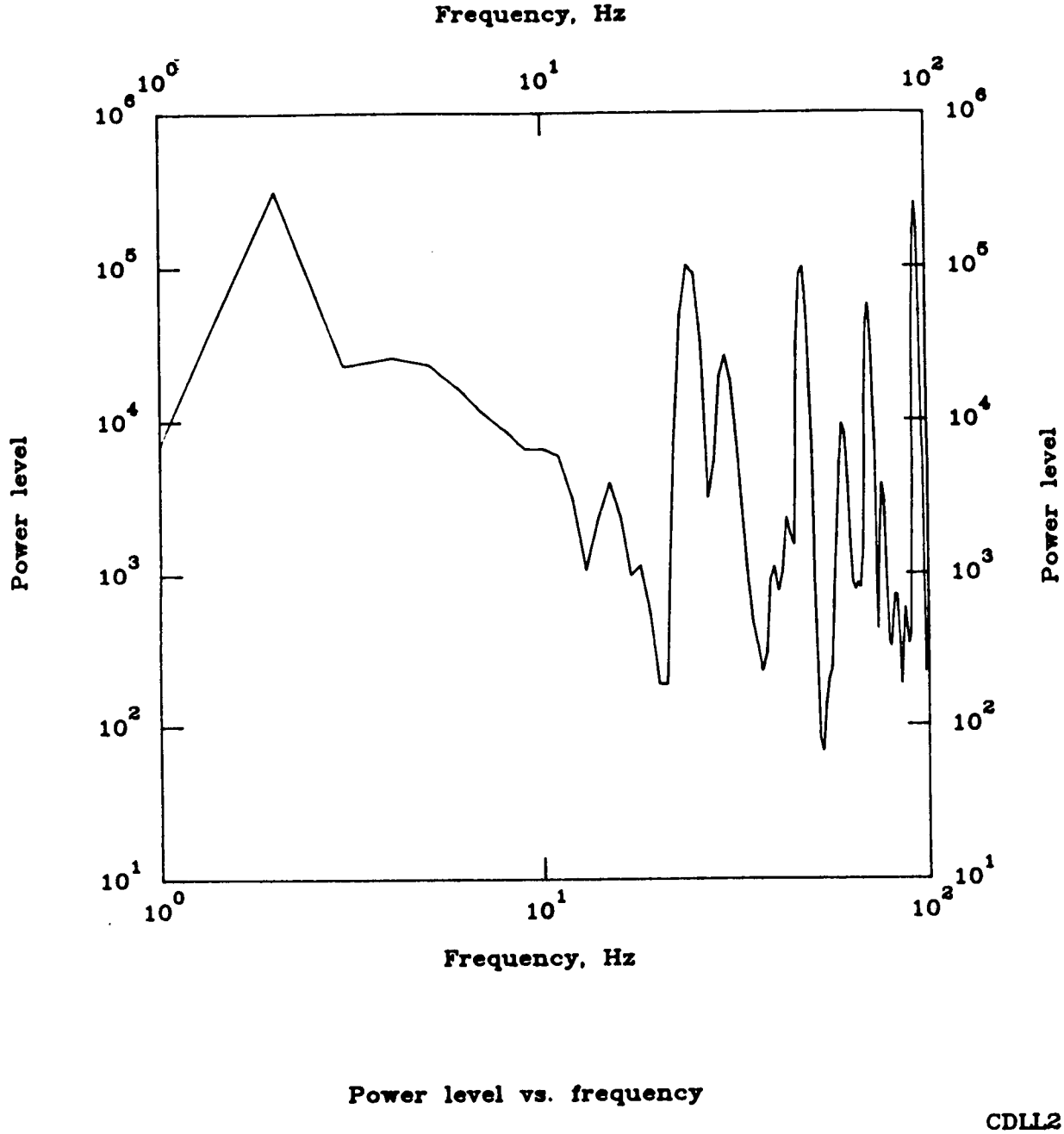


Figure CDLL2. Program interfacing the low-level routines with the LEZ Routines.

```
      PROGRAM CDLL2
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
     + 23314.47811626, 16222.72126064, 11123.46224798, 8573.673551723,
     + 6533.280733426, 6567.484820724, 5931.92717933,  3100.185764845,
     + 1075.542233512, 2304.552081512, 3938.878376374, 2372.668800947,
     + 990.5827153465, 1139.692923046, 539.1035104119, 186.2399260743,
     + 185.9121866205, 6922.275564162, 47871.80671708, 101610.8109471,
     + 90151.05656181, 32565.81919631, 3126.25985372,  5378.883275207,
     + 19139.27829711, 26311.51279586, 18051.10302768, 7595.925085295,
     + 2769.064915013, 1114.906853713, 472.3435990609, 336.5553643497,
     + 224.6112529072, 292.8078714839, 913.1650006148, 1111.290988571,
     + 763.2538952391, 1036.757022708, 2318.840862193, 1815.115702427,
     + 1546.468077257, 30998.79018191, 90446.39960755, 99542.65318885,
     + 46517.56984878, 8264.407149583, 477.4873134733, 82.183079214,
     + 67.7501808438,  136.6466583528, 197.6147132091, 229.3505457034,
     + 643.2165902202, 1824.921600301, 5246.018888617, 9355.942325835,
     + 8240.202957789, 4042.516486596, 1615.454279283, 876.7523693048,
     + 780.321551955,  869.2380712535, 806.9102907816, 1417.485863451,
     + 13314.08558278, 42705.82833591, 56933.22987158, 34522.64829539,
     + 8369.70034301,  423.4410260173, 1721.308093883, 3815.021552281/
      DATA (POWER(I),I=77,100)/
     + 3096.896818539, 1130.33337685,  351.6629052997, 323.9146852672,
     + 496.8216394068, 716.3789266813, 707.1830537755, 395.452882063,
     + 185.9595078555, 349.7278623283, 579.1256929328, 427.7957796537,
     + 339.4866401359, 404.7726592585, 3303.494231505, 44732.3344563,
     + 168132.2764905, 259620.2781873, 179647.069922,  51634.19154093,
     + 4701.338454798, 220.74733211,   377.2257021375, 552.6383507909/
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
      IAXTYP=3
      IHLOGI=0
```

```
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDLL2',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C SUPPRESS THE LEZ DEFAULTS OF PAUSE AND CLEAR
      CALL LEZOPT('PAUSE',.FALSE.)
      CALL LEZOPT('CLEAR',.FALSE.)
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
      CALL JCLOSE
C-----------------------------------------------------------------------
C NOW PAUSE AND CLEAR
      IF(IDEV.GE.1)CALL JPAUSE(1)
      CALL JFRAME
```

```
C-----------------------------------------------------------------------
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```

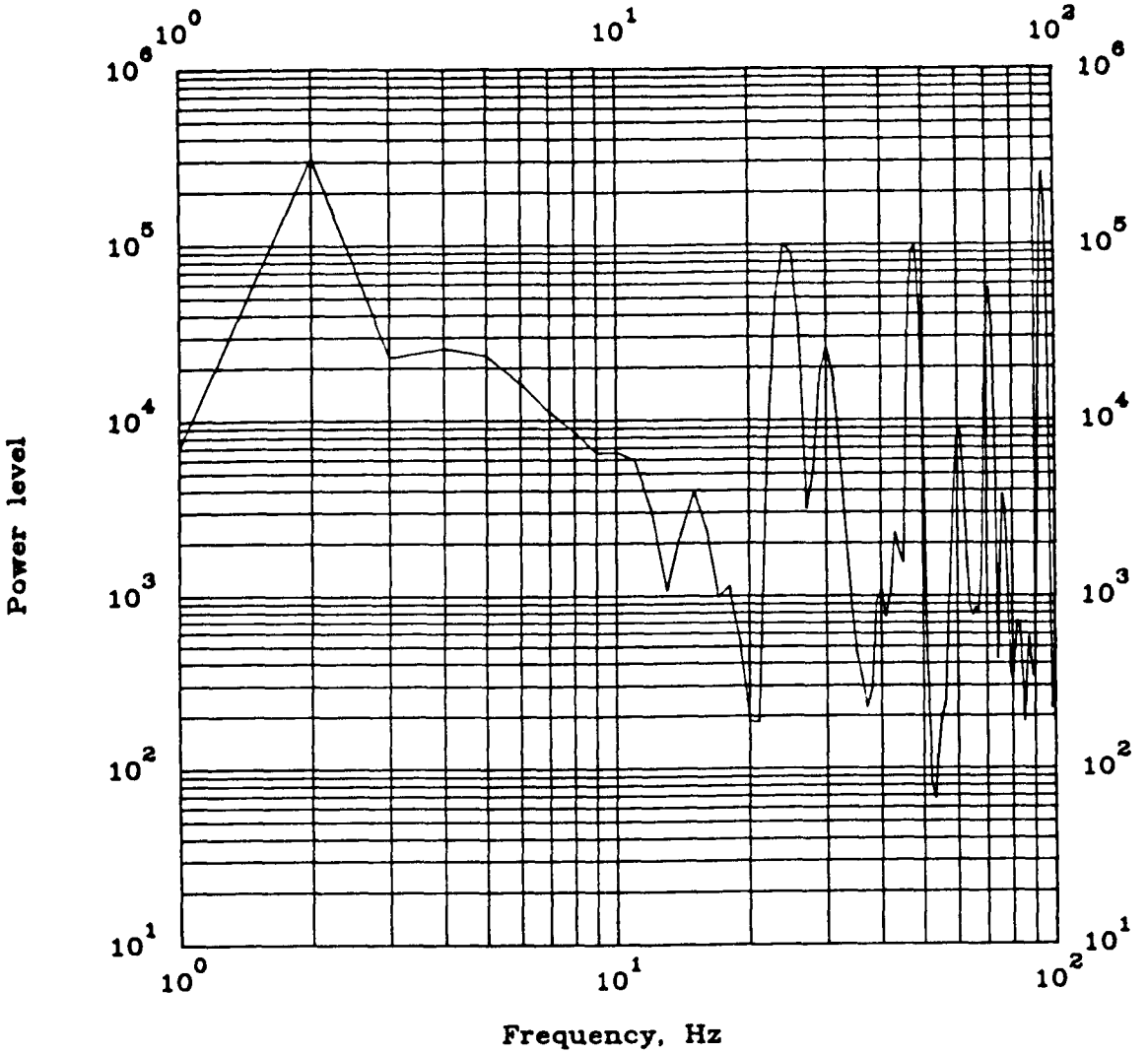
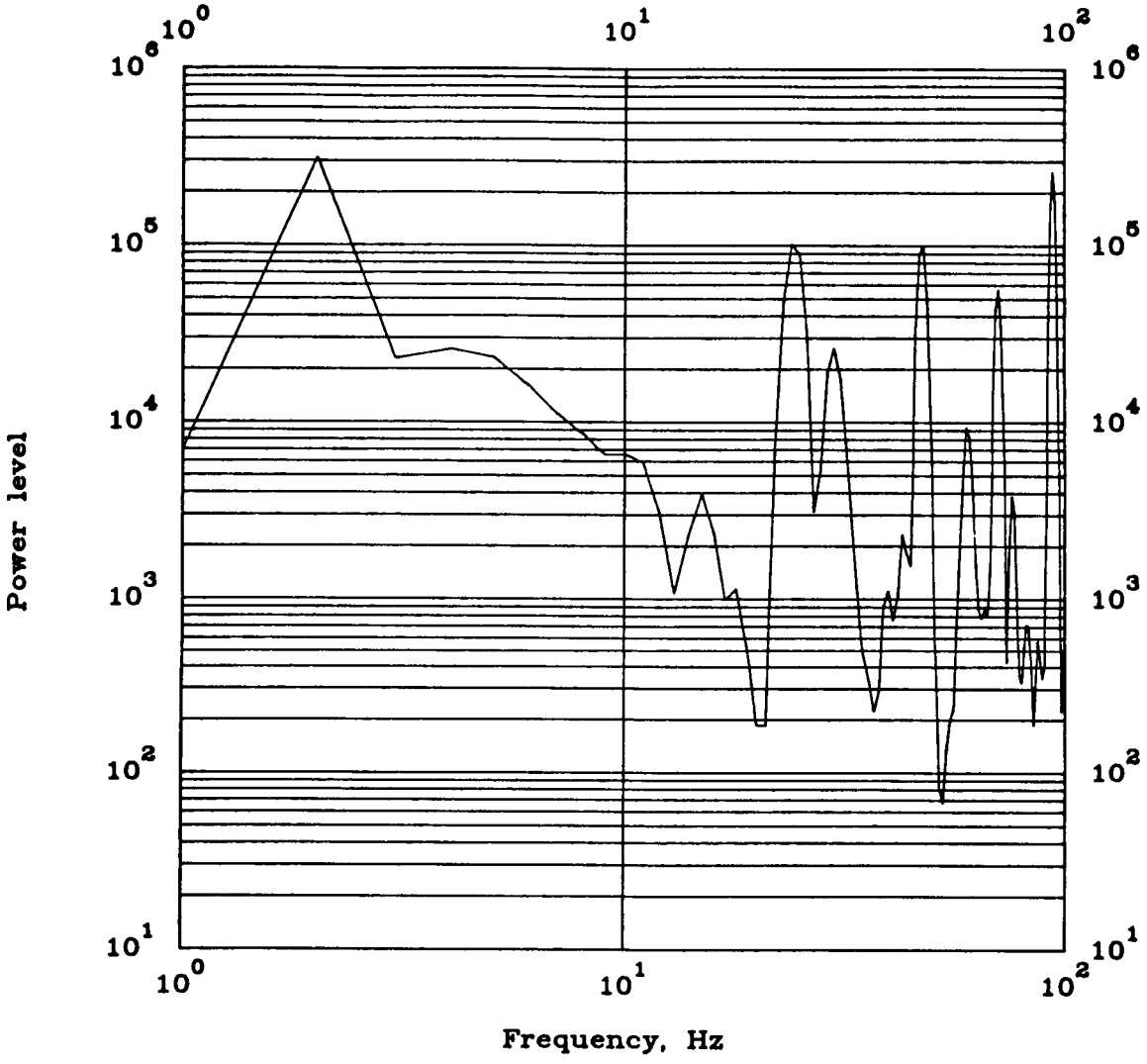


Figure CDLL3. Program interfacing the low-level routines with the LEZ Routines (this program is CDLL1 modified to CALL CLGRID(log-log)).

```
      PROGRAM CDLL3
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C NOTE: THIS PROGRAM IS CDLL1 MODIFIED TO CALL CLGRID (LOG-LOG).
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX,BLANK(4)
      DATA BLANK/0.,0.,0.,0./
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
     + 23314.47811626, 16222.72126064, 11123.46224798, 8573.673551723,
     + 6533.280733426, 6567.484820724, 5931.92717933,  3100.185764845,
     + 1075.542233512, 2304.552081512, 3938.878376374, 2372.668800947,
     + 990.5827153465, 1139.692923046, 539.1035104119, 186.2399260743,
     + 185.9121866205, 6922.275564162, 47871.80671708, 101610.8109471,
     + 90151.05656181, 32565.81919631, 3126.25985372,  5378.883275207,
     + 19139.27829711, 26311.51279586, 18051.10302768, 7595.925085295,
     + 2769.064915013, 1114.906853713, 472.3435990609, 336.5553643497,
     + 224.6112529072, 292.8078714839, 913.1650006148, 1111.290988571,
     + 763.2538952391, 1036.757022708, 2318.840862193, 1815.115702427,
     + 1546.468077257, 30998.79018191, 90446.39960755, 99542.65318885,
     + 46517.56984878, 8264.407149583, 477.4873134733, 82.183079214,
     + 67.7501808438,  136.6466583528, 197.6147132091, 229.3505457034,
     + 643.2165902202, 1824.921600301, 5246.018888617, 9355.942325835,
     + 8240.202957789, 4042.516486596, 1615.454279283, 876.7523693048,
     + 780.321551955,  869.2380712535, 806.9102907816, 1417.485863451,
     + 13314.08558278, 42705.82833591, 56933.22987158, 34522.64829539,
     + 8369.70034301,  423.4410260173, 1721.308093883, 3815.021552281/
      DATA (POWER(I),I=77,100)/
     + 3096.896818539, 1130.33337685,  351.6629052997, 323.9146852672,
     + 496.8216394068, 716.3789266813, 707.1830537755, 395.452882063,
     + 185.9595078555, 349.7278623283, 579.1256929328, 427.7957796537,
     + 339.4866401359, 404.7726592585, 3303.494231505, 44732.3344563,
     + 168132.2764905, 259620.2781873, 179647.069922,  51634.19154093,
     + 4701.338454798, 220.74733211,   377.2257021375, 552.6383507909/
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
```

```
      IAXTYP=3
      IHLOGI=0
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDLL3',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
C-----------------------------------------------------------------------
C CALL CLGRID
      CALL JMOVE(XOR,YOR)
      IXC=NTICH-1
      IYC=NTICV-1
      XSC=XLEN/REAL(IXC)
      YSC=YLEN/REAL(IYC)
      XGS=1.
      YGS=1.
```

```
      XGI=1.
      YGI=1.
      NBLANK=1
      CALL CLGRID(XSC,YSC,IXC,IYC,XGS,YGS,XGI,YGI,BLANK,NBLANK)
      CALL JCLOSE
C-----------------------------------------------------------------------
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```



Power level vs. frequency

CDL5E4

Figure CDLL4. Program interfacing the low-level routines with the LEZ Routines (this program is CDLL3 modified to CALL CLGRID(semi-log)).

C-5

```
      PROGRAM CDLL4
C-----------------------------------------------------------------------
C PROGRAM INTERFACING THE LOW-LEVEL ROUTINES WITH THE LEZ ROUTINES.
C NOTE: THIS PROGRAM IS CDLL3 MODIFIED TO CALL CLGRID (SEMI-LOG).
C-----------------------------------------------------------------------
      CHARACTER TITLE*80,IAXLBL*80,DAXLBL*80
      REAL FREQ(100),POWER(100),INDMIN,INDMAX,BLANK(4)
      DATA BLANK/0.,0.,0.,0./
      DATA (POWER(I),I=1,76)/
     + 7066.0770323,   313987.5856597, 23005.34014683, 25971.13637497,
     + 23314.47811626, 16222.72126064, 11123.46224798, 8573.673551723,
     + 6533.280733426, 6567.484820724, 5931.92717933,  3100.185764845,
     + 1075.542233512, 2304.552081512, 3938.878376374, 2372.668800947,
     + 990.5827153465, 1139.692923046, 539.1035104119, 186.2399260743,
     + 185.9121866205, 6922.275564162, 47871.80671708, 101610.8109471,
     + 90151.05656181, 32565.81919631, 3126.25985372,  5378.883275207,
     + 19139.27829711, 26311.51279586, 18051.10302768, 7595.925085295,
     + 2769.064915013, 1114.906853713, 472.3435990609, 336.5553643497,
     + 224.6112529072, 292.8078714839, 913.1650006148, 1111.290988571,
     + 763.2538952391, 1036.757022708, 2318.840862193, 1815.115702427,
     + 1546.468077257, 30998.79018191, 90446.39960755, 99542.65318885,
     + 46517.56984878, 8264.407149583, 477.4873134733, 82.183079214,
     + 67.7501808438,  136.6466583528, 197.6147132091, 229.3505457034,
     + 643.2165902202, 1824.921600301, 5246.018888617, 9355.942325835,
     + 8240.202957789, 4042.516486596, 1615.454279283, 876.7523693048,
     + 780.321551955,  869.2380712535, 806.9102907816, 1417.485863451,
     + 13314.08558278, 42705.82833591, 56933.22987158, 34522.64829539,
     + 8369.70034301,  423.4410260173, 1721.308093883, 3815.021552281/
      DATA (POWER(I),I=77,100)/
     + 3096.896818539, 1130.33337685,  351.6629052997, 323.9146852672,
     + 496.8216394068, 716.3789266813, 707.1830537755, 395.452882063,
     + 185.9595078555, 349.7278623283, 579.1256929328, 427.7957796537,
     + 339.4866401359, 404.7726592585, 3303.494231505, 44732.3344563,
     + 168132.2764905, 259620.2781873, 179647.069922,  51634.19154093,
     + 4701.338454798, 220.74733211,   377.2257021375, 552.6383507909/
C-----------------------------------------------------------------------
C CONVERT DATA TO BE REPRESENTED AS LOGARITHMIC
      DO 1 I=1,100
      FREQ(I)=LOG10(REAL(I))
1     POWER(I)=LOG10(POWER(I))
C-----------------------------------------------------------------------
C INITIALIZE GRAPHICS PACKAGE, THEN INITIALIZE AND SELECT DISPLAY DEVICE.
      IDEV=0
      CALL LEZINI(IDEV)
C-----------------------------------------------------------------------
C SET LEZ ATTRIBUTES
      NVALS=100
      TITLE='P[BLC]OWER LEVEL VS. FREQUENCY'
      IAXLBL='F[BLC]REQUENCY, [ELC]H[BLC]Z'
      DAXLBL='P[BLC]OWER LEVEL'
      INDMIN=LOG10(1.)
      INDMAX=LOG10(100.0)
      DEPMIN=LOG10(65.0)
      DEPMAX=LOG10(290000.0)
```

```
      IAXTYP=3
      IHLOGI=0
      IVLOGI=0
      INDHV=1
      LINAPP=1
C-----------------------------------------------------------------------
C CALL LEZLOG (SET LEZ LOG ATTRIBUTES)
      CALL LEZLOG(FREQ,POWER,NVALS,TITLE,IAXLBL,DAXLBL,INDMIN,INDMAX,
     +DEPMIN,DEPMAX,IAXTYP,IHLOGI,IVLOGI,INDHV,LINAPP)
C OUTPUT A NOTE ON THE DISPLAY DEVICE
      CALL LEZNOT('CDL5E4',7.,0.,3,1,0)
C-----------------------------------------------------------------------
C USE LOW-LEVEL ROUTINES TO ADD EXTRA AXES.
C SET WINDOW AND VIEWPORT TO MATCH LEZ'S DEFAULTS.
      CALL JWINDO(0.,7.,0.,7.)
      CALL JVPORT(-1.,1.,-1.,1.)
C GET CURRENT LEZ AXES VALUES
      CALL LEZGET('XORIGIN',XOR,NVALS,MVALS)
      CALL LEZGET('HAXLEN',XLEN,NVALS,MVALS)
      CALL LEZGET('YORIGIN',YOR,NVALS,MVALS)
      CALL LEZGET('VAXLEN',YLEN,NVALS,MVALS)
      CALL JOPEN
C HORIZONTAL AXIS
      CALL JMOVE(XOR,YOR+YLEN)
C SEE INDMIN,INDMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(1.,100.,MINH,MAXH)
      NTICH=(MAXH-MINH)+1
      CALL CHLABJ(1)
      CALL CHLOGS(MINH)
      CALL CHLOGF(1)
      CALL CHLOG(XLEN,NTICH,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CHLABJ(0)
C-----------------------------------------------------------------------
C VERTICAL AXIS
      CALL JMOVE(XOR+XLEN,YOR)
C SEE DEPMIN,DEPMAX ABOVE FOR THE FOLLOWING VALUES
      CALL CEXP(65.0,290000.0,MINV,MAXV)
      NTICV=(MAXV-MINV)+1
      CALL CVLABJ(1)
      CALL CVLOGS(MINV)
      CALL CVLOGF(1)
      CALL CVLOG(YLEN,NTICV,0)
C RESTORE VALUES FOR LEZ ROUTINES
      CALL CVLABJ(0)
C-----------------------------------------------------------------------
C CALL CLGRID
      CALL JMOVE(XOR,YOR)
      IXC=-1
      IYC=NTICV-1
      XSC=XLEN
      YSC=YLEN/REAL(IYC)
      XGS=0.
      YGS=1.
```

```
      XGI=XLEN/REAL(NTICH-1)
      YGI=1.
      NBLANK=1
      CALL CLGRID(XSC,YSC,IXC,IYC,XGS,YGS,XGI,YGI,BLANK,NBLANK)
      CALL JCLOSE
C-----------------------------------------------------------------------
C GENERATE GRAPHICS OUTPUT FOR THE LEZ ROUTINES
      CALL LEZSHW
C TERMINATE ALL GRAPHICS
      CALL LEZTRM
C-----------------------------------------------------------------------
      STOP
      END
```

Appendix C

A List of DI-3000 Routines

This appendix provides a list of DI-3000 routines, and a one line description of their purpose. The contents of this appendix was obtained from PVI's DI-3000 User's Guide [G-5].

3.18 DI-3000 SUBROUTINE SUMMARY

Listed below are all the DI-3000 subroutines grouped by function. Some subroutines appear in more than one place to show all their functional groupings. The functions of these subroutines have been discussed in the preceding sections of Chapter 3. Each individual subroutine is completely described in Appendix A of this User's Guide. Subroutines marked [E] are part of DI-3000 Extended; those marked [M] are part of the Metafile System. Both are options to DI-3000 and may not be available at all installations.

3.18.1 SYSTEM AND DEVICE CONTROL

JBEGIN	Initialize DI-3000
JEND	Terminate DI-3000
JDINIT	Initialize a device
JDEVON	Select a device
JDEVOF	Deselect a device
JDEND	Terminate a device

3.18.2 DRAWING PRIMITIVES

JMOVE	Invoke an absolute, 2D move
JRMOVE	Invoke a relative, 2D move
J3MOVE	Invoke an absolute, 3D move
JR3MOV	Invoke a relative, 3D move
JDRAW	Invoke an absolute, 2D draw
JRDRAW	Invoke a relative, 2D draw
J3DRAW	Invoke an absolute, 3D draw
JR3DRA	Invoke a relative, 3D draw
JPOLY	Draw an absolute, 2D polyline
JRPOLY	Draw a relative, 2D polyline
J3POLY	Draw an absolute, 3D polyline
JR3PLY	Draw a relative, 3D polyline
JMARK	Draw an absolute, 2D marker
JRMARK	Draw a relative, 2D marker
J3MARK	Draw an absolute, 3D marker
JR3MRK	Draw a relative, 3D marker
JPMARK	Draw an absolute, 2D polymarker
JPRMRK	Draw a relative, 2D polymarker
JP3MRK	Draw an absolute, 3D polymarker
JPR3MR	Draw a relative, 3D polymarker
JPOLGN	Draw an absolute, 2D polygon
JRPLGN	Draw a relative, 2D polygon
J3PLGN	Draw an absolute, 3D polygon
JR3PGN	Draw a relative, 3D polygon
JRECT	Draw an absolute rectangle
JRRECT	Draw a relative rectangle
JCIRCL	Draw a circle
JSECTR	Draw a sector
JARC	Draw an arc
JF2PLN	Generate a "smoothed" polyline

3.18.3 PRIMITIVE ATTRIBUTES

JCOLOR	Set the current color
JDCOLR	Set the default color
JINTEN	Set the current intensity
JDINTE	Set the default intensity

JLSTYL	Set the current linestyle
JDLSTY	Set the default linestyle
JLWIDE	Set the current linewidth
JDLWID	Set the default linewidth
JPEN	Set the current pen
JDPEN	Set the default pen
JPEDGE	Set the current polygon edge style
JDPEDG	Set the default polygon edge style
JPFSIM	Set the polygon fill simulation mode
JPINTR	Set the current polygon interior style
JDPINT	Set the default polygon interior style
JPIDEX	Set the current polygon interior color and intensity
JDPIDX	Set the default polygon interior color and intensity
JCMARK	Set the current marker symbol
JDMARK	Set the default marker symbol
JATTRB	Set any current primitive attribute
JDATTR	Set any default primitive attribute
JASAVE	Save all current primitive attribute values
JALOAD	Load previously saved primitive attribute values
JARSET	Reset all current attributes to default values

3.18.4 TEXT PRIMITIVES AND ATTRIBUTES

Text output primitives:

J1TEXT	Output string precision text (FORTRAN 66)
J1STRG	Output string precision text (FORTRAN 77)
J2TEXT	Output character precision text (FORTRAN 66)
J2STRG	Output character precision text (FORTRAN 77)
J3TEXT	Output stroke precision text (FORTRAN 66)
J3STRG	Output stroke precision text (FORTRAN 77)
JHTEXT	Output graphic arts precision text (FORTRAN 66)
JHSTRG	Output graphic arts precision text (FORTRAN 77)

Text attributes:

JPATH	Set the current character path
JDPATH	Set the default character path
JFONT	Set the current character font
JDFONT	Set the default character font
JJUST	Set the current text justification
JDJUST	Set the default text justification
JSIZE	Set the current character size
JDSIZE	Set the default character size
JGAP	Set the current character gap
JDGAP	Set the default character gap
JBASE	Set the current character base line
JDBASE	Set the default character base line
JPLANE	Set the current character plane
JDPLAN	Set the default character plane
JCESUB	Set the current size and position of subscripts.
JDESUB	Set the default size and position of subscripts
JCESUP	Set the current size and position of superscripts
JDESUP	Set the default size and position of superscripts
JCEFLG	Set the current sentinel character flags (FORTRAN 66)

JDEFLG	Set the default sentinel character flags (FORTRAN 66)
JSCEFL	Set the current sentinel character flags (FORTRAN 77)
JSDEFL	Set the default sentinel character flags (FORTRAN 77)

Text positioning and text inquiry:

JCRLF	Generate a carriage return/line feed
JMARGN	Set the character margin
JEXTNT	Return the string extent, stroke precision (FORTRAN 66)
JSTXTN	Return the string extent, stroke precision (FORTRAN 77)
JHXTNT	Return the string extent, graphic arts precision (FORTRAN 66)
JHSXTN	Return the string extent, graphic arts precision (FORTRAN 77)
JIQTXT	Return the hardware text character size

3.18.5 SEGMENTS AND SEGMENT ATTRIBUTES

Segment creation:

JOPEN	Open a temporary segment
JCLOSE	Close a temporary segment
JROPEN[E]	Open a retained segment
JRCLOS[E]	Close a retained segment

Segment attributes:

JTTYPE[E]	Define a global image transformation type
JVISBL[E]	Set the visibility of a retained segment
JDVISB[E]	Set the default visibility of subsequent retained segments
JVSALL[E]	Set the visibility of all retained segments on all devices
JHILIT[E]	Set the highlighting of a retained segment
JDHILI[E]	Set the default highlighting of subsequent retained segments
JDETEC[E]	Set the detectability and PICK priority of a retained segment
JDDETE[E]	Set the default detectability and PICK priority of subsequent retained segments
JSGPRI[E]	Set the segment priority of a retained segment
JDSGPR[E]	Set the default segment priority for subsequently created segments
JPKID[E]	Set the PICK-ID of subsequent primitives within a retained segment
JDPKID[E]	Set the default PICK-ID of subsequent primitives within a retained segment

Segment operations:

JRENAM[E]	Rename a retained segment
JPURGE[E]	Delete a retained segment
JCLEAR[E]	Delete all retained segments
JSASSO[E]	Associate a retained segment with a specific device
JSDASS[E]	Disassociate a retained segment from a specific device
JSAALL[E]	Associate or disassociate all segments with a specific device

JFRAME	Clear the display area and redraw all retained segments
JUPDAT	Flush the internal buffers to ensure a current image
JSCOPY[E]	Make a copy of a retained segment
JSSAVE[E]	Save the current Segment Storage to an internal save file
JSREST[E]	Restore Segment Storage from an internal save file
JSAPND[E]	Append an internal save file to current Segment Storage
JEPSEG[E]	Define a PICK list of retained segments
JISGMT[E]	Return information on a specified retained segment

Segment image transformations:

JT2TRA[E]	Set a 2D translation image transformation for a retained segment
JT2ALL[E]	Set a complete image transformation for a retained segment
JELSEG[E]	Associate the image transformation with a LOCATOR input echo
JEVSEG[E]	Associate the image transformation with a VALUATOR input echo
JIQ2TR[E]	Return the 2D translation components of an image transformation
JIQ2AL[E]	Return all 2D components of an image transformation

BATCH-OF-UPDATES:

JBGBAT[E]	Begin a BATCH-OF-UPDATES
JENBAT[E]	End a BATCH-OF-UPDATES

3.18.6 SHIELDING REGIONS

JSHLDW	Define a shielding region in world coordinates
JSHLDV	Define a shielding region in virtual coordinates
JSHCLP	Enable or disable shielding within a shielding region
JSCALL	Enable or disable shielding within all shielding regions
JIQSHL	Determine if a shielding region has been defined and return the shielding status and coordinates
JIQNSH	List the identifiers of all currently defined shielding regions
JSHPRG	Delete a shielding region
JSPALL	Delete all shielding regions

3.18.7 VIEWING TRANSFORMATIONS

JWINDO	Define the viewing window
JVPORT	Define the viewport
JVUPNT	Define the view reference point
JNORML	Define the viewplane normal
JVUPLN	Define the viewplane distance
JUPVEC	Define the viewup vector
JPARAL	Set the projection to parallel
JPAROB	Set the projection to parallel oblique
JPERSP	Set the projection to perspective
JPEROB	Set the projection to perspective oblique
JCVIEW	Define a viewing transformation in world Cartesian coordinates

JSVIEW	Define a viewing transformation in world spherical coordinates
JHITHR	Set the hither plane distance
JYON	Set the yon plane distance
JWCLIP	Turn window clipping on or off
JHCLIP	Turn hither clipping on or off
JYCLIP	Turn yon clipping on or off
JVSPAC	Redefine the dimensions of the virtual coordinates
JRIGHT	Change the "handedness" of the world coordinate system
JRESET	Reset the viewing transformation to default
JVSAVE	Save the current viewing parameters into an array
JVLOAD	Replace the current viewing parameters from a previously saved array

3.18.8 INPUT FUNCTIONS

JIENAB	Associate a physical input device with a virtual input function on a specific display device
JIDISA	Disassociate a physical input device from a virtual input function on a specific display device
JBUTTN	Request BUTTON input
JLOCAT	Request LOCATOR input
JVALUE	Request VALUATOR input
JKEYBD	Request KEYBOARD input (FORTRAN 66)
JKEYBS	Request KEYBOARD input (FORTRAN 77)
JSTROK	Request STROKE input
JPICK[E]	Request PICK input
JSPICK[E]	Request a software simulation of PICK
JPKAPR[E]	Define the size of the PICK aperture
JCONPK[E]	Continue a PICK function
JEPSEG[E]	Establish a PICK list of retained segments
JPECHO	Set the echo position for subsequent input functions
JELSEG[E]	Associate the image transformation with a LOCATOR input echo
JEVSEG[E]	Associate the image transformation with a VALUATOR input echo

3.18.9 IMAGE TRANSFORMATIONS

JTTYPE[E]	Define a global image transformation type
JT2TRA[E]	Set a 2D translation image transformation for a retained segment
JT2ALL[E]	Set a complete image transformation for a retained segment
JELSEG[E]	Associate the image transformation with a LOCATOR input echo
JEVSEG[E]	Associate the image transformation with a VALUATOR input echo
JIQ2TR[E]	Return the 2D translation components of an image transformation
JIQ2AL[E]	Return all 2D components of an image transformation

3.18.10 MODELING TRANSFORMATIONS

JTRANS[E]	Define a modeling transformation matrix
JBUILD[E]	Build on an existing modeling transformation matrix
JMODEL	Select a modeling matrix for subsequent primitives
JMODON	Enable or disable the current modeling matrix
JCONCT[E]	Concatenate two existing transformation matrices

3.18.11 DEVICE TRANSFORMATIONS

JDEVWN	Define the device window
JDEVVP	Define the device viewport

3.18.12 SYSTEM AND DEVICE INQUIRY

JCP	Return the current position in world coordinates
JIQDDL	Return the release level of the device driver
JIQDIL	Return the release level of DI-3000
JIQDIM	Return the maximum dimensions of the display device in meters

JIQERR	Return the most recent error number detected
JIWIND	Return the window location in 3D, untransformed world coordinates
J1IGET	Return a single INTEGER value (e.g., a current attribute value)
J3RGET	Return three floating point values (e.g., the view reference point)
J4RGET	Return four floating point values (e.g., the viewport borders)
J16GET	Return a four-by-four floating point matrix defining a transformation
JASPEK	Return the aspect ratio of a physical device
JIQTXT	Return the hardware text character size
JIQDEV	Return an INTEGER characteristic of a device (e.g., the number of colors supported)
JISGMT[E]	Return an array of information on a particular retained segment
JIQ2TR[E]	Return the 2D translation components of an image transformation
JIQ2AL[E]	Return all 2D components of an image transformation

3.18.13 METAFILE SYSTEM

JDINIT	Initialize the Metafile Driver
JDEVON	Select the Metafile Driver
JDEVOF	Deselect the Metafile Driver
JDEND	Terminate the Metafile Driver
JMFDAT[M]	Write a data record to the metafile
JMFFMT[M]	Set the metafile format
JIQFMT[M]	Inquire capability of system to determine input file format
JMTEXT[M]	Write a picture title to the metafile (FORTRAN 66)
JMSTRG[M]	Write a picture title to the metafile (FORTRAN 77)
JMGET[M]	Return the current item code from the metafile
JMREAD[M]	Return the current item from the metafile
JMINTR[M]	Translate the metafile item into a DI-3000 action

3.18.14 ERROR PROCESSING AND DEBUGGING

JSETER	Set the error fatality level
JSETDB	Set the debugging commentary level
JFILES	Set the logical unit number of the error log or debug file
JFTOPN	Set the logical unit number and/or file name of the error log or debug file (FORTRAN 66)
JFSOPN	Set the logical unit number and/or file name of the error log or debug file (FORTRAN 77)

3.18.15 SPECIAL DEVICE FUNCTIONS

JPAUSE	Request a pause action
JDD3D	Use 3D virtual coordinates at the device dependent level
JBACKG	Set the background color
JCOTBL	Download a color table of device dependent color indices
JESCAP	Invoke an output escape function
JIESCP	Invoke an input escape function
JINESC	Inquire a physical device on the availability of escape functions

3.18.16 OTHER FUNCTIONS

JBEAM	Explicitly position the drawing beam on a device
JFILES	Set the logical unit number for several internal files
JFTOPN	Set the logical unit number and/or name for several internal files (FORTRAN 66)
JFSOPN	Set the logical unit number and/or name for several internal files (FORTRAN 77)
JCONVW	Convert virtual coordinates to world coordinates
JCONWV	Convert world coordinates to virtual coordinates
JFRAME	Clear the display surface and redraw all retained segments
JUPDAT	Flush the internal buffers to ensure that the picture is current

Appendix D

How to Access and Execute the Common Graphics Library

This appendix explains how to access and execute the CGL on several computer systems. It contains the following sections:

Any FORTRAN 77 code will be able to access the CGL routines which currently use the DI-3000 based CGL. The CGL does not contain any PVI software; therefore, the device drivers, DI-3000 library, extended library, and any other DI-3000 based software must be loaded with the CGL. The user should consider the CGL as an extension of their graphics program. Because NOS and PRIME are ACD supported systems, procedures for these machines are given in this section. As the CGL is moved onto other ACD supported systems, this section will be expanded.

1.0 Accessing The Common Graphics Library On NOS

The CGL is available as a user library under the user number UN=LIBRARY. It may be accessed and loaded manually in a job stream or terminal session. There are also Cyber Control Language (CCL) procedures to facilitate both batch and interactive operation of the CGL.

1.1 Manual Load Sequence

Metafile Driver LOAD/EXECUTE Sequence (NOS)

```
ATTACH,DI3000,DIERFN,MFNODE,SSDUMMY,DD4014/UN=LIBRARY.
ATTACH,DICOMLB/UN=LIBRARY.
LDSET,LIB=DI3000/DICOMLB,MAP=N.
LOAD,MFNODE,SSDUMMY,DD4014.
LGO.
```

NGS LOAD/EXECUTE Sequence

```
ATTACH,DI3000,DIERFN,MFDUMMY,SSDUMMY,DD=DD4014/UN=LIBRARY.
ATTACH,DICOMLB/UN=LIBRARY.
LDSET,LIB=DI3000/DICOMLB,MAP=N.
LOAD,MFDUMMY,SSDUMMY,DD.
LGO.
```

1.2 Batch CCL Procedure

There are two batch procedures to run the CGL. The first procedure uses the standard NOS load sequence and is called CGLGO. The second, called CGLCGO, uses the capsule loader version of DI-3000. Their usage is given below:

```
GET,CGLGO/UN=PVINFO.
CGLGO,DRIVER,MF,SS,ACCOUNT.
```

Where:
- DRIVER - the device driver to be loaded.
- MF - the metafile (omit if none is desired)
- MFNODE - if one is desired.
- SS - use the extended version of DI-3000
- ACCOUNT- ACCOUNT other than LIBRARY (not recommended unless a special driver is used.)

```
GET,CGLCGO/UN=PVINFO.
CGLCGO,LGO,DRIVER,MF,SS,ACCOUNT.
```

Where:
- LGO - the relocatables of the user's program.
- DRIVER - the device driver to be loaded.
- MF - the metafile (omit if none is desired)
- MFNODE - if one is desired.
- SS - use the extended version of DI-3000
- ACCOUNT- ACCOUNT other than LIBRARY (not recommended unless a special device driver is used).

2.0 Accessing The Common Graphics Library On PRIMOS

The CGL is available as a user library on the PRIME network. The CGL pathname is:

PVI.INFO>CGL>DICOM.LIB

and it is located on the M machine. The user can use the load sequences below to build CPL files to create and execute a SEG file for a CGL application program. These sequences can also be performed manually.

The following assumptions hold for all the load sequences in this section:

1) application program resides on a file named TEST.F77.
2) all programs compiled using F77 compiler (i.e., F77 TEST).
3) target device is Tektronix 4014.

2.1 Load Sequence Using Unshared Versions of DI-3000

```
SEG -LOAD
ST 177774 (OPTIONAL)
LO TEST
LO PVI.INFO>CGL>DICOM.LIB
LI DIBASIC
LI DD4014
LI VAPPLB
LI
Q
```

To execute:

```
SEG TEST
```

2.2 Load Sequence Using Shared Version of DI-3000

```
SEG -LOAD
ST 177774 (OPTIONAL)
LO TEST
LO PVI.INFO>CGL>DICOM.LIB
LI D13.SHR
LI VAPPLB
LI
Q
```

To execute:

```
SEG TEST
```

All conventions for using the shared version of DI-3000 (WHOAMI, metafiles, etc.) remain unchanged. See Section 7.1.4.2 of the Graphics Mini-Manual (Central Scientific Computing Complex Document G-1) for complete details.

3.0 Accessing The Common Graphics Library On Vax

Several VAX machines on the field have the CGL available to the users. The load sequence below can be used either in a .COM file or entered interactively.

The following assumptions hold for the given load sequence:

1) application program resides on a file of the form XXX.FOR.
2) all programs are compiled by executing the command FOR XXX, where XXX is the XXX.FOR file from 1).
3) the target device is a Tektronix 410X series terminal.

The load sequence is:

```
LINK 'P1' -
Dxxx:[CGL directory]DICOM.OLB/LIBRARY -
Dyyy:[DI3000 directory]DILIB.OLB/LIBRARY -
Dyyy:[DI3000 directory]DDR405.OBJ -
Dyyy:[DI3000 directory]MFNODE.OBJ -
Dyyy:[DI3000 directory]DIMFLIB.OLB/LIBRARY -
Dyyy:[DI3000 directory]UTILLIB.OLB/LIBRARY -
```

where:

'P1'	-	XXX from 1) and 2) above
Dxxx	-	disk where the CGL directory resides
Dyyy	-	disk where the DI3000 directory resides
CGL directory	-	pathname to the CGL files
DI3000 directory	-	pathname to the DI3000 files
DICOM.OLB	-	CGL code (executables)
DILIB.OLB	-	DI3000 code
DDR405.OBJ	-	device driver
• MFNODE	-	metafile device driver
• DIMFLIB.OLB	-	library containing I/O routines for the Metafile device driver
UTILLIB.OLB	-	library containing utility routines for DI3000

• These two files need not be included in the load sequence if the metafile is not used.

NOTE: Since there is no centralized control over the different VAX machines on the field, the exact pathnames and filenames may be different from machine to machine. The system administrator for the user's machine should be contacted for specific directory and file information.

List of References

The following list of references are denoted by:

G-x refers to the ACD document series.

G-1 GRAPHICS MINI MANUAL (September 1985)

G-2 Guidelines in Preparing Computer-Generated Plots for NASA Technical Reports with the LaRC Graphics Output System (NASA TM 81908 (revised), August 1983)

G-3 Langley Graphics System (January 1983)

G-5 DI-3000 User's Guide (Version 4)

G-6 Metafile System User's Guide

G-7 GRAFMAKER User's Guide, Precision Visuals Incorporated (PVI)

G-10 Device Driver Guides

G-12 Common Graphics Library (CGL) Volume I: LEZ User's Guide (May 1988 - Preliminary)

NMI 1020.1E NASA Langley Research Center Management Manual (Paragraph 13b)

NASA National Aeronautics and Space Administration

Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
NASA TM-101596		

4. Title and Subtitle	5. Report Date
Common Graphics Library (CGL) Volume II: Low-Level User's Guide	April 1989
	6. Performing Organization Code

7. Author(s)	8. Performing Organization Report No.
Nancy L. Taylor Dana P. Hammond Pauline M. Theophilos	CSCC Doc. G-13
	10. Work Unit No. 505-60-01-01

9. Performing Organization Name and Address	11. Contract or Grant No.
NASA Langley Research Center Hampton, Virginia 23665-5225	
	13. Type of Report and Period Covered Technical Memorandum

12. Sponsoring Agency Name and Address	14. Sponsoring Agency Code
National Aeronautics and Space Administration Washington, DC 20546-0001	

15. Supplementary Notes

Nancy L. Taylor: Langley Research Center, Hampton, Virginia.
Dana P. Hammond and Pauline M. Theophilos: Computer Sciences Corporation, Hampton, Virginia.

16. Abstract

The intent of this report is to introduce and instruct the users how to use the Low-Level routines of the Common Graphics Library (CGL). The Low-Level routines form an application-independent graphics package enabling the user community to construct and design scientific charts conforming to the publication and/or viewgraph process. The Low-Level routines allow the user to design unique or unusual report-quality charts from a set of graphics utilities. The features of these routines can be used stand-alone or in conjunction with other packages to enhance or augment their capabilities. This library is written in ANSI FORTRAN 77, and currently uses a CORE-based underlying graphics package, and is therefore machine-independent, providing support for centralized and/or distributed computer systems.

17. Key Words (Suggested by Author(s))	18. Distribution Statement
publication-quality graphics graphics software	Unclassified - Unlimited Subject Category - 61

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of pages	22. Price
Unclassified	Unclassified	425	A18

NASA FORM 1626 OCT 86

☆U.S. GOVERNMENT PRINTING OFFICE: 1989 -629-233/ 00021