

GDDM-PGF



Application Programming Guide

Version 2 Release 1.3

GDDM-PGF



Application Programming Guide

Version 2 Release 1.3

Note!

Before using this information and the product it supports, be sure to read the general information under "Notices" on page ix.

Second Edition (December 1996)

This edition applies to Version 2 Release 1 Modification 3 of the IBM licensed program GDDM-PGF, program number 5668-812, to GDDM-PGF as an optional feature of OS/390 (program number 5645-001), and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the addresses given below.

At the back of this publication is a page titled "Sending your comments to IBM". If you want to make comments, but the methods described are not available to you, please address them to:

IBM United Kingdom Laboratories, Information Development,
Mail Point 095, Hursley Park, Winchester, Hampshire, England, SO21 2JN.

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

This publication contains sample programs. Permission is hereby granted to copy and store the sample programs into a data processing machine and to use the stored copies for internal study and instruction only. No permission is granted to use the sample programs for any other purpose.

© Copyright International Business Machines Corporation 1982, 1996. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Notices	ix
Trademarks and service marks	ix
Preface	xi
Who this book is for	xi
How to use this book	xi
Special note for online users	xi
Latest GDDM information	xii
GDDM publications	xii
Chapter 1. Introduction to business charts	1
Chapter 2. Exploiting the ICU	3
Before you read about the ICU API	3
Loading a chart, modifying it, and displaying the result	7
Some other values that you can set	13
Options and attributes that you can set using calls CSCHA, CSFLT, CSINT, and CSNUM	14
Querying the result of end-user interaction	18
Accessing the ICU directory of GDDM objects	21
Calling the ICU from a program using call CHART	21
Chapter 3. Introducing the Presentation Graphics Routines	23
Example 1	23
Example 2	24
The nine chart types	26
Absolute and relative data	28
How charts are shaded	29
Chapter 4. PG routines that apply to most chart types	31
State-1 and state-2	32
Chart layout calls	32
Chart legend calls	34
Chart axis calls	37
Axis label and tick-mark calls	45
Grid line calls	48
Datum line calls	49
Heading calls	49
Chart note calls	50
Chart termination call	52
Chapter 5. PG routines specific to each chart type	55
Line graphs and scatter plots	55
Surface charts	63
Histograms	69
Bar charts	72
Tower charts	82
Polar charts	88
Pie charts	93
Venn diagrams	101

Table charts	104
Chapter 6. Using PG routines to create complex charts	107
More than one chart on a page	107
Several plotting calls for one chart	110
Mixing chart types	115
Using secondary axes	115
How to prevent the data overwriting the axes	118
Appendix A. Example 1. Display of sales figures	119
Description of the program	119
Source code	119
Appendix B. Example 2. COBOL graphical query example using mapping	141
GDDM glossary	151
Index	161

Figures

1.	"DATA ENTRY" panel of ICU	4
2.	"SAVE AND LOAD CHART" panel of ICU	5
3.	Output from minimum business graphics program	24
4.	Output from simple business graphics program	26
5.	Chart margins and plotting area	33
6.	y axis position options in tower charts	38
7.	x and y axis position options	39
8.	Primary axis intercept	40
9.	Secondary axis intercept	41
10.	Axis range and orientation	42
11.	Output from axis options program	43
12.	Grid lines and tick marks	48
13.	GDDM line types and line widths	57
14.	GDDM-supplied vector marker set ADMDHIMJ	58
15.	Line graph (with curve fitting)	62
16.	Ordering the components effectively	64
17.	Normal and mountain-range shading	65
18.	Surface chart	67
19.	Histogram	71
20.	The three types of bar chart	74
21.	A bar chart on a numeric x axis	75
22.	Bar chart	81
23.	Tower chart	87
24.	Polar chart with numeric x data	88
25.	Polar chart with logical x data	92
26.	Comparison of composite bar chart with multiple pie chart	93
27.	Pie chart where the total size of each pie is proportionate to that year's sales figures.	97
28.	Venn diagram	103
29.	Table chart	105
30.	Several plots on one chart	112
31.	Using secondary axes	116
32.	Sales program: A surface chart	121
33.	Sales program: A floating bar chart	123
34.	Sales program: Proportional pie charts	124
35.	Sales program: The help information	127
36.	Main display for the graphical query program	141
37.	Data-only display for the graphical query program	141
38.	Chart-only display for the graphical query program	142

Tables

1. Breakdown of ICU API options and attributes	15
--	----

Notices

The following paragraph does not apply to any country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

References in this book to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the responsibility of the user.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact Laboratory Counsel, Mail Point 151, IBM United Kingdom Laboratories, Hursley Park, Winchester, Hampshire SO21 2JN, England. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

This publication contains sample programs. Permission is hereby granted to copy and store the sample programs into a data-processing machine and to use the stored copies for internal study and instruction only. No permission is granted to use the sample programs for any other purpose.

Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

APL2	CICS	GDDM
IBM	OS/390	System/370

PC Direct is a trademark of Ziff Communications Company and is used by IBM Corporation under license.

notices

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.

C-bus is a trademark of Corollary, Inc.

Microsoft, Windows and the Windows 95 Logo are trademarks or registered trademarks of Microsoft Corporation.

Preface

The *GDDM-PGF Application Programming Guide* provides guidance information for users of the application-programming interfaces of the IBM licensed program GDDM-PGF.

Who this book is for

This book is for application designers and programmers who are experienced in the following areas:

- Application programming in at least one of the languages supported by GDDM Base and GDDM-PGF. These are:
 - APL2
 - SYSTEM/370 Assembler
 - IBM BASIC
 - COBOL
 - C/370
 - FORTRAN
 - PL/I
- The subsystem under which the GDDM programs are to run
- The information contained in the *GDDM General Information* book.

How to use this book

This book is a companion volume to the *GDDM Base Application Programming Guide*, and should be used in conjunction with that book. All basic instruction in the writing of GDDM application programs is provided in the *GDDM Base Application Programming Guide*. This book (the *GDDM-PGF Application Programming Guide*), provides application-programming guidance for GDDM-PGF users only.

You can read this book sequentially, or read only those chapters that interest you. (The structure of the book is defined in the table of contents.) The appendixes contain complete example programs.

Special note for online users

The pictures in this book may differ from those in the online version because of the characteristics of your graphics display. In the online version, the pictures show the colors described in the text. The representations of line types, shading patterns, and symbols vary depending on your graphic display, but they are usually more accurate in the online version than in this book.

Latest GDDM information

For up-to-date information on GDDM products, check our Home Page on the Internet at the following URL:

<http://www.hursley.ibm.com/gddm/>

You might also like to look at the IBM Software Home Page at:

<http://www.software.ibm.com>

GDDM publications

GDDM Base	<i>GDDM Base Application Programming Guide</i> , SC33-0867
	<i>GDDM Base Application Programming Reference</i> , SC33-0868
	<i>GDDM Diagnosis</i> , SC33-0870
	<i>GDDM General Information</i> , GC33-0866
	<i>GDDM/MVS Program Directory</i> , GC33-1801
	<i>GDDM/VM Program Directory</i> , GC33-1802
	<i>GDDM/VSE Program Directory</i> , GC33-1803
	<i>GDDM Messages</i> , SC33-0869
	<i>GDDM Series Licensed Program Specifications</i> , GC33-0876
	<i>GDDM System Customization and Administration</i> , SC33-0871
GDDM-GKS	<i>GDDM User's Guide</i> , SC33-0875
	<i>GDDM Using the Image Symbol Editor</i> , SC33-0920
GDDM-IMD	<i>GDDM Interactive Map Definition</i> , SC33-0338
GDDM-IVU	<i>GDDM Image View Utility</i> , SC33-0479
GDDM-PGF	
	<i>GDDM-PGF Application Programming Guide</i> , SC33-0913
	<i>GDDM-PGF Programming Reference</i> , SC33-0333
	<i>GDDM-PGF Interactive Chart Utility</i> , SC33-0328
	<i>GDDM-PGF Vector Symbol Editor</i> , SC33-0330
	<i>GDDM-PGF OPS User's Guide</i> , SC33-1776

GDDM/MVS is an element of OS/390. GDDM-REXX/MVS and GDDM-PGF are optional features of OS/390. For a complete list of the publications associated with OS/390, see the *OS/390 Information Roadmap*, GC28-1727.

Chapter 1. Introduction to business charts

GDDM-PGF provides four different ways of creating and displaying business graphics (for example, line graphs and pie charts):

- **The Interactive Chart Utility (ICU)** is an interactive program that lets people without programming experience create charts at the terminal. For instance, the chart-by-example panels prompt the terminal user to enter some chart data, and can display the user's data in several alternative formats. All the user has to do is to select the most suitable.

Initially, the ICU assumes default values for all options and attributes, such as shading colors, margin widths, and label types. The user can then type in changes to these values, thereby modifying the chart to its most suitable appearance. As an example of the fineness of detail that can be requested, the user could specify that the chart heading is: (1) red, (2) displayed using a Gothic symbol set, (3) is 1.9 times the basic character size, and (4) is left-aligned at the top of the chart. At each stage, the user can display the chart on the terminal to see the effect of the latest changes.

When changes have been created, they can be saved on disk and subsequently redisplayed by using the ICU interactively, or by a direct call to the ICU from an application program (see the next item in this list). Such a call can pass different data from that originally typed in, for example, data that has been calculated by the application program or retrieved from a data base.

The ICU is the quickest and easiest way to interactively create business charts. The ICU is described in the *GDDM-PGF Interactive Chart Utility* book.

- **The ICU application programming interface (API)** is a set of calls for creating and displaying business charts that gives you control over all of the options and attributes of the various ICU panels, and lets you present them to the user as parts or all of an ICU-like interface for your program. When the user finishes the ICU session, control returns to your program, which can then query and use the contents of the ICU panel entry fields. Calls in the ICU API always start with the letters CS. You should use this interface in preference to the interface described below, unless you require specially complex charts.
- **The Presentation Graphics Routines (PG routines)** are a set of calls for creating and displaying business charts. They differ from the ICU API in that they do not give you a panel-based end-user interface. Call statements to PG routines specify the attributes and options required, any text that is to appear on the chart (such as the chart heading), and the chart type and data. PG routines call names always start with the letters CH.
- **The Online Presentation System (OPS)** is an interactive program that lets people:
 - Create and display online presentations either for “public” display on a video monitor or video projector, or for “private” viewing at a terminal. OPS is suitable for all types of presentation, including displays of business results or education sessions.
 - Create high quality material for plotting or printing
 - Create and modify graphics files.

The *GDDM-PGF OPS User's Guide* describes the capabilities of OPS and how to use them.

This book, the *GDDM-PGF Application Programming Guide*, provides guidance information on the use of the ICU application-programming interface and the PG routines. Chapter 2, “Exploiting the ICU” on page 3 describes the ICU API., and Chapter 3, “Introducing the Presentation Graphics Routines” on page 23 through Chapter 6, “Using PG routines to create complex charts” on page 107 cover the PG routines. The calls from both interfaces are listed alphabetically and described in the *GDDM-PGF Programming Reference* book.

Chapter 2. Exploiting the ICU

The ICU API is very powerful. You can use it to perform various functions:

- To start an interactive session, on the basis of the chart passed to it
- To display the chart on a screen, but not permit an interactive session
- To send the chart to a printer
- To save the passed data on disk in ICU format
- To save a chart on disk in GDF format
- To add the chart to the current GDDM page
- To manage GDDM libraries of chart data and formats, symbol sets, saved pictures, GDF files, and generated maps

Many of the calls in the ICU API have a wide range of possible parameter values, allowing you to control the ICU panels, and the options, attributes, and values in them. In most of your programs, you would probably not want to use every single call, or every possible parameter value. Instead, you would use most of the defaults that are already set up for the ICU, and change only those that are not appropriate to the chart you are trying to produce. In this way, it is possible to produce complex business graphics with very few ICU API calls.

This chapter does not attempt to show you how to use the ICU API to change every possible panel, option, attribute, and value in the ICU. Instead, it uses example programs to illustrate typical tasks that you might want to do. For a complete list of all the calls in the ICU API, and their parameters, see the *GDDM-PGF Programming Reference* book.

Before you use the ICU API, you should read the *GDDM-PGF Interactive Chart Utility* book and use the ICU. You may then become familiar with the panels, options, attributes, and values that you can control using the API. The main advantage of the ICU is that the user can inspect the partly built chart at any stage to see the effect of the entered options. It is, therefore, a very quick process to achieve the final desired presentation.

Before you read about the ICU API

When you have created a chart, you can save it on disk for future use. The example program in the next section loads a previously saved chart. If you want to try the code, do the following tasks, using the ICU interactively to create and save a chart. Look up “saving” in the *GDDM-PGF Interactive Chart Utility* book if you get stuck:

1. Start up the ICU.
2. From the Home panel, go to the Data Entry panel 2.1.

```

2.1                                     DATA ENTRY
ADM1008 I SAMPLE DATA PROVIDED

Command ==> _____
Data Group Names -----> HONG KONG      MIAMI
X Values      X Lables    Y1            Y2            Y3            Y4
1              JAN        5                8                .                .
2              FEB        4                8                .                .
3              MAR        3               8.5                .                .
4              APR        4                9                .                .
5              MAY       4.8                9                .                .
6              JUN        5                9                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
.                  .          .                .                .                .
Commands: SAMPLE   (Provide Sample Data)   CLEAR   (Remove ALL Data)

PF: 1=Help 2=Save/Load 3=End 4=Print 5=Display 7=Up 8=Down 10=Left 11=Right
     12=Home

```

Figure 1. "DATA ENTRY" panel of ICU

3. Type in the word `SAMPLE` on the command line. This loads sample data about daily hours of sun for Hong Kong and Miami. You get a message, "SAMPLE DATA PROVIDED," as shown in Figure 1.

Look at the sample data. The X data are the months 1 through 6. The X labels are the months January through June. The Y1 data are the daily hours of sun for Hong Kong for those months. The Y2 data are the daily hours of sun for Miami for the same months.

4. Press the key marked PF2. This gives you the “SAVE AND LOAD CHART” panel. Do not worry about the warning message:

CURRENT CHART DATA HAS UNSAVED CHANGES

This simply tells you that you have not yet saved the data.

```

                                SAVE AND LOAD CHART
ADM1100 W  CURRENT CHART HAS UNSAVED CHANGES
What do you want to do?  ==> 3      1 - Load from a File
                                      2 - Reset to initial defaults
                                      3 - Save in a File
                                      4 - List Files, to pick or delete

Which part of chart?  ==> 3      1 - Format
                                      2 - Data
                                      3 - Both Format and Data
                                      4 - Whole chart as Graphic Data Format (GDF)
                                      (not for Load or Reset)

Replace ? NO      (Type YES for Load or Reset to replace your
                  current chart even if it has unsaved changes,
                  or for Save to replace an existing file)

      File name  Description of file
Format ==> sunshine
Data   ==> =      =
GDF    ==>
      (May be '=' meaning the same as the Format name or description)

PF: 1=Help 2=Save/Load 3=End 4=Print 5=Display 12=Home

```

Figure 2. "SAVE AND LOAD CHART" panel of ICU

On the panel, shown in Figure 2 specify:

What do you want to do? ==> 3

to indicate that you want to save it.

Which part of chart? ==> 3

for "Both Format and Data." These are the two components of a chart.

Replace ? YES

indicates that you want to save your new data and to delete any existing file with the same name.

Format ==> SUNSHINE

Data ==> = =

as the filename of the format and data files.

Press ENTER. You get a message:

'SAVE' SUCCESSFUL FOR CHART FORMAT AND DATA

5. Press the key marked PF12 to take you to the Home Panel.

6. Press the key marked PF9 to take you out of the ICU.

You should now have a saved chart that you can load and use with the first example program. The chart is saved in two parts, the format and the data:

- An ICU format file consists of:
 - All options and attributes
 - Axis labels
 - Margin sizes
 - Legend positioning
 - Chart notes
- An ICU data file consists of everything else:
 - The x data, the y data, and the z data
 - The data labels
 - The chart heading text
 - The legend text

These files are each assigned an 8-character name, for example, SUNSHINE. On CMS, the full filenames are SUNSHINE ADMCFORM A and SUNSHINE ADMCDATA A. The same name can therefore be used for both the format and the data. On other subsystems, the 8-byte name is a member name within a library.

Loading a chart, modifying it, and displaying the result

The following program loads the SUNSHINE chart data and format files, appends some (fictional) data for London, alters the legend accordingly, repositions the legend, displays the results, and saves the format and data into new format and chart data files called SUNSHINE, overwriting the old.

```

TRYAPI: PROC OPTIONS(MAIN);
DCL ID FIXED BIN(31);
DCL COMP FIXED BIN(31);
DCL ELEM FIXED BIN(31);
DCL FLOAT_ARRAY (6) FLOAT DEC(6);
DCL CSFLT_CONTROL (1) FIXED BIN(31);
DCL CSSICU_CONTROL (1) FIXED BIN(31);
DCL CSFLT_FLT (1) FLOAT DEC(6);

CALL FSINIT;

CALL CSQUID(ID);      /* Unreserved chart number returned in id *//*A*/
CALL CSCCRT(ID,1);    /* Initialize new chart *//*B*/

CALL CSLOAD(ID,1,'SUNSHINE');      /* Load format *//*C*/
CALL CSLOAD(ID,2,'SUNSHINE');      /* Load data *//*C*/

CALL CSQNUM(ID,5,COMP);      /* Query number of data groups *//*D*/
CALL CSQNUM(ID,6,ELEM);      /* Query number of elements *//*D*/
COMP = COMP + 1;              /* Add 1 to data group number */
CALL CSNUM(ID,5,COMP);        /* Create new data group *//*E*/

FLOAT_ARRAY(1)=1.2;          /* Initialize float_array with */
FLOAT_ARRAY(2)=1.4;          /* */
FLOAT_ARRAY(3)=1.5;          /* London daily sunshine hours */
FLOAT_ARRAY(4)=2.0;          /* */
FLOAT_ARRAY(5)=2.5;          /* that is the Y3 data */
FLOAT_ARRAY(6)=3.0;          /* */

CALL CSYDT(ID,COMP,ELEM,FLOAT_ARRAY); /* Add Y3 array to the *//*F*/
/* new data group. */
CALL CSCHA(ID,6,COMP,1,6,'LONDON'); /* Add data-group name. *//*G*/

CSFLT_CONTROL(1)=0;
CSFLT_FLT(1)=6;              /* Set offset to 6 to */
/* reposition legend. */
CALL CSFLT(ID,18,2,1,CSFLT_CONTROL,CSFLT_FLT); /*H*/

CSSICU_CONTROL(1)=7;          /* ICU to display only. */
CALL CSSICU(ID,1,CSSICU_CONTROL); /* Start ICU - to only *//*I*/
/* display the chart. */

CALL CSSAVE(ID,1,'SUNSHINE',2); /* Save updated format *//*J*/
CALL CSSAVE(ID,2,'SUNSHINE',2); /* Save updated data *//*J*/
CALL CSCDEL(ID);              /* Delete chart *//*K*/

CALL FSTERM;
%INCLUDE ADMUPINF;            /*L*/
%INCLUDE ADMUPINC;
END TRYAPI;

```

The first three data declarations, ID, COMP, and ELEM, are to hold data returned by query calls. The other three declarations are arrays that are used in some of the calls.

Go through the program and look at what the ICU API calls do:

Getting an identifier for a chart, using call CSQUID

The CSQUID call at /*A*/ is a query call that asks GDDM for a new chart identifier. GDDM returns a value in the single parameter of this call.

Creating a chart, using call CSCCRT

The CSCCRT call at /*B*/ initializes a default set of chart data and format for a new chart session. This call is required even if you are going to subsequently load a saved format or data. The call has two parameters:

- The first parameter is a fullword integer containing the identifier of the current chart. All but two of the calls in the ICU API have the current chart identifier as their first parameter.
- The second parameter always has a value of 1.

Loading chart format and data, using call CSLOAD

The two CSLOAD calls at /*C*/ restore the format and data files that were saved in the earlier ICU session. CSLOAD has three parameters:

- The first parameter is again the current chart identifier.
- The second parameter is a fullword integer specifying the type of object to be loaded:
 - 1 Format object
 - 2 Data object
 - 5 Data definition object
 - 6 Import data as specified in the current data definition
- The third parameter contains an 8-byte character string specifying the name of the object to be loaded. This parameter is ignored for object type 6.

Querying chart control values, using call CSQNUM

The next two calls, at /*D*/ in the example program are both CSQNUM. CSQNUM is used to query various control values for the current chart session. These values may have been explicitly set, using the CSNUM call, or may be existing defaults. The control values returned by CSQNUM may be used in subsequent calls. The CSQNUM call has three parameters:

- The first parameter is the current chart identifier.
- The second is a number from 1 through 13, corresponding to the 13 different types of control value that you can query. For a complete list, see the *GDDM-PGF Programming Reference* book.
- The last parameter is the control value (returned by GDDM) The CSQNUM call in the example queries the following types of control value:
 - 5 Number of data groups (Y1, Y2, and so on) returned in COMP
 - 6 Number of data elements (X values) returned in ELEM.

The values returned in COMP and ELEM are 2 and 6 respectively, because there are two data groups (Hong Kong and Miami), each containing six elements.

Setting chart control values, using call CSNUM

The program uses the value returned in COMP by the first CSQNUM call, adds 1 to it, and uses it in the CSNUM call at /*E*/ to append a third data group to the existing two data groups. The three parameters correspond to the three parameters of CSQNUM:

- The first parameter is the current chart identifier.
- The second is the type of control value that you want to set.
- In the last parameter you specify the control value.

Setting y-data values, using call CSYDT

FLOAT_ARRAY holds the data for the new data group that is added to the restored data. The CSYDT call at /*F*/ copies the y data from the array FLOAT_ARRAY into the third data group created by CSNUM. The data is plotted when the chart is displayed. The call has four parameters:

- Current chart identifier.
- The second parameter is the number of the component that the y data is going into. It must be less than or equal to the current number of components as set by CSNUM, or returned by a CSQNUM, preceding the CSYDT.
- The third parameter is a **count** of the number of y-data values in the component. In the example, this is the existing number of y-data values as returned by CSQNUM.

For tied data, if this number is less than the current number of y-data values, the array in the last parameter is substituted for the first **count** values of the component specified in the second parameter, and those values not altered are set to missing values.

(For free data, the number in **count** can be less than or equal to the current number of y-data values. Missing y values are valid for free data.)

Tied and free data are explained in the *GDDM-PGF Interactive Chart Utility* and the *GDDM-PGF Programming Reference* books.

- The last parameter is the name of an array holding the data. In the example, FLOAT_ARRAY is explicitly initialized with some data, but this data could just have easily come from an input record.

There is a corresponding query call CSQYDT that you can use to query the y-data values. There is an example of the use of CSQYDT at “Querying the result of end-user interaction” on page 18. See the *GDDM-PGF Programming Reference* book for details.

Setting character option values, using call CSCHA

The CSCHA call at /*G*/ gives the new data group a name. It is one of the most versatile calls in the API. You can use it to set the headings, titles, names, labels, or notes that the end user would enter in text input fields on the ICU. (You should note, however, that items that the end user would set by entering yes or no are set by another call, CSINT, using 1 or 2 instead of yes or no.)

In the example program, CSCHA specifies the name “LONDON” for the new data group. If you were using the ICU interactively, you would enter this name above the data group on the Data Entry panel 2.1. This name appears in the legend when the chart is displayed. The CSCHA call has six parameters:

- Current chart identifier.
- The second parameter is the type of character string to be set. This contains a type number in the range 1 through 33, representing the 33 different character items that you can set. In the example, a type value of 6 means data-group name. Table 1 on page 15 gives you a brief breakdown of the types. For a complete list, see the *GDDM-PGF Programming Reference* book.
- For character items that have a number of entries (like data-group names in the example) the third parameter gives the entry that the character string relates to. In the example, COMP is used as it has a value of 3 for the third data group.
- The fourth parameter is a **count** of the number of character strings in the final parameter (see below).
- The fifth parameter is the **length** of the longest string in the final parameter.
- The final parameter is a character array made up of **count** strings, with each string being the same **length** and padded with blanks where necessary. The example has only one 6-character string containing the new data-group name.

You can query the character option values that can be set by CSCHA, and the character string lengths, with the query calls CSQCHA and CSQCHL respectively. See the *GDDM-PGF Programming Reference* book for details.

Setting floating-point option values, using call CSFLT

You can use the CSFLT call to set the floating-point values for the ICU to use. The parameters are:

- Current chart identifier.
- The type of option value to be set. Type 18 in the example means that the vertical offset of the legend from base is to be set. There are 30 valid types that you can set. Table 1 on page 15 gives a quick breakdown of the types.
- The start position, in the array that is the final parameter, of the first floating-point value to be set.
- A count of the number of values in the final two parameters.
- A control array that determines how the values in the final parameter are used. The first element in this array corresponds to the first element in the final parameter, and so on:
 - 0 Change the current value of this item to the value in the corresponding element of the final parameter.
 - 1 Leave the item at its current setting. Ignore the value in the corresponding element.
 - 3 Set the item as if * had been entered in the input field on the ICU.
- An array of floating-point numbers for the set of items defined by the type parameter.

For a list of all the valid types and values, see CSFLT and its corresponding query call CSQFLT in the *GDDM-PGF Programming Reference* book.

Starting an ICU session, using call CSSICU

Now that the new data has been set up and given a name, the program is ready to start the ICU session for the end user. The CSSICU call at /*I*/ does this. It is another very versatile call. It has three parameters:

- Current chart identifier.
- Number of elements in the control array. Our example has only the first element, which is initialized with a value of 7.
- A control array having up to three elements that define how the ICU operates:
 1. The **DISPLAY** element can take any value in the range 1 through 8, with these different effects:
 - 1 Present the ICU Home Panel to the end user and begin an interactive session on the basis of the format and data passed. All panels are available, subject to the ISOLATE element (see below). This option is not available under IMS/VS.
 - 2 Display the chart to the end user and begin an interactive session on the basis of the format and data passed. This option is not available under IMS/VS.
 - 3 Display the chart, but do not permit an interactive session. The chart is therefore protected from modification by the end user. Access to the Save/Get and Print panels may be allowed (see ISOLATE element below). This option is not available under IMS/VS.
 - 4 Print the passed chart on the device identified by a CSCHA call, then return control to the application program.
 - 5 Add the chart to the **current** page.
 - 6 Add the chart to the **current** page but do not make any calls to CHTERM. CHTERM is the PG routine call that terminates the PG routines, and frees associated storage. Some other PG routine calls (principally CHAREA) can be made before the call to the ICU. On return from the ICU, additional plots can be added to the current axes.
 - 7 Display the chart but do not permit any interaction. This value is used in the first example program. The first interrupt returns control to the program.
 - 8 Display only the Directory panel in advanced mode format and associated Help panels. With one exception, all functions not concerned with library management are omitted. The exception is that GDF files may be shown and printed. The ICU becomes an interactive library manager, allowing the operator to manipulate chart data and formats, symbol sets, saved pictures, GDF files, and maps. This option is not available under IMS/VS.

When element 1 has a value of 1, 2, 3, 4, or 5, the ICU processing is preceded and followed by calls to PG routine call CHTERM and is therefore independent of any PGF calls made before or after the call to the ICU.

When element 1 has a value of 1, 2, 3, 4, or 7, CSSICU constructs the chart in a **unique** page that is inaccessible to your program, so it cannot subsequently influence or access the result of any end-user interaction.

2. The **HELP** element determines whether the PF-key information is shown initially on the display and display GDF panels. The parameter is ignored unless element 1 is 1, 2, 3, or 8.
 - 0 Not shown. Can be shown by pressing ENTER.
 - 1 Shown. Can be removed by pressing ENTER.
3. The **ISOLATE** element determines whether the operator can access the Save/Load, and Directory panels. This parameter is ignored unless element 1 is 1, 2, or 3.
 - 0 The user can access panels specified using the ICUISOL default parameter in the GDDM external defaults (For more information, see the *GDDM Base Application Programming Guide* book. By default, this is **all** panels. Other values of ICUISOL give the same as values 1 and 2 below.
 - 1 Save/Load and Directory panels may not be accessed.
 - 2 Save/Load panel may be accessed, but not the Directory panels.

Saving a chart, using call CSSAVE

The CSSAVE call at /*J*/ saves chart objects. The example changes the position of the legend and so affects the format, and adds a data group and its name to the data. The program therefore stores the format and data files. CSSAVE has four parameters:

- Current chart identifier.
- Type of object to be saved:
 - 1 Chart format
 - 2 Chart data. This option is used in the example program, because the program has added a data group and its name to the data.
 - 4 Chart data and format as a GDF object
 - 5 Data definition
- Name of the object to be saved. This character string must be 8 characters long.
- The last parameter specifies the action to be taken if a file already exists with the filename specified in the third parameter:
 - 1 Protect the existing file
 - 2 Overwrite the existing file.

Deleting a chart, using call CSCDEL

The CSCDEL call at /*K*/ simply deletes the chart format and data for the chart that was created with CSCCRT. CSCDEL only deletes the chart from memory – it does not affect any charts saved on disk. The call has one parameter, the identifier of the chart to be deleted.

Including the PGF entry point declarations

When using the ICU API, you have to include a set of GDDM entry point declarations, as at /*L*/. The file ADMUPINF contains PL/I declarations for the GDDM calls beginning FS, and ADMUPINC for the ICU API calls, that all begin with CS.

Some other values that you can set

That completes the explanation of the calls in the first example program. The program showed you that you create a chart session, and set various values before starting up the ICU. The various values set were:

- Control values, using call CSNUM
- Character option values, using call CSCHA
- Floating-point option values, using call CSFLT
- Y-data values, using call CSYDT.

In addition to the above list, you can also set integer option values, and set, select, and exclude x- and z-data values:

Setting integer option values, using call CSINT

In the same way that CSCHA corresponds to character input fields, and CSFLT corresponds to floating-point input fields, the call CSINT normally corresponds to integer input fields on the ICU menus. Here is an example of the call, and its declarations:

```
DCL CONTROL_ARRAY (1) FIXED BIN(31) INIT(0);
DCL VALUE_ARRAY (1) FIXED BIN(31) INIT(4);

CALL CSINT(ID,1,1,1,CONTROL_ARRAY,VALUE_ARRAY);
/* Change chart type to bar chart */
```

The parameters are as follows:

- Current chart identifier.
- The type of option value to be set. Type 1 in the example means that the chart type is to be set. There are 51 valid types that you can set. Table 1 on page 15 gives a quick breakdown of the types. For a complete list see CSINT in the *GDDM-PGF Programming Reference* book.
- The start position, in the array that is the final parameter, of the first option value to be set.
- A count of the number of option values in the final two parameters.
- A control array that determines how the option values in the final parameter are used. The first element in this array corresponds to the first element in the final parameter, and so on:
 - 0 Change the current value of this item to the value in the corresponding element of the final parameter.
 - 1 Leave the item at its current setting. Ignore the value in the corresponding element.
 - 3 Set the item as if * had been entered in the input field on the ICU.
- An array of integer values for the set of items defined by the type parameter. In the above example, there is one element only, a value of 4 meaning bar chart.

For a list of all the valid types and values, see CSINT and its corresponding query call CSQINT in the *GDDM-PGF Programming Reference* book.

Options and attributes that you can set using calls CSCHA, CSFLT, CSINT, and CSNUM

Table 1 on page 15 is a breakdown of the options and attributes that you can set using the calls CSCHA, CSFLT, CSINT, and CSNUM. The numbers in the second, third, fourth, and fifth columns correspond to the type of option or attribute that you can specify in the second parameter of each call. The final column contains the numbers of the corresponding ICU panels where the ICU user can set the options and attributes.

Table 1. Breakdown of ICU API options and attributes

	CSCHA type	CSFLT type	CSINT type	CSNUM type	ICU PANEL
Axis-label options	7, 10–12, 33	11–14	27–30		4.4, 2.4, 2.8
Axis-line options			23, 24		4.2
Axis scale, range and position		9, 10	25, 26, 39		4.3, 5.2
Axis-title options	13–16	29, 30	21, 22, 46, 47		4.1
Bar-chart options	18	2, 8	5, 6		1.4
Chart heading	8, 9	17	38, 39		5.1, 5.2
Chart-note options	19, 20	20, 26	42, 48	1, 2	3
Chart proportions and dimensions		19, 28	41		5.4
Chart-type options			1, 16		1, 2.6
Data attributes		4, 5	11–17	7	2.6
Data-group options	6, 17	7	20	5, 6, 12, 13	2.7, 2.3
Data-import options	25, 26, 31, 32		44		2.9
Data interpretation and indexing		6	19		2.5
Datum-line options		24, 25	36, 37		4.8
Directory options	4, 5		18	8	M
Histogram options			4		1.3
Legend options		18	40		5.3
Line-graph options			2		1.1
Menu-control options			18		M
Pie-chart options			7, 17		1.5
Polar chart options			9		1.7
Printer/plotter options	3	1	45		PF4
Reference-line options		22, 23	35, 51		4.7
Save/load options	1, 2, 27–32				S or PF2
Scale marks and grids		15, 16	31 –34		4.3, 4.6
Subchart options	21, 22	21	43	3, 4, 10, 11	6
Surface-chart options			3		1.2
Table-chart options	23, 24	27, 31	49, 50		1.9
Tower-chart options		3	10		1.8
User-defined labels	10, 33				4.4
Venn-diagram options			8		1.6

Setting x-data values, using call CSXDT

The call that you use to set x data is CSXDT. Here is a typical call:

```
CALL CSXDT(ID,0,6,X_ARRAY);          /* Set x data */
```

The call has four parameters:

- Current chart identifier.
- The number of the component that the x data is going into.

For tied data this must be 0. The x values are then used as the x component for all of the y data to be plotted against.

(For free data, it must be less than or equal to the current number of components as set by CSNUM, or returned by a CSQNUM, preceding the CSXDT call. Missing x values are not valid for free data.)

- A **count** of the number of x-data values in the component. In the example, this is 6.

For tied data, if this number is less than the current number of x-data values, the array in the last parameter replaces the first **count** values in the component specified in the second parameter, and those not replaced are set to missing values.

(For free data, the number in **count** can be less than or equal to the current number of x-data values.)

- The name of an array holding the data.

There is a corresponding query call CSQXDT that you can use to query the x-data values. See the *GDDM-PGF Programming Reference* book for details.

Setting z-data values, using call CSZDT

The call that you use to set z data (for tower and table charts) is CSZDT. Here is a typical call:

```
CALL CSZDT(ID,6,Z_ARRAY);          /* Set z data */
```

The call has three parameters:

- Current chart identifier.
- A **count** of the number of z-data values in the component. In the example, this is 6.

It must be less than or equal to the current number of components as set by CSNUM, or returned by a CSQNUM, preceding the CSZDT call.

- The name of an array holding the data.

There is a corresponding query call CSQZDT that you can use to query the z-data values. See the *GDDM-PGF Programming Reference* book for details.

Selecting, excluding, and querying data

When using the ICU interactively, the end user can select and exclude x data, or z data, or both, using the DATA MANIPULATION panel 2.2.

With the ICU API you can do the same operations, using the calls CSXSL and CSZSL. You should always ensure that your data values are correct before you use select or exclude, because if you use select/exclude, and then change the data, all of the data is automatically reselected.

Here is a typical CSXSL call and its necessary declaration:

```
DCL SEL_ARRAY(3) FIXED BIN(31);
.
.
.
SEL_ARRAY(1)=2;
SEL_ARRAY(2)=1;
SEL_ARRAY(3)=2;

CALL CSXSL(ID,0,3,SEL_ARRAY);
```

The parameters are:

- Current chart identifier.
- Number of the component for which the x-data values are to be selected or excluded. For tied data this is always 0.
(For free data, it must be less than or equal to the current number of components returned by CSQNUM type 5.)
- **Count** of the number of elements in the select/exclude array. This must be less than or equal to the number of elements as returned by CSQNUM type 6. If it is less, any elements outside of the array stay selected.
- An array corresponding to the first **count** values of the x-values array of the CSXDT call. Valid values in the array are:

- 1 Select
- 2 Exclude

You can use the CSZSL call to select or exclude a data group. Here is a typical call, and its necessary declaration:

```
DCL SEL_ARRAY(3) FIXED BIN(31);
.
.
.
SEL_ARRAY(1)=2;
SEL_ARRAY(2)=1;
SEL_ARRAY(3)=2;

CALL CSZSL(ID,3,SEL_ARRAY);
```

The parameters are:

- Current chart identifier.
- **Count** of the number of elements in the select/exclude array. This must be less than or equal to the number of components as returned by CSQNUM type 5. If it is less, then any components outside of the array stay selected.
- An array corresponding to the first **count** values of the z-values array of the CSZDT call. Valid values in the array are:

- 1 Select
- 2 Exclude

The query calls corresponding to CSXSL and CSZSL are CSQXSL and CSQZSL.

Deleting chart notes or subcharts, using call CSDEL

You can use the CSDEL call to delete two types of chart items: chart notes or subcharts. Here is an example, that deletes a chart note:

```
CALL CSDEL(ID,1,1); /* Delete chart note 1 */
```

The three parameters have the following meanings:

- The current chart identifier.
- The type of item to be deleted:
 - 1 Delete the chart note specified in the final parameter.
 - 3 Delete the subchart specified in the final parameter.
- The number of the note or subchart that you want to delete. The numbers are as specified in type parameter values 1 through 4 on the CSNUM call, and as returned in type parameter values 1 through 4 on the CSQNUM call.

Deletion of a chart note by CSDEL has the same effect as the “delete” line command on panel 3 (Chart notes) in the ICU.

Deletion of a subchart by CSDEL has the same effect as the “delete” line command on panel 6 (Multiple charts) in the ICU.

All subsequent notes or subcharts are renumbered, and the maximum number is reduced by one.

Querying the result of end-user interaction

You have already seen that the call CSSICU starts an ICU session for an end user. In the example, the user is given access to the display panel of the ICU only. If the user is given access to a full interactive session, he or she can use the ICU to make amendments (for example, add two new y-data groups) to the chart.

When the terminal user ends the ICU session, control passes back to your program. Your program can then query the result of the terminal user's interaction. For example, here is a small piece of code that you might add to the example program. The CSSICU call for a full interactive session is also shown:

```

DCL NEW_DATA (6) FLOAT DEC(6);
DCL SUB1 FIXED BIN (31);
DCL SUB2 FIXED BIN (31);
DCL OLD_COMP FIXED BIN(31);

CSSICU_CONTROL(1)=1;
CALL CSSICU(ID,1,CSSICU_CONTROL); /* Start full ICU session */

OLD_COMP=COMP;          /* Save old number of data groups *//*M*/
CALL CSQNUM(ID,5,COMP);  /* Query new number of data groups *//*N*/
IF COMP > OLD_COMP       /* If user has added data groups */
  THEN CALL HOURS_CHECK; /* then call the checking routine */
.
.
.
HOURS_CHECK: PROC;
DO SUB1 = OLD_COMP+1 TO COMP; /*0*/
  CALL CSQYDT(ID,SUB1,ELEM,NEW_DATA); /* Get a new data group. *//*P*/
  DO SUB2 = 1 TO ELEM;        /* Loop to examine data. */
    IF NEW_DATA(SUB2) > 24 THEN /* If user thinks there */
      .                        /* are more than 24 hours */
      .                        /* in a day, take action. */
      .
  END;
END;
END HOURS_CHECK;

```

The code uses the CSQNUM and CSQYDT query calls to check if one of the items of data (hours of sunshine each day) that the user entered into two new data groups is greater than twenty-four.

The earlier program example restores a chart that has two data groups, and adds a third data group to it. In the above example, COMP initially contains a value of 3, the number of data groups that existed before the ICU session was started. This value is saved in OLD_COMP at /*M*/.

If the ICU user then adds, for example, two new data groups during the interactive session, the value returned in COMP by the CSQNUM call at /*N*/ is 5 – the number of data groups that exist **after** the ICU session.

OLD_COMP, incremented by 1, and COMP, are used to set the bottom and top limits of SUB1, the subscript used in the do-loop at /*0*/. The do-loop contains a CSQYDT call at /*P*/. The parameters of CSQYDT are:

1. Specifies the current chart identifier.
2. Specifies the number of the data group (SUB1 in the example) for which you want data returned in the array in the final parameter.
3. Specifies the number of elements in the final parameter.

The CSQYDT in the example is written assuming that a value was returned in ELEM by a previous CSQNUM call, as in the earlier example program.

4. Returns the values in specified elements of the specified data group.

There then follows another do-loop that checks each element of the returned data group to check if it is greater than 24.

The CSQYDT call is quite easy to use, because it corresponds to a call (CSYDT) that sets a small number of values. But what about query calls such as CSQINT and CSQFLT, that relate to calls such as CSINT and CSFLT, that set an extremely wide range of options and attributes? And why might you query such values? For example, you might want to query some options and attributes for an existing chart and use them to construct a new chart. There is a query call that relieves your program from the burden of having to remember the number of different types of option and attribute that can be set and queried. The call is CSQCS, and here is an example of one of its uses:

```
DCL TYPE (1) FIXED BIN(31);
DCL SUB FIXED BIN(31);
DCL ECOUNT (3) FIXED BIN(31);
DCL CONTROL_ARRAY (51) FIXED BIN(31);
DCL VALUE_ARRAY(51) FIXED BIN(31);

CALL CSQCS(ID1,1,0,1,TYPE); /* Returns maximum CSINT type value *//*A*/

DO SUB = 1 TO TYPE(1);      /* Uses TYPE as top subscript value *//*B*/

    CALL CSQCS(ID1,1,SUB,1,ECOUNT);/* Returns maximum element value*//*C*/
                                   /* allowable for CSINT type SUB */

    /* CHART-ID TYPE START ELEMENTS      CONTROL      VALUES */
    CALL CSQINT(ID1, SUB, 1, ECOUNT(1), CONTROL_ARRAY, VALUE_ARRAY);/*D*/

    CALL CSINT(ID2, SUB, 1, ECOUNT(1), CONTROL_ARRAY, VALUE_ARRAY); /*E*/
END;
```

The above section of code contains a CSQINT call within a loop. The program goes through the loop the sufficient number of times so that GDDM returns all the CSINT information for the current chart. That information is subsequently used by a CSINT call to set the same values for a different chart.

The CSQCS call at /*A*/ returns in TYPE the maximum type parameter value that GDDM allows for CSINT. The do-loop that starts at /*B*/ uses the value returned in TYPE as the top limit for the subscript SUB that controls the loop. This use of the subscript ensures that the program passes through the loop the sufficient number of times to issue the subsequent CSQCS call once for every possible value of the CSINT type parameter.

The CSQCS call at /*C*/ returns in ECOUNT the maximum element parameter value that GDDM allows for the CSINT type parameter value, as defined by the subscript SUB.

The CSQINT call at /*D*/ then returns, in VALUE_ARRAY, an array containing all the integer option values for all the elements (as defined by ECOUNT(1)) of the CSINT type parameter value, as defined by the subscript SUB.

Finally, the CSINT call at /*E*/ uses the values returned by CSQINT, to initialize the integer option values for a new chart ID2.

You can use the CSQCS call to return maximum parameter values, and other information, for CSNUM, CSINT, CSFLT, and CSCHA (and their corresponding query calls) and for CSQCHL. For more details, see the *GDDM Base Application Programming Reference* book.

Accessing the ICU directory of GDDM objects

You can use the CSDIR call to build a list of GDDM objects, as an end user would using the ICU directory function. You can subsequently query the list using the CSQDIR call. Here is a typical CSDIR call:

```
CSDIR('SUNSHINE',5,'STATSLIB',COUNT); /* Add data to directory */
```

The meanings of the parameters are:

- The filename of the object to be placed in the directory.
- The type of object. Valid types are:

0	Image symbol set
1	Vector symbol set
2	Generated map group
3	Saved pictures (in device-dependent form)
4	Chart format
5	Chart data
6	Chart format and data
7	GDF files
9	Chart data definition
10	Projection
11	Image data
- The library that contains the named object. This has the same effect as the library field on the directory panel of the ICU. Under TSO, it contains a ddname. Under CMS, it contains a filemode. Information about other subsystems is given in the *GDDM-PGF Programming Reference* book.
- A value returned by GDDM of the number of objects in the directory list.

Once you have created your directory, you can query it in your program using CSQDIR:

```
CALL CSQDIR(31,NAME,OBJECT_TYPE,LIBRARY,DATE,24,DESCRIPTION);
```

The parameters for CSQDIR specify:

- The number of the object in the list about which information is required.
- The name of the object whose number is given in the first parameter.
- The type of the object at **count**, returned by GDDM. The value is one of the type values listed for CSDIR above.
- The library of the object, returned by GDDM.
- The date and time that the object was last updated. The format of the date and time are as specified in your GDDM defaults.
- The **length** of the description string to be returned in the seventh parameter.
- The description of the object, truncated or padded to **length**.

Calling the ICU from a program using call CHART

You can also call the ICU directly from an application program, passing the data to be presented and the name of a previously saved chart format, using call CHART. It is similar in some ways to the CSSICU call in the ICU API. However, CHART does not give you anything like the flexibility that you get using CSSICU with the other calls in the ICU API.

CHART is described in the *GDDM-PGF Programming Reference* book.

Chapter 3. Introducing the Presentation Graphics Routines

This chapter, and the following three chapters, describe the Presentation Graphics Routines (PG routines). This chapter introduces the use of the PG routines with two program examples. The first is elementary, the second a little more elaborate. The chapter also describes briefly the types of business chart that the routines can produce. Finally, the concept of absolute and relative data is discussed, together with the method used by the routines to shade business charts.

Example 1

The first example is a program with the minimum number of PG routine calls. As with the GDDM API, all attributes that are not given a value explicitly are given a default value by the PG routines.

```
BIZ1: PROC OPTIONS(MAIN);
CALL FSINIT;                                /* Initialize GDDM */
DCL X08(8) FLOAT DEC(6)
    INIT(3, 8, 12,13, 16, 19,22, 24);        /* X data */
DCL Y08(16) FLOAT DEC(6)
    INIT(8,1E72, 13,14, 12,10.4, 9,7.2,      /* Y data */
        7, 9,14.1,16,1E72,1E72, 8,6.4);

CALL CHPLOT(2,8,X08,Y08);                  /* Plot 2 components, */
                                          /* each with 8 points */
CALL CTERM;                                /* Terminate PGF, free associated storage */
CALL ASREAD(TYPE,MOD,COUNT);               /* Send business chart to screen */
CALL FSTERM;                                /* Terminate GDDM, free associated storage */

%INCLUDE(ADMUPINA);                         /* Include declarations of */
%INCLUDE(ADMUPINC);                         /* GDDM and PGF entries */
%INCLUDE(ADMUPINF);
END BIZ1;
```

The output from this basic program is shown in Figure 3 on page 24. The following points are of interest.

PG routine defaults: The program's output shows that several default values have been applied by PG routines. For example, the colors, the line types, the margin sizes, and the label type have all been chosen by PG routines in the absence of any specific settings by the programmer. The axes have been autoscaled according to the actual data.

FSINIT call with PG routines: The FSINIT call is required, as it is in GDDM general graphics programs. No further call is required to initialize PG routines.

Chart added to current page: The two PG routine calls in the example (each of the form CALL CHxxxx) may be thought of as subroutine calls. The plotting calls in particular, such as CHPLOT in the example, cause a business chart to be added to the current page's graphics field (more precisely, to its picture space. See the chapter that describes the hierarchy of GDDM concepts in the *GDDM Base Application Programming Guide*). As always, the graphics occupies the complete screen (or printer page) by default.

Presenting the x and y data: For a single plotting call there is **one shared set of x values**. There may be several corresponding sets of y values, each of which is called a **data group** or **component**. The present example has two components, one shown as a blue line, the other as a red line. The appearance of components varies from one chart type to another. A two-component surface chart would have two shaded layers, a two-component bar chart would have two sets of bars.

Missing y values: Each data group must contain the same number of y values as there are x values. If one or more y values are missing from your data, you should supply a dummy value of 10^{72} . In the example, the first data group has one missing value, and the second, two.

Sending the chart to the device: As with general graphics, no transmission occurs until an ASREAD or FSFRCE call is issued. The graphics that result from PG routines are merged with any other graphics and alphanumerics in the normal way.

Including the PGF entry-point declarations: When using PG routines, it is necessary to include a more comprehensive set of GDDM entry declarations. The files ADMUPINA and ADMUPINF contain PL/I declarations for the GDDM calls beginning AS and FS, and ADMUPINC, those for the PG routines, which all begin with CH.

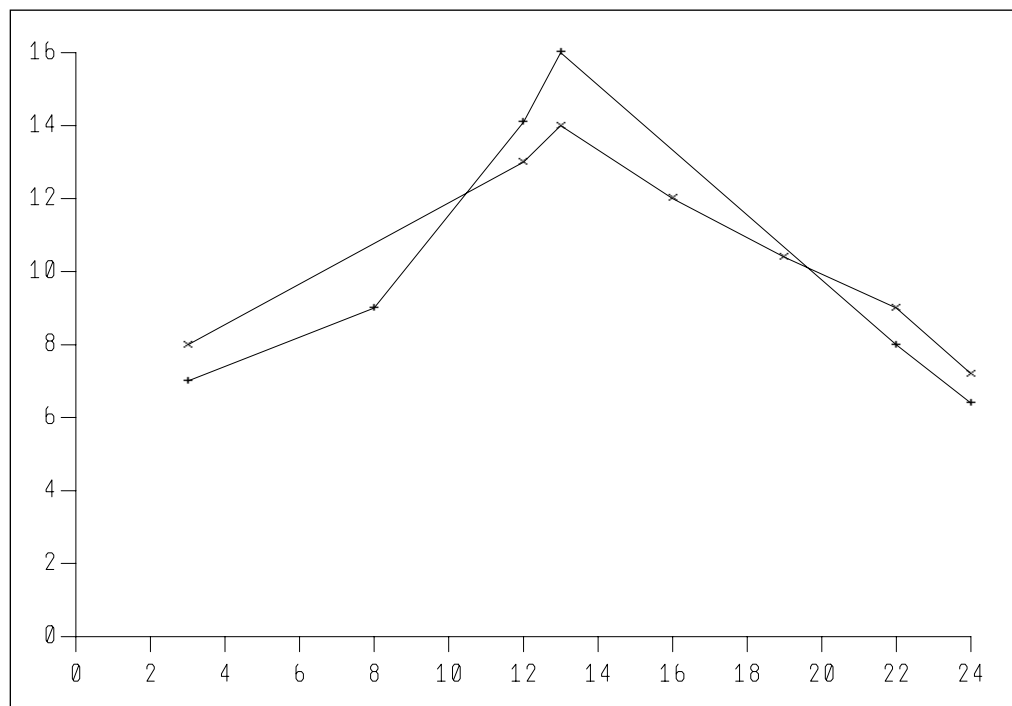


Figure 3. Output from minimum business graphics program

Example 2

The number of calls to PG routines for a simple chart is typically between six and ten. The next example is an enhanced version of the previous one. Various attributes have been set explicitly that were previously left to take the default value.

```

BIZ2: PROC OPTIONS(MAIN);
CALL FSINIT;                                /* Initialize GDDM */
DCL X08(8) FLOAT DEC(6)
      INIT(3, 8, 12,13, 16, 19,22, 24);      /* X data */
DCL Y08(16) FLOAT DEC(6)
      INIT(8,1E72,13 ,14, 12,10.4, 9,7.2,    /* Y data */
          7, 9,14.1,16,1E72,1E72, 8,6.4);

/* Declare heading attributes: color ch_mode s_set id ch_size*/
DCL HEAD_ATTS(4) FIXED BIN(31) INIT(2, 3, 193, 200);
/* Declare label attributes: color ch_mode */
DCL LAB_ATTS(2) FIXED BIN(31) INIT(6, 2);

/* Declare chart-box attributes: color */
DCL BOX_ATTS(1) FIXED BIN(31) INIT(5);
/* Load symbol set: vector set s_set s_set id */
CALL GSLSS( 2, 'ADMUVGEP', 193);

CALL CHHATT(4,HEAD_ATTS); /* Set attributes for chart heading */
CALL CHLATT(2,LAB_ATTS); /* Set attributes for axis labels */
CALL CHBATT(1,BOX_ATTS); /* Set attributes for chart box */
CALL CHHEAD(31,'AIR TEMPERATURE ON 27 JULY 1992');/*Cht heading */
CALL CHSET('CBOX');      /* Request frame around the chart */
CALL CHSET('KBOX');      /* Request frame around the legend */

CALL CHVMAR(7,15);        /* Vertical margins: left 7,right 15*/

CALL CHKEY(2,7,'LONDON CARDIFF');          /* Specify the 2 */
                                          /* keys for legend */
CALL CHXTTL(36,'TIME OF DAY TEMPERATURE WAS RECORDED');/*X-axis */
                                          /*title. */
CALL CHYTTL(18,'DEGREES CENTIGRADE');      /*Y-axis */
                                          /*title. */

CALL CHPLOT(2,8,X08,Y08);/*Plot 2 components, each with 8 points*/
CALL CHTERM; /* Terminate PGF, free associated storage */
CALL ASREAD(TYPE,MOD,COUNT);/*Send out business chart to device */
CALL FSTERM; /* Terminate GDDM, free associated storage*/

%INCLUDE(ADMUPINA); /* Include GDDM and PGF entry points */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END BIZ2;

```

The effect of the PG routines used in the above program can be observed by comparing the output in Figure 4 on page 26 with that of the previous example, Figure 3 on page 24. The calls themselves are discussed in detail in Chapter 4, “PG routines that apply to most chart types” on page 31.

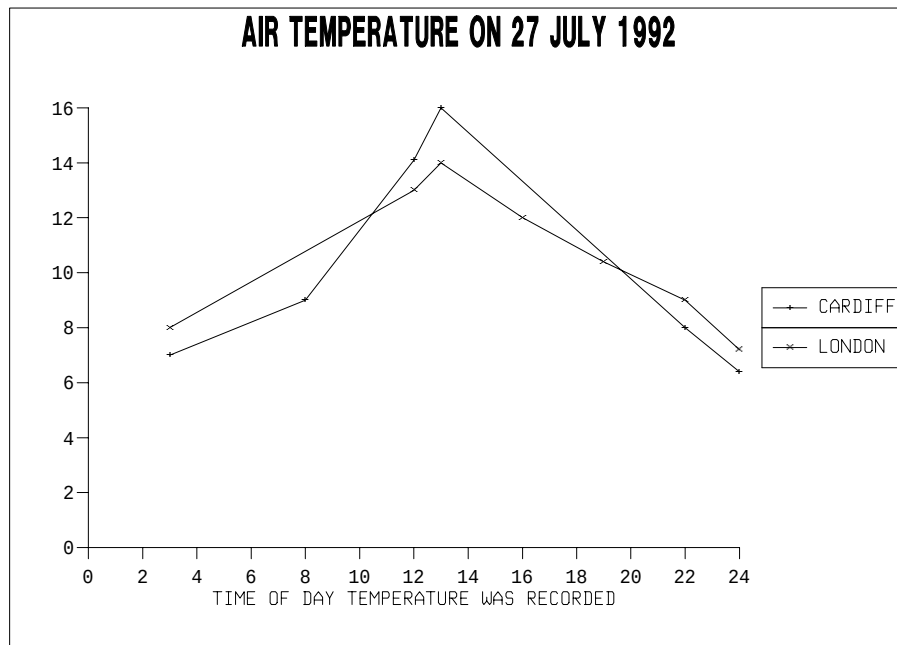


Figure 4. Output from simple business graphics program

The nine chart types

PG routines let you draw these different types of chart:

- Line graphs (and scatter plots)
- Surface charts
- Histograms
- Bar charts
- Tower charts
- Polar charts
- Pie charts
- Venn diagrams
- Table charts.

Line graphs: The plotted points for each component are joined by a polyline (or, optionally, a fitted curve), and each plotted point is indicated by a marker. In general each component uses a different color, line type, and marker to aid recognition. A typical line graph is shown in Figure 15 on page 62. A special case of the line graph is when the lines themselves are suppressed. The resultant chart is known as a **scatter plot**. An example is shown in Figure 12 on page 48. Alternatively, the lines may be shown without markers on the plotted data points.

Surface charts: These are similar to line graphs but the areas between the successive data lines are shaded and the plotted points are not indicated by markers. A typical surface chart is shown in Figure 18 on page 67. This is one of the so-called **stacked** chart types. The value of a component for a particular x coordinate is naturally interpreted as the **depth** of the layer at that point – in other words the difference between the y values of that component and those of the previous component. Stacked charts therefore normally use “relative data.” This concept is discussed in “Absolute and relative data” on page 28.

Histograms: In a histogram a particular bar represents the y data value over the range of the x axis indicated by the width of the bar. For example, in the typical histogram shown in Figure 19 on page 71, the first bar indicates that 12.5% (the y value) of the population are aged between 0-10 (the x range, indicated by the bar width). Histograms, like the previous two chart types, have a numeric x axis. Any further components are stacked on top of the first component.

Bar charts: Here each bar (or set of bars if there is more than one component) represents a separate x entity. For example, each bar might represent the copper production of a particular country. The bars are of equal width. Often they are associated with successive days, weeks, months, or years. The bars can be regularly spaced out along the x axis without representing any significant x values, as when showing copper production by country. Or they can be placed at significant x values, as when the x axis represents time. If the x values are not significant, the chart is said to have a **logical x axis**. If they are, the chart is said to have a **numeric x axis**. If the y data represents values occurring at regular time intervals, as in the bar chart shown in Figure 22 on page 81, then logical and numeric x axes would give similar results.

Tower charts: These are similar to bar charts, but have a three-dimensional appearance and a third axis, the z axis. They are sometimes called skyscraper or Manhattan charts. They usually have several components, each being drawn at a different specified z value, like the example in Figure 23 on page 87. Bars can be stacked but towers cannot; otherwise, the choice between bar and tower charts depends mainly on appearance.

Polar charts: These are like circular line graphs. They are sometimes called star or radar charts. If the x axis represents a single continuous variable, then polar charts are most useful for displaying cyclical data. A patient's temperature taken at intervals over a day would be a typical example. This type of polar chart is illustrated in Figure 24 on page 88. Another important use is for comparing two or more entities, such as companies, on several different parameters, such as revenue, profit, capital invested, and so on. There is an example of this type in Figure 25 on page 92.

Pie charts: The sectors of a pie chart indicate the relative size of various elements within a whole. The pie chart shown in Figure 27 on page 97, for example, shows that sales in the London region accounted for 16% of the 1991 sales. In a multiple pie chart, several pies are placed side by side for comparison.

Venn diagrams: This uncommon form of chart displays the logical relationship between two overlapping sets. For example, the Venn diagram shown in Figure 28 on page 103 shows that 37 of the 113 female students entering St. Basil's College intend to study mathematics.

Table charts: This form of chart is simply data displayed in rows and columns. The table is contained within a ruled box, and ruled lines also separate the columns, the component names, and x-labels. There is a table chart at Figure 29 on page 105.

All the example charts mentioned above were created using the PG routines. They might equally well have been built using the Interactive Chart Utility (ICU). The ICU translates the requests made by the operator into PG routine calls. These are subsequently translated into GDDM line, arc, and area calls.

Absolute and relative data

In nonstacked chart types (such as the line graph shown in Figure 15 on page 62) the data passed to PG routines correspond to the actual plotted points for each component. This form of data is called **absolute data**, and may be requested by this call:

```
CALL CHSET('ABSOLUTE');    /* Plot each component independently */
```

If, instead, you look at a typical stacked chart type (the surface chart in Figure 18 on page 67), you see that it is the depth of a component that indicates its actual data value. The pink (North America) layer and the turquoise (Middle East) layer are the deepest, because those two regions have the greatest oil production.

For this type of chart the natural way to supply the data is to give the actual oil production for each region and request that PG routines stack the various components. The call required is:

```
CALL CHSET('RELATIVE');    /* Plot each component relative to the */
                           /* previous one                        */
```

So, the actual points to be plotted for a particular component are obtained by adding the y data of that component to the y data of the previous components.

Note that absolute data is the default, so you must explicitly specify relative data for all stacked chart-types (surface charts, histograms, composite and floating-bar charts). Of course, there is no need to request relative data if only one component is to be plotted.

How charts are shaded

Various types of “stacked chart” are all shaded in the same way:

1. The first component is shaded from the x axis (or state-1 y datum line, if any). See “Datum line calls” on page 49). This line is called the **reference line** or the **shading reference line**.
2. Every other component is shaded from the previous component’s data line.

The most common case is where there is no y datum line and the data is all positive (as in Figure 18 on page 67). In this case there is no possibility of shading overlap. Because relative data is used, each component’s data line is always be above the previous one.

Shading overlap may occur in the follow circumstances if:

- One or more relative data items are negative. This causes the component to be shaded “backward” (that is, over the previous component). Because stacked charts are designed to display the relative contribution of various parts to a total value, negative data should not be met in any practical chart.
- A stacked chart is being used, but the programmer has forgotten to specify that the data is relative. This mistake might be made by converting a line-graph program to a surface-chart program simply by changing the CHPLOT into a CHSURF. The shading would then move back and forth between the various absolute data lines, creating a meaningless and unreadable chart. The statement `CALL CHSET('RELATIVE')` must be added if you switch to a stacked chart type.
- The x axis (or the state-1 y datum line, if any) has been placed above the first data component. If the y data values for a particular x are, say, 4 9 12 18 26, and you have your x axis (or datum line) at y=11, then the first component is shaded from y=11 to y=4. The second component is shaded from y=4 back over the first component to y=9, and so on.

So, in general, your shading reference line should be below your first data component. Special effects can be achieved with a shading reference line in the middle of your data, but you then have to adjust the data to prevent double shading (see an example of this technique in “Bar chart example” on page 78).

Another way of reducing the visual confusion that can arise with shading overlap is to use **mountain range shading**. More information is given in “Surface charts” on page 63.

Chapter 4. PG routines that apply to most chart types

This chapter describes the various calls to PG routines in groups of logically associated calls:

- Chart layout calls
- Chart legend calls
- Chart axis calls
- Axis label and tick-mark calls
- Grid line calls
- Datum line calls
- Heading calls
- Chart note calls
- Chart termination call.

There are several types of PG routine. Some set an attribute or option that may affect any subsequent chart plotting call. For example:

```
CALL CHLT(4,LINE_TYPES);    /* Set line types for the next plot */
CALL CHSET('CBOX');         /* Request frame around the chart  */
CALL CHLATT(4,LAB_ATTRS);   /* Set label text attributes   */
```

Others specify text that appears somewhere on the chart:

```
CALL CHHEAD(22,'SALES ANALYZED BY AREA'); /* Chart heading */
CALL CHKEY(2,6,'S-WESTS-EAST');          /* Chart keys    */
CALL CHNOTE('C1',19,'Provisional Figures'); /* Chart note    */
```

There are also the ten plotting calls:

```
CALL CHPLOT(2,8,X_DATA,Y_DATA);           /* Line graph    */
CALL CHSURF(3,15,X_DATA,Y_DATA);          /* Surface chart  */
CALL CHHIST(1,7,RNG1,RNG2,Y_DATA);        /* Histogram     */
CALL CHBAR(3,7,Y_DATA);                   /* Bar chart (logical x axis) */
CALL CHBARX(3,9,X_DATA,Y_DATA);           /* Bar chart (numeric x axis) */
CALL CHTOWR(1,3,6,Z_DATA,X_DATA,Y_DATA);  /* Tower chart    */
CALL CHPIE(1,12,Y_DATA);                  /* Pie chart      */
CALL CHVENN(241.0,134.0,63.0);            /* Venn diagram   */
CALL CHPOLR(3,6,X_DATA,Y_DATA);           /* Polar chart    */
CALL CHDTAB(4,6,Z_ARRAY,X_ARRAY,Y_DATA);  /* Table chart    */
```

State-1 and state-2

Before any plotting call has been made, your program is said to be in **state-1**. This is the state in which most of the options and attributes are set.

When the first plotting call is made, the chart axes are constructed by the PG routines and your program is said to move into **state-2**. Many options and attributes cannot be specified in state-2. For example, there is no point in allowing an axis-range to be specified when the axes have already been constructed.

When you are constructing more than one chart, you may want to return to state-1 after plotting the first chart. This process is described in Chapter 6, “Using PG routines to create complex charts” on page 107.

Chart layout calls

By default, the business chart occupies the whole screen (or printer page). But if a GDDM graphics field or a GDDM picture space has been established, the business chart lies within these objects.

Defining the chart area, using call CHAREA

To present two business charts on a page (or to confine the business chart to part of the picture space, so that general graphics may occupy the remainder) the CHAREA call must be used. Here is an example:

```
DCL (WIDTH,DEPTH) FLOAT DEC(6); /* Declare temporary variables */
CALL GSQPS(WIDTH,DEPTH);        /* Query the picture space */
CALL CHAREA(0.0,0.5*WIDTH,0.0,DEPTH);/* Specify chart area that */
                                   /* occupies the left half */
                                   /* of screen */
```

So, the chart area is expressed in “picture-space coordinates.”

Note that it is **not** permitted to use GDDM viewports. Any such specification is ignored.

Requesting a framing box or background shading, using call CHSET

To request a framing box around the chart area (as shown in Figure 4 on page 26), the following call must be issued:

```
CALL CHSET('CBOX'); /* Request frame around the chart area */
```

Any such options and attributes remain in effect for all subsequent plotting calls. To reset this particular option, the following call can be issued:

```
CALL CHSET('NCBOX'); /* Suppress frame around the chart area */
```

On color display units and printers, it is possible to request background shading of the whole chart area:

```
CALL CHSET('CBACK'); /* Shade the whole chart background */
```

This call gives the effect of a business chart drawn on colored paper. It should be used with caution, particularly with Venn diagrams (which set the mixing mode to “mix”) or if the chart is to be sent to a printer. The resultant data stream is a large one, and may exhaust the triple-plane PS-stores on a display device. This is

known as PS overflow, which is covered in the *GDDM Base Application Programming Guide*.

To remove the 'CBACK' option, you must respecify 'CBOX' or 'NCBOX'.

Setting box and background attributes, using call CHBATT

The color, line type, and line width of the box framing the chart area, and the color of the chart background, can be specified by this call:

```
/*                      Color line-type line-width */
DCL BOX_ATTRS FIXED BIN(31) INIT( 2,      0,      2);
CALL CHBATT(3,BOX_ATTRS); /*Set chart box and background attribs*/
```

There are several similar calls that set line attributes for other parts of the chart (such as axes or grid lines).

Setting the chart margins, using calls CHHMAR AND CHVMAR

The central part of the chart area is known as the **plotting area**. It is surrounded by the horizontal and vertical chart margins which are (by default) 5 rows deep and 10 columns wide respectively. To adjust the size of the margins (and therefore the plotting area), these calls are used:

```
CALL CHHMAR(4,7); /* Set horizontal margins: 4 (bottom), 7 (top)*/
CALL CHVMAR(6,15);/* Set vertical margins: 6 (left), 15 (right) */
```

The effect of these two calls is shown in Figure 5.

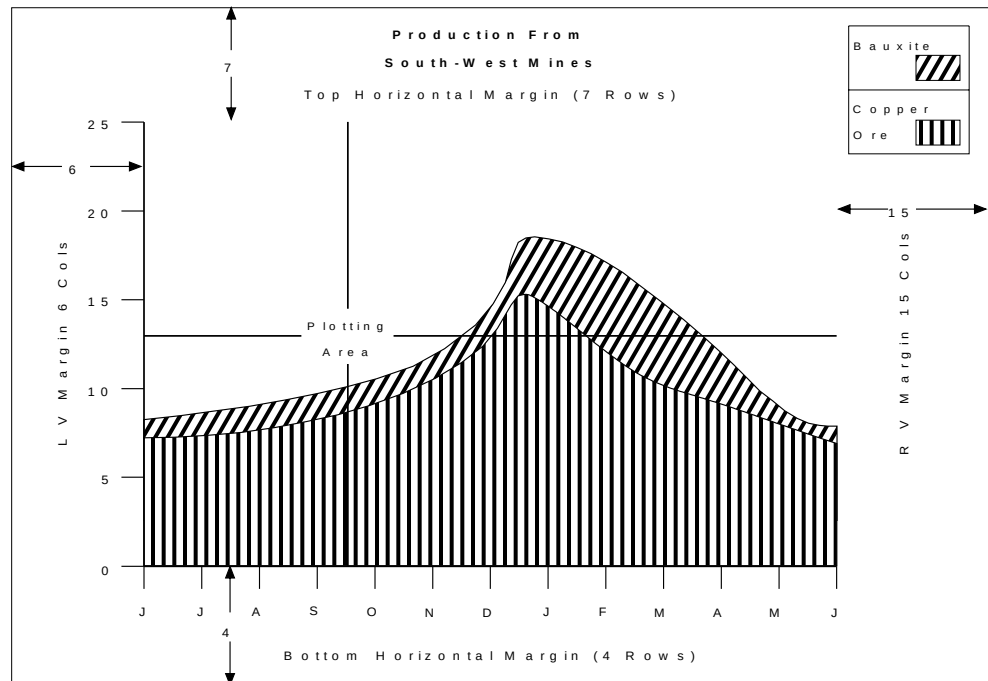


Figure 5. Chart margins and plotting area

The default position of the chart legend is in the right-hand vertical margin. Ten columns are rarely sufficient to accommodate the legend, so a CHVMAR call similar to the one shown is usually issued to prevent the legend encroaching on the plotting area.

Chart legend calls

The chart legend identifies the various components by associating a key symbol (showing the color and line type or shading pattern used for a particular component) with some **key text** describing that component. A typical legend is shown in Figure 18 on page 67.

Specifying the legend key labels using call CHKEY

This call specifies the text of the various keys:

```
CALL CHKEY(4,8,'ITALY  BELGIUM PORTUGAL FRANCE ');
```

The first parameter, (4), specifies the number of components in the chart, and therefore the number of legend entries. The second parameter, (8), gives the length of each key. The third parameter consists of the four keys concatenated together. Those keys that are less than 8 characters long must be padded with blanks (or nulls). The length of this character string is the product therefore of the first two parameters.

When the keys are placed side-by-side in a horizontal legend, space can be saved by padding the keys with nulls (=X'00'), instead of blanks (=X'40'). Trailing nulls are discarded by the PG routines and therefore give the effect of varying length key text. A chart using this technique is shown in Figure 16 on page 64.

Two special characters are permitted within the text for a particular key. The ; (semicolon) character requests a line break, thereby enabling multiline key text. The _ (underscore) character indicates the required position of the key symbol with respect to the key text. If no underscore appears in the text for a particular key, the key symbol (which is 3 characters long) appears to the left of the key text.

This example uses both the special characters:

```
CALL CHKEY(2,13,'COPPER;ORE _ BAUXITE;  _'); /* Two multi- */
                                              /* line keys */
```

The resultant legend has two 2-line key entries, each with the key symbol to the right of the second text line (as shown in Figure 5 on page 33).

If you want ; or _ in the key text, you must code ;; or __ (that is, coding each character twice).

Specifying the legend base position, using call CHKEYP

The key entries may either be placed on top of each other in a vertical margin (a **vertical legend**), or be placed side by side in a horizontal margin (a **horizontal legend**). This is a typical call to specify the legend position:

```
CALL CHKEYP('V','L','C'); /* Request a vertical legend, in the */
                          /* left-hand margin, centered */
```

- The first parameter may be set to V (vertical legend) or H (horizontal).
- The second parameter may be set to L (left), R (right), T (top), or B (bottom), specifying which margin the legend should occupy.
- The third parameter may be set to C (centered within the specified margin), L or R (left- or right-adjusted within a horizontal margin), T or B (top- or bottom-adjusted within a vertical margin).

Invalid combinations are diagnosed and rejected.

Setting the legend offset, using call CHKOFF

To adjust the position of the legend relative to the base position set by CHKEYP, the following call is used:

```
CALL CHKOFF(-15.0,4.0);      /* Adjust legend position: left by*/
                             /* 15 cols and up by 4 rows      */
```

This call is used most often to shift the legend into a vacant space within the plotting area.

The margin chosen for the legend also controls its format. If legends are placed in the left or right margin, they are arranged vertically in columns, thus:

-X- FUEL
-*- HOUSING
-#- TRANSPORT
-0- FOOD AND CLOTHING

If they are placed in the top or bottom margin, they are arranged horizontally in lines, thus:

-0- FOOD AND CLOTHING	-#- TRANSPORT	-*- HOUSING	-X- FUEL
-----------------------	---------------	-------------	----------

Setting the maximum legend width/height, using call CHKMAX

If a horizontal legend is too wide to fit into the chart area, PGF breaks it into two (or more) lines, one below the other. Similarly a vertical legend may be split into two or more columns. To control the maximum legend width/height, this call is used:

```
CALL CHKMAX(60,24);  /* If the legend is horizontal, break to */
                     /* a new line if the next key would take */
                     /* you past 60 columns. If it is vertical,*/
                     /* break to a new column rather than      */
                     /* exceed a depth of 24 rows              */
```

By specifying a maximum width or height in a CHKMAX call, you can have linear legends arranged in a number of lines, and vertical ones arranged in a number of columns thus:

- Columnar legend with width of two columns specified by CHKMAX:

-0- FOOD AND CLOTHING	-*- HOUSING
-#- TRANSPORT	-X- FUEL

- Linear legend with two lines specified by CHKMAX:

-0- FOOD AND CLOTHING	-#- TRANSPORT
-*- HOUSING	-X- FUEL

PG routines that apply to most chart types

The margin and the orientation within it establish the **base position** of the legend. (The default base position is the right margin with central orientation.)

You can move the legend to any offset from the base position by a call to CHKOFF.

A chart using split legends is shown in Figure 16 on page 64.

Setting legend box or blanking, using call CHSET

These options affect the legend's appearance:

```

/*****
/* Legend boxing */
/*****
CALL CHSET('KBOX');      /* Each key entry is boxed          */

CALL CHSET('NKBOX');     /* Key entries are not boxed          */
                        /* (this is the default option)      */

/*****
/* Legend blanking */
/*****
CALL CHSET('BKEY');      /* Blank out the legend area before its */
                        /* construction, so the legend takes */
                        /* precedence over the chart itself */

CALL CHSET('NBKEY');     /* Do not blank out the legend area before */
                        /* its construction (default option) */

/*****
/* Legend order */
/*****
CALL CHSET('KREVERSED');
                        /* Reverse the normal order of key entries */
                        /* Vertical legends are now to be built */
                        /* from top to bottom, horizontal legends */
                        /* are to be built from right to left */

CALL CHSET('KNORMAL'); /*Legends built in normal order (default)*/
```

An example of a boxed legend is given in Figure 15 on page 62.

Every option has a default value. If the default value is suitable, there is no need to set the option explicitly. For example, the default action for key boxing is that no key box is drawn. If you do not want a key box, there is no need to specify CHSET('NKBOX'). The NKBOX option is provided in case you need to reset a previous CHSET('KBOX') specification.

Setting legend text attributes, using call CHKATT

Like all text in the chart, the legend text can have its attributes set.

```

DCL KTEXT_ATTRS(5) FIXED BIN(31)
/*                               Color mode s-set char-width char-ht*/
                               INIT(2, 3, 193, 120, 90 );
```

```

CALL CHKATT(5,KTEXT_ATTRS);      /* Specify key text attributes */
```

2 specifies that the key text should appear in red

- 3** specifies that the text should use mode-3 (vector) text. (See the description of the GSCB call in the *GDDM Base Application Programming Reference* book.)
- 193** identifies the vector symbol set to be used for the key text. (See the description of the GSLSS call in the *GDDM Base Application Programming Reference* book.)
- 120** expresses the character width as a percentage of the width of the character grid. On IBM 3270 family devices the default character grid is the size of a hardware cell. For mode-3 (vector) symbols on these devices, the size specification affects the width of each symbol, for mode-2 (image) symbols it affects only their spacing, and mode-1 symbols are not affected. On other devices (family-4 printers), the width specification determines the width of the symbols in all modes. For further information, see the discussion of the matching GDDM function 'GSCB' in the *GDDM Base Application Programming Guide* book.
- 90** specifies the height of the character box in terms of the width specification and the character grid size. It is the percentage of the width multiplier that is to be applied to the height. The default is 100%. In the example, the height is increased by only 90% of the increase applied to the width, making the character box slightly fatter in shape than the default.

The attributes are always specified in the given order, starting with color. For example, if the attribute count is 2, then the specified attributes are color and mode. There are several similar calls that specify attributes for other forms of chart text (main heading, axis titles, labels, and chart notes).

Chart axis calls

By default, a PGF business chart has an x axis along the bottom of the chart and a y axis up the left-hand side of the chart. Various calls can be made to alter the axis positions.

Orienting the axes, using call CHSET

Orientation affects all types of charts except tower charts, and applies to both primary and secondary axes.

```
CALL CHSET('YVERTICAL');           /* Make y axis vertical */
```

makes the y axis vertical, and the x axis horizontal, makes pie charts and Venn diagrams horizontally aligned, and aligns components (data groups) of table charts in vertical columns.

```
CALL CHSET('XVERTICAL');           /* Make x axis vertical */
```

makes the x axis vertical and the y axis horizontal, makes pie charts and Venn diagrams vertically aligned, and aligns components (data groups) of table charts in vertical columns.

```
CALL CHSET('ZVERTICAL');           /* Make x axis vertical */
```

and aligns components (data groups) of table charts in horizontal columns. All other chart types are arranged as for YVERTICAL.

Note that for all keyword option settings, **only the first four characters are required**. Extra characters may be added to aid readability of the program. So, CALL CHSET('XVER') is the minimum required.

Setting explicit axis intercepts, using calls CHXSET or CHYSET

Options that affect only one axis are of the form CALL CHXSET('option') or CHYSET('option'), rather than the more general CALL CHSET('option'). To specify where a given axis should intercept the other axis, one of these four options can be set:

```
CALL CHXSET('LOWAXIS'); /* x axis intercepts y axis at bottom */
                        /* or left-hand side of the plot area */
                        /* (This is the default position)      */

CALL CHXSET('MIDDLE'); /* x axis intercepts y axis in the      */
                        /* middle of the plot area              */

CALL CHXSET('HIGHAXIS'); /* x axis intercepts y axis at top or */
                          /* right-hand side of the plot area */

CALL CHXSET('INTERCEPT'); /* x axis position controlled by */
                             /* CHYINT                          */
```

Figure 6 shows the positioning options for a y axis in a tower chart.

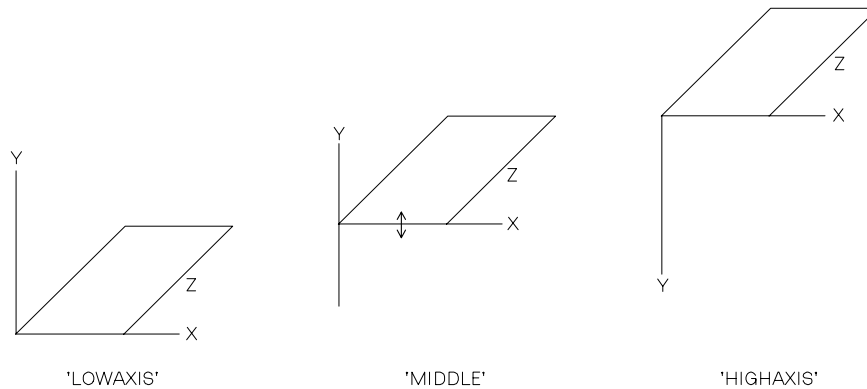


Figure 6. y axis position options in tower charts

Figure 7 on page 39 shows the flexibility provided by interception points, in combination with axis-positioning options.

Y Axis Position X Axis Position				
	L	M	H	YINT
L				
M				
H				
XINT				

Figure 7. x and y axis position options

As an example of the use of the INTERCEPT option, suppose that the primary axes have been selected (by default or specific calls), and that the y axis is required to intercept the x axis at the value $x=3$. A pair of statements is required, one to say that the axis is to be positioned by intercept, the other to say what that intercept is:

```
CALL CHYSET ('INTERCEPT'); /* The y-axis position is given by a */
                             /* matching CHXINT call */
CALL CHXINT (3);             /* The y axis crosses the x axis at */
                             /* an x coordinate of 80 */
```

Another example is shown below, again assuming that the primary axes are selected:

```
CALL CHXRNG (-5,7);
CALL CHYRNG (0,10);
CALL CHYSET ('INTERCEPT');
CALL CHXINT (-2);
CALL CHXSET ('INTERCEPT');
CALL CHYINT (3);
```

In this case, the y axis intercepts the x axis at the value $x=-2$ and the x axis intercepts the y axis at the value $y=3$, as shown in Figure 8 on page 40.

PG routines that apply to most chart types

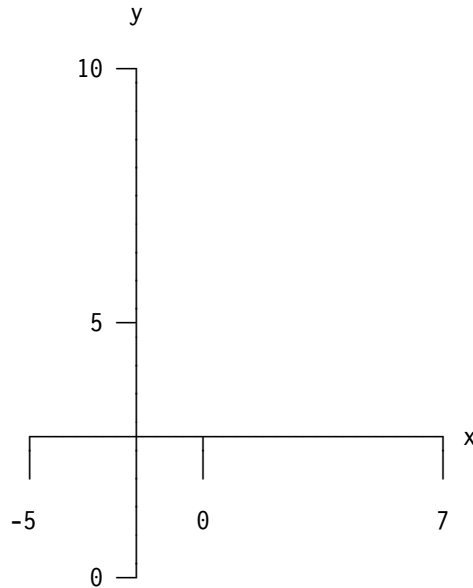


Figure 8. Primary axis intercept

It is important to note that these calls refer to the axes **selected** at the time the specification is made. Thus, primary axes can be forced to intercept secondaries, and **conversely**. An example of intercepts using secondary axes follows.

Assume that two sets of axes are needed for plotting two sets of data, using different scales. The default positions for the primary axes and the secondary y axis are used, but suppose you want to specify the intersection of the secondary x and y axes. The following sequence of calls applies:

```
CALL CHXRNG (0,10);          /* Set primary x range          */
CALL CHYRNG (0,50);          /* Set primary y range          */
CALL CHXSEL (2);             /* Select secondary x axis      */
CALL CHYSEL (2);             /* Select secondary y axis      */
CALL CHXRNG (0,100);         /* Set secondary x range        */
CALL CHYRNG (2,8);           /* Set secondary y range        */
CALL CHXSET ('INTERCEPT'); /* Set intercept of secondary x axis */
CALL CHYINT (6);             /* on secondary y axis          */
```

The following sequence could then be used to plot two sets of data in respect to the axes:

```
CALL CHXSEL(1);              /* Select primary x axis        */
CALL CHYSEL(1);              /* Select primary y axis        */
CALL CHPLOT(1,n1,x1,y1);     /* Plot first data set          */
CALL CHXSEL(2);              /* Select secondary x axis      */
CALL CHYSEL(2);              /* Select secondary y axis      */
CALL CHPLOT(1,n2,x2,y2);     /* Plot second data set         */
```

Figure 9 shows the resulting axis positions on the chart. The data points, some axis labels, scale marks, and datum and/or grid lines have been omitted for clarity in the figure.

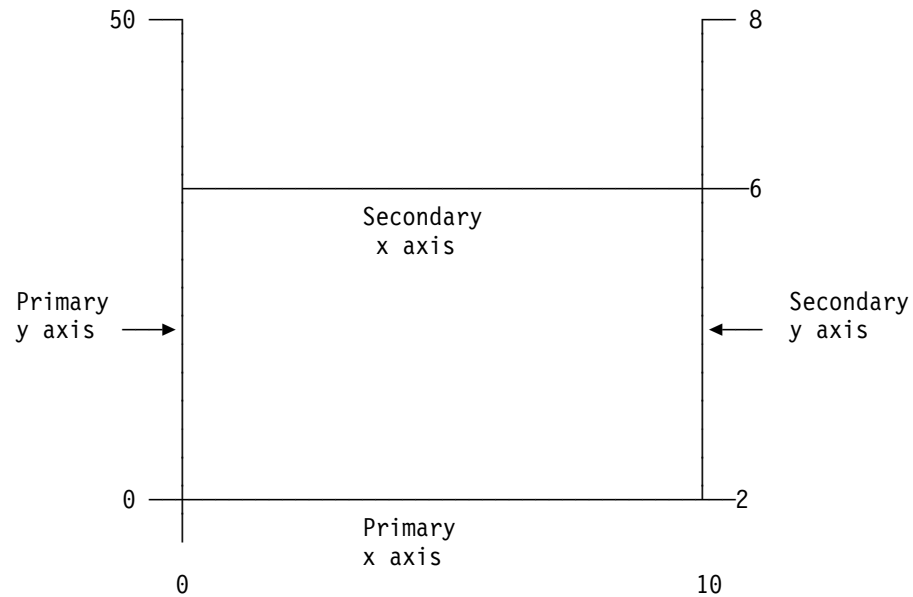


Figure 9. Secondary axis intercept

The attributes of axis lines are set by this call:

```
DCL AXIS_ATTRS(12) FIXED BIN(31) INIT(
  /* Primary x axis:      color      line type      line width */
                        2,          7,          2,
  /* Primary y axis:      color      line type      line width */
                        4,          7,          2,
  /* Secondary x axis:    color      line type      line width */
                        3,          1,          1,
  /* Secondary y axis:    color      line type      line width */
                        6,          1,          1);

CALL CHAATT(12,AXIS_ATTRS); /* Set attributes for all 4 axes */
```

Secondary axes are discussed in Chapter 6, “Using PG routines to create complex charts” on page 107. This call permits the setting of different attributes for each of the four axes. The count can be set to any value between 1 and 12. If you want to set attributes for just the secondary x axis, you must give a count of 9 and also pass values for the first six attributes.

Setting the axis scale, using calls CHXSET or CHYSET

Each axis may be linear (the default) or logarithmic. These calls set the option:

```
CALL CHXSET('LINEAR'); /* Make the x axis linear */
CALL CHYSET('LOGARITHMIC'); /* Make the y axis logarithmic */
```

Setting the axis range

The range of each axis is, by default, between the limits of the specified data. Allowing the range to default is known as **autoscaling** or **autoranging**.

PG routines that apply to most chart types

By default, PGF **extends the ranges to include zero**. This is a common cause of misunderstanding for inexperienced users of PG routines. If, say, the x values are 1988,1989,1990,1991 (representing years), then the default x range is **not** 1988-1991, it is 0-1991, giving quite the wrong form of chart. To suppress the inclusion of zero when autoranging, this option is requested:

```
CALL CHXSET('NOFORCEZERO');      /* Do not extend x range to */
                                  /* include the point x=0      */
```

The option can be reset by requesting:

```
CALL CHXSET('FORCEZERO');         /* Extend x range to include 0 */
```

An explicit axis range can be set by this call:

```
CALL CHXRNG(4000.0,12500.0);      /* x axis runs from x=4000 */
                                  /* to x=12500              */
```

All CHX... calls except CHXDLB have CHY... equivalents for the other axis. If an explicit range is requested for an axis, any force-zero option is ignored. The relationships between axis range and axis orientation are shown in Figure 10.

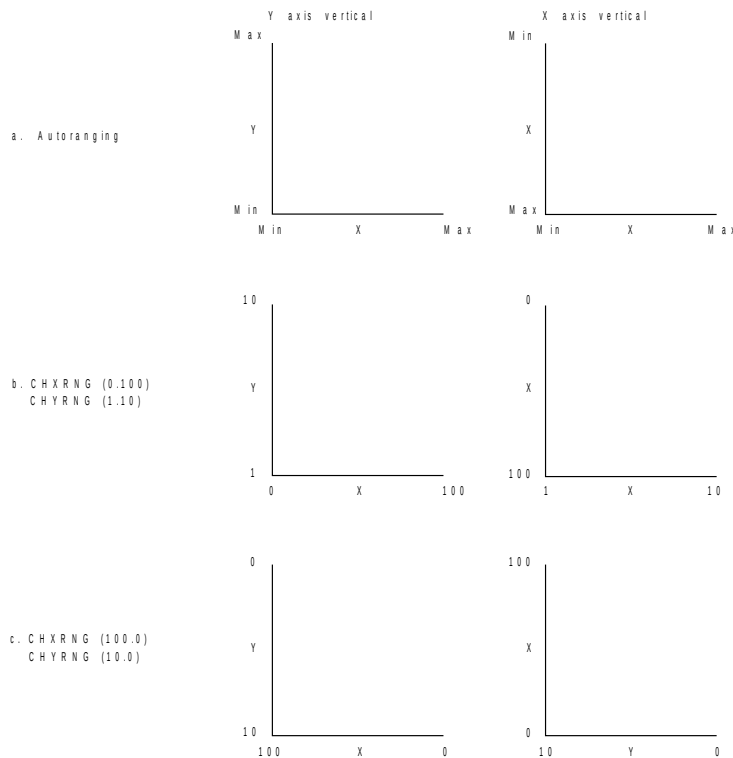


Figure 10. Axis range and orientation

Setting axis titles, using calls CHXTTL and CHYTTL

To specify an axis title, this call, or its y-axis equivalent CHYTTL, is used;
CALL CHXTTL(31,'ELAPSED TIME (IN MILLI-SECONDS)');/*x-axis title*/

The text attributes for all axis titles are specified with a CHTATT call:

```
DCL TTEXT_ATTRS(5) FIXED BIN(31)
/*                               Color mode s-set char-width char-ht*/
                               INIT(2, 2, 196, 130, 80 );

CALL CHTATT(5,TTEXT_ATTRS); /*Specify axis-title text attributes*/
```

The position of the axis title can be set, using one of these options:

```
CALL CHXSET('ATCENTER');/*Title in center of axis (the default) */
CALL CHXSET('ATEND'); /*Title at right (h. axis), top (v.axis)*/
CALL CHXSET('ATABOVE'); /*Title above a vertical axis */
```

You can specify that a semicolon in an axis title causes a line break, using the TLBREAK option on the CHSET call:

```
CALL CHSET('TLBREAK'); /* Line breaks at semicolon */
```

Program example for axis options

The next program example illustrates the effect of the various axis options.

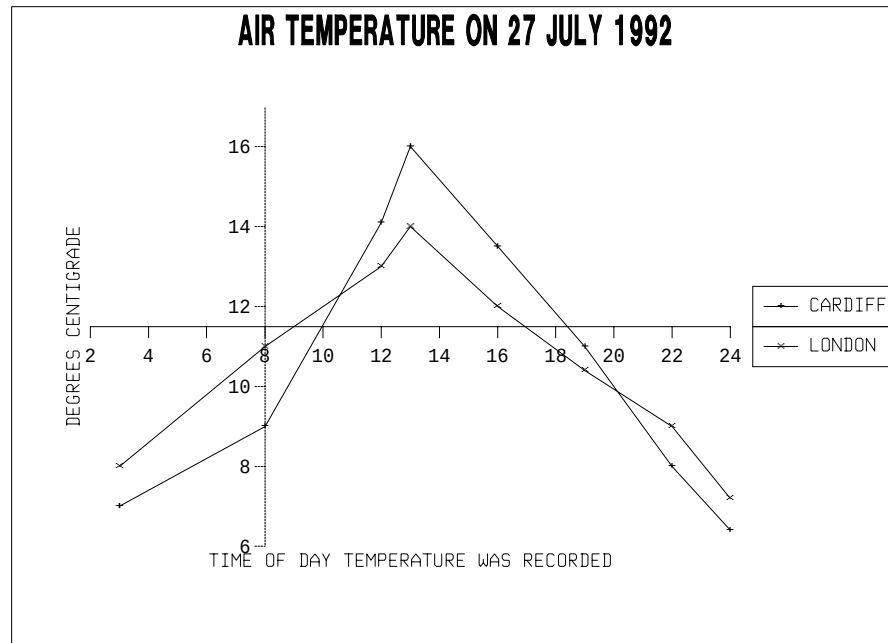


Figure 11. Output from axis options program

```
AXIS: PROC OPTIONS(MAIN);
CALL FSINIT; /* Initialize GDDM */
DCL X08(8) FLOAT DEC(6) INIT(3,8,12,13,16,19,22,24); /* x data */
DCL Y08(16) FLOAT DEC(6) INIT(8,11,13,14,12,10.4,9,7.2, /* y data */
7, 9,14.1,16,13.5,11 ,8,6.4);
/* Attributes: color char-mode s-set width height*/
DCL HEAD_ATTTS(4) FIXED BIN(31) INIT(2, 3, 193, 200, 100 );
DCL LAB_ATTTS(2) FIXED BIN(31) INIT(4, 2 );
DCL BOX_ATTTS(1) FIXED BIN(31) INIT(5 );
```

PG routines that apply to most chart types

```
CALL GSLSS(2,'ADMUUKSF',193); /* Load symbol set for heading */
CALL CHHATT(4,HEAD_ATTTS); /* Set attributes for chart heading */
CALL CHLATT(2,LAB_ATTTS); /* Set attributes for axis labels */
CALL CHBATT(1,BOX_ATTTS); /* Set attributes for chart box */
CALL CHHEAD(34,'AIR TEMPERATURE ON 27 JULY 1992');/*Cht heading */
CALL CHSET('CBOX'); /* Request frame around the chart */
CALL CHSET('KBOX'); /* Request frame around the legend */
CALL CHVMAR(7,15); /* Vertical margins: left 7,right 15*/
CALL CHKEY(2,7,'LONDON CARDIFF');/* Specify the 2 keys for legend */

/*****
/* Specify the axis options */
*****/

CALL CHYRNG(6.0,17.0); /* Set explicit y range (6-17 degrees) */

CALL CHXSET('NOFORCEZERO');/* Do not extend x axis to include x=0 */

CALL CHXSET('MIDDLE'); /* Place x axis in middle of y axis */
CALL CHYSET('INTERCEPT'); /* Position y axis by intercept.... */
CALL CHXINT(8); /* .... at x coordinate = 8 */

/* x axis y axis */
DCL AX_ATTTS(6) FIXED BIN(31) INIT(4, 0, 1, 6, 1, 2);
CALL CHAATT(6,AX_ATTTS); /* Set attributes for x axis & y axis */

CALL CHXTTL(36,'TIME OF DAY TEMPERATURE WAS RECORDED');
/*x-axis title*/
CALL CHYTTL(18,'DEGREES CENTIGRADE'); /*y-axis title*/

DCL TITL_ATTTS(1) FIXED BIN(31) INIT(5); /* Turquoise axis titles */
CALL CHTATT(1,TITL_ATTTS); /* Axis title attributes */

/*****
/* End of the axis options */
*****/

CALL CHPLOT(2,8,X08,Y08);/* Plot 2 components, each with 8 points */

CALL CHTERM; /* Terminate PGF, free associated storage*/
CALL ASREAD(TYPE,MOD,COUNT); /* Send out business chart to device */
CALL FSTERM; /*Terminate GDDM, free associated storage*/

%INCLUDE(ADMUPINA); /* Include GDDM and PGF entry points */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END AXIS;
```

The effect of the “axis” calls used in the above program can be observed by comparing the program’s output (shown in Figure 11 on page 43) with that of the previous program example (shown in Figure 4 on page 26).

Axis label and tick-mark calls

Axes are usually subdivided by **tick marks** or **scale marks**. Between each pair of major tick marks you may (optionally) specify one or more minor tick marks. The y axis of Figure 15 on page 62 shows both types of tick marks.

Labels are associated one-for-one with the major tick marks.

Setting the tick-mark interval, using calls CHXTIC and CHYTIC

By default, PGF places the major ticks at specific multiples of a power of ten, at intervals of approximately nine character grid units. The user can specify the tick interval by using this call:

```
CALL CHXTIC(100.0,4.0);/*Place major ticks every 100 x-units apart*/
                        /* with 4 minor ticks between each pair */
```

This is the scheme for linear axes. CALL CHXTIC(0.0,1.0) would give default major ticks with one minor tick between each pair.

For logarithmic axes, the default major tick positions are at the successive powers of 10. Minor ticks are not allowed, but extra major ticks can be generated at specific multiples of powers of 10:

```
CALL CHYSET('LOGARITHMIC'); /* y axis is logarithmic */
CALL CHYTIC(2.0,0.0); /* Extra major ticks should be generated at */
                        /* every 2*10**N, 4*10**N, 6*10**N, 8*10**N */
```

The first parameter specifies which multiples of powers of 10 should receive the tick marks. A setting of 3, for example, would give extra ticks at 3, 6, or 9 times each power of 10. The second parameter is ignored for logarithmic axes.

The style of tick mark can be set by one of these calls:

```
CALL CHXSET('NTICK'); /* Tick marks outside the plotting area */
                        /* (This is the default style) */
CALL CHXSET('PTICK'); /* Tick marks inside the plotting area */
CALL CHXSET('XTICK'); /* Tick marks crossing the axis */
CALL CHXSET('PLAIN'); /* No tick marks (but labels remain) */
```

Figure 12 on page 48 shows the different styles of tick marks permitted.

Positioning the labels, using calls CHXSET and CHYSET

By default, the axis labels are placed adjacent to the major tick marks (as in Figure 15 on page 62). When the axis represents a time-scale and the labels are, perhaps, days or months, it may be desirable to have the labels placed between the major tick marks. This is the option:

```
CALL CHXSET('LABADJACENT');/*Labels are placed next to tick marks*/
                        /* (this is the default) */
CALL CHXSET('LABMIDDLE'); /*Labels are placed between tick marks*/
CALL CHXSET('NOLAB'); /* No labels on the axis */
```

Specifying the type of label

The default type of label is numeric (both axes of Figure 15 on page 62 have numeric labels). It is possible, though, to place user-specified alphanumeric labels on or between the successive tick marks or to request date labels of various types. Date labels in abbreviated-month format are shown on the x axis of Figure 22 on page 81.

These are the calls affecting the label type:

Alphanumeric (user) labels: There are two ways of putting user labels on the x axis. You can specify labels for bars using the CHXLAB call:

```
CALL CHXLAB(6,10,  
  'HARDWARE RECORDS LIGHTING FASHION SPORTS TEA SHOP ');  
/* Set user labels for barchart x axis */  
/* 6 label specifications, each of 10 characters*/
```

PGF selects each label in turn. The first label is placed under the first major tick mark, the second under the second major tick mark, and so on.

The other method is to use the CHXDLB call, which puts labels at particular values, rather than at each successive tick mark:

```
DECLARE XVALUES(10) FLOAT DEC(6) INIT  
( 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0);  
  
DECLARE XLABELS(10) CHARACTER(4) INIT  
( '1Q83', '2Q83', '3Q83', '4Q83', '1Q84',  
  '2Q84', '3Q84', '4Q84', '1Q85', '2Q85');  
CALL CHXDLB(10,4,XVALUES,XLABELS); /*10 labels 4 characters long*/
```

This example specifies that the first label is to be placed under the tick mark at x=1.0, the second under the tick at x=2.0, and so on. If there is no tick mark at a specified x value, the label is placed under the nearest tick, whenever there is one within 10% of the tick mark interval. If there is no tick mark within this 10% range, the label is not used.

In both calls, a “;” is a new line character, giving multiline labels. A semicolon must be specified as “;”.

Numeric labels:

```
CALL CHXSET('NUMERIC'); /* Numeric labels for the x axis */  
/* (this is the default) */
```

Date labels:

```
CALL CHXDAY(7); /* Day labels are used on successive */  
/* tick marks. The parameter 7 shows */  
/* that the first label is to be Sunday */  
/* (the 7th day), the next label is to */  
/* be Monday, and so on... */  
CALL CHXMTH(2); /* Month labels are used on successive */  
/* tick marks. The 1st label is to be */  
/* February (the 2nd month) */
```

Both alphanumeric and date labels are reused in a cycle if there are insufficient labels for every major tick mark. There are matching calls for the y axis, as always (for example, CALL CHYMTH(5));

Date labels may appear in three different forms: in full, first three letters only, or first letter only. This option decides which:

```
CALL CHSET('ABREV');    /* Date labels appear as 3-letter    */
                        /* abbreviations. (this is the default) */

CALL CHSET('FULL');     /* Date labels appear in full      */

CALL CHSET('LETTER');   /* Date labels appear as 1-letter  */
```

Note that this is a CHSET option (because you would not have date labels on both axes) - it is not a CHXSET/CHYSET option. Note also that ABREV is (mis-)spelled, as shown.

Setting label attributes, using call CHLATT

To set label attributes for both axes, you must make this call:

```
DCL LTEXT_ATTRS(6) FIXED BIN(31)
/*                               Color mode s-set width height angle*/
                               INIT(2,  2, 196,  80, 100, 3000);

CALL CHLATT(6,LTEXT_ATTRS);    /* Specify label text attributes */
```

The angle is given in 1/100ths of a degree, and can range from -36000 to 36000. The example specifies that the note is to be rotated by 30 degrees in a counterclockwise direction. To obtain a clockwise rotation, you would make the value negative. The angle specification is ignored if the text of the label is mode-1.

To set the label attributes for the x axis or y axis separately, use the CHXLAT or CHYLAT call. These calls have the same parameters as CHLATT.

Suppressing label punctuation, using call CHSET

By default, labels (and bar values) of 1,000 and over appear using an installation-defined punctuation scheme. Three methods are widely used:

```
Period decimal  1,205.34
Comma decimal   1.205,34
French          1 205,34
```

To suppress or reinstate the punctuation, these calls are used:

```
CALL CHSET('PGFS');      /* Suppress all punctuation, except */
                        /* for the decimal point            */

CALL CHSET('NPGFS');     /* Use the installation-defined     */
                        /* default punctuation method      */
```

The 'PGFS' option is useful when axis labels are to represent years (for example, 1992). Without it, the year would appear as 1,992 or 1.992 or 1 992 (according to the installation default).

CHSET applies to both axes. You can control the labeling of each axis independently by using the PGFS and NPGFS options of the CHXSET and CHYSET calls. CHYSET also sets the punctuation of bar values in bar charts.

Blanking the label space, using call CHSET

When an axis crosses the plotting area, or when the CBACK chart background option is in use, it may be convenient to blank out the space to be occupied by the labels. This is done by issuing:

```
CALL CHSET('BLABEL');          /* Blank label space */
```

The default can be reinstated by issuing:

```
CALL CHSET('NBLABEL'); /* Do not blank label space (the default)*/
```

Grid line calls

The user can specify **grid lines** on each of the axes. These lines start from the major tick marks of the axis and cross the plotting area perpendicular to the axis involved. So, x grid lines pass through the x-axis major tick marks and are drawn parallel to the y-axis. Grid lines on both axes are shown in Figure 12. These are the relevant calls:

```
CALL CHXSET('GRID');          /* Draw x-axis grid lines */
```

```
CALL CHYSET('NOGRID');       /* No grid lines on y axis (default) */
```

The attributes of grid lines are set by calling CHGATT (see description of the similar call CHAATT in “Chart axis calls” on page 37).

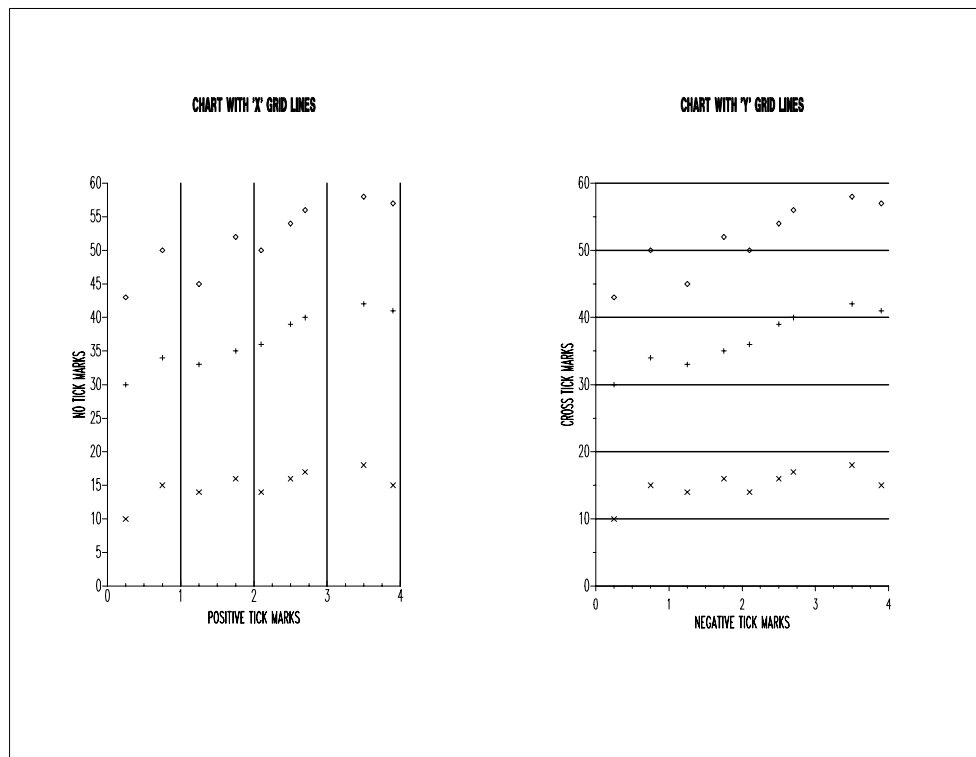


Figure 12. Grid lines and tick marks

Datum line calls

A **datum line** is a line drawn parallel to one of the two axes. An x datum line passes through a specified coordinate on the x axis and is drawn parallel to the y axis. A y datum line is drawn parallel to the x axis and passes through a specified y coordinate.

These are the two calls involved:

```
CALL CHXDTM(24.0);    /* Draw x datum line parallel to the    */
                      /* y axis, crossing the x axis at (x=24) */

CALL CHYDTM(-15.0);   /* Draw y datum line parallel to the    */
                      /* x axis, crossing the y axis at (y=-15) */
```

When a datum line is drawn before the chart is plotted, it is known as a **state-1 datum line**. When drawn after the chart has been plotted, it is a **state-2 datum line**.

A state-1 y datum line is used as a reference line for all surface charts, histograms, and bar charts. In other words, the first component of the chart is shaded from the datum line rather than from the x axis.

Whether or not shading has been specified, the bars (on a bar chart) and the risers (on a histogram) start from the state-1 y datum line rather than from the x axis.

An example of a state-1 datum line that affects the chart shading is shown in Figure 22 on page 81.

The x datum lines and state-2 y datum lines merely add extra lines to the chart. They have no effect on the shading.

The line attributes for all datum lines can be set by calling CHDATT:

```
/*                      Color line type line width */
DCL DTM_ATTRS FIXED BIN(31) INIT( 2, 0, 2);
CALL CHDATT(3,DTM_ATTRS);    /* Set datum line attributes */
```

Heading calls

The main chart heading is specified by this call:

```
CALL CHHEAD(45,'SALES FIGURES BY REGION;(MARCH - AUGUST 1992)');
                      /* Request two-line chart heading */
```

A ; (semicolon) in the heading text is used as a line-break character. If a real semicolon is required in the text, two consecutive semicolons must be supplied. This causes a single semicolon to appear (and no line break occurs).

The text attributes for the heading are set by calling CHHATT:

```
DCL HTEXT_ATTRS(5) FIXED BIN(31)
/*                      Color mode s-set char-width char-ht*/
INIT(2, 2, 196, 180 75 );

CALL CHHATT(5,HTEXT_ATTRS); /* Specify heading text attributes */
```

The position of the heading can be set by one of these calls:

```
CALL CHSET('HCENTER');/* Each line of the heading is centered */
/* In the chart area (this is the default)*/
CALL CHSET('HLEFT'); /* Each line justified left of chart area */
CALL CHSET('HRIGHT'); /* Each line justified right of chart area*/
```

The heading is usually placed at the top of the chart, but it can be moved to the bottom:

```
CALL CHSET('HTOP'); /*Heading at the top (this is the default)*/
CALL CHSET('HBOTTOM');/*Heading at the bottom */
```

An example of a chart heading with specified text attributes is given in Figure 4 on page 26 and the accompanying program.

Chart note calls

Annotative text (known as chart notes) can be added anywhere on the chart. Examples of chart notes are the “gross turnover figures” shown in Figure 27 on page 97 and the yellow text in the corners of Figure 28 on page 103.

Specifying note text, using call CHNOTE

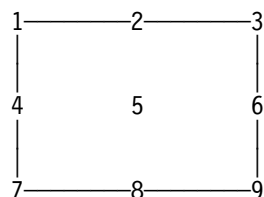
The following call specifies the text of a chart note:

```
CALL CHNOTE('C1',14,'IN MILLIMETERS'); /* Add 14-character note */
```

These are the three parameters:

- 'C1' is the **base-position code**. The scheme for positioning notes is complicated but comprehensive, using the CHNOFF call (described below). The first character in the position code states what units the CHNOFF call uses to specify the note position. These are the possibilities:
 - C Both offsets expressed in character-grid units (for which the default on IBM 3270 family terminals is hardware cells). In other words, horizontal offsets are expressed in columns and vertical offsets in rows.
 - H Horizontal offset in chart-axis units, vertical offset in character grid units.
 - V Vertical offset in chart-axis units, horizontal offset in character grid units.
 - Z Both offsets in chart-axis units.

The second character in the position code specifies which part of an imaginary note rectangle surrounding the note should be placed at the note position. Any corner, the middle of any side, or the center of the note rectangle can be chosen. This is the scheme:



So, a position code of 'C1' states that the top-left corner of the note should be placed at the note position (which was specified by a CHNOFF using character-grid coordinates for both the offsets).

- The second parameter is simply the length of the note text.

- The third parameter contains the text of the note.

Specifying the position of a chart note, using call CHNOFF

The position of a chart note is determined by this call:

```
CALL CHNOFF(24.0,40.0); /* Offset of next note is 24 units to */
                        /* the right, and 40 units upward */
```

These offsets can be expressed in character grid units or in chart axis units, as specified by the note position code (described above).

The interpretation of the offset varies according to the type. Character-grid offsets are taken from the bottom left-hand corner of the chart area. Axis-unit offsets are taken from the (0,0) origin of the chart. In other words, for a Z-type base position code (both offsets in axis coordinates), the offset specified above would position the note at (x=24,y=40), wherever these points might be in the plotting area.

The position must be specified **before** the corresponding CHNOTE call.

Setting note text attributes, using call CHNATT

Chart note text attributes are set by calling CHNATT:

```
DCL NTEXT_ATTRS(6) FIXED BIN(31)
/*                               Color mode s-set width height angle*/
                               INIT(0,  2,  0,   100,  100,  4500);

CALL CHNATT(6,NTEXT_ATTRS);    /* Specify note text attributes */
```

By default, all chart notes are horizontal. You can rotate a note using a sixth value in the array parameter. The angle is given in 1/100ths of a degree. The example specifies that the note is to be rotated by 45 degrees in a counterclockwise direction. To get a clockwise rotation, you would make the value negative. The rotation takes place about the base position specified in the first parameter of the CHNOTE call. The angle specification is ignored when the text of the note is mode-1.

Blanking the note space, using call CHSET

When notes are written on top of the plotting area, it may be convenient to blank out the space to be occupied by the notes. This is done by issuing:

```
CALL CHSET('BNOTE');          /* Blank note space */
```

An example of the BNOTE option is shown in Figure 28 on page 103 (the three numbers within the circles are chart notes). The default can be reinstated by issuing:

```
CALL CHSET('NBNOTE'); /* Do not blank note space (the default) */
```

Example of using chart notes

This example may help to clarify the method of positioning notes.

PG routines that apply to most chart types

```
CALL CHXRNG(0.0,100.0);    /* x-axis range (the horizontal axis) */
CALL CHYRNG(20.0,35.0);   /* y-axis range (the vertical axis) */
    and so on...          /* Various other options */
CALL CHSURF(2,8,X_DATA,Y_DATA); /* Plot a surface chart */
/* Chart notes may not be written until after the 1st plotting call */

/*                                Color  ch-mode s-set-id  ch-size*/
DCL NOTE_ATTRS1(4) FIXED BIN(31) INIT(2,      3,      193,      150);
DCL NOTE_ATTRS2(4) FIXED BIN(31) INIT(6,      2,       0,       100);

CALL CHNATT(4,NOTE_ATTRS1); /* Set attributes for any further */
                           /* CHNOTE calls */

CALL CHNOFF(45.0,20.0);    /* Horiz. offset:45, vert. offset:20 */
CALL CHNOTE('C7',22,'THIS IS THE FIRST NOTE');
                           /* The bottom left of the note is to */
                           /* be positioned 45 hardware cells */
                           /* Right of the bottom l-hand corner */
                           /* and 20 cells above it */

CALL CHNOFF(90.0,27.0);    /* Horiz. offset:90, vert. offset:27 */
CALL CHNOTE('Z9',23,'THIS IS THE SECOND NOTE');
                           /* The bottom right of the note is */
                           /* positioned at (x=90,y=27). The */
                           /* same attributes are used as for */
                           /* the previous note, namely those */
                           /* specified in 'NOTE_ATTRS1' */

CALL CHNATT(1,NOTE_ATTRS2); /* Change note color to 6 (yellow), */
                           /* the other 3 attributes remain */
                           /* unchanged (= 3, 193, 150) */

CALL CHNOFF(50.0,12.0);    /* Horiz. offset:50, vert. offset:12 */
CALL CHNOTE('H5',22,'THIS IS THE THIRD NOTE');
                           /* The center of the note is to be */
                           /* positioned at x=50, 12 hardware */
                           /* cells above the bottom of the */
                           /* chart area */

CALL ASREAD(TYPE,MOD,COUNT); /* Send out chart and notes */
```

So, the points to remember are:

- Chart notes must be written in state-2 (in other words, **after** the first plotting call). This is because they may need axis units to position them, and these are established only at plotting time.
- The text attributes and position of the note must be established before the CHNOTE call.
- Each of the two note offsets may be interpreted differently according to the note position code.

Chart termination call

When you have finished using the PG routines, you issue this call:

```
CALL CHTERM; /* Terminate PG routines, free associated storage */
```

CHTERM terminates the PG routines, and frees the associated storage. You may choose to issue CHTERM **before** the ASREAD (or FSFRCE) that sends the chart to the device. GDDM then has more storage available to it for constructing the data stream.

If you have already reached the end of your program, there is no need to issue a CHTERM. The FSTERM frees all the storage, both for GDDM and for the PG routines.

Chapter 5. PG routines specific to each chart type

Most of the calls described in the previous chapter can be used with all chart types. This chapter covers each type of chart separately, and the controls that are specific to it. The different forms of each type are discussed, and any further options that can be set are described.

Line graphs and scatter plots

The main options to be chosen are:

- Whether the plot points should be indicated by markers
- Whether the lines themselves should be drawn (if not, a scatter plot results)
- Whether curve fitting should be applied.

The calls and options that affect line graphs and scatter plots are listed below with default options shown first.

CHPLOT	Plots the line graph
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'LINES' 'NOLINES'	Line graph or scatter plot
'NOCURVE' 'CURVE'	Straight or curved lines
'MARKERS' 'NOMARKERS'	Markers on plotted points
'NOMSCALE' 'MSCALE'	Scaling of markers
CHMARK	Specify marker table values
CHMKSC	Specify marker scale factors
CHFINE	Specify smoothness of curve
CHCOL	Specify basic color table for markers
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table

Suppressing the markers or the lines themselves, using call CHSET

By default, the plot points are indicated by markers. These can be suppressed or reinstated by these two calls:

```
CALL CHSET('NOMARKERS');    /* Suppress markers on line graph */
CALL CHSET('MARKERS');      /* Reinstat markers on line graph */
```

The lines joining the data points can themselves be suppressed, to give a scatter plot:

```
CALL CHSET('NOLINES');      /* Suppress lines on line graph */
CALL CHSET('LINES');        /* Reinstat lines on line graph */
```

An example of a scatter plot is shown in Figure 12 on page 48.

Curve fitting, using call CHSET

Curve fitting can be specified both for line graphs and surface charts. This is the option:

```
CALL CHSET('CURVE');           /* Fit curve to data points */
                                /* of each component          */

CALL CHSET('NOCURVE'); /* Suppress curve fitting (the default) */
```

No mathematical significance should be attached to the fitted curve. The facility is a cosmetic one that may provide a more attractive chart. Note that the curve **passes through all the data points**, not between them as with a typical best-fit curve.

Setting curve smoothness, using call CHFINE

The smoothness of the fitted curve (that is, the number of straight-line segments interpolated between each pair of data points) can be set by this call:

```
CALL CHFINE(6);                /* 6 straight-line segments are */
                                /* interpolated, to give the effect */
                                /* of a smooth curve              */
```

By default the smoothness is set to 10. Setting a very high value significantly increases the processor time used for the plot, but higher values do not necessarily give you a better curve. This is because the accuracy of the positioning of the ends of each straight-line segment is limited by the fineness with which the display screen or printed page can be addressed.

Setting the line types, using call CHLT

By default, PG routines use solid (that is, continuous) lines for every component of a line graph, but you can create your own **line-type table** – a table of line types to be used for successive components. This is most useful for line graphs, where it aids distinction between the different data lines.

An **attribute table** of line types is established by calling CHLT:

```
DCL LINE_TYPES(5) FIXED BIN(31) INIT(7,1,5,2,3); /*5-entry table */

CALL CHLT(5,LINE_TYPES);          /* Establish line-type table */
```

So, the first component of a subsequently drawn line graph uses line type 7, the next uses line type 1, and so on. When the line graph has more than five components, the table is reused from the start for components six and beyond. The different line types and their associated code numbers are shown in Figure 13 on page 57.

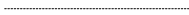
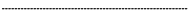
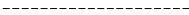
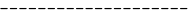
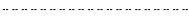
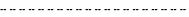
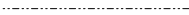
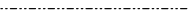
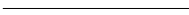
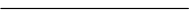
	LINE-WIDTH = 1-PEL		LINE-WIDTH = 2-PEL	
0		(DEFAULT)		0
1		(DOTTED)		1
2		(SHORT-DASHED)		2
3		(DASH-DOT)		3
4		(DOUBLE-DOT)		4
5		(LONG-DASHED)		5
6		(DASH-DOT-DOT)		6
7		(SOLID)		7
8		(INVISIBLE)		8

Figure 13. GDDM line types and line widths

Setting line-widths, using call CHLW

You can create a **line-width table** in a similar way to the line-type table:

```
DCL LINE_WIDTHS(3) FIXED BIN(31) INIT(1,0.3,2); /*3-entry table */
CALL CHLW(3,LINE_WIDTHS);          /* Establish line-width table */
```

The line widths are specified as a multiple of a device-dependent standard. For a 3270 device, the standard width is 1 pixel, for instance, and for an IBM 4250 printer it is 6 pixels. The specified number is rounded down to the nearest pixel. When the resulting value is less than 1 pixel, the lines are 1 pixel wide. When they are more than a device-dependent maximum, the maximum is used. For a 3270 device, the maximum is 2 pixels, and for an IBM 4250 printer it is 600 pixels.

The above calls set up line widths of 1 pixel, 1 pixel, and 2 pixels on a 3270 device, or 6 pixels, 1 pixel, and 12 pixels on a 4250.

Specifying markers, using call CHMARK

Similarly, it is possible to create a **marker table**, but this is needed less often because PG routines provide a default marker table containing the eight GDDM system markers.

							
65	66	67	68	69	70	71	72
							
73	74	75	76	77	78	79	80
							
81	82	83	84	85	86	87	88
							
89	90	91	92	93	94	95	96
							
97	98	99	100	101	102	103	104
							
105	106	107	108	109	110	111	112
							
113	114	115	116	117	118	119	120
							
121	122	123	124	125	126	127	

Figure 14. GDDM-supplied vector marker set ADMDHIMJ

To create a table using these markers in a different way from the default, this call is used:

```
DCL MARKERS(7) FIXED BIN(31) INIT(8,2,7,3,6,4,5);/*7-entry table*/
CALL CHMARK(7,MARKERS);                          /* Establish marker table */
```

The system markers and their associated code numbers are shown in the *GDDM Base Application Programming Reference* book.

You can create your own markers in a symbol set using the Image or Vector Symbol Editor. To use them, you must first load the symbol set with a GSLSS call:

```
DCL MARKERS(6) FIXED BIN(31) INIT(65,66,67,65,68,69);
CALL GSLSS(4,'VMARKS',0); /*Load user marker set called 'VMARKS'*/
CALL CHMARK(6,MARKERS);    /* Establish marker table */
```

All user-created markers must have numbers in the range 65 to 239 (X'41' to X'EF').

Rescaling vector markers, using call CHMKSC

You can rescale any vector marker using the CHMKSC call:

```
DCL SCALES(4) FLOAT DEC(6) INIT(1.0,1.0,1.0,2.0);
                                     /*Scaling factors*/
```

```
CALL CHMKSC(4,SCALES);           /* Establish 4-entry scaling table */
```

The fourth marker in the table established by the CHMARK call are doubled in size. The first three markers are specified explicitly to be normal size. When the marker table has more than four markers, the scaling factors are reused in the normal cyclic fashion. A call to CHMKSC with the first parameter set to zero resets all markers to their normal size.

The GDDM-supplied markers (the ones you get if you specify a number from 1 through 8) cannot be scaled.

GDDM supplies a vector marker set called ADMDHIMJ that you can load and scale like a user-defined set:

```
DCL MTABLE(4) FIXED BIN(31) INIT(67,68,69,70);
DCL SCALES(2) FLOAT DEC(6) INIT(0.75,1.5);
                                     /* Markers alternately small & large */
```

```
CALL GSLSS(4,'ADMDHIMJ',0);       /* Load GDDM vector marker set */
```

```
CALL CHMARK(4,MTABLE);           /* Establish marker table      */
```

```
CALL CHMKSC(2,SCALES);           /* Set up scaling factors    */
```

The markers in this set have numbers from 65 through 127. They are illustrated in Figure 14 on page 58. For this figure they were drawn at three times their default size on a 3279 terminal. Numbers 65 through 73 are vector versions of the system markers 0 through 8. Number 74 is a blank marker.

Setting basic colors, using call CHCOL

It is possible to control colors of the shading patterns and markers on a chart using a **basic color table**. The default-component basic color table runs through the eight colors in sequence: blue, red, pink, green, turquoise, yellow, neutral (which means white on a display unit, or black on a printer), and background (black on a display unit, or white on a printer). To replace this table, the call used is:

```
                                     /*      green  red  blue      */
DCL COLORS(3) FIXED BIN(31) INIT(4, 2, 1); /*3-entry table*/
```

```
CALL CHCOL(3,COLORS);           /* Establish color table */
```

The code numbers used for colors run from 1 (blue) to 8 (background) in the sequence mentioned above. So, the above color table causes the shading and markers (if any) of the first component to be shown in green, the second in red, and the third in blue. The fourth component (if any) appears in green, because the table is used in cyclic fashion, like the other attribute tables.

When the business chart is to appear on a four-color printer, it is normal to choose green, red, and blue (as in the example shown here).

Setting line colors, using call CHLC

The colors of the lines in each component can be controlled separately from the colors of any shading and markers. The CHLC call sets up a **line-color table**:

```

                                /* pink  yellow  turquoise  neutral */
DCL LCOLS(4) FIXED BIN(31) INIT(3,      6,      5,      7);

CALL CHLC(4,LCOLS);                                /* Set line colors */

```

Following this call, any lines in the first component are pink, any in the second are yellow, and so on. For the fifth component the color recycles back to pink.

If there is no CHLC call, the lines use the basic color table, making their colors the same as any shading and markers.

Sometimes it is useful to make all lines the same color for all components. In charts that involve shading, such as surface charts and tower charts, an outline in a contrasting color stands out better than one in the same color as the shading. This is how to do it:

```

DCL CONSTCOL(1) FIXED BIN(31) INIT(7);

CALL CHLC(1,CONSTCOL);    /* Make lines white in all components */

```

The CHLC or CHCOL call affects the color of lines in a line graph, surface chart, or polar chart, and the outlines of bars, towers, and pies. Neither call affects axes, legend boxes, or spider lines.

Line graph example

The following program produces the output shown in Figure 15 on page 62:

```

LINEG: PROC OPTIONS(MAIN);
DCL (TYPE,MOD,COUNT) FIXED BIN(31);    /* ASREAD parameters */
DCL COLORS(3) FIXED BIN(31) INIT(6,2,5); /* Color table entries */
DCL MARKERS(3) FIXED BIN(31) INIT(1,4,3); /* Marker table entries */
DCL LINE_TYPES(3) FIXED BIN(31) INIT(7,2,1); /* L-type table */
                                /* Entries */
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,193,190); /* Heading text */
                                /* attrs */
DCL N_ATTRS(2) FIXED BIN(31) INIT(6,3); /* Note text attributes */
DCL T_ATTRS(4) FIXED BIN(31) INIT(5,3,0,130); /* Ax-title
TEXT ATTRS*/

CALL FSINIT;                                /* Initialize GDDM */
CALL GSLSS(2,'ADMUVGEP',193); /* Load Gothic symbol set */
/*****
/*          Define chart margins          */
*****/
CALL CHHMAR(6,5);    /* 6 rows (bottom), 5 rows (top) */
CALL CHVMAR(10,1);   /* 10 cols (left), 1 col (right) */

/*****
/*          Define chart heading          */
*****/
CALL CHHATT(4,H_ATTRS);    /* Set heading attributes */
CALL CHHEAD(31,'Laboratory Results;(J.G.Lyster)');

```

```

/*****
/*      Define chart drawing attribute tables      */
/*****
CALL CHCOL(3,COLORS);          /* 6=yellow, 2=red, 5=turquoise */
CALL CHMARK(3,MARKERS);       /* 1=cross, 4=diamond, 3=square */
CALL CHLT(3,LINE_TYPES);      /* 7=solid, 2=short-dashed, 1=dots*/

/*****
/*      Define chart legend                        */
/*****
CALL CHSET('KBOX');           /* Legend is boxed */
CALL CHKEYP('H','B','C');     /* Horizontal legend, in the */
                                /* bottom margin, centered */
CALL CHKEY(3,12,'Experiment 1Experiment 2Experiment 3');
                                /* Key text */

/*****
/*      Define axis options                        */
/*****
CALL CHXRNG(0.0,13.0);        /* Define x-axis range */
CALL CHYRNG(0.0,0.801);      /* Define y-axis range */
CALL CHTATT(4,T_ATTRS);       /* Set axis-title attributes */
CALL CHXTTL(30,'Elapsed Time (in milliseconds)'); /*x-axis title*/
CALL CHYTTL(27,'Particle Speed (meters/sec)');   /*y-axis title*/

/*****
/*      Declare chart data                        */
/*****
DCL X_DATA(7)  FLOAT DEC(6) INIT(0,2,4,6,8,10,12); /* x data */
DCL Y_DATA(21) FLOAT DEC(6) INIT(
    0.79, 0.78, 0.74, 0.66, 0.54, 0.37, 0.11, /*1st y component*/
    0.5 , 0.5 , 0.47, 0.42, 0.33, 0.13, 0.01, /*2nd y component*/
    0.36, 0.36, 0.34, 0.31, 0.27, 0.25, 0.24); /*3rd y component*/

/*****
/*      Plotting call                             */
/*****
CALL CHSET('CURV');           /* Request curve fitting */
CALL CHPLOT(3,7,X_DATA,Y_DATA);/*Plot the line graph,which has 3*/
                                /* components, each of 7 elements*/

/*****
/*      Add chart notes                           */
/*****
CALL CHNATT(2,N_ATTRS);       /* Set note attributes */
CALL CHNOFF(52.0,22.0);       /* Set note position */
CALL CHSET('NBOX');           /* Notes are boxed */
CALL CHNOTE('C7',14,'ALPHA PARTICLE');
N_ATTRS(1)=2;                 /* Change note color */
CALL CHNATT(2,N_ATTRS);       /* Set note attributes */
CALL CHNOFF(16.0,20.0);       /* Set note position */
CALL CHNOTE('C7',25,'DE-IONIZED ALPHA PARTICLE');
N_ATTRS(1)=5;                 /* Change note color */
CALL CHNATT(2,N_ATTRS);       /* Set note attributes */
CALL CHNOFF(27.0,12.0);       /* Set note position */
CALL CHNOTE('C7',13,'BETA PARTICLE');

```

PG routines specific to each chart type

```

/*****
/*      Send chart to terminal      */
/*****
CALL CHTERM;          /*Terminate PGF, free storage */
CALL ASREAD(TYPE,MOD,COUNT); /*Send chart to terminal */
CALL FSTERM;          /*Terminate GDDM, free storage */
%INCLUDE(ADMUPINA);    /*Include GDDM & PGF entry points*/
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END LINEG;

```

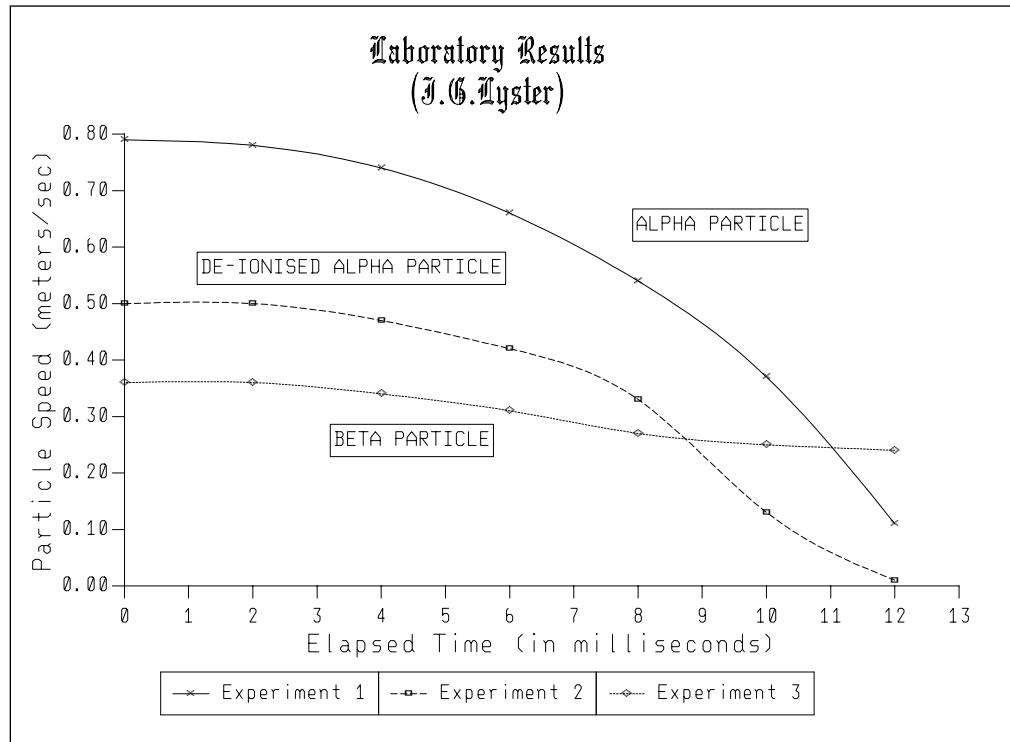


Figure 15. Line graph (with curve fitting)

Surface charts

Surface charts are very similar to line graphs. All the options (except markers discussed in the previous section) apply equally to surface charts. The main difference is that the various components are stacked on top of each other and shaded. As with all stacked chart types, CHSET('RELATIVE') should be specified to request stacking of the components.

These are the main options for surface charts:

- Whether curve fitting should be applied
- Whether the first component should be left unshaded.

The calls and options that affect surface charts are listed below. The default options are given first.

CHSURF	Plots the surface chart
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'NOCURVE' 'CURVE'	Straight or curved lines
'FILL' 'INFILL' 'NOFILL'	Shading method, or ...
'MOUNTAIN' 'NOMOUNTAIN'	... mountain-range shading
CHFINE	Specify smoothness of curve
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table
CHCOL	Specify basic color table for shading
CHPAT	Specify shading-pattern table values

Curve fitting

The same calls are used as with line graphs. CALL CHSET('CURVE'); requests curve fitting and CALL CHFINE(n); specifies the fineness of the curve.

Suppressing shading of the first component, using call CHSET

The first component of a surface chart is left unshaded if this option is specified:

```
CALL CHSET('INFILL');      /* Suppress shading of 1st component */
```

The default filling option (namely, every component shaded) can be reinstated by specifying CALL CHSET('FILL');

top

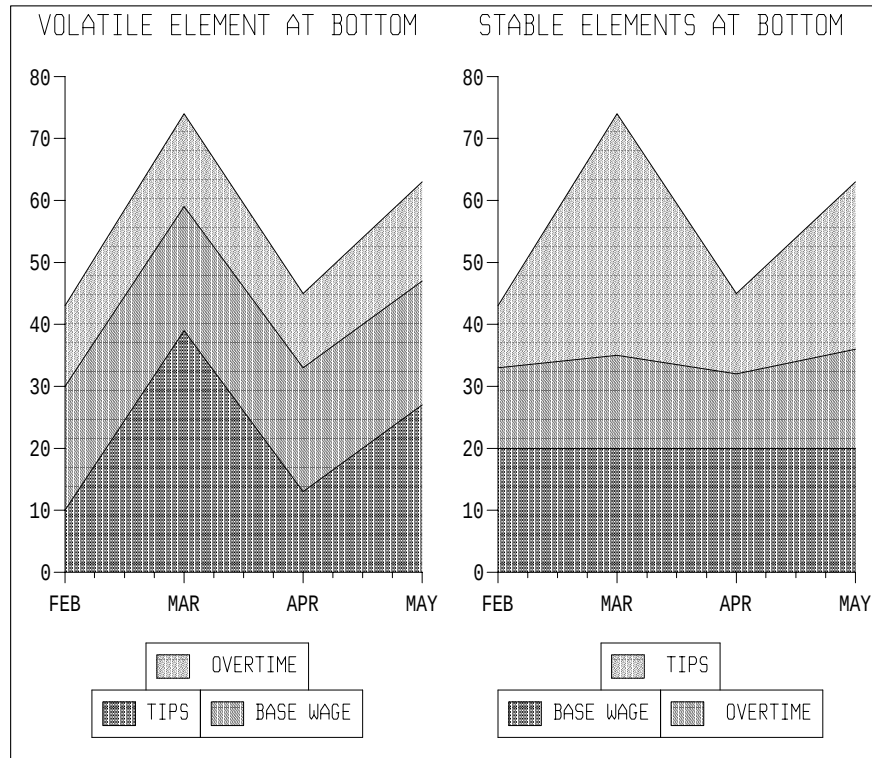


Figure 16. Ordering the components effectively

Ordering the components effectively

Stacked charts are easiest to read when the most stable components are at the bottom. This is most easily explained with an example. Figure 16 shows two surface charts representing the same data. The first chart is hard to read because the turquoise component has a distorting effect on all the other components.

Mountain-range shading

In some circumstances (see “How charts are shaded” on page 29), the shading for two or more components may overlap. This can make the chart confusing. You can then simplify the appearance by specifying mountain-range shading:

```
CALL CHSET('MOUN');          /*Specify mountain-range shading*/
```

Normal shading, which is the default, can be specified by coding the CHSET parameter 'NOMO'.

As Figure 17 on page 65 shows, the additional clarity can come at the expense of obscuring some of the data.

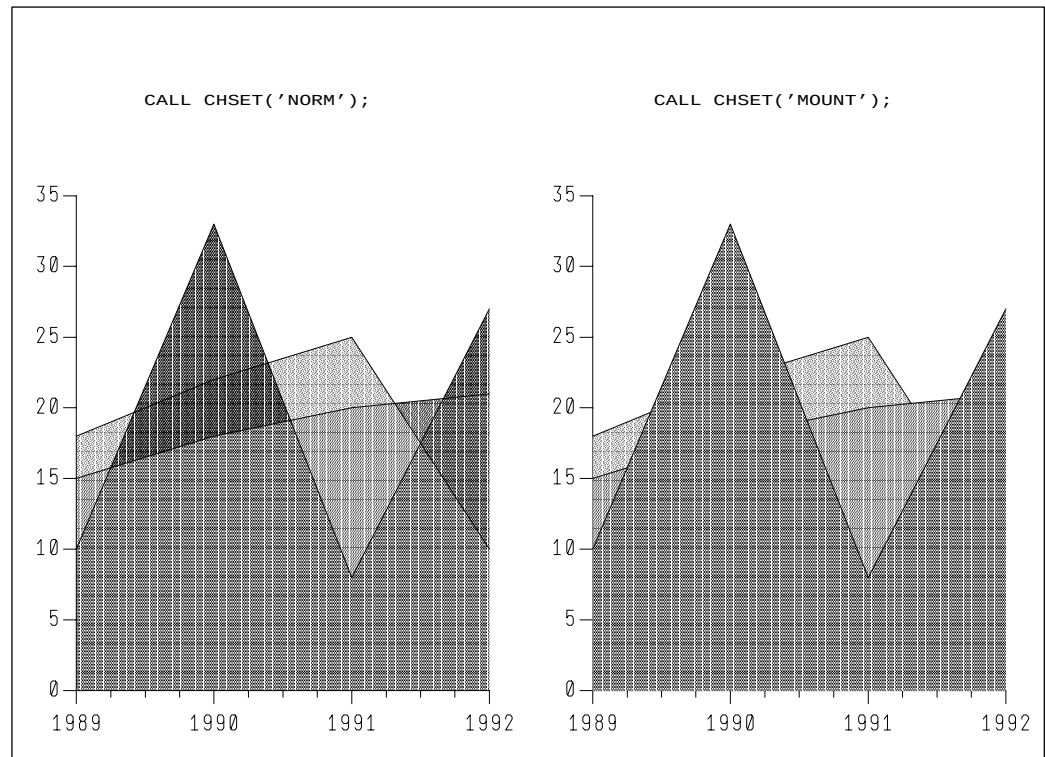


Figure 17. Normal and mountain-range shading

Setting a pattern table, using call CHPAT

In addition to the three attribute tables mentioned in the line graph section (line types, markers, and colors), it is possible to set a pattern table:

```
DCL PATTERNS(4) FIXED BIN(31) INIT(14,8,5,12);
/* Patterns in table */

CALL CHPAT(4,PATTERNS);          /* Establish a pattern table */
```

If system patterns are being used, the pattern numbers must lie in the range 0-16. If a user pattern set has been loaded (using the GSLSS or PSLSS call), then patterns in the range 65-254 can be used also.

Surface chart example

The next example produces the surface chart shown in Figure 18 on page 67:

```
SURFC: PROC OPTIONS(MAIN);
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,193,240); /*Heading attribs*/
DCL N_ATTRS(3) FIXED BIN(31) INIT(7,2,194);    /* Note attribs */
DCL T_ATTRS(4) FIXED BIN(31) INIT(5,3,0,130);  /* Title attribs */
DCL (TYPE,MOD,COUNT) FIXED BIN(31,0);         /* ASREAD parameters */
CALL FSINIT;                                  /* Initialize GDDM */
CALL GSLSS(2,'ADMUVICIP',193);                /* Load Complex Italic */
/* Principal font */
CALL GSLSS(1,'IBM',194);                      /* Load company-logo */
/* Image symbol set */
```

PG routines specific to each chart type

```

/*****
/*      Define chart margins
*****/
CALL CHHMAR(5,7);          /* 5 rows (bottom), 7 rows (top) */
CALL CHVMAR(8,20);         /* 8 cols (left), 20 col (right) */

/*****
/*      Define chart heading
*****/
CALL CHHATT(4,H_ATTRS);    /* Set heading attributes
CALL CHHEAD(30,'OIL MANUFACTURING;LAST 3 YEARS');/*Chart heading*/

/*****
/*      Define chart legend
*****/
CALL CHSET('KBOX');        /* Legend is boxed
CALL CHKEY(6,13,'Africa    Far East    North America'
           ||'USSR        Middle East Europe    ');

/*****
/*      Define axis options
*****/
CALL CHXTIC(1.0,3.0);      /* Major ticks every 1 (year) apart
                           /* 3 minor ticks per interval to
                           /* mark the quarters of the years

CALL CHXLAB(4,4,'****199019911992'); /* x-axis user labels

                           /* (See notes 1 and 2)
CALL CHXSET('LABMIDDLE');    /* Labels between major ticks
CALL CHXSET('GRID');        /* Grid lines on major ticks
CALL CHXTTL(4,'YEAR');      /* x-axis title
CALL CHYTTL(24,'Million Barrels per Year'); /* y-axis title
CALL CHTATT(4,T_ATTRS);     /* Title attributes

/*****
/*      Declare chart data
*****/
DCL X_DATA(7)  FLOAT DEC(6) INIT(
    1989,1989.5,1990,1990.5,1991,1991.5,1992);

DCL Y_DATA(42) FLOAT DEC(6) INIT(
    450, 467, 490, 513, 623, 714, 820,    /* 1st y component */
    200, 200, 189, 178, 298, 280, 312,    /* 2nd y component */
    987,1100, 800,1300,1320,1456,1290,    /* 3rd y component */
    1180, 800, 670, 820, 970, 880,1020,    /* 4th y component */
    1400,1450,1500,1780,2000,1800,1820,    /* 5th y component */
    560, 570, 640, 600, 870, 910,1004); /* 6th y component */
CALL CHSET('RELATIVE');     /* Stacked-chart type, so data is
                           /* plotted relative to the
                           /* previously plotted component
CALL CHXSET('NOFORCEZERO'); /* so x range is 1989-1992
                           /* and not 0-1992

/*****
/*      Plotting call
*****/
CALL CHSURF(6,7,X_DATA,Y_DATA);/* Plot the surface chart, with 6*/
                           /* components each of 7 elements */

```

```

/*****
/*      Add chart notes
*****/
CALL CHNOFF(18.0,0.0);      /* Set note position
CALL CHNOTE('C7',43,'(ALL DATA IN THIS SAMPLE CHART IS SPURIOUS)');
CALL CHNATT(3,N_ATTRS);    /* Set note attributes: white,
/* image symbols, IBM-logo s-set
CALL CHNOFF(65.0,5.0);    /* Set note position
CALL CHNOTE('C7',1,'A');  /* Character code 'A' has the
/* IBM logo (See note 3)

/*****
/*      Send chart to terminal
*****/
CALL CHTERM;               /* Terminate PGF, free storage
CALL ASREAD(TYPE,MOD,COUNT); /* Send chart to terminal
CALL FSTERM;               /* Terminate GDDM, free storage
%INCLUDE(ADMUPINA);        /* Include GDDM and PGF entries
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END SURFC;

```

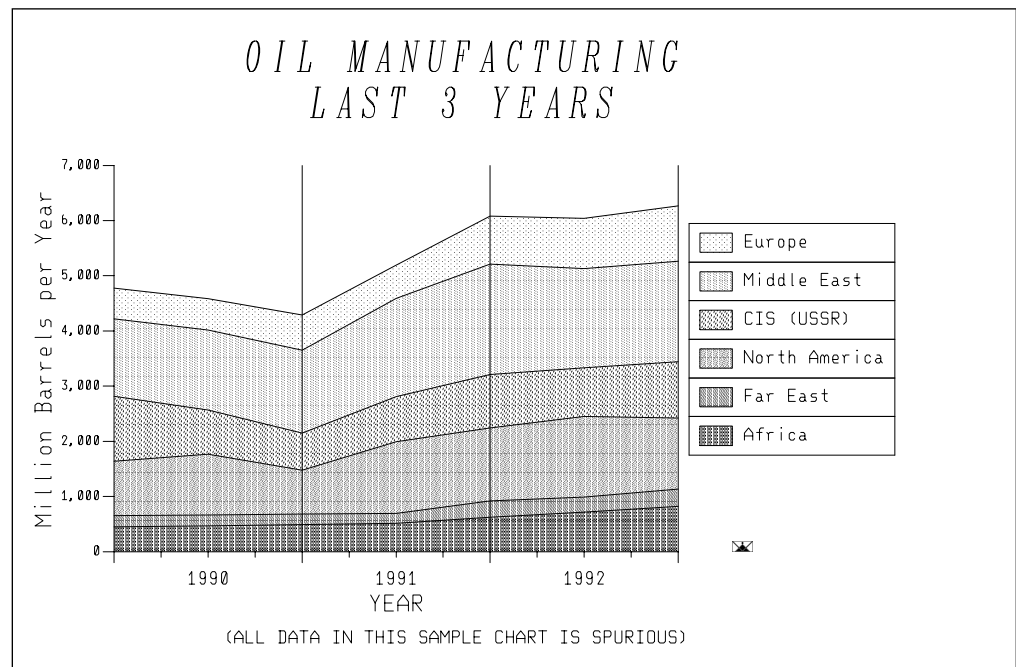


Figure 18. Surface chart

Three points of interest are discussed in the following notes:

Notes:

1. **Commas in numeric labels.** If numeric labels are used, they appear as 1,980, 1,981 and 1,982. In other words, the year-numbers are shown in the same notation as any other four-digit number. To circumvent this problem, alphanumeric "user" labels are employed.

Or, instead, CHSET('PGFS') could be used to suppress the commas (see description in "Axis label and tick-mark calls" on page 45).

2. **User labels with LABMIDDLE.** The LABMIDDLE option shifts all labels half a major-tick position to the **left**. When the labels are numeric, this gives the desired effect, for example, 1 appears between the major ticks for 0 and 1.

When alphanumeric labels are in use, the label for the first major tick mark is lost. So, if three user labels are required (using LABMIDDLE), four labels must be supplied. The first is not used.

If just three labels were supplied (1990, 1991, 1992), they would appear apparently in the wrong order: 1991 1992 1990. This is because the label list is reused in cyclic fashion.

3. **Company logos on charts.** Company logos can be added to charts, using the note facility. In GDDM terminology, this is equivalent to a GSCHAR using a special symbol set containing the company logo. In this example, the mode-2 (image) symbol set called "IBM" has just one symbol in it, at character code 'A'.

Histograms

Histograms feature bars whose width is numerically significant. The bars are sometimes of equal width, but this is not so in the general case (see Figure 19 on page 71).

The calls and options that affect histograms are listed below. The default options are shown first:

CHHIST	Plots the histogram
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'RISERS' 'NORISERS'	Lines between steps of histogram
'FILL' 'INFILL' 'NOFILL'	Shading method
CHPAT	Specify shading-pattern table values
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table
CHCOL	Specify basic color table for shading

Two sets of x data are required, the range starts and the range ends. The range starts correspond to the left side of each bar, the range ends to the right side of each bar.

Here is a typical histogram specification:

```
DCL RANGE_STARTS(5) FLOAT DEC(6) INIT( 0,10,20,30,40);
DCL RANGE_ENDS(5)   FLOAT DEC(6) INIT(10,20,30,40,50);
DCL Y_DATA(10) FLOAT DEC(6) INIT(54,67,79,92,71, /*1st component*/
                                14,23,36,34,42); /*2nd component*/

CALL CHSET('RELATIVE'); /* Stacked-chart type, so relative */
                        /* data is used */

/* Components elements bar-left-sides bar-right-sides 'Y'*/
CALL CHHIST(2,          5,      RANGE_STARTS,      RANGE_ENDS, Y_DATA);
```

Two components are being plotted here. Each component has 5 y-data values, each corresponding to a given range of x values. The y values 54 and 14 correspond to the x range 0 through 10, and so on. The first bar is shaded in one color for the first 54 units and in another for the next 14 units. Its height is therefore 68 units.

In this example the bars were touching – the right-hand side of one bar was the left-hand side of the next. This need not be so.

Suppressing the histogram's risers, using call CHSET

By default, each bar has its side lines drawn (these lines are known as **risers**). The histogram shown in Figure 19 on page 71 has its risers drawn, for example. They can be suppressed or reinstated with these two calls:

```
CALL CHSET('NORISERS'); /* Suppress histogram's risers */

CALL CHSET('RISERS'); /* Draw the risers (default) */
```

Histogram examples

The next program produces the chart shown in Figure 19 on page 71:

```

HISTO: PROC OPTIONS(MAIN);
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,0,190);
DCL N_ATTRS(2) FIXED BIN(31) INIT(3,2);
DCL T_ATTRS(4) FIXED BIN(31) INIT(5,2,0,110);
DCL L_ATTRS(2) FIXED BIN(31) INIT(3,2);
CALL FSINIT;

/*****
/*          Define chart heading          */
*****/
CALL CHHATT(4,H_ATTRS);          /* Set heading attributes */
CALL CHHEAD(53,'Percentage of Population;(Distribution by Age Groups)');
/*****
/*          Define axis options          */
*****/

DCL XVALUES(6) FLOAT DEC(6)      /* Positions of x-axis labels*/
  INIT(5.0,15.0,27.5,42.5,60.0,85.0);
DCL XLABELS(6) CHAR(5)           /* x-axis labels          */
  INIT(' 0-10','11-20','21-35','36-50','50-70',' 71+ ');
CALL CHXTIC(0.5,0.0);           /* Tick interval for x labels*/
CALL CHXSET('PLAIN');           /* Omit tick marks        */
CALL CHXDLB(6,5,XVALUES,XLABELS); /* Add labels            */
CALL CHXTTL(9,'AGE-BANDS');      /* x-axis title           */
CALL CHYTTL(10,'PERCENTAGE');    /* y-axis title           */
CALL CHTATT(4,T_ATTRS);          /* Title attributes      */
CALL CHLATT(2,L_ATTRS);          /* Label attributes      */

/*****
/*          Declare chart data          */
*****/
DCL XLOW(6) FLOAT DEC(6) INIT( 0, 10, 20, 35, 50, 70);
                                           /* Range starts */
DCL XHI(6)  FLOAT DEC(6) INIT(10, 20, 35, 50, 70,100);
                                           /* Range ends  */

DCL Y_DATA(6) FLOAT DEC(6) INIT(12.5, 14, 30, 28, 12.5, 3);

/*****
/*          Plotting call          */
*****/
CALL CHHIST(1,6,XLOW,XHI,Y_DATA); /* Plot the histogram */
/*****
/*          Send chart to terminal          */
*****/

CALL ASREAD(TYPE,MODE,COUNT); /* Send chart to terminal */
CALL FSTERM;                 /* Terminate GDDM        */
%INCLUDE ADMUPINA;            /* Include GDDM and PGF entries */
%INCLUDE ADMUPINC;
%INCLUDE ADMUPINF;
END HISTO;

```

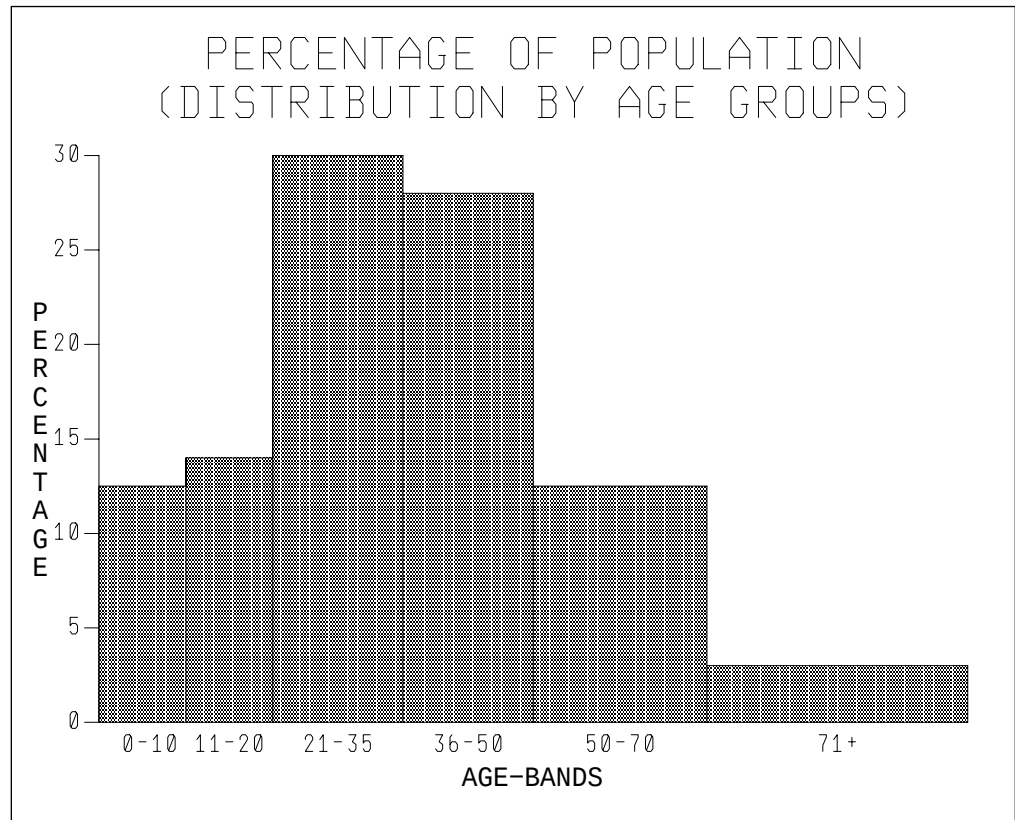


Figure 19. Histogram

The annotation of the various ranges was done with the CHXDLB call. The positions of the labels are specified in this call as the middles of the ranges. The tick-mark interval is defined in the first CHXTIC call to be 0.5, the lowest common denominator of all the label positions. The tick marks themselves are suppressed by the CHXSET call.

Bar charts

In a bar chart, all the bars have equal width. They can be drawn on either a logical x axis or a numeric one.

The calls and options that affect bar charts are listed below. The default options are shown first:

CHBAR or CHBARX	Plots the bar chart
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'CVALUES' 'NOVALUES'	Format/position of values ...
'VSCIENTI' 'VFIXED'	... format (scientific or fixed-point)
'VINSIDE' 'VONTOP'	... position (inside/on top of bars)
'FILL' 'INFILL' 'NOFILL'	Shading method
'MBAR' 'CBAR' 'FBAR'	Multiple, composite(stacked), or floating
'NNOTES' 'INOTES'	Notes to be repositioned or clipped
'YVERTICAL' 'XVERTICAL'	Vertical or horizontal bars
CHPAT	Specify shading-pattern table values
CHCOL	Specify basic color table for shading
CHVCHR	Number of characters in values on bars
CHVATT	Attributes of values on bars
CHVDIG	Decimal digits after decimal point
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table
CHGAP	Distance between bars
CHGGAP	Distance between groups of bars

With a logical x axis, each bar represents a logical x entity, usually indicated by a label. The bars are always spaced out evenly along the x axis. So, the x data of this type of bar chart is not a set of numbers, it is a set of labels. Nevertheless, these labels are specified in the normal way (using CHXLAB, CHXDAY, or CHXMTH).

The logical x axis has a nominal numeric scale: 1 for the first bar, 2 for the second, and so on, and a range of 0.5 to N + 0.5 (where N is the number of bars). Tick marks (and therefore labels) appear under each bar or group of bars; any CHXTIC specification is ignored.

This is a typical specification for a bar chart with a logical x axis:

```
DCL X_LABS CHAR(32) INIT('ITALY FRANCE SPAIN GREECE SWEDEN');
DCL Y_DATA(15) FLOAT DEC(6) INIT(
    27, 32, 36, 47, 52, /* Data for 1st component */
    12, 18, 27, 23, 31, /* Data for 2nd component */
    31, 23, 18, 29, 45); /* Data for 3rd component */

CALL CHXLAB(5,6,X_LABS);          /* Specify x-axis data labels */
CALL CHBAR(3,5,Y_DATA);          /* Plot chart, 3 components of 5 bars */
```

A numeric x axis differs from a logical one in that you supply a set of x values at which the bars are to be drawn. The call that plots this type of chart is exactly similar to the CHPLOT call that plots line graphs:

```

DCL X_DATA(5)  FLOAT DEC(6) INIT
              (8.1, 3.0, 3.9, 4.8, 6.5); /* x values          */
DCL Y_DATA(15) FLOAT DEC(6) INIT
              ( 27, 32, 36, 47, 52, /*Y data for 1st component*/
                12, 18, 27, 23, 31, /*Y data for 2nd component*/
                31, 23, 18, 29, 45); /*Y data for 3rd component*/

CALL CHBARX(3,5,X_DATA,Y_DATA);          /* Plot chart, 3      */
                                         /* Components of 5 bars */

```

Both of these examples specify bar charts with three components, each component having five data values. There are three different ways of displaying multicomponent bar charts:

- **Multiple bar charts** have their components placed side by side in a group. So, in these examples there would be a cluster of three bars at each major tick mark (in the CHBAR case) or each x value (in the CHBARX case).
- **Composite bar charts** have their components stacked one above the other. So, there would be a stack of three bars at each major x-axis tick mark or x value.
- **Floating-bar charts** are identical to composite bar charts except that the first component is not displayed. In this case there would be a stack of two bars floating somewhere above each major x-axis tick mark or x value.

These three types of bar chart are shown on logical x axes in Figure 20 on page 74.

An example of a bar chart on a numeric x axis is shown in Figure 21 on page 75. It was drawn by the program that drew the line graph in Figure 4 on page 26, amended by replacing the CHPLOT call with a CHBARX. Notice that PG routines extend the x axis beyond the highest x value, to accommodate the width of the last pair of bars.

Specifying the type of bar chart, using call CHSET

This is the option that decides the bar-chart type:

```

CALL CHSET('CBAR'); /* Composite bar chart format is used */
CALL CHSET('FBAR'); /* Floating bar chart format is used  */
CALL CHSET('MBAR'); /* Multiple bar chart format is used  */
                    /*          (This is the default form)  */

```

This setting has an effect on the bar chart's appearance only when there is more than one component (except that FBAR is pointless with only one component – no bar is drawn).

Setting the bar-chart gap, using call CHGAP

For a simple bar chart, the gaps between bars are 0.5 times as wide as the bars themselves. If the chart has a numeric x axis and the bars are not evenly spaced, they are drawn so that the smallest gap is 0.5 times the bar width.

The ratio can be adjusted:

```

CALL CHGAP(1.5);          /* Make the bar gaps 1.5      */
                           /* times as wide as the bars */

```

PG routines specific to each chart type

This ratio also applies to composite bar charts or floating bar charts.

Multiple bar charts are more complicated, because there are two different gaps to consider. CHGAP now specifies the ratio of gap to bar **within a group of bars**. Another call determines the gaps between groups of bars:

```
CALL CHGGAP(3.5);      /* Make the gap between groups of bars */  
                        /* 3.5 times as wide as a single bar */
```

The default value for the ratio is 2.0 times the bar width.

If bars or groups of bars drawn with a CHBARX call are not evenly distributed along the x axis, in all cases “gap” means the smallest gap.

You can make bars overlap by specifying a negative gap.

Setting bar chart shading options, using call CHSET

The following shading options can be set:

```
CALL CHSET('NOFILL');  /* No bars are shaded. Just the      */  
                        /* outlines appear                          */  
  
CALL CHSET('FILL');    /* All bars are shaded (default)          */  
  
CALL CHSET('INFILL');  /* (For composite bar charts) The        */  
                        /* first component are not shaded        */
```

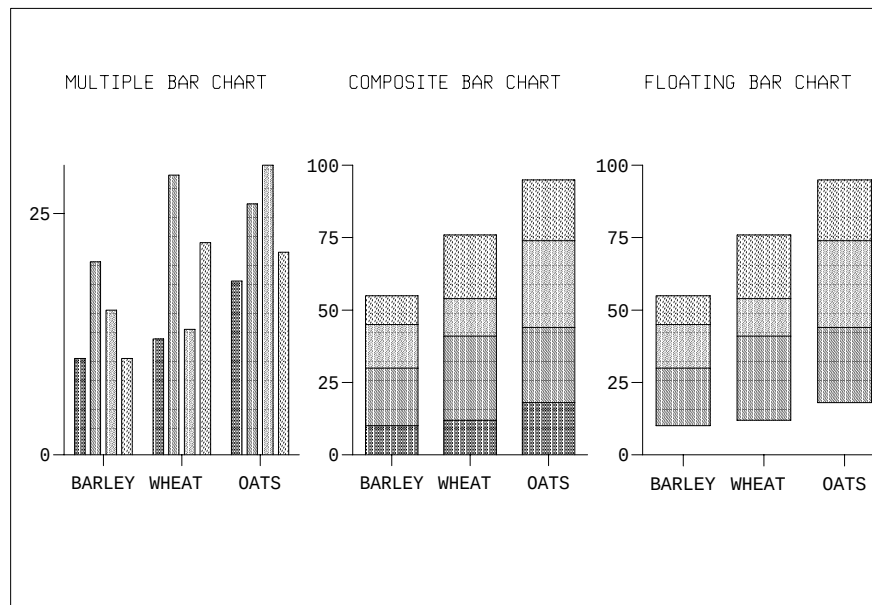


Figure 20. The three types of bar chart

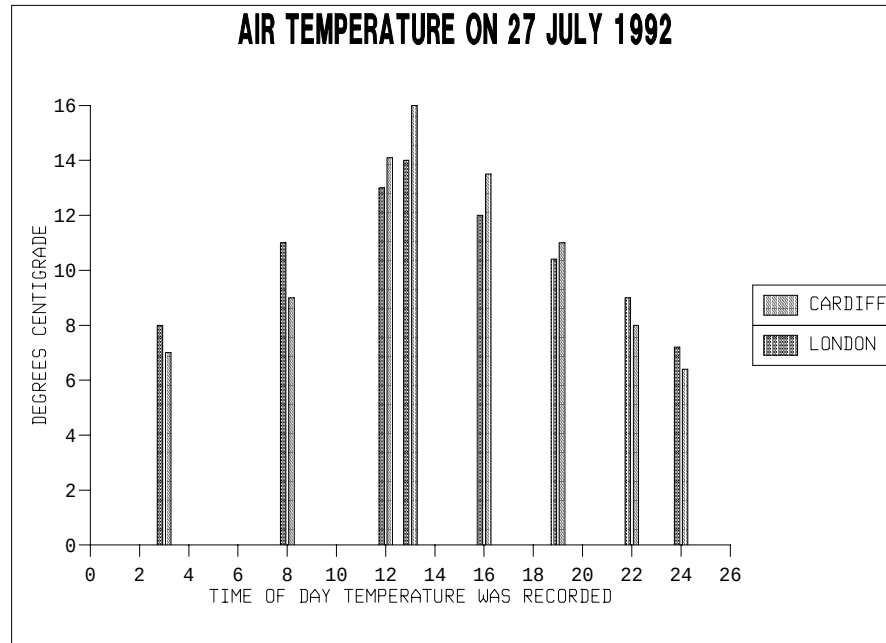


Figure 21. A bar chart on a numeric x axis

Setting bar values, using call CHSET

To request a character representation of the y-data value at the end of each bar, this option is used:

```
CALL CHSET('VALUES');           /* Bar values to appear */
                                /* At the end of each bar */
```

For multicomponent bar charts, there is one bar value for every bar when they are placed side by side (multiple bar chart), but just one value at the end of the stacked bars for the other two types (composite and floating bar charts). The default option can be reinstated by calling CHSET('NOVALUES').

The exact format of the bar values depends on many factors - the size and span of the data, the specified precision of the individual data values and the format of the matching y-axis labels. The same format is used for every bar value in the chart. The user can specify the maximum number of characters to be displayed:

```
CALL CHVCHR(6);                 /* Bar values may not be more */
                                /* than 6 characters long */
```

The default is 9, and the maximum permissible is 15. Note, however, that a bar value with many characters, such as $-1.5462\text{E}+12$, is unlikely to have room on top of a vertical bar (unless vector (mode-3) characters are used, and even then the characters are unreadably small).

Instead of having values at the end of each bar in a GDDM-defined format, you can gain a degree of control over their appearance by first executing this call:

```
CALL CHSET('CVALUES'); /* Controlled-format bar values */
```

Then there are a number of controls you can apply.

- You can choose one of two formats for the bar values:

```
CALL CHSET('VSCIENTI'); /* Set format to scientific (1.3E+01) */
```

```
CALL CHSET('VFIXED'); /* Reset to default of fixed point (13) */
```

and you can specify the number of digits that should be displayed after the decimal point (the default being two):

```
CALL CHVDIG(1); /* One decimal digit only (27.3) */
```

- You can place the bar values either at the end of the bars or inside them:

```
CALL CHSET('VINSIDE'); /* Values inside bars */
```

```
CALL CHSET('VONTOP'); /* Reset to default */  
/* (Values at end of bars) */
```

VONTOP displays the y-data value of the end of each bar, or the end of the stack for composite and floating bar charts. VINSIDE displays the length of each bar in y units, putting a value inside all bars in a stack. So the values displayed by these two options differ if the bars are stacked. They also differ if you draw a datum line above or below the x axis.

If you specify VINSIDE, some bars may be too short to accommodate their values. To avoid this problem, you can specify a threshold below which no bars display their values:

```
DECLARE THOLD(1) INIT(25);  
CALL CHTHRS(1,THOLD); /* Threshold is 25% of y-axis range */
```

To set a threshold, the first parameter must be 1. The second parameter is a one-element array specifying the threshold value. In the example, this is 25, meaning that no values are displayed inside bars shorter than 25% of the y-axis range. So if the y axis extends from 0 to 80, no value of less than 20 is displayed. You can reinstate the default action of not omitting any values by executing a CHTHRS call with the first parameter set to 0.

- You can control the angle of the text using the sixth element of the array parameter of CHVATT.

Text attributes of bar values, using call CHVATT

CHVATT sets the text attributes of bar values:

```
DCL VTEXT_ATTRS(6) FIXED BIN(31)  
/* color mode s-set char-width char-HT angle */  
INIT (2, 2, 196, 90, 110 4500);  
  
CALL CHVATT(6,VTEXT_ATTRS); /* Specify value text attributes */
```

The values in the second parameter of CHVATT are similar to those in the other attribute-setting calls. The character width value is the percentage by which the character grid width is to be multiplied. The height value is the percentage of the width multiplier by which the character grid height is to be multiplied. The character angle is expressed in 1/100ths of a degree. A positive angle means a counterclockwise rotation, and a negative value a clockwise one. The permissible range for the angle is -9000 to +9000.

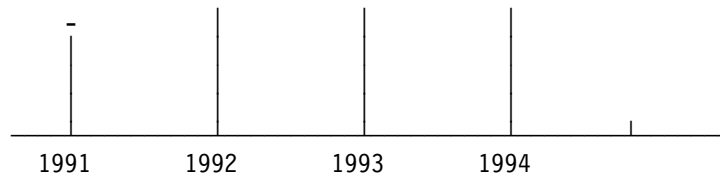
The last element must be omitted unless CHSET('CVALUES') has been specified.

Bar chart x-axis scale marks, using call CHBARX

By using the CHBARX call, you can specify the x-axis values at which the bars are to be placed. The x axis is a numeric axis (rather than a logical one, which is the case with bar charts produced by using the CHBAR call). With CHBARX, each successive group of bars is positioned around user-defined x values, rather than around the arbitrary values 1, 2, 3, (and so on). If autoscaling is in effect, the x axis is extended at each end by 0.5 times the minimum interval between any two successive x values.

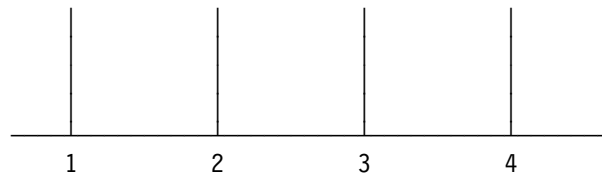
CHBARX calls can specify different x values; if this is the case, the chart layout is changed for each successive call.

Each bar (or bar group) is centered on the specified x values, as shown below.



As a special case of the CHBARX call, the CHBAR call draws major scale marks at integer values (1, 2, 3, and so on), and the bar positioning and label positions are based on these points, as shown in the following examples:

For autoranging, the x axis runs from 0.5 to count+0.5, where count is the number of bars in each component. For count=4, the axis would appear as follows:

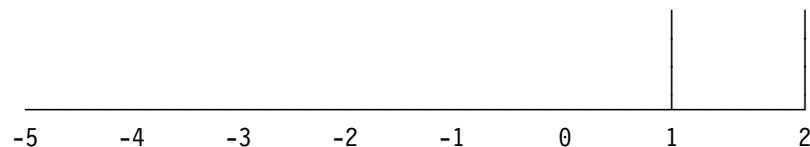


Each bar (or bar group) is centered on the vertical lines shown above.

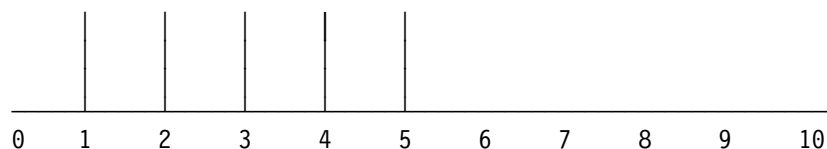
Care must be used when explicitly defining the axis range (by CHXRNG), as shown in the following examples:

1. Range: -5 to 2; count = 4

only the first bar (or bar group) and half of the second appears; the remaining two are lost.



2. Range: 0 to 10; count = 5



Techniques for handling values in bar charts

PG routines provide options that you can specify to improve the appearance of the values in your bar charts.

The important option to specify is CHSET('CVALUES'); without this, your bar charts can only be shown either with or without bar values. By specifying CHSET('CVALUES') you have control over how the bar values are shown. You can reset this option by using CHSET('NOVALUES') if you do not want to have bar values shown with the options described below.

The types of option you can use for your bar chart values fall into six classes:

1. Changing the width and depth of the character box size by specifying a multiplier value, which changes the overall scale of the characters in bar values without altering their aspect ratio. Use the CHVATT call to do this.
2. Rotating the bar values so that they are inclined to one side or the other. A positive value inclines the value to the left (counterclockwise); a negative value inclines the value to the right (clockwise). Again, use CHVATT.
3. Showing the bar values as either scientific values (for example, 1.34E+01) or fixed-point values (for example, 13.4). Use the VSCIENTI and VFIXED options of the CHSET call to do this.
4. Specifying the number of decimal digits that are to be displayed after the decimal point. Use the CHVDIG call to do this.
5. Placing the values either on top of the bar or centered within each bar. Use the VONTOP and VINSIDE options of CHSET to do this.
6. Specifying a threshold limit that controls which bars have values drawn for them. For example, if the y-axis range is 0 through 20 and you specify limit of 50, any bar value that is less than 50% of 20 (that is, 10) is not drawn. Use the CHTHRS call to do this.

You can also use CHSET('BVALUES') to blank out the note areas before the notes are written.

Controlling the appearance of bars in bar charts

PG routines allow you to specify a negative value in the CHGAP call; this gives you a chart that has the second and subsequent elements partially hidden behind the previous one. In other words, the bars in the chart overlap one another.

You can, of course, apply some of the other techniques described above to these overlapping bars; however, you have to experiment to find the combination of techniques that produces the best results for your particular chart.

Bar chart example

This program produces the output shown in Figure 22 on page 81:

```

BARS: PROC OPTIONS(MAIN);
DCL H_ATTRS(4) FIXED BIN(31) INIT(5,3,0,190); /*Heading attribs */
DCL N_ATTRS(4) FIXED BIN(31) INIT(6,3,0,120); /*Note attributes */
DCL T_ATTRS(4) FIXED BIN(31) INIT(5,3,0,170); /*Ax-title attribs*/
DCL L_ATTRS(4) FIXED BIN(31) INIT(4,3,0,80); /*Label attributes*/
DCL COLORS(1) FIXED BIN(31) INIT(7); /*1 color: white */
DCL PATS(2) FIXED BIN(31) INIT(77,117); /* 2 multicolor patterns*/
DCL (TYPE,MOD,COUNT) FIXED BIN(31); /* Parameters for ASREAD*/

CALL FSINIT; /* Initialize GDDM */
CALL GSLSS(3,'ADMCOLSD',0); /* Load GDDM 64-color set */
/*****
/* Define chart heading */
/*****
CALL CHHATT(4,H_ATTRS); /* Set heading attributes */
CALL CHHEAD(32,'OIL PRODUCTION;FROM FIELDS 16-19');/*CHT heading*/
/*****
/* Define axis options */
/*****
CALL CHXMTH(4); /* Month labels on x-axis (starting with April)*/
CALL CHYRNG(0.0,20.0); /* y-axis range */
CALL CHYTIC(5.0,4.0); /* y-axis tick marks */
CALL CHYTTL(15,'Million Barrels'); /* y-axis title */
CALL CHXTTL(26,' 1990 1991'); /* x-axis title */
CALL CHTATT(4,T_ATTRS); /* Title attribs */
CALL CHLATT(4,L_ATTRS); /* Label attribs */
/*****
/* Define color table (using GDDM 64-color set, and color white)*/
/*****
CALL CHCOL(1,COLORS); /* 1-element color table */
CALL CHPAT(2,PATS); /* 2-element pattern table */
/*****
/* Define state-1 y datum line (as starting line for shading) */
/*****
CALL CHYDTM(8.0);

/*****
/* Declare chart data. The 1st component shows the below-target */
/* months (in red) and the 2nd component shows the above-target */
/* months (in green) */
/*****
DCL Y(32) FLOAT DEC(6) INIT(
    4, 6, 5,1E72, 7, 6,
    1E72,1E72,1E72,1E72,1E72,1E72, 7.5, 7,1E72,1E72,
    1E72,1E72,1E72, 9,1E72,1E72,
    10, 13, 15, 18, 11, 13,1E72,1E72, 15, 19);

```

PG routines specific to each chart type

```

/*****
/*      Plotting call
*****/
CALL CHGAP(0.2);          /* Set gap-to-bar ratio
CALL CHSET('CBAR');       /* Composite bar chart
CALL CHBAR(2,16,Y);       /* Plot two-component bar chart

/*****
/* Define state-2 x and y datum lines
*****/
CALL CHXDTM(9.5);
CALL CHYDTM(8.0);

/*****
/*      Add chart notes
*****/
CALL CHSET('NBOX');       /* Notes are boxed
CALL CHNATT(4,N_ATTRS);   /* Note attributes: yellow, mode-2
CALL CHNOFF(25.0,17.0);   /* Set note position
CALL CHNOTE('C7',15,'Field 19;Opened');

CALL CHSET('NONBOX');     /* Notes are not boxed
CALL CHNOFF(29.0,15.8);   /* Set note position
CALL CHNOTE('C7',1,'|');  /* Create arrow, using notes
CALL CHNOFF(29.0,15.0);   /* Set note position
CALL CHNOTE('C7',1,'|');  /*
CALL CHNOFF(29.6,15.0);   /* Set note position
CALL CHNOTE('C7',1,'_');  /*

CALL CHNOFF(30.5,15.0);   /* Set note position
CALL CHNOTE('C7',1,'_');  /*
CALL CHNOFF(31.2,14.45);  /* Set note position
CALL CHNOTE('C7',1,'>');  /* Tip of note arrow

/*****
/*      Send chart to terminal
*****/
CALL CHTERM;              /* Terminate PGF
CALL ASREAD(TYPE,MOD,COUNT); /* Send chart to terminal
CALL FSTERM;              /* Terminate GDDM
%INCLUDE(ADMUPINA);        /* Include GDDM and PGF entries
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END BARS;
```

Some rather advanced techniques were used in this program:

- **Use of state-1 y-datum line.** The aim of the chart was to show below-target months in red and above-target months in green. The bar chart therefore had two components (one shown in red, the other in green), where there was really only one set of data. A state-1 y-datum line (set at y=8) is used as the initial shading line, so that below-target months are shaded downward, above-target months are shaded upward.

The red data is the first component. When the month is above target, the red data value is set to 10^{72} , which indicates, in PG routines, a missing value. When the month is below target, the green data is set to 10^{72} ,

- **Use of 64-color shading patterns.** ADMCOLSD, the GDDM 64-color pattern set was used. As with all multicolor symbols, the base color has to be set to neutral. This would be a typical specification of five colors, two basic and three from the 64-color set:

```
CALL GSLSS(3,'ADMCOLSD',0);      /* Load GDDM 64-color set */

/*                               white red  white  blue  white */
DCL COLORS(5) FIXED BIN(31) INIT(7,  2,  7,  1,  7);

/*                               64-col system 64-col system 64-col*/
DCL PATTS(5) FIXED BIN(31) INIT(74,  8,  102,  0,  116);

CALL CHCOL(5,COLORS);            /* Set 5-element color table */
CALL CHPAT(5,PATTS);            /* Set 5-element pattern table */
```

The first component is shaded using pattern 74 of the GDDM 64-color set. The second component is shaded in red, using system pattern 8.

So, system (monochrome) patterns can be matched with any of the seven basic colors. Multicolor patterns must always be matched with white.

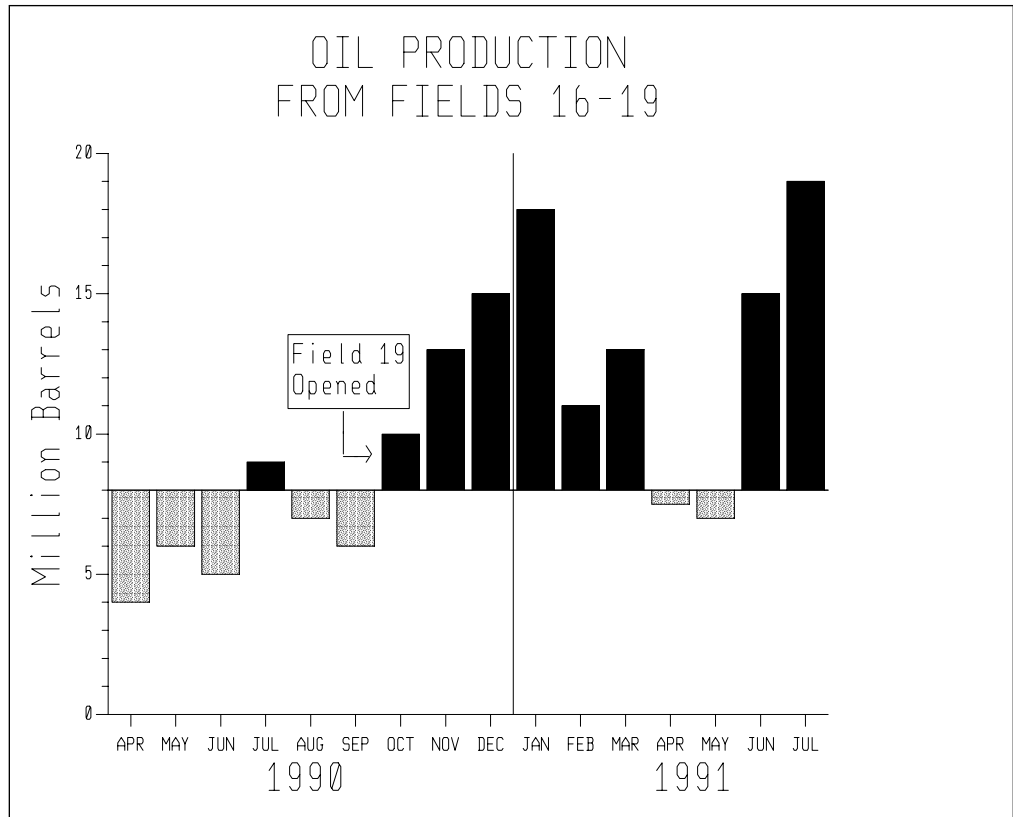


Figure 22. Bar chart

Tower charts

Tower charts are three-dimensional bar charts with numeric x axes. In addition to the x and y axes, they have a z axis. This is drawn as if perpendicular to the x,y plane. Each data value is represented by the height of a three-dimensional tower. Each component is drawn at a specified z value. Tower charts are sometimes called Manhattan or skyscraper charts.

The calls and options that affect tower charts are listed below. The default options are shown first:

CHTOWR	Plots the tower chart
CHSET	
'BACK' 'NOBACK'	Back planes
'BGBASE' 'TCBASE'	Background or tower-chart color
'FILL' 'INFILL' 'NOFILL'	Method of shading
'PGFS' 'NPGFS'	Numeric values punctuated
'SCALETOWER' 'NOSCALETOWER'	Scale tower to be drawn
'SIDE' 'NOSIDE'	Side planes to be drawn
'TOWERTICK' 'NOTOWERTICK'	Major tick marks to be drawn
'ZPICK' 'XPICK'	Pick attributes according to tower position on Z or x axis
CHZSET	
'AXIS' 'NOAXIS'	Axis drawn
'GRID' 'NOGRID'	Grid drawn
'NTICK' 'PTICK' 'XTICK' 'PLAIN'	Tick-mark style
'NUMERIC' 'ALPHANUMERIC'	Type of label
CHPAT	Shading-pattern table values
CHCOL	Basic color table for shading
CHLC	Line-color table
CHLT	Line-type table
CHLW	Line-width table
CHTPRJ	Set projection for next tower chart

Here is a typical specification for a tower chart:

```
DCL X_DATA(5)  FLOAT DEC(6) INIT(1.1, 2.6, 3.3, 4.9, 7.7);
DCL Y_DATA(20) FLOAT DEC(6) INIT( 31, 65, 97, 11, 21,
                                88, 10, 98, 98, 3,
                                43, 40, 67, 23, 54,
                                94, 35, 59, 83, 43);
DCL Z_DATA(4)  FLOAT DEC(6) INIT(1,3,4,6);

CALL CHTOWR(1,4,5,Z_DATA,X_DATA,Y_DATA);
```

The first parameter to the CHTOWR call is not used by PG routines; it must be 0 or 1. The second parameter specifies, in this example, that there are four different z values – four components, in other words. The third parameter states that each component has five data values. The remaining parameters are the z, x, and y values.

Angle and scale of the z axis

By default, the z axis is drawn at 45 degrees to the x axis, and one z unit is the same length as one x unit. You can specify your own values for these projection options by executing a CHTPRJ call:

```
DCL PROJ_OPT(2) FLOAT DEC(6) INIT(60.0,2.0);
```

```
CALL CHTPRJ(1,2,1,PROJ_OPT);    /* Reset projection options    */
```

The first parameter of CHTPRJ must always be 1. The second specifies how many items there are in the list of projection options. The maximum is 2. The third parameter specifies the first option that is to be implemented; any earlier option in the list being ignored. The first option (60.0 in the example) is the angle, in degrees, between the x and z axes (measured counterclockwise). Values in the range 0 to 180 are permitted, but those between about 20 and 70 or 110 and 160 are recommended. The second option in the example, 2.0, specifies that one z unit is to be twice as long as one x unit.

Showing the x and y scales

You can show the y scale by drawing a scale tower beside the chart:

```
CALL CHSET('SCALETOWER');
```

Lines are drawn around the tower at the tick-mark intervals, and marked with y values.

By default, PG routines build a back and a side wall for the chart, on which you can draw x, y, or z grid lines, or any combination of them:

```
CALL CHXSET('GRID');    /* x-grid lines on floor and back wall */
CALL CHYSET('GRID');    /* y-grid lines on back and side walls */
CALL CHZSET('GRID');    /* z-grid lines on floor and side wall */
```

You can draw lines around all the towers at the major y-tick mark values:

```
CALL CHSET('TOWERTICK');
```

These lines are useful even if you have a scale tower or y-grid lines, because the viewer does not always find it easy to project values from the scale tower or grid onto all the towers.

If you do not want either the back or side walls, or no walls, you can suppress them:

```
CALL CHSET('NOBACK');    /* Suppress back wall of tower chart */
CALL CHSET('NOSIDE');    /* Suppress side wall of tower chart */
```

The defaults are BACK and SIDE.

Setting the inter-tower gap, using call CHGAP

For spacing the towers along the x axis, you use the same call as for bar charts:

```
CALL CHGAP(1.0);    /* x-direction gap same as width of towers */
```

A separate call controls the gap in the z direction:

```
CALL CHZGAP(1.5);    /* Z-direction gap 1.5 times depth of towers */
```

The towers cannot overlap. If you specify a negative value for either gap, 0 is used instead.

Axis range, labels, and tick marks

You can let PG routines choose the ranges of the three axes or you can specify them explicitly. CHXRNG and CHYRNG (see “Chart axis calls” on page 37) set the x and y ranges, and CHZRNG, which has an exactly similar format, the z range.

You can use the CHXDLB call (see “Axis label and tick-mark calls” on page 45) to label the x axis, and the similar CHZDLB call to label the z axis.

The default tick-mark scheme for the x and y axes, and the calls (CHXTIC and CHYTIC) that set them, are as described in “Axis label and tick-mark calls.” The default scheme for the z axis, and the explicit call, CHZTIC, are exactly similar.

Other z-axis options

Most of the options described in Chapter 4, “PG routines that apply to most chart types” on page 31 for the CHXSET and CHYSET calls are allowed on the CHZSET call, although a few are invalid. Full details are given in the *GDDM-PGF Programming Reference* book.

Reference and datum planes

The CHXDTM call has no effect on a tower chart. The CHYDTM call builds a reference plane when executed in state-1. It cannot be executed in state-2. (For more information, see “Datum line calls” on page 49).

Selecting color, pattern, and line type

The colors, shading patterns, and types of outline for the towers are selected from the three attribute lists according to their order along either the x axis (following this call):

```
CALL CHSET('XPICK');
```

or along the z axis (following this call):

```
CALL CHSET('ZPICK');
```

The default is ZPICK, which means that all the towers in a data group have the same color, shading pattern, and type of outline.

Shading descending towers

To help identify towers that go downward from the plane of the x and z axes, or from an explicit reference plane, you can specify that their bases should be shaded in the background color:

```
CALL CHSET('BGBASE');
```

Tower chart example

This program created the chart shown in Figure 23 on page 87:

```
TOWERC: PROC OPTIONS(MAIN);  
DCL (TYPE,NUM,COUNT) FIXED BIN(31);  
DCL X_DATA(7) FLOAT DEC(6) INIT( 0, 10, 20, 30, 40, 50, 60);  
DCL Y_DATA(21) FLOAT DEC(6) INIT(0.2, 0.6, 0.9, 0.4, 0.3, 0.2, 0.1,  
                                0.3, 0.8, 1.1, 0.5, 0.4, 0.2, 0.1,  
                                0.5, 1.1, 1.6, 0.7, 0.5, 0.2, 0.1);
```

```

DCL Z_DATA(3) FLOAT DEC(6) INIT(1940,1960,1980);
DCL PROJECTION(2) FLOAT DEC(6) INIT(60.0,0.2); /* z-axis angle */
                                           /* and scale */
DCL H_ATTRS(4) FLOAT DEC(6) INIT(7,3,195,200); /*Heading attribs*/
DCL N_ATTRS(6) FLOAT DEC(6)
      INIT(4,3,0,100,100,3500); /*Note attributes */
DCL PATTERN(1) FIXED BIN(31) INIT(0); /*Shading pattern */
DCL L_ATTRS(1) FLOAT DEC(6) INIT(6); /*General label color */
DCL XL_ATTRS(6) FLOAT DEC(6)
      INIT(2,3,0,100,100,2500); /* incl. color & angle*/
DCL COUT_COL(1) FIXED BIN(31) INIT(7); /*Outline color & ... */
DCL COUT_LW(1) FLOAT DEC(6); /*... (not used) width*/

CALL FSINIT;

/*****
/*      Define chart heading
*****/
CALL CHHATT(4,H_ATTRS); /* Set heading attributes */
CALL GSLSS(2,'ADMUWCRP',195); /* Load symbol set for heading */
CALL CHHEAD(28,'WORLD POPULATION BY LATITUDE');

/*****
/*      Define chart drawing options
*****/

CALL CHTPRJ(1,2,1,PROJECTION);/* Set projection of z axis */
CALL CHSET('NOBACK'); /* Do not draw back or ... */
CALL CHSET('NOSIDE'); /* ... side walls */
CALL CHPAT(1,PATTERN); /* Solid shading for towers */
CALL CHLC(1,COUT_COL); /* Set constant white outline */
CALL CHLATT(1,L_ATTRS); /* Set color for x, y, & z labels */

/*****
/*      Define x-axis options
*****/

CALL CHXLAT(6,XL_ATTRS); /* Set x-label attribs incl. angle*/
CALL CHXRNG(-5.0,65.0); /* Range allows for width of */
                        /* end towers */
CALL CHXLAB(7,8, /* x-axis labels */
' 0 TO 1010 TO 2020 TO 3030 TO 4040 TO 5050 TO 60 OVER 60');

/*****
/*      Define y-axis option
*****/

CALL CHYTTL(26,'POPULATION (1000 MILLIONS)');

/*****
/*      Define z-axis options
*****/

CALL CHZRNG(1935,1985); /* Range allows for depth of */
                        /* end towers */
CALL CHZSET('PGFS'); /* Suppress z axis label function */
CALL CHZTIC(20.0,0.0); /* z tick marks at */
                        /* 20-year intervals */
CALL CHZGAP(2.0); /* Inter-tower gap in */
                  /* the z direction */

```

PG routines specific to each chart type

```

/*****
/*          Draw chart and notes          */
*****/
CALL CHTOWR(1,3,7,Z_DATA,X_DATA,Y_DATA);
CALL CHNOFF(35.0,0.0);      /* Position of note          */
CALL CHNOTE('C7',8,'LATITUDE'); /* Draw x-axis title as a note */
                                /* Now draw note across heading:- */
CALL CHNATT(6,N_ATTRS);      /* Set note attribs, incl. angle */
CALL CHNOFF(31.0,29.0);      /* Position of note          */
CALL CHSET('BNOTE');         /* Blank out space for note      */
CALL CHNOTE('C7',15,'FICTITIOUS DATA'); /* Text of note          */

/*****
/*          Send chart to terminal          */
*****/
CALL ASREAD(TYPE,NUM,COUNT);
CALL FSTERM;
%INCLUDE (ADMUPINA);
%INCLUDE (ADMUPINC);
%INCLUDE (ADMUPINF);
%INCLUDE (ADMUPING);
END TOWERC;
```

Notes:

1. The x- and z-axis ranges extend beyond the ranges of the data. This is to allow for the width and depth of the towers. Otherwise, parts of the towers at the ends of each row and column would be cut off.
2. The CHLATT call defines the same set of attributes for all three axes. However, for the x axis, they are overridden by the CHXLAT call.
3. For a given set of data, several factors affect which towers are hidden. The angle and scale of the z axis, as specified by the CHTPRJ call, are important. So are the inter-tower gap in both directions, and the x- and z-axis ranges. When experimenting to find the best values for these various parameters, you are advised to vary only one at time.
4. The x-axis title is a note created with a CHNOTE call. If the CHXTTL call were used, the title would overlap the labels because they are angled.

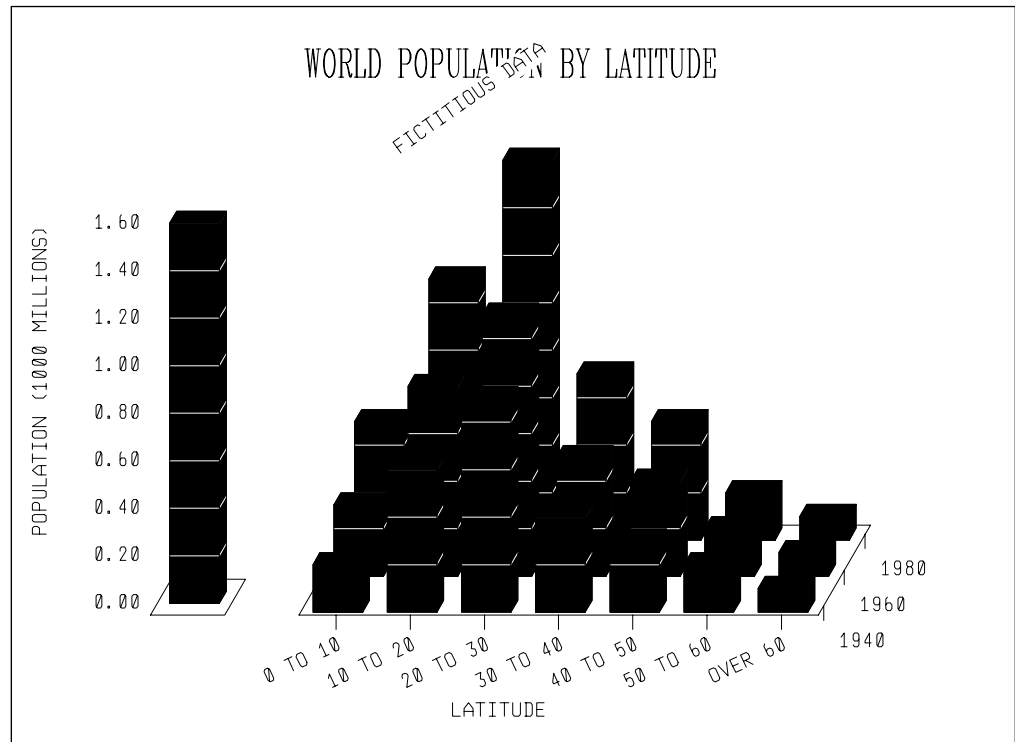


Figure 23. Tower chart

The online version of this book shows Figure 23 in the colors described.

Polar charts

Polar charts are circular line graphs. The x axis is drawn around the circle's perimeter, and y values are represented by the distance from its center.

They have two major uses. One is illustrated in Figure 24. The x values represent a cyclic variable, in this case, time. The other use is illustrated by Figure 25 on page 92. Two or more entities are plotted against several parameters for comparison. In the second type, the x axis is really a logical one, although the application program must supply a set of numeric x values.

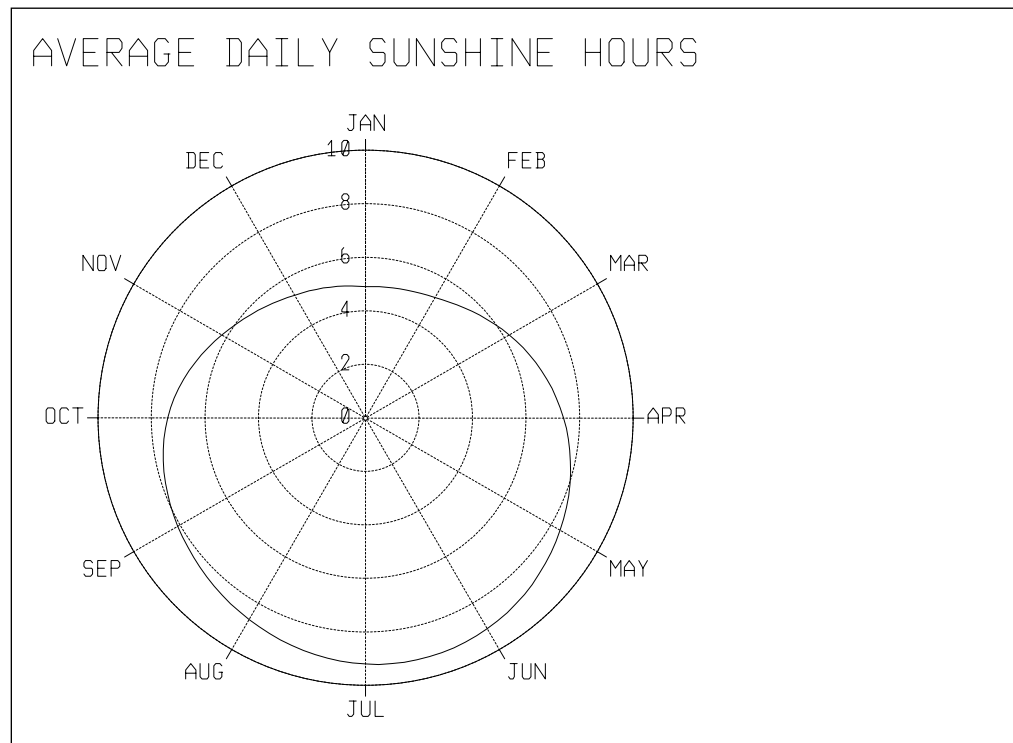


Figure 24. Polar chart with numeric x data

The calls and options that affect polar charts are listed below. The default options are shown first:

CHPOLR	Plots the polar chart
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'MARKERS' 'NOMARKERS'	Markers on plotted points
'MOUNTAIN' 'NOMOUNTAIN'	Mountain-range shading, or ...
'FILL' 'INFILL' 'NOFILL'	... ordinary shading
'MSCALE' 'NOMSCALE'	Specify scaling of markers
'NOCURVE' 'CURVE'	Straight or curved lines
CHFINE	Specify smoothness of curve
CHMARK	Specify marker-table values
CHMKSC	Specify marker scale factors
CHPAT	Specify shading-pattern table values
CHCOL	Specify basic color table for shading patterns and markers
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table

The CHPOLR call for drawing a polar chart is exactly similar to the CHPLOT call for a line graph:

```
DCL X_DATA(6)  FLOAT DEC(6) INIT(  4,  8, 12, 16, 20, 24);

DCL Y_DATA(12) FLOAT DEC(6) INIT(  9, 13, 20, 21, 14, 11,
                                   7, 14, 23, 25, 16,  9);

CALL CHPOLR(2,6,X_DATA,Y_DATA);          /* Plot 2 components, */
                                          /* 6 elements each   */
```

Most of the calls and options that apply to line graphs can be applied to polar charts. The following special points apply to polar charts:

- By default, the x axis is the perimeter of the chart, and the y axis a radial line in the 12 o'clock position. You can make the x axis a circle of a different diameter like this:

```
/*      Make x axis a circle 10 Y units in diameter      */

CALL CHXSET('INTERCEPT'); /* x axis to intercept y axis ... */
CALL CHYINT(10.0);          /* ... at y=10                */
```

You can make the y axis a different radial line like this:

```
/*      Make y axis a radial line at x=12                */

CALL CHYSET('INTERCEPT'); /* y axis to intercept x axis ... */
CALL CHXINT(12.0);          /* ... at x=12                */
```

You cannot change the positions of the axes with the LOWAXIS, MIDDLE, and HIGHAXIS options of the CHXSET and CHYSET calls.

- The chart cannot have secondary axes.
- An x grid, as specified by a CHXSET('GRID') call, is a set of radial lines. A y grid, as specified by a CHYSET('GRID') call, is a set of circles concentric with the x axis.
- You can draw a circular y-datum line with a CHYDTM call and a radial x-datum line with a CHXDTM call.
- By default, the charts are shaded from the center of the circle. You can suppress shading with a CHSET('NOFILL') call. If a y-datum line is drawn in state 1 (that is, if a reference line is drawn), the first component is shaded down or up to it. If you specify the x-axis circle with CHSET('INTERCEPT') and CHYINT calls, it becomes the reference line for shading.

Polar chart example

This program created the chart shown in Figure 25 on page 92:

```
POLARC: PROC OPTIONS(MAIN);

DCL (TYPE,NUM,COUNT) FIXED BIN(31);
DCL H_ATTRS(4) FIXED BIN(31) INIT(3,3,0,175);/*Head text attribs*/
DCL N_ATTRS(4) FIXED BIN(31) INIT(7,3,0,200);/*Note text attribs*/
DCL A_ATTRS(2) FIXED BIN(31) INIT(7,1); /* x-axis color & line */
DCL XL_ATTRS(1) FIXED BIN(31) INIT(5); /* x-label color      */
DCL K_ATTRS(1) FIXED BIN(31) INIT(5); /* Key text color      */

CALL FSINIT;
```

PG routines specific to each chart type

```

/*****
/*          Define heading          */
/*****
CALL CHHATT(4,H_ATTRS);      /* Set heading text attributes */
CALL CHHEAD(40,'TWO CARS COMPARED USING SEVEN PARAMETERS');

/*****
/*          Define x axis (perimeter of chart)          */
/*****

CALL CHAATT(2,A_ATTRS); /* Set attrs of x axis (perimeter line)*/
CALL CHXTIC(1.0,0.0);   /* Set tick interval of 1          */
CALL CHXLAT(1,XL_ATTRS); /* Set x label attributes          */
CALL CHXLAB(7,31,       /* Alphanumeric axis labels          */
'PURCHASE PRICE ;    $15,000    INSURANCE      ;$300/YEAR      ' ||
'$0.25/MILE      ;SERVICING      $0.070/MILE      ;FUEL        ' ||
'    100 BHP/TON; POWER/WT RATIO          6;          SEATS' ||
'  BAGGAGE SPACE;    20 CU FT');

/*****
/*          Define y axis          */
/*****

CALL CHYRNG(0.5,1.0); /* Set Y range to emphasize variation */
CALL CHYSET('NOAXIS'); /* Suppress y axis ...          */
CALL CHYSET('NOLABEL'); /* ... and y labels ...          */
CALL CHYSET('PLAIN'); /* ... and y tick marks          */

/*****
/*          Define legend          */
/*****

CALL CHSET('KBOX'); /* Box around legend          */
CALL CHKATT(1,K_ATTRS); /* Set attributes of key text          */
CALL CHKEY(2,5,'CAR ACAR B'); /* Key labels for legend          */

/*****
/*          Draw the chart          */
/*****

DCL X_DATA(8)
  FLOAT DEC(6)
  INIT( 0.0, 1.0, 2.0, 3.0, 4.0, 5.0, 6.0, 7.0);
DCL Y_DATA(16)
  FLOAT DEC(6)
  INIT(14190.0, 260.0, 0.21, 0.066, 83.3, 6.0, 19.1, 14190.0,
      12986.0, 290.0, 0.23, 0.066, 95.6, 5.0, 16.2, 12986.0);
DCL MAXVALS(16)
  FLOAT DEC(6)
  INIT(15000.0, 300.0, 0.25, 0.070,100.0, 6.0, 20.0, 15000.0,
      15000.0, 300.0, 0.25, 0.070,100.0, 6.0, 20.0, 15000.0);

Y_DATA = Y_DATA/MAXVALS; /* Normalize y data          */
CALL CHPOLR(2,8,X_DATA,Y_DATA);

```

```

/*****
/*          Add notes          */
/*****
CALL CHNATT(4,N_ATTRS);      /* Set color of '+' in note      */
CALL CHNOFF(0.0,0.53);      /* Position '+' at center of chart*/
CALL CHNOTE('Z2',1,'+');    /* Write the '+'              */
CALL CHSET('BNOTE');        /* Blank out note text area    */
N_ATTRS(4) = 75;            /* Redefine size of note text  */
CALL CHNATT(4,N_ATTRS);      /* Reset color and size        */
CALL CHNOFF(0.0,0.58);      /* Position next note text     */
CALL CHNOTE('Z2',12,'CENTER VALUE');
CALL CHNOFF(0.0,0.55);      /* Position next note text     */
CALL CHNOTE('Z2',23,'= 1/2 X PERIMETER VALUE');

/*****
/*          Send chart to terminal      */
/*****
CALL ASREAD(TYPE,NUM,COUNT);

CALL FSTERM;
%INCLUDE(ADMUPINA);
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
END POLARC;

```

Notes:

1. If you intend to draw a line all the way round a polar chart, you need to do two things. Omitting either of them results in a missing segment.
 - Ensure that the range of x values supplied by your program is not less than the x-axis range on the chart. By default, the axis range starts at zero. In the example, the first x value was chosen to be 0 rather than 1. An alternative to starting at x=0 is to specify the axis range explicitly in a CHXRNG call.
 - Plot the last y value in the data group on the same x value as the first. In the example, the polygon is closed by repeating the first y value at the end of each data group. In other types of polar chart, repetition of y values may not be required. In a 24-hour temperature plot, for instance, the last reading, taken at 2400 hours, can be plotted at the same x value (probably x=0) as the first reading, taken at 0000 hours.
2. In a polar chart of this type, the various parameters are likely to be measured by widely differing ranges of numbers. For instance, the highest number that could represent the purchase price is 15,000, whereas the number representing the fuel cost cannot exceed 0.070. There is only one y scale, so the numbers must be adjusted. In the example, each y value is divided by the maximum value of the parameter it represents, so that the y values passed to the CHPOLR call are all in the range 0 to 1. The y scale is then meaningless, so the y axis and its tick marks and labels are suppressed.
3. To emphasize the differences between the two data groups, the y range is set to 0.5 to 1. This means that, instead of representing zero for each parameter, the center of the chart represents half the perimeter value, namely \$7,500 for the purchase price, \$150 for the insurance cost, and so on.

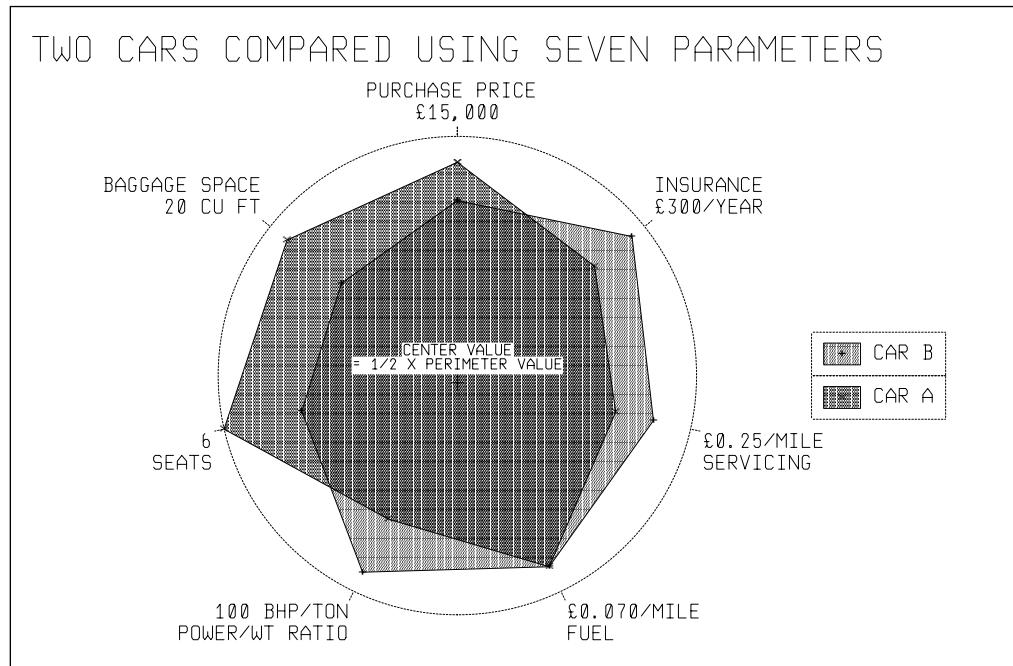


Figure 25. Polar chart with logical x data

Pie charts

Pie charts are noticeably different from the previous chart types. They do not have any axes, and the data for a multicomponent pie chart is displayed differently from that of other multicomponent charts.

This difference is illustrated in Figure 26, which shows two charts representing the same data, a multiple pie chart and a composite bar chart.

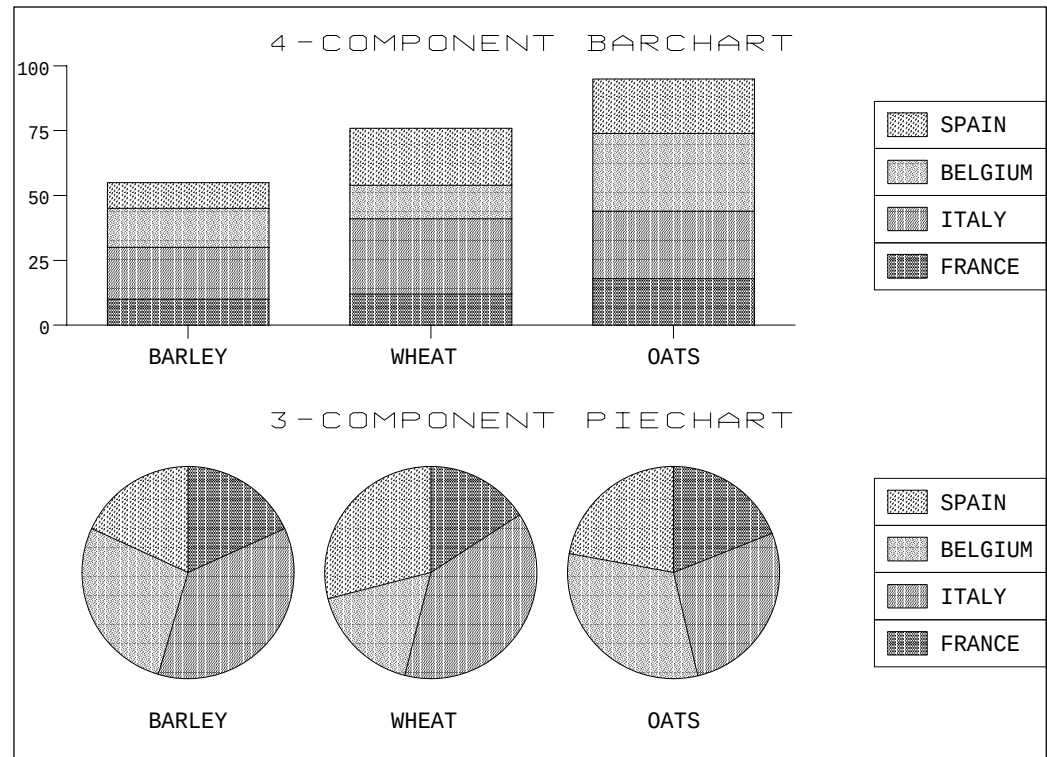


Figure 26. Comparison of composite bar chart with multiple pie chart

The calls and options that affect pie charts are listed below. The default options are shown first.

PG routines specific to each chart type

CHPIE	Plots the pie chart
CHSET	
'PERPIE' 'ABPIE'	Percentage or absolute data
'NOVALUES' 'VALUES'	Values on pie slices
'PIEKEY' 'SPIDER'	Spider labels or legend
'SPISLICE' 'SPILABEL'	Type of spider label if used
'NOPROPIE' 'PROPIE'	Pie size proportional to total values
'FILL' 'INFILL' 'NOFILL'	Shading method
'YVERTICAL' 'XVERTICAL'	Orientation of multiple pies
CHPAT	Specify shading-pattern table values
CHCOL	Specify basic color table for shading
CHLC	Specify line-color table
CHLT	Specify line-type table
CHLW	Specify line-width table
CHPCTL	Specify explosion factor, tilt, and thickness
CHPEXP	Slices of the pie to be moved out (exploded)
CHPIER	Size reduction to fit multiple pies
CHNUM	Number of pies to be plotted if multiple calls are used
CHVATT	Attributes of values on pies

These are the main calls that produced the two charts:

```
DCL Y_DATA(12) FLOAT DEC(6) INIT(10,12,18, /* 1st component */
                                20,29,26, /* 2nd component */
                                15,13,30, /* 3rd component */
                                10,22,21); /* 4th component */
CALL CHXLAB(3,6,'BARLEYWHEAT OATS ');
CALL CHKEY(4,7,'FRANCE ITALY BELGIUM SPAIN ');
CALL CHBAR(4,3,Y_DATA); /* Plot 4-component bar chart */
.....
.....
DCL Y_DAT2(12) FLOAT DEC(6) INIT(10,20,15,10, /* 1st component */
                                12,29,13,22, /* 2nd component */
                                18,26,30,21); /* 3rd component */
CALL CHXLAB(3,6,'BARLEYWHEAT OATS ');
CALL CHKEY(4,7,'FRANCE ITALY BELGIUM SPAIN ');
CALL CHPIE(3,4,Y_DAT2); /* Plot 3-component pie chart */
```

So, a component (or data group) is a different concept for a pie chart. Up to now a component has meant “all the blue bits” (blue markers joined with a blue line on a line graph, a blue layer on a surface chart, sets of blue bars on a histogram or bar chart). Now a component means “one pie” instead.

CHKEY still provides the text that explains the different colors and patterns used in the chart. CHXLAB provides the pie titles.

Exploding slices

You can highlight one or more slices of a pie chart by exploding them from the pie. You specify which slices by creating an explosion table:

```
DCL SECTOR_NUMS(10) FIXED BIN(31) INIT(0, 0, 0, 0, 1, 0, 0, 0, 0, 1);

CALL CHPEXP(10,SECTOR_NUMS); /*Create 10-element explosion table*/
```

The first parameter of CHPEXP specifies how many elements the table is to contain. The second is an array containing a 1 in each element corresponding to a slice that is to be exploded, and a 0 in every other element. The example specifies that the fifth and tenth slices are to be exploded. If there are more elements in the table than slices in the pie, the excess elements are ignored. If there are more slices than elements, the table is reused. A value of 0 in the first parameter reinstates the default action of not exploding any slices.

3-dimensional pies

You can make your pies more graphically effective, especially when they contain exploded slices, by drawing them with three dimensions. They then have a thickness, and are displayed as if tilted or viewed from an angle. At the same time, you can specify the amount by which exploded slices are to be moved out from the pie:

```
/* Explosion-factor tilt thickness */
DCL PIECTL(3) FLOAT DEC(6) INIT( 0.3,      0.7,      0.5);

CALL CHPCTL(3,PIECTL);
```

The first parameter to the CHPCTL call specifies that the second parameter has three elements in it, which is the maximum allowed. These elements specify the following information:

- The first specifies that all the exploded slices are to be moved out by 0.3 times the horizontal radius of the pie. Any value from 0 through 100 is allowed. Values greater than about 1 are not recommended, because PG routines may then have to make the size of the pie excessively small to keep the exploded slices within the chart area. The default is 0.2.
- The second element specifies a tilt factor of 0.7. The permitted range is 0 (which makes the pie into a simple circle) through 0.99 (which reduces the top surface to a line). Values greater than about 0.8 are not recommended. The default is 0.
- The last element specifies that the thickness of the pie is to be 0.5 times the horizontal radius. The permitted range is 0 through 100. Values greater than about 10 are not recommended because PG routines may then have to make the radius of the pie excessively small. The default is 0.

A CHPCTL call with the first parameter set to 0 restores all the default values.

Labeling pie slices

Pie charts can have **spider tags** attached to each of their slices (see Figure 27 on page 97, for example). The key text can be attached to these spider tags instead of appearing in a legend. These are the two options involved:

PG routines specific to each chart type

```
CALL CHSET('SPIDER');      /* Key text attached to spider tags */  
CALL CHSET('PIEKEY');      /* Key text in legend (the default) */
```

The numerical values of each slice can also be attached to the spider tags. These values are always expressed as integral percentages.

```
CALL CHSET('VALUES');      /* Attach percentage values to tags */  
CALL CHSET('NOVALUES');    /* No values attached to tags (default) */
```

If neither SPIDER nor VALUES is specified, the spider tags do not appear. The color of the tags and their associated text is, by default, that of the slice itself (see Figure 27 on page 97 again). If you want to control the color of the labels yourself, you must request the SPILABEL option:

```
CALL CHSET('SPILABEL'); /* The spider tags appear in the      */  
                        /* default color. Text attributes    */  
                        /* (including color) are determined by */  
                        /* CHKATT if spider is requested,      */  
                        /* otherwise by CHVATT.                */  
CALL CHSET('SPISECTOR'); /* The spider tags and text appear in */  
                        /* the sector color. The other three  */  
                        /* text attributes are set by CHKATT or */  
                        /* CHVATT as just described.          */
```

Absolute or percentage data?

The pie slice data can be provided already in percentage form (the default) or in absolute form, using any units. This option determines the data type:

```
CALL CHSET('ABPIE');       /* Data provided in absolute form */  
CALL CHSET('PERPIE');      /* Data already in percentages  */
```

When the data is in absolute form, every pie is a whole pie. The slice sizes indicate the proportion of each slice's data relative to the whole.

When the data is in percentage form, the pies are incomplete unless the percentages add up to exactly 100. If the total for a pie exceeds 100, the pie is not drawn and an error message is issued.

When a multiple pie chart is based on absolute data, you can request that the total size of each pie is in proportion to the total data it represents.

This is the option required:

```
CALL CHSET('PROPIE');      /* Proportionally sized pies */  
CALL CHSET('NOPROPIE');    /* Pies equal in size (default) */
```

An example of proportional pies is shown in Figure 27 on page 97.

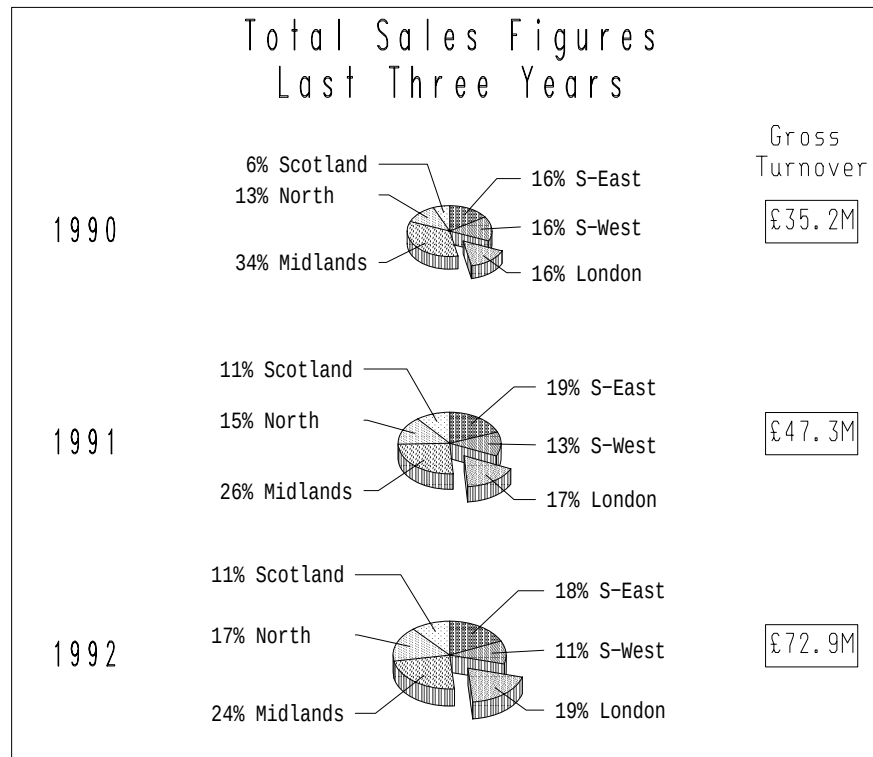


Figure 27. Pie chart where the total size of each pie is proportionate to that year's sales figures.

Insufficient room for pies?

When you try to display multiple pie charts, it sometimes happens that there is not enough room in the chart area. When this is the case, the error message

ADM0540 E PIE CHART CANNOT BE DRAWN IN THE AVAILABLE CHART AREA

is issued. There are several ways to correct this condition:

- You can set the vertical margins of the chart area to zero. Pies and spider labels are then built in the full plotting area.
- If the pies are side by side (the default), the spider tags are competing for the same space. It may be better to draw the pies one above the next (as in Figure 27). This is the call required:

```
CALL CHSET('XVERTICAL');          /* Draw multiple pies on */
                                   /* top of one another */
```

A pie chart does not have any axes, of course. This option merely determines the relative positioning of multiple pie charts.

- Another way to prevent the spider labels competing for space is to move the labels into a shared legend (with the PIEKEY option, described above).
- If your chart has a heading, the margins are automatically increased to make room for it. You can omit the CHHEAD call and write the heading with a CHNOTE call, which does not affect the margins.
- As a last resort, you can shrink all the pies. This is the call:

```
CALL CHPIER(20); /* Reduce the size of the pies by 20 percent */
```

You can also use this option if full-sized pies make the chart appear too congested.

Pie chart example

The following example program draws the pie charts shown in Figure 27 on page 97

```
PIE: PROC OPTIONS(MAIN);
DCL H_ATTRS(4) FIXED BIN(31) INIT(5,3,193,200);
DCL L_ATTRS(4) FIXED BIN(31) INIT(5,3,193,160);
DCL K_ATTRS(2) FIXED BIN(31) INIT(0,2);
DCL N_ATTRS(4) FIXED BIN(31) INIT(6,3,0,130);
DCL B_ATTRS(1) FIXED BIN(31) INIT(4);
DCL COLORS(6) FIXED BIN(31) INIT(6,2,3,4,5,1);
DCL EXPLIST(6) FIXED BIN(31) INIT(0,0,1,0,0,0);
DCL THREE_D_LIST(3) FLOAT DEC(6) INIT(0.5,0.4,0.3);
CALL FSINIT;
CALL GSFLD(1,1,32,79);
CALL GSLSS(2,'ADMUVDRP',193);/*Load duplex Roman Principal s-set*/

/*****
/*          Define chart heading                               */
*****/
CALL CHHATT(4,H_ATTRS);          /* Set heading attributes */
CALL CHHEAD(36,'Total Sales Figures;Last Three Years');

/*****
/*  Define chart margins, framing box and text attributes    */
*****/
CALL CHHMAR(0,5);
CALL CHCOL(6,COLORS);
CALL CHSET('CBOX');
CALL CHBATT(1,B_ATTRS);
CALL CHLATT(4,L_ATTRS);          /* Attributes for pie title text */
CALL CHKATT(2,K_ATTRS);          /* Attributes for pie label text */
```

```

/*****
/*      Define pie chart options      */
/*****
CALL CHSET('ABPIE');      /* Data is absolute, not percentage*/
CALL CHSET('PROPIE');      /* Total size of each pie varies */
CALL CHSET('XVERTICAL');    /* Pies on top of one another */
CALL CHPEXP(6,EXPLIST);    /* Exploded sector(s) */
CALL CHPCTL(3,THREE_D_LIST); /* 3-d control parameters */
CALL CHSET('VALUES');      /* Values on spider tags */
CALL CHSET('SPIDER');      /* Labels on spider tags */

CALL CHXLAB(3,4,'199019911992'); /* Pie chart titles */
CALL CHKEY(6,8,'S-East S-West London MidlandsNorth Scotland');

/*****
/*      Declare chart data      */
/*****

DCL Y_DATA(18) FLOAT DEC(6) INIT(5, 5, 5, 11, 4, 2, /*1st pie*/
                                9, 6, 8, 12, 7, 5, /*2nd pie*/
                                13, 8, 14, 17, 12, 8); /*3rd pie*/

/*****
/*      Plotting call      */
/*****
CALL CHPIE(3,6,Y_DATA);      /* Plot the pie chart */

/*****
/*      Add chart notes      */
/*****
CALL CHNATT(4,N_ATTRS);      /* Note attributes: yellow,mode-3*/
CALL CHNOFF(67,24.5);      /* Set note position */
CALL CHNOTE('C7',15,' Gross;Turnover');
CALL CHSET('NBOX');      /* Subsequent notes to be boxed */
CALL CHNOFF(68,22);      /* Set note position */
CALL CHNOTE('C7',6,'$35.2M');
CALL CHNOFF(68,13);      /* Set note position */
CALL CHNOTE('C7',6,'$47.3M');
CALL CHNOFF(68,4);      /* Set note position */
CALL CHNOTE('C7',6,'$72.9M');

/*****
/*      Send chart to terminal      */
/*****
CALL CHTERM;      /* Terminate PGF */
CALL ASREAD(TYPE,MODE,COUNT); /* Send chart to terminal */

CALL FSTERM;      /* Terminate GDDM */
%INCLUDE(ADMUPINA); /* Include GDDM and PGF entries */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END PIE;

```

Note: The program is intended to execute on a 3279 display unit, of which the bottom right-hand cell is used as an attribute byte, and is therefore not available for

PG routines specific to each chart type

graphics. If the graphics field were allowed to default to the complete screen, the bottom right-hand corner of the chart framing box would be missing. To prevent this, the graphics field has been set to omit the right-hand column of the screen.

Venn diagrams

This rare form of chart is illustrated in Figure 28 on page 103. The data consists of three values: two populations, and an overlap. This is the plotting call:

```
/*      1st population      2nd population      overlap      */
CALL CHVENN(240,              520,              119);
```

It is an error if the overlap exceeds either of the populations. The populations and the overlap can be annotated by using the CHKEY call. The three keys are attached to the first population, the second population, and the overlap, in that order (see the example Venn diagram mentioned above). As with pie charts, the XVERTICAL option can be used to place the circles one above the other, rather than side by side.

The calls and options that affect Venn diagrams are listed below. The default options are shown first:

CHVENN	Plots the Venn diagram
CHSET	
'FILL' 'INFILL' 'NOFILL'	Method of shading
'YVERTICAL' 'XVERTICAL'	Orientation of circles
CHCOL	Basic color table for shading
CHLC	Line-color table
CHLT	Line-type table
CHLW	Line-width table
CHPAT	Shading-pattern table values

Venn diagram examples

This example program produces the output shown in Figure 28 on page 103:

```
VENN: PROC OPTIONS(MAIN);
DCL K_ATTRS(2) FIXED BIN(31) INIT(5,2);      /*Key text attribs*/
DCL N_ATTRS(4) FIXED BIN(31) INIT(6,3,0,160); /*Note attribs (1)*/
DCL N_ATTRS2(4) FIXED BIN(31) INIT(5,3,0,140); /*Note attribs (2)*/
DCL (TYPE,MOD,COUNT) FIXED BIN(31);          /* Parameters for ASREAD */
CALL FSINIT;                                  /*Initialize GDDM */

/*****
/*   Define Venn labels and their attributes
*****/
CALL CHKATT(2,K_ATTRS);                      /* Attributes for Venn labels */
CALL CHKEY(3,21,'  Female Students  '||'  Maths Students  '
           ||'Female Maths Students'); /* Specify 3 Venn labels*/
                                           /* (See note 1) */

/*****
/*   Plotting call
*****/
CALL CHVENN(113,86,37);                      /* Plot the Venn diagram */
```

PG routines specific to each chart type

```

/*****
/*      Add chart notes
*****/
/*****
CALL CHNATT(4,N_ATTRS);      /* Note attributes: yellow,mode-3 */
CALL CHNOFF(0.0,32.0);      /* Set note position
CALL CHNOTE('C1',30,'Student Intake;Statistics 1992'); /* Write
/* note
*/

CALL CHNOFF(60.0,32.0);      /* Set note position
CALL CHNOTE('C7',23,'St. Basil's; College'); /*(See note 2)*/
CALL CHNOFF(0.0,0.0);      /* Set note position
CALL CHNOTE('C7',14,'CHART;NUMBER 6');      /* Write note
CALL CHNOFF(80.0,0.0);      /* Set note position
CALL CHNOTE('C9',25,'Compiled by ; C.Alison '); /* Write note
*/

CALL CHNATT(4,N_ATTRS2);      /* Note attributes: turquoise*/
CALL CHSET('BNOTE');      /* Blank before subsequent
/* notes (See note 3)
*/

CALL CHNOFF(28.0,16.0);      /* Set note position
CALL CHNOTE('C7',3,'113');      /* Write note
*/

CALL CHNOFF(40.7,16.0);      /* Set note position
CALL CHNOTE('C7',2,'37');      /* Write note
CALL CHNOFF(54.0,16.0);      /* Set note position
CALL CHNOTE('C7',2,'86');      /* Write note
*/

/*****
/*      Send chart to terminal
*****/
/*****
CALL ASREAD(TYPE,MOD,COUNT);      /* Send chart to terminal
CALL FSTERM;      /* Terminate GDDM
%INCLUDE(ADMUPINA);      /* Include GDDM and
%INCLUDE(ADMUPINC);      /* PGF entries
%INCLUDE(ADMUPINF);
END VENN;

```

Notes:

1. Blanks were used to center the two shortest labels specified with CHKEY.
2. Similarly, leading blanks were used to right-align the word "College" in one of the chart notes.
3. The BNOTE option blanked out some space in the middle of the two data circles, to make room for the notes giving the data values.

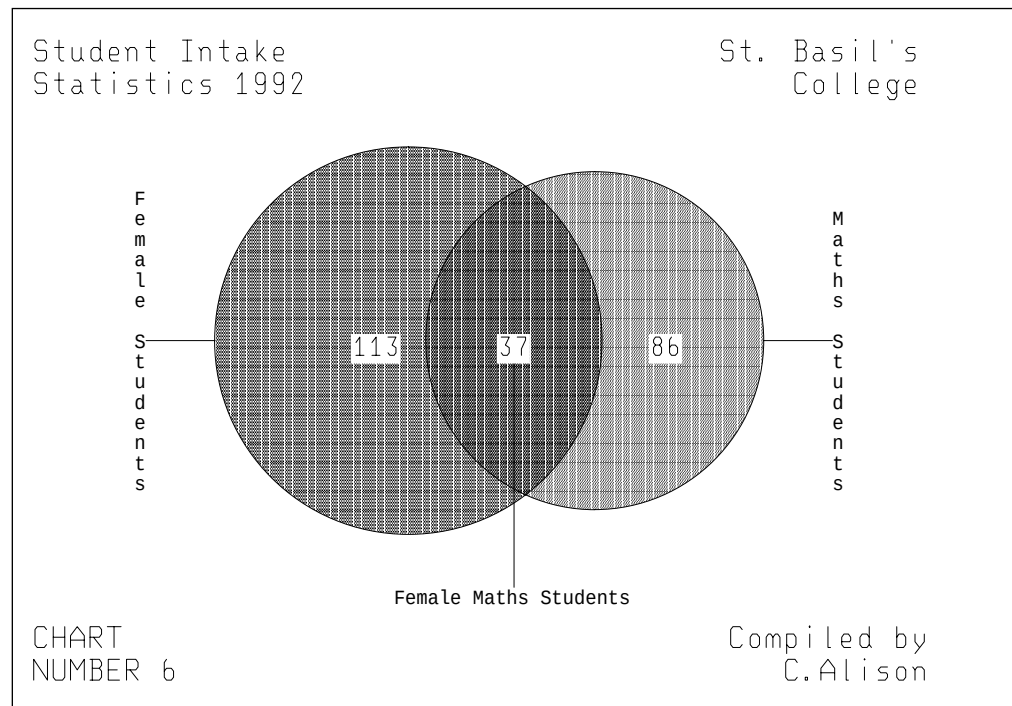


Figure 28. Venn diagram

Table charts

A table chart is simply data displayed in a tabular form. There is an illustration of a table chart at Figure 29. The horizontal dimension of a standard table chart is the z axis. The vertical dimension is the x axis. The data shown in the cells of the table is the y data.

The x and z axes of a table chart are treated similarly. You can put a set of labels around the edge of the table using calls CHXSET and CHZSET. The x labels go down the left edge of the table, and the z labels go along the top of the table. The labels can be:

Numeric values supplied on the CHDTAB call

Data labels supplied by the CHXDLB and CHZDLB calls

For the x axis only, they may also be:

Labels supplied by the CHXLAB call

Month or day labels as specified by the CHXDAY or CHXMTH calls.

The calls and options that affect table charts are shown below. Using these items as reference, decide on what you need to change to produce your own chart, and on the calls or options to do it. The default options are shown first:

CHDCTL	Layout of the table chart
CHDTAB	Plots the table chart
CHMISS	Set character string to replace missing values
CHSET	
'ABSOLUTE' 'RELATIVE'	Absolute or relative data
'CVALUES' 'NOVALUES'	Format and position of values
'PGFS' 'NPGFS'	Numeric values punctuated
'VFIXED' 'VSCIENTIFIC'	y values in scientific notation or not
'YVERTICAL' 'XVERTICAL' 'ZVERTICAL'	Vertical or rotated table
'ZPICK' 'XPICK'	Assign attributes in either direction
CHCOL	Basic color table for shading
CHLC	Line-color table
CHVATT	Attributes of values in table
CHVCHR	Number of digits in values
CHVDIG	Decimal digits after decimal point

PRODUCT SALES OVER FOUR YEARS				
	1989	1990	1991	1992
NUT	40.00	49.00	51.00	52.00
BOLT	45.00	53.00	57.00	59.00
PIN	62.00	60.00	61.00	69.00
NAIL	53.00	56.00	58.00	56.00
TACK	58.00	70.00	72.00	81.00
SCREW	46.00	47.00	47.00	49.00

Figure 29. Table chart

Table chart example: The following program draws the table chart shown in Figure 29.

```

TABLE: PROC OPTIONS(MAIN);
DCL TAB_LIST(5) FIXED BIN(31) INIT(0,1,100,100,160);
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,0,200);
DCL XLIST (4) FLOAT DECIMAL INIT(0,0,0,0);
DCL ZLIST (4) FLOAT DECIMAL INIT(0,0,0,0);
DCL X_ARRAY(24) FLOAT DECIMAL INIT(0,0,0,0,0,0,0,0,0,0,0,0,
                                     0,0,0,0,0,0,0,0,0,0,0,0);
DCL Y_ARRAY(24) FLOAT DECIMAL;
DCL Z_ARRAY(24) FLOAT DECIMAL INIT(0,0,0,0,0,0,0,0,0,0,0,0,
                                     0,0,0,0,0,0,0,0,0,0,0,0);
DCL DATA_ARRAY(24) FLOAT DECIMAL /* Data to be plotted. */
      INIT (40,45,62,53,58,46, /* First group */
            49,53,60,56,70,47, /* Second group */
            51,57,61,58,72,47, /* Third group */
            52,59,69,56,81,49); /* Fourth group */
DCL ATTYPE FIXED BINARY (31) INIT (0);
DCL ATTVAL FIXED BINARY (31) INIT (0);
DCL COUNT FIXED BINARY (31) INIT (0);

CALL FSINIT;

CALL CHDCTL(5,TAB_LIST); /* Enlarge table chart */

CALL CHZDLB(4,4,ZLIST,'1989199019911992');

```

PG routines specific to each chart type

```
CALL CHXSET('ALPHA');          /* Labels to data labels */
CALL CHXDLB(6,5,XLIST,'NUT  BOLT PIN  NAIL TACK SCREW');

/*****
/*          Define chart heading          */
*****/

CALL CHHATT(4,H_ATTRS);
CALL CHHEAD(29,'PRODUCT SALES OVER FOUR YEARS'); /* Heading */

/*****
/*          Plotting call          */
*****/
CALL CHDTAB(4,6,Z_ARRAY,X_ARRAY,DATA_ARRAY);
/* Plot table chart */
CALL CHTERM;          /* Terminate PGF to free storage */
/*****
/*          Send chart to the terminal          */
*****/
CALL ASREAD(ATTTYPE,ATTVAL,COUNT); /* Transmit to the terminal*/
CALL FSTERM;          /* Terminate GDDM and PGF */
%INCLUDE ADMUPINA;    /* Include library of standard */
%INCLUDE ADMUPINC;    /* GDDM BASE and PGF */
%INCLUDE ADMUPINF;    /* declarations */
END;
```

Chapter 6. Using PG routines to create complex charts

This chapter deals with some complex chart construction techniques:

- More than one chart on a page
- Several plotting calls for one chart
- Mixing chart types
- Secondary axes.

More than one chart on a page

The next example program uses two chart areas. It produces the output shown in Figure 26 on page 93. Several points that arise are discussed after the program.

```

BARPIE: PROC OPTIONS(MAIN);      /* Program to create two charts */
DCL (WIDTH,DEPTH) FLOAT DEC(6);  /* Temporary variables */
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,0,200); /* Heading attribs*/
DCL L_ATTRS(2) FIXED BIN(31) INIT(6,2); /* Label attributes */
DCL K_ATTRS(2) FIXED BIN(31) INIT(4,2); /* Key text attributes*/
DCL COLORS(4) FIXED BIN(31) INIT(1,2,5,4); /* Color table */
DCL (TYPE,MOD,COUNT) FIXED BIN(31); /* Parameters for ASREAD */

CALL FSINIT;                      /* Initialize GDDM */
CALL GSFLD(1,2,32,78);           /* Define graphics field */
/* (See note 1) */
CALL GSSEG(0);                   /* Create graphics segment */
/* (See note 2) */
CALL GSCOL(5);                   /* Set color to turquoise */
CALL GSLW(2);                    /* Set line width to thick */
CALL GSLINE(100.0,0.0);          /* Draw turquoise frame */
CALL GSLINE(100.0,100.0);        /* " */
CALL GSLINE(0.0,100.0);          /* " */
CALL GSLINE(0.0,0.0);            /* " */
CALL GSSCLS;                     /* Close segment before PGF*/

CALL GSQPS(WIDTH,DEPTH);         /* Query picture space */
CALL CHAREA(0.0,WIDTH,0.5*DEPTH,DEPTH); /* Chart area top half */
CALL CHVMAR(4,19);              /* Set vertical margins */
/* (See note 3) */
CALL CHCOL(4,COLORS);            /* Set chart colors */

/*****
/* Define chart heading */
/*****/
CALL CHHATT(4,H_ATTRS);          /* Set heading attributes */
CALL CHHEAD(21,';4-COMPONENT BARCHART'); /*Define chart heading */

/*****
/* Define chart labels, keys, and data */
/*****/
CALL CHXLAB(3,6,'BARLEYWHEAT OATS '); /* Set X-data labels*/
CALL CHKEY(4,7,'FRANCE ITALY BELGIUMSPAIN '); /* Set key text */
CALL CHSET('KBOX');              /* Boxed legend */
CALL CHYTIC(25.0,0.0);           /* y-axis tick marks*/

```

```

CALL CHKATT(2,K_ATTRS);          /* Set key text attributes */
/*                               (See note 4) */
CALL CHLATT(2,L_ATTRS);          /* Set label text attributes*/

DCL Y_DATA(12) FLOAT DEC(6) INIT(10,12,18, /* 1st component */
                                20,29,26, /* 2nd component */
                                15,13,30, /* 3rd component */
                                10,22,21); /* 4th component */

/*****
/*      Plotting call
*****/
CALL CHSET('CBAR');              /* Composite bar chart */
CALL CHSET('RELATIVE');          /* Stack the components */
CALL CHBAR(4,3,Y_DATA);          /* Plot the bar chart */

/*****
/*      Return to state-1 and define second chart area
*****/
CALL CHSTRT;                     /* Return to state-1, but do not
/*                               reset any parameters or options*/
/*                               (See note 5) */
CALL CHAREA(0.0,WIDTH,0,0.5*DEPTH); /* Chart area is bottom half*/

/*****
/*      Define chart heading
*****/
CALL CHHEAD(22,';3-COMPONENT PIE CHART'); /* Pie heading */
/*                                         (See note 6)*/

/*****
/*      Re-define chart data (the labels and keys remain)
*****/
DCL Y_DAT2(12) FLOAT DEC(6) INIT(10,20,15,10, /* 1st component */
                                12,29,13,22, /* 2nd component */
                                18,26,30,21); /* 3rd component */

/*****
/*      Plotting call
*****/
CALL CHSET('ABPIE');             /* Absolute data, not percentage*/
CALL CHPIE(3,4,Y_DAT2);          /* Plot the pie chart */

/*****
/*      Send chart to terminal
*****/
CALL ASREAD(TYPE,MOD,COUNT);      /* Send chart to terminal */
CALL FSTERM;                      /* Terminate GDDM */
%INCLUDE(ADMUPINA);               /* Include GDDM and PGF entries */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END BARPIE;

```

Notes to consider when drawing more than one chart on a page:

1. **Mixing PG routines and general graphics.** The graphics field definition applies equally to the PG routine plotting calls and any general graphics that may be specified.
2. **Segments and PG routines.** In this program, some general graphics are written to the page before plotting any charts. These general graphics must belong to an explicitly specified segment. *The current segment must always be closed before making a PG routine plotting call.* This is because the PG routines use segments for their own graphics. GDDM would not allow the PG routines to open a new segment if the program had left one open.
3. **Character grid size.** Dimensions such as margins and heading text sizes are often specified in terms of the character grid. In the absence of a chart area, the character grid unit size defaults to the hardware cell size on devices that use hardware cells to display graphics.

When there is a chart area, the default character grid size is reduced in proportion to the ratio of the chart area to the picture space (horizontally and vertically).

Assuming that the picture space is defaulted to the complete screen, and that mode-3 (vector) text is used, this has the desired effect on the complete chart. If, for example, the chart area is halved in width, the text used on the chart is also halved in width, as are the chart margins. The exact proportions are maintained.

4. **CHAREA stretches mode-3 text.** This program divides the screen area into two horizontal slices. As just described, the character grid size is elongated in the horizontal direction. Vector characters (such as those used in the chart heading) is therefore stretched horizontally.

For that reason, mode-2 (image) characters have been selected for the labels and key text. They are of fixed size and aspect ratio, and are therefore more readable under this chart area. As an alternative, the aspect ratio of the mode-3 text could be corrected using the CHKATT and CHLATT calls.

If you want to set an explicit character grid size, you must use CHCGRD, as described below.

5. **Returning to state-1.** A simple chart passes through three stages:
 - State-1, where most of the options and attributes are set.
 - The plotting call itself, which causes the axes to be constructed.
 - State-2, where chart notes, further plotting calls, and a limited subset of options can be set.

Many of the options and attributes affect the plotting call and can therefore be set **only when the program is in state-1**. GDDM therefore provides two calls that return the program to state-1, thereby enabling the program to set (or alter) state-1 options before a second plot. These are the two calls:

```
CALL CHSTR;      /* Return to state-1, but leave all state-1 */
                  /* options and attributes intact          */

CALL CHRNI;      /* Return to state-1, and reset all options */
                  /* and attributes to their default values */
```

In the present program, CHSTRT is used, because the program intends to use the same CHKEY, CHXLAB, and CHHATT values as were used previously.

Note that the setting of PG routine options is independent of GDDM concepts such as device and page. The options are global and apply to any device or page that is made current. In particular, if you are in state-2 (because you have completed a plot), you are still in state-2 when you move to a new page or a new device. To return to state-1, you must issue CHSTRT or CHRNIT.

6. **New-line character in heading.** The first character in the heading text is a ; (semicolon) new-line character. This creates a blank line at the top of the heading, thereby lowering the actual heading to the required position.

Setting the character grid size

The character grid size, mentioned above, can be set explicitly by the program. This setting affects all text on the chart. It also affects the margin sizes (which are specified or defaulted in terms of character grid widths and heights), and therefore the size of the plotting area. This is the call required:

```
CALL CHCGRD(80,43); /* Set character grid so that there are 80  */
                    /* grid-columns across the chart area and  */
                    /* 43 grid-rows down the chart area        */
```

CHCGRD must be used with caution as it has widespread effects.

Several plotting calls for one chart

Occasionally it is convenient to create a multicomponent chart in stages. This is particularly useful when the x data is different from one component to another. Such a chart is then said to use **paired data** – each component has its own x data in addition to its own y data. Paired data is sometimes called **free data**, to contrast it with **tied data**, in which there is one set of x data for all components.

The following example program plots two components of a line graph and sends the chart to the terminal. It then adds three further components (in two stages) and displays the full chart on the terminal.

The final output is shown in Figure 30 on page 112, and various points of interest are discussed after the program.

```
MPLT: PROC OPTIONS(MAIN);
DCL (TYPE,MOD,COUNT) FIXED BIN(31);          /* Params for ASREAD */
DCL HATTS(4) FIXED BIN(31) INIT(2,3,0,210); /* Heading attribs  */
DCL LATTS(2) FIXED BIN(31) INIT(5,2);        /* Label attributes  */
DCL COLORS(5) FIXED BIN(31) INIT(2,7,5,4,6); /*Color attrib table*/
CALL FSINIT;                                  /* Initialize GDDM   */

CALL CHVMAR(7,14);
CALL CHHEAD(30,'MULTIPLE PLOT;OF 5 COMPONENTS'); /*Chart heading*/
CALL CHHATT(4,HATTS);                          /* Set heading attributes*/
CALL CHLATT(2,LATTS);                          /* Set label attributes  */
CALL CHCOL(5,COLORS);                         /* Set data colors       */
CALL CHSET('KBOX');                          /* Legend is boxed       */
CALL CHKEY(5,5,'LINE1LINE2LINE3LINE4LINE5'); /* Key text              */
                                           /*      (See note 1)      */
```

```

CALL CHXRNG(0.0,20.0);          /* Define x-axis range */
CALL CHYRNG(0.0,30.0);          /* Define y-axis range */
                                /*      (See note 2)      */

DCL X_DATA(7)  FLOAT DEC(6) INIT(0,2,4,6,8,10,12); /* x data */
DCL Y_DATA(14) FLOAT DEC(6) INIT(
    5, 9, 13, 16, 19, 21, 14,      /* 1st y component */
    7, 11, 16, 21, 20, 15, 9);    /* 2nd y component */
/*****/
/* 1st plot */
/*****/
CALL CHPLOT(2,7,X_DATA,Y_DATA);    /* Plot 1st 2 components */
                                /*      (See note 3)      */

/*****/
/*      Send chart to terminal      */
/*****/
CALL ASREAD(TYPE,MOD,COUNT);        /* Send chart to terminal*/

DCL X_DAT2(5)  FLOAT DEC(6) INIT(11,13,14,17,20); /* Different */
                                                /* x data */
DCL Y_DAT2(5)  FLOAT DEC(6) INIT(
    3, 10, 17, 26, 29);            /* 3rd y component */
/*****/
/* 2nd plot */
/*****/
CALL CHPLOT(1,5,X_DAT2,Y_DAT2);    /* Plot 3rd component */
                                /*      (See note 4)      */

DCL Y_DAT3(10) FLOAT DEC(6) INIT(
    13, 15, 17, 18, 11,          /* 4th y component */
    3, 5, 8, 5, 1);            /* 5th y component */
/*****/
/* 3rd plot */
/*****/
CALL CHPLOT(2,5,X_DATA,Y_DAT3);    /* Plot last 2 components */
                                /* using original x data */
                                /*      (See note 5)      */

/*****/
/*      Send chart to terminal      */
/*****/
CALL CHTERM;                      /* Terminate PG routines */
CALL ASREAD(TYPE,MOD,COUNT);      /* Send chart to terminal */
CALL FSTERM;                      /* Terminate GDDM */
%INCLUDE(ADMUPINA);               /* Declarations of */
%INCLUDE(ADMUPINC);               /* entry points */
%INCLUDE(ADMUPINF);
END MPLOT;

```

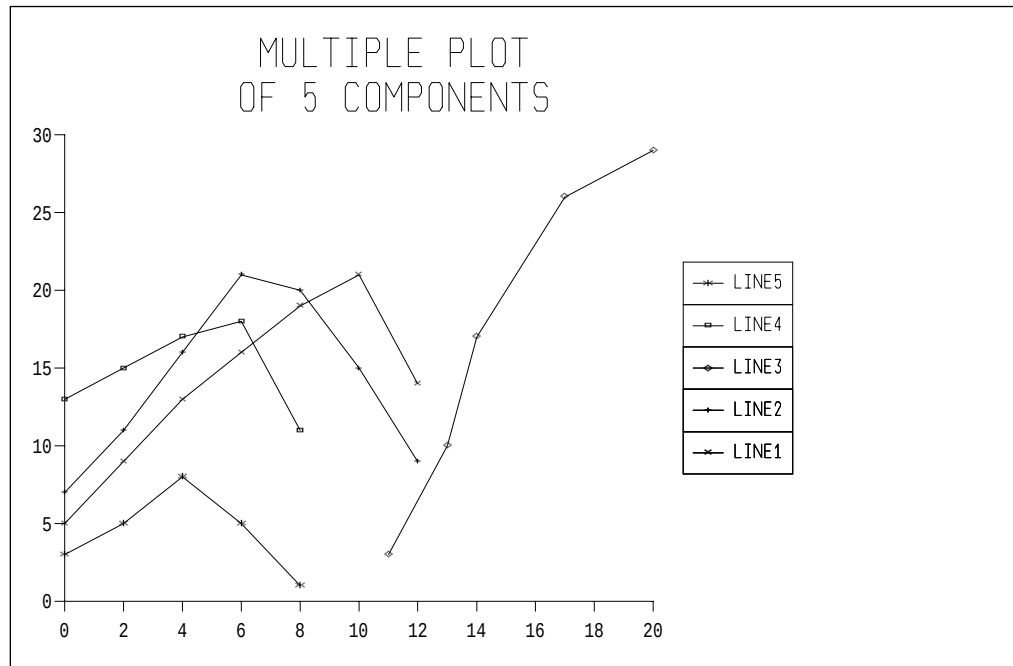


Figure 30. Several plots on one chart

Notes:

1. **Complete CHKEY before the first plot.** This CHKEY call is for the *final chart*. Space for four key entries are allocated, even though the chart has only two components, and therefore only two keys, when it is first plotted and displayed.

The eventual (four-entry) legend is centered in the right-hand margin. The partial (two-entry) legend displayed on the first chart is slightly below center. The remaining two entries are **added** when the later plot occurs.

2. **Auto-ranging of multiple plot axes.** The axes are drawn, once and for all, when the first plot occurs (that is, when you move into state-2). If the axes are allowed to autorange, they do so according to the data on the first plotting call. In the present example, this would give a y range of 0 to 21, which would prove inadequate for the later plotting calls. An explicit (and larger) y range is therefore set before the first plot. CHYRNG is a good example of an option that can be set only in state-1, because it affects the first chart plot.

Another advanced technique using CHSET('RANGE') calls is described in the *GDDM-PGF Programming Reference* book.

3. **Effect of partial plot.** This first plot draws the axes (with labels, tick marks, and titles, if any). In this case, it plots the first two components, one red line and one white line. The chart heading is drawn, and also the chart legend. At this stage, the legend contains two entries, one for the red line and one for the white.

4. **Adding further plots.** The second plot adds the third component to the chart. This component uses the next entry in the attribute tables (the specified color table and the default line-type table). The third line is solid and turquoise (just as it would be if the first three components were plotted together). A third entry is added to the legend.

The new data line, of course, overwrites any chart notes that were written at the time of the first plot.

The range of the second set of x data lies within the autorange on the basis of the first set of x data. If this were not so, an explicit CHXRNG would be required before the first plot.

5. **Too few CHKEY entries.** Two more data lines are added to the chart, using the next two colors in the color table – green and yellow. Only *one* further entry is added to the legend, because the original CHKEY call specified space (and text) for only four keys.

Multiplotting for other chart types

Multiplotting is not, in general, possible for any of the stacked chart types. The shading would start at the reference line (the x-axis or y-datum line) for *each* plot, because one plot would have no memory of the previous plots.

Multiplotting is possible for line graphs, as you have seen, and also for pie charts and multiple bar charts (as opposed to composite bar charts).

In the last two cases, PGF needs to know in advance how many pie charts (or sets of bars) are eventually plotted. For example, if three pie charts are to be plotted with three separate calls, PGF displays the first one in just one-third of the available space. This is the call that informs PGF of the planned number of components:

```
CALL CHNUM(5); /* The pie chart (or multiple bar chart) will */
               /* eventually have 5 components. Plot the */
               /* partial charts accordingly */
```

The next example program uses CHNUM to prespecify a multiple pie chart:

```
PIE3: PROC OPTIONS(MAIN);
DCL H_ATT(4) FIXED BIN(31) INIT(2,3,196,210); /* Heading size */
DCL L_ATT(4) FIXED BIN(31) INIT(4,3,196,160); /* Title size */
DCL COLORS(6) FIXED BIN(31) INIT(1,2,5,4,5,2); /* Color table */
DCL PATTS(6) FIXED BIN(31) INIT(0,0,4,0,0,5); /* Pattern table */
DCL (TYPE,MOD,COUNT) FIXED BIN(31); /* Parameters for ASREAD */

CALL FSINIT; /* Initialize GDDM */
CALL GSFLD(1,1,32,79); /* Using chart background, & */
                       /* running on IBM 3279 screen, */
                       /* so exclude bottom right cell */
CALL GSLSS(2,'ADMUVDRP',196); /* Load roman symbol set */
CALL CHSET('CBACK'); /* Request chart background */

CALL CHHMR(0,2); /* No bottom horizontal margin */
CALL CHHEAD(30,'3 SEPARATELY PLOTTED PIECHARTS'); /* Chart heading */
CALL CHHATT(4,H_ATT); /* Set heading attributes */
CALL CHSET('XVERT'); /* Pies arranged vertically */
CALL CHXLAB(3,4,'199019911992'); /* Pie chart titles */
CALL CHLATT(4,L_ATT); /* Set pie title attributes */

CALL CHCOL(6,COLORS); /* Set colors of pie slices */
CALL CHPAT(6,PATTS); /* Set shading patterns */

CALL CHKEY(6,8,'S-East S-West London MidlandsNorth Scotland');
CALL CHSET('KBOX'); /* Boxed legend */

/*****
/* Prespecify 3 components */
*****/
CALL CHNUM(3); /* Leave room for 3 pie charts */

CALL CHSET('ABPIE'); /* Absolute data, not percentage */
CALL CHSET('VALUES'); /* Values on spider tags */

DCL Y_DAT1(5) FLOAT DEC(6) INIT(5, 5, 5, 11, 4); /* 1st pie */
CALL CHPIE(1,5,Y_DAT1); /* Plot 1st pie chart */

DCL Y_DAT2(4) FLOAT DEC(6) INIT(9, 6, 8, 12); /* 2nd pie */
CALL CHPIE(1,4,Y_DAT2); /* Plot 2nd pie chart */

DCL Y_DAT3(6) FLOAT DEC(6) INIT(13, 8, 14, 17, 4, 13); /* 3rd pie */
CALL CHPIE(1,6,Y_DAT3); /* Plot 3rd pie chart */

CALL CHTERM; /* Terminate PGF */
CALL ASREAD(TYPE,MOD,COUNT); /* Send chart to terminal */
CALL FSTERM; /* Terminate GDDM */
%INCLUDE(ADMUPINA); /* Include GDDM and PGF entries */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
%INCLUDE(ADMUPING);
END PIE3;
```

So, the three pie charts may have different numbers of sectors. Once again it was necessary to issue the full CHKEY information before the first plot, so that PGF could allocate enough space for the full legend.

Mixing chart types

Once you have moved to state-2 (by making the first plot), you can add further plots of a different chart type on to the established axes. The most common example is to add one or more lines of a line graph on top of a bar chart or histogram. The line might represent the sales forecast where the bars showed the actual sales.

Even though the plots are of different chart type, the entries in the attribute tables are still used in succession, as are the key texts specified in CHKEY. So, a legend might have four entries, three for a multiple bar chart and one for an overlaid line graph.

Using secondary axes

Sometimes, when combining business charts on one display, it is convenient to have a second y axis (or, more rarely, a second x axis).

For example, you may need to build a chart that shows both the sales figures for the last six years and the number of sales people employed. The bar chart giving the sales figures would have a y axis scaled in thousands of dollars. A second y axis would be used for the overlaid line graph showing the corresponding numbers of sales people.

These calls are used for selecting a particular axis:

```
CALL CHXSEL(1);           /* Select the primary x axis */
CALL CHYSEL(2);           /* Select the secondary y axis */
```

All calls that refer to an axis apply to the *currently selected axis*. So, this set of calls is needed to set axis titles for both y axes:

```
CALL CHYSEL(1);           /* Make primary y axis current */
CALL CHYTTL(11,'TOTAL SALES'); /* Set primary y-axis title */

CALL CHYSEL(2);           /* Make secondary y axis current */
CALL CHYTTL(15,'TOTAL PERSONNEL'); /* Set secondary y-axis title*/
```

At the start of a program, both primary axes are automatically selected.

Example program using secondary axes

The next example illustrates the use of secondary axes. The output of the program is shown in Figure 31 on page 116.

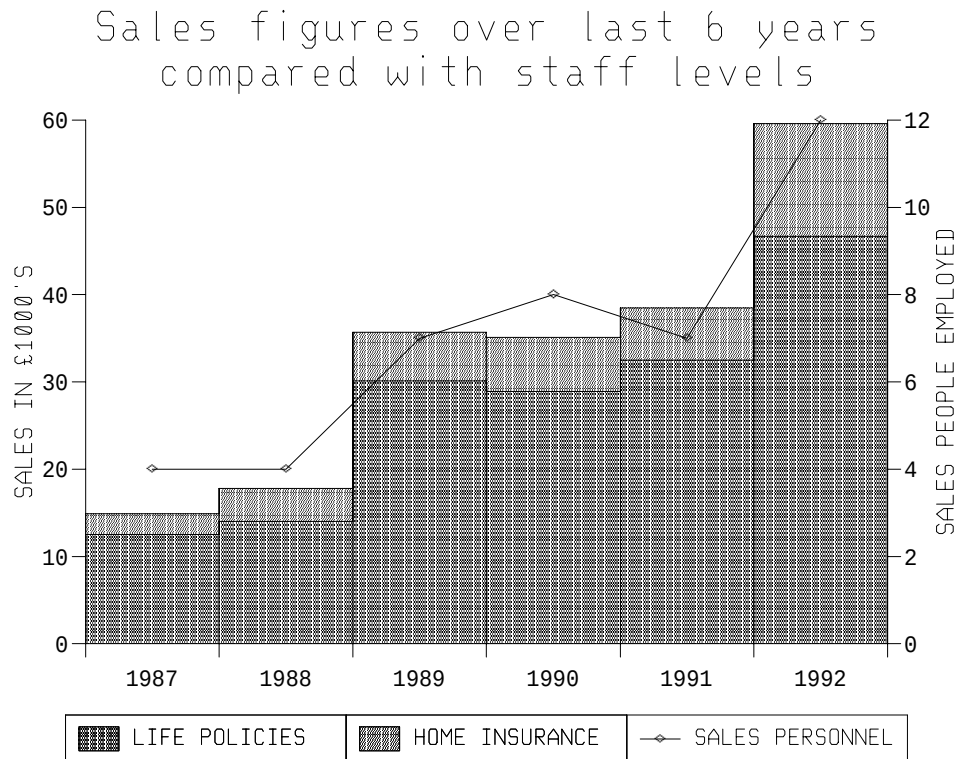


Figure 31. Using secondary axes

```

SALES: PROC OPTIONS(MAIN);
DCL H_ATTRS(4) FIXED BIN(31) INIT(2,3,0,190);/* Heading attribs */
DCL T_ATTRS(4) FIXED BIN(31) INIT(5,3,0,110);/* Title attribs */
DCL L_ATTRS(2) FIXED BIN(31) INIT(4,2);      /* Label attribs */
DCL COLORS(3) FIXED BIN(31) INIT(1,2,6);    /* Color table */
DCL (TYPE,MOD,COUNT) FIXED BIN(31);         /* Parameters for ASREAD */
CALL FSINIT;                                /* Initialize GDDM */

/*****
/*          Define chart heading
*****/
CALL CHHATT(4,H_ATTRS);                      /* Set heading attributes */
CALL CHHEAD(58,
'Sales figures over last 6 years;compared with staff levels');

/*****
/*          Define axis options
*****/
CALL CHXLAB(7,4,'****198719881989199019911992'); /* 1st label */
/* not used */
CALL CHXSET('LABMIDDLE');                    /* Labels between ticks */
CALL CHYTTL(16,'SALES IN $1000'S');          /* Primary y-axis title */
CALL CHTATT(4,T_ATTRS);                      /* Axis title attributes */
CALL CHLATT(2,L_ATTRS);                      /* Axis label attributes */
CALL CHYSEL(2);                              /* Select secondary y axis */
CALL CHYTTL(21,'SALES PEOPLE EMPLOYED');    /* Secondary y-axis title */
CALL CHYSET('PTICK');                        /* Positive ticks for
/* the secondary y axis */

```

```

/*****
/* Define color table, legend options, and text */
/*****
CALL CHCOL(3,COLORS);          /* Set color table */
CALL CHSET('KBOX');           /* Legend is boxed */
CALL CHKEYP('H','B','C');     /* Legend in bottom margin*/
CALL CHKEY(3,15,'LIFE POLICIES HOME INSURANCE SALES PERSONNEL');

/*****
/* Declare histogram chart data */
/*****
DCL XLOW(6) FLOAT DEC(6) INIT( 0, 1, 2, 3, 4, 5); /*Range starts*/
DCL XHI(6)  FLOAT DEC(6) INIT( 1, 2, 3, 4, 5, 6); /*Range ends */

DCL YDATA(12) FLOAT DEC(6) INIT(12.5, 14, 30.1, 28.9, 32.5, 46.7,
                                2.4, 3.8, 5.6, 6.2, 6.0, 12.9);

/*****
/* Histogram plotting call */
/*****
CALL CHSET('RELATIVE');       /* Stacked chart type, so relative*/
                                /* Data should be used */
CALL CHYSEL(1);               /* Reselect primary y axis */
                                /* to plot the sales figures */
CALL CHHIST(2,6,XLOW,XHI,YDATA); /* Plot the histogram */

/*****
/* Declare line-graph chart data */
/*****
                                /* Plotting points for the line */
                                /* graph are the midpoints of bars*/
DCL XDAT2(6) FLOAT DEC(6) INIT( 1, 2, 3, 4, 5, 6);

DCL YDAT2(6) FLOAT DEC(6) INIT( 4, 4, 7, 8, 7, 12);

/*****
/* Line-graph plotting call */
/*****
CALL CHYSEL(2);               /* Select secondary y axis for */
                                /* the second (line-graph) plot */
CALL CHPLOT(1,6,XDAT2,YDAT2);

/*****
/* Send chart to terminal */
/*****
CALL CHTERM;                  /* Terminate PGF */
CALL ASREAD(TYPE,MOD,COUNT);  /* Send chart to terminal */
CALL FSTERM;                  /* Terminate GDDM */
%INCLUDE(ADMUPINA);           /* Include GDDM and */
%INCLUDE(ADMUPINC);           /* PGF entries */
%INCLUDE(ADMUPINF);
END SALES;

```

These points should be noted when using secondary axes:

- Most options and attributes can be set only in state-1. Where these options apply to a secondary axis, they must nevertheless be set before the first plotting statement for the primary axes. That is the purpose of using CHXSEL or CHYSEL before the first plot - to set options for one or other set of axes.

- After you have moved to state-2 (by issuing the first plotting call), CHXSEL and CHYSEL are used to indicate the axes against which the next plot should be made.

Initially, or when you return to state-1 with a CHSTRT or CHRNIT call, the primary axes are selected. Thus, in the above program the first CHYTTL call refers to the primary y axis.

- Secondary axes are autoscaled according to the data used for the first plot against them. Explicit scaling (using CHXRNG or CHYRNG) must be made in state-1, before the first plot against the primary axes.
- Either x axis can be paired with either y axis for a particular plot.

Duplicate axes

A duplicate axis is a special case of a secondary axis. It is a copy of the matching primary axis (with the same range, tick marks, scaling factor, and axis attributes). The copying of such options applies regardless of whether other values were originally set for the secondary axis.

A duplicate axis is positioned at the opposite extreme of the chart from the matching primary axis, and is built at the same time as the primary axis.

These are the options that control the appearance of duplicate axes:

```
CALL CHSET('XDUP');           /* Create duplicate x axis      */
CALL CHSET('XNODUP');         /* No duplicate x axis (default) */

CALL CHSET('YDUP');           /* Create duplicate y axis      */
CALL CHSET('YNODUP');         /* No duplicate y axis (default) */
```

A duplicate axis can have its own title, if required.

How to prevent the data overwriting the axes

By default, the current axes are built before the first plot is made. The first plot and any following plots may then obliterate part of the axes if they lie within the plotting area. This affects grid lines and labels too, because they are drawn at plotting time.

This problem can be overcome by delaying (or repeating) the step of drawing the axes. These are the calls involved:

```
CALL CHSET('IDRAW');          /* Axes are drawn immediately before */
                               /* the first plot (default)           */

CALL CHSET('DRAW');           /* Axes are (re)drawn after every     */
                               /* plotting call, overlaying the data */

CALL CHSET('NDRAW');          /* Axes are not drawn on any plotting */
                               /* call. (CHDRAX is used instead).    */

CALL CHDRAX;                  /* Construct the axes now. (This call can */
                               /* be made only in state-2).           */
```

The DRAW option involves extra processing on each plotting call, so it is preferable to use NDRAW combined with a final CHDRAX call.

Appendix A. Example 1. Display of sales figures

Description of the program

The first example program displays sales figures for three different sales regions. They are shown in a procedural alphanumeric grid on the left-hand side of the screen and displayed in presentation graphics form on the right-hand side of the screen.

The user can alter the sales data by typing new figures into the grid and then choose which time periods (months) and sales regions the presentation graphics should display.

He or she can also choose which chart type is used. Line graphs, surface charts, histograms, composite and floating bar charts, pie charts, and scatter plots are permitted.

A second page provides help information for the operator who is not familiar with the program.

Source code

This is the source code of the program:

```
SALES: PROC OPTIONS(MAIN);
/*****
/*
/* THIS PROGRAM DISPLAYS A GRID OF SALES FIGURES */
/* ON THE LEFT HALF OF THE SCREEN AND A MATCHING */
/* BUSINESS CHART ON THE RIGHT HALF OF THE      */
/* SCREEN. NEW SALES FIGURES MAY BE TYPED INTO  */
/* THE GRID AND VARIOUS CHART TYPES REQUESTED.  */
/*
/*
*****/
```

example programs

```
DCL (TYPE,PFKEY,COUNT) FIXED BIN(31);    /* PARAMETERS FOR 'ASREAD' */
DCL MONTH(12) CHAR(3) INIT('JAN','FEB','MAR','APR','MAY','JUN','JLY',
                             'AUG','SEP','OCT','NOV','DEC');

DCL DATA(3,12) FLOAT DEC(6)              /* INITIAL SALES FIGURES */
INIT(100,110,112,123,147,110,134,146,178,176,164,190,
     80, 72, 90,110,120, 92,117,126,132,108,134,165,
     134,145,123,145,176,143,156,160,139,150,144,155);
                                           /* (See note 1) */
DCL X_DATA(12) INIT(1,2,3,4,5,6,7,8,9,10,11,12); /* X DATA (MONTHS) */

DCL PLOT_DATA(36) FLOAT DEC(6);           /* Y DATA PASSED TO PGF */
DCL GTOTAL FIXED BIN(15);                 /* HOLDS GRAND SALES TOTAL */
DCL (DEPTH,WIDTH) FLOAT DEC(6);           /* PARAMETERS FOR GSQPS */
DCL XLOW FLOAT DEC(6) INIT(1);             /* LOW END OF X RANGE */
DCL XHI FLOAT DEC(6) INIT(12);            /* HIGH END OF X RANGE */
DCL (FIELD_IDS(1),LENGTHS(1)) FIXED BIN(31); /* ASQMOD PARAMETERS */

                                           /* INTENSITY    UPPERCASE TRANSLN. */
DCL ARITH_ATTRS(9) FIXED BIN(31) INIT(0,1, 0,0,0,0,0,0, 1);
                                           /* (See note 2) */

DCL COMPS FIXED BIN(31) INIT(3);          /* NUMBER OF COMPONENTS */
DCL KEY_TEXT CHAR(15);                    /* LEGEND TEXT */
DCL ELEMENTS FIXED BIN(31) INIT(12);      /* NUMBER OF ELEMENTS */
DCL PICZZZ9 PIC'ZZZ9';                    /* PICTURE VARIABLES FOR.. */
DCL PICZZZZ9 PIC'ZZZZ9';                  /* ..EDITING INPUT NUMBERS */
DCL CHAR2 CHAR(2);                         /* CHARACTER TEMPORARIES */
DCL CHAR4 CHAR(4);                         /*          "          */
DCL CHAR5 CHAR(5);                         /*          "          */
```

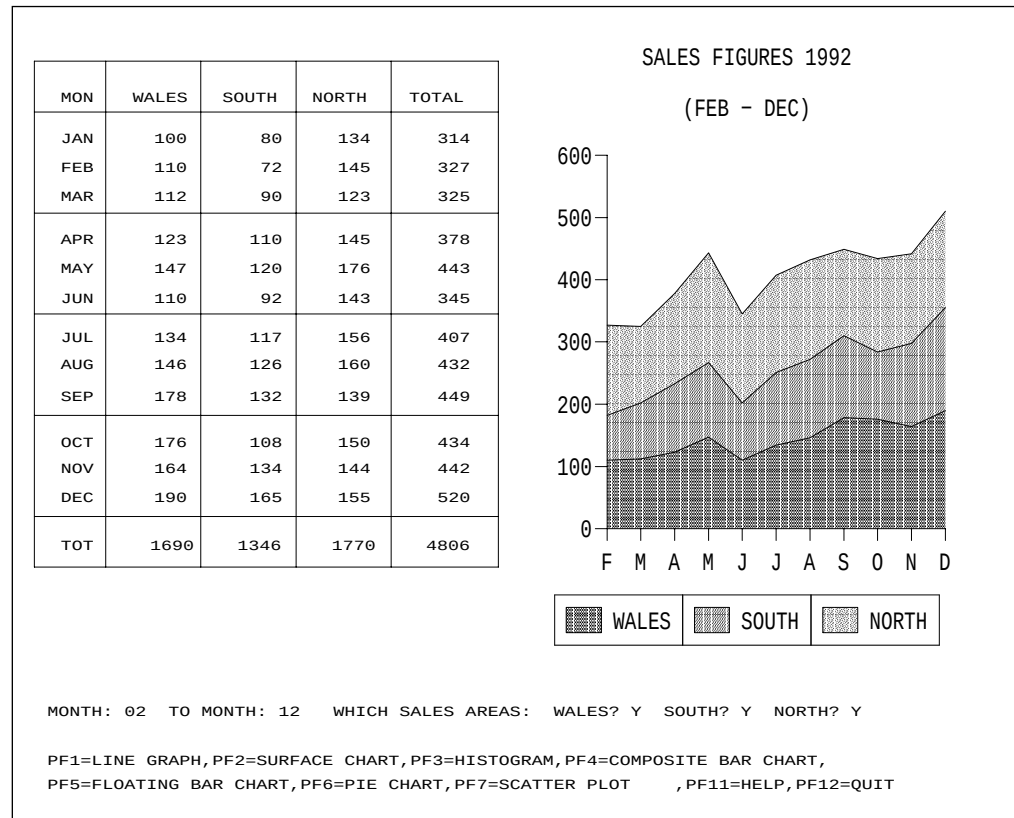


Figure 32. Sales program: A surface chart

```

DCL A_NAME(3) CHAR(5) INIT('WALES','SOUTH','NORTH'); /* LEGEND TEXT */
DCL COLUMN(3) CHAR(1) INIT('Y','Y','Y'); /* WHICH COMPTS. DISPLAYED */
DCL (ONSOURCE,MOD) BUILTIN; /* PL/I BUILTIN FUNCTIONS */
DCL HATTS(2) FIXED BIN(31) INIT(2,2); /* PGF ATTRIBUTES */
DCL KATTS(2) FIXED BIN(31) INIT(7,2); /* " " */
DCL LATTS(2) FIXED BIN(31) INIT(4,2); /* " " */
DCL BATTS(1) FIXED BIN(31) INIT(8); /* " " */
DCL DCOL(3) FIXED BIN(31) INIT(6, /* FIRST SALES AREA YELLOW */
1, /* SECOND BLUE */
5); /* THIRD TURQUOISE */

CALL FSINIT; /* INITIALIZE GDDM */

/*****
/*
/* CREATE PAGE TO HOLD ALPHANUMERIC
/* ON THE LEFT, AND PRESENTATION
/* GRAPHICS ON THE RIGHT
/*
*****/
CALL FSPCRT(1,0,0,2); /* CREATE PAGE 1
CALL GSFLD(1,1,32,80); /* CREATE GRAPHICS FIELD
CALL GSWIN(0.5,80.5,32.5,0.5); /* DEFINE WINDOW TO MATCH
/* (See note 3)
/* ROWS AND COLUMNS

```

example programs

```
CALL GSSEG(0);                                /* OPEN GRAPHICS SEGMENT */
CALL ASDFLT(9,ARITH_ATTRS);                   /* MAKE UPPERCASE TRANS- */
                                              /* LATION THE DEFAULT    */
                                              /* (See note 2)          */
CALL ALPHA_GRID;                             /* CALL SUBROUTINE TO    */
                                              /* DISPLAY THE SALES GRID*/
CALL ASDFLD(80,31,2,2,77,2);                 /* DEFINE ALPHA FIELD TO */
                                              /* DISPLAY PFKEY INFO.   */

/* ASSIGN PFKEY INFORMATION TO ALPHANUMERIC FIELD 80 */
CALL ASCPUT(80,154,'PF1=LINE GRAPH,PF2=SURFACE CHART,PF3=HI'||
                'STOGRAM,PF4=COMPOSITE BAR CHART,PF5=FLOATING BAR CHART,PF6=PIE CHART,PF7=SCATTER PLOT,PF11=HELP,PF12=QUIT');
/* ASSIGN CHARACTER COLOR ATTRIBUTES TO ALPHANUMERIC FIELD 80 */
CALL ASCCOL(80,154,'44445555555555554444555555555555444455' ||
                '55555554444455555555555555555555554' ||
                '444455555555555555555555555544445555555555444' ||
                '445555555555555555555544444455554444445555');
                                              /* (See note 4)          */

CALL ASDFLD(81,29,2,1,6,2);                  /* PROTECTED FIELD (PROMPT) */
CALL ASCPUT(81,6,'MONTH:');                  /* ASSIGN PROMPT DATA      */
CALL ASFCOL(81,4);                           /* SET FIELD COLOR TO GREEN */
CALL ASDFLD(82,29,9,1,2,0);                  /* UNPROTECTED FIELD..     */
CALL ASCPUT(82,2,' 1');                      /* .. INITIALIZED TO 1      */
CALL ASFCOL(82,2);                           /* SET FIELD COLOR TO RED   */
CALL ASDFLD(83,29,12,1,9,2);
CALL ASCPUT(83,9,'TO MONTH:');
CALL ASFCOL(83,4);
CALL ASDFLD(84,29,22,1,2,0);
CALL ASCPUT(84,2,'12');
CALL ASFCOL(84,2);

CALL ASDFLD(95,29,28,1,18,2);
CALL ASCPUT(95,18,'WHICH SALES AREAS:');
CALL ASFCOL(95,4);
CALL ASDFLD(85,29,48,1,6,2);
CALL ASCPUT(85,6,'WALES?');
CALL ASFCOL(85,4);
CALL ASDFLD(86,29,55,1,1,0);
CALL ASFCOL(86,2);
CALL ASCPUT(86,1,'Y');
CALL ASDFLD(87,29,58,1,6,2);
CALL ASCPUT(87,6,'SOUTH?');
CALL ASFCOL(87,4);
```

```

CALL ASDFLD(88,29,65,1,1,0);
CALL ASFCOL(88,2);
CALL ASCPUT(88,1,'Y');
CALL ASDFLD(89,29,68,1,6,2);
CALL ASCPUT(89,6,'NORTH?');
CALL ASFCOL(89,4);
CALL ASDFLD(90,29,75,1,1,0);
CALL ASFCOL(90,2);
CALL ASCPUT(90,1,'Y');

```

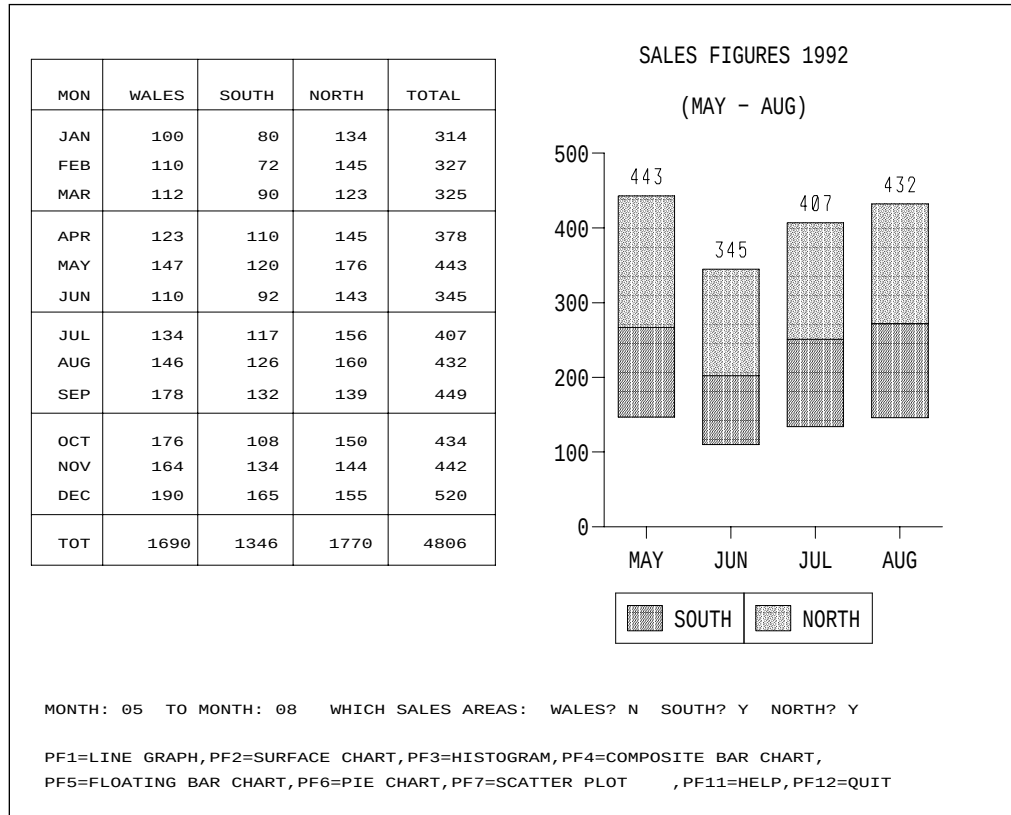


Figure 33. Sales program: A floating bar chart

```

/*****
/* QUERY PICTURE SPACE */
/*****
CALL GSQPS(WIDTH,DEPTH);
/*****
/* SET CHART AREA */
/*****
CALL CHAREA(0.51*WIDTH,WIDTH,0.2*DEPTH,0.95*DEPTH); /* (See note 5) */
CALL CHBATT(1,BATTS); /* SET CHART BOX ATTRIBUTES */
CALL CHHATT(2,HATTS); /* SET HEADING TEXT ATTRIBUTES */
CALL CHKATT(2,KATTS); /* SET LEGEND TEXT ATTRIBUTES */
CALL CHLATT(2,LATTS); /* SET LABEL TEXT ATTRIBUTES */
CALL CHVMAR(8,3); /* SET VERTICAL MARGINS */
CALL CHXTIC(1.0,0.0); /* SPECIFY X-AXIS TICK INTERVAL*/
CALL CHSET('LETTER'); /* 1-CHAR MONTH ABBREVIATIONS */
CALL CHSET('CBAC'); /* SPECIFY A CHART BACKGROUND */
/* (See note 6) */

```

example programs

```

CALL CHSET('KBOX');          /* SPECIFY A LEGEND BOX */
CALL CHKEYP('H','B','C');    /* LEGEND CENTERED AT BOTTOM */
CALL CHKEY(3,5,'WALESSOUTHNORTH'); /* INITIAL KEY TEXT */
CALL CHCOL(3,DCOL);          /* SET DATA COLORS TO MATCH*/
                              /* FIELD COLORS */
CALL CHCGRD(40,26);          /* SET CHART GRID (See note 7) */

/*****
/*
/*    NOW CREATE PAGE TO HOLD HELP INFORMATION
/*
/*
/*****
CALL FSPCRT(2,0,0,1);        /* CREATE PAGE 2 */
CALL ASDFLD(1,1,17,1,45,2); /* ALPHA FIELD FOR HEADING */
CALL ASCPUT(1,45,'INSTRUCTIONS FOR THE BUSINESS GRAPHICS SAMPLE');
CALL ASFCOL(1,5);           /* HEADING IN TURQUOISE */

```

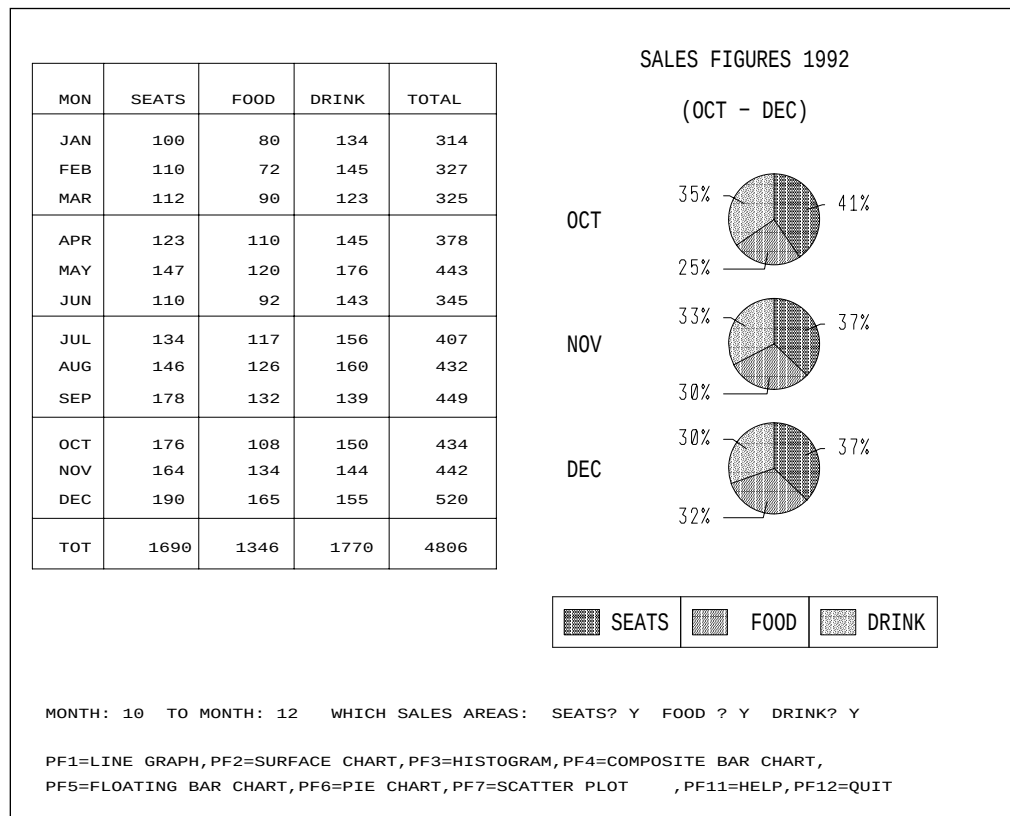


Figure 34. Sales program: Proportional pie charts

```

CALL ASDFLD(2,3,10,10,60,2);          /* 10 BY 60 FIELD FOR HELP INFO*/
CALL ASFCOL(2,4);                      /* HELP INFORMATION IN GREEN */
CALL ASCPUT(2,600,
'1. You may oertype the names of the sales-areas (at the top' ||
'of the grid). These names then appear in the legend.      ' ||
'2. You may oertype also the month-by-month sales data. The ' ||
'totals are then updated automatically.                    ' ||
'3. You may select which of the three sales-areas should be' ||
'plotted. This determines the number of data components.   ' ||
'4. You may select which months'' sales figures should be  ' ||
'included. This determines the x-axis range.                ' ||
'5. Finally, you may choose which form of chart is used. The' ||
'possibilities are listed in the PFkey information.         ');

```

```

CALL ASDFLD(3,17,10,1,60,2);
CALL ASFCOL(3,4);
CALL ASCPUT(3,60,
'      These points should be noted:                        ');
CALL ASDFLD(4,19,10,1,54,2);
CALL ASFCOL(4,4);
CALL ASCPUT(4,60,
'      (1) Faulty arithmetic input is cancelled and');

```

```

CALL ASDFLD(5,19,65,1,4,2);
CALL ASFCOL(5,2);
CALL ASCPUT(5,4,'0000');
CALL ASDFLD(6,20,10,9,60,2);
CALL ASFCOL(6,4);

```

```

CALL ASCPUT(6,540,
'displayed in red. You should then correct it.            ' ||
'      (2) There is a limit of 3 pie charts. If your month- ' ||
'range is more than 3, the first 3 are displayed.          ' ||
'      (3) All arithmetic input should be entered as integers.' ||
'No check is made, though, and input such as 13.4 (or even ' ||
'3E5) are not rejected by the program. Any results from such ' ||
'data are not guaranteed.                                   ' ||
'      (4) Alphanumeric information may be entered in upper or' ||
'lower case. It is folded to upper case and corrected.     ');

```

```

CALL ASDFLD(7,32,18,1,43,2);
CALL ASCPUT(7,43,'HIT ANY KEY TO RETURN TO THE EDITING SCREEN');
CALL ASFCOL(7,6);          /* RETURN PROMPT IN YELLOW */
/*****/
/* RESELECT PAGE 1 */
/*****/
CALL FSPSEL(1);            /* RESELECT PAGE 1 */

```

example programs

```

/*****
/* TOP OF LOOP FOR ACCEPTING INPUT AND DISPLAYING CHART */
*****/
DISPLAY;;
CALL ASFCUR(82,1,1);          /* CURSOR ON 1ST INPUT FIELD */

/*****/
/* ISSUE SCREEN READ */
/*****/
CALL ASREAD(TYPE,PFKEY,COUNT); /* SEND GRID AND CHART */

IF PFKEY=12 THEN GOTO EXIT;    /* EXIT IF PF12 WAS PRESSED */

IF PFKEY=11 THEN DO;          /* DISPLAY HELP INFO, IF PF11 */
                              /* (See note 8) */
                              /* SELECT PAGE 2 (HELP INFO) */
CALL FSPSEL(2);               /* DISPLAY IT ON SCREEN */
CALL ASREAD(TYPE,PFKEY,COUNT); /* RESELECT PAGE 1 */
CALL FSPSEL(1);               /* RETURN TO TOP OF LOOP */
GOTO DISPLAY;                 /* END - DISPLAY HELP INFO */
END;

/*****/
/*
/* QUERY MODIFIED FIELDS AND PROCESS ALL INPUT DATA
/*
*****/
QMOD_AGAIN;;
DO I=1 TO COUNT;
CALL ASQMOD(1,FIELD_IDS,LENGTHS,LENGTHS);
ON CONVERSION BEGIN;          /* CORRECT IF CONVERSION ERROR */
ONSOURCE='0000';              /* (See note 11) */
CALL ASCPUT(FIELD_IDS(1),4,'0000');
CALL ASFCOL(FIELD_IDS(1),2);   /* BAD FIELD IN RED */
CALL FSALRM;
END;

SELECT;                        /* PL/I SELECT STATEMENT; */
                              /* ACTION TAKEN FOLLOWS FIRST */
                              /* TRUE-WHEN CLAUSE */

/*****/
/* START-MONTH SET */
/*****/
WHEN (FIELD_IDS(1)=82) DO;     /* START-MONTH HAS BEEN SET */
CALL ASCGET(82,2,CHAR2);      /* RETRIEVE THE NEW START-MONTH*/

```

```

      INSTRUCTIONS FOR THE BUSINESS GRAPHICS SAMPLE

1. You may overtype the names of the sales-areas (at the top
of the grid). These names will then appear in the legend.
2. You may overtype also the month-by-month sales data. The
totals will then be updated automatically
3. You may select which of the three sales-areas should be
plotted. This will determine the number of data components
4. You may select which months' sales figures should be
included. This will determine the x-axis range.
5. Finally, ycu may choose which form of chart is used. The
possibilities are listed in the PFkey information.

      These points should be noted:

      (1) Faulty arithmetic input will be cancelled and      0000
displayed in red. You should then correct it
      (2) There is a limit of 3 pie charts. If your month-
range is more than 3, the first 3 will be displayed
      (3) All arithmetic input should be entered as integers.
No check is made, though, and input such as 13.4 (or even
3E5) will not be rejected by the program. Any results from
such data are not guaranteed.
      (4) Alphanumeric information may be entered in upper or
lower case. It will be folded to upper case and corrected.

      HIT ANY KEY TO RETURN TO THE EDITING SCREEN

```

Figure 35. Sales program: The help information

```

ON CONVERSION BEGIN;                                /* CORRECT THE INPUT IF.. */
ONSOURCE='01';                                       /* ..CONVERSION ERROR OCCURRED */
CALL ASCPUT(82,2,'01');                             /* SEND OUT CORRECTED FIELD */
END;                                                  /* (See note 9) */

XLOW=CHAR2;                                           /* SAVE THE START-MONTH */
IF (XLOW<1)|(XLOW>11) THEN DO;                       /* CORRECT, IF INVALID */
XLOW=1;
CALL ASCPUT(82,2,'01');
END;                                                  /* END - CORRECT, IF INVALID */
END;                                                  /* END - START-MONTH SET */

/*****
/* END-MONTH SET */
*****/
WHEN (FIELD_IDS(1)=84) DO;
CALL ASCGET(84,2,CHAR2);                             /* RETRIEVE END-MONTH */
ON CONVERSION BEGIN;                                 /* CORRECT, IF CONVERSION ERROR*/
ONSOURCE='12';
CALL ASCPUT(84,2,'12');
END;

```

```
XHI=CHAR2;                                /* SAVE END-MONTH          */
IF (XHI<2)|(XHI>12) THEN DO;              /* CORRECT, IF INVALID    */
XHI=12;
CALL ASCPUT(84,2,'12');
END;
END;
```

```

/*****
/* INCLUSION OF 1ST COMPONENT SET */
*****/
WHEN (FIELD_IDS(1)=86) DO;
CALL ASCGET(86,1,COLUMN(1));              /* RETRIEVE INCLUSION REQUEST */
IF (COLUMN(1)≠'Y')&(COLUMN(1)≠'N') THEN DO;
COLUMN(1)='Y';                            /* IF OPERATOR DOES NOT TYPE */
END;                                       /* X OR Y, SET Y           */
CALL ASCPUT(86,1,COLUMN(1));              /* RE-SEND, IN CASE FIXED   */
END;
```

```

/*****
/* INCLUSION OF 2ND COMPONENT SET */
*****/
WHEN (FIELD_IDS(1)=88) DO;
CALL ASCGET(88,1,COLUMN(2));
IF (COLUMN(2)≠'Y')&(COLUMN(2)≠'N') THEN DO;
COLUMN(2)='Y';
END;
CALL ASCPUT(88,1,COLUMN(2));
END;
```

```

/*****
/* INCLUSION OF 3RD COMPONENT SET */
*****/
WHEN (FIELD_IDS(1)=90) DO;
CALL ASCGET(90,1,COLUMN(3));              /*      (See note 10)      */
IF (COLUMN(3)≠'Y')&(COLUMN(3)≠'N') THEN DO;
COLUMN(3)='Y';
END;
CALL ASCPUT(90,1,COLUMN(3));
END;
```

```

/*****
/* NAME OF 1ST SALES AREA SET */
*****/
WHEN (FIELD_IDS(1)=102) DO;
CALL ASCGET(102,5,A_NAME(1));
CALL ASCPUT(102,5,A_NAME(1));
/* REDISPLAY FIELD 102, AS */
/* UPPERCASE TRANSLATION MAY */
/* HAVE OCCURRED (See note 10) */

CALL ASCPUT(85,6,A_NAME(1)||'?');
END;

```

```

/*****
/* NAME OF 2ND SALES AREA SET */
*****/
WHEN (FIELD_IDS(1)=103) DO;
CALL ASCGET(103,5,A_NAME(2));
CALL ASCPUT(103,5,A_NAME(2));
CALL ASCPUT(87,6,A_NAME(2)||'?');
END;

```

```

/*****
/* NAME OF 3RD SALES AREA SET */
*****/
WHEN (FIELD_IDS(1)=104) DO;
CALL ASCGET(104,5,A_NAME(3));
CALL ASCPUT(104,5,A_NAME(3));
CALL ASCPUT(89,6,A_NAME(3)||'?');
END;

```

```

/*****
/* SALES DATA FROM AREA 1 HAS BEEN CHANGED */
*****/
WHEN (FIELD_IDS(1)<223) DO;          /* IN RANGE 211-222 */
CALL ASCGET(FIELD_IDS(1),4,CHAR4); /* RETRIEVE NEW SALES DATA */
CALL ASFCOL(FIELD_IDS(1),DCOL(1)); /* GOOD FIELD IN NORMAL COLOR */
DATA(1,FIELD_IDS(1)-210)=CHAR4;
END;

```

```

/*****
/* SALES DATA FROM AREA 2 HAS BEEN CHANGED */
*****/
WHEN (FIELD_IDS(1)<323) DO;          /* IN RANGE 311-322 */
CALL ASCGET(FIELD_IDS(1),4,CHAR4);
CALL ASFCOL(FIELD_IDS(1),DCOL(2)); /* GOOD FIELD IN NORMAL COLOR */
DATA(2,FIELD_IDS(1)-310)=CHAR4;
END;

```

example programs

```

/*****
/* SALES DATA FROM AREA 3 HAS BEEN CHANGED */
*****/
WHEN (FIELD_IDS(1)<423) DO;          /* IN RANGE 411-422          */
CALL ASCGET(FIELD_IDS(1),4,CHAR4);
CALL ASFCOL(FIELD_IDS(1),DCOL(3)); /* GOOD FIELD IN NORMAL COLOR */
DATA(3,FIELD_IDS(1)-410)=CHAR4;
END;

END;                                /* END OF SELECT STATEMENT */

END; /* I-LOOP PROCESSING EACH FIELD STARTED AT QMOD_AGAIN */

/*****
/* ALL MODIFIED FIELDS PROCESSED */
*****/
ALL_DONE;;
/*****
/* NOW CALCULATE THE CROSS-TOTALS */
*****/

CALL DISPLAY_TOTALS;                /* CALL SUBROUTINE THAT.. */
                                     /* CALCULATES AND DISPLAYS */
                                     /* THE SUB-TOTALS IN GRID */

IF TYPE=0 THEN GOTO DISPLAY;        /* NO PLOT REQUESTED, RETURN*/

/*****
/*
/* DISCOVER HOW MANY COMPONENTS (SALES AREAS) TO BE PLOTTED */
/*
*****/
COMPS=0;                            /* INITIALIZE COMPONENT COUNT */
IF COLUMN(1)='Y' THEN DO;           /* 1ST AREA IS PLOTTED      */
COMPS=1;                            /* BUMP COMPONENT COUNT     */
KEY_TEXT=A_NAME(1);                /* PUT 1ST NAME IN LEGEND TEXT*/
END;                                /* END - 1ST AREA          */
IF COLUMN(2)='Y' THEN DO;           /* 2ND AREA IS PLOTTED     */
COMPS=COMPS+1;                    /* BUMP COMPONENT COUNT     */
SUBSTR(KEY_TEXT,(COMPS-1)*5+1,5)=A_NAME(2); /* 2ND NAME TO
                                     LEGEND TEXT
END;                                /* END - 2ND AREA          */

```

```

IF COLUMN(3)='Y' THEN DO;          /* 3RD AREA IS PLOTTED      */
COMPS=COMPS+1;                     /* BUMP COMPONENT COUNT    */
SUBSTR(KEY_TEXT,(COMPS-1)*5+1,5)=A_NAME(3);/*3RD NAME TO LEGEND TEXT*/
END;                               /* END - 3RD AREA          */

CALL CHKEY(COMPS,5,KEY_TEXT);      /* SPECIFY LEGEND TEXT     */
/* (See note 12)                  */
CALL CHXSET('LABADJACENT');        /* LABELS NEXT TO TICKS   */

CALL CHXMTH(XLOW);                 /* SPECIFY MONTH LABELS,   */
/* STARTING WITH FIRST-MONTH      */

/*****
/*
/* DISCOVER HOW MANY ELEMENTS (MONTHS) ARE TO BE PLOTTED
/*
/*
*****/
IF (XHI<XLOW) THEN DO;             /* CORRECT XHI IF TOO LOW */
XHI=12;
CALL ASCPUT(84,2,'12');
END;

CALL CHHEAD(32,
'Sales Figures 1992; ;('||MONTH(XLOW)||' - '||MONTH(XHI)||')');

IF PFKEY=3 THEN CALL CHXRNG(XLOW-1.0,XHI);/* EXTEND THE RANGE, IF A*/
/* HISTOGRAM (See note 13) */
ELSE IF (PFKEY=4)|(PFKEY=5) THEN      /* EXTEND THE RANGE, IF A */
CALL CHXRNG(0.5,XHI-XLOW+1.5);/* BAR CHART (See note 14) */
ELSE CALL CHXRNG(XLOW,XHI);
ELEMENTS=XHI-XLOW+1;                 /* CALCULATE NO. OF ELEMENTS*/
IF ELEMENTS<5 THEN CALL CHSET('ABREV'); /* (See note 15)          */
ELSE CALL CHSET('LETTER');

/*****
/*
/* COPY ELEMENTS INTO THE ARRAY TO BE PASSED TO PGF
/*
/*
*****/
/* PIE CHART */
/*****
IF PFKEY=6 THEN DO;                 /* PIE CHART DATA TREATED SEPARATELY */

```

example programs

```
INDEX=0;                                /* INITIALIZE COUNT OF ELEMENTS */
COMPS=0;                                /* INITIALIZE COMPONENT COUNT */
DO I=1 TO 3;                            /* MAXIMUM OF 3 PIE CHARTS */
  COMPS=COMPS+1;                        /* BUMP COUNT OF COMPONENTS */
  DO J=1 TO 3;                          /* MAXIMUM OF 3 SALES-AREAS */
    IF COLUMN(J)='Y' THEN DO;          /* THIS SALES-AREA TO BE PLOTTED */
      INDEX=INDEX+1;                  /* BUMP TOTAL ELEMENT COUNT */
      PLOT_DATA(INDEX)=DATA(J,XLOW+I-1); /* ASSIGN DATA TO ARRAY */
    END;                              /* THIS SALES-AREA TO BE PLOTTED */
  END;                                  /* J-LOOP */
  IF (XHI-XLOW)<I THEN GOTO LEAVE_LOOP; /* ALL PIES PROCESSED */
END;                                    /* I-LOOP */

LEAVE_LOOP;;
ELEMENTS=INDEX/COMPS;                  /* CALCULATE ELEMENTS PER PIE */

END;                                    /* PIE CHART DATA PROCESSING */

/*****/
/* NOT PIE CHART */
/*****/
ELSE DO;                              /* NOT PIE CHART */
  IF COLUMN(1)='Y' THEN SALES_AREA=1; /* INITIALIZE SALES AREA */
  ELSE SALES_AREA=2;
END;

DO J=1 TO COMPS;                      /* LOOP BY COMPONENT */
  IF COLUMN(SALES_AREA)='N' THEN SALES_AREA=SALES_AREA+1; /* NEXT S-AREA */
  DO I=XLOW TO XHI;                   /* MOVE DATA TO ARRAY */
    PLOT_DATA((J-1)*ELEMENTS+I-XLOW+1)=DATA(SALES_AREA,I);
  END;                                /* I-LOOP */
  SALES_AREA=SALES_AREA+1;            /* BUMP TO NEXT SALES-AREA */
END;                                  /* J-LOOP */

/*****/
/* SET THE X DATA */
/*****/
DO I=1 TO ELEMENTS;
  X_DATA(I)=XLOW+I-1;
END;
END;                                  /* NOT PIE CHART */
```

```

/*****
/*
/*      PF KEY 1 PRESSED: PLOT LINE GRAPH      */
/*      PF KEY 7 PRESSED: SCATTER PLOT          */
/*
*****/
IF (PFKEY=1)|(PFKEY=7) THEN DO;
CALL CHSET('ABSOLUTE');          /* REQUEST ABSOLUTE DATA      */
IF PFKEY=1 THEN CALL CHSET('LINES'); /* PF1 REQUESTS LINE GRAPH    */
      ELSE CALL CHSET('NOLINES'); /* PF7 REQUESTS SCATTER PLOT  */
CALL CHSET('YVERT');             /* RESET Y VERTICAL           */

CALL CHPLOT(COMPS,ELEMENTS,X_DATA,PLOT_DATA); /* MAKE THE PLOT      */
END;

/*****
/*
/*      PF KEY 2 PRESSED: PLOT SURFACE CHART    */
/*
*****/
ELSE IF PFKEY=2 THEN DO;
CALL CHSET('RELATIVE');          /* BECAUSE STACKED-CHART TYPE */
CALL CHSET('YVERT');             /* RESET Y VERTICAL           */
CALL CHSURF(COMPS,ELEMENTS,X_DATA,PLOT_DATA); /* PLOT SURFACE CHART        */
END;

/*****
/*
/*      PF KEY 3 PRESSED: PLOT HISTOGRAM        */
/*
*****/
ELSE IF PFKEY=3 THEN DO;
CALL CHSET('RELATIVE');          /* BECAUSE STACKED-CHART TYPE */
CALL CHXSET('LABMIDDLE');        /* LABELS BETWEEN TICKS      */
CALL CHSET('YVERT');             /* RESET Y VERTICAL           */
CALL CHHIST(COMPS,ELEMENTS,X_DATA-1,X_DATA,PLOT_DATA); /* PLOT THE          */
/*      HISTOGRAM (See note 16) */
END;

```

```

/*****
/*
/*      PF KEY 4 PRESSED: PLOT COMPOSITE BAR CHART      */
/*      PF KEY 5 PRESSED: PLOT FLOATING BAR CHART        */
/*
*****/
ELSE IF (PFKEY=4)|(PFKEY=5) THEN DO;
CALL CHSET('RELATIVE');          /* BECAUSE STACKED-CHART TYPE */
IF PFKEY=4 THEN CALL CHSET('CBAR'); /* PF4 REQUESTS COMPOSITE */
      ELSE CALL CHSET('FBAR');      /* PF5 REQUESTS FLOATING */
CALL CHSET('VALUES');            /* VALUES ON BARS */
CALL CHSET('YVERT');             /* RESET Y VERTICAL */
CALL CHBAR(COMPS,ELEMENTS,PLOT_DATA); /* PLOT BAR CHART */
END;

/*****
/*
/*      PF KEY 6 PRESSED: PLOT MULTIPLE PIE CHART      */
/*
*****/
ELSE IF PFKEY=6 THEN DO;
CALL CHSET('PROPIE');            /* PIES PROPORTIONAL TO DATA */
CALL CHSET('ABPIE');            /* DATA ABSOLUTE, NOT %AGE */
CALL CHSET('VALUES');           /* VALUES ON SPIDER TAGS */
CALL CHSET('XVERT');            /* PIES VERTICAL */
CALL CHSET('ABREV');            /* 3-LETTER MONTH ABBREVIATN.*/
CALL CHPIE(COMPS,ELEMENTS,PLOT_DATA); /* CONSTRUCT PIE CHARTS */
END;
CALL CHSTRT;                    /* RETURN TO STATE-1 */
GOTO DISPLAY;                   /* GO TO TOP OF LOOP */

EXIT;
CALL FSTERM;                    /* TERMINATE GDDM */

/*****
/*
/* THIS SUBROUTINE PUTS OUT THE FRAMEWORK OF */
/* THE ALPHANUMERIC GRID CONTAINING THE DATA */
/*
*****/
ALPHA_GRID: PROC;
CALL GSCOL(7);                  /* CURRENT COLOR TO WHITE */
DO Y=3,5,9,13,17,21,23;        /* LOOP OF HORIZONTAL LINES */
CALL GSMOVE(1.0,Y);
CALL GSLINE(40.0,Y);
END;

```

```

DO X=1,7,15,23,31,40;                /* LOOP OF VERTICAL LINES */
CALL GSMOVE(X,3.0);
CALL GSLINE(X,23.0);
END;
CALL GSSCLS;                          /* CLOSE GRAPHICS SEGMENT */
                                      /* (SEE NOTE 14) */

CALL ASDFLD(101,4,3,1,3,2);          /* PROTECTED PROMPT FIELD */
CALL ASDFLD(102,4,9,1,5,0);          /* 1ST SALES AREA NAME */
CALL ASDFLD(103,4,17,1,5,0);         /* 2ND SALES AREA NAME */
CALL ASDFLD(104,4,25,1,5,0);         /* 3RD SALES AREA NAME */
CALL ASDFLD(105,4,33,1,6,2);         /* PROTECTED PROMPT FIELD */
CALL ASCPUT(101,3,'MON');             /* ASSIGN DATA */
CALL ASFCOL(101,7);                  /* FIELD COLOR = WHITE */

CALL ASCPUT(102,5,'WALES');           /* INITIALIZE REGIONS ... */
CALL ASFCOL(102,DCOL(1));            /* ... AND SET THEIR COLORS */
CALL ASCPUT(103,5,'SOUTH');
CALL ASFCOL(103,DCOL(2));
CALL ASCPUT(104,5,'NORTH');
CALL ASFCOL(104,DCOL(3));
CALL ASCPUT(105,6,'TOTAL ');
CALL ASFCOL(105,7);                  /* TOTAL SHOWN IN WHITE */

```

```

/*****
/*          FIELD NUMBERING SCHEME          */
/*****
/*      * SALES AREA 1 * SALES AREA 2 * SALES AREA 3 * TOTAL */
/*****
/* JAN *   211      *   311      *   411      *   511 */
/* FEB *   212      *   312      *   412      *   512 */
/*    *    "        *    "        *    "        *    " */
/* DEC *   222      *   322      *   422      *   522 */
/*****
/* TOT *   223      *   323      *   423      *   523 */
/*****

```

example programs

```
J=111;
DO I=6,7,8,10,11,12,14,15,16,18,19,20; /* DEFINE SALES DATA FIELDS */
CALL ASDFLD(J,I,3,1,3,2);
CALL ASCPUT(J,3,MONTH(J-110));          /* ASSIGN MONTH NAMES      */
CALL ASDFLD(J+100,I,10,1,4,0);
CALL ASFCOL(J+100,DCOL(1));
CALL ASDFLD(J+200,I,18,1,4,0);
CALL ASFCOL(J+200,DCOL(2));
CALL ASDFLD(J+300,I,26,1,4,0);
CALL ASFCOL(J+300,DCOL(3));
CALL ASDFLD(J+400,I,34,1,5,2);          /* PROTECTED TOTAL COLUMN */
J=J+1;
END; /* I-LOOP */
```

```
CALL ASDFLD(123,22,3,1,3,2);            /* PROTECTED PROMPT FIELD */
CALL ASCPUT(123,3,'TOT');                /* ASSIGN DATA            */
CALL ASFCOL(123,7);                      /* FIELD COLOR = WHITE     */
CALL ASDFLD(223,22,9,1,5,2);             /* 1ST SALES AREA TOTAL   */
CALL ASFCOL(223,DCOL(1));
CALL ASDFLD(323,22,17,1,5,2);            /* 2ND SALES AREA TOTAL   */
CALL ASFCOL(323,DCOL(2));
CALL ASDFLD(423,22,25,1,5,2);            /* 3RD SALES AREA TOTAL   */
CALL ASFCOL(423,DCOL(3));
CALL ASDFLD(523,22,34,1,5,2);            /*      GRAND TOTAL       */
CALL ASFCOL(523,7);                      /* FIELD COLOR = WHITE     */
DO J=1 TO 12;                            /* DISPLAY SALES DATA     */
PICZZZ9=DATA(1,J);                      /* CONVERT TO CHARACTER    */
CALL ASCPUT(J+210,4,PICZZZ9);            /* REGION 1, MONTH J      */
PICZZZ9=DATA(2,J);                      /* CONVERT TO CHARACTER    */
CALL ASCPUT(J+310,4,PICZZZ9);            /* REGION 2, MONTH J      */
PICZZZ9=DATA(3,J);                      /* CONVERT TO CHARACTER    */
CALL ASCPUT(J+410,4,PICZZZ9);            /* REGION 3, MONTH J      */
END; /* J-LOOP */
CALL DISPLAY_TOTALS;                     /* CALL SUBROUTINE TO CALCU-*/
/*      LATE AND DISPLAY TOTALS*/
```

```
END ALPHA_GRID;
```

```
/******
/* THIS SUBROUTINE CALCULATES AND DISPLAYS */
/* THE VARIOUS SUBTOTALS IN THE GRID      */
/******
DISPLAY_TOTALS: PROC;
```

```
DO J=1 TO 12;                            /* CALCULATE MONTHLY TOTALS */
PICZZZZ9=DATA(1,J)+DATA(2,J)+DATA(3,J); /* TOTAL FOR MONTH J        */
CALL ASCPUT(J+510,5,PICZZZZ9);          /* DISPLAY MONTHLY TOTAL    */
END; /* J-LOOP */
```

```

/*****
/* CALCULATE MONTH TOTALS */
*****/

PICZZZZ9=0;
DO I=1 TO 12;
PICZZZZ9=PICZZZZ9+DATA(1,I);
END;
GTOTAL=PICZZZZ9;
CALL ASCPUT(223,5,PICZZZZ9);

PICZZZZ9=0;
DO I=1 TO 12;
PICZZZZ9=PICZZZZ9+DATA(2,I);
END;
GTOTAL=GTOTAL+PICZZZZ9;
CALL ASCPUT(323,5,PICZZZZ9);

PICZZZZ9=0;
DO I=1 TO 12;
PICZZZZ9=PICZZZZ9+DATA(3,I);
END;
GTOTAL=GTOTAL+PICZZZZ9;
CALL ASCPUT(423,5,PICZZZZ9);

/*****
/* DISPLAY GRAND TOTAL */
*****/
PICZZZZ9=GTOTAL;
CALL ASCPUT(523,5,PICZZZZ9);
END DISPLAY_TOTALS;

/* INITIALIZE YEAR TOTAL */
/* LOOP THROUGH 12 MONTHS */
/* BUMP YEAR TOTAL */
/* END I-LOOP */
/* INITIALIZE GRAND TOTAL */
/* DISPLAY YEAR TOTAL */
/* FOR SALES REGION 1 */

/* REPEAT PROCESS FOR */
/* SALES REGION 2 */

/* BUMP THE GRAND TOTAL */

/* REPEAT PROCESS FOR */
/* SALES REGION 3 */

/* BUMP THE GRAND TOTAL */

/* CONVERT TO CHARACTER */
/* DISPLAY GRAND TOTAL */

```

```

%INCLUDE(ADMUPINA);          /* INCLUDE GDDM AND PGF ENTRIES */
%INCLUDE(ADMUPINC);
%INCLUDE(ADMUPINF);
DECLARE (
  GSCOL  ENTRY(  BIN FIXED(31)                                ),
                                     /* SET COLOR */
  GSFLD  ENTRY(  BIN FIXED(31),BIN FIXED(31),BIN FIXED(31),
                                     BIN FIXED(31)            ),
                                     /* GRAPHICS FIELD */
  GSLINE ENTRY(  DEC FLOAT(6),DEC FLOAT(6)                    ),
                                     /* LINE TO */
  GSMOVE ENTRY(  DEC FLOAT(6),DEC FLOAT(6)                    ),
                                     /* MOVE TO */
  GSQPS  ENTRY(  DEC FLOAT(6),DEC FLOAT(6)                    ),
                                     /* QUERY PICTURE SPACE */
  GSSCLS ENTRY(                                     ),
                                     /* CLOSE SEGMENT */
  GSSEG  ENTRY(  BIN FIXED(31)                                ),
                                     /* CREATE SEGMENT */
  GSWIN  ENTRY(  DEC FLOAT(6),DEC FLOAT(6),DEC FLOAT(6),
                                     DEC FLOAT(6)            ),
                                     /* SPECIFY WINDOW */
                                     )OPTIONS(ASM,INTER);

END SALES;

```

Notes:

1. **Sales data usually on disk.** For the purpose of this example, the sales data is declared in-line. A real-life program would doubtless have its sales figures stored in a data base. It would retrieve the figures for the requested months and sales areas, and insert them into the alphanumeric grid.
2. **Uppercase translation made the default.** Several fields on page 1 accept alphanumeric input. To ensure that the input into these fields is always translated to uppercase, such translation is made the default (by calling ASDFLT).
3. **Window chosen to match row/column coordinates.** The yellow lines on the sales figures grid have to run between the alphanumeric fields in the grid. To ease the positioning of the lines, the window has been chosen to match the row/column coordinates.
4. **Using character color attributes to aid readability.** These green and turquoise character color attributes aid the readability of the PF-key information. See Figure 32 on page 121.
5. **Using CHAREA call to position the chart.** The CHAREA call positions the business graphics on the right half of the page.
6. **Specifying a black chart background.** A special trick has been used here. To remove the previous business graphics a "black chart background" is requested for each chart.
7. **Setting the character grid to match the chart-area.** To ensure that the graphics text on the chart appears correctly, the character grid has been set to match the rows and columns of the chart area.

8. **Mechanism for displaying help information.** When help is requested, page 2 is selected and displayed with an ASREAD. When control returns to the program, the main display page must be reselected.
9. **Checking and correcting input data.** The input “start-month” has a numeric significance to the program. If some faulty input (such as ‘6R’) is typed in, a conversion error occurs. This PL/I conversion on-unit traps the error and sets the faulty field to ‘01’. The corrected data is then reassigned with an ASCPUT.
10. **Redisplaying fields that may have been corrected.** Field 90 is redisplayed immediately after it has been read in, because uppercase translation may have occurred. The name then appears in uppercase on the screen.
11. **Prompting operator to correct faulty data.** Faulty sales data is set to 0000 and displayed in red. The alarm is also sounded to alert the operator that the field must be corrected. See Figure 32 on page 121.
12. **Dynamic setting of legend text.** This is an example of a dynamic CHKEY call. The number of components and the key text itself vary from call to call.
13. **X range different for histogram.** If the month range for a histogram is, say, 1 to 6, then the range must be set as 0 to 6. The first month’s bar spans $x=0$ to $x=1$, and so on.
14. **X range different for bar chart.** The x range for a bar chart must be extended by 0.5 in each direction to allow for the width of the bars.
15. **Varying format of month labels.** If there are less than 5 components, there is room to display the month labels in 3-letter format. Otherwise, they are displayed in 1-letter format to avoid a “not enough room for labels” diagnostic.
16. **Short-cut in specifying histogram x data.** The histogram bars are always 1 wide. Instead of passing two arrays of x data (as usual), the first array is set as 1 less than the second array.

Appendix B. Example 2. COBOL graphical query example using mapping

The program displays charts and the data on which they are based. The terminal operator can scroll through the data to find a particular item, or clear the chart altogether so that the whole of the screen can be used for the data. Figure 36 is a main display.

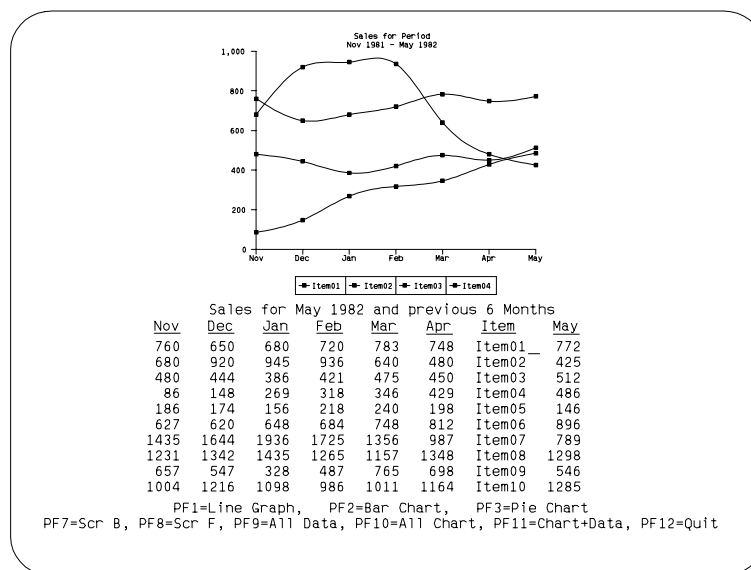


Figure 36. Main display for the graphical query program

In Figure 37, the screen shows the data only, thus enlarging the area that can be used for data and making its examination more practical.

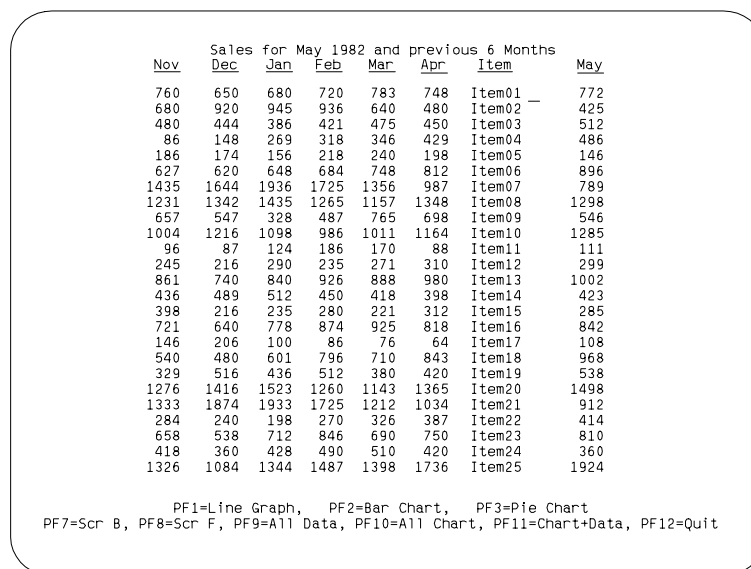


Figure 37. Data-only display for the graphical query program

The screen in Figure 38 shows only the chart and complements the data-only screen. It enables the chart to be shown at the largest possible size.

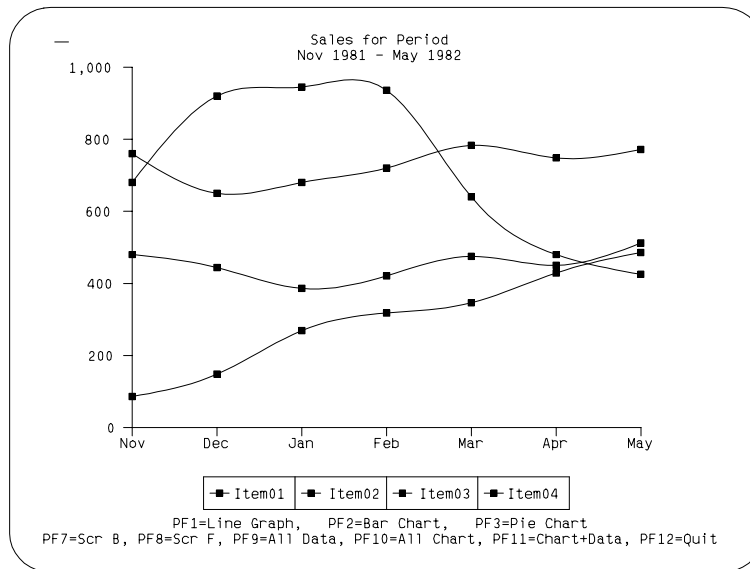


Figure 38. Chart-only display for the graphical query program

IDENTIFICATION DIVISION.
PROGRAM-ID. GIMEXMPL.
REMARKS.

```
*****
*
*          COBOL Version of GDDM Mapping
*
*****
```

ENVIRONMENT DIVISION.
CONFIGURATION SECTION.
SOURCE-COMPUTER.
OBJECT-COMPUTER.

*

DATA DIVISION.

*

WORKING-STORAGE SECTION.

*

* Parameters to the screen read operation - ASREAD.

*

77 AT-TYPE PIC S9(8) COMP.

77 AT-VAL PIC S9(8) COMP.

77 FLD-CNT PIC S9(8) COMP.

*

* Parameters used in Mapping Calls :

*

* MSPCRT - Page Id, Default Row/Col values and Mapgroups.

*

```

77 PAGE-ID1          PIC S9(8) COMP VALUE IS +1.
77 PAGE-ID2          PIC S9(8) COMP VALUE IS +2.
77 PAGE-ID3          PIC S9(8) COMP VALUE IS +3.
77 DEF-ROW           PIC S9(8) COMP VALUE IS -1.
77 DEF-COL           PIC S9(8) COMP VALUE IS -1.
77 CANDD             PIC X(8)          VALUE IS 'CANDDD6 '.
77 ALLD              PIC X(8)          VALUE IS 'ALLDD6 '.
77 ALLC              PIC X(8)          VALUE IS 'ALLCD6 '.
*
*      MSDFLD - Field Id, Row/Col as MSPCRT and Map names.
*
77 FIELD-ID1         PIC S9(8) COMP VALUE IS +1.
77 FIELD-ID2         PIC S9(8) COMP VALUE IS +2.
77 FIELD-ID3         PIC S9(8) COMP VALUE IS +3.
77 FIELD-ID4         PIC S9(8) COMP VALUE IS +4.
77 AREA1             PIC X(8)          VALUE IS 'AREA1  '.
77 AREA2             PIC X(8)          VALUE IS 'AREA2  '.
77 HEADER            PIC X(8)          VALUE IS 'HEADER  '.
77 FLOATER           PIC X(8)          VALUE IS 'FLOATER '.
77 TRAILER           PIC X(8)          VALUE IS 'TRAILER '.
*
*      MSPUT  - Field Id as MSDFLD, write operations, length of
*               ADS, and ADS itself (IMD-generated structure
*               is copied from the specified MACLIB).
*
77 WR-OP             PIC S9(8) COMP VALUE IS +0.
77 RW-OP             PIC S9(8) COMP VALUE IS +1.
77 CO-OP             PIC S9(8) COMP VALUE IS +4.
01 AREA1-ADS.
  COPY AREA1.
01 AREA2-ADS.
  COPY AREA2.
01 HEADER-ADS.
  COPY HEADER.
01 FLOATER-ADS.
  COPY FLOATER.
01 TRAILER-ADS.
  COPY TRAILER.
*
*      MSGET  - Field Id as MSDFLD, get data type, and ADS
*               length and ADS itself as for MSPUT.
*

```

example programs

```

77 GET-D          PIC S9(8) COMP VALUE IS +0.
*
*      MSQMOD - Count (uses ASREAD FLD-CNT), field id and
*              ADS length.
*
77 FIELD-ID       PIC S9(8) COMP.
77 ADS-LENGTH     PIC S9(8) COMP.
*
*      GSLSS parameters - to make the ISS 'ADMITALC' available.
*
77 GSLSS-TYPE     PIC S9(8) COMP VALUE IS +1.
77 GSLSS-SSNAME   PIC X(8)      VALUE IS 'ADMITALC'.
77 GSLSS-SSID     PIC S9(8) COMP VALUE IS +199.
*
*      General variables.
*
77 CURRENT-PAGE   PIC S9(8) COMP.
77 LOOP-CTL       PIC S9(8) COMP.
77 FIRST-ON-PAGE  PIC S9(8) COMP VALUE IS +1.
77 LAST-ON-PAGE   PIC S9(8) COMP.
77 SCROLL-VALUE   PIC S9(8) COMP.
77 PAGE1-DEFINED  PIC S9(8) COMP VALUE IS +0.
77 PAGE2-DEFINED  PIC S9(8) COMP VALUE IS +0.
77 PAGE-DEFINED   PIC S9(8) COMP.
77 HI-VALUE       PIC S9(8) COMP VALUE IS +29.
77 CHART-TYPE     PIC S9(8) COMP VALUE IS +1.
77 SELECTED-ITEM  PIC S9(8) COMP VALUE IS +1.
77 NEW-SELECTION  PIC S9(8) COMP.
*
*      PF Key Line Information.
*
77 PF-LINE1       PIC X(46) VALUE IS
                  'PF1=Line Graph, PF2=Bar Chart, PF3=Pie Chart'.
77 PF-MASK1       PIC X(46) VALUE IS
                  '      7777 77777      777 77777      777 77777'.
77 PF-LINE2       PIC X(78) VALUE IS
                  'PF7=Scr B, PF8=Scr F, PF9=All Data, PF10=All Chart, PF1
-                  '1=Chart+Data, PF12=Quit'.
77 PF-MASK2       PIC X(78) VALUE IS
                  '      777 7      777 7      777 7777      777 77777
-                  ' 7777777777      7777'.
77 PF-LINE3       PIC X(78) VALUE IS
                  '      PF9=All Data, PF10=All Chart, PF11=Chart+Dat
-                  'a, PF12=Quit'.
77 PF-MASK3       PIC X(78) VALUE IS
                  '      777 7777      777 77777      7777777777
-                  '7      7777      '.
*

```

```

PROCEDURE DIVISION.
*
*   Initialize GDDM.
*
*   CALL 'FSINIT'.
*
*   Load up the ISS 'ADMITALC'.
*
*   CALL 'GSLSS' USING GSLSS-TYPE, GSLSS-SSNAME, GSLSS-SSID.
*
*   Define ALL 3 mapped pages.
*
*   Create the first mapped page (Id=1), using mapgroup-defined
*   row and column positions for mapgroup 'CANDDD6'.
*
*   CALL 'MSPCRT' USING PAGE-ID1, DEF-ROW, DEF-COL, CANDD.
*
*   Define the fixed mapped fields for page 1, that is AREA1,
*   HEADER and TRAILER at their IMD-defined positions.
*
*   CALL 'MSDFLD' USING FIELD-ID1, DEF-ROW, DEF-COL, AREA1.
*   CALL 'MSDFLD' USING FIELD-ID2, DEF-ROW, DEF-COL, HEADER.
*   CALL 'MSDFLD' USING FIELD-ID3, DEF-ROW, DEF-COL, TRAILER.
*
*   Output the trailer record, using character highlighting
*   to distinguish the instructions.
*
*   MOVE '1' TO F1-SEL.
*   MOVE '1' TO F2-SEL.
*   MOVE PF-LINE1 TO F1.
*   MOVE PF-LINE2 TO F2.
*   CALL 'MSPUT' USING FIELD-ID3, WR-OP, TRAILER-ASLENGTH,
*                   TRAILER-ADS.
*   MOVE PF-MASK1 TO F1.
*   MOVE PF-MASK2 TO F2.
*   CALL 'MSPUT' USING FIELD-ID3, CO-OP, TRAILER-ASLENGTH,
*                   TRAILER-ADS.
*
*   Create the second mapped page (Id=2), using mapgroup-defined
*   row and column positions for mapgroup 'ALLD'.
*
*   CALL 'MSPCRT' USING PAGE-ID2, DEF-ROW, DEF-COL, ALLD.
*
*   Define the mapped fields for page 2, that is HEADER,
*   and TRAILER at their IMD-defined positions.
*
*   CALL 'MSDFLD' USING FIELD-ID2, DEF-ROW, DEF-COL, HEADER.
*   CALL 'MSDFLD' USING FIELD-ID3, DEF-ROW, DEF-COL, TRAILER.
*
*   Output the trailer record, using character highlighting
*   to distinguish the instructions.
*

```

```

MOVE PF-LINE1 TO F1.
MOVE PF-LINE2 TO F2.
CALL 'MSPUT' USING FIELD-ID3, WR-OP, TRAILER-ASLENGTH,
                TRAILER-ADS.

MOVE PF-MASK1 TO F1.
MOVE PF-MASK2 TO F2.
CALL 'MSPUT' USING FIELD-ID3, CO-OP, TRAILER-ASLENGTH,
                TRAILER-ADS.

*
* Create the third mapped page (Id=3), using mapgroup-defined
* row and column positions for mapgroup 'ALLCD6'.
*
CALL 'MSPCRT' USING PAGE-ID3, DEF-ROW, DEF-COL, ALLC.

*
* Define the mapped fields for page 3, that is AREA2
* and TRAILER at their IMD-defined positions.
*
CALL 'MSDFLD' USING FIELD-ID1, DEF-ROW, DEF-COL, AREA2.
CALL 'MSDFLD' USING FIELD-ID3, DEF-ROW, DEF-COL, TRAILER.

*
* Output the trailer record, using character highlighting
* to distinguish the instructions.
*

MOVE PF-LINE1 TO F1.
MOVE PF-LINE3 TO F2.
CALL 'MSPUT' USING FIELD-ID3, WR-OP, TRAILER-ASLENGTH,
                TRAILER-ADS.

MOVE PF-MASK1 TO F1.
MOVE PF-MASK3 TO F2.
CALL 'MSPUT' USING FIELD-ID3, CO-OP, TRAILER-ASLENGTH,
                TRAILER-ADS.

*
* Output page1 - Chart + Data mixture.
* The mapped fields are defined, so just invoke the chart-
* producing procedure 'PUTCHRT2' and fill in the floating
* area, before putting up the mapped page.
*

CHART-AND-DATA.
    CALL 'FSPSEL' USING PAGE-ID1.
    MOVE PAGE-ID1 TO CURRENT-PAGE.
CHART1.
    CALL 'PUTCHRT2' USING CURRENT-PAGE, CHART-TYPE,
                        SELECTED-ITEM.

FLOAT1.
    MOVE 10 TO SCROLL-VALUE.
    MOVE PAGE1-DEFINED TO PAGE-DEFINED.
    PERFORM PUT-FLOATER.
    MOVE 1 TO PAGE1-DEFINED.

READ1.
    GO TO GET-RESPONSE.
ALL-DATA.

*
* Output page2 - All Data.
* The mapped fields are defined, so just fill in the
* floating area, before putting up the mapped page.
*

```

```

        CALL 'FSPSEL' USING PAGE-ID2.
        MOVE PAGE-ID2 TO CURRENT-PAGE.
    FLOAT2.
        MOVE 25 TO SCROLL-VALUE.
        MOVE PAGE2-DEFINED TO PAGE-DEFINED.
        PERFORM PUT-FLOATER.
        MOVE 1 TO PAGE2-DEFINED.
    READ2.
        GO TO GET-RESPONSE.
    ALL-CHART.
*
*   Output page3 - All Chart.
*   The mapped fields are defined, so just invoke the chart-
*   drawing procedure 'PUTCHRT2', before putting up the mapped page.
*
        CALL 'FSPSEL' USING PAGE-ID3.
        MOVE PAGE-ID3 TO CURRENT-PAGE.
    CHART2.
        CALL 'PUTCHRT2' USING CURRENT-PAGE, CHART-TYPE,
                                SELECTED-ITEM.
    READ3.
        GO TO GET-RESPONSE.
*
*   Output the current page and wait for operator input.
*
    GET-RESPONSE.
        MOVE 0 TO NEW-SELECTION.
        CALL 'ASREAD' USING AT-TYPE, AT-VAL, FLD-CNT.
*
*   Analyze the response. First check if the operator has
*   selected an item.
*
        IF FLD-CNT IS GREATER THAN 0 THEN GO TO TEST0.
*
*   Analyze the PF key pressed.
*
        IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 7
            THEN SUBTRACT SCROLL-VALUE FROM FIRST-ON-PAGE
                IF FIRST-ON-PAGE IS LESS THAN 1
                    THEN MOVE 1 TO FIRST-ON-PAGE.
        IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 8
            THEN ADD SCROLL-VALUE TO FIRST-ON-PAGE
                IF FIRST-ON-PAGE IS GREATER THAN HI-VALUE
                    THEN MOVE HI-VALUE TO FIRST-ON-PAGE.
        IF AT-TYPE IS EQUAL TO 1
            AND AT-VAL IS EQUAL TO 7 OR AT-VAL IS EQUAL TO 8
            THEN GO TO TEST1.

```

```

CHECK-PF.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 12
    THEN GO TO GET-OUT.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 1
    THEN MOVE 1 TO CHART-TYPE.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 2
    THEN MOVE 2 TO CHART-TYPE.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 3
    THEN MOVE 3 TO CHART-TYPE.
  IF AT-TYPE IS EQUAL TO 1 AND
    AT-VAL IS EQUAL TO 1 OR AT-VAL IS EQUAL TO 2
    OR AT-VAL IS EQUAL TO 3
    THEN GO TO TEST2.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 9
    THEN GO TO ALL-DATA.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 10
    THEN GO TO ALL-CHART.
  IF AT-TYPE IS EQUAL TO 1 AND AT-VAL IS EQUAL TO 11
    THEN GO TO CHART-AND-DATA.
  IF NEW-SELECTION IS EQUAL TO 1 THEN GO TO CHART1.
  IF NEW-SELECTION IS EQUAL TO 2 THEN GO TO FLOAT2.
  GO TO GET-RESPONSE.

*
*   Get the id of the selected item.
*   and convert that to an actual record number and update
*   the page pointers.
*
TEST0.
  MOVE 1 TO FLD-CNT.
  CALL 'MSQMOD' USING FLD-CNT, FIELD-ID, ADS-LENGTH.
  SUBTRACT 101 FROM FIELD-ID GIVING SELECTED-ITEM.
  ADD FIRST-ON-PAGE TO SELECTED-ITEM.
  MOVE SELECTED-ITEM TO FIRST-ON-PAGE.
  IF CURRENT-PAGE IS EQUAL TO 1
    THEN MOVE 1 TO NEW-SELECTION.
  IF CURRENT-PAGE IS EQUAL TO 2
    THEN MOVE 2 TO NEW-SELECTION.
  GO TO CHECK-PF.

*
*   For scrolling, determine where to return to.
*
TEST1.
  IF CURRENT-PAGE IS EQUAL TO 1
    THEN GO TO FLOAT1.
  IF CURRENT-PAGE IS EQUAL TO 2
    THEN GO TO FLOAT2.
  IF CURRENT-PAGE IS EQUAL TO 3
    THEN GO TO GET-RESPONSE.

*
*   For chart type, determine where to return to.
*

```

```

TEST2.
  IF CURRENT-PAGE IS EQUAL TO 1
    THEN GO TO CHART1.
  IF CURRENT-PAGE IS EQUAL TO 2
    THEN GO TO GET-RESPONSE.
  IF CURRENT-PAGE IS EQUAL TO 3
    THEN GO TO CHART2.
*
*   Perform the output of the floating maps.
*
PUT-FLOATER.
  ADD SCROLL-VALUE FIRST-ON-PAGE GIVING LAST-ON-PAGE.
  MOVE +101 TO FIELD-ID4.
  PERFORM MOVE-DATA VARYING LOOP-CTL
    FROM FIRST-ON-PAGE BY 1
    UNTIL LOOP-CTL IS EQUAL TO LAST-ON-PAGE.
*
*   Move the data from the table to the floating map and put it
*   to the page.
*   If it is the first time the floating maps are being output,
*   define the field first.
*
MOVE-DATA.
  CALL 'FILLIT1' USING FLOATER-ADS, LOOP-CTL.
  IF FIELD-ID4 IS EQUAL TO 101
    THEN
      MOVE '1' TO CHOSEN-CURSOR.
  IF PAGE-DEFINED IS EQUAL TO 0
    THEN
      CALL 'MSDFLD' USING FIELD-ID4, DEF-ROW, DEF-COL, FLOATER
      CALL 'MSPUT'  USING FIELD-ID4, WR-OP, FLOATER-ASLENGTH,
        FLOATER-ADS
    ELSE
      CALL 'MSPUT'  USING FIELD-ID4, RW-OP, FLOATER-ASLENGTH,
        FLOATER-ADS.
  ADD +1 TO FIELD-ID4.
*
*   Terminate GDDM.
*
GET-OUT.
  CALL 'FSTERM'.
  STOP RUN.

```


GDDM glossary

This glossary defines technical terms used in GDDM documentation. In the definitions, the qualification "In GDDM" means in the GDDM Base products or in GDDM-PGF (or both). "In GDDM-PGF" means in GDDM-PGF, which includes the ICU.

If you do not find the term you are looking for, refer to the index of the appropriate GDDM manual or view the *IBM Dictionary of Computing*, located on the Internet at:

<http://www.networking.ibm.com/nsg/nsgmain.htm>

A

AAB. Application anchor block. A control block that identifies an instance of GDDM. It must be used for all calls that use the GDDM reentrant interface.

absolute data. In GDDM-PGF, the actual y values to be plotted. Contrast with **relative data**.

alphanumeric character attributes. In GDDM, comprise the highlighting, color, and symbol set to be used.

alphanumeric cursor. A physical indicator on a display. It may be moved from one hardware cell to another.

alphanumeric field. A field (area of a screen or printer page) that can contain alphabetic, numeric, or special characters. In GDDM, contrast with **graphics field**.

alphanumeric field attributes. In GDDM, comprise intensity, highlighting, color, symbol set to be used, field type, field end output conversion, input conversion, translate table assignment, transparency, field outlining, and mixed-string fields.

alphanumeric label. In GDDM-PGF or the ICU, a user-specified alphanumeric string used to annotate an x-axis or y-axis scale mark. Contrast with **numeric label**.

alternate device. In GDDM, a device to which copies are sent of the primary device's output. Usually the alternate device is a printer or plotter. See also **primary device**.

annotation. An added descriptive comment or explanatory note.

API. Application programming interface. The formally defined programming-language interface between an IBM system control program or program product and its user.

APL. One of the programming languages supported by GDDM.

aspect ratio. The width-to-height ratio of an area, symbol, or shape.

attribute table. In GDDM-PGF, a set of values for one particular attribute (for example, shading pattern), that are used in sequence to display the components of a business chart.

attributes. Characteristics or properties that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also **alphanumeric character attributes**, **alphanumeric field attributes**, and **graphics attributes**.

autoranging. In GDDM-PGF, the process in which the axis ranges are determined by the extremes of the data values passed by the application. Synonymous with **autoscaling**.

autoscaling. Synonym for **autoranging**.

axis. In a chart, a line that is drawn to indicate units of measurement against which items in the chart can be viewed. GDDM-PGF charts have an x, y, and (in the case of tower charts) z axis.

axis label. In GDDM-PGF, text appearing at or between axis major scale marks on a business chart. Such labels may be numeric or alphanumeric. Contrast with **data label**.

axis title. In GDDM-PGF, a text string describing what an axis represents.

B

background color. Black on a display, white on a printer. The initial color of the display medium. Contrast with **neutral color**.

bar chart. A chart consisting of several bars of equal width. The value of the dependent variable is indicated by the height of each bar. Synonymous with **column chart**.

BASIC. One of the programming languages supported by GDDM.

blank character. An empty character represented by X'40' in the EBCDIC code. In GDDM-PGF, such a character occupies one position in a label or a key and

may be used for positioning purposes. Contrast with **null character**.

business graphics. The methods and techniques for presenting commercial and administrative information in chart form. For example, the creation and display of a sales bar chart. Contrast with **general graphics**.

C

cell. See **character cell**.

character. A letter, digit, or other symbol.

character attributes. See **alphanumeric character attributes**. See also **graphics text attributes**.

character box. In GDDM, the rectangle or (for sheared characters) the parallelogram boundaries that govern the size, orientation, spacing, and italicizing of individual symbols or characters to be shown on a display screen or printer page.

The box width, height, and if required, shear, are specified in world coordinates and may be program-controlled. See also **character mode**. Contrast with **character cell**.

character cell. The physical, rectangular space in which any single character or symbol is displayed on a screen or printer device. The size and position of a character cell are fixed. Size is usually specified in pixels on a given device, for example, 9 by 12 on an IBM 3279 Model 3 display. Position is addressed by row and column coordinates. Synonymous with **hardware cell** and **symbol cell**. Contrast with **character box**.

character code. The means of addressing a symbol in a symbol set, sometimes called **code point**.

The particular form and range of codes depends on the GDDM context, for example:

- For the Image Symbol Editor, a hexadecimal constant in the range X'41'–X'FE', or its EBCDIC character equivalent.
- For the Vector Symbol Editor, a hexadecimal constant in the range X'00'–X'FF' or its EBCDIC character equivalent.
- For the GDDM API, a decimal constant in the range 0 through 239, or subsets of this range (for example, a marker symbol code range of 1 through 8).

character grid. A notional grid that covers the **chart area**. The size of the grid determines the basic size of the characters in all text constructed by PG routines. It is the fundamental measurement in chart layout,

governing the spacing of mode-2 characters and the size of mode-3 characters. It also governs the size of the chart margins and thus the plotting area.

character matrix. Synonym for **dot matrix**.

character mode. In GDDM, the type of characters to be used. There are three modes:

- Mode-1 characters are loadable into PS and are of device-dependent fixed size, spacing, and orientation, as are hardware characters.
- Mode-2 characters are image (ISS) characters. Size and orientation are fixed. Spacing is variable by program.
- Mode-3 characters are vector (VSS) characters. Box size, position, spacing, orientation, and shear of individual characters are variable by program.

chart. In GDDM, usually means business chart, for example, a **bar chart**.

chart annotation. Annotative text added to a business chart. In GDDM-PGF, referred to as **chart notes**.

chart area. In GDDM-PGF, the part of the picture space in which a business chart is to be drawn.

chart attributes. In GDDM-PGF, define how each part of the chart appears - for example, the font to be used for the chart heading.

chart data. An ICU chart is saved in two distinct parts, the data and the format. The chart data consists of the x and y values, the data labels, the data group names, and the chart heading.

chart data attributes. In GDDM-PGF, define the appearance of the data representation. For example, the color of the lines on a line graph or the shading patterns used for the sectors of a pie chart.

chart format. An ICU chart is saved in two distinct parts, the data and the format. The chart format consists of the chart type, the chart attributes, the axis characteristics, the chart layout, and the chart notes.

chart notes. In GDDM-PGF, additional text to annotate a business chart. May be used in isolation to create alphanumeric presentation material (using the ICU interactive notes facility).

chart type. In GDDM-PGF, specifies whether the business chart should be a line graph, surface chart, histogram, bar chart, pie chart, Venn diagram, polar chart, table chart, or tower chart.

choice device. A logical input device that enables the application program to identify keys pressed by the terminal operator.

CICS. Customer Information Control System. A subsystem of MVS or VSE under which GDDM can be used.

CMS. Conversational Monitor System. A time-sharing subsystem that runs under VM/SP.

COBOL. One of the programming languages supported by GDDM.

code point. Synonym for **character code**.

column chart. See **bar chart**.

compass keys. In the GDDM Vector and Image Symbol Editors, a set of PF keys predefined to draw a line in vector symbols or add a dot in image symbols, in directions corresponding to points of the compass.

component (data). In GDDM-PGF, synonym for **data group**. One line on a line graph, for example, or one set of bars on a bar chart.

composite bar chart. In GDDM-PGF, a bar chart in which multiple y values for the same x value or x label are stacked one on top of another. Contrast with **multiple bar chart**. See also **floating bar chart**.

cursor. A physical indicator that may be moved around a display screen. See **alphanumeric cursor** and **graphics cursor**.

curve construction line. In the GDDM Vector Symbol Editor, one of a series of vectors that is used in the construction of a curve.

curve fitting. The construction of a smooth curve through a sequence of plot points, as opposed to their connection by straight lines. In GDDM-PGF or the ICU, curve fitting may be requested for line graphs or polar charts.

D

data group. In GDDM-PGF, one set of y values corresponding to a given set of x values (for example, the data values for one line on a line graph). Synonymous with **component**.

data indexing. In ICU, the display of y values relative to other y values in the same chart, rather than as originally specified. For example, all bars might be displayed as a percentage of the set of bars at X=1993.

data label. In ICU, text specified on the data entry panel rather than on the x-axis label panel. If the chart has axes, the label is displayed on a tick mark that is close to the matching numeric x value. Data labels are attached to the sectors on a pie chart and to the circles

and overlap area of a Venn diagram. Contrast with **axis label**.

data values. In GDDM-PGF, the x and y values that are plotted on a business chart.

datum line. In GDDM-PGF, a line drawn parallel to a chart axis, through a specified value along the other axis. See also **datum reference line**.

datum reference line. In GDDM-PGF, a datum line that also acts as a shading boundary for the first component of a surface chart, histogram or composite bar chart, or for all the components of a polar chart or multiple bar chart. If no datum reference line is present, such components are shaded from the x axis.

data set. The major unit of data storage and retrieval, consisting of a collection of data in one of several prescribed arrangements and described by control information to which the system has access.

DBCS. Double-byte character set.

default value. A value chosen by GDDM when no value is explicitly specified by the user. For example, the default line type is a solid line.

device family. In GDDM, a device classification that governs the general way in which I/O is processed. See also **processing options**. For example:

- Family 1: 3270 display or printer
- Family 2: queued printer
- Family 3: system printer (alphanumerics only)
- Family 4: high-resolution printer.

device token. In GDDM, an 8-byte code giving entry to a table of pre-established device hardware characteristics that are required when the device is opened (initialized).

display device. Any output unit that gives a visual representation of data. For example, a screen or printer. More commonly, the term is used to mean a screen as opposed to a printer.

display point. Synonym for **pixel**.

display terminal. An input/output unit by which a user communicates with a data-processing system or subsystem. Usually includes a keyboard and always provides a visual presentation of data. For example, an IBM 3179 display.

double-byte character set (DBCS). A set of characters in which each character occupies two byte positions in internal storage and in display buffers. Used for oriental languages.

dual characters. In GDDM, characters that each occupy two bytes in internal storage and in display buffers. They are used to display Kanji or Hangeul symbols.

E

edit. To enter, modify, or delete data.

exploded pie chart. A pie chart in which one or more sectors have been moved outward from the center of the pie, to have a greater impact on the eye.

external defaults. GDDM-supplied values that users can change to suit their own needs.

F

field attributes. See **alphanumeric field attributes**.

floating bar chart. In GDDM-PGF, a special type of composite bar chart in which the first data group is not displayed. The stacks of bars representing the remaining data groups therefore appear to “float”.

foil. A transparency for overhead projection.

font. A particular style of typeface (for example, Gothic English). In GDDM, a font may exist as a programmed symbol set.

FORTRAN. One of the programming languages supported by GDDM.

four-button cursor. A hand-held device, with cross-hair sight, for indicating positions on the surface of a tablet. Synonymous with **puck**.

free data. In GDDM-PGF, data that has a separate set of x points for each component. Formerly known as **paired data**. Contrast with **tied data**.

G

GDDM. Graphical Data Display Manager.

GDDM storage. The portion of host computer main storage used by GDDM.

GDF. Graphics data format.

general graphics. The methods and techniques for converting data to or from graphics display in mathematical, scientific, or engineering applications; that is, any application other than business graphics. See also **business graphics**.

graphics. A picture defined in terms of graphics primitives and graphics attributes.

graphics attributes. In GDDM, comprise color selection, color mix, line type, line width, graphics text attributes, marker symbol, and shading pattern definition.

graphics cursor. A physical indicator that can be moved (often with a joystick, mouse, or stylus) to any position on the screen.

graphics data format (GDF). A picture definition in an encoded order format used internally by GDDM and, optionally, providing the user with a lower-level programming interface than the GDDM API.

graphics text attributes. In GDDM, comprise symbol (character) set to be used, character box size, character angle, character mode, character shear angle, and character direction.

grid lines. In GDDM-PGF, lines drawn parallel to one axis and through the major scale marks of the other axis.

H

hardware cell. Synonym for **character cell**.

hardware characters. Synonym for **hardware symbols**.

hardware symbols. The characters that are supplied with the device. The term is loosely used also for GDDM mode-1 symbols that are loaded into a PS store for subsequent display. Synonymous with **hardware characters**.

help panel. A panel presenting tutorial text to assist the terminal user. All the GDDM interactive utilities possess comprehensive help panels.

hidden bars. See **overlapping bar chart**.

high-resolution image file. An intermediate form, residing on disk, of a picture destined for a high-resolution printer.

high-resolution printer. A printer, such as the 4250 or 3800-3, that has a high density of pixels to the inch and therefore produces output of good quality.

histogram. A chart in which each value of the dependent variable corresponds to a range of values of the independent variable (represented by the width of the associated bar). Such a chart might display the number of persons in various age ranges, for example.

home panel. The first panel that is displayed by the ICU. It is the starting point for access to the other panels.

I

ICU. Interactive Chart Utility.

image. Pictorial information that is specified in terms of the dots (pixels) of which it is made up.

image symbol. A character or symbol defined as a dot pattern.

Image Symbol Editor (ISE). A GDDM-supplied interactive editor that lets users create or modify their own image symbol sets (ISS).

image symbol set (ISS). A set of symbols each of which was created as a pattern of dots. Contrast with **vector symbol set (VSS)**.

IMS/VS. Information Management System/Virtual Storage. A subsystem of MVS under which GDDM can be used.

include member. A collection of source statements stored as a library member for later inclusion in a compilation.

indexing. In ICU, see **data indexing**.

integer. A whole number (for example, -2, 3, 457).

Interactive Chart Utility (ICU). A GDDM-PGF menu-driven program that allows business charts to be created interactively by nonprogrammers.

interactive graphics. In GDDM, those graphics that can be moved or manipulated by a user at a terminal.

interactive mode. A mode of application operation in which each entry receives a response from a system or program, as in an inquiry system or an airline reservation system. An interactive system may also be conversational, implying a continuous dialog between the user and the system.

interactive subsystem. (1) One or more terminals, printers, and any associated local controllers capable of operation in interactive mode. (2) One or more system programs or licensed programs that enable user applications to operate in interactive mode. For example, CICS.

intercept. In a chart, a method of describing the position of one axis relative to another. For example, the x axis can be specified so that it intercepts (crosses) the y axis at the bottom, middle, or top of the plotting area of a chart.

ISE. Image Symbol Editor.

ISS. Image symbol set.

J

JCL. Job Control Language. A problem-oriented language designed to express statements in a job that identify the job or describe its requirements to the operating system.

joystick. A lever that can pivot in all directions, used as a locator device.

K

Kanji. A character set of symbols used in Japanese ideographic alphabets.

key. In a legend, a symbol and an associated data group name. A key might, for example, indicate that the pink line on a graph represents "Predicted Profit." See also **legend**.

key symbol. A small part of a line (from a line graph) or an area (from a shaded chart) used in a legend to identify the various data groups.

L

legend. A set of symbolic keys used to identify the data groups in a business chart.

line attributes. In GDDM, color, line type, and line width.

line graph. In GDDM-PGF, a chart in which the plotted points (each optionally represented by a marker) are joined by straight or curved lines. If only the markers are displayed, the chart is known as a scatter plot.

link edit. To create a loadable computer program by means of a linkage editor.

load module. A program unit that is suitable for loading into main storage for execution; it is usually the output of a linkage editor.

logarithmic axis. In GDDM-PGF, an axis on which ascending powers of 10 are equally spaced.

M

Manhattan chart. Synonym for **tower chart**.

menu. A displayed list of logically grouped functions from which the operator may make a selection. Sometimes called a menu panel.

menu-driven. Describes a program that is driven by an operator responding to one or more displayed menus.

missing values. In GDDM-PGF or the ICU, x or y values that are omitted from a chart. For example, one line on a graph might represent a sales forecast and extend to the end of the year on the x axis, while a second line might represent actual sales and extend only to the current month.

mixed character string. A string containing a mixture of Latin (one-byte) and Kanji (two-byte) characters.

mixed chart. In GDDM-PGF or the ICU, the combination of more than one chart type in a business graph. For example, the overlaying of a line graph on top of a bar chart.

mode 1/2/3 characters. See **character mode**.

mountain shading. A method of shading surface charts where each component is shaded separately from the base line, instead of being shaded from the data line of the previous component.

mouse. A hand-held device (the IBM 5277 Mouse) that is moved around a locator pad to position the graphics cursor on the screen.

multicomponent chart. In GDDM-PGF, a chart presenting more than one data group.

multiple axis chart. In GDDM-PGF, a chart in which more than one x axis or y axis, or both, is used. See also **secondary axis**.

multiple bar chart. In GDDM-PGF, a form of bar chart in which the bars at a given x value or label are placed side by side. Contrast with **composite bar chart**.

multiple charts. Two or more charts appearing together on the display screen or page. Multiple charts can be of the same type or different types, and can be derived from one or more sets of data.

N

National Language (NL) feature. The translations of the ICU panels and GDDM messages into a variety of languages other than English.

neutral color. White on a display, black on a printer. Contrast with **background color**.

nickname. In GDDM, a quick and easy means of referring to a device, the characteristics and identity of which have been predefined.

non-paired data. See **tied data**.

null character. An empty character represented by X'00' in the EBCDIC code. In GDDM-PGF, such a character does not occupy a screen position. The trailing positions of short keys or labels may be filled with nulls. Contrast with **blank character**.

numeric label. In GDDM-PGF, an axis major scale mark label derived directly from the data value at that scale mark. Contrast with **alphanumeric label**.

O

overlapping bar chart. A form of business chart where adjacent bars partly overlap each other. Overlapping bars are sometimes called **hidden bars**.

P

page. In GDDM, the main unit of output and input. All specified alphanumerics and graphics are added to the "current page." An output statement always sends the current page to the device, and an input statement always receives the current page from the device.

paired data. See **free data**.

panel. A predefined display that defines the locations and characteristics of alphanumeric fields on a display terminal. When the panel offers the operator a selection of alternatives it may be called a menu panel. Synonymous with **frame**.

PDS. In OS/TSO, a partitioned data set.

pel. Synonym for **pixel**.

PGF. Presentation Graphics Facility.

picture element. Synonym for **pixel**.

picture interchange format (PIF) file. In graphics systems, the type of file, containing picture data, that can be transferred between GDDM and a 3270-PC/G or 3270-PC/GX work station.

pie chart. A chart that takes the form of one or more circles divided into sectors, the angles of which represent the contributions of each data value to the group total.

PIF. Picture interchange format (PIF) file.

pixel. The smallest area of a display screen capable of being addressed and switched between visible and invisible states. Synonymous with **display point**, **pel**, and **picture element**.

PL/I. One of the programming languages supported by GDDM.

plotter. An output device that uses pens to draw its output on paper or transparency foils.

polar chart. A form of business chart where the x axis is circular and the y axis is radial.

presentation graphics. Computer graphics products or systems, the functions of which are primarily concerned with graphics output presentation. For example, the display of business planning bar charts.

Presentation Graphics Facility (PGF). A member of the GDDM family of program products. It is concerned with business graphics, as opposed to general graphics.

preview chart. A small version of the current chart that may be displayed on ICU menu panels.

primary device. In GDDM, the main destination device for the application program's output, usually a display terminal. The default primary device is the user console. See also **alternate device**.

processing options. Describe how a device's I/O is processed. These device-family-dependent and subsystem-dependent options are specified when the device is opened. An example is the choice between CMS attention-handling protocols.

program library. (1) A collection of available computer programs and routines. (2) An organized collection of computer programs. (3) Synonym for **partitioned data set**.

programmed symbols (PS). Dot patterns loaded by GDDM into the PS stores of an output device.

PS. Programmed symbols.

PS overflow. A condition where the graphics cannot be displayed in its entirety because the picture is too complex to be contained in the device's PS stores.

puck. Synonym for **four-button cursor**.

Q

queued printer. A printer belonging to the subsystem under which GDDM runs, to which output is sent indirectly by means of the GDDM Print Utility program. In some subsystems, this may allow the printer to be shared between multiple users. Contrast with **system printer**.

R

RCP. Request control parameter.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

reference line. See **datum reference line**.

regression line. In ICU, the conversion of a set of y values into other values that form a straight line most closely resembling the original values.

relative data. In GDDM-PGF, real y-data values that are to be presented in a stacked chart-type. The actual points to be plotted for a particular component are obtained by adding the y data of that component to the y data of the previous components. Contrast with **absolute data**.

reverse video. A form of alphanumeric highlighting for a character, field, or cursor, in which its color is exchanged with that of its background. For example, changing a red character on a black background to a black character on a red background.

S

scalable markers. In GDDM-PGF, (vector) markers on a line graph or polar chart that may be varied in size.

scale marks. In GDDM-PGF, markings spaced at equal intervals along an axis. Each pair of "major scale marks" may have one or more "minor scale marks" in between. Synonymous with **tick marks**.

scatter plot. In GDDM-PGF, a variety of line graph in which only the marked points, and not their joining lines, are drawn.

scrolling. In computer graphics, moving a display image vertically or horizontally in a manner such that new data appears at one edge as existing data disappears at the opposite edge.

secondary axis. In GDDM-PGF, an x- or y-axis line drawn parallel to the primary axis and capable of having scale marks, labels, and title different from those of the primary axis. Permits the combination of two business charts.

skyscraper chart. Synonym for **tower chart**.

spider labels. In GDDM-PGF, labels that annotate pie-chart sectors. Each label is joined to its associated sector by a line, giving the resulting chart a spider-like appearance.

stacked chart type. A surface chart, composite bar chart, or histogram where the data components are stacked one on top of another. The data value of a particular component is indicated by the depth of the band at that point. See also **relative data**.

stand-alone (mode). Operation that is independent of another device, program, or system.

state-1. In GDDM-PGF, the state of a business graphics program, before the first plot has been made.

state-2. In GDDM-PGF, the state of a business graphics program after the first plot has been made, thereby constructing the axes.

stylus. A pen-like pointer for indicating positions on the surface of a tablet.

surface chart. A chart similar to a line graph, except that no markers appear and the areas between successive lines are shaded.

symbol. Synonymous with **character**. For example, the following terms all have the same meaning: vector symbols, vector characters, vector text.

symbol cell. Synonym for **character cell**.

symbol matrix. Synonym for **dot matrix**.

symbol set. A collection of symbols, usually but not necessarily forming a font. GDDM applications may use the hardware device's own symbol set. Alternatively, they can use image or vector symbol sets, which the user may have created.

symbol set identifier. In GDDM, an integer (or the equivalent EBCDIC character) by which the programmer refers to a loaded symbol set.

system printer. A printer belonging to the subsystem under which GDDM runs, to which output is sent indirectly by means of system spooling facilities. Contrast with **queued printer**.

T

table chart. In GDDM-PGF, a chart in which the data is presented as numbers arranged in rows and columns.

tablet. (1) A locator device with a flat surface and a mechanism that converts indicated positions on the surface into coordinate data. (2) The IBM 5083 Tablet Model 2, which, with a four-button cursor or stylus, allows positions on the screen to be addressed and the graphics cursor to be moved without use of the keyboard.

tag. In interactive graphics, an identifier associated with one or more primitives that is returned to the program if such primitives are subsequently picked.

terminal. A device, usually equipped with a keyboard and a display unit, capable of sending and receiving information over a link. See also **display terminal**.

text. Characters or symbols sent to the device. GDDM provides alphanumeric text and graphics text.

text attributes. See **graphics text attributes**.

tick marks. In GDDM-PGF, synonym for **scale marks**.

tied data. In GDDM-PGF, data that shares the same set of x points for each component. This is the most common form of data. It was formerly known as **non-paired data**. Contrast with **free data**.

tilted pie chart. A pie chart drawn in three dimensions, which has been tilted away from full face to reveal its three-dimensional properties.

tower chart. A form of business chart in which rows of towers stand on a two-dimensional base. Synonymous with **Manhattan chart** and **skyscraper chart**.

transparency. (1) A document on transparent material suitable for overhead projection. (2) An alphanumeric attribute that allows underlying graphics to show.

TSO. Time sharing option. A subsystem of OS/VS under which GDDM can be used.

U

user exit. A point in GDDM execution where a user routine gains control if such has been requested.

V

variable cell size. In most devices, the hardware cell size is fixed. But the 3290 Information Panel has a cell size that can be varied. This in turn causes the number of rows or columns on the device to alter.

vector symbol. A character or shape made up of a series of lines or curves.

Vector Symbol Editor. A program supplied with GDDM-PGF, the function of which is to create and edit vector symbol sets (VSS).

vector symbol set (VSS). A set of symbols each of which was originally created as a series of lines and curves.

Venn diagram. A form of business chart in which two populations and their intersection are represented by two overlapping circles.

VM/SP CMS. IBM Virtual Machine/System Product Conversational Monitor System. A system under which GDDM can be used.

VSE. Virtual storage extended. An operating system consisting of VSE/Advanced Functions and other IBM programs. In GDDM, the abbreviation VSE has sometimes been used to refer to the Vector Symbol Editor, but to avoid confusion, this usage is deprecated.

VSS. Vector symbol set.

Index

Numerics

4250 printer
 line widths
 on charts 57
 64-color shading (PG routines) 81

A

abbreviated labels 47
 abbreviations of PG routines options 37
 ABPIE option 96
 ABREV option 47
 absolute pie chart data 96
 ADMCDATA file contents 6
 ADMCFORM file contents 6
 ADMUPINx
 for PG routines 24
 alphanumeric labels (PG routines) 46
 alphanumerics, mapped
 and graphics
 example program 141
 alphanumerics, procedural
 example program 119
 annotating charts 50
 area
 chart 32
 aspect ratio
 of text in chart with CHAREA 109
 ATABOVE option 43
 ATCENTER option 43
 ATEND option 43
 attribute table (PG routines) 113, 115
 color 59
 line type 56
 line width 57
 marker 58
 pattern 65
 attributes, chart
 See chart attributes
 autoranging 41
 of secondary axis 118
 when moving to state-2 113
 autoscaling 41
 axis
 attributes 41
 autoranging 41, 113
 autoscaling 41
 bar chart x-axis scale marks 77
 characteristics
 bar chart x-axis scale marks 77
 duplicate 118

axis (*continued*)

intercepts 38, 39
 label 45
 label type 46
 labels, position of 45
 linear 41
 logarithmic 41
 markings 45
 obscured by data 118
 option calls 42
 options, example program 43
 PG routines calls 37
 positioning 38
 range 36, 41
 scale 41
 secondary 115, 118
 selection 117
 title position 43
 when drawn 118

B

BACK option 83
 background
 of chart 32
 bar chart 72
 CHBAR (create bar chart) 72
 CHBARX (create bar charts with numeric x values) 72
 composite 73, 94
 description of 27
 example 81
 floating 73
 gap between bar groups 74
 gap between bars 73
 multiple 73
 nominal numeric x axis 72
 shading options 74
 specify type 73
 values 75
 x-axis labels 72
 x-axis scale marks 77
 base position of chart note 50
 BGBASE option 84
 bibliography xii
 BKEY option 36
 blanking
 of label areas 48
 of legend area 36
 of note area 51
 of note area, example 102

index

blanks
 in year labels 47
BNOTE option 51
books, list of xii
bottom right cell of screen 100
box attributes (PG routines) 33
boxed legend 36
business charts
 ways of producing 1

C

calling ICU from application program 3
CBACK option 32
CBAR option 73
CBOX option 32
CHAATT (set axis line attributes) 41
character grid size (PG routines) 109, 110
CHAREA (define chart area) 32, 107
chart
 add company logos 68
 area 107
 background 32
 complex 107
 creating interactively 1
 framing box 32
 heading 49
 layout PG routines 32
 legend PG routines 34
 margins 33
 notes 50
 plotting
 table 104
 shading rules 29
 types 26
chart attributes
 axis line 41
 axis-title text 43
 chart background 33
 chart box 33
 datum line 49
 grid lines 48
 heading text 49
 key text 36
 label text 47
 note text 51
 spider label text 96
 spider tags 96
 value text 76
CHART call 21
chart types, mixing 115
CHBAR (create bar chart) 72
CHBARX (create bar charts with numeric x values) 72
CHBATT (set framing box attributes) 33
CHCGRD (set character grid) 110

CHCOL (set color table) 59
CHDATT (set datum line attributes) 49
CHDRAX (draw axes) 118
CHFINE (set curve-fitting smoothness) 56
CHGAP (set spacing between bars) 73
CHGATT (set grid line attributes) 48
CHGGAP (set spacing between bar groups) 74
CHHATT (set heading text attributes) 49
CHHEAD (set heading text) 49
CHHIST (create histogram) 69
CHHMAR (set horizontal margins) 33
CHKATT (set legend text attributes) 36
CHKEY (set legend key labels) 34, 94
CHKEYP (set base position of legend) 34
CHKMAX (set maximum legend height/width) 35
CHKOFF (set legend offsets) 35
CHLATT (set axis label text attributes) 47
CHLC (set component line color table) 60
CHLT (set component line type table) 56
CHLW (set component line width table) 57
CHMARK (set component marker table) 58
CHMKSC (set marker scale values) 59
CHNATT (set note attributes) 51
CHNOFF (set note offset) 51
CHNOTE (specify notes) 50
CHNUM (set number of components) 113
CHPAT (set component shading pattern table) 65
CHPCTL (control pie chart slices) 95
CHPEXP (exploded slices in pie chart) 95
CHPIE (create pie chart) 93
CHPIER (reduce pie chart size) 97
CHPLOT (create line graphs and scatter plots) 60
CHPOLR (create polar chart) 88, 91
CHRNIT (reinitialize chart definition options) 109
CHSET (set chart options) 32
CHSTRT (reset processing state to state-1) 109
CHSURF (create surface chart) 65
CHTATT (set text attributes) 43
CHTERM (terminate the PG routines) 52
CHTHRS (set bar-value threshold limit) 76
CHTOWR (create tower charts) 82
CHTPRJ (set tower chart projection) 83
CHVATT (set value of text attributes) 76
CHVCHR (set number of bar value characters) 75
CHVENN (create Venn diagram) 101
CHVMAR (set vertical margins) 33
CHXDAY (set x-axis day labels) 46
CHXDLB (set x-axis data labels) 46
CHXDTM (specify x-axis datum line) 49
CHXINT (set x-axis interception point) 39
CHXLAB (specify x-axis label text) 46, 94
CHXLAT (set x-axis label attributes) 47
CHXMTH (set x-axis month labels) 46
CHXRNG (set an explicit range of x axis) 42
CHXSEL (select x axis) 117

CHXSET (x-axis options) 38
 CHXTIC (set x-axis scale mark interval) 45
 CHXTTL (specify x-axis title) 43
 CHYDAY (set y-axis day labels) 46
 CHYDTM (specify y-axis datum line) 49
 CHYINT (set y-axis interception point) 39
 CHYLAB (set y-axis label text) 46
 CHYLAT (set y-axis label attributes) 47
 CHYMTH (set y-axis month labels) 46
 CHYRNG (specify an explicit range of y axis) 42, 113
 CHYSEL (select y axis) 117
 CHYSET (y-axis options) 38
 CHYTIC (set y-axis scale mark interval) 45
 CHYTTL (specify y-axis title) 43
 CHZDLB (set z-axis data labels) 84
 CHZGAP (set spacing between towers) 83
 CHZRNG (set an explicit range of z axis) 84
 CHZSET (z-axis options) 83, 84
 CHZTIC (set z-axis scale mark interval) 84
 close segment
 before PG routine plot 109
 COBOL
 example program 141
 color
 lines on charts 60
 of components (PG routines) 59
 table, PG routines 59
 commas
 in year labels 47, 67
 company logo
 on business chart 68
 complex legends 35
 complex PG routine charts 107
 component (PG routines) 24
 color table 59
 of pie chart 94
 component (PGF)
 preset how many 113
 composite bar charts 73
 control
 appearance of bars 78
 cross tick marks 45
 CSCCRT (create a chart) 8
 CSCDEL (delete a chart) 12
 CSCCHA (create character items) 9
 CSDEL (delete chart items) 18
 CSDIR (build a directory) 21
 CSFLT (set floating-point values) 10
 CSINT (set integer values) 9, 13
 CSLOAD (restore a chart) 8
 CSNUM (set control values for a chart) 8, 9
 CSQCHA (query character items) 10
 CSQCHL (query character string lengths) 10
 CSQCS (query CSxxxx call information) 20
 CSQDIR (query directory) 21

CSQFLT (query floating-point values) 10
 CSQINT (query integer values) 13
 CSQNUM (query control values) 8, 18
 CSQUID (query chart identification number) 8
 CSQXDT (query independent (x) values) 16
 CSQXSL (query selected x data) 17
 CSQYDT (query dependent (y) values) 9, 18
 CSQZDT (query data group (z) values) 16
 CSQZSL (query selected data groups (z)) 17
 CSSAVE (save a chart) 12
 CSSICU (start an ICU session) 11
 CSXDT (set independent (x) values) 16
 CSXSL (set data selection) 16
 CSYDT (set dependent (y) values) 9
 CSZDT (set data group (z) data values) 16
 CSZSL (select data groups (z)) 16
 curve fitting (PG routines) 56
 set smoothness 56
 CURVE option 56
 CVALUES option 75

D

data
 free 110
 overwrites axes 118
 paired 110
 pie chart 96
 tied 110
 data component (see data group)
 data file, ICU, contents of 6
 data group 24
 missing y values 24, 81
 date labels 46
 datum line (PG routines) 29, 49
 attributes of 49
 tower charts 84
 day labels 46
 default
 axis positions 37
 delay axis drawing 118
 draw
 axes 118
 DRAW option 118
 duplicate axis selection 118

E

example programs
 alphanumerics and charts 119
 chart notes 51
 COBOL 141
 ICU API (call CSxxxx) 7
 issuing several plotting calls 111
 mapping and graphics 141
 minimum business graphics 23

index

example programs (*continued*)
 mixing chart types 116
 multiple pie charts 114
 PG routines axis options 43
 PG routines line graph 60
 pie chart 98
 polar chart 89
 secondary axes 116
 simple business graphics 24
 surface chart 65
 tower chart 84
 two charts on one page 107
 Venn diagram 101
exploded pie charts 95
extend axis range to include zero 42

F

FBAR option 73
FILL option 63, 74
fineness of fitted curve 56
fitting curves (PG routines) 56
floating bar charts 73
FORCEZERO option 42
format file, ICU, contents of 6
framing box around chart 32
free data 110
FULL option 47

G

gap between bars 73
 groups of bars 74
 towers 83
GDDM Internet home page xii
graphics
 and mapping
 example program 141
grid lines (PG routines) 48
 attributes of 48
 tower charts 83
GRID option 48
grid size (PG routines) 109, 110

H

HBOTTOM option 50
HCENTER option 49
heading 49
 altering position of 110
 position 49
 text attributes 49
 using line-breaks in 49
HIGHAXIS option 38
histogram
 description of 27

histogram (*continued*)
 example 71
 summary of options 69
 suppress risers 69
 touching bars 69
HLEFT option 49
home page for GDDM xii
horizontal legend (PG routines) 34
horizontal margins 33
how charts are shaded 29
how to place legend within plotting area 35
HRIGHT option 49
HTOP option 50

I

ICU (Interactive Chart Utility) 1
 API introduction 1
 API uses 3
 API, call CHART 21
 API, call CSxxxx
 adding chart to current page 11
 attributes and options summary 14
 building directory 21
 creating a chart 8
 deleting a chart 12
 deleting chart items 18
 directory accessing 21
 directory functions 11
 displaying chart on output-only device 11
 example program 7
 excluding data 16
 getting a chart identifier 8
 library functions 11
 loading chart format and data 8
 options and attributes summary 14
 printing a chart 11
 querying chart control values 8
 querying data 16
 querying directory 21
 querying interaction result 18
 saving a chart 12
 selecting data 16
 setting character option values 9
 setting chart control values 9
 setting floating point option values 10
 setting integer option values 13
 setting x-data values 16
 setting y-data values 9
 setting z-data values 16
 starting an ICU session 11
calling from an application program
 call CHART 21
 call CSxxxx 3—21
data file contents 6
format file contents 6

ICU (Interactive Chart Utility) (*continued*)

- introduction 1
- methods of use 1
- naming of saved charts 6
- saving a chart 3
- stand-alone use 1
- IDRAW option 118
- incomplete pie chart 96
- INFILL option 63, 74
- Interactive Chart Utility (ICU)
 - See ICU
- intercept of axes 38
- INTERCEPT option 38, 39
- Internet home page for GDDM xii

K

- KBOX option 36
- keys for legend 34
- keys for legends
 - on multiplot charts 113
- KNORMAL option 36
- KREVERSED option 36

L

- LABADJACENT option 45
- label 45
 - abbreviated 47
 - area blanking 48
 - attributes 47
 - between ticks 45
 - day 46
 - month 46
 - numeric 46
 - rotating 47
 - user 46
- labels
 - apparently in wrong order 68
 - between ticks 68
 - centering by adding blanks 102
 - data 72
- LABMIDDLE option 45, 68
- legend 34
 - area blanking 36
 - boxing 36
 - height 35
 - horizontal 34
 - of mixed chart 115
 - offset 35
 - position 34
 - reversing key order 36
 - text attributes 36
 - vertical 34
 - width 35

- legend encroaches on chart 33
- LETTER option 47
- line graph 55
 - curve fitting 56
 - description of 27
 - example of graph 62
 - example program 60
 - markers 55
 - scatter plot 27, 55
 - set line types 56
 - set line width 57
 - set marker types 58
- line-break in heading 49
- line-break in key text 34
- line-type table (PG routines) 56
- linear axes 41
- LINEAR option 41
- LINES option 55
- logarithmic axes 41, 45
- LOGARITHMIC option 41
- logical x axis (PG routines) 72
- LOWAXIS option 38

M

- major tick marks 45
- Manhattan chart
 - (see tower chart)
- manuals, list of xii
- margins (PG routines) 33
 - when using CHCGRD 110
- marker table (PG routines) 58
- markers on line graph 55, 58
- MARKERS option 55
- markings on axis 45
- MBAR option 73
- MIDDLE option 38
- minor tick marks 45
- missing y values 24, 81
- mixing chart types 115
 - example program 116
- mixing PG routines and general graphics 109
- month labels 46
- MOUNTAIN option 64
- mountain-range shading 64
- multiline keys (PG routines) 34
- multiple bar charts 73
- multiple pie charts 97, 113, 114
 - do not fit 97
- multistage plot 110
 - of pie charts 113

N

- naming of saved ICU charts 6

index

- NBKEY option 36
- NBNOTE option 51
- NCBOX option 32
- NDRAW option 118
- negative tick marks 45
- NKBOX option 36
- NOBACK option 83
- NOCURVE option 56
- NOFILL option 74
- NOFORCEZERO option 42
- NOGRID option 48
- NOLAB option 45
- NOLINES option 55
- NOMARKERS option 55
- NOMOUNTAIN option 64
- NOPROPIE option 96
- NORISERS option 69
- NOSIDE option 83
- notes (PG routines) 50
 - area blanking 51
 - attributes of 51
 - example program 51
 - offset of 51
 - on chart 50
 - position of 50
 - rotating 51
 - text of 50
- NOVALUES option 75, 96
- NPGFS option 47
- NTICK option 45
- nulls used to pad keys (PG routines) 34
- numeric labels. 46
- NUMERIC option 46

O

- offset
 - of PG routines note 51
 - PG routines legend 35
- OPS 1
- option setting (PG routines) 37
- outlines on charts, color of 60
- overlap
 - of Venn diagram 101
- overlap shading, causes of 29

P

- padding keys with nulls 34
- paired data 110
- partial pie chart 96
- pattern table 65
- percentages in pie charts 96
- periods in year labels 47
- PERPIE option 96

- PG routines
 - call interface 31
 - chart layout calls 32
 - comparison with ICU API 1
 - complex charts 107
 - default values 36
 - grid line calls 48
 - legend calls 34
 - minimum example program 23
 - mixing with general graphics 109
 - note calls 50
 - option abbreviations 37
 - plotting calls 23
- PGF 1
 - entry-points 24
- PGF (Presentation Graphics Facility)
 - example program 119
- PGFS option 47
- pie chart 93
 - absolute data 96
 - description of 28
 - example program 98
 - incomplete 96
 - legend 95
 - multiple 94
 - multiple plot 113
 - multiple, does not fit 97
 - one above another 97
 - percentage data 96
 - proportional 96
 - shrink 97
 - specify title 94
 - spider labels 95
 - spider tags 95
- PIEKEY option 96
- placement of labels 45
- plain axis 45
- PLAIN option 45
- plotting
 - bar chart 72
 - histogram 69
 - line graph 55
 - pie chart 93
 - polar chart 88, 91
 - surface chart 63
 - table charts 104
 - tower chart 82
 - Venn diagram 101
- plotting against secondary axes 118
- plotting area 33
- polar chart 88, 91
 - description of 27
 - example 88, 92
- population of Venn diagram 101
- position legend 34

position of axis title 43
 position of chart heading 49
 positive tick marks 45
 PROPIE option 96
 proportional pie charts 96
 PTICK option 45
 publications, list of xii
 punctuation of labels and bar values 47

R

radar chart
 (see polar chart)
 range of axis 36, 41
 reference line (PG routines) 29, 49
 tower charts 84
 reinitialize PG routine options 109
 relative data 27
 return to state-1 109
 reverse key order 36
 risers 69
 RISERS option 69

S

scale marks (PG routines) 45
 scale of axis 41
 SCALETOWER option 83
 scatter plot 27, 55
 secondary axes 115
 secondary axis
 autoranging 118
 using 118
 segment
 close
 before PG routine plot 109
 selection of axis 117
 set color table 59
 set line-type table 56
 set line-width table 57
 set marker table 58
 set tick-mark interval 45
 set tick-mark style 45
 shading (PG routines)
 charts 29
 errors 29
 mountain range 64
 overlap, causes of 29
 patterns 65
 reference line 29
 shading errors, apparent (PG routines) 29
 shrink pie charts 97
 SIDE option 83
 skyscraper chart
 (see tower chart)

smoothness of fitted curve 56
 special characters for key text 34
 spider labels 95
 SPIDER option 96
 spider tags 95
 SPILABEL option 96
 SPISECTOR option 96
 stacked chart type 27
 star chart
 (see polar chart)
 state-1 (PG routines) 109
 state-1 datum line 29
 state-2 (PG routines) 109
 suppress
 axis drawing 118
 axis labels 45
 histogram's risers 69
 label punctuation 47
 lines on line graph 55
 markers on line graph 55
 shading of 1st component 63
 tick marks 45
 surface chart 63
 curve fitting 56
 description of 27
 example 67
 example program 65
 shading options 63
 switch axes 37

T

Table chart
 description of 28
 table charts
 PG plotting routines 104
 table of attributes (PG routines) 56
 terminate PG routines (CHTERM) 52
 text used with CHAREA 109
 tick marks (PG routines) 45
 on logarithmic axis 45
 style of 45
 tower charts 83
 z-axis 84
 tied data 110
 title for chart axis 43
 tower chart 82
 description of 27
 example 87
 TOWERTICK option 83
 trademarks ix
 two charts on one page 107

U

unboxed legend 36
user labels 46

V

values in bar charts, techniques 78
values on bar charts 75
VALUES option
 for bar chart 75
 for pie chart 96
varying length key text 34
VDIG option 76
vector text
 used with PG routines 109
Venn diagram 101
 description of 28
 example 103
 example program 101
 labels 101
vertical legend (PG routines) 34
vertical margins 33
VFIXED option 76
VINSIDE option 76
VONTOP option 76
VSCIENTI option 76

X

x axis vertical 37
x-axis scale marks, bar chart 77
XDUP option 118
XNODUP option 118
XPICK option 84
XTICK option 45
XVERTICAL option 37
 for pie charts 97
 for Venn diagram 101

Y

y axis vertical 37
YDUP option 118
year labels 67
YNODUP option 118
YVERTICAL option 37

Z

z axis angle and scale 83
zero included on autoscale 42
ZPICK option 84
ZVERTICAL option 37

Sending your comments to IBM

GDDM-PGF Version 2 Release 1.3

Application Programming Guide

SC33-0913-01

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form.
- By fax:
 - From outside the U.K., use your international access code + 44 1962 870229
 - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
 - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
 - IBMLink: HURSLEY(IDRCF)
 - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

GDDM-PGF Version 2 Release 1.3 Application Programming Guide

SC33-0913-01

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name

Address

Company or Organization

Telephone

Email

You can send your comments **POST FREE** on this form from any one of these countries:

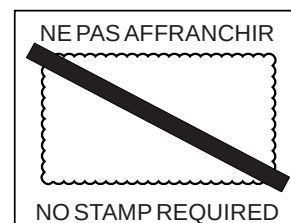
Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

2 Fold along this line

By air mail
Par avion

IBRS/CCR NUMBER: PHQ - D/1348/SO



REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____
Company or Organization _____
Address _____

EMAIL _____
Telephone _____

4 Fasten here with adhesive tape



1 Cut along this line

1 Cut along this line



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-0913-01

