

GDDM-PGF



OPS User's Guide

Version 2 Release 1.3

GDDM-PGF



OPS User's 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 xi.

First Edition (September 1996)

This edition applies to Version 2 Release 1 Modification 3 of the Graphical Data Display Manager – Presentation Graphics Facility (GDDM-PGF)

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.

This edition obsoletes a previous edition of information on OPS, the *OPS User's Guide*, order number SB11-1185-01.

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.

© **Copyright International Business Machines Corporation 1988, 1996. All rights reserved.**

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

Contents

Notices	xi
About this book	xiii
How to use this book	xiii
An important note on national languages	xiii
Latest GDDM information	xiii
GDDM publications	xiv

Part 1. Introducing OPS 1

Chapter 1. An overview of OPS	3
Creating online presentations with OPS	3
What you need to use OPS	4
Chapter 2. Starting OPS	5
Choosing the EXEC for your language	5
Starting OPS under CMS	5
Starting OPS under TSO	6
Chapter 3. Getting to know OPS	7
Choosing the sample you need	8
Making your own presentation from the sample	9
Printing or plotting the changed sample	11
Saving the changed sample for later use	11
Beyond the samples	11

Part 2. OPS: a user's guide 13

Chapter 4. Presentations	19
Types of presentations	19
Basic principles	20
How to create presentations	22
Chapter 5. OPS commands	37
Command structure	37
Infile commands	39
Comments	40
Defining keyboard keys as commands	41
Recalling commands	42
Error messages	42
Where to find information on commands	43
Chapter 6. Controlling OPS display	53
Selecting the mode	53
Using normal mode	53
Using dark mode	54
Controlling alphanumeric fields	55
Controlling the aspect ratio of the graphics field	58
Clearing the graphics field	59

Invoking GDDM user control mode	60
Using the coordinate system	60
Chapter 7. Text in presentations	65
Basic text handling	65
Advanced text handling	73
Chapter 8. Line art	103
Line and area attributes	103
Setting line attributes	104
Setting area attributes	105
Controlling color mixing	109
Drawing wide lines	110
Setting points on the screen	110
Drawing lines	111
Drawing polygons	114
Drawing arrows	115
Drawing boxes	119
Drawing circles	123
Drawing arcs	124
Drawing pie slices	125
Drawing rays	125
Drawing DASD symbols	126
Grouping objects together	128
Common additional features for fillable objects	130
Displacing objects	131
Filling the background	133
Drawing frames	134
Chapter 9. Displaying pictures	135
Displaying charts created by the Interactive Chart Utility	135
Using GDFs with OPS	138
Displaying images	146
Chapter 10. Creating dynamic sequences	151
Inserting pauses in presentations	151
Forcing the graphics field to the screen	152
Creating animated sequences	153
Displaying messages in dynamic sequences	153
Sounding the alarm	154
Examples of dynamic sequences	154
Chapter 11. Controlling the sequence of execution	157
Flow control commands	157
Using labels	158
Tagging objects	159
Getting tag values	160
Reading the command field	161
Chapter 12. Using symbolic variables	163
Defining and retrieving variables	163
How values are interpreted and handled	164
Concatenating variables	166
Using system variables	167

Sharing variables with ISPF	168
Querying variables	169
Chapter 13. Executing functions outside OPS	171
Browsing files	171
Editing files	171
Executing functions	171
Using the Interactive Chart Utility	172
Using symbol editors	173
Executing CMS commands	174
Executing TSO commands	174
Using ISPF functions	175
Chapter 14. Scrolling	177
Using scroll commands	177
Setting and using labels	179
Refreshing the page	179
Searching the infile	180
Automatic scrolling	181
Chapter 15. Printing and plotting	183
Printing panels	183
Plotting presentations	185
Creating page segments (PSEGs)	186
Chapter 16. Using dialed terminals under CMS	191
Establishing secondary terminals	191
Issuing commands from a secondary terminal	192
Chapter 17. Getting help	195
Using the OPS tutorial	195
Displaying the OPS help file	195
Getting command syntax help	196
Bypassing the help menu and command syntax help	197
Displaying file lists	197
Displaying the graphics overviews	198
Querying the value of system variables	199
Appendix A. OPS startup procedures	201
Starting OPS under TSO	201
Starting OPS under CMS	205
Appendix B. Runtime customization	209
Appendix C. OPS sample presentation (ADM7OPXA ADMOPS)	211
Appendix D. Messages	219
Message listing	219
Glossary	227
Index	235

contents

Figures

1.	First six sample layouts	7
2.	Second six sample layouts	8
3.	Sample presentation number five	9
4.	Infile for fifth sample	10
5.	First screen of ADM7OPXA	23
6.	Creating and saving graphics commands	30
7.	Adding objects: results of the exercise	33
8.	OPS presentation from PAUSE EXEC	34
9.	REXX EXEC to create and invoke OPS presentation	35
10.	OPS command list	44
11.	Panel layout in dark mode	54
12.	Graphics field with aspect ratio A4	59
13.	Coordinate system	61
14.	List of symbol sets	66
15.	Effects of variation in character box size	68
16.	Effect of CSP values on text	74
17.	Using CSP with non-proportional symbol sets	75
18.	Effect of different LSP values	77
19.	Text shearing	78
20.	Sample ADJUST commands	82
21.	TEXT alignment sample	88
22.	Centering text in a circle	88
23.	How to mirror text and turn it upside down	91
24.	Rotating and shearing text	92
25.	Text in simple boxes	93
26.	Examples of HEAD and FOOT	94
27.	Sample source for a text foil showing TRC codes and formatting	100
28.	A text foil using text flow functions	101
29.	Selecting of markers	101
30.	Line art samples	103
31.	Help panel for line attributes	105
32.	FILL help panel (first of two)	106
33.	Patterns in range 1000 to 1032	107
34.	Some wide lines	110
35.	Sample lines	112
36.	Examples of LINE and CURVE	113
37.	SHEAR and REFP	114
38.	Sample polygons	115
39.	Some simple arrows	116
40.	Help panel for arrow heads and tails	117
41.	Scaling arrow heads and tails	118
42.	Sample arrows with TURN and SHEAR	118
43.	BOX with various alignment codes	120
44.	Sample boxes	121
45.	Filling boxes	121
46.	Samples of TURN and SHEAR with BOX	122
47.	Using BOX with TEXT	123
48.	Circles and ellipses	124
49.	Circular commands: ARC, SECTOR, RAYS	125
50.	DASD commands: DASD, DASDX, DASDF	126

figures

51.	Samples of DASD and DASDF	128
52.	Examples of area filling in GDDM	128
53.	Result of AREA example	129
54.	Shadow samples	131
55.	Sample with local and global offset	132
56.	Fading background sample	133
57.	Sample management information system	137
58.	CHART versus CHARTX	137
59.	Some GDFs included in OPS	139
60.	Display from ADM7SAMP ADMGDF	140
61.	Displaying a GDF at reduced size	141
62.	Displaying GDF files with transformation	142
63.	Image using ASIS	149
64.	Sample Management Information System (extended)	155
65.	Print and plot from OPS	183
66.	Establishing a dialed terminal under VM	192
67.	The file list panel	198
68.	ADMOPSLx EXEC syntax under TSO	202
69.	ADMOPSLx EXEC syntax under CMS	205

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 publication 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, are 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, IBM Corporation, 500 Columbus Avenue, Thornwood, New York, 10594, U.S.A.

_____ Trademark Attribution _____

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

BookManager	BookMaster
GDDM	IBM
MVS/ESA	OS/2
OS/390	PS/2

_____ End of Trademark Attribution _____

notices

About this book

This book describes the Online Presentation System (OPS), a tool for developing presentations or online information using graphics. It introduces OPS step-by-step, with examples that you can try at your terminal.

How to use this book

To get an idea of what OPS is, and what it has to offer, read Chapter 1, “An overview of OPS” on page 3 and Chapter 2, “Starting OPS” on page 5. At a terminal, go through Chapter 3, “Getting to know OPS” on page 7.

You can then move on to the guidance section. Make sure that you read “How to create presentations” on page 22; then, using the OPS tutorial, try the commands (exercises) suggested in the various sections. Each section of the tutorial has a parallel chapter in this book to help your understanding.

Finally, once you’re familiar with OPS, you can get reference information, while you’re actually creating presentations, from the help built into OPS.

An important note on national languages

This book has been written in US English. If you are using OPS with a terminal and keyboard other than those configured for US English, note the following:

Notes:

1. Throughout this book, OPS variables are shown with the \$ character, entered from the keyboard as the primary currency symbol. To find the equivalent character on your keyboard, execute:

```
query sysvars
```

from within OPS. This displays the system variables, preceded by the character that applies to your setup.
2. The COLATTR command uses various characters as color attribute characters. Some of them, particularly characters shown in this book as ¢ and #, may be different for your setup. The best way of finding out is to try them, and use whatever works for you.

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
GDDM Base Application Programming Guide, SC33-0867
GDDM Base Application Programming Reference, SC33-0868
GDDM Diagnosis, SC33-0870
GDDM General Information, GC33-0866
GDDM/MVS Program Directory, GC33-1801
GDDM/VM Program Directory, GC33-1802
GDDM/VSE Program Directory, GC33-1803
GDDM Messages, SC33-0869
GDDM Series Licensed Program Specifications, GC33-0876
GDDM System Customization and Administration, SC33-0871
GDDM User's Guide, SC33-0875
GDDM Using the Image Symbol Editor, SC33-0920

GDDM-GKS *GDDM-GKS Programming Guide and Reference*, SC33-0334

GDDM-IMD *GDDM Interactive Map Definition*, SC33-0338

GDDM-IVU *GDDM Image View Utility*, SC33-0479

GDDM-PGF
GDDM-PGF Application Programming Guide, SC33-0913
GDDM-PGF Programming Reference, SC33-0333
GDDM-PGF Interactive Chart Utility, SC33-0328
GDDM-PGF Vector Symbol Editor, SC33-0330
GDDM-PGF OPS User's Guide, 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.

Part 1. Introducing OPS

Chapter 1. An overview of OPS	3
Creating online presentations with OPS	3
Advantages of online presentations	3
What you need to use OPS	4
Chapter 2. Starting OPS	5
Choosing the EXEC for your language	5
Starting OPS under CMS	5
Starting OPS under TSO	6
Allocating output data sets under TSO	6
Chapter 3. Getting to know OPS	7
Choosing the sample you need	8
Working with the presentation samples	8
Choosing your sample	9
Making your own presentation from the sample	9
Printing or plotting the changed sample	11
Saving the changed sample for later use	11
Beyond the samples	11

Chapter 1. An overview of OPS

The Online Presentation System (OPS) helps you to:

- Create and display online presentations with graphics
- Create high quality material for plotting or printing
- Create and modify graphics files

Creating online presentations with OPS

You can use OPS to create 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.

OPS presentations can include:

- Text in any available Graphical Data Display Manager (GDDM) font.
- Line art.
- Business charts created with the Interactive Chart Utility.
- Pictures created by OPS or other systems, in ADMGDF format. This includes pictures in Computer Graphics Metafile (CGM) format, created by products such as Freelance, and converted into ADMGDF format.
- Images supplied in ADMIMG or PSEG format

You can create the following output from OPS:

- Pictures in ADMGDF format
- Page segments (PSEGs) to be included in documents
- GDDM queued printer (ADMPRINT) files

Advantages of online presentations

OPS presentations have many advantages over conventional foil presentations:

- Once you’ve created a presentation, you can use it immediately, without creating foils or plots.
- People viewing your presentation will see exactly what you’ve created. You avoid problems of reproducing colors, patterns, and images on devices like plotters and printers.
- You can still create handouts, which do not require the same quality as a presentation.
- It’s easy to keep online presentations up-to-date.
- OPS lets you create more interesting presentations, providing:
 - Advanced line art: lines and arrows with arbitrary thickness, curvature and a number of points; a selection of arrow heads and tails; boxes with rounded corners or 3-D effect, and more
 - Dynamic or animated sequences
- When you’re displaying a presentation, you can change the content dynamically. You have access to all the graphics you have saved.

where OPS works

- It's easy to distribute OPS presentations to any VM/CMS or MVS/TSO user with a graphics terminal.
- Under CMS, OPS supports screen copying to dialed terminals. You can display presentations to others without sending them the presentation itself. You can therefore use OPS as part of videoconferences or teleconferences.

What you need to use OPS

OPS runs under MVS/TSO or VM/CMS. OPS requires Graphical Data Display Manager (GDDM) Version 3 Release 2.

OPS requires 500KB of user storage, in addition to GDDM's standard requirements. You need more if you display large images, or create page segments.

Under TSO, you must have Interactive System Productivity Facility/Program Development Facility (ISPF/PDF). Under CMS, you must have XEDIT.

You can use OPS from any display terminal supported by GDDM. Examples are the IBM products 3279, 3290, 3179G, 3192G, 3274G, 3270 PC (AT) with PS card, 3270 PC (AT)/G/GX, and any PS/2 with GDDM OS/2 LINK installed.

Chapter 2. Starting OPS

This chapter tells you how to start OPS under CMS and TSO, including how to select the version you need for the language you use.

Choosing the EXEC for your language

OPS is available in several language versions. Your enterprise may have installed one or more, and choosing the one most appropriate to you ensures that you get messages in a language you can understand.

To start OPS, use the appropriate startup EXEC name from:

ADMOPSLA US English
ADMOPSLC Simplified Chinese
ADMOPSLD Danish
ADMOPSLF French
ADMOPSLI Italian
ADMOPSLK Japanese Kanji
ADMOPSLN Norwegian
ADMOPSLS Spanish
ADMOPSLV Swedish

For simplicity, throughout this book, we assume that you are using the US English version, and use ADMOPSLA as the startup command, showing all displays in US English.

Starting OPS under CMS

Use the ADMOPSLA EXEC to start the US English version of OPS:

```
admopsla <filename <filetype <filemode>>> < ( options >
```

where filename filetype filemode identify a file containing an OPS presentation.

Unless the filetype is ADMOPS, the file must already exist. If the filetype is ADMOPS, OPS asks you whether you want to create a new file. If you do, OPS gives you the opportunity to copy a sample presentation into the new file.

For information on the options available on the OPS startup call, see Appendix A, "OPS startup procedures" on page 201.

If you enter the command ADMOPSLA with no other parameters, you'll find yourself using the OPS sample presentation files. To find out what these are, and how they can help you to use OPS as easily as possible, see Chapter 3, "Getting to know OPS" on page 7.

Starting OPS under TSO

Use the ADMOPSLA EXEC to start the US English version of OPS. Assuming that this is installed in a common procedure data set, the basic syntax is:

```
admopsla infile (<project project > <level level > <options>
```

where *infile* is an unqualified name of up to eight characters. For example:

```
admopsla mytest
```

infile is assumed to be the name of an OPS presentation residing in either your user OPS data set or the system OPS data set.

The default user OPS data set is *userprefix.ADMOPS*, but you can customize this in the EXEC.

The system OPS data set holds the OPS tutorial and a number of sample presentations mentioned throughout this book. If your private OPS data set does not exist, OPS offers to allocate it for you.

If you specify PROJECT as a startup parameter, OPS searches *project.level.ADMOPS*, before it looks in your user data sets or the system OPS data sets.

If *infile* is a new member in an OPS data set (either the default data set or any named data set), OPS asks whether you want to copy the sample presentation into the new file.

Allocating output data sets under TSO

If you want to save any graphics or create page segments, you must allocate appropriate output data sets. The ADMOPSLA EXEC will allocate the following libraries for output (and input), if found:

<i>userprefix.ADMSYMBL</i>	For GDDM symbol sets
<i>userprefix.ADMGDF</i>	For GDF files
<i>userprefix.ADMCDATA</i>	For ICU data
<i>userprefix.ADMCFORM</i>	For ICU format

If you don't have the appropriate output data set, you won't be able to use the associated output function. For example, if you don't allocate an ADMGDF data set, the GDFSAVE command is suspended.

If you know in advance that you won't need any output data sets, you can avoid allocating them by specifying BROWSE as an option on ADMOPSLA EXEC.

You can also tell OPS to create new data sets for you by specifying ALLOC, as in:

```
admopsla myfirst ( alloc
```

For a full description of ADMOPSLA EXEC options, see Appendix A, "OPS startup procedures" on page 201.

Chapter 3. Getting to know OPS

To help you to get started with OPS, we've supplied twelve sample presentation files, based on the most popular presentation style formats.

Typing the appropriate startup command without a file name or any other parameters takes you to the sample page layout menus. These are provided **only** in US English, whichever startup EXEC you use.

There are six pages on the first screen of the sample, as shown in Figure 1.

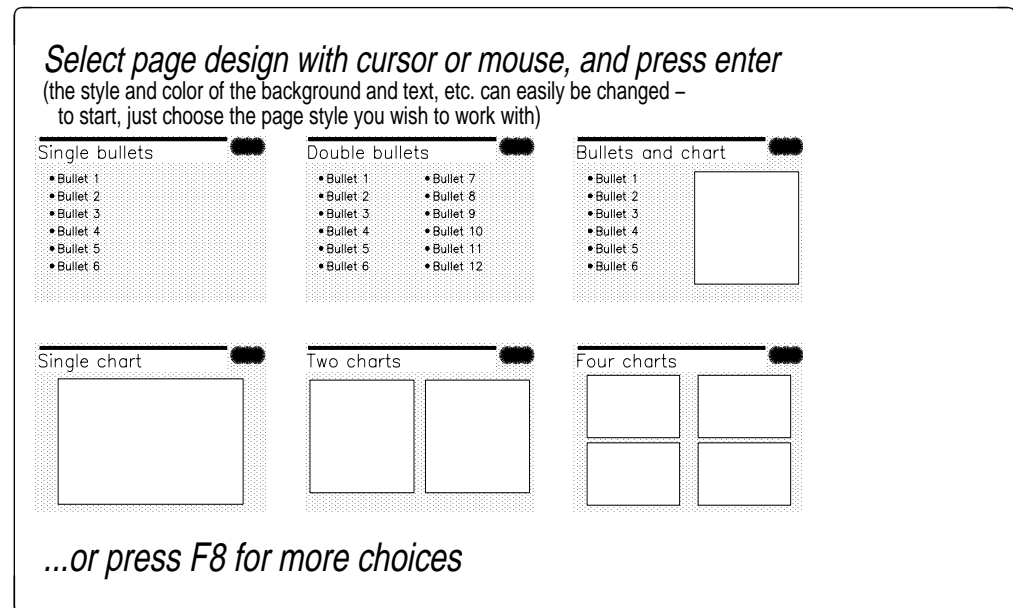


Figure 1. First six sample layouts

Press PF8 to show the second set of six, shown in Figure 2 on page 8.

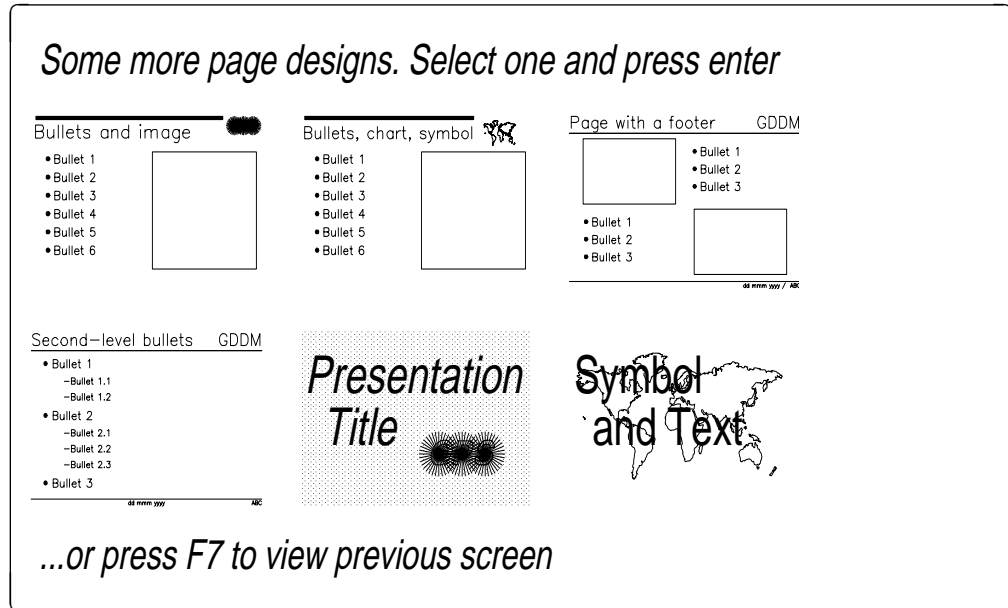


Figure 2. Second six sample layouts

You can use these samples, not just to learn about OPS, but also as the basis for your own presentations. Here's a summary of the process:

1. Choose the sample closest to what you need.
2. Change it to create your own presentation.
3. Print or plot the presentation.
4. Save the changed sample for your own use.

The rest of this chapter describes each step in the process.

Choosing the sample you need

Before choosing the sample you want, you need to understand what happens when OPS displays the sample presentation.

Working with the presentation samples

OPS keeps the samples on the system read-only disks (CMS) or the system read-only data sets (TSO). So that you can work with your chosen sample, OPS first copies them into your own storage. Under CMS, it does this sample by sample; under TSO, it copies all the samples when it starts.

Using the samples under TSO

When ADMOPSLA starts, the first thing you see is the message:

```
Copying examples myops0 through myops12 to user.admops
```

OPS prompts you to let it continue. When you do so, it copies the samples from the system data sets to your own ADMOPS data sets, replacing any that already exist. From then on, you're working on these copies, which you can edit and save.

Using the samples under CMS

When you select the presentation you want, from either the first or second screen of samples, OPS automatically copies the appropriate sample to a file called MYOPS n ADMOPS on your A-disk, where n is the number of the presentation. If a file of the same name already exists, OPS lets you choose whether to use that one or overwrite it with the new one.

From then on, you're working on this MYOPS n file, which you can edit and save.

Choosing your sample

Look at the 12 samples on offer, and choose the one with the page layout closest to what you want to achieve. Don't worry about colors and styles at the moment; you can easily change these by editing the source (in OPS terms, the **infile**), as explained in "Making your own presentation from the sample."

Let's assume that you're working under CMS, and want the fifth sample (the two-chart sample in Figure 1 on page 7). Using your mouse or cursor, you can select it, and then press ENTER. This displays `infile MYOPS5` at the bottom of the screen.

Pressing ENTER again starts a new OPS session with the display shown in Figure 3.

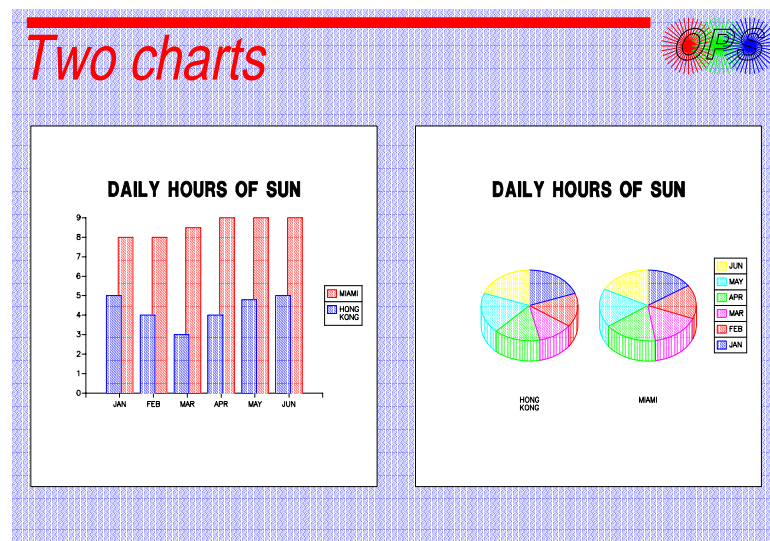


Figure 3. Sample presentation number five

Making your own presentation from the sample

Now tailor the sample you've chosen to your own requirements.

You'll want to change the text of the heading, and perhaps import new data for the charts.

Press PF12 to edit the command file used to create the page. There are comments in the file to help you identify the various commands, but you'll find most of them self-evident.

Continuing with our use of sample five, here's the infile (source) you see when you press PF12:

```
1 )ops*****
)ops prefix ;; reset; trc off
)ops dark; aspect 100 67; reset
: * draw the red line
: line 2 65.4 83 w 1.2 dr r
: mix on
: * blue background, shading pattern 3
: backgr b 3
: rays 88 62.5 7 dr r
: rays 92 62.5 7 dr g
: rays 96 62.5 7 dr b
: text 'OPS' cc 92 62.5 ss admuukso h 5 sh 15 col w
: mix off
: * The next line is the chart heading. Uses admuuhi as the symbol set
: * is drawn in red, with a height of 7.8.
: text 'Two charts' 1 57 ss admuuhi col r h 7.8
: * draw two black boxes
: fill neutral
: box 75 30 45 45
: box 25 30 45 45
: * the following are the chart load statements...
: gdf adm7dhsb 25 30 45 45
: gdf adm7dhsp 75 30 45 45
: msg 'Press F12 to work with the commands for this picture. F3 to quit OPS'
```

Figure 4. Infile for fifth sample

Try changing one or two things, like the color of the lines or the background. Then press PF3 to see the result of your changes. Pressing PF12 will then take you back to the file edit session, and you can use PF3 and PF12 to switch between viewing and editing until you're satisfied with the results.

(You can enter commands directly on the OPS command line at the bottom of the screen, and see the results immediately, rather than going in and out of the edit session. "Testing commands" on page 27 tells you how to do this.)

Note: If you intend to output your finished page to a pen plotter rather than a printer, don't include a background. Many GDDM patterns (which are used for backgrounds, amongst other things) are composed of dots, and take a very long time to plot. Remove the background command (for example, BACKGR B 3) by deleting the whole line, or by commenting out the command by prefixing it with an asterisk.

If you're using an older-style pen plotter, such as a 7371 or 7372, we also recommend that you use non-filled vector fonts for your text (fonts such as ADMUUSRP, ADMUUDRP, or ADMUUKSO), rather than the filled fonts (ADMUUH, ADMUUT, and so on) used in the sample files. This is because the filled variety take a long time to plot on these devices. On faster plotters, such as the 7374 or 6184, this is not a problem.

Printing or plotting the changed sample

When you are satisfied with your page, you can print or plot it using:

- One of the OPS commands available (see Chapter 15, “Printing and plotting” on page 183).
- GDDM’s User Control function to output it to any device you have defined. This is especially useful if you want PostScript output.

You can also save your picture in ADMGDF format using the OPS command GDFSAVE.

Saving the changed sample for later use

To save your changed sample, use the appropriate file command from within the editor. It’s a good idea to change the name of the file from the default, because if, in the future, you display the same sample again from the sample presentation, you might overwrite the changed version.

Using sample five again, and assuming that you’re using XEDIT under CMS, you can save your finished presentation by executing:

```
file newops5
```

creating NEWOPS5 ADMOPS on your A-disk.

You can then use OPS to view and change it in the future by:

```
ADMOPSLA NEWOPS5
```

Note: Under TSO, the process is different, because displaying a sample *always* replaces one with the same name that already exists in your own ADMOPS data set. To save your changes for future use, you must rename or copy the saved sample using TSO commands.

Beyond the samples

The 12 samples offer some typical, basic, presentation layouts, which should help to get you started. OPS has a wealth of additional function, including arrows, curves, and boxes, which you can use as you get more familiar with the tool. You can find examples of these functions in other OPS command files.

However, using the information in this chapter alone helps you create one page of a presentation. A full, multipage presentation is simply a collection of these single pages.

For more information on creating presentations, see Chapter 4, “Presentations” on page 19.

beyond the samples

Part 2. OPS: a user's guide

Chapter 4. Presentations	19
Types of presentations	19
Public presentations	19
Private presentations (online information)	19
Basic principles	20
Presentation pages	20
Presentation principles by example	20
Knowing where you are at the start of presentation pages	21
The first page of a presentation	22
How to create presentations	22
How to start	22
Editing	23
Displaying the coordinate system	24
Adding objects to the picture	24
Controlling the accuracy of cursor reading	25
Reading coordinates from the screen	26
Deleting objects	26
Working in draft mode	27
Testing commands	27
Copying and moving objects	29
Deleting objects	30
Adding commands to your presentation	30
Changing your infile	31
Leaving OPS	31
Exercise: adding and moving objects	32
Creating presentations using EXECs or CLISTS	34
Chapter 5. OPS commands	37
Command structure	37
Command words	37
Operands	38
Infile commands	39
Changing the command prefix	40
Comments	40
Defining keyboard keys as commands	41
Assigning the PF keys to commands	41
Assigning the Enter key to commands	41
Recalling commands	42
Recalling the last command	42
Recalling the last 100 commands	42
Error messages	42
Where to find information on commands	43
OPS command list	44
Chapter 6. Controlling OPS display	53
Selecting the mode	53
Using normal mode	53
The message field	54
The command field	54
The graphics field	54

Using dark mode	54
Controlling alphanumeric fields	55
Controlling the message field	56
Controlling the command field	56
Controlling PF keys	57
Controlling the aspect ratio of the graphics field	58
Clearing the graphics field	59
Invoking GDDM user control mode	60
Using the coordinate system	60
Units in the coordinate system	62
Adjusting the coordinate system	64
Chapter 7. Text in presentations	65
Basic text handling	65
Symbol sets	65
Controlling the character box	67
Coloring text	69
Using translation tables	71
Releasing symbol sets	72
Advanced text handling	73
Setting defaults for text	73
Controlling the space between characters	74
Putting text in the infile	76
Controlling the space between lines	76
Italicizing text	78
Positioning text	79
Coloring individual words and characters	84
Manipulating text strings	86
Forcing a line break	93
Creating headers and footers	93
Text formatting commands	95
Writing markers on the screen	101
Chapter 8. Line art	103
Line and area attributes	103
Setting line attributes	104
Unspecific DRAW attributes	105
Setting area attributes	105
Using basic colors and patterns	106
Erasing areas before filling	107
Unspecific FILL attributes	108
Using general colors with OPS	108
Controlling color mixing	109
Drawing wide lines	110
Setting points on the screen	110
Drawing lines	111
Defining the line	111
Drawing curves	113
Turning and shearing lines	113
Drawing polygons	114
Drawing arrows	115
Controlling arrow heads and tails	116
Drawing boxes	119
Controlling the alignment of boxes	119

Adding a 3-D effect to boxes	120
Controlling attributes of boxes	120
Turning and shearing boxes	122
Using BOX as a subcommand of TEXT	122
Drawing circles	123
Drawing arcs	124
Drawing pie slices	125
Drawing rays	125
Drawing DASD symbols	126
Grouping objects together	128
Common additional features for fillable objects	130
Erasing fillable objects	130
Drawing shadows behind fillable objects	130
Displacing objects	131
Filling the background	133
Drawing frames	134
Chapter 9. Displaying pictures	135
Displaying charts created by the Interactive Chart Utility	135
Adding an ICU chart to the current picture	135
Displaying a chart on a clear screen	137
Using GDFs with OPS	138
Transforming GDFs	139
Handling symbol sets with GDFs	139
Displaying GDFs	140
Displaying GDFs on a clear screen	144
Saving the graphics field as a GDF	144
Editing GDFs without the OPS source	145
Displaying images	146
Basic image display	147
Scaling images	147
Defining the position and size of images	148
Controlling the color of images	149
Chapter 10. Creating dynamic sequences	151
Inserting pauses in presentations	151
Forcing the graphics field to the screen	152
Creating animated sequences	153
Displaying messages in dynamic sequences	153
Sounding the alarm	154
Examples of dynamic sequences	154
Chapter 11. Controlling the sequence of execution	157
Flow control commands	157
Using labels	158
Tagging objects	159
Getting tag values	160
Reading the command field	161
Chapter 12. Using symbolic variables	163
Defining and retrieving variables	163
How values are interpreted and handled	164
Interpreting numeric values	164
Interpreting quoted strings	165

Interpreting general phrases	165
Concatenating variables	166
Using system variables	167
Status values	167
Miscellaneous	167
Coordinates	168
Sharing variables with ISPF	168
Querying variables	169
Chapter 13. Executing functions outside OPS	171
Browsing files	171
Editing files	171
Executing functions	171
Using the Interactive Chart Utility	172
Using symbol editors	173
Executing CMS commands	174
Executing TSO commands	174
Using ISPF functions	175
Chapter 14. Scrolling	177
Using scroll commands	177
Synchronizing the page number	178
Sideways scrolling	178
Setting and using labels	179
Refreshing the page	179
Searching the infile	180
Searching an infile containing symbolic variables	180
Automatic scrolling	181
Chapter 15. Printing and plotting	183
Printing panels	183
Creating PostScript output	185
Plotting presentations	185
Creating GL files	186
Creating page segments (PSEGs)	186
Controlling output under CMS	186
Controlling output under TSO	187
Page segment formats	188
Converting colors into gray shades	188
Previewing the result	188
Automatic scaling of images	188
Creating PSEGs with AUTO	189
Chapter 16. Using dialed terminals under CMS	191
Establishing secondary terminals	191
Defining a secondary terminal	191
Dialling from a secondary terminal	191
Issuing commands from a secondary terminal	192
Copying the graphics field to a secondary terminal	193
Chapter 17. Getting help	195
Using the OPS tutorial	195
Displaying the OPS help file	195
Finding your way round the help file	195

Getting command syntax help	196
Bypassing the help menu and command syntax help	197
Displaying file lists	197
Displaying the graphics overviews	198
Querying the value of system variables	199

Chapter 4. Presentations

OPS presentations are infiles containing a mix of presentation content and **infile commands**. Under CMS, the filetype must be ADMOPS. Under TSO, the presentation must exist in the data set allocated to *ddname* ADMOPS.

The infile commands let you create and present screens consisting of text, charts, images, and other graphic objects. The screens can be presented dynamically, in a step-by-step manner controlled by the user or time. They can even include animation.

This chapter introduces you to OPS presentations, and tells you how you create them. Later chapters describe the commands you put in the infile.

Types of presentations

Presentations are of two main kinds:

1. Public presentations are shown to an audience by the author.
2. Private presentations are intended for display to one OPS user; they are often called **online information** because their most common use is to distribute information.

However, you can also use OPS to display information on a secondary dialed terminal (CMS only) to a person with whom you also have audio connection.

Public presentations

In public presentations, OPS must be as discreet as possible: no messages, no commands, everything controlled by the PF keys. If you need to issue a command for activities you had not planned, for example, this must be done as discreetly as possible.

To help this, OPS provides **dark** mode, where hardly a trace of OPS can be seen on the screen. The command field is invisible and placed at the bottom of the screen where you are not likely to see much of it during presentations on a video projector. For the same reason, the default color of messages is blue.

Private presentations (online information)

Online information is presented to users who do not know the content or the sequence; naturally they need help. You can still use dark mode, but you should provide visible information about what to do by, for example, defining the PF keys and displaying that information.

OPS also allows other messages to be displayed in the message field. If the user needs to enter commands, you can use the CMDFLD command to show leading text with, say, the arrow (“===>”).

Basic principles

Here we tell you what you need to understand about OPS presentations before you start to develop your own.

Presentation pages

OPS presentations are divided into **pages**; each page should describe one panel.

There are four rules about presentation pages:

1. Column 1 is reserved for carriage control according to **American Standards Association (ASA) standards**:

- “1” in column 1 means **new page**
- “+” in column 1 means **overwrite**

Other ASA codes are not relevant. OPS will automatically assume that the two ASA standards above apply when the file has a filetype of ADMOPS under CMS, or is an ADMOPS data set under TSO.

2. A page can contain any mixture of text lines and lines with OPS file commands. The command lines start with the command prefix, default)OPS, to distinguish these from text lines.
3. Several commands in the same line are separated with “;” (semicolon).
4. You can put comments anywhere you like using the comment command “*.”

Presentation principles by example

The following example, which presents online information, shows you the principles:

```
>.<--- Column 1 in the infile ("1" is "new page")

1 )ops                * as discreet as possible - DARK
)ops dark ;           * this is also a comment
)ops ss admuwtrp; col y ;* symbol set and color
)ops height 30mm ;    * text height in absolute terms

        Press PF8 for more information

1 )ops chart adm7smp3; * an ICU chart in full size
1 )ops gdf adm7samp;  * this time a gdf
1 )ops gdf mygdf ;    * and finally ...
1 )ops                * last page is dark, with a message
)ops msg 'The End. Press PF3 to exit.'
```

This example consists of five pages.

The first contains a text line (“Press PF8 ...”) which, because of the OPS commands, is shown with a graphics symbol set and in yellow.

The next three pages show pictures (an ICU chart and two GDF-pictures).

The last page tells you that the presentation is over.

Note that even the first page has a 1 in column 1 (“new page”). This ensures that OPS understands carriage control ASA, even if the file has a filetype other than ADMOPS (under CMS), or the file is not an ADMOPS data set (under TSO), in which case, ASA is assumed. It causes the first page to be page 1, and not page 0.

Note the use of “;” which separates commands (and comments) from each other. Do **not** use “;” as “end of command” where no other command follows.

Also note that)OPS does not need to be in front of all commands, but only once at the beginning of a line that includes commands. The space between)OPS and the command name is optional.

Knowing where you are at the start of presentation pages

OPS provides ways of letting you set up the environment at the start of each page to ensure that you can use your presentations flexibly. There are three commands involved: PREFIX, RESET, and DEFAULTS.

Rules for using the PREFIX command

The PREFIX command lets you change)OPS to something else (usually to something shorter). You’ll find a description of this in “Infile commands” on page 39.

When you skip pages while browsing the infile, OPS does not analyze the content of what you skipped. This means that it misses any changes to the command prefix in the skipped part, and won’t be able to recognize command lines with the new prefix.

Never change the prefix in pages other than the first, unless you **either** change back within the same page **or** repeat the PREFIX command on the top of each page. In fact, defining the PREFIX at the start of each page is a good idea. It makes it easier to copy pages into other presentations.

Rules for using RESET and DEFAULTS

It is a good idea to ensure that you know the values of important variables at the start of each new page. As with using the PREFIX command, this ensures that you know the results of jumping between the pages in a presentation, and makes it easier to move a page from one presentation to another.

The RESET command resets indentation, centering, line space, colors, patterns, the message field, and so on, to their defaults.

The defaults are defined in the system, but you can redefine them with the DEFAULTS command. For example, you can use DEFAULTS to change the default symbol set (VSS), text color (turquoise), and text size (the size of the cursor). For more information on the DEFAULTS command, see “Setting defaults for text” on page 73.

The first page of a presentation

It is important that you get a good start, which each of the twelve sample presentations will give you. Just type ADMOPSLA with no parameters, and go on from there as explained in Chapter 3, "Getting to know OPS" on page 7. Figure 4 on page 10 shows the source for one of these presentations.

How to create presentations

Having established the basic principles, let's look at how you go about creating a presentation.

Obviously you could edit the infile, entering text and commands just as you would do with any document or source program, and then start OPS against the file and check the result. However, this would be slow. OPS includes a number of development tools to speed up the process.

In the sections that follow, you will find detailed descriptions of the following commands, which help you develop presentations:

EDIT	Edit the infile
GRID	Display the coordinate system
GRAIN	Set the accuracy of screen readings
GETC	Read coordinates from the screen
UNDO	Remove the last or a selected object
DRAFT	Use simplified drawing mode
TEST	Test an object command
COPY	Copy an object
MOVE	Move an object
DELETE	Delete an object
PUT	Put a finished command into a file
END	Leave OPS

You'll also find your work easier if you have the the following PF keys assigned to OPS commands (as OPS does by default):

PF3	END
PF4	GRID SW
PF5	LAST ALL
PF6	LAST
PF9	UNDO
PF10	TEST
PF11	PUT
PF12	EDIT

How to start

The simplest way to write an OPS presentation is to base it on an existing presentation, making use of what you, or somebody else, already wrote.

You can use the sample presentations described in Chapter 3, "Getting to know OPS" on page 7, or the sample ADM7OPXA ADMOPS, which you can copy into any new presentation you create.

Note: The ADM7OPXA file comes in a US English version only, no matter what startup EXEC you use.

Start a new presentation, say TEST91, using the US English startup EXEC:

```
admopsla test91
```

When you start OPS with a new file name, it first asks:

```
Do you want a new file created? 1 (Yes) 0 (No)
```

Answer 1, and OPS asks:

```
Do you want sample ADM7OPXA ADMOPS copied? 1 (Yes) 0 (no)
```

Answer 1, and under CMS OPS creates file TEST91 ADMOPS, containing ADM7OPXA, and immediately displays the first sample screen, as shown in Figure 5. Under TSO, you get a new member called TEST91 in the data set allocated with *ddname* ADMOPS.

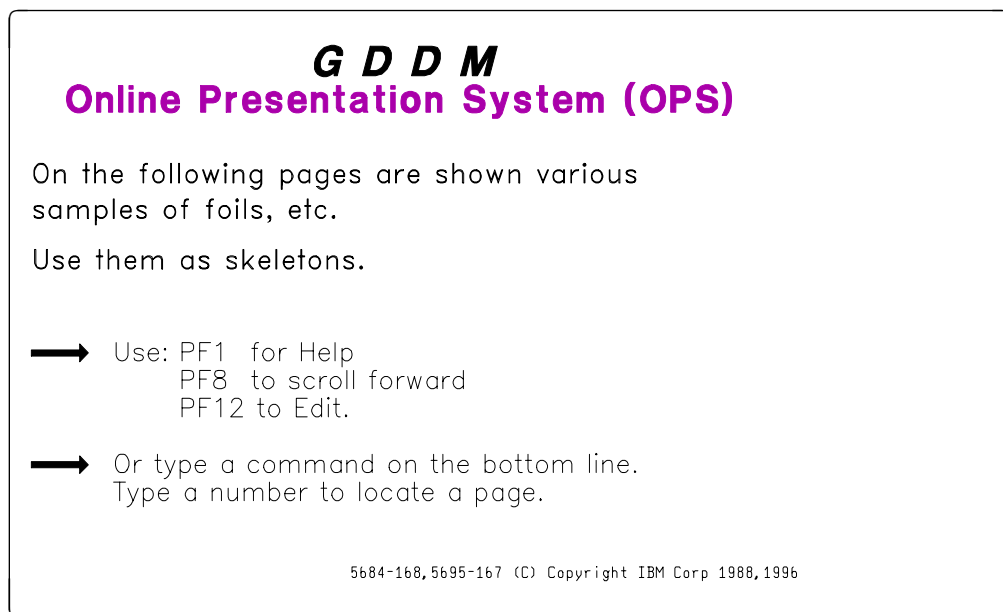
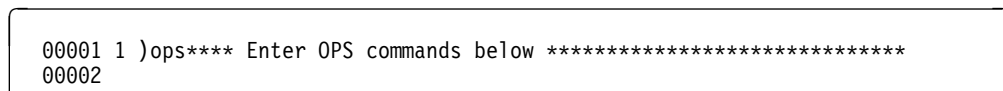


Figure 5. First screen of ADM7OPXA

If you answer 0 (no) to the sample file prompt, you get an almost empty file, looking something like:



ADM7OPXA ADMOPS is reproduced in Appendix C, “OPS sample presentation (ADM7OPXA ADMOPS)” on page 211.

Editing

You start the editor with the EDIT command (normally assigned to PF12), and go on editing the file, until you’ve finished your presentation.

The edit macro, ADMPAGE, is useful for navigating round the file. Type ADMPAGE without an operand, and OPS positions the file on the page you were viewing when you entered the editor. That’s usually the page you want to change.

adding objects

You can also add a specific page number or a relative page amount:

```
===> admpage 12
===> admpage +2
```

The first example finds page 12; the second scrolls forward (down) two pages.

When you leave the editor using PF3, OPS restarts the presentation. It displays a message telling you which page you last viewed.

```
Restarting after Edit. Last page viewed was page 3.
```

The last viewed page is probably also the page you want to see, so OPS puts the page number in the command field. Just press Enter to go to the last viewed page.

Displaying the coordinate system

Use GRID to display the coordinate system. You can specify color as an operand, optionally preceded by the keyword COLOR. For example:

```
grid red
grid col 11
```

The default is grey or blue, depending on what your terminal supports.

The coordinate system is drawn with dotted lines horizontally and vertically. The maximum y-value, which is terminal-dependent, is displayed in the message field.

```
&$YMAX 66.75
```

If you need a denser grid, specify DOUBLE (or D). For example:

```
grid d
grid b do
```

will draw the grid lines with a separation of 5 units.

To remove the coordinate system, use:

```
grid off
```

You can also use the keyword SWITCH to toggle the grid status on or off.

If you often use percentages in coordinates and dimensions, use the **percentage coordinate system**, in which both x and y range from 0 to 100. To see the percentage coordinate system, type:

```
grid % white
```

Adding objects to the picture

To add an object (graphics or text), type the complete command in the command field and press Enter.

However, you can also leave out all coordinates from the command and let OPS prompt you. For example:


```
box
box fi red
circle w 2
```

Depending on the object, OPS asks you to point at, for example, the center of the object, or the next point of the line, or whatever is appropriate. Use your mouse or your cursor to point at the screen.

The coordinates you identify are inserted automatically in the command. OPS calculates the width, height, or diameter of the object as appropriate, and inserts this into the command. You might get:

```
box 50 32 20 10
box 40 30 30 20 fi red
circle 30 20 10 w 2
```

For BOX, GDF, CHART, IMAGE, and TEXT, the prompting depends on your choice of syntax. For example, by default, OPS requests the center of a BOX, but by specifying CO or CE immediately after the BOX command, you make OPS prompt you for box corners instead. For TEXT, the lower left hand corner is requested by default, but by specifying CC, CE, or RI immediately after the text string, you can modify the prompting:

```
box co dr yel
gdf mygdf co
text 'hello there' cc
```

By default, after drawing the object, OPS leaves the command in the command field, ready for you to modify. You can change anything, add new parameters, or change to a completely different kind of object. When you press Enter again, the object on the screen changes in accordance with your request. To avoid changing the object, clear the command field and press Enter.

This mode of operation is called **test mode**, and is further discussed in “Testing commands” on page 27.

Controlling the accuracy of cursor reading

You can control the accuracy of the cursor reading with the GRAIN command. For example:

```
grain 0.1
```

rounds coordinates read from the screen to one decimal place. The maximum accuracy supported is two decimal places.

Increasing the accuracy is useful when you use OPS in GDDM’s user control mode, and zoom into more detail and try to position things accurately.

Decreasing the accuracy is useful when you draw a grid, and want the lines in the grid to go through points with x- and y-values divisible by, for example, 5.

```
grain 5
```

Reading coordinates from the screen

GETC reads the coordinates directly from the screen and into a command. Use GETC as a subcommand anywhere you need a coordinate pair, and where the automatic prompt for coordinates is what you need. Look at the following examples:

```
box getc 50mm 40mm, dr red  
line getc 50.25 30 g
```

In the first example, you want a box of a specific size. In the second example, you want a line to pass through a specific point which could be difficult to pinpoint on the screen. The second example includes two GETC subcommands, the second showing that you can abbreviate GETC to G.

GETC prompts you to move the cursor to a point and press Enter. If you have a graphics cursor (a mouse, puck, or stylus), GETC enables it. GETC reads the coordinates in the OPS coordinate system, by default rounds off to the nearest integer, puts a green cross at the point, and inserts the two coordinate numbers in the command field. The result of this on the two commands shown above could be:

```
box 32 25 50mm 40mm, dr red  
line 32 25 50.25 30 getc
```

Then the processing of the command continues.

If you have only the alphanumeric cursor, subsequent GETC commands prompt in the same way. After the second point has been established, a dotted green line is drawn between already established points to show the result. You'd see this if you continued processing the second example above, executing the second GETC.

The green crosses and dotted lines are **temporary graphics**. They are not a part of the picture (they will not be saved if you GDFSAVE the picture), and will disappear when you next update the screen.

Deleting objects

When you're experimenting with a composition of graphics and text using direct commands, you will often need to delete objects that have been positioned wrongly. You do this with the UNDO command, one object per command.

To delete the last drawn object, the command is simply:

```
undo
```

By default, UNDO is assigned to PF9 for ADMOPS files.

Use UNDO repeatedly to delete objects one at a time, from the last drawn to the first drawn.

You can delete the following objects:

- Text from text lines positioned by START.
UNDO deletes all the text between START and any graphics commands that display on the screen, WAIT, FORCE, another START, or end-of-page.
- Text positioned via a TEXT, HEAD, or FOOT command.

- All graphics positioned with OPS commands (POINT, LINE, and so on).
- Most GDF files.
- Images.

You **cannot** delete ICU charts.

To delete an object other than the last drawn, put UNDO in front of the command that created the object. For example:

```
undo box 30 20 30mm 10
```

undoes the box drawn by BOX 30 20 30mm 10.

In practice, you can do this without any typing. Use LASTcmd (PF6) or LAST ALL (PF5) to retrieve the object command in the command field, then press PF9 (UNDO) to delete the object.

UNDO leaves the object command in the command field, which allows you to redo it immediately.

Working in draft mode

GDDM supports a simplified way of working, called **draft mode**, in which the screen is not redrawn when you delete segments. You get a simplified picture, where objects may overlay each other incorrectly or be transparent to underlying objects. Filled symbol sets are also reproduced in a simplified way.

Draft mode is well suited for development, where you often want to delete segments, and where economy and speed are more important than quality.

Note: Draft mode is not supported on all terminals. For example, it is not supported on the IBM 3279, under GDDM OS/2 link, or under version 3 and beyond of PC3270, which contains the OS/2 link code within the emulator.

You enter draft mode either from the GDDM control mode panel (reached, by default, when you press PA3), or using the OPS DRAFT command:

```
draft
```

To leave draft mode, use DRAFT OFF.

Testing commands

Use TEST to produce those OPS commands which generate objects that you can delete, as listed in “Deleting objects” on page 26; that is, almost anything but ICU charts.

Using the TEST command

In practice, you never type the command TEST. You use it either from a PF-key or **implicitly** whenever you press Enter on a suitable object command.

By default, PF10 is set to TEST. To use TEST **explicitly**, type the graphics command only and press PF10:

```
box 50 40 30 20      + PF10
```

testing commands

We used TEST implicitly in “Adding objects to the picture” on page 24, where just pressing Enter on an object-creating command, such as BOX, started a test cycle.

You turn the implicit use on and off by:

```
test on
test off
```

When using implicit mode, you **must** remember to stop the test cycle, or each new object will replace the previous one. As soon as you have completed an object, use a command such as PUT, or an empty line, to stop testing.

Modifying an existing object

Use TEST to modify an already positioned object (other than the last), as follows:

1. Retrieve the associated command in the command field.
2. Undo the old object.
3. Continue with TEST.

You can achieve steps 1 and 2 using UNDO, but it’s better to retrieve the command and enter TEST mode in one step, by using either of the following:

- The LAST ALL command
- Object picking

Retrieving commands:

1. Enter LAST ALL (PF5) and find the command you’re interested in.
2. Press PF10 (TEST) on the last-command panel, and you return to the main panel with the selected command and “TEST active.”
3. Modify the command, press PF10 or Enter, and the object gets modified.

Object picking:

1. Enter, with an empty command field:

```
test
```

2. OPS prompts you:

Pick object to enter TEST mode with picked object.

3. To pick an object, touch any line or filled area in the object with the graphics cursor or the alphanumeric cursor.

OPS retrieves the last command associated with the graphics segment you have picked. If several commands have drawn in the same spot, you will only be able to retrieve the last command in this way. Use “LAST ALL” to retrieve any other.

Any graphics object, like a box or a GDF, will retrieve the command you expect, BOX or GDF.

4. Modify the command directly and press PF10 or Enter.
5. The picked object is then erased and the new object appears.

Retrieving the START command: Text positioned with START retrieves the START command. Although you cannot TEST the START command, you can substitute it by a graphics command with similar syntax (POINT) to get an idea of

what the new START position should be. For example, if you pick text which retrieves:

```
start 10 21
```

overtyping this to:

```
point 10 21
```

and press PF10 or Enter. This removes the text and displays a point at (10,21). Use TEST to move the point to another position, say:

```
point 15 26
```

Overtyping POINT to START, and continue as described in “Adding commands to your presentation” on page 30.

Copying and moving objects

You can do anything you want to an object using UNDO and TEST as already described. However, for copying or moving objects, OPS offers alternatives involving only your mouse or cursor. Here, we describe the process for **copying** an object. The process for **moving** an object is the same, as you can see if you try the exercise described in “Moving objects” on page 33. The difference between the two is, of course, that copying an object leaves you with two copies of the object; moving an object leaves you with just the original object, but in a different place.

To copy an object, enter:

```
copy
```

OPS prompts you to pick an object, using your mouse or cursor. The point you pick is marked by a green cross, or, if your terminal supports segment echoing, your cursor changes into a copy of the object.

Point at the new location of the object (the green cross if used), and click any mouse button or press Enter. A copy of the object appears at the new location, and the associated command appears in the command field.

The command created by the COPY operation has coordinates updated by an amount corresponding to the offset you requested interactively. The offset is rounded according to the current GRAIN setting. However, the coordinates are **not** rounded, unless they already were. This ensures that your objects keep their original shape. For example, assuming GRAIN to be 1, copying:

```
poly 10 10, 34.5 43.4, 70 5.6
```

to a new position offset interactively by (3.6, 5.2) gives:

```
poly 14 15, 38.5 48.4, 74 10.6
```

which has the same shape as the original object.

Note: If you try to use these commands while in TEST mode, OPS switches TEST mode off. When you use the commands, you’re working with more than one object; TEST mode is for testing one object at a time.

Deleting objects

DELETE is used just as COPY and MOVE. Enter:

```
delete
```

and you are prompted to pick an object. When you have done so, the picked object is deleted from the screen. The command behind the object is left in the command field for reworking.

Adding commands to your presentation

Once you are satisfied with a graphics command, you can put it into the presentation file, using the PUT command, followed by the ADMGET command.

Figure 6 summarizes the process of creating and saving graphics commands.

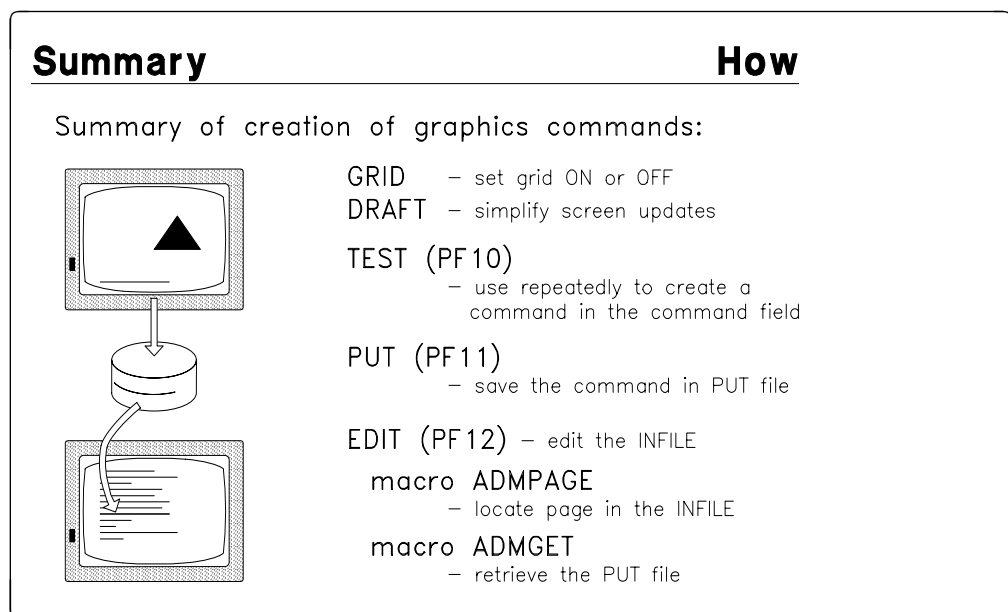


Figure 6. Creating and saving graphics commands

Creating a put file

The argument for PUT is the command text you want to transfer to the infile. For example:

```
put box 50 47 30 20
```

PUT creates the infile command by inserting two blanks, the current command prefix, and a single blank in front of the argument for PUT. For the above example:

```
)ops box 50 47 30 20
```

The infile command is then added to a **put file**. Under CMS this is a file named ADMPUT DATA; under TSO it is a temporary ISPF table named ADM7PUTA.

To save time, PUT as many commands as possible between each edit.

By default PUT is assigned to PF11.

Copying the put file into the infile

To get the command into the presentation (the infile), enter the EDIT command (PF12), and move the contents of the put file into a suitable place in the presentation. You can use the edit macro ADMPAGE to find the right page in the infile.

Use ADMGET to transfer the put file to the infile:

```
==> admget
```

together with a line command giving the destination.

If you use PDF, write ADMGET in the command field, and “a” in the number field in the line after which you want the OPS commands retrieved.

If you use XEDIT, specify the destination either with a slash (/) or an f in the prefix area, or by pointing with the cursor at the line after which you want the put file retrieved.

The ADMGET macro erases the put file after transfer.

Changing your infile

If you’re in the middle of developing a presentation, and you want to change to another presentation without leaving OPS, use the INFILE command to switch to another infile. Type INFILE followed by the name of an ADMOPS file:

```
INFILE SAMPLE
```

Selecting a new infile effectively restarts OPS, so you’re not nesting presentations. When you press PF3 on the presentation you’ve switched to, you leave OPS completely; you don’t go back to the original presentation.

If you want to change the infile, but aren’t sure which one you want, you can get a list of ADMOPS files. For example:

```
INFILE MY*
```

displays a list of all ADMOPS files with names beginning MY. The selection list looks like other selection lists in OPS, but without a graphics field. When you position the cursor on a name and press Enter, you restart OPS with the new infile.

Leaving OPS

When you’ve finished working with OPS, press PF3 (or type END on the command line and press Enter) on any OPS presentation screen. OPS displays the message:

```
Press PF3 if you want to leave GDDM-OPS
```

Press PF3 to close the presentation and leave OPS.

Exercise: adding and moving objects

Having gone through the processes for manipulating objects, let's see them in action.

Adding objects

Start OPS against an empty file, such as NEWFILE ADMOPS:

```
admops!a newfile
```

Answer "1" (yes) to the "new file" prompt, and "0" (no) to the prompt offering you the sample.

Enter the following commands in the command field in OPS and press the keys as noted.

Action	Result
box	Press Enter. Point at the center and at a corner as requested by OPS.
box 40 40 40 20	Press Enter. The box moves.
box 40 40 30 20	Press Enter. The box gets smaller.
	Add whatever operands you like, using PF1 to get help. Keep using Enter. When you're satisfied...
Press PF11	OPS responds: Command field added
Press PF12	Start edit.

Move the put file into the infile as described in "Adding commands to your presentation" on page 30, and return to OPS with PF3.

What is displayed is what you saved in NEWFILE ADMOPS.

Add another object, for example a circle, starting with:

```
circle fill g 3
```

Keep pressing Enter until satisfied. You can add ? after DRAW, FILL, and WFILL to get graphics overviews:

```
circle 21 43 30 draw ? fill ?
```

Finally press PF11 (PUT) and then PF12 (EDIT).

Figure 7 shows the result of the exercise, where the two final commands were:

```
)ops box 25 40 30 20 draw 3 2 shadow
)ops circle 70 30 25 fill g 3
```

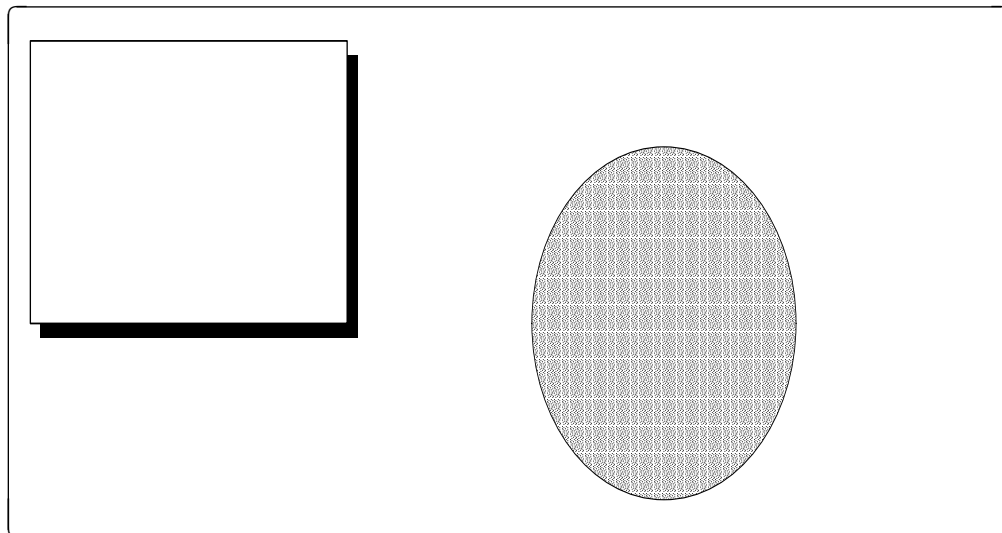



Figure 7. Adding objects: results of the exercise

If you happen to drop out of the TEST cycle at any time:

1. Press PF6 (LASTcmd) to retrieve the last command.
2. Press PF9 (UNDO) to erase the object.
3. Press PF10 (TEST) or Enter again.

This takes you back to the interactive process.

Moving objects

Using the result of the above exercise (two objects on the screen), move one or both objects.

1. Press PF2 (REFresh) to ensure that you are on the right page and that the commands have been entered into the infile.
2. Type MOVE, and pick the box, by pointing with the cursor and pressing Enter. OPS asks you to point to the new location you want. Do so, and press Enter.
3. The BOX command should appear in the command field.
4. Overtyping the center coordinates (the first two numbers), either with new values or with getc:


```
box getc 30 20 r 5, w 2, fill t 4, wf 1
```

 (Everything after getc is just sample data.)
5. Press Enter and you're back in the process of positioning an object.
6. When satisfied, press PF11 (PUT) and PF12 (EDIT). Transfer the put file into the infile, and remember to erase the old BOX command.

If you want more practice, try **copying** the circle.

Creating presentations using EXECs or CLISTs

An OPS presentation is just an infile with commands, so you can use an EXEC (or CLIST) to generate a presentation and show it immediately by invoking OPS.

The following sample is a REXX EXEC for CMS, but the principle applies equally to TSO. The EXEC creates an OPS file, and displays it, as shown in Figure 8.



Figure 8. OPS presentation from PAUSE EXEC

The EXEC looks like this:

```

/*REXX*****
 * This REXX EXEC creates and displays an OPS presentation *
 * Syntax:                                                    *
 *      PAUSE hhmm                                           *
*****/
trace off; address command
Parse arg hhmm .
if hhmm = '' then do
  Say 'Syntax is: PAUSE hhmm'
  exit 8
end
'ERASE $PAUSE ADMOPS A (NOTYPE'

queue "1 )ops dark; prefix %"
queue " % backgr whi 2; ss admuuksf; h 12"
queue " % force"
queue " % text 'Intermission' 49 44 ce col n"
queue " % text 'Intermission' 50 45 ce col r"
queue " % force"
queue " % text 'until' 49 29 ce col n"
queue " % text 'until' 50 30 ce col r"
queue " % force"
queue " % text '"hhmm"' 49 14 ce col n"
queue " % text '"hhmm"' 50 15 ce col r"
queue "1"
queue ''

'EXECIO * DISKW $PAUSE ADMOPS A (FINIS'
'EXEC ADMOPSLA $PAUSE'
Return

```

Figure 9. REXX EXEC to create and invoke OPS presentation

using EXECs or CLISTs

Chapter 5. OPS commands

In the last chapter, we showed some of the common commands you use when creating presentations. Now, we'll look at the structure of OPS commands in more detail.

OPS is controlled by commands, received either from the command field or from the file you are browsing (the infile). Commands from the command field are called **direct commands**; commands in the infile are called **infile commands**.

Most commands can be used either as direct commands or as infile commands, but a few can be used as direct commands only.

This chapter tells you how to use commands, and gives the rules governing their use. You'll find guidance on using individual commands in the relevant chapter in this part of the book. You'll find a complete list of OPS commands, with a brief description, and a pointer to where you can find more information in this book, in "OPS command list" on page 44.

Command structure

A command consists of a **command word**, telling OPS what to do, followed by one or more **operands**; for example:

```
wait
draw red
arrow 20 20 40 30 double, draw red, head 2
text "hello there" size 5mm
```

A command can include up to 150 characters.

Command words and operands can be entered in any mixture of uppercase and lowercase. (In this book, we usually show commands in the text in uppercase, and commands in examples in lowercase.)

You can specify several commands at a time by separating them with a **command separator character**. The default is a semicolon (;), but you can change this with the SEPCH command. For example, to designate a slash (/) as separator, enter:

```
sepch /
```

Note: The separator character is reset to semicolon when you scroll to the top of the file. This lets you use SEPCH as an infile command.

Command words

Command words have a maximum of 8 characters. Many command words have an abbreviated form; some also have an alternative form. You can abbreviate the command word COLOR to COL; you can specify the command word MESSAGE as MSG. For operands, you need only enter the minimum valid abbreviation. (In the examples, we usually use the full command words to make their significance clearer.)

Operands

Operands can be numbers, arithmetic expressions, special characters, text strings with or without quotes, or special words known to OPS.

Numbers can be integers or floating-point (numbers with a decimal point).

When you specify coordinate values (position on the screen), dimensions or angles, you can mix floating-point and integers as you like. However, when you specify a number indicating some choice, for example a color number, you must supply an integer.

Types of operands

There are four kinds of operands:

- Positional operands
- Keywords with value
- Keywords
- Subcommands

Positional operands: Positional operands are known to OPS by their position alone. They are always the first (and perhaps the only) operands on a command. For example, in the command:

```
arrow 20 20 40 30 double, draw red, head 2
```

the first four numbers are positional operands; their significance is given by their positions.

Keywords with value: These consist of a keyword followed by a value which OPS is to use. In the ARROW command above, head 2 tells OPS about the head of the arrow.

Keywords: These are operands with no associated value. In the ARROW example, double defines an arrow with a head at each end.

Subcommands: These are OPS commands used as operands on other OPS commands. In the ARROW command, the OPS DRAW command is used as a subcommand (draw red).

Rules for using operands

Operands (except positional operands) can appear in any order, separated by blanks or commas. You can use commas to make the wording clearer. The following examples are all valid versions of the same command:

```
arrow 20 20 40 30 double, draw red, head 2
arrow 20 20 40 30 head 2 double draw red
arrow 20 20 40 30 h 2 d dr r
```

In the last example, the operands have been abbreviated as much as possible.

Arithmetic expressions in operands

In general, you can use arithmetic expressions anywhere outside quoted strings and infile text, giving you a useful tool for:

- Coordinate manipulation
- Selection
- Computation

You can use the four basic arithmetic operators (+, -, *, /) and parentheses, mixed and nested to any depth, within an expression. The priority of the operators is:

1. parenthesis
2. * and /
3. + and -

Thus:

Expression	Result
2+3*4	14
(2+3)*4	20

Note that division always yields a floating-point value as result, so the expression:

```
color 8/2
```

is invalid because 8/2 is a floating-point number, and COLOR expects an integer operand.

Infile commands

To enable OPS to recognize an infile command, the line (record) must start with the **command prefix**. The default prefix is:

```
)ops
```

Following the command prefix, OPS recognizes the rest of the line as an OPS command. For example, to make the color red, include the following line in the infile:

```
)ops color red
```

A sample of an infile might look like this:

```
This infile to OPS contains mainly free text like
this line, and several others.
But the next line is a command line to OPS.
)ops color yellow
This line is just more text. When the file is viewed with
OPS, this section will appear in yellow.
)ops color red
And this line will be red
```

The command prefix, like command words and keywords, can be entered in any mixture of uppercase and lowercase.

There are two rules controlling where you can put your infile commands:

1. You cannot place the command prefix in column 1 if the file has any control characters in this column (it usually has).
2. You cannot place the command prefix in column 2 if the file has a table reference code in this column.

Changing the command prefix

You can define an alternative command prefix with the PREFIX command, which can be used both as a direct command and as an infile command. For example, to set the prefix to “%,” type:

```
prefix %           (direct command)
)ops prefix %     (infile command)
```

If the prefix is longer than a single character, put it in quotes; for example:

```
prefix ')ops'
```

Notes:

1.)OPS is **always** a valid prefix, even when you define an alternative.
2. Be careful when you change the prefix. After you set the alternative, it will be treated as the command prefix whenever it appears at the start of a line.

Comments

You can enter comments, in both infile and (less usefully) in direct commands, by:

1. Putting an * in a position where OPS expects a command
2. Enclosing part of an OPS command in /* */

So, assuming)ops as the command word, infile comments look like this:

```
)ops * this is a comment
)ops *pseg 3820 p1200 120mm
)ops arrow 20 20 30 40 /* head 2 */ double
```

The first two examples will be fully ignored; in the third, the arrow will be drawn with the default head (rather than head 2).

Defining keyboard keys as commands

You can make your work easier by assigning the PF keys and the Enter key to the commands you use most frequently.

Assigning the PF keys to commands

Use the PFK command to assign commands to PF keys 1-12. The following command:

```
pfk 11 down p
```

sets PF11 to the command DOWN P.

You can define several PF keys with the same command by putting single quotes around the values:

```
pf 1 '?' 2 'REFR' 11 'SS VSS'
```

The example also shows that PFK can be abbreviated to PF.

Note: When you execute a command by pressing a PF key, OPS puts that command at the **start** of the command line. If you assign a command that takes parameters to a PF key, make sure that the command line is empty before you press the PF key, or OPS will interpret any text it finds as extra parameters.

Use CLEAR to delete all previous definitions:

```
pf clear 3 'END'
```

You can temporarily display the current values of the PF keys in a window by entering either of the following commands:

```
query pfk
q pf
```

For more information about the PFK command, see Chapter 6, “Controlling OPS display” on page 53.

Assigning the Enter key to commands

Use the ENTER command to assign a command to the Enter key. For example:

```
enter down p
```

sets Enter to “scroll down a page.”

When defining Enter, remember that the function is executed **only** when the command field is empty; otherwise it would be impossible to execute commands.

You can display the definition of the Enter key by typing:

```
query enter
```

Recalling commands

You can recall the last 100 commands, either one at a time or all 100 in a list, using the LASTCMD command.

Recalling the last command

You recall only the last command by entering LASTCMD without any argument:

```
lastcmd
last
```

This recalls the last command to the command field, where you can use it again by pressing Enter.

The command which is recalled is the command actually **executed**: the total of any PF function and the text in the command field. For example, if you've defined PF8 as `down` and you enter:

```
12; color r
```

and press PF8, LASTCMD recalls:

```
color r
down 12
```

LASTCMD recalls infile commands as well as direct commands, and remembers all commands, except LASTCMD itself and PUT.

Recalling the last 100 commands

You can get a list of the last 100 commands by adding the keyword ALL:

```
lastcmd all
last all
```

This displays a panel listing the commands, on which you can select a specific command by moving the cursor to it, and pressing Enter or PF2. This takes you back to the main panel with the selected command ready in the command field.

The functions of PF10 and PF11 (TEST and PUT) are described in "How to create presentations" on page 22.

Error messages

Error messages for direct commands are shown in the message field on the main panel. The incorrect command will remain in the command field.

Error messages for infile commands are shown on a special panel which also displays the line in the infile causing the error.

Most messages are self-explanatory. If you need help, however, you can:

- Press PF1 for a command syntax error in a direct command. This displays part of the OPS help file, as explained in Chapter 17, "Getting help" on page 195.

- Look in Appendix D, “Messages” on page 219, where more information is provided on messages. To find the message, you need to know the message number. You can get that by using the QUERY command:

```
query msg
q msg
```

which redisplay the last message along with its message number.

Where to find information on commands

This book gives you guidance information about the OPS commands. In Figure 10 on page 44, you’ll find a list, in alphabetic order, of all the OPS commands, giving a brief description of each, and showing you where you can find more information in this book.

Most commands can be used both as **infile** and as **direct** commands. If a command can be used in only one of these contexts, we say so in the description.

Some commands apply only to CMS or TSO; again, this is indicated in the description.

Many commands can be abbreviated. In Figure 10 on page 44, we show the minimum abbreviation, where applicable, immediately beneath the full command name.

For complete reference information for every command, use the online help, available in many ways:

- Enter a command name and press PF1 to display a syntax sample in the command field. Just remove the leading * to execute it as a command.
- Press PF1 again immediately to display the help file, positioned at the explanation of that command. (Executing HELP command does the same.)
- Press PF1 without an argument to display a “help window.” (Executing HELP on its own does the same.)
- Put ? to the right of any of the commands CHARTX, ICU, GDFX, IMAGE, and SS, to display a scrollable listing of the objects available. You can then get more help by pressing PF1.
- Put ? after any of the commands ARROW, COLor, DRaw, FIll, and WFill, to display the available options.

For more information on using the help file, see “Displaying the OPS help file” on page 195.

OPS command list

Figure 10 (Page 1 of 8). OPS command list

Command	Description	For more information see ...
ADJUST	Align text left, right, or center.	"Aligning text" on page 81
ALARM	Sound the alarm, or set it on or off.	"Sounding the alarm" on page 154
ARC	Draw a circular arc.	"Drawing arcs" on page 124
AREA	Infile only Begin or end a shaded area defined by a series of line art objects.	"Grouping objects together" on page 128
ARROW	Draw and, optionally, fill an arrow.	"Drawing arrows" on page 115
ASPECT	Specify the aspect ratio of graphics fields.	"Controlling the aspect ratio of the graphics field" on page 58
AUTO	Let users browse through files without interaction.	"Automatic scrolling" on page 181
BACKGR	Add or remove the background color.	"Filling the background" on page 133
BIND	Adjust the text to the left.	"Sideways scrolling" on page 178 (infile). Appendix A, "OPS startup procedures" on page 201 (setting the bind value at call time)
BOTTOM BOT	Scroll to the last page in a presentation.	"Using scroll commands" on page 177
BOX	Draw and, optionally fill, a box.	"Drawing boxes" on page 119
BREAK BR	Set the start position for the next text at the next line and at the left margin.	"Forcing a line break" on page 93
BROWSE	Browse an external file.	"Browsing files" on page 171
CENTER CE	Center text.	"Centering text" on page 81
CHART	Add charts created by the Interactive Chart Utility (ICU) to presentations.	"Adding an ICU chart to the current picture" on page 135
CHARTX	Display an ICU chart on a clear or dialed screen.	"Displaying a chart on a clear screen" on page 137
CIRCLE CIRC	Draw, and optionally fill, a circle or ellipse.	"Drawing circles" on page 123
CLEAR CL	Clear the primary screen and any dialed screens.	"Clearing the graphics field" on page 59
CMDFLD	Specify the color and prompting attributes of the command field.	"Controlling the command field" on page 56
CMS	CMS & direct only Execute a CMS subset command, or enter CMS subset.	"Executing CMS commands" on page 174

Figure 10 (Page 2 of 8). OPS command list

Command	Description	For more information see ...
COLATTR	Specify color attribute characters for text.	"Coloring individual words and characters" on page 84
COLOR COL	Set from one to four text colors, corresponding to table reference code (TRC) values 0 (or no TRC) to 3.	"Coloring text" on page 69
COPY	Direct only Copy objects using a mouse or cursor.	"Copying and moving objects" on page 29
CSP	Set character interspacing.	"Controlling the space between characters" on page 74
CURVE CU	Draws an elliptical curved line.	"Drawing lines" on page 111
DARK	Set dark display mode, in which OPS runs without a visible frame or command field.	"Using dark mode" on page 54
DASD	Draw, and optionally fill, a DASD symbol.	"Drawing DASD symbols" on page 126
DASDF	Draw, and optionally fill, a file in a DASD symbol.	"Drawing DASD symbols" on page 126
DASDX	Draw a file separator line in a DASD symbol.	"Drawing DASD symbols" on page 126
DEFAULTS DEFLT	Set various non-standard defaults, including the symbol set and colors.	"Setting defaults for text" on page 73 and "Rules for using RESET and DEFAULTS" on page 21
DELETE DEL	Direct only Delete an object selected by the mouse or cursor. The command corresponding to the deleted object is left in the command field, ready for you to change.	"Deleting objects" on page 30
DOWN D	Scroll the infile down a specified number of pages or lines.	"Using scroll commands" on page 177
DRAFT	Set draft (GDDM update) mode. Use draft mode to develop presentations.	"Working in draft mode" on page 27
DRAW DR	Set the line drawing attribute for commands such as BOX and CIRCLE.	"Setting line attributes" on page 104
EDIT E	Direct only Edit the infile or, if the file is a formatted list (scripted file), the source text. Under CMS, XEDIT is used; under TSO, PDF is used.	"Editing files" on page 171
ELSE	Execute a command if the most recent IF-test proves false.	"Flow control commands" on page 157
END	Leave OPS.	"Leaving OPS" on page 31

command list

Figure 10 (Page 3 of 8). OPS command list

Command	Description	For more information see ...
ENTER	Assign a function to the Enter key.	"Assigning the Enter key to commands" on page 41
EXEC	Invoke an external EXEC; for security reasons, this must have the same name as the infile.	"Executing functions" on page 171
FILL FI	Set the fill attributes for commands such as BOX, CIRCLE, and DASD.	"Setting area attributes" on page 105
FIND F	Direct only Search the infile, forward from the bottom of the current screen, for a specified string.	"Searching the infile" on page 180
FOOT	Write a standard foil footer on the screen. The text is written at position y=0, and OPS draws a horizontal line at y=1t (just above the character box).	"Creating headers and footers" on page 93
FORCE	Force part of the page to the screen.	"Forcing the graphics field to the screen" on page 152
FRAME	Reset or remove the frame around the graphics field	"Drawing frames" on page 134
GDF	Add a GDF file to a picture.	"Displaying GDFs" on page 140
GDFEDIT GDFE	Direct only Enter GDF editing mode, in which you can manipulate graphics segments on the current screen, without generating any OPS source.	"Editing GDFs without the OPS source" on page 145
GDFSAVE	Save the current screen as a GDF file.	"Saving the graphics field as a GDF" on page 144
GDFX	Display a GDF file on a clear or dialed screen.	"Displaying GDFs on a clear screen" on page 144
GOTO	Transfer control to a specified label, line, or page.	"Flow control commands" on page 157
GRAIN	Set the accuracy of the subcommand GETC.	"Controlling the accuracy of cursor reading" on page 25
GRID GR	Display the coordinate system, a grid of lines with a separation of 1/10 of the screen width.	"Displaying the coordinate system" on page 24
HEAD	Write a standard foil header on the screen at position &\$ymax-1t, with a horizontal line just below the character box.	"Creating headers and footers" on page 93
HEIGHT H	Set the character box height.	"Controlling the character box" on page 67

Figure 10 (Page 4 of 8). OPS command list

Command	Description	For more information see ...
HELP	Display OPS help. The alias for the HELP command is ?, and PF1 is set to HELP.	"Getting command syntax help" on page 196
HELPMODE HELPM	Set the help mode, enabling you to bypass the top-level help.	"Bypassing the help menu and command syntax help" on page 197
ICU	Direct only Invoke the Interactive Chart Utility.	"Using the Interactive Chart Utility" on page 172
IF	Evaluate a logical expression, and set the IF-ELSE flag.	"Flow control commands" on page 157
IMAGE	Add an image, in ADMIMG or PSEG38xx format, to a picture.	"Basic image display" on page 147
IMGLIB	Specify non-standard image filetypes for use with IMAGE.	"Basic image display" on page 147
IN	Set the left margin for text flow.	"Indenting text" on page 80
INFILE	Direct only Allocate a new infile and restart OPS.	"Changing your infile" on page 31
ISE	Direct only Invoke the GDDM Image Symbol Editor.	"Using symbol editors" on page 173
ISPEXEC	TSO only Get (or put) variables from (or to) the ISPF shared pool.	"Sharing variables with ISPF" on page 168
ISPF	TSO & direct only Invoke ISPF functions.	"Using ISPF functions" on page 175
LABEL	Define a label for use with "positioning" commands, such as GOTO.	"Using labels" on page 158
LASTCMD LAST	Direct only Display up to the last 100 commands executed.	"Recalling commands" on page 42
LINE	Draw a broken line between the points specified.	"Drawing lines" on page 111
LOCATE L	Find a record, page, or label, or scroll backward or forward a specified number of records or pages.	"Using labels" on page 158
LSP	Set up to four values for leading space.	"Controlling the space between lines" on page 76
MARK	Draw one or more markers in text.	"Writing markers on the screen" on page 101
MIX	Set the color mixing.	"Controlling color mixing" on page 109

command list

<i>Figure 10 (Page 5 of 8). OPS command list</i>		
Command	Description	For more information see ...
MODIFY MOD	Query an option or value, and put the result in the command field.	"Querying the value of system variables" on page 199
MOVE	Direct only Move objects using a mouse or cursor. The command that corresponds to the object being moved is put into the command field, ready for TEST or PUT.	"Copying and moving objects" on page 29
MOVIE	Force a picture to the screen item by item.	"Creating animated sequences" on page 153
MSG	Display a message in the message field, or set message mode to on or off, or short or long.	"Displaying messages in dynamic sequences" on page 153
MSGFLD	Set message field attributes.	"Controlling the message field" on page 56
NOP N	Specify "no operation." Use it to break an automatic TEST cycle.	"Testing commands" on page 27
NORMAL NORM	Establish normal display mode: the "other side" of the DARK command.	"Selecting the mode" on page 53
OFFSET O	Set the offset of line art, text, and pictures.	"Adjusting the coordinate system" on page 64
OGRAIN	Set the precision of floating-point presentation.	"Interpreting numeric values" on page 164
PAGESET PSET	Set the internal page count to a defined value.	"Synchronizing the page number" on page 178
PFK PF	Define and display the current PF key setup.	"Assigning the PF keys to commands" on page 41 and "Controlling PF keys" on page 57
PLOT	Direct only Output graphics to a directly-connected plotter.	"Plotting presentations" on page 185
POINT	Put one or more dots (points) on the screen, with the attributes of the points set using DRAW.	"Setting points on the screen" on page 110
POLYGON POLY	Draw, and optionally fill, a polygon (a closed, multi-cornered area).	"Drawing polygons" on page 114
PREFIX PREF	Define the prefix for infile commands. The default is)ops, which is always a valid infile command prefix.	"Knowing where you are at the start of presentation pages" on page 21
PRINT PR	Direct only Save a screen image in ADMPRINT format.	"Printing panels" on page 183

Figure 10 (Page 6 of 8). OPS command list

Command	Description	For more information see ...
PSEG	Create a page segment to include in a DCF document.	"Creating page segments (PSEGs)" on page 186
PSEGLIB	Set the default filetype and filemode for PSEGs (CMS), or the name of the partitioned data set in which to store page segments (TSO).	"Controlling output under CMS" on page 186 and "Controlling output under TSO" on page 187
PUT	Direct only Put (append) a specified string to the OPS put file, for later retrieval while editing.	"Adding commands to your presentation" on page 30
QUERY Q	Direct only Query the status of OPS system and user variables.	"Querying variables" on page 169 and "Querying the value of system variables" on page 199
RAYS	Draw rays coming from a defined point (a "radiant star").	"Drawing rays" on page 125
READCMD	Read input from the command field into the variables &\$RCMD and &\$RCMDU in string format.	"Reading the command field" on page 161
READTAG	Read the tag value of an object by pointing at the object with the mouse or cursor, and store the tag in variable &\$RTAG.	"Getting tag values" on page 160
REFRESH REFR	Refresh the current screen using the current option values, or set automatic refresh on or off.	"Refreshing the page" on page 179
RELEASE REL	Release loaded symbol or pattern sets.	"Releasing symbol sets" on page 72
RESET	Reset selected system values to default settings.	"Rules for using RESET and DEFAULTS" on page 21 and "Setting defaults for text" on page 73
RFind RF	Direct only Repeat the most recent FIND command, or locate a specified string, searching forward from the bottom of the screen.	"Searching the infile" on page 180
SCOPY	Copy the current screen contents to a dialed screen.	"Copying the graphics field to a secondary terminal" on page 193
SECTOR	Draw, and optionally fill, a sector of a circle.	"Drawing pie slices" on page 125
SEPCH	Specify the OPS command separation character as something other than the default, semicolon (;).	"Command structure" on page 37

command list

Figure 10 (Page 7 of 8). OPS command list

Command	Description	For more information see ...
SET	Assign a value to a user variable. You can get a display of the values of all user variables using Q VARS.	"Defining and retrieving variables" on page 163
SHEAR SH	Set the shear angle of text.	"Italicizing text" on page 78
SIZE	Set the character box size for up to four symbol sets.	"Controlling the character box" on page 67
SPACE SP	Insert space before writing the next text line.	"Inserting space between text lines" on page 79
SS	Select between one and four symbol sets.	"Selecting symbol sets" on page 65
START	Set the starting position for the next text line or TEXT command.	"Positioning text exactly" on page 82.
TABCH	Specify a tab character other than the default (-).	"Arranging text in columns" on page 83
TABS	Set one or more tab positions. Having set up the tab positions, you jump to (consecutive) tab positions using the tab character (see TABCH above).	"Arranging text in columns" on page 83
TAG	Set the tag attribute of the text or objects that follow it.	"Tagging objects" on page 159
TEST	Direct only Execute (test) a command.	"Testing commands" on page 27
TEXT	Write text on the screen.	"Manipulating text strings" on page 86
TF	Set text flow on and off. Text flow means automatic text cutoff at the right margin, and continuation on a new line, properly adjusted.	"Starting text formatting" on page 95
TFEOL	End ordered lists in text flow.	"Creating ordered and unordered lists" on page 97
TFEUL	End unordered lists in text flow.	"Creating ordered and unordered lists" on page 97
TFEXMP	End examples in text flow.	"Temporarily suspending text formatting" on page 98
TFH	Write a text flow header.	"Writing text in paragraphs" on page 96
TFLI	Write a text flow list item within a list started by TFOL or TFUL.	"Creating ordered and unordered lists" on page 97
TFOL	Start an ordered list in text flow.	"Creating ordered and unordered lists" on page 97
TFP	Write a paragraph in text flow.	"Writing text in paragraphs" on page 96
TFUL	Start an unordered list in text flow.	"Creating ordered and unordered lists" on page 97

Figure 10 (Page 8 of 8). OPS command list

Command	Description	For more information see ...
TFXMP	Start an example in text flow.	"Temporarily suspending text formatting" on page 98
THEN	Conditionally execute a command, depending on the result of an IF command	"Flow control commands" on page 157
TI	Translate text on input.	"Using translation tables" on page 71
TOP	Scroll to the top of the file.	"Using scroll commands" on page 177
TRC	Define whether you are using table reference codes (TRCs), which appear in column 2 of the file.	"Selecting symbol sets with TRC codes" on page 66 and "Using TRCs to select symbol sets" on page 99
TRTDEF	Define or modify a translation table for text.	"Using translation tables" on page 71
TRTDROP	Stop using a translation table.	"Using translation tables" on page 71
TRTUSE	Define a translation table to use, either globally, or only for specific symbol sets.	"Named translation tables" on page 71
TSO	TSO & direct only Execute a supplied command for TSO.	"Executing TSO commands" on page 174
UNDO	Delete an object from the screen. You can delete text, line art, and most GDFs. You cannot delete ICU charts.	"Deleting objects" on page 26
UP	Scroll up (backward) a specified number of pages or lines.	"Using scroll commands" on page 177
USERCTL	Invoke GDDM user control mode, which lets you zoom into and out of the picture.	"Invoking GDDM user control mode" on page 60
VARSUBST VAR	Set variable substitution on and off.	"Defining and retrieving variables" on page 163
VSE	Direct only Invoke the GDDM Vector Symbol Editor.	"Using symbol editors" on page 173
WAIT	Suspend processing for a specified number of seconds, or until an event.	"Inserting pauses in presentations" on page 151
WFILL WF	Specify special filling of wide lines. Default filling is by attributes on the DRAW command.	"Drawing wide lines" on page 110
Asterisk (*)	Enter a comment.	"Comments" on page 40

command list

Chapter 6. Controlling OPS display

This chapter describes the features you can use to control the way that OPS displays presentations on the screen:

- The two display modes: NORMAL mode for browsing and DARK mode for presentations
- Controlling the display of alphanumeric fields, such as the command and message field
- Controlling the aspect ratio of the graphics field
- Using GDDM's user control mode

Finally, this chapter describes the coordinate system, used by many OPS commands to position text and graphics on the screen.

Selecting the mode

In both normal and dark mode, two lines of the panel are used for the command field and the message field, while the rest is intended for a variable graphics field. Additional alphanumeric fields may overlay the graphics field, but the graphics field can never be bigger than the size of the panel less two lines.

Select the mode using the NORMAL and DARK commands; for example:

```
normal
```

At startup, normal is the default mode, but you can modify this using a DARK command in the infile, or the DARK operand on the ADMOPSLx EXEC.

You can change from one mode to the other at any time. If you change mode with a direct command, the page is shown again (using a REFRESH command). If the page contains a contrary mode command as an infile command, the change will not be executed. You usually put a DARK command at the beginning of a presentation; in this case, you cannot enter NORMAL when the first page is being shown.

A mode command in the infile changes to the specified display mode and then continues. The page is *not* shown again, so you must place a mode command before displaying any text or commands.

Using normal mode

In normal mode, OPS looks like PDF BROWSE. Display any of your presentations in normal mode, and you'll see that the panel consists of three fixed fields:

1. The message field, consisting of Title and Page/Line/Message
2. The command field
3. The graphics field, optionally overlaid by PF keys

The message field

The title shows OPS/symbol set name - file name, where symbol set name is the name of the current symbol set, and file name specifies the name of the file you are viewing. The color is, by default, yellow.

The **Page/Line/Message** line usually shows the internal page number. You can synchronize this with a current document page number, using the PSET command. Error or information messages are also displayed in this field; they will overlap any other text in the line. Finally, the field can, if you want, permanently show the definition of the PF keys, overlapping the title. The color is, by default, yellow.

The command field

This is where you enter direct OPS commands.

The graphics field

This occupies the rest of the panel. You can fix the exact proportions of the field using the ASPECT command, described in "Controlling the aspect ratio of the graphics field" on page 58. When data is displayed with a graphics symbol set, the field is framed. If the symbol set is the hardware set or a replacement of this put in a PS store, the frame consists of a single line only.

At the bottom of the graphics field, you can optionally display the function of the PF keys, overlaying the graphics field, Graphics information behind this field is then invisible (OPS usually avoids writing text with the hardware symbol set in this line).

Using dark mode

A panel in dark mode contains only three fields: the message line at the top, the graphics filling the screen, and the command line at the bottom. This is shown in Figure 11.

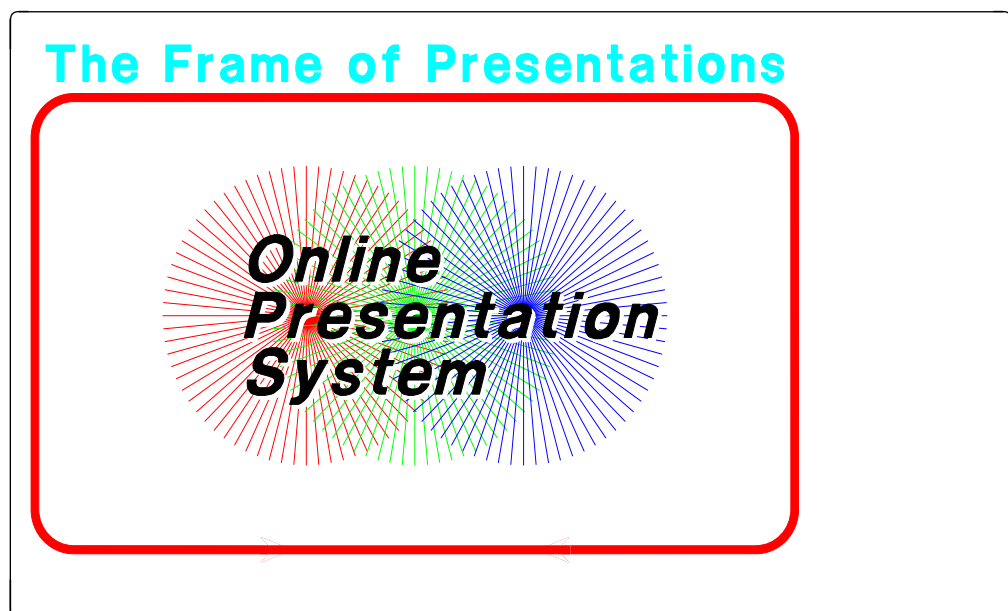


Figure 11. Panel layout in dark mode

Note that you see no fixed information such as title, page number or arrow in the command field. However, you can display the arrow with the CMDFLD command.

If you want information, about a page number for instance, use the QUERY command. You can display the function of the PF keys permanently in the message field using the PFK command.

The color of messages is by default blue. OPS suppresses all messages except error messages.

The graphics field has the same size as in normal mode, the size of the entire panel less two lines. You can set the exact proportions of the field using the ASPECT command described in "Controlling the aspect ratio of the graphics field" on page 58.

Use dark mode for presentations on video monitors or video projectors, and for user-controlled presentations of online information. In the first case, you leave the message field empty; in the second, you can show instructions to the user, for instance the function of the necessary PF keys in the message field. In both cases, you seldom need to enter commands.

Controlling alphanumeric fields

You can set the colors and highlighting of the different alphanumeric fields.

NORMAL and DARK always set the default colors (yellow in normal mode and blue in dark mode), so you must make any changes *after* a mode command.

The fields are controlled by the following commands: MSGFLD, CMDFLD, and PFK. The color is set for all three using the subcommand COLOR, as in:

```
msgfld color red
```

You can use COL as an abbreviated form of COLOR.

You can specify the color value either by a number between 1 and 7, corresponding to the color codes in GDDM, or by the English name of one of the base colors. You can abbreviate these names (only the first letter is important):

- 1 Blue
- 2 Red
- 3 Pink
- 4 Green
- 5 Turquoise
- 6 Yellow
- 7 White

This is the authorized way to use COLOR in all contexts. However, with alphanumeric fields *only*, you can also specify colors as a (minimum) 3-character abbreviation of the color name, dropping the word COLOR. Thus the following samples are all valid, and request the same thing:

```
msgfld color red
msgfld col r      ;* with color only 1 character required
msgfld col 2
msgfld red        ;* no less than 3 characters
```

controlling the command field

You set highlighting of alphanumeric fields with one of four keywords, which you can abbreviate to three characters:

```
BLInk  
REVerse  
UNDeRscore  
NORmal (default)
```

Controlling the message field

Use the MSGFLD command to control the color and highlight of the message field. For example, the following gives a red, blinking field:

```
msgfld bli, col r
```

Attributes set by MSGFLD are valid when messages are displayed but not when the PF keys are displayed in the message field.

Controlling the command field

Use CMDFLD to control the command field. Color and highlight apply to the *fixed* prompting information that normally appears in normal mode only. Here's an example:

```
cmdfld color pink
```

You can change or define the fixed prompting information in both normal and dark mode using the keyword PROMPT. Alternatively, you can delete the prompting information using NOPROMPT. Here are two examples:

```
cmdfld prompt '===>' white  
cmdfld noprompt
```

Specify the fixed prompting information in quotation marks. You can abbreviate PROMPT and NOPROMPT to three characters.

If you want to change the color of entered text from the conventional green to another color, use the keyword INPut:

```
cmdfld input pink, col red
```

You cannot control highlighting of the entered text.

Sometimes, in a presentation, you want to suggest a command to users so that they can, for example, test it. You can do this using the CMD keyword with a value, as in:

```
cmdfld cmd 'arrow, width 5mm, curve'  
cmdfld cmd '*pseg 3820 mypseg 120mm'
```

The first example leaves an ARROW command in the command field, ready to use.

The second leaves a comment, which, by removing the *, becomes a ready-to-use PSEG command.

Controlling PF keys

Use the PFK command, introduced in “Defining keyboard keys as commands” on page 41, to control the display of PF keys:

```
pfk on
pf
```

give a permanent display of the functions in the message field. To remove the display, enter:

```
pfk off
```

You can also use PFK as a switch:

```
pfk switch
```

which switches the display on and off, by turns.

By default, OPS creates a suitable message line (maximum 79 characters) showing the functions of PF keys. To make this useful, always use the short command forms when you define the PF keys; for example:

```
pf 2 'REFR' 4 'PR' 5 'F' 6 'LAST'
```

You can specify the contents of the text yourself with the keyword TEXT on the PFK command, as in:

```
pfk text 'PF: 7=Up 8=Down. Return with PF3'
```

where the text string has a maximum of 79 characters. This variant of the PFK command is mainly useful as an infile command: you can set a permanent instruction for the user of a presentation or online information.

If the text is not enclosed in single quotation marks, the entire command line from TEXT is considered as the PFK text. However, the input line is parsed for variables and arithmetic expressions, and therefore broken up in elements separated by blanks. Thus the text is likely to be longer than you planned, and different in appearance.

To stop OPS modifying the existing text when you define PF key functions, use the keyword NOTEXT (NOT):

```
pf 4 'Page 12', 5 'Page 34' notext
```

You can add color and highlight, as in:

```
pf reverse, col pink
```

which also implies PFK ON. OPS remembers the specified color and highlight, together with any user-defined TEXT, from when PFK OFF is executed until you leave OPS. Before that, if you ask for PFK ON, the color of the message field changes to the remembered color.

In normal mode, you can control where the PF key functions are displayed, using HIGH (HI) for the top of the panel, or LOW (LO) for the bottom:

```
pf lo
pf hi
```

controlling aspect ratio

The following example defines the PF keys, sets up a text, and defines Enter to display and not display the PF keys in turn:

```
pf clear 1 '?' 2 'Refresh' 3 'End' 7 'Up' 8 'Down'  
pf text 'PF: 1=? 2=Refresh 3=End 7=Up 8=Down'  
enter pf sw low white, reverse
```

Controlling the aspect ratio of the graphics field

Use the ASPECT command to fix the proportions of the graphics field within a frame defined by the width of the panel (80 columns) and its height less two lines (in most cases 22 or 30 lines).

Here's an example:

```
aspect 2 1
```

This sets the width to 2 and the height to 1, generating a graphics field twice as wide as high. The field is as big as possible; on most screens the width is 80 columns and the height somewhat less than the allowed maximum.

You can request that the graphics field fills the entire available space by:

```
aspect *
```

Don't use this option in a presentation aimed at more than one type of terminal; you'll get different results on different screens. However, you'll find it useful to establish the exact proportions of your terminal session or window.

If you are viewing a file of type ADMOPS, the default ASPECT is 100 67, which matches most terminals. This default ensures that, even if ASPECT is not set explicitly in the presentation, the presentation most probably will look as intended on all screens.

If you are viewing anything else, the default ASPECT is *, ensuring maximum use of the screen.

When you are using OPS to create foils or print, you will often want aspect ratios different from the screen. The following command, for example, generates a graphics field with the same format as vertical A4:

```
aspect 21.0 29.7
```

The height is the allowed maximum; the width is about 38 columns on most screens. In dark mode, it looks like Figure 12 on page 59. (This is an A4 version of the graphic shown in Figure 36 on page 113.)

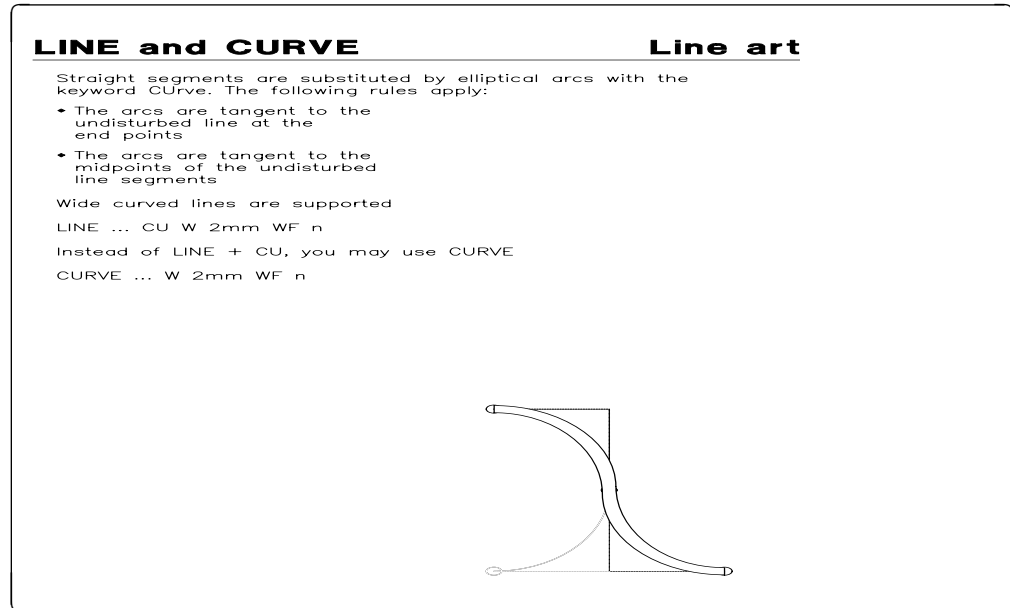


Figure 12. Graphics field with aspect ratio A4

ASPECT also lets you use **aliases** to make this easier, such as:

```
aspect a4
```

When the aspect ratio is taken from an alias, you can add L (**landscape**) or P (**portrait**) following the alias. A landscape picture is wider than it is high; a portrait is higher than it is wide. The illustration above shows a portrait-oriented graphics field.

Here's an example:

```
aspect a4 l      ;* give me landscape A4
aspect a4 p      ;* portrait - but that's also the default
```

When you use the ASPECT command as an infile command, OPS deletes the existing graphics field, and creates a new one in accordance with the command. It doesn't display the page again, so ASPECT must be executed before text and graphics commands that draw on the page in question. Also, because ASPECT modifies the coordinate system (described below), execute ASPECT before any command referring to coordinates, including RESET.

If you use the ASPECT command as a direct command, OPS refreshes the page when the new graphics field has been created. The appearance of the page depends on how text and graphics are defined.

Clearing the graphics field

Use the CLEAR command to clear the graphics field, removing text, graphics, and OPS-generated frames. The command is:

```
clear
```

You use CLEAR mainly when making a presentation, or for special effects with dynamic or animated pictures. Under CMS, you can also use CLEAR to clear

diald screens. See Chapter 16, "Using diald terminals under CMS" on page 191 for more details.

Invoking GDDM user control mode

Use the USERCTL command to invoke GDDM user control mode for pictures you are displaying with OPS. User control mode lets you zoom in or out of the picture, switch between draft and full drawing mode, and invoke GDDM output functions.

When you invoke user control mode, you'll see a new menu of PF keys at the bottom of your screen:

- PF1** Switch the display of PF keys on and off.
- PF2** Change the center point of view of your picture. GDDM asks you to move the cursor to the center point you want, then refreshes the screen around the new center.
- PF3** Leave user control mode.
- PF4** Display a menu of options from which you can print or plot the picture using ADMPRINT or PLOT.
- PF5** Zoom in to the picture, and display the chosen subsection full screen. GDDM asks you to move the cursor to select opposite diagonal corners of the part of the picture that you want to zoom into.
- PF6** Zoom out of the picture, and display the reduced result full screen. GDDM asks you to move the cursor to select opposite diagonal corners of the part of the picture that you want to zoom out of.
- PF8** Switch between draft and full draw modes. Draft mode is a simplified, and quicker, mode of drawing, provided in OPS through the DRAFT command.
- PF9** Leave user control mode.
- PF10** Reset anything you've done with PF2 (center), PF5 (zoom in), or PF6 (zoom out), and return to the original display.

Using the coordinate system

You specify positions on the screen using an addressing system called a **coordinate system**.

A point is addressed by specifying an **x-coordinate** (the horizontal distance from the left edge of the graphics field to the point) and a **y-coordinate** (the vertical distance from the bottom of the graphics field to the point). You must always specify both coordinates, the x-coordinate preceding the y-coordinate. Here's an example:

```
point 10 20      ;* put a point at x=10 and y=20
```

The coordinates in OPS run from 0 to 100 horizontally and from 0 to the value &ymax vertically, as shown in Figure 13 on page 61.

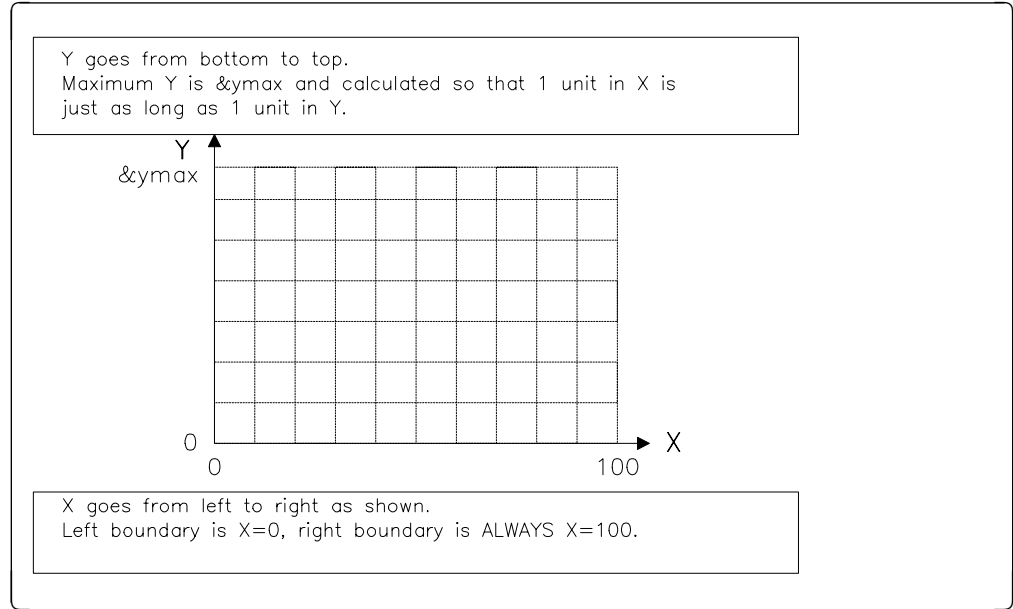


Figure 13. Coordinate system

The value of &ymax depends on the dimensions of the graphics field. It is determined in such a way that a y-unit is as long as an x-unit. In other words, &ymax is equal to 100 times the aspect ratio of the graphics field. If the graphics field is defaulted, &ymax is 67.

Use GRID to display the coordinate system, as described in “How to create presentations” on page 22.

Use the help command GETC to read coordinates from the screen using the cursor, as explained in “Reading coordinates from the screen” on page 26.

Use the ASPECT command to ensure a specific &ymax value. The following command ensures that &ymax is exactly 65:

```
aspect 100 65
```

To make the graphics field fill the entire screen (less the two lines reserved for alphanumeric fields), use:

```
aspect *
```

Here are some other examples. (Remember that &xmax is always equal to 100.)

```
aspect 1 1 ---> &ymax = 100 (square graphics field)
```

```
aspect 210 297 or  
aspect a4 ---> &ymax = 141.4 (vertical A4)
```

You can query the &ymax value using the command:

```
q ymax
```

Units in the coordinate system

To make it easy for you to manipulate coordinates, OPS has a set of practical units, which it converts to the internal coordinate system. You specify a number, immediately followed by the **unit descriptor** of the unit system you want to use.

Valid unit descriptors are:

- %** Percentage of maximum in the given direction
- %x** Percentage of maximum in the x-direction
- %y** Percentage of maximum in the y-direction
- m** **Ems**, horizontal character size box
- t** **Tees**, vertical character size box
- l** Current line spacing
- p** Pels (display points)
- px** Pels as measured in the x-direction
- py** Pels as measured in the y-direction
- r** Rows
- c** Columns
- mm** Millimeters, measured on the screen
- cm** Centimeters, measured on the screen
- i** Inches, measured on the screen

Here are some simple examples, where the POINT command is being used to color the point specified by the coordinate pair:

POINT command	Coordinate used
POINT 12.5mm 20r	Point 12.5mm in from left boundary, 20 rows up
POINT 10 10	Point 10 units in, 10 units up in the internal coordinate system of OPS
POINT 50% 50%	Point in the middle of the screen
POINT 50 50%	Point in the middle of the screen (&xmax = 100)

You can also specify coordinates and dimensions as arithmetic expressions with a free mixture of units, as shown below:

POINT command	Coordinate used
POINT &x+2mm &y	Point 2mm to the right of the last written text
POINT &xmax-1p &ymax-1p	Point one pel from the right edge and one pel below the upper edge
POINT &x+2mm+1p &y	Point 2mm + 1pel to the right of the last written text

You can use most of the units in any context. However % and p are **context sensitive**: the coordinate values they convert into depend on the direction involved. 50% of y-max is not the same as 50% of x-max. Pels need not measure the same in both directions.

In the POINT commands above, % and p are used only in contexts where there is no doubt about which direction is involved. The first number after POINT refers to the x-direction, the second to the y-direction.

Used out of context, that is in non-graphics commands, % and p are still useful, as long as you don't use them in arithmetic expressions. All other units can appear in arithmetic expressions in any context.

Here are some examples of valid expressions:

```
line 2mm+5% 6+2p 100%+&x+2*1c    ;* context OK
set h = 1t
set center = 50% 50%              ;* treated as "text"
set a = 5mm 2.3+10mm
```

The next-to-last example assigns the character string "50% 50%" to *center*. No conversion is required at this point, but when *¢er* is used in a command later on, "50% 50%" must be in an appropriate context.

The last example is acceptable because 1mm represents the same distance in the x- and y-directions. 5mm and the expression 2.3+10mm are evaluated in terms of the basic coordinate system.

The following examples are *not* valid:

```
set point2 = 5h 50%+5
set dx = &y+1p
```

because 50% and 1p cannot be converted as they stand. In the SET command there are no x- or y-directions involved, so OPS doesn't know what 50% means.

To get around the last problem, use the STR function, described in "Interpreting general phrases" on page 165, or use context insensitive units for percentage and pels: %x, %y, px, and py. A correct form of the above is:

```
set point2 = str(5h 50%+5)
set point2 = 5h 50%y+5
set dx = &y+1py
```

The first example saves *&point2* as a general phrase, which is not evaluated before it is used. The second example saves *&point2* as a general phrase of two floating-point values. If the height (h) is changed between the above assignments and actual use of *&point2*, the value of *&point2* is affected.

Note: OPS converts the expressions to the internal coordinate system as soon as it finds the coordinate or dimension. Any change in the coordinate system (using an ASPECT command) *after* conversion may cause unexpected results. For example, if you specify the text height as 10% (of *&ymax*), OPS converts it to a height of 6.7 if *&ymax* is 67. If you change *&ymax* to 100 (using ASPECT 100 100), the height will correspond to 6.7%, and not the 10% expected. To avoid such problems, let ASPECT be the first command on any page where you need it.

Adjusting the coordinate system

Use OFFSET to offset (translate) the coordinate system from its normal position. OFFSET is available both as a primary command and as a subcommand on all functions which draw on the screen, such as TEXT, BOX, and GDF.

Here are two examples:

```
offset 10  
offset 20 -10
```

where the first value is the x-offset, the second the y-offset. You can abbreviate OFFSET to OFF or O.

The default coordinate position corresponds to zero offset:

```
off 0
```

For more detail on OFFSET, see “Displacing objects” on page 131.

Chapter 7. Text in presentations

This chapter tells you all about using text in your presentations. It starts with “Basic text handling,” which deals with the concepts of characters and symbol sets. The second topic, “Advanced text handling” on page 73, looks in more detail at the commands available, using a number of examples to show you the results.

Basic text handling

This topic describes the basic concepts of text handling and the associated commands. Three things determine the appearance of text:

1. The typestyle or font, in OPS named the **symbol set**
2. The **size** of the characters
3. The **color**

Of course, there are many more options and commands that affect the appearance of text. You will learn more about them in “Advanced text handling” on page 73.

This topic also includes a section about **translation tables**.

Symbol sets

OPS can display text using the symbol set that is built into the terminal, or using any available GDDM symbol set. Under CMS, GDDM symbol sets have the filetype ADMSYMBL; under TSO, they are data sets allocated to the *ddname* ADMSYMBL.

The OPS tutorial contains a section titled “GDDM Symbol Sets” which illustrates the sets.

Selecting symbol sets

You select symbol sets using the command SS (Symbol Set). For example, to select the hardware symbol set, use:

```
ss hw
```

To select a GDDM symbol set called ADMUWTRP, use:

```
ss admuwtrp
```

You do not need to specify the type of the symbol set; this is determined by OPS.

You can get a list of available symbol sets by typing a ? instead of the name of the symbol set:

```
ss ?
```

or, more usefully, a limited list such as:

```
ss admu*
```

This gives you a list of all symbol sets beginning with ADMU, as shown in Figure 14 on page 66.

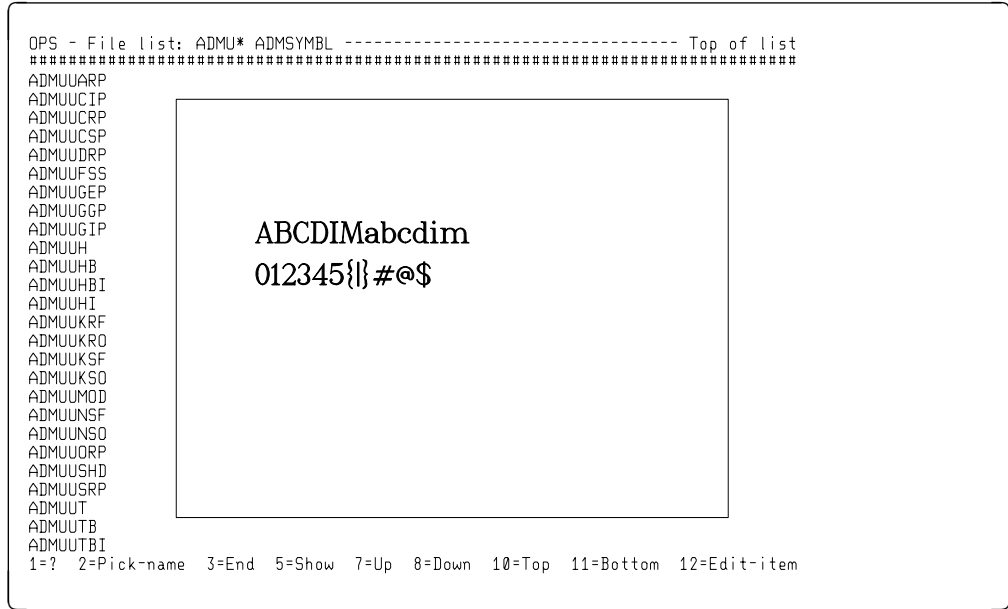


Figure 14. List of symbol sets

This list includes the boxed display you get when you press PF5 (Show) with the cursor on a particular symbol set. To select that set, press PF2.

When OPS starts, you get the default GDDM vector symbol set, VSS.

```
ss vss
```

Similarly, you can select the GDDM default image symbol set by:

```
ss iss
```

Selecting symbol sets with TRC codes: If the infile contains **Table Reference Code** (TRC) control characters in column 2, each TRC-value (0, 1, 2, or 3) can be associated with a symbol set. For example:

```
ss admuvtrp admuvtip admuvsrp
```

sets the first symbol set (ADMUVTRP) to TRC-value 0, the second (ADMUVTIP) to TRC-value 1, and the third (ADMUVSRP) to TRC-value 2. You can specify a maximum of four symbol sets. If the file does not contain any TRC control characters, only the first symbol set is active. If the file contains more TRC-values than the selected number of symbol sets, the last symbol set defined is used for the additional TRC-values.

An example of an infile with ASA control characters in column 1 and TRC control characters in column 2 is:

```
>12<--- specifies column 1 and 2 in the infile

1 Beginning of a new page because of "1" in col. 1
)ops ss admuvtrp admuvtip admvsrp
0 Is shown with amduvtrp (TRC-value 0)
1 Is shown with admuvtip
2 Is shown with admvsrp
3 Is shown with admvsrp which was the last symbol set -
3 - defined with the ss-command
```

To make OPS accept TRC control characters, either specify TRC as a startup operand, or issue a TRC command like this:

```
trc on
```

When TRC is ON, text and OPS commands must not begin before column 3, because TRC characters are assumed in column 2. If the first character of text seems to be missing on the screen, it could be because it begins in column 2 while TRC is ON.

Controlling the character box

You can specify the character box size using the SIZE and HEIGHT commands.

```
size 9p 12p
```

specifies that the character box should be 9 pels (display points) horizontally (x-direction) and 12 pels vertically (y-direction). OPS accepts sizes in a number of units (here pels); see "Using the coordinate system" on page 60.

You can always specify the dimensions of the character box with the SIZE command. However, for image symbol sets, you often want to use the natural size (design size) which OPS sets automatically when you select an image symbol set. This means that you only need to specify SIZE for an image symbol set if you want to use another size for the character box. Remember that the size of each image character is independent of the character box: it always takes up the same number of display points. Variations in the character box will show only in the distance between each character.

Note: Be aware that most of the vector symbol sets you are likely to use have **proportionally** spaced characters ("i" is less wide than "m"). Each character in such a set has a defined **width**, which is less than or equal to the character box width. Thus, setting a very precise character box size with SIZE may not always yield the result you imagine.

Using a vector symbol set, you will often want to use the design aspect ratio which OPS is able to establish as soon as the height of the character box is specified. The following command:

```
size * 10mm
```

character box

specifies that the character box must be 10mm high and that OPS must set the width to keep the design aspect ratio. You can do this more easily using the HEIGHT command:

```
height 10mm
```

Figure 15 shows the effect of various character box sizes for image and vector symbol sets.

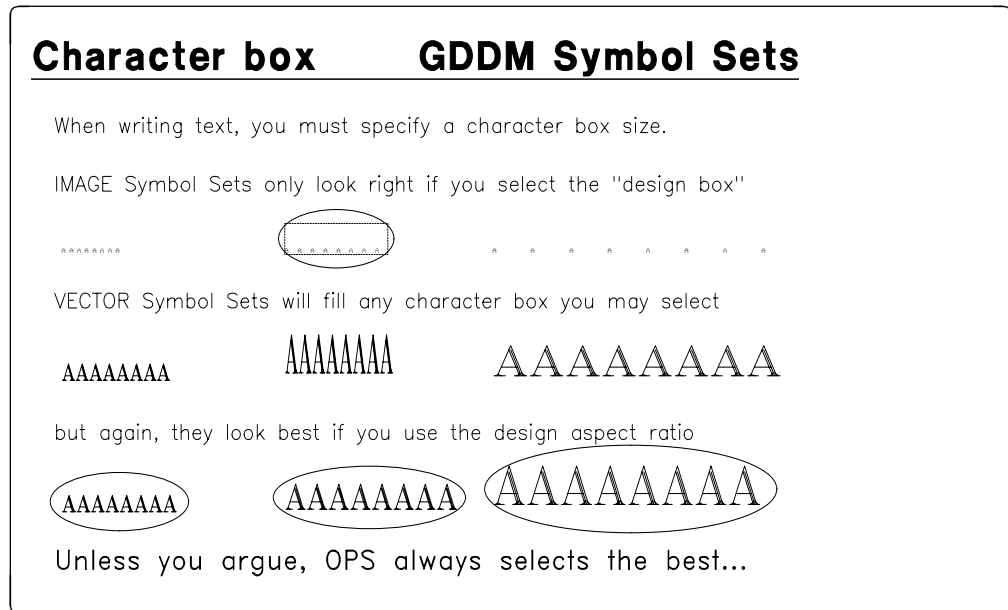


Figure 15. Effects of variation in character box size

When you select a symbol set with the SS command, the character box is set in the following way by OPS:

- Image symbol set: the character box is set to design size.
- Vector symbol set: the height of the character box remains unchanged; the width is set in accordance with the design aspect ratio.

If the file contains TRC control characters in column 2, you can specify up to four character boxes corresponding to the four symbol sets that you can specify with the SS command. For example:

```
size 9p 12p, 9p 12p, 1cm 5%, 10mm 1.5cm  
height 5 5 6 6
```

If you specify fewer character boxes than the actual number of TRC-values, the last character box is used for the additional TRC-values. Typically only one character box is specified for all TRC-values.

Coloring text

Specify the color of text with the COLOR command:

```
color red
col red
```

Text is not limited to the seven colors valid for alphanumeric fields but can be written with any of the **basic GDDM colors** listed below.

All color values may be expressed either by number (-2 to 16), or by a valid, unique abbreviation of the English color name.

The full range of color values and their associated names looks as follows, with minimum abbreviations shown in uppercase:

Basic GDDM colors

- | | |
|---|-------------------|
| 1 | Blue |
| 2 | Red |
| 3 | Pink or MAgenta |
| 4 | Green |
| 5 | Turquoise or Cyan |
| 6 | Yellow |
| 7 | White |

(for text and graphics also):

- | | |
|----|-------------------|
| -2 | WW (always white) |
| -1 | NN (always black) |
| 0 | Default |
| 8 | None or NEutral |
| 9 | DARkblue |
| 10 | Orange |
| 11 | PURple |
| 12 | DARKGreen |
| 13 | DARKTurquoise |
| 14 | Mustard |
| 15 | GRAY or GREY |
| 16 | BRown |

-2 (always white) appears white on a display, and white on a hardcopy. -1 (always black) is black on a display and black on a hardcopy. 0 (default) is green on a display, and black on a hardcopy.

The hue of values above 8 depends on the device. If the device supports only eight colors, which is true for all IBM plotters and displays except the PS/2 and 3270 PC (AT)/GX, the values 9 to 16 are mapped onto 1 to 8 by GDDM.

coloring text

The following scheme applies (where PS/2 also covers the GX):

Color number	Hue on PS/2	Hue on non-PS/2	6-pen plotter pen number	8-pen plotter pen number
-2	White	White	none	none
-1	Black	Black	6	7
0	Green	Green	6	8
1	Blue	Blue	1	1
2	Red	Red	2	2
3	Pink	Pink	3	3
4	Green	Green	4	4
5	Turquoise	Turquoise	5	5
6	Yellow	Yellow	6	6
7	White	White	6	7
8	None	None	None	None
9	Dark blue	Blue	1	1
10	Orange	Red	2	2
11	Purple	Pink	3	3
12	Dark green	Green	4	4
13	Dark Turquoise	Turquoise	5	5
14	Mustard	Yellow	6	6
15	Gray	White	6	7
16	Brown	Green	6	8

To get a graphic overview of available colors, type ? after COLOR:

```
color ?
```

If the infile contains TRC control characters in column 2, you can define a color for up to four TRC-values using the COLOR command. The following example:

```
color r g brown pu
```

makes text in lines with TRC-value 0 red, text in lines with TRC-value 1 green, and so on.

If you specify fewer colors than the number of TRC-values, the last color defined is used for the additional TRC-values. However, OPS will select unique colors for codes that do not have their own symbol set, so you can always distinguish the different TRC-coded lines, either because of the symbol set or because of the color.

Using translation tables

Printer symbol sets can contain up to 255 characters, showing all combinations in a byte (eight bits). However, GDDM symbol sets are more limited. An image symbol set contains only 192 characters: the characters from position 65 (hexadecimal 41) to 254 (hexadecimal FE). Similarly, a number of positions in a vector symbol set are invalid (such as hexadecimal 11, 12, and 13). See *GDDM Base Application Programming Reference*.

To reproduce a printer symbol set on the screen using a GDDM symbol set, you need to translate characters in the infile from one position to another, using translation tables. These are either **global** (they apply to all text) or **symbol set specific** (they apply only to one or more specific symbol sets).

You define one table without any name, and up to 10 tables with names. The nameless table is automatically global when it is defined, while a named table has to be used specifically. The nameless table is deleted by the RESET command; it is temporary, while the named tables are more permanent.

The nameless translation table

You define the nameless global translation table with the command TI (Translate on Input). The TI command specifies connected pairs of characters, where the first character in the pair specifies the position in the infile and the second specifies the new position. For example, to translate every lowercase *a* and *b* into uppercase, use:

```
ti a A b B
```

To specify positions not available on the keyboard, use a hexadecimal value with two hexadecimal digits within the range 00 to FF. For example, to translate hexadecimal 11, 12, and 13 into 41, 42, and 43 use:

```
ti 11 41 12 42 13 43
```

The TI command on its own resets the table. If you want to delete it, use:

```
ti
```

To extend a table which has already been defined or to make the table definition step-by-step, use the keyword MOD:

```
ti mod b B 13 43 ...
```

The nameless global table is always reset by RESET.

Named translation tables

Named tables are handled by three commands beginning with TRT: TRTDEF, TRTUSE, and TRTDROP.

You define the table with TRTDEF, which names the table:

```
trtdef mytab a A 11 41
```

The name consists of up to eight characters, numbers, or letters.

You start using a named table with the TRTUSE command. You can choose between making the table global and making it symbol set specific. To make a table global, use:

releasing symbol sets

```
trtuse mytab
```

If another table is already active and global, it becomes inactive; only one table at a time is allowed.

To make a table symbol set specific, specify the names of one or more symbol sets in the command. For example:

```
trtuse mytab admuwtrp admuwtip
```

makes MYTAB active for text printed with one of the symbol sets ADMUWTRP or ADMUWTIP.

When you no longer want to use a named table, use the TRTDROP command. If the table is global, use:

```
trtdrop
```

To dissolve the connection to a symbol set, specify the name or names:

```
trtdrop mytab admuwtrp
```

You can dissolve the connection to all symbol sets using:

```
trtdrop mytab *
```

Note: TRTDROP does *not* delete the table, but only the connection between the table and the symbol set.

When OPS starts, it defines a named table called UPCASE, which simply folds lowercase letters into uppercase. You can modify the table using TRTDEF.

Releasing symbol sets

Selected symbol sets are loaded into user storage, where they occupy some space, just as they do in the internal tables of OPS.

Normally, you do not have to think about releasing this space, but if the internal tables become full, OPS will inform you (in connection with an SS command) and ask you to release one or more symbol sets. You do this with the RELEASE command:

```
release admuwtrp admuvtip
```

If you release an active symbol set (a symbol set mentioned on the previous SS command), it is replaced by VSS.

Loaded symbol sets are assigned an internal number by GDDM. In connection with GDF files (see Chapter 9, “Displaying pictures” on page 135), the same symbol set may have been loaded several times but with different numbers. In this case, you can release a specific instance of the symbol set by referring to the number instead of the name, as in:

```
relss 200 201 202
```

You can mix names and numbers of symbol sets on the RELEASE command.

You can get a list of loaded symbol sets using the QUERY command:

```
query ss all
```


This shows the symbol set name, symbol set number, and symbol set type (Image, Vector, and so on) of all loaded symbol sets.

You can release **all** loaded symbol sets in one operation by specifying:

```
release *
```

This also releases any active symbol set.

Advanced text handling

Here we describe the advanced OPS facilities for text handling. They are meant for OPS presentations, so the samples are mainly fragments of fictitious presentations. Try them out as exercises.

Don't start to use this information before you are familiar with the material in "Basic text handling" on page 65.

Setting defaults for text

Symbol set, character box, and text color are controlled by SS, HEIGHT, and COLOR as described in "Basic text handling" on page 65. When you use RESET, the symbol set is reset to VSS, the size to the size of the cursor, and the color to turquoise.

However, you can customize these values as you like with the DEFAULTS command. Here are some examples:

```
defaults ss symbol-set-name ...
defaults height character-box-height ...
defaults size character-box-width -height ...
defaults color color ...
```

You can specify up to four values on each subcommand (eight on the SIZE subcommand), just as for the parallel primary commands (SS, HEIGHT, and so on).

The commands you specify will become the defaults that OPS applies when you execute RESET.

You can also make RESET leave the symbol set and attributes alone, using:

```
defaults ss * ;* RESET should not touch symbol sets
```

You can also control the TRC setting using DEFAULTS. The initial default is for RESET to leave TRC as it is. This corresponds to:

```
defaults trc * ;* RESET should not touch TRC
```

But you can change the default action using the commands:

```
defaults trc off
defaults trc on
```

You'll find more examples of using DEFAULTS in "Controlling the space between characters" on page 74 and "Controlling the space between lines" on page 76.

Controlling the space between characters

Text is written character by character, with one character box beginning immediately after the other, as illustrated in Figure 15 on page 68.

The CSP command controls the **character interspacing**: the space inserted between consecutive character boxes in a text string. Positive interspace gives a wider text; negative interspace condenses the text.

You can specify up to four values on CSP, relating to the four possible TRC values (0-3). If you don't use TRC codes, only the first value matters.

For example, after executing:

```
csp 1mm -.05m .2
```

text written with TRC value 0 has a character interspace of 1mm, text with TRC value 1 has a negative character interspace of 5% of "m" (the width of the character box), and text with TRC value 2 has a character interspace of 0.2.

Note that the characters do not change appearance when you insert character interspace. The character box is affected only by HEIGHT or SIZE.

The following figure illustrates the effect of various CSP values. The text lines were written with symbol set ADMUUKSF. Again, remember that unit m denotes the width of a character box.

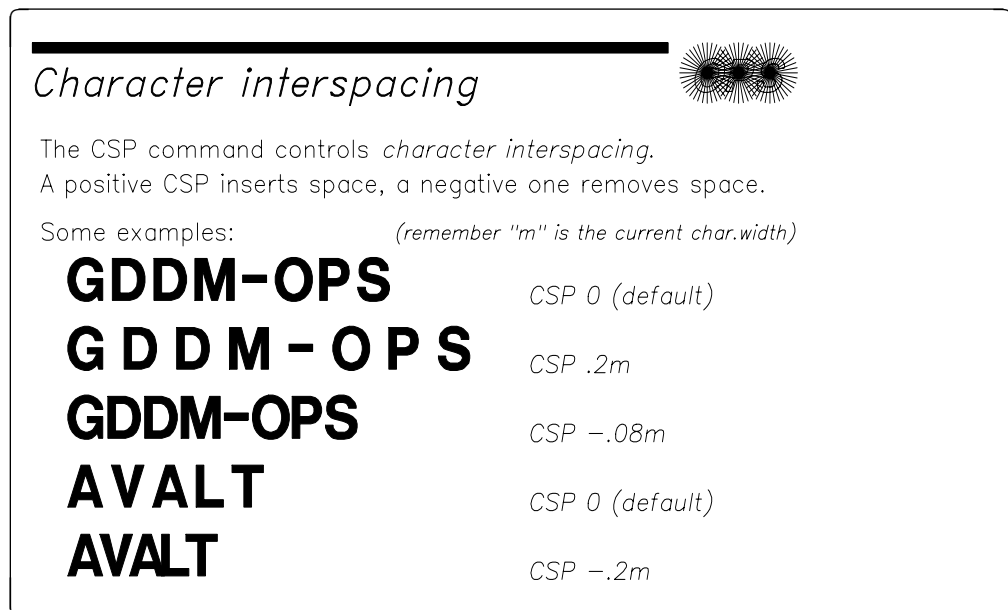


Figure 16. Effect of CSP values on text

It makes a difference whether the CSP value is **absolute** or **relative to the character box**. In the first case CSP does not change with the character box size, but stays with its absolute value. In the second case CSP is automatically adjusted after the character box:

```

1 ...
)ops size 17mm 10mm
)ops csp 0.1m
  Hello
  Hello
)ops size 8.5mm 5mm
  Hello
  Hello
1 ...

```

In this example, the four occurrences of “Hello” will show a character interspacing of 10% of the character box width; the first two with a spacing of 1.7mm, the last two with a spacing of 0.85mm.

Note that this is an exception to the rule that a coordinate expression with a unit is evaluated immediately in terms of OPS coordinates. In this case, OPS remembers that the spacing was set relative to the character box.

For tables or tabular data, non-proportional symbol sets can be very practical. They eliminate the need to set up tabs; you can align the columns with your text editor. However, the GDDM non-proportional sets are all very wide in the character box, making them impractical for tables.

Use CSP to squeeze the blank space out of the GDDM non-proportional symbol sets, as shown in Figure 17.

This header uses ADMUVSRP with CSP 0			
	1995	1996	Growth
SALES			
Hardware	12.430	15.732	26.6%
Software	2.300	3.012	31.0%
Services	3.230	5.320	64.7%
Total sales	17.960	24.064	
Salaries	10.095	14.210	
Materials	3.123	3.654	
House rent	2.012	2.402	
Expenses	15.230	20.266	
Income before taxes	2.730	3.798	

The text used ADMUVSRP with CSP -0.3m

Figure 17. Using CSP with non-proportional symbol sets

Using the DEFAULTS command, you can also set defaults for CSP other than 0:

```
defaults csp 0 -.05m
```

This sets CSP to 0 and -.05m when RESET executes.

Putting text in the infile

Text lines in the infile make a new line on the screen, with line space equal to the height of the character box.

Look at this example:

```
1 )ops reset
  )ops ss admuutrp; h 20mm; color red

      Hello
  )ops color green
      Hello
1 ...
```

It writes “Hello” twice on the screen, with a character box height and line distance of 20mm. The words appear, aligned almost as in the infile. The empty line gives a distance of 20mm from the top edge of the screen to the first character box.

You can write two words on the same line by using the ASA code “+” (writing without line break) in column 1 of the last line:

```
...
  Hello
  )ops color g
+           Hello
1 ...
```

This example uses a symbol set with individual character width (ADMUUTRP) which means that you cannot be quite sure about the position of the last “Hello” (the space character does not take up as much room as the letters).

Controlling the space between lines

Just as you can control character interspacing, you can control the spacing between lines using the command LSP (**leading space**). LSP specifies space to be inserted between vertically consecutive character boxes. An example is:

```
lsp 3mm
```

This inserts a leading space of 3mm between the text lines that follow. The set LSP value is active until modified or set to 0 by RESET.

The sum of leading space and character box height gives the **line spacing** which is available for reference purposes as unit *l*. When LSP is 0, *l* is equal to *t*. See Figure 18 on page 77 for an illustration.

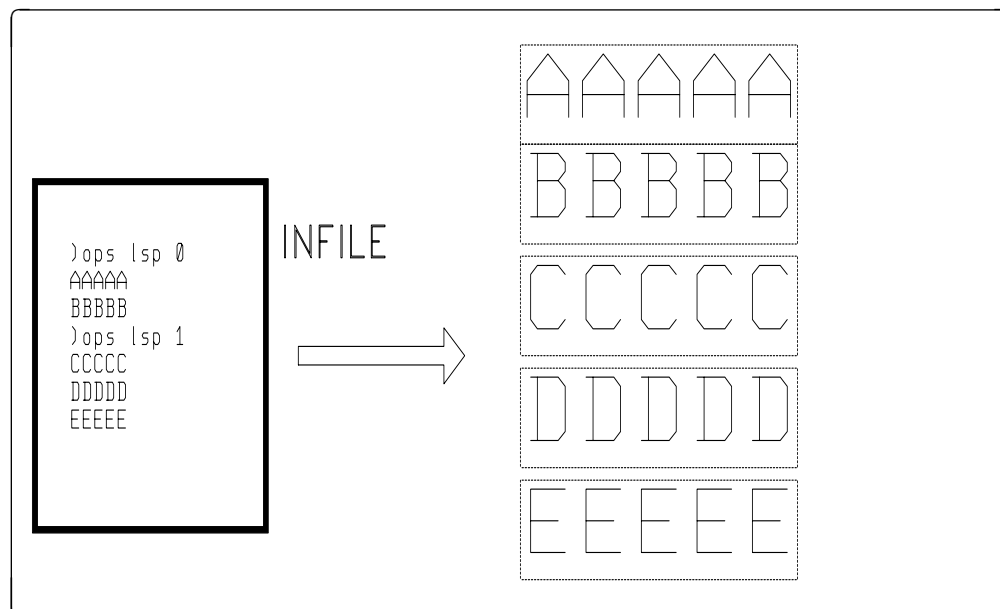


Figure 18. Effect of different LSP values

As with symbol set, character box, and color, you can specify from one to four values on the LSP command, each associated with a TRC-value. An example is:

```
1 )ops reset
  )ops ss admuudrp admuusrp
  )ops h 4.5 3
  )ops lsp 0 .5
  ...
  )ops trc on
0This is a header
1
1This is a detail line
1And so is this
  ...
```

The detail lines, including the empty line, are written with a leading space of .5 units. This will give a lighter appearance to the detail text.

Using the DEFAULTS command, you can also set defaults for LSP other than 0.

```
defaults lsp 0 .5
```

sets LSP to 0 and .5 when RESET executes.

Once again, the character box height is often a good reference. An example is:

```
1 ...
  )ops lsp 0.5t
    Hello
    Hello
    Hello
1 ...
```

This writes “Hello” three times with a leading space of half the character box height, no matter what the character box size is.

italicizing text

As with CSP, it makes a difference whether the LSP value is **absolute** or **relative to the character box**. In the first case LSP does not change with the character box size, but stays with its absolute value. In the second case LSP is automatically adjusted after the character box:

```
1 ...  
  )ops h 10mm  
  )ops 1sp 0.5t  
    Hello  
    Hello  
  )ops h 5mm  
    Hello  
    Hello  
1 ...
```

The four occurrences of “Hello” show a leading space of 0.5 times the character box; the first two with a spacing of 5mm, the last two with a spacing of 2.5mm. The spacing between the second and the third “Hello” is also 2.5mm.

Note that this is an exception to the rule that a coordinate expression with a unit is evaluated immediately in terms of OPS coordinates. In this case, OPS remembers that the leading space was set relative to the height.

Italicizing text

If you are using a vector symbol set, you can **shear** (italicize) the text for emphasis with the SHEAR command:

```
shear 20
```

Any text following the SHEAR command will be sheared until it is reset.

Shear is specified in degrees and counted clockwise. 0 corresponds to undisturbed text. See Figure 19 for an illustration.

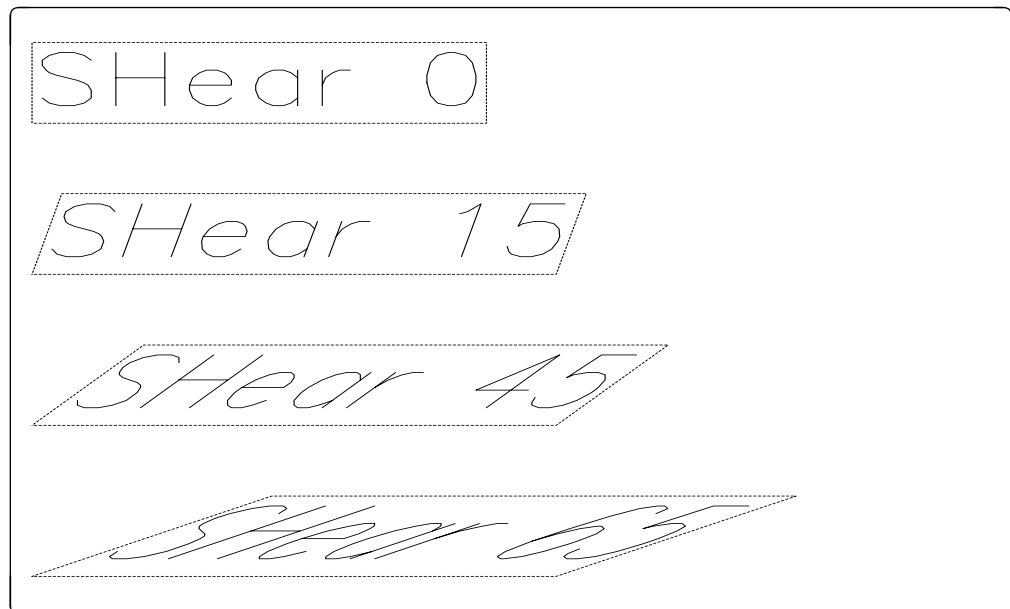


Figure 19. Text shearing

SHEAR can be reset to 0 by RESET or SHEAR 0.

SHEAR has no effect on image symbol sets.

Positioning text

The commands introduced so far control the general appearance of a text block. Using these commands you can make text large or small, light or condensed, and so on.

To control text positioning more closely, OPS provides:

SPACE Insert vertical space
IN Set left hand margin
CENTER Center text
ADJUST Align text left, center, or right
START Set start position for next text
TABS Set tab positions

These commands influence, or are dependent on, **the current point**.

The current point

The current point is the point relative to which a new text line will be positioned, or where the next TEXT command will start writing.

After writing an infile text line, the current point is set to the point **where the next character would have been written** had there been any more in the line. After writing the following:

```
A program from IBM
```

the current point would be after the last M.

You can access the current point using the variable & (contains x and y), or &x and &y, and use them to position text or graphics.

When OPS writes a new text line from the infile, it moves the current point down one line (character box height) and to the left-hand margin; the writing starts from the point.

You can also use OFFSET (introduced in “Adjusting the coordinate system” on page 64) as a way of controlling text position. Reserve OFFSET for control of the **total** picture, and use SPACE, IN, START, etc, for local text control.

Inserting space between text lines

You can lower or raise the current point between writing two text lines with the SPACE command. For example:

```
space 2mm
sp    -3l
```

SPACE moves the current point the specified amount down (or up if length is negative), and adjusts to the left margin.

The character box height is a useful unit to use with SPACE. For example:

indenting text

```
1 ...
   Hello
)ops space .5t
   Hello
```

makes a space of half a character box, no matter what size it is.

The difference between SPACE and LSP is that SPACE inserts space here and now, while LSP is associated with the symbol set, and inserts space between lines as long as it is active (different from 0).

Indenting text

For horizontal control, use the IN command to adjust the left margin. For example:

```
in 30
```

sets the left margin at x=30 (30% of the width). Text lines from the infile are left-justified against this limit.

You can adjust the left margin relative to the current position by putting a + or - sign in front of the value. For example:

```
in +5mm
in -3c
```

moves the left margin 5mm to the right or 3 columns to the left.

You can also specify the indentation as a text string, as in:

```
in '* '
```

The indentation is exactly the width of the text string if the text is written with the current symbol set and character box. For example:

```
1 ...
  * This is worth considering
)ops in '* '
  as blah blah
...
```

positions "as" directly below "This" because the indentation is the same. You get the original positioning at the left edge of the screen by entering:

```
in 0
```

You can also reset IN with the RESET command.

The following example:

```
1 ...
Options:
)ops in &x+2m
+First
Second
...
```

gives the following result on the screen:


```
Options:  First
         Second
         ...
```

(&x is the position immediately after “:.”; “2m” means the width of two characters.) If the symbol set has a fixed character width, you can write the above in the infile exactly as it is supposed to look. However, if you’re using a symbol set with variable character width, it’s difficult to make “Second” appear exactly below “First” without using IN.

You can also use IN to scroll left and right when using OPS for text browsing. For example:

```
pf 10 'in +40m'
pf 11 'in -40m'
```

sets PF10 and PF11 to scroll 40 characters to the left and right. You can also select other units, such as half a screen (“50”).

Centering text

CENTER places subsequent text around the center (vertical midline) of the screen:

```
center on
ce
```

You can also specify an x-value around which the text is to be centered, as in:

```
center 25
```

This centers the text around x=25 (indented one quarter of the screen).

The following example shows you how to get a centered header in a picture:

```
1 )ops reset; ss admuutrp; h 7%; color y
  )ops ce on
  Centering text
  )ops ce off
  ...
1
```

CENTER is ended by CENTER OFF or RESET.

CENTER is *not* influenced by IN. On the contrary, CENTER OFF restores the original left margin.

Aligning text

The ADJUST command aligns text to the left, the right, or about the center of the screen.

Figure 20 on page 82 shows the result of various ADJUST commands on simple text strings.

positioning text exactly

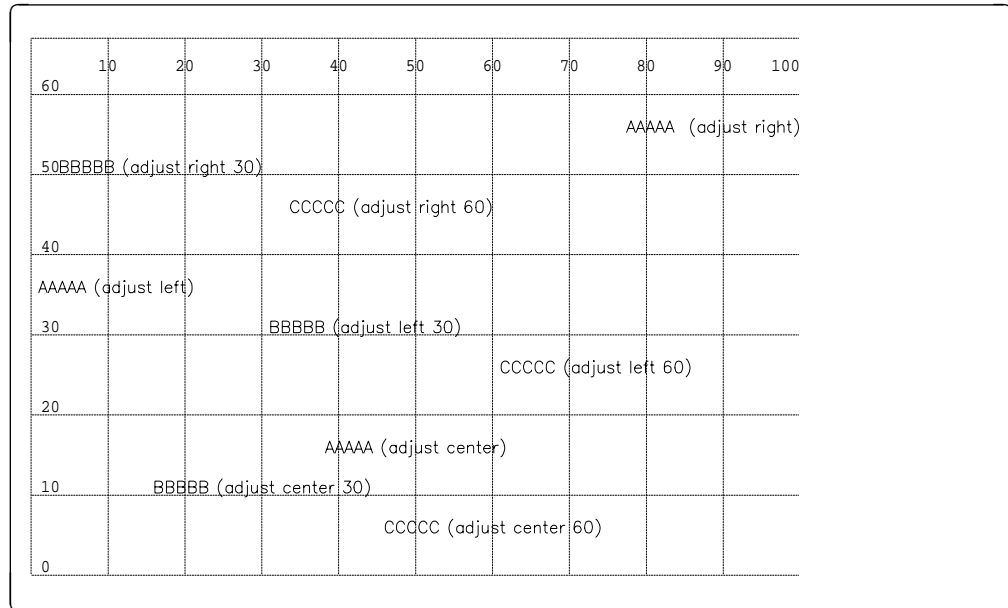


Figure 20. Sample ADJUST commands

The defaults for the options, as shown in Figure 20, are:

LEFT x=0
CENTER x=50
RIGHT x=100

Positioning text exactly

The START command specifies exactly where the next text is to begin. For example:

```
start 10c 15r
```

writes the next text line from a point 10 columns in and 15 rows up.

START also adjusts the left margin to the specified x-value to make it easy to position blocks of text with the START command. For example:

```
1 ...  
  )ops start 10c 40%  
  Line one  
  Line two  
  Line three  
  ...
```

writes the entire block left-justified against x=10c, beginning 40% up.

Writing text in several columns is easy with START. For example:

```

1 ...
  Here follows a list of our products:

  )ops set y0 = &y      ;* remember current y-level
  )ops start 10 &y0
  Apples
  Oranges
  Lemons
  ...
  )ops color pink      ;* might change other attributes too
  )ops start 30 &y ;* start at the saved y-level
  - round and red
  - orange they are
  - a bit bitter
  ...

```

produces:

```

  Here follows a list of our products:

  Apples      - round and red
  Oranges     - orange they are
  Lemons      - a bit bitter
  ...         ...

```

Writing multi-column text this way is especially useful in a dynamic sequence where you want the columns to appear one by one (using a WAIT command between the columns).

Arranging text in columns

TABS lets you arrange text in columns. For example, to set tabs at x=10 and x=30, use:

```

  tabs 10 30

```

To mark that text is to be positioned in a tab position, place a **tab character** in front of the text. The default tab character is `~` (the *logical not* character). The following example writes text in two columns beginning at x=10 and x=30:

```

1 ...
  )ops tabs 10 30
  ~Article~Description

  ~123~Bananas
  ~234~Oranges
  ...

```

The text is left-justified by default. However, you can also center or right justify text, by putting a C (center) or R (right) after the position specification, as in:

```

  tabs 10, 50 r, 55

```

Right adjustment is useful when writing amounts. For example:

coloring words

```
1 ...
)ops tabs 10 50 r 55
~Article~Price~Description

~123~16.50~Bananas
~345~3.25~Potatoes
...
```

produces:

Article	Price	Description
123	16.50	Bananas
345	3.25	Potatoes
...		

Note the difference between using START and TABS for multi-column text: START is suitable when you want to write a **column** at a time, while TABS is aimed at presenting a **line** at a time.

You can use a tab character other than ~ using the TABCH command, as in:

```
tabch #
```

Cancel the tab function with RESET or an empty TABS command.

Coloring individual words and characters

In general, the COLOR command determines the color in which text is written. If you issue the command COLOR RED, the subsequent text will be written in red.

However, you may want to highlight particular words or characters in another color, and you can do this using **color attribute characters**.

A color attribute character is a selected character chosen from:

```
< > ( ) | # ! % / \ ~ * :
```

which you associate with a color. You then put that character in front of a word in text to display the word in that color.

Notes:

1. You cannot use the following as color attribute characters:

```
" ' , &
```

2. Some of the color attribute characters available to you may be different from those shown in this book. See "An important note on national languages" on page xiii for more information.

Use COLATTR to set the color attribute characters, as in:

```
colattr / red < yellow
```

(See "Coloring text" on page 69 for the rules for specifying colors.)

After issuing the above command, you can highlight words as shown below:

```
Ripe apples are normally
either/red or <yellow.
```

By default, the special characters are replaced by blanks when writing. You can also decide that the character does not take up any space in the text, by specifying a 0 after the color, as in:

```
colattr # t 0 < yellow
```

With a color attribute character of length 0, you can change color in the middle of a word. For example:

```
use color#attributes
```

displays “color attributes” in two colors without any visible space.

You usually use color attribute characters of length 0 in column arrangements, or when centering about a certain x-value where a leading color attribute character may disturb the adjustment. For example:

```
1 )ops * a bad example
  )ops colattr / red < yellow
  )ops ce ;* centering around x=50
  Use more
  <OPS
  ...
```

does not center the word OPS around x=50, because < is regarded as a space.

You can solve the problem by using a color attribute character of length 0, as in:

```
...
)ops colattr / red < yellow 0
...
```

Another example, with tabs arrangement, is:

```
1 )ops reset
  )ops tabs 10, 40, 60 r
  )ops colattr # pink 0
  -Article-Model-Price

  -ABC-Standard-5,200
  -DEF-#Special-12,400
  ...
```

Here, the word “Special” is exactly below “Standard” because the color attribute character # is of length 0.

If you want several successive words in the same color, repeat the color attribute character. However, you can also tie the words together with another special character which you define with the keyword KEEP:

```
colattr / r < y _ keep
```

You would typically use “_” or “-” for this purpose. Apart from making the process faster, this character also increases the readability of the infile, as in:

```
After/25_years with the Company
you get an<extra_week_off.
```

specifying text strings

Cancel the color attribute characters with the RESET command, or with an empty COLATTR command:

```
colattr
```

Manipulating text strings

Use the TEXT command to fix, rotate, shear, and frame a single text string.

Specifying the text string

The simplest form of the command specifies only the string to write:

```
text 'hello there'
```

This writes the text "Hello there" starting from the current point.

You can use single or double quotes round a text string, so you could enter the example above as:

```
text "hello there"
```

Symbolic variables are resolved within text strings just as they are within infile text. Unresolved variables do **not** cause error messages; they are simply displayed as entered.

For example:

```
text 'current time is &$time'  
text 'did i define variable &gugu ?'
```

might display:

```
Current time is 120357  
Did I define variable &gugu ?
```

without any error message.

If you **want** to display a variable name in a text string without getting the value resolved, use double-**&**. For example:

```
text 'the value of &&yymax is &yymax'
```

displays:

```
The value of &yymax is 67.00
```

For more details about symbolic variables, see Chapter 12, "Using symbolic variables" on page 163.

Arithmetic expressions are **not** evaluated within quoted strings or in infile text. However, by opening the quotes you can use arithmetic expressions in your text.

For example:

```
text '&x+10' is 10 units to the right of the last text'
```

The precision (number of decimals) is controlled by OGRAIN as explained in “Interpreting numeric values” on page 164.

Aligning text

After writing, TEXT moves the current point to the lower right-hand corner of the last character box, without changing the left margin. A new TEXT command without any explicit start position writes at the current point, for example:

```
1 ...
  )ops text 'The next word is colored '
  )ops color red
  )ops text 'red'
```

However, you usually use TEXT with a start position, as in:

```
text 'hello there' 10 40%
```

Here, the text “Hello there” is written from a point 10 units in and 40% up. The start coordinates are positional operands; if you use them, they must be immediately after the text string.

Sometimes, you don’t want TEXT to move the current point; for example, when using TEXT with text lines in the infile or with text formatting functions. In this case, use the keyword NOSET, as in:

```
text 'hello there' 10 40% noiset
```

The TEXT command can also center or right-justify the written text with one of the following alignment codes:

- CE** To center on the line through the starting point
- CC** To center the full text box around the starting point
- RI** To right-justify text

The alignment keywords must be put in front of any coordinates. Figure 21 on page 88 shows some samples of TEXT alignment.

TEXT command	Text
TEXT supports centering and right adjustment. Use keywords:	
CE – write centered ON (x,y)	
CC – write centered ABOUT (x,y)	
RI – right adjust from (x,y)	
Examples:	
TEXT 'CECECE' 80 30 CE	CECECE
TEXT 'CCGCCC' 80 20 CC	CGGCCC
TEXT 'RIRIRI' 80 10 RI	RIRIRI
(more...)	

Figure 21. TEXT alignment sample

You can use CC to center in a box or a circle, as in:

```

1 ...
  )ops ss admuksf; h 5
  )ops circle 25% 50% 20
  )ops text 'OPS' cc 25% 50%
...

```

The left-hand side of Figure 22 shows the result.

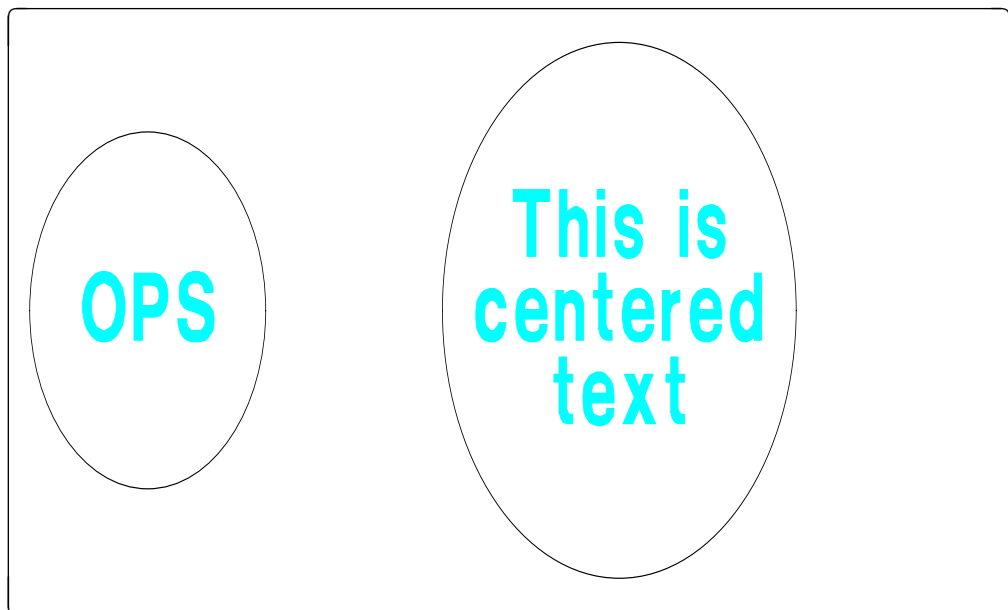


Figure 22. Centering text in a circle

For more than one line of text, either use multiple TEXT commands or START. The following examples give identical results (as shown in the right-hand side of Figure 22):

Using multiple TEXTs

```
1 ...
  )ops ss admuksf; h 5
  )ops circle 65% 50% 30
  )ops text 'This is' cc 65% 57%
  )ops text 'centered' cc 65% 50%
  )ops text 'text' cc 65% 43%
  ...
```

Using multiple STARTs

```
1 ...
  )ops circle 50% 50% 30
  )ops start 50% 56%; center on; h 5%; lsp 1%
  This is
  centered
  text
  )ops center off
  ...
```

You can write a conventional foil header, with a title to the left and a major chapter to the right, using:

```
1 ...
  )ops text 'Foil title' 0 &ymax-1.2t
  )ops text 'Chapter 23' 100 &y ri
```

The first TEXT command writes to the left 1.2t below ymax; the second writes to the right on the same line as the last text. (For an alternative way, see “Creating headers and footers” on page 93.)

You can use the OFFSET command (introduced in “Adjusting the coordinate system” on page 64) as a subcommand on TEXT to offset the text from the specified x and y values. The following example shows how to create a shadow effect:

```
)ops ss admuksf; h 10
)ops text 'Hello world' 10 20 color blue, offset .5 -.5 ;* the shadow
)ops text 'Hello world' 10 20 color turq
```

You can generalize the shadow effect by using a symbolic variable with any TEXT. For example:

```
)ops set tshadow = str(color darkblue, off .05t -.05t)
)ops text 'Hello world' 10 20 &tshadow ;* the shadow
)ops text 'Hello world' 10 20
```

uses a shadow offset of .05t (5% of the current character box height.)

Note that a local OFFSET is **added** to any global OFFSET; it doesn't override it.

Specifying the font

In the TEXT command you can also specify anything else needed to fix a single text (symbol set, size, color, spacing and, shearing), using the subcommands SS, SIZE or HEIGHT, COLOR, CSP, LSP, and SHEAR. When you use these primary commands as subcommands, they cover only one symbol set, and the specified values apply only to what you are writing in the TEXT command. If you do not specify any symbol set, size, color, and so on, the current values apply. For example:

```
text 'bold graphics' 20 20, ss admuksf h 10mm, col r
```

Remember that the start position, if specified, must be put immediately after the text string. As usual, you can use commas to increase readability.

Both SS and COLOR support the use of ? in direct TEXT commands. SS also supports the use of a name pattern with *, as in:

```
text 'help me' 20 20 ss admuu* h 10mm, col ?
```

This lists all symbol sets beginning with ADMUU, and then gives a graphic overview of the colors.

The following example shows how to emphasize a single word in a text by changing the symbol set:

```
1 ...
)ops ss admuutrp; col y; h 7mm
It is
)ops text ' IMPORTANT' col red, h 10mm, ss admuksf
)ops text ' to remember that:'
...
```

Note the leading blank in the text string ' IMPORTANT', which makes space between “is” and “important.”

You may find that the word you highlight in this way appears to have a higher baseline than the rest of the text. This is because the individual characters are not written exactly on the bottom line of the character boxes (there must be room for descenders), and because the offset from base line to the characters varies from symbol set to symbol set. If this happens, you can improve the result by moving the current point downwards before writing the highlighted word:

```
1 ...
)ops ss admuutrp; col y; h 7mm
It is
)ops text ' IMPORTANT' &x &y-.5 col red, h 10mm, ss admuksf
)ops text ' to remember that:' &x &y+.5
```

Transforming the text string

TEXT lets you write text upside-down or mirrored, by using negative values for the character box width or height. See Figure 23 on page 91 for an illustration of how to achieve the various effects. All examples have the form:

```
text 'Size x y' 50 50% size x y
```

where *x* and *y* vary.

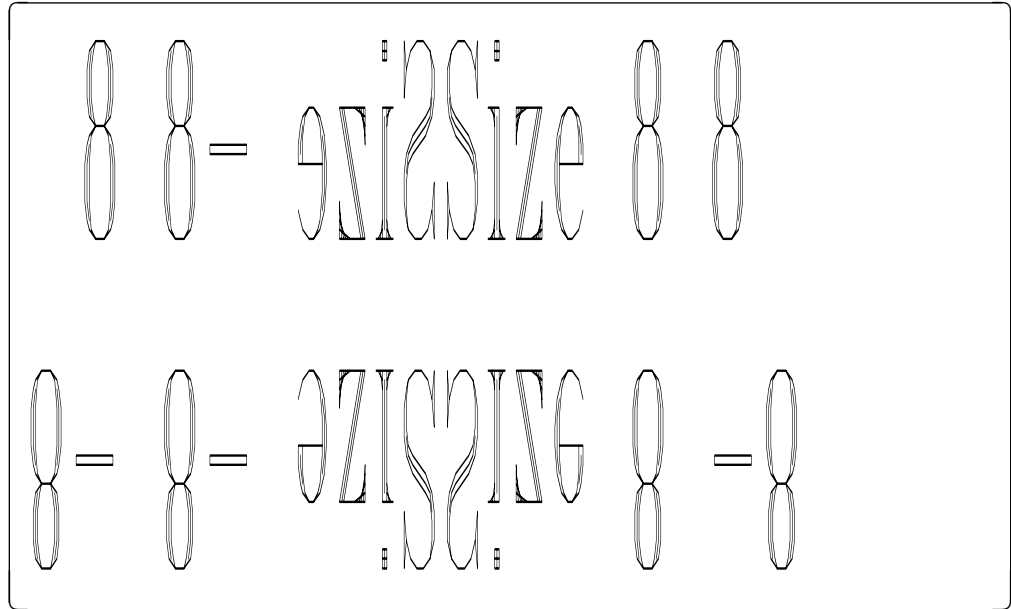


Figure 23. How to mirror text and turn it upside down

Using the TEXT command, you can rotate a text, as shown in Figure 24 on page 92. Use TURN, followed by the angle in degrees, to specify the rotation. Positive angles give counterclockwise rotation, as in:

```
text 'Rotated' 40 10 turn 45
```

You can only really rotate vector symbols; image symbols appear as if written on a ramp, but the individual characters are not rotated.

TURN goes well with centering, as in:

```
text 'OPS' cc 50% 50% turn 180
```

which writes the word OPS upside down in the middle of the screen.

You can also shear a text with the keyword SHEAR followed by an angle in degrees. The angle is counted from the y-axis towards the x-axis, so a small positive angle will result in a text which tilts slightly to the right:

```
text 'Very important' 10 20, shear 20
```

You can combine TURN and SHEAR, as in:

```
text 'Words on a ramp' 2 2, turn 45, shear 45
```

This rotates the entire text 45 degrees counterclockwise while tilting the letters 45 degrees clockwise. The result is a text line pointing up towards the right, with all letters normal. See Figure 24 on page 92.

putting text in a box

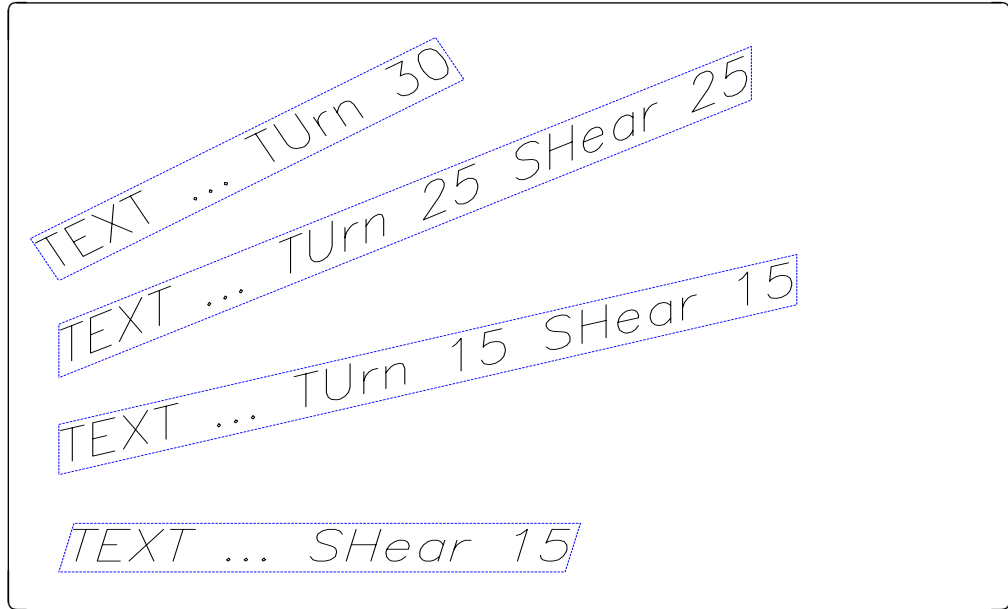


Figure 24. Rotating and shearing text

Putting text in a box

Use the subcommand BOX to frame the text. Apart from the positional operands, it is identical to the primary BOX command (described in “Drawing boxes” on page 119).

To make a simple frame, use:

```
text 'Words in a frame' 50 50% box
```

The box is drawn with the current attributes as described in Chapter 8, “Line art” on page 103.

The framing follows the outline of the character boxes of the text. However, you can adjust the size of the frame using one or two dimension values or scale factors with the BOX subcommand. The dimension values adjust the size in the direction of the text (normally horizontal); the scale factors adjust the size vertically. If you specify only one value, it applies to both directions.

You can specify an absolute box size using something like:

```
text 'A box' 30 30 box 40mm 20mm
```

You can also use a scale factor relative to the text size, using a unit of *. For example:

```
text 'Another box' 30 30 box 1.5*
```

creates a box 1.5 times as large as the text box in each direction. (The default size corresponds to scale factor 1.) See Figure 25 on page 93 for illustrations.

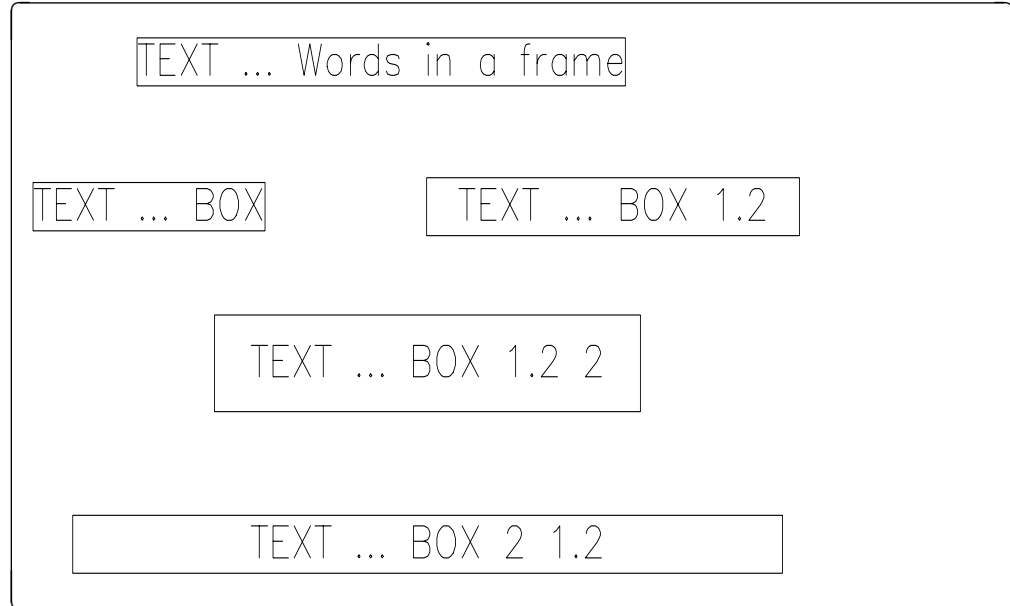


Figure 25. Text in simple boxes

You can omit * if the factor is less than the current text box. For example, if the current text height is 5, you can omit * for all scale factors below 5.

Note: You can refer to the current text height in the BOX subcommand, but, before doing so, specify any SIZE or HEIGHT subcommand. The TEXT command is scanned only once, from left to right, and any absolute box size is evaluated when it reaches BOX. Thus the following two commands may not give identical results:

```
text 'hello' 20 20 h 10, box 20 1.5t
text 'hello' 20 20 box 20 1.5t, h 10
```

For more on the BOX command, see “Drawing boxes” on page 119.

Forcing a line break

Use the BREAK command to force a line break, as in:

```
1 ...
)ops start 30 50
)ops text 'Header' col white
)ops break; text 'Detail' col tur
...
```

BREAK moves the current point to the beginning of the next line. In the example, “Detail” appears just below “Header.” You could also do this by typing “Detail” as an infile text line.

Creating headers and footers

Use HEAD and FOOT to create foil headers and footings.

HEAD writes text up against &ymax and draws a line just below the character boxes. You can use the tab character to split the text into a maximum of three

parts, where the first part is placed to the left against x=0, the next part is centered around x=50, and any third part is placed to the right against x=100. For example:

```
head 'left-center-right'
```

FOOT writes on y=0 and draws a line just above the character boxes.

Figure 26 shows the basic functions.

HEAD and FOOT	Text			
Use HEAD and FOOT commands to create page headings and footings.				
Basic syntax: HEAD FOOT 'Left-Center-Right' ...				
Add keywords SS, SIZE, H, COL, NOSET to them – just as for for the TEXT command. (Graphics attributes are controlled by DRAW.)				
The heading and footing of this page were created by:				
<pre>HEAD 'HEAD and FOOT' ss admuuksf, h 4.5, col r, draw red HEAD 'Text' ss admuuksf, h 4.5, col t, draw off FOOT 'Date-Company Use-N.N' ss admuwrp, h 4, col r, draw yel</pre>				
The separator character is set by TABCH. The default is ~ (the "logical not" symbol).				
<hr style="border: 0.5px solid black;"/> <table style="width: 100%; border: none;"> <tr> <td style="width: 33%; text-align: left;">Date</td> <td style="width: 33%; text-align: center;">Company Use</td> <td style="width: 33%; text-align: right;">N.N</td> </tr> </table>		Date	Company Use	N.N
Date	Company Use	N.N		

Figure 26. Examples of HEAD and FOOT

You can specify SS, HEIGHT or SIZE, and COLOR to control the text, and DRAW to control the color and style of the line. See “Setting line attributes” on page 104 for more detail about DRAW.

You can have multiple HEAD and FOOT commands on the same page, letting you use different symbol sets in the various text parts:

```
1 ...
)ops head 'Agenda' ss admuutrp h 10mm , col r
)ops head '~OPS' ss admuuksf h 10mm, col y, draw off
```

Here, “Agenda” is written to the left with ADMUUTRP, “OPS” is written to the right (due to the tabs characters) with ADMUUKSF. In the second HEAD command, draw off signifies that the line is not to be drawn (it has already been drawn).

The current point is influenced by both HEAD and FOOT. After HEAD the point is set at (0,&y_{max}-1t), at the left terminal point of the line; after FOOT it is at (0,1t), also the left terminal point of the line.

If you do not want to change the current point, add the NOSET option:

```
1 ...
)ops head 'Main subject'
)ops foot 'Company Use~1991' noset
...
```

The NOSET option on FOOT ensures that the current point is left as set by the preceding HEAD command.

Text formatting commands

To help you create neatly formatted text, OPS provides **text formatting**, which automatically positions the text and splits it into lines of suitable lengths.

All text formatting commands start with TF:

TF

Starts and ends text formatting

TFH and TFP

For headers and paragraphs

TFOL, TFUL, TFLI, TFEOL, TFEUL

For lists

TFXMP, TFEEMP

For examples

Text formatting is started explicitly by TF, and implicitly by TFH, TFP, TFOL, and TFUL. Text formatting is ended by TF or RESET.

You can mix the TF commands with the text commands previously described, except for CENTER and TABS. To center text, temporarily suspend text formatting using the TFXMP command. You can define tab positions during text formatting, but you can't use the tab character.

Starting text formatting

TF starts and ends text formatting. Use:

```
tf
```

to start text formatting, writing from the current point. For example:

```
1 )ops reset; start 10 50  ;* sets starting point and left margin
  )ops tf
  This is running text, which will be
  formatted when OPS reads
  it. The left margin will be at x=10.
  ...
```

TF OFF stops text formatting.

You can specify the starting point (left margin) directly on the TF command, as in:

```
1 )ops reset
  )ops tf 10 50
  ...
```

If you omit the y-value, formatting starts at the current y position.

You can add the right margin as a third value:

```
tf 10 50 90      ;* Format between x=10 and x=90. Start at y=50.
```

You can also use the keyword XMAX:

```
tf xmax 95      ;* Left margin controlled by IN or START
```

writing text in paragraphs

You can specify **paragraph spacing** with the keyword PSP:

```
tf psp .75l
```

Paragraph spacing is the space inserted before paragraphs and list items (TFP and TFLI).

The default is XMAX 100 and PSP 1L (a blank line).

You can use IN or START at any time during text formatting to adjust the left margin. There is always only one left margin for all text, formatted or not formatted.

Note that a global OFFSET command will shift all text, including formatted text. Both the left and the right margins are offset.

Writing text in paragraphs

Usually, text is written in paragraphs controlled by TFP. You can write free text after TFP in the same line; it is processed exactly as an ordinary text line in the infile. For example:

```
1 )ops reset
  )ops tf xmax 50; in 10
  )ops tfp This paragraph will be
    formatted and written by OPS. Left margin
    is set by "IN".
  )ops tfp
    And so is this paragraph...
  ...
```

produces:

```
      10                               50
-----+-----+-----> X

    This paragraph will be
    formatted and written by OPS.
    Left margin is set by "IN".

    And so is this paragraph...
```

Text in the infile is formatted until the next TF command or RESET.

Note: A paragraph *cannot* be longer than one screen. However, one page of formatted text can cover any number of screens.

TFH differs from TFP only in that the leading space is 1.5 times the current paragraph spacing, making TFH useful for headers:

```
1 ...
  )ops tfh Please remember that:
  )ops tfp Articles to be delivered
    must be ordered before 4 o'clock
  )ops tfh Particular conditions:
  )ops tfp None
```

TFH does not change the symbol set or height; if you want something special in your headers, use the appropriate commands.

Creating ordered and unordered lists

Using these five commands, you can generate **ordered** or **unordered** lists. You start the list with a TFXL command, end it with TFEXL, and add list items with TFLI. For example:

```
1 ...
  )ops tfh Agenda:
  )ops tfol ; * start of an ordered list
  )ops tfli Item one
  )ops tfli Item two
  )ops tfeol ; * end of list
```

produces:

```
Agenda:
  1. Item one
  2. Item two
```

Ordered lists are marked with sequential numbers and letters; unordered lists are marked with bullets and dashes. The lists are identically structured.

You can nest lists, and mix the different kinds together. You can have up to 10 lists active at the same time. For example:

```
1 )ops reset; prefix :
  :tfh Agenda:
  :tfol ; * start of ordered list
  :tfli Morning:
  :tful ; * start of unordered list
  :tfli Coffee
  :tfli Lunch
  :tfeul ; * end of unordered list
  :tfli Afternoon:
  :tful ; * start of unordered list
  :tfli Break
  :tfeul ; * end of unordered list
  :tfeol ; * end of ordered list
```

produces:

```
Agenda:
  1. Morning:
    o Coffee
    o Lunch
  2. Afternoon:
    o Break
```

suspending text formatting

Each list item is processed as a paragraph, and the space is the default paragraph spacing. You can adjust this with the keyword operand PSP on TFOL and TFUL, which defines a local paragraph distance within the list. For example:

```
tful psp .5l
```

reduces the space to half a line. You can also specify space 0 with the keyword COMPACT:

```
tfol compact
```

Adding COMPACT to the example above gives:

```
Agenda:
  1. Morning:
    o Coffee
    o Lunch
  2. Afternoon:
    o Break
```

Ordered lists are marked with numbers for the first list, uppercase letters for the second, and then with lowercase letters. If you begin more ordered lists, the sequence is repeated. Default marking fills as much space as “9. ” (a nine, a period and a space). If you want another width, use BW (Bullet Width), as in:

```
tfol bw '1. '
tfol bw 30mm
```

You can specify the width as either a text string or a value.

Unordered lists are marked with filled circles (bullets) for the first list, and by hyphens for subsequent lists. The width of the mark is two spaces by default, or whatever you set using BW.

You can also select another marking using BULLET, as in:

```
tful bullet '*'
tful bullet af
```

You specify the character in the same way as when defining translation tables: either as a text string of one character (quotes are optional) or as a hexadecimal value of two characters.

Temporarily suspending text formatting

These commands stop and then start text formatting, typically around an “example,” as in:

```
1 ...
  )ops tfp Another way to do it
  is this:
  )ops tfxmp
    arrow 10 10 30 head 3
  )ops tfexmp
  )ops tfp which is easy to remember.
  ...
```

Using TRCs to select symbol sets

TRC (table reference code) characters control the choice of symbol sets made by a printer. The codes are in column 2, and assume values from 0 to 3, corresponding to four possible symbol sets. As described in “Basic text handling” on page 65, you can control the symbol set of OPS in exactly the same way. For example:

```

1 ...
  )ops ss font0 font1 font2 font3
  )ops trc on; * column 2 is now considered as TRC code
0 Written with font0
1 Written with font1
2 Written with font2
3 Written with font3
  )ops trc off; * if it is used here only...
1

```

You can also attach individual character box sizes, colors, and leading space to the different TRC values. These are used in the example below, which creates a text foil using TF commands and TRC codes.

Note: This example is actually part of the OPS sample file, ADM7OPXA, so you can get the infile and use it yourself as the starting point for one of your presentations.

```

%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A more complicated text foil using text flow functions and TRC
%*****
% reset;                * The recommended way to start a new page
% trc on;                * TRC means: column 2 controls the fonts
%*** Define 3 fonts for TRC use:
% ss  admuudrp admuucip admuusrp
% h   3.5      3      2.3
% col tur      ye1    whi
% lsp 0        0      .1t
%*** now 0 in column points to 1st font, 1 points to 2nd font, etc.
%*** Do NOT add further (global) ss, h, color, or lsp commands
%*** Local commands, like color on the head-command, are OK:
0% head 'Text flow functions' dr r 2 color r
0% head '---OPS' ss admuksf, col t dr off; * Even local SS is OK
%*
%*** Start text flow.
% tf xmax 96; in 4
0% tfh Here are the OPS text flow commands:
1% tful
1% tfli TF - Text flow start
2% tfp May be used to set Xmax and paragraph spacing (psp keyword)
1% tfli TFH and TFP - Header and Paragraph
2% tfp For all the bulk text. TFH spaces 1.5 paragraph spaces -
2thats the only difference.
1% tfli TFUL, TFOL, TFEUL, TFEOL, TFLI - for lists
2% tfp This foil uses an unordered list. You can nest and mix
2unordered and ordered lists to any depth
1% tfli TFXMP, TFEXMP
2% tfp Use these around
lexamples
2or text with
1tabs
% tfeul
% trc off

```

Figure 27. Sample source for a text foil showing TRC codes and formatting

Figure 28 on page 101 shows the result.

Text flow functions
OPS

Here are the OPS text flow commands:

- *TF* – *Text flow start*
May be used to set Xmax and paragraph spacing (psp keyword)
- *TFH and TFP* – *Header and Paragraph*
For all the bulk text. TFH spaces 1.5 paragraph spaces – thats the only difference.
- *TFUL, TFOL, TFEUL, TFEOL, TFLI* – *for lists*
This foil uses an unordered list. You can nest and mix unordered and ordered lists to any depth
- *TFXMP, TFEEMP*
Use these around *examples* or text with *tabs*

Figure 28. A text foil using text flow functions

Writing markers on the screen

The MARK command lets you put **markers** on the screen. Markers are small figures, such as dots and crosses, that you often use to mark points in a coordinate system (as in a line graph). In total 62 markers are available, numbered 1 to 62. They are taken from symbol set ADMDHIMJ, the one used by the Interactive Chart Utility.

The following figure shows all the markers available; you can display it within OPS, using MARK ?.

Syntax: MARK marker-value x y ...
Valid marker-values are 1-254. Markers are read from symbol set ADMDHIMJ.
(The markers you see are also found at b5 + the value shown).

	0	1	2	3	4	5	6	7	8	9
00	×	×	+	◇	□	✱	✱	◆	■	
10	.	\$	£	¢	%	△	▽	▷	◁	▲
20	▼	▶	◀	↑	↓	↔	←	↑	↓	→
30	←	○	⊙	⊖	⊕	⊕	●	⊙	⊙	N
40	∨	Z	Σ	⊗	⊗	⊗	⊗			
50			-	-	-	-	-	+	+	+
60	+	×	×							

5645-001,5684-168,5695-167 (C)Copyright IBM Corp.1988,1996

Figure 29. Selecting of markers

writing markers

Chapter 8. Line art

Line art is what we call graphics other than complete pictures: lines, circles, and so on. Figure 30 shows some samples of OPS line art.

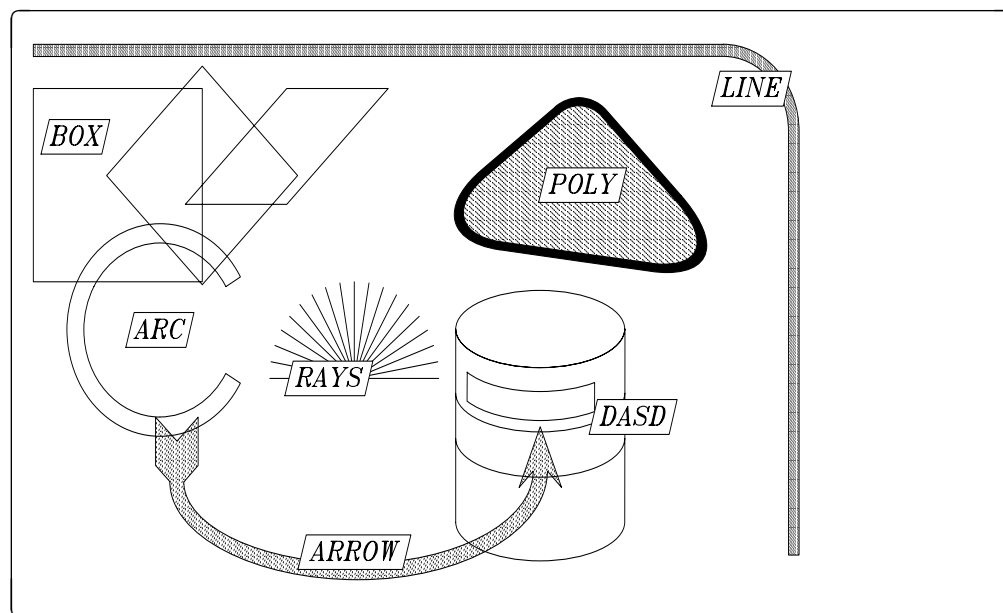


Figure 30. Line art samples

Line and area attributes

All line art is composed of straight or elliptically curved **lines**, possibly enclosing **fillable areas**.

Lines can be drawn in one of the basic GDDM **colors** of the screen (blue, red, pink, green, turquoise, yellow, white, neutral, and so on), in one of two **widths** (one or two display points), and in one of seven **styles** (fully drawn, dot-and-dash, dotted, and so on).

The three attributes (color, width, and style) determine the appearance of a line and must be specified before drawing.

Fillable areas are areas limited by **closed** lines. You can fill them with **colors** and **patterns**, choosing the color from the basic GDDM colors, and the pattern from a number of pattern sets, including the standard 16 GDDM patterns. You can also select a **general color**, or have color fading from any value to any other.

GDDM includes a series of pattern sets of 64 multi-colored patterns. They are called ADMCOLSx, where "x" is a character indicating the device for which the pattern is suited. If your terminal supports pattern loading, OPS will use this pattern set to emulate general colors. Otherwise it uses multiple layers of the basic colors and patterns. You can use only the 16 standard patterns for plotters. In general, be warned that colors and patterns are device-dependent.

setting line attributes

Areas are filled using the GDDM technique summarized in “Grouping objects together” on page 128.

You can display all possible attributes from the OPS help menu, or by using ? on any command.

You set the five basic attributes with the commands DRAW and FILL. If you use DRAW or FILL as a primary command, the attribute values that you set apply to all subsequent graphics commands without any attribute specifications. If you use DRAW or FILL as a subcommand on another graphics command, the attribute values apply only to that command.

Setting line attributes

You set line attributes with the DRAW command; for example:

```
draw red 2 5
```

The color (RED above) can be any of the basic GDDM colors listed in “Coloring text” on page 69. The width (2 above) can be 0, 1, or 2, and the style (5 above) is an integer in the range 0 to 7.

The defaults are:

```
Color    7 (white)
Width    0 (same as 1, normal)
Style    0 (same as 7, unbroken)
```

If you want to draw with a normal width and an unbroken line, all you need to specify is the color. An example:

```
draw blue
```

Note that BLACK on the screen is specified as NEUTRAL.

If you want a thicker line (normally 2 pels), use:

```
draw blue 2
```

You can get a graphic overview of possible combinations of color and line styles by specifying ? after DRAW, in any position. For example:

```
draw ?
draw red ?
```

Figure 31 on page 105 shows the help panel for line attributes.

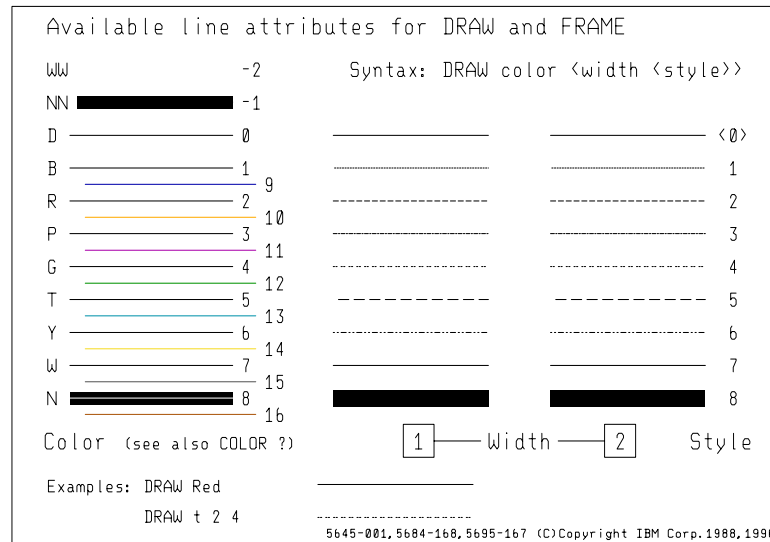


Figure 31. Help panel for line attributes

If you don't want to draw the edge of a figure (such as a circle), use:
draw off

Executing the RESET command sets the line attributes to their default values.

You can set an alternative default line color with the DEFAULTS command:
defaults draw green

Unspecific DRAW attributes

Unspecific attributes are those with a value of 0. The unspecific color (0) is green on screens, but black on printers, including color printers. Unspecific width and style always correspond to a thin, unbroken line.

To request unspecific attributes, use:
draw 0

(OPS defaults for width and style are already equal to the unspecific values.)

You'd most usually use unspecific attributes with ADMGDF files: when you load an ADMGDF, you then give any unspecific attributes new values. This lets you recolor all or part of an ADMGDF.

Setting area attributes

Set area attributes with the FILL command. To let you get the most out of the range of colors on PostScript devices, FILL supports an almost unlimited number of colors. However, few devices support the full range of OPS colors, and the colors are emulated with a level of precision that varies considerably from device to device.

To see the colors and patterns available, put a ? after FILL. This displays two panels, the first showing basic colors and patterns, the second showing the 64

basic colors and patterns

colors used for emulating general colors. The first panel is shown in Figure 32 on page 106.

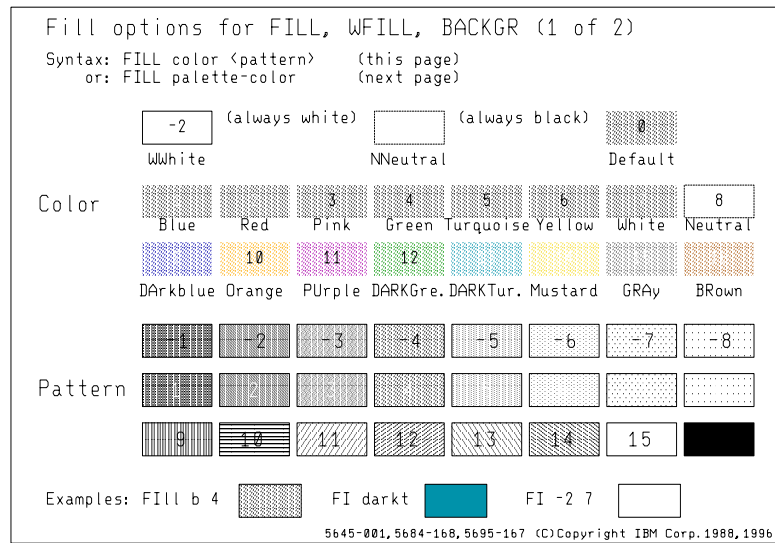


Figure 32. FILL help panel (first of two)

Using basic colors and patterns

The basic GDDM colors you can specify using FILL are those described in “Coloring text” on page 69 (the color name, or the numbers -2 to 16). All color devices support colors in the range 1 to 8; only workstations, such as the PS/2, support the values above 8. If the high-value colors are not available, the values are mapped onto existing colors.

The patterns available are more than you can get from basic GDDM, as follows:

- 0 to 8** GDDM pattern emulating darkening shades
- 1 to -8** OPS extension emulating lightening shades
- 9 to 14** GDDM line-composed patterns (mainly for charts)
- 15** No filling
- 16** Solid fill; the default value
- 1000 to 1032** Gray scale for PSEGs

If you specify a basic color, the default pattern is 16 (solid fill), as in:

```
fill darkblue            ;* fill with solid dark blue
```

Patterns in the range 0 to 16 are standard GDDM patterns. For example:

```
fill red 3
```

requests filling with red, and standard pattern 3.

Note that the GDDM patterns are **device dependent**. On some devices (such as the IBM 3279), the difference between pattern 1 and 8 is large; on others (such as a PS/2 with OS/2 GDDM link), the difference is quite small.

Here's a summary of the effect of all pattern-defining numbers:

0 to 8

Patterns of increasing darkness when used on a neutral background. (On a screen, fewer and fewer pels are lit.) OPS emulates this darkening also in PSEGs. See Figure 32.

-1 to -8

The further from 0 you get, the lighter the color. On a screen the light colors are emulated by putting the standard patterns 1 to 8 on top of a white background.

9 to 14

Mainly aimed at charts, allowing a clear distinction between the various data groups in black-and-white print.

1000 to 1032

Specifically aimed at PSEGs. They address a specific pattern, ADMDHPL, used to emulate gray levels in PSEGs. Pattern 1000 is empty, 1001 to 1031 give increasingly filled patterns, and 1032 is solid. In print, 1032 is all black. See Figure 33.

If your terminal supports pattern loading, the patterns in the range 1000 to 1032 are visible on the screen.

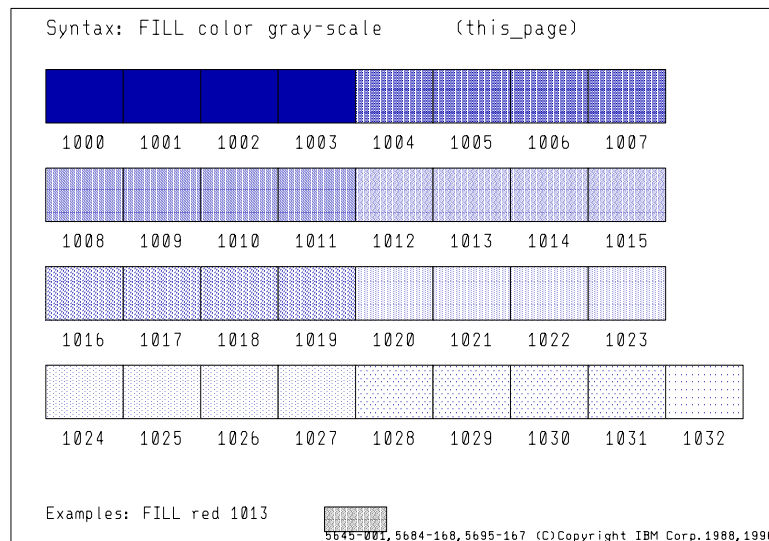


Figure 33. Patterns in range 1000 to 1032

In general, patterns -8 to 16 will give you what you need. Use the patterns above 1000 **only** when you want very specific control of a PSEG rendering.

Note: GDDM allows only one loaded pattern set at a time. You cannot use the 64-color palette (ADMCOLS) and the gray-level palette in the same picture.

Erasing areas before filling

Use the ERASE option to erase an area before filling. This means filling the area with Neutral (the device background color). For example:

```
fill pur 2 erase
```

You can specify ERASE with the standard GDDM patterns (0 to 16); for all other patterns, it's the default, and you can't override it. If you omit ERASE from a FILL command specifying a standard GDDM pattern, the underlying object or

using general colors

background will shine through the filled area. If MIX is ON (see “Controlling color mixing” on page 109), colors will also be mixed with solid filling (16).

Unspecific FILL attributes

The unspecific FILL attributes (color 0 and pattern 0) correspond to solid green on screens, halftoned black on impact printers, and solid black in PSEGs. To request these use:

```
fill 0 0
```

As explained in “Setting line attributes” on page 104, unspecific attributes let you recolor ADMGDFs loaded into the picture.

Using general colors with OPS

OPS recognizes a request for a general color described by the RGB model, when the color begins with an R and is followed by three groups of numbers separated by periods. For example:

```
fill r32.80.0  
fill r1.20.100
```

The three numbers denote percentages of each of the three primary colors—red, green, and blue—so each value must be in the range 0 to 100. Here are some examples:

Value	Meaning
r30.20.80	.30 red + .20 green + .80 blue (bluish gray)
r87.100.9	.87 red + 1.00 green + .09 blue (yellow)
r70.70.70	.70 red + .70 green + .70 blue (light gray)
r0.100.0	1.00 green (green)

A simple rule is: the higher the numbers, the lighter the color.

OPS also accepts the **3rgb** scheme, where r, g, and b denote *thirds* of full intensity. For example:

```
fill 3100          (1/3 red)  
fill 3031          (3/3 green + 1/3 blue)
```

This scheme covers 64 colors, and matches the GDDM ADMCOLSx image symbol set.

OPS recognizes a request for a CMY(K) color when the color begins with a C and has three or four groups of numbers following, each number representing a percentage of (in sequence) cyan, magenta, yellow, and black (K). For example:

Value	Meaning
c10.5.30	0.10 cyan + 0.05 magenta + 0.30 yellow
c12.100.0.11	0.12 cyan + 1.00 magenta + 0.00 yellow + 0.11 black

You can use the FROM and TO keywords on FILL to fade one color into another from the top to the bottom of an object. You can mix basic GDDM colors and general colors freely on these commands, but you cannot specify a basic color with a pattern. Here are some examples:

```
fill    from c11.0.32  to purple
fill    from r10.20.30 to c32.19.100
```

Note: With the exception of a background fading between basic GDDM colors, color fading is not displayed on the terminal, or in PSEGs generated by the PSEG command.

You can omit the TO color to fade to Neutral.

Controlling color mixing

If two colored objects overlap each other, and ERASE is not active, you can do one of three things:

1. Blank out the first object with the most recent object (overpainting)
2. Blank out the most recent object with the first object (underpainting)
3. Mix the colors

You can do this only when merging display points; if the objects are drawn with non-solid patterns (1 to 14) you will always be able to see through these patterns.

The rules for color mixing are:

- The primary colors Red, Green, and Blue are mixed additively.
- A primary color is mixed with a compound color (Pink, Turquoise, Yellow, or White) by additive color mixing of the primary color with the primary colors in the compound color. For example, Red + Pink = Red + (Red + Blue) = Pink.

You get the three possibilities by:

```
mix on    mixing
mix off   overpainting (the default)
mix over  overpainting
mix under underpainting
```

The following OPS presentation illustrates the rules of additive color mixing:

```
1 )ops reset; * this will set MIX OFF
   )ops mix on
   )ops ss admuutrp; size 12mm; col white
   )ops text 'Additive Color mixing' 50 &ymax-20mm ce
   )ops draw white
   )ops circle 40 40 40 fill red
   )ops circle 60 40 40 fill gre
   )ops circle 50 23 40 fill blu
1
```

Note: Underpainting is *not* supported on the 3270 PC/G(X).

Drawing wide lines

GDDM supports lines of only two widths, with one or two display points. However, OPS lets you emulate lines, both straight and curved, of almost any width, using WFILL. You can use WFILL either as a global command, or as a subcommand on graphics commands that support wide lines. See Figure 34 for some illustrations.

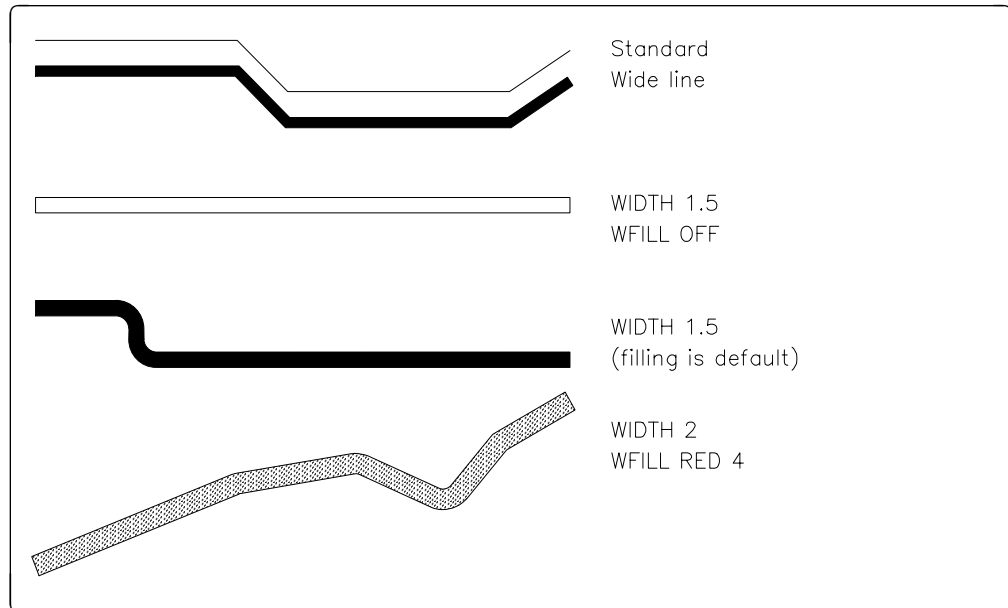


Figure 34. Some wide lines

Wide lines are usually filled with the color specified by DRAW, creating the illusion of a “wide brush” stroke. However, you can fill with special colors and patterns using the WFILL command, which has the same syntax as FILL. For example:

```
wfill red 5
wfill r20.30.5
```

In the same way as with DRAW and FILL, you can use WFILL globally (as an independent command) or locally (as a subcommand on a graphics command).

Setting points on the screen

The POINT command lets you set one or more points (dots) on the screen by specifying coordinate pairs (x,y). The following command puts three points on the screen:

```
point 10 10 20 25 40 35
```

The points are drawn with the line attributes set by DRAW, but only the color and the width are important. A normal line width will make a small point, a wide line width will make a bigger point.

You can use DRAW (DR as an abbreviated form) as a subcommand:

```
point 25 25 dr red 2
```

Drawing lines

The LINE command draws a line through one or more points. The following example:

```
line 10 10 20 25 40 35
```

draws a line through the three points created in “Setting points on the screen” on page 110. You can put commas between the pairs to increase readability, as in:

```
line 10 10, 20 25, 40 35, 15 55
```

If you omit the last coordinate (the last y-value), OPS assumes it to be identical with the last y-value but one, and the line ends with a horizontal line. So, for a horizontal line, you only need three numbers:

```
line 0 55.3, 100
```

You can specify the coordinates as described in “Using the coordinate system” on page 60. If you want to draw a (horizontal) line below some text, use:

```
1 ...
  This is a major header
  )ops line 0 &y-3p 100
  ...
```

This draws a line across the screen three display points below the bottom line of the character box.

Defining the line

After the coordinates, you can specify one or more operands (keywords) or subcommands, in any order, to define the line in more detail.

Moving the current point

Use the keyword SET to move the current point to the last point of the line. You can then use this as a starting point for new text or more graphics, as in:

```
text 'A' 10 10 h 5
line &x+5mm &y+5mm &x+20mm set
text 'B' &x &y-5mm
line &x+5mm &y+5mm &x+20mm set
text 'C' &x &y-5mm
...
```

This writes the letters with horizontal lines between them.

Setting the line attributes

Use DRAW (DR) as a subcommand to set the attributes of the line, as in:

```
line 10 10 40 50 draw blu 2 1
```

This produces a dotted blue line, two points wide.

Setting the line width

Use the keyword WIDTH (W) to set the width of the line, as in:

```
line 10 10 40 50 width 3mm
```

Filling wide lines

Use the subcommand WFILL to specify how you want to fill wide lines. For example:

```
line 10 10 40 50 w 1cm, wfill y 3, draw b
```

draws a line one centimeter wide, filled with pattern 3 in yellow, with the outline drawn in blue. If you omit WFILL, the line is filled with the line color set by DRAW.

Creating rounded corners

Use the keyword RADIUS (R) to create rounded corners. For example:

```
line 10 10, 10 50, 90 50, w 5mm, r 2cm
```

draws a circular arc with a radius of 2cm at the corner point (10,50). The width is 5mm.

The radius cannot be more than half the distance between the corner point and the nearest point. For example, if the line limits a square, the radius can be at most half the length of the side, creating a circle. If you specify a radius larger than the limit, OPS will reduce it to the maximum without giving an error message. See Figure 35 for some examples.

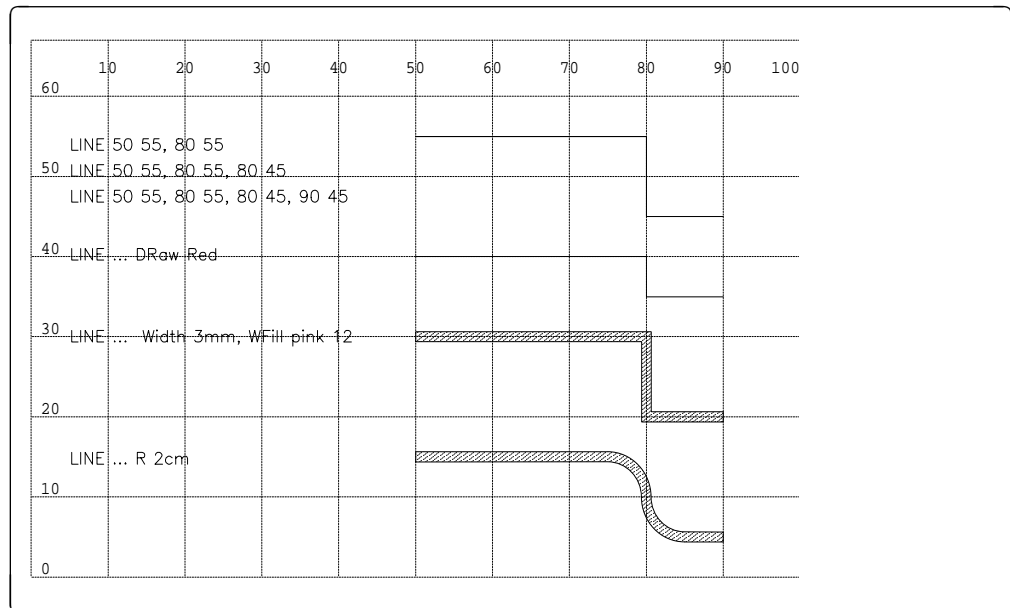


Figure 35. Sample lines

Drawing curves

Use the CURVE (CU) keyword to draw the line as a “freehand curve,” in which all curve segments are elliptical arcs. For example:

```
line 5 5, 20 5, 20 40, 60 40, 60 5, cu, w 3p
```

CURVE and RADIUS (see “Creating rounded corners” on page 112) are two different ways of drawing the line; you cannot specify them both at the same time.

You can also use CURVE as a command, as in:

```
curve 5 5, 20 5, 20 40, 60 40, 60 5, w 3p
```

This produces exactly the same curve as the LINE command above.

Figure 36 illustrates the basic rules about drawing curves.

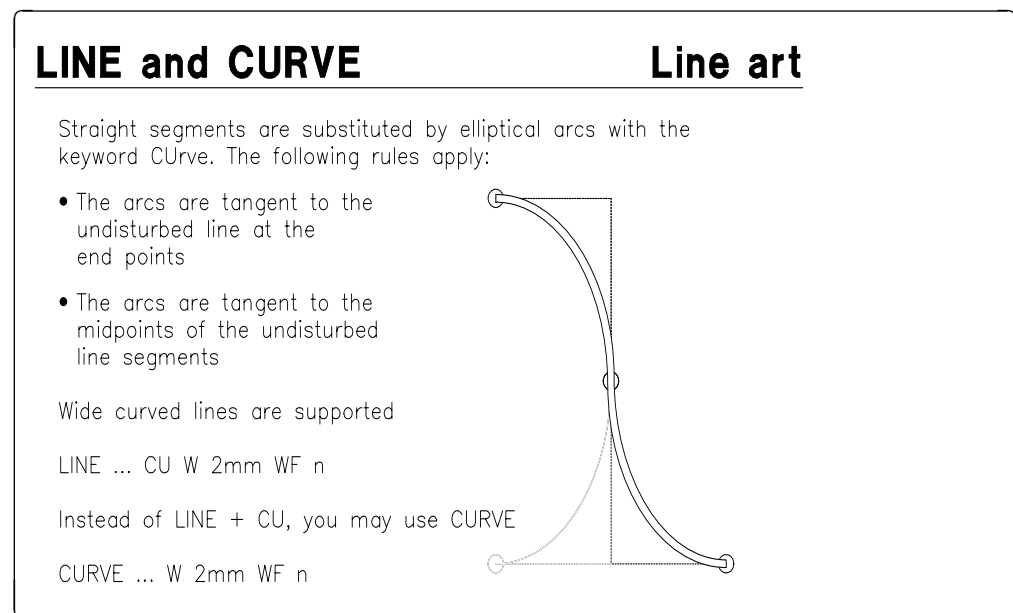


Figure 36. Examples of LINE and CURVE

Turning and shearing lines

You can turn and shear lines and curves using the TURN and SHEAR operands. For example:

```
line 10 10 40 turn 45
line 20 20 40 60 w 1cm shear 30
```

The values after TURN and SHEAR are **degrees** of arc. TURN is measured counterclockwise; SHEAR is measured clockwise.

drawing polygons

Changing the center of the line

By default, the geometrical **center** of the line (or curve) is the **fix-point**. You can choose another fix-point with the REFP operand, as in:

```
line 30 30 30 +5cm turn 15, refp 30 30
```

This draws a line from (30,30), 5 centimeters long and turned 15 degrees about (30,30).

You can also use the following special values with REFP:

- CC** The center of the object (the default)
- CO** The first specified coordinate
- RI** The last specified point

For example:

```
line 30 30 30+5cm tu 15, refp co
```

produces the same effect as the previous LINE command.

See Figure 37 for an example of the effect of REFP on shearing.

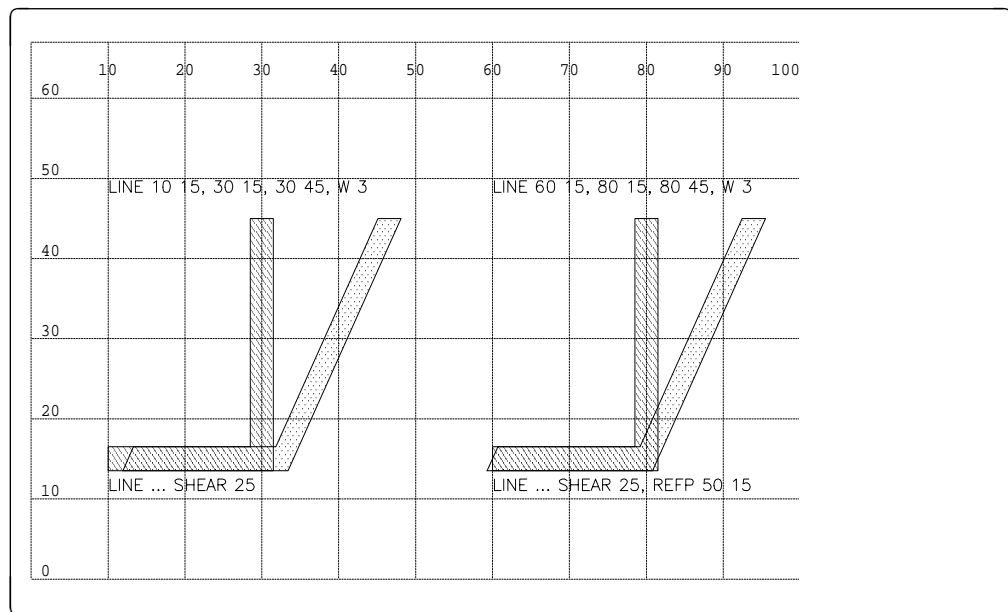


Figure 37. SHEAR and REFP

Drawing polygons

The POLYGON command draws polygons. For example:

```
poly 10 10 50 50 90 10, fill r20.20.20, draw y 2
```

draws a dark gray triangle with a yellow outline.

Figure 38 on page 115 shows some polygons.

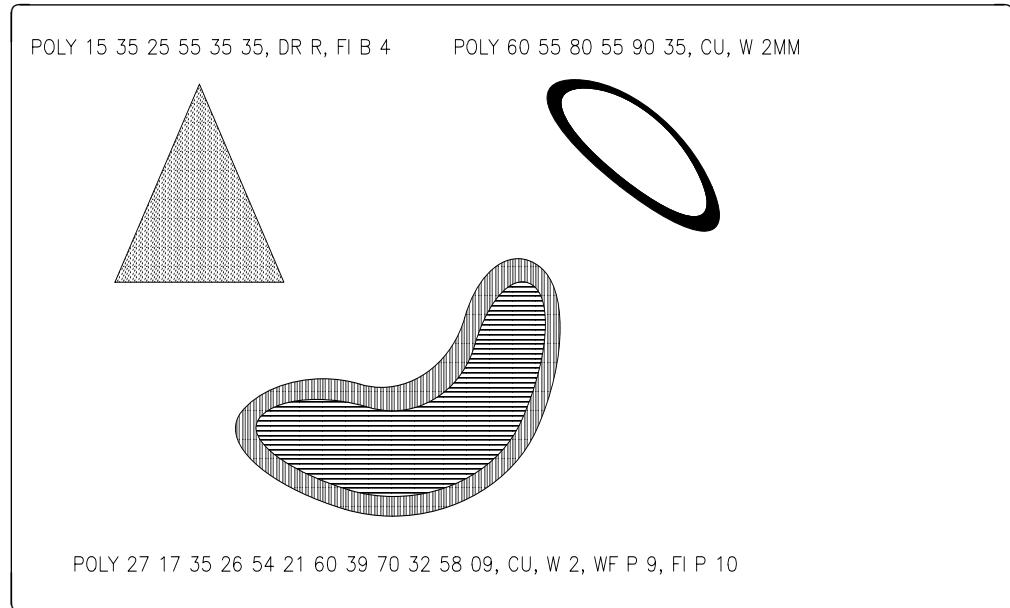


Figure 38. Sample polygons

You can turn and shear polygons. The default fix-point is the center of the object, calculated as the mean value of the coordinates. You can also use REFP to choose another center, as in:

```
poly 20 30,50 60,80 30 turn 45          ;* turn about the center
poly 20 30,50 60,80 30 turn 45 refp 20 30 ;* turn about (20,30)
```

The first example turns the triangle about the center, calculated as $((20+50+80)/3, (30+60+30)/3)$ or (50,40). The second example could also be written as:

```
poly 20 30,50 60,80 30 turn 45 refp co    ;* turn about (20,30)
```

with CO being a mnemonic for the COrner, or first point in the command.

Drawing arrows

The ARROW command generates arrows between two points: single, broken, thin, wide, or with different heads and tails.

You must provide the pair of coordinates for the start of the arrow (tail) and the end of the arrow (head). You can omit the last y-value when the last or only section is horizontal, so the simplest ARROW command is:

```
arrow 20 30 90
```

which draws a single-line arrow from (20,30) to (90,30), with a head at (90,30). The line is drawn using the current line attributes and the head is solidly filled with the current line color. This is shown in the first example in Figure 39 on page 116.

You can draw arrows without a tail, and with a right triangle as the head, with commands such as:

```
arrow 10 10, 10 40, 90 40, w 5mm, r 2cm
```

drawing arrows

This draws a 5 millimeter arrow with a rounded corner in (10,40). By default, wide arrows are filled with the line color set by DRAW. You can change this using WFILL.

Figure 39 illustrates the simplest arrows.

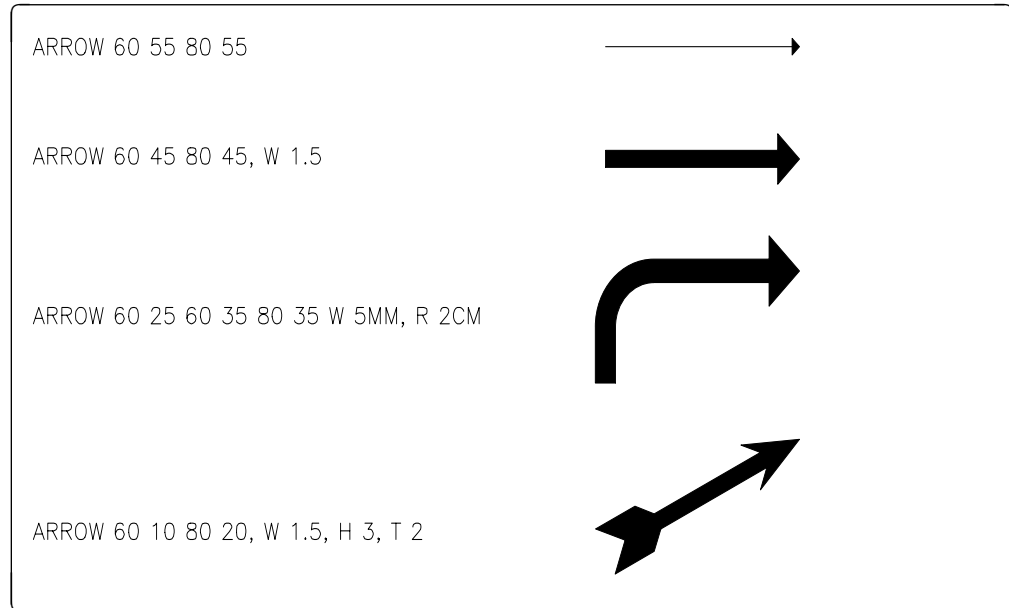


Figure 39. Some simple arrows

Controlling arrow heads and tails

Beside the subcommands and keywords supported by LINE (WIDTH, RADIUS, CURVE, DRAW, WFILL, TURN, and SHEAR), you can use the following, which are unique to ARROW: HEAD, TAIL, DOUBLE, and HSCALE.

You can choose between six heads, using the keyword HEAD and a numerical value from -1 to 5. The default head is a right triangle (HEAD 0). HEAD 1 applies to wide lines, and narrows them to a point, as in:

```
arrow 20 20 50 55 w 3, head 1
```

You can set the tail style using the keyword TAIL with a numerical value from 0 to 4. The default is no tail (TAIL 0). HEAD 1 and TAIL 1 go together, while you can use TAIL 2 together with HEAD 0, 2, or 3. HEAD and TAIL can be abbreviated to H and T, as in:

```
arrow 20 20 50 55 h 3 t 2
```

You can display the available heads and tails by typing ? after H or T, or in a position where you might put a new operand. For example:

```
arrow ?  
arrow 10 10 40 40 h ?  
arrow 10 10 40 40 h 4 ?
```

Figure 40 on page 117 shows the resulting display:

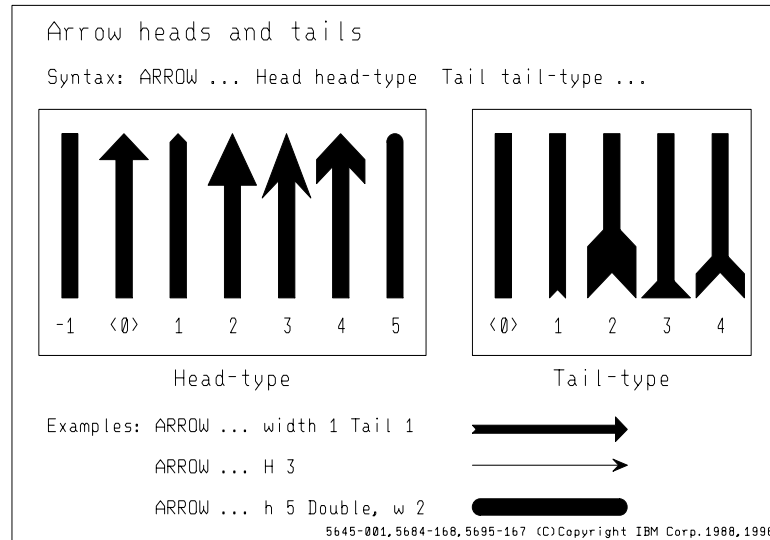


Figure 40. Help panel for arrow heads and tails

If you want heads at both ends, use the keyword **DOUBLE (D)**, as in:

```
arrow 20 20 50% 50% d w 2mm
```

Heads and tails are drawn with fixed sizes in proportion to the width of the arrow, but at least 1.5 units wide. **HEAD 1** and **TAIL 1** are as wide as the arrow; other heads and tails are three times as wide as the arrow. You can change this using the keyword **HSCALE (HS)**.

HSCALE adjusts the proportion between the width of the head and tail and the width of the arrow, as in:

```
arrow 5 5 60 60 w 2mm, h 2, hs 2
```

This produces a head twice as big as the default. The default corresponds to **HSCALE 1**: three times the width of the arrow, or 6mm. The head in the example above is therefore 12mm wide.

Figure 41 on page 118 illustrates the effect of **HSCALE**.

drawing arrows

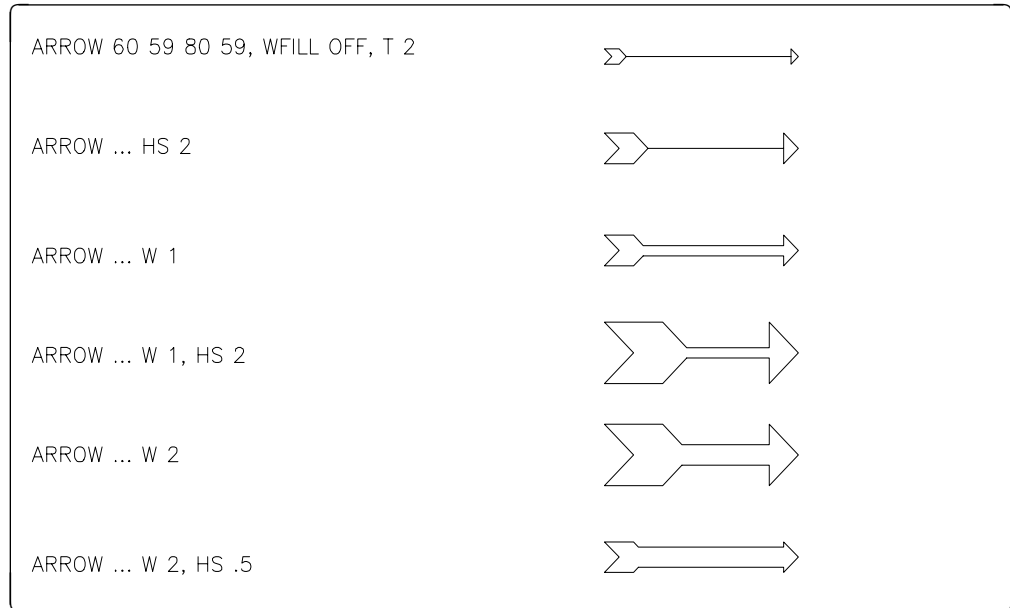


Figure 41. Scaling arrow heads and tails

You can turn, shear, and use REFP to change the fix-point, just as you can with LINE. Figure 42 illustrates the effects you can achieve with TURN and SHEAR.

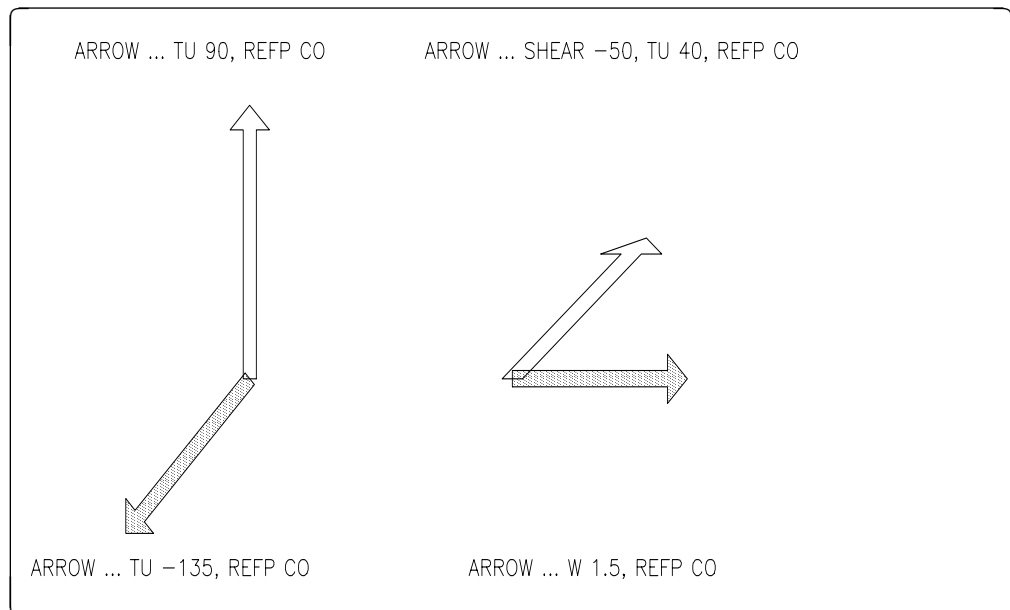


Figure 42. Sample arrows with TURN and SHEAR

Drawing boxes

Use the BOX command to draw boxes, either by itself or as a subcommand on TEXT.

Here's an example of the simplest use of BOX:

```
box 20 30 10
```

This draws a square box with its center at (20,30) and a side length of 10.

Boxes other than squares require at least 4 operands, the fourth specifying the height of the box, as in:

```
box 50% 50% 80mm 50mm
```

This draws a box with its center in the middle of the screen, a width of 80mm, and a height of 50mm.

Controlling the alignment of boxes

The default alignment point for boxes is the center, consistent with other commands. However, using the alignment code, you can align boxes in other ways, as in:

Command	Alignment
BOX CO 10 10 30 20	Lower-left-hand corner at (10,10)
BOX CO 40 10 -30 20	Lower-right-hand corner at (40,10)
BOX CO 40 10 -30 -20	Upper-right-hand corner at (40,10)
BOX CE 20 10 30 20	Bottom-center at (20,10)

When the alignment code is CC (the default), the signs of width and height are irrelevant.

When the alignment code is CO, OPS draws the box from the alignment point in the directions indicated by the signs of width and height. If the width is positive, the box is drawn to the right from the alignment point; if it is negative, the box is drawn to the left. Similarly, for height, positive corresponds to upwards.

When the alignment code is CE, the sign of width is immaterial. The sign of height determines whether the box is drawn upwards or downwards relative to the alignment point.

Figure 43 on page 120 illustrates various alignment codes.

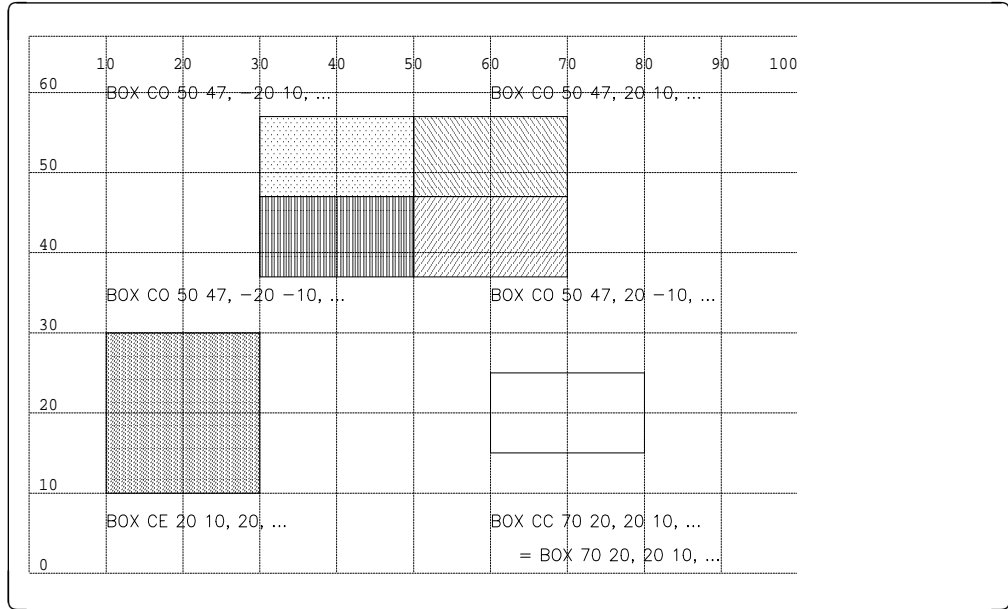


Figure 43. BOX with various alignment codes

Adding a 3-D effect to boxes

You can add a 3-D effect to boxes using two positional operands: the 3-D depth of the box and the angle between the x-axis and the projected z-axis (depth) in degrees counterclockwise. The default angle is -45 , creating a 3-D box seen from below and from the right. Here is the command:

```
box 50% 50% 80mm 50mm 5mm
```

The following example:

```
box 50% 50% 80mm 50mm 5mm 45
```

draws a 3-D box with a positive angle of 45 degrees; that is, the box is seen from above and from the right.

Controlling attributes of boxes

Use the RADIUS (R) and WIDTH (W) operands to put rounded corners and wide boundaries on your boxes. Use DRAW, FILL, and WFILL to control coloring and drawing attributes. Here are some examples:

```
box 40 40 30 fi red           ;* a red square
box 40 40 30 20 w 5mm        ;* rectangle with a wide frame
box 40 40 30 20 5 dr y       ;* a yellow 3-D box
box 40 40 30 20 5 r 1cm      ;* 3-D box with rounded corners
```

Figure 44 on page 121 illustrates the effect of the various operands.

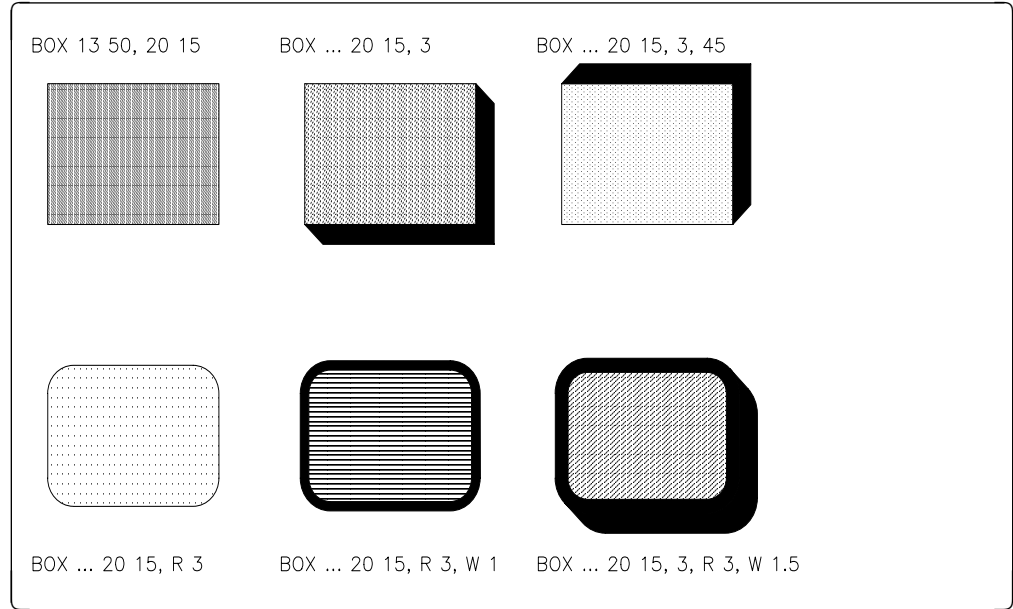


Figure 44. Sample boxes

You can fill the box in one color and pattern (using FILL), and any wide border in another color and pattern (using WFILL). For example, the command:

```
box 40 40 30 20 4, w 5mm, wf 3211, r 2cm, fi 3121, dr whi 2
```

draws a 3-D box with a wide frame and rounded corners. The box and the frame are filled with different palette-colors, and the outline of the frame is drawn with a width of two pels. Figure 45 illustrates something similar.

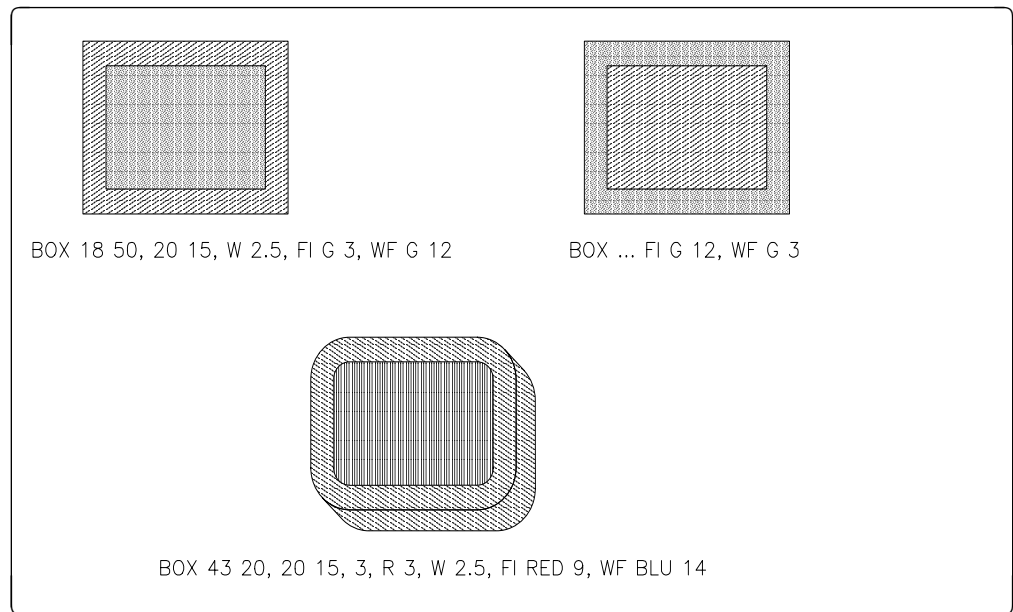


Figure 45. Filling boxes

Turning and shearing boxes

You can turn and shear the completed box, as shown below:

```
box 30 30 20 turn 45, shear 45
```

TURN turns counterclockwise; SHEAR shears clockwise from “12 o’clock.” Angles are specified in degrees. See Figure 47 on page 123 for some examples.

Whatever the alignment code, TURN and SHEAR use **the center** of the box as fix-point. You can change this with the REFP operand, specifying either a pair of x- and y-values, or one of the mnemonics: CC, CO, CE, or RI.

Figure 46 shows some examples of TURN and SHEAR with alignment other than the center.

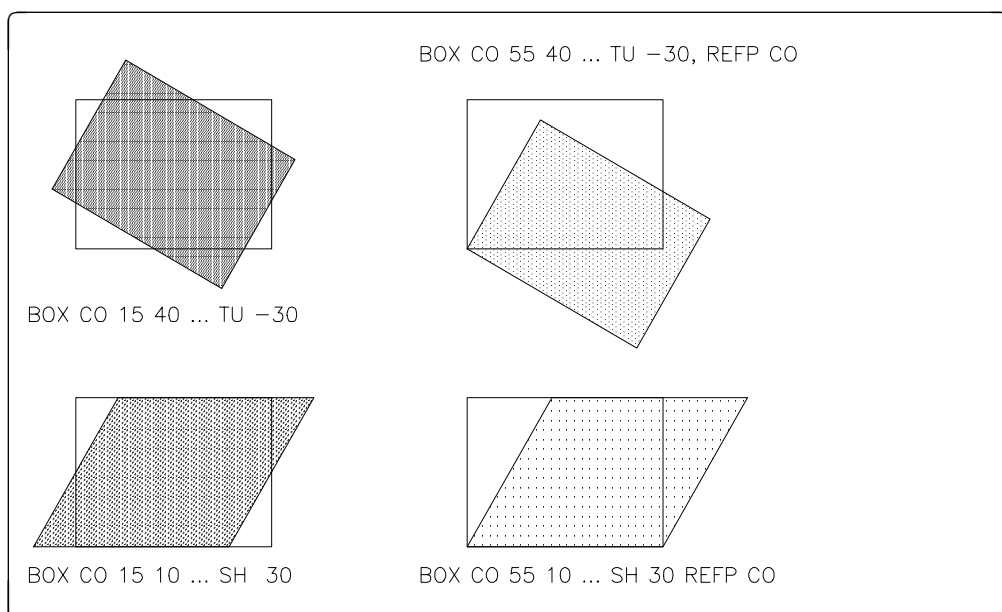


Figure 46. Samples of TURN and SHEAR with BOX

Using BOX as a subcommand of TEXT

The BOX subcommand on the TEXT command includes all the operands allowed on the primary BOX command except for the positional operands. The center of the box will be at the center of the text. You can also specify dimensions as a factor of the character boxes in the text (see “Manipulating text strings” on page 86), as illustrated in Figure 25 on page 93 and Figure 47 on page 123.

TURN and SHEAR influence both text and box.

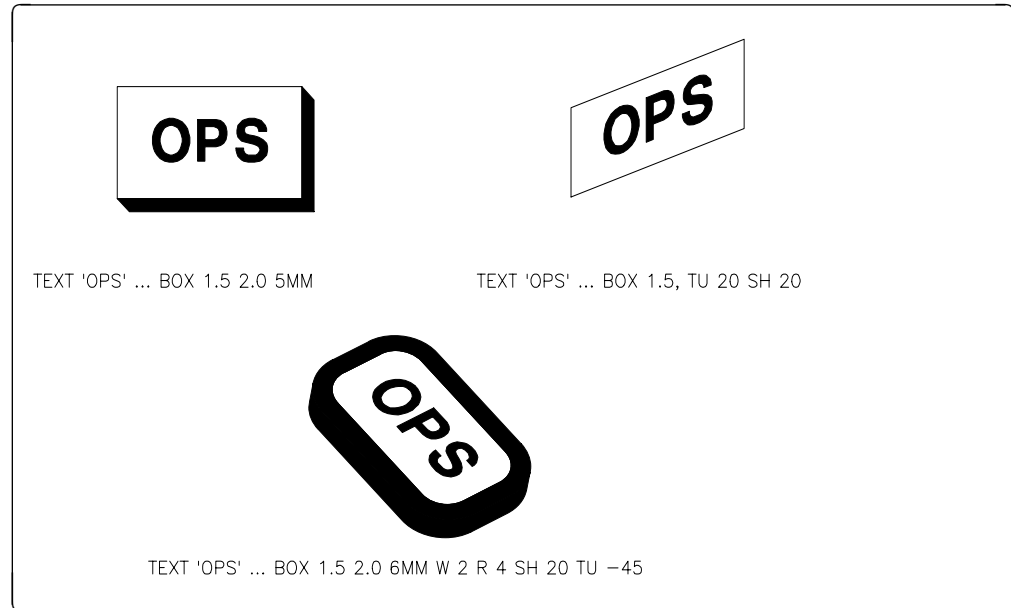


Figure 47. Using BOX with TEXT

Drawing circles

Use the CIRCLE command to create circles or ellipses with the following positional operands:

- The center coordinates of the circle
- The diameter in the x-direction
- The diameter in the y-direction (ellipses only)

Here are examples of drawing a circle and an ellipse:

```
circle 50 50% 80mm      ;* a circle
circle 50 50% 80mm 40mm ;* an ellipse
```

The CIRCLE command also supports the subcommands DRAW (D), FILL, and WFILL, and the keyword operand WIDTH (W). Here's an example with all the parameters:

```
circle 50 50% 30 w 4% wf b, fi r 3, dr y 2
```

You can draw wide boundaries with the specified width along the full circumference. When you draw an ellipse with a wide boundary, it doesn't look like a circle tilted from the picture plane (which would have a boundary of varying width). If you want that, use GDFSAVE and GDF, as described in Chapter 9, "Displaying pictures" on page 135.

You can turn and shear circles and ellipses using TURN and SHEAR, as in:

```
circle 40 40 40 20 shear 30, turn 20
```

Figure 48 on page 124 shows various circles and ellipses.

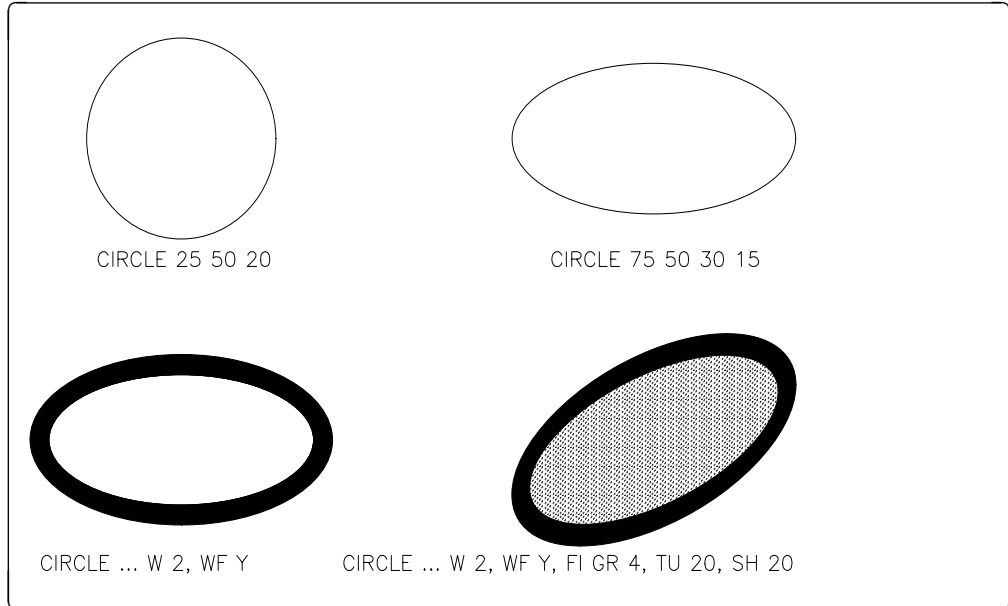


Figure 48. Circles and ellipses

Drawing arcs

Use the ARC command to draw circular arcs with a center and diameter specified as positional operands, as in:

```
arc 50% 50% 90mm
```

The default arc covers 360 degrees from the x-axis (0 degrees) counterclockwise (a full circle). You can get smaller arcs using the operands V1 and V2, as in:

```
arc 50% 50% 80mm v1 45 v2 135
```

See Figure 49 on page 125 for an illustration.

You specify the angles in degrees counted counterclockwise from the x-axis (0 degrees). The arc is **always** drawn from V1 to V2. If V2 is numerically bigger than V1, the drawing will be counterclockwise (as in the example); otherwise it will be clockwise. Both V1 and V2 must be between -360 and +360, and the difference between the two must not exceed 360 degrees. Default values are 0 (V1) and 360 (V2) degrees.

Use the keyword SET to save the endpoint of the arc as the current point. You can then write text relative to the point, or start a new line, as in:

```
arc 40 40 30 v2 45 set
line &x &y 10 &y
```

Use WIDTH (W) to create wide arcs, and the subcommands DRAW (D), FILL, and WFILL to control color.

Note: ARC supports perfect circular arcs only. Use CURVE to draw elliptical arcs.

Drawing pie slices

Use the SECTOR command to draw circle sectors (pie slices). The basic syntax is the same as for ARC, but wide lines are **not** supported and FILL is a valid subcommand. Here's an example:

```
sector 30 50% 60mm v1 10 v2 -10, fill red 2, dr w
```

Note: SECTOR only supports perfect circle sectors. To design more complicated pie charts, use the GDDM Interactive Chart Utility.

See Figure 49 for an illustration.

Drawing rays

Use the RAYS command to draw rays from a central point.

The syntax is the same as for ARC, but RAYS also supports elliptical figures, using a fourth positional operand that specifies the diameter in the direction of the y-axis. Here's an example:

```
rays 50 50% 50 30 v2 180, draw red 2 1
```

RAYS doesn't support wide lines. The rays are controlled by DRAW.

The rays are evenly distributed with 10 degrees between each ray. The even distribution is also used when the rays cover an elliptical area. You can get another angle between the rays by using the keyword operand DV, as in:

```
rays 40 30 90mm dv 5
```

Figure 49 illustrates the various circular functions (ARC, SECTOR, and RAYS).

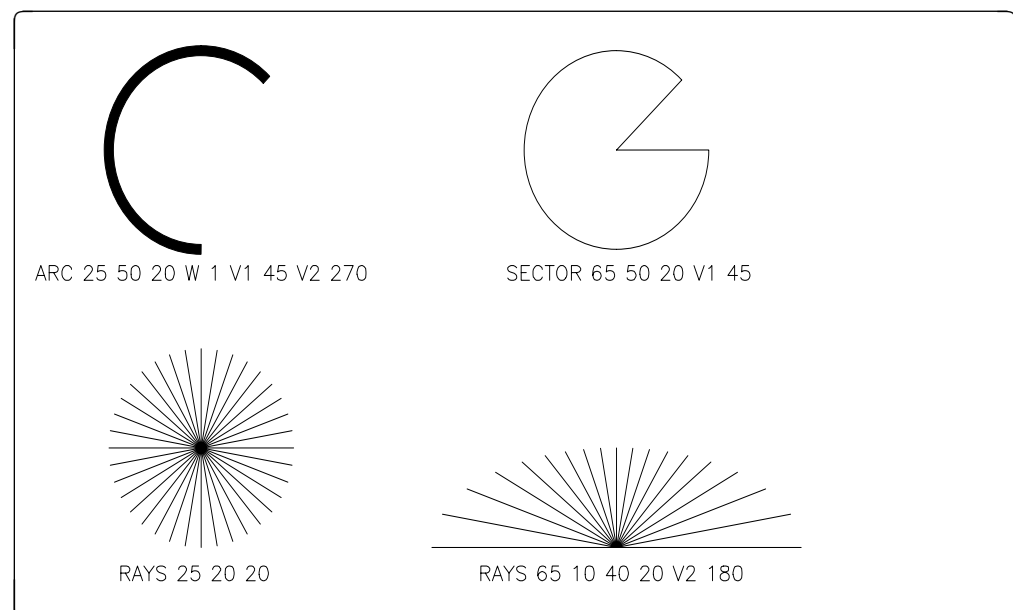


Figure 49. Circular commands: ARC, SECTOR, RAYS

Drawing DASD symbols

Use these commands to illustrate computer systems, as follows:

DASD The familiar symbol for a DASD (direct access storage device)

DASDF The symbol for a file on the DASD

DASDX The file separator line across a DASD

See Figure 50 for examples.

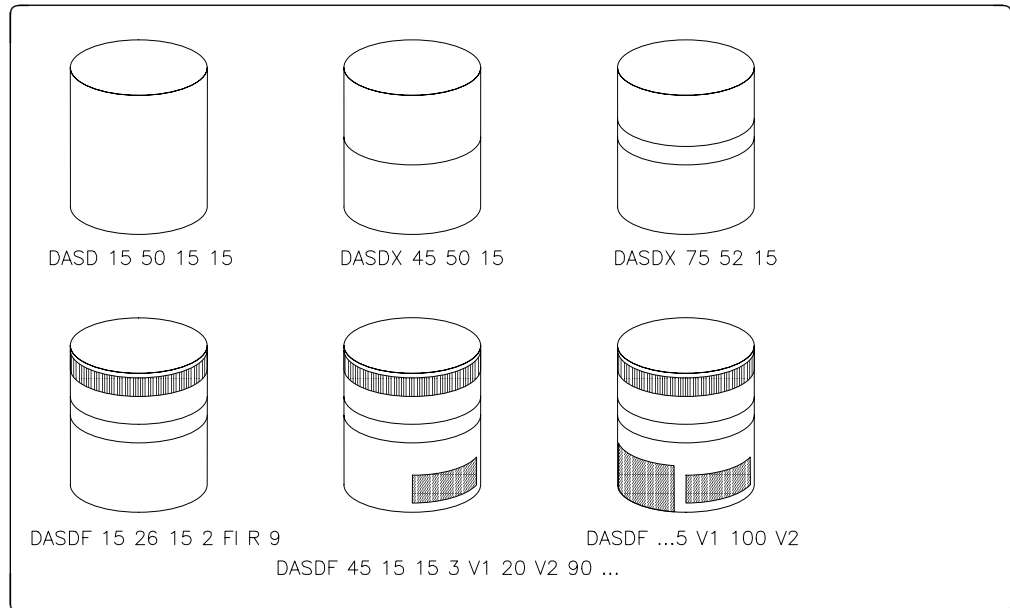


Figure 50. DASD commands: *DASD*, *DASDX*, *DASDF*

You must specify at least the center coordinates and the width of the cylinder for all three commands. For *DASD* and *DASDF*, which draw surfaces, you must also specify the height of the surface. Note that the height on *DASD* is the height of the cylindrical surface, not of the full object. This allows you to create very thin dasd objects, such as coins, by specifying a near-zero height and forgetting about the elliptical part. This is illustrated in Figure 51 on page 128.

The final operand on all three commands is the aspect-ratio of the elliptical terminal surface (the relation between the short and the long axis). The default is 0.4.

Here are some examples of the three commands:

Command	Effect
DASD 40 30 10 20 0.5	A DASE with the center in (40,30), a width of 10, a height of 20 for the cylindrical part, and an aspect ratio of 0.5 in the elliptical terminal surface.
DASDF 40 30 10 20 0.5	A DASE file covering the entire cylinder from the previous DASE command.
DASDX 40 30 10 0.5	An elliptical file separator line in the middle of the DASE from the previous examples. The center of the ellipse is at (40,30). You do not need to specify the height—it is only a line.

DASDF includes two operands, V1 and V2, specifying the angles between which the file is to be placed. V1 and V2 can be either positive or negative, but they must have the *same sign*. Both must be between -180 and $+180$. 0 degrees corresponds to the right edge of the cylinder (the intersection with the x-axis), 180 or -180 corresponds to the left edge. No matter what the sign of V1 and V2 is, the DASE file is drawn on the visible side of the DASE. Here's an example:

```
dasd 40 30 20 30 ;* a DASE
dasdf 40 25 20 3 v1 45 v2 110 ;* a file on the DASE
dasdf 40 35 20 2 v1 20 v2 80 ;* a file on the DASE
```

The default values for V1 and V2 are 0 and 180 degrees, corresponding to a DASE file that covers the entire cylindrical surface. See Figure 51 on page 128 for some examples.

DASE and DASDF support the subcommands DRAW and FILL; DASDX supports DRAW. None of them supports wide lines. Here are some examples:

```
dasd 50 50% 40mm 60mm fi 3122 dr w 2
dasdf 50 40% 40mm 4 v1 20 v2 80 fi red
dasdx 50 50% 40mm dr w 2
```

All the commands support rotation and shear (TURN and SHEAR). The following example draws a coin askew:

```
dasd 50 30 20 2 fi 3220, dr w, turn 30
```

You can also make the color of the edge of the coin a little darker, as in:

```
dasd 50 30 20 2 fi 3220, dr w, turn 30
dasdf 50 30 20 2 fi n, dr off, turn 30
dasdf 50 30 20 2 fi yel 2, dr off, turn 30
```

The second of the commands above draws over the light yellow cylinder surface from the DASE command with black, creating the darker yellow of DASDF.

Figure 51 on page 128 shows examples of DASE and DASDF commands.

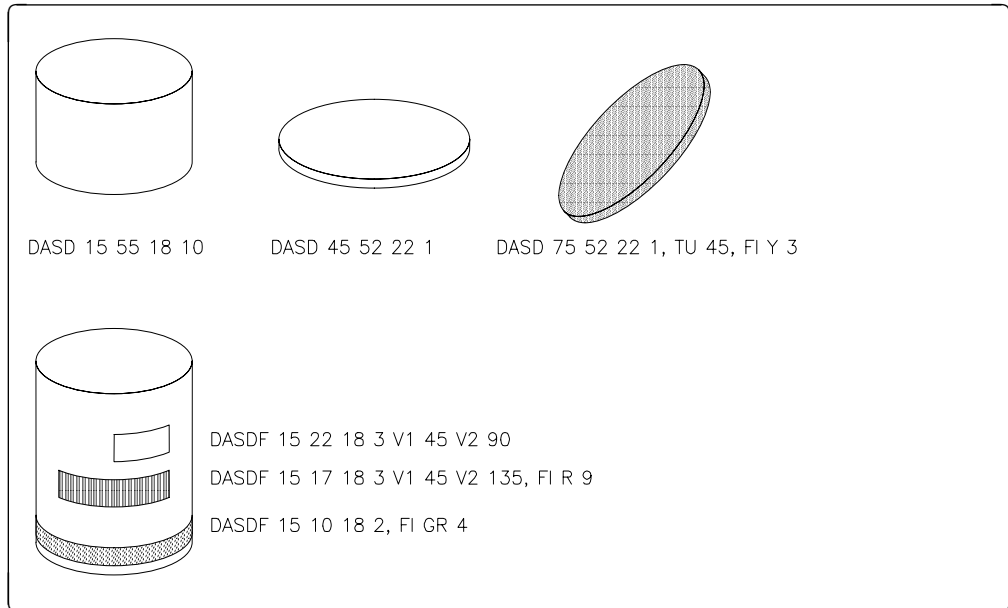


Figure 51. Samples of DASD and DASDF

Grouping objects together

Use the AREA command to group several objects into one graphics object and then fill the area occupied by the objects, as illustrated in Figure 52.

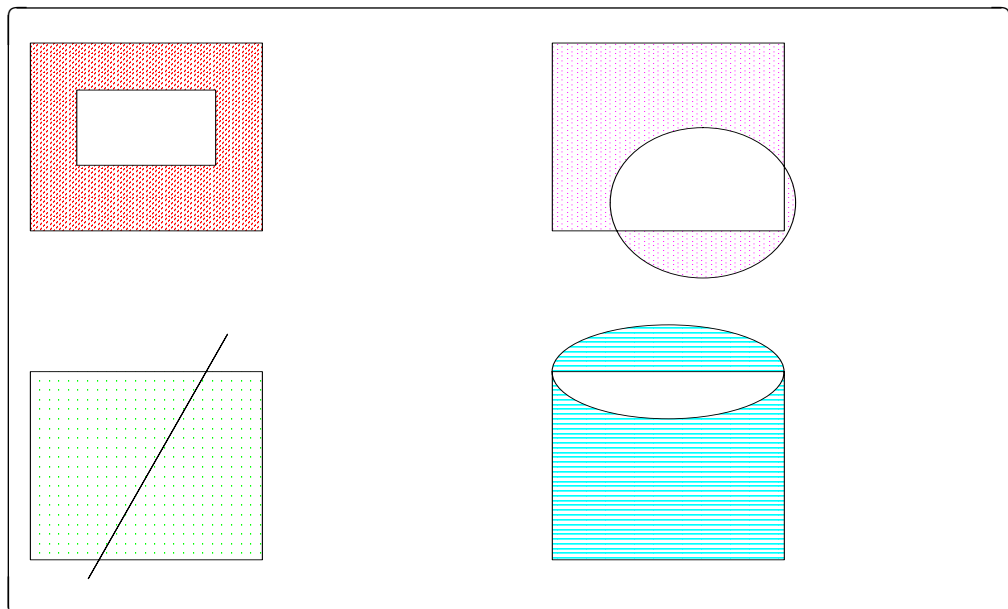


Figure 52. Examples of area filling in GDDM

You must set the area attributes before filling, and you cannot change them during the filling. You can, however, change the line attributes (such as the color of the line) at any time. Within an AREA group, wide lines are not filled; they represent

an area limited by two curves and it is not possible to fill more than one area at a time.

The AREA command uses the keywords BEGIN and END around the graphics commands that draw the different objects. You can specify FILL and DRAW in the first AREA command.

Here's an example:

```

1 )ops reset; prefix %
  %circle 50 40 40 20 fi y 3           ;* A yellow ellipse
  %area fill red 5, draw whi 2         ;* BEGIN is default
  %poly 10 10 50 50, 90 10           ;* a triangle
  %box 50 30 30mm, draw blu 2        ;* with a hole
  %area end
1

```

Figure 53 shows the result.

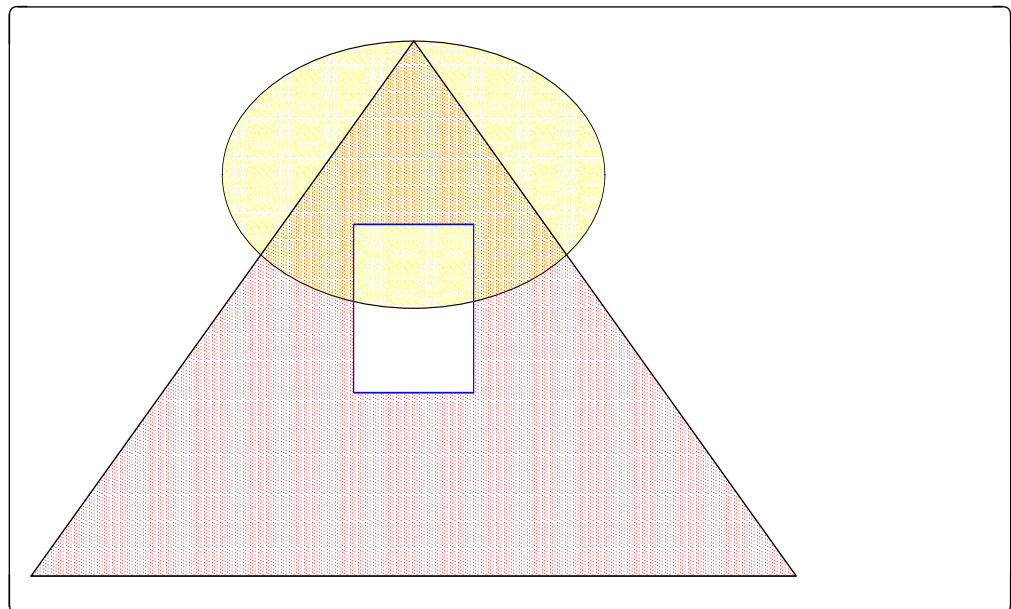


Figure 53. Result of AREA example

You always use AREA with filling; any attempt to do so without specifying FILL, either as a primary command or as a subcommand in AREA, raises an error.

You can use only line art functions in an AREA block; you can't use TEXT, TURN, or SHEAR.

Note: You can use AREA as an infile command *only*. You must close the AREA command using AREA END before displaying the page; that is, before any WAIT, FORCE, or end-of-page commands.

Common additional features for fillable objects

You can erase fillable objects (ERASE), and draw shadows behind them (SHADOW).

Erasing fillable objects

The ERASE (E) option is valid on all commands creating fillable objects, and on the picture commands described in Chapter 9, “Displaying pictures” on page 135.

When used with line art, such as boxes and circles, it erases all underlying objects from the fillable area before filling it. This is useful when you want a sparse pattern to appear as if it were opaque.

Here are some examples:

```
poly 10 10 50 50 70 20 erase fill red 11  
line 10 10 50 50 70 20 w 3mm e wfill yellow 8
```

ERASE is the default for line art when you have a background active, or when you ask for a shadow on an object.

Drawing shadows behind fillable objects

Use SHADOW to draw a shadow behind any of the fillable objects. (The non-fillable objects are DASDX, RAYS, and POINT.)

The shadow is drawn as a copy of the object, offset +1 in the x-direction and -1 in the y-direction. The shadow is drawn **before** the object is turned or sheared, so the shadow turns and shears with the object.

The shadow is filled with white if there is no filled background, or with “always black” (NNeutral) if there is. This way the shadow is always visible on the screen whether there is a background or not. In print or plot, the shadow always appears black.

When you use SHADOW, ERASE is selected by default. This erases the area covered by the object.

Figure 54 on page 131 shows some examples of shadows.

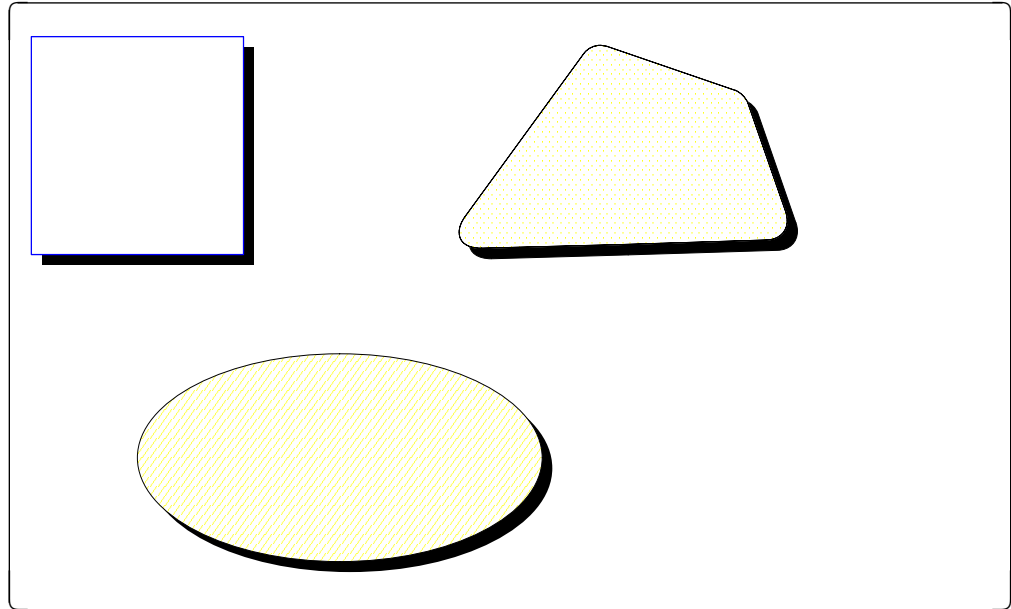


Figure 54. Shadow samples

Displacing objects

Use **OFFSET** as a primary command or as a subcommand on any object command (**TEXT**, **BOX**, **CIRCLE**, **GDF**, and so on) to displace text and objects by a specified amount. Here are two examples:

```
)ops offset 20 -5                ;* move following objects
                                   ;* 20 in x, and -5 in y
)ops box ...
)ops text ...
)ops offset 0                    ;* reset offset to 0 0
)ops box ... off -6              ;* displace box -6 in x
```

Use the primary **OFFSET** command to move larger groups of objects. However, be careful when you are working interactively, because the objects you draw are also subject to the current **OFFSET** setting. If you move such object commands out of the **OFFSET**-block, they will be positioned without offset.

The primary **OFFSET** command also affects infile text, as in:

```
)ops ss admuksf; h 5
This is just infile text
)ops offset +5 -4
and this text will be indented relative the previous
lines and spaced 4 down.
)ops off 0                        ;* OFFSET may be shortened to OFF
```

Use the local **OFFSET** command to repeat an object at different positions, as in:

```
set mybox = str(20 55 20 10 dr red 1 2) ;* basic box
box &mybox
box &mybox off 25                  ;* same box offset 25 in x
box &mybox off 50                  ;* same box offset 50 in x
```

displacing objects

This example draws three boxes separated by 25 units in x (the dashed boxes in Figure 55 on page 132).

The following example draws 3 vertical parallel lines:

```
start 10 20                                     ;* Set current point to 10 20
line &x 20, &x 30, offset 2, dr red 1 2, set    ;* Set current x to 12
line &x 20, &x 30, offset 2, dr red 1 2, set    ;* Set current x to 14
line &x 20, &x 30, offset 2, dr red 1 2, set
```

Each line is drawn at the current x, offset 2 to the right. It then sets the last point to be the new current point, allowing the same command to be reused indefinitely. The result is shown as dashed lines in Figure 55.

The final example shows you how to use OFFSET to introduce a local coordinate system. When you add a figure measured from, say, a drawing, it is often convenient to have a local origin somewhere in the object. The example draws a butterfly with the local origin at the center:

```
line 0 0,10 5,10 -5,-10 5,-10 -5,0 0 off 50 20
```

Global offset and local offset added together give you a total offset. You can use global offset around groups of commands with or without local offset to move the group of objects just the amount you request.

Figure 55 shows the three last examples with a global offset of (15,-15). The objects in dashed lines are the original objects without global offset.

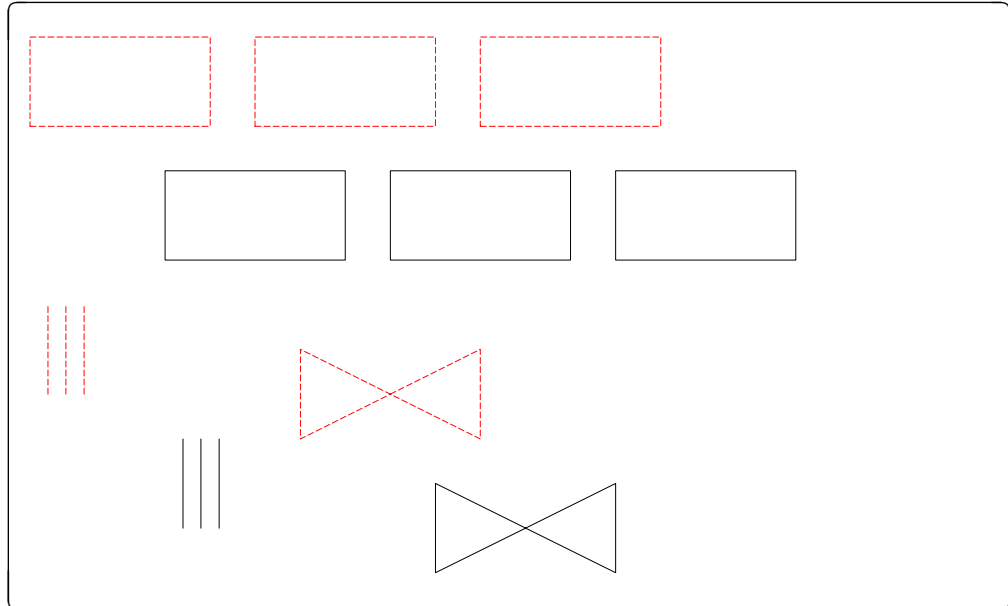


Figure 55. Sample with local and global offset

Filling the background

Use BACKGR to fill the background, as in:

```
backgr red 3
```

BACKGR has the same operands as the FILL command. Figure 56 includes a background created by:

```
backgr from nn to ww
```

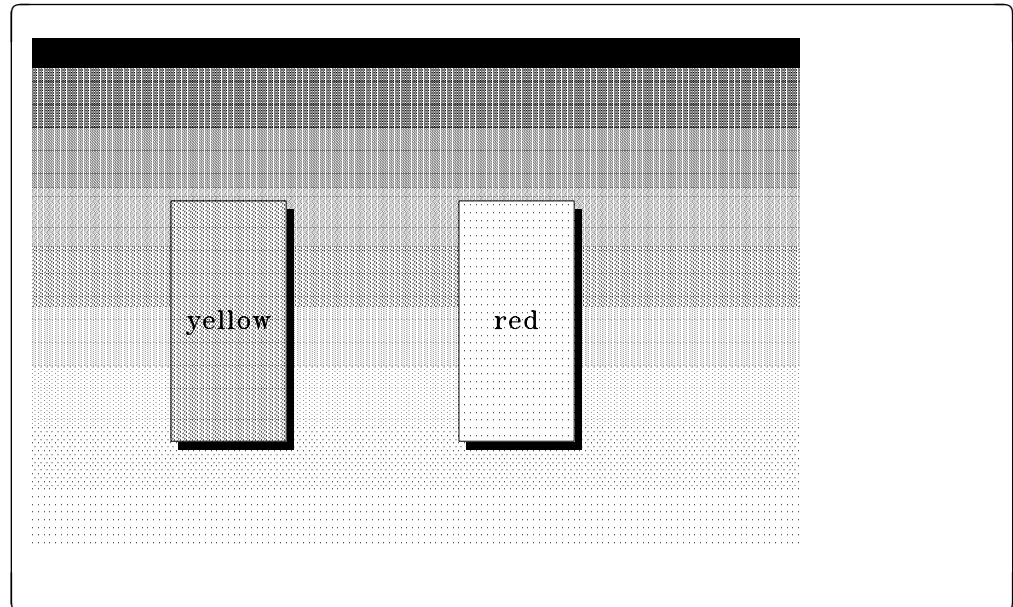


Figure 56. Fading background sample

You can get an overview of the basic GDDM colors and patterns by typing:

```
backgr ?
```

To remove a background, use:

```
backgr off
```

Note: Use the BACKGR command carefully on PS type terminals (such as the 3279 and 3270 PC), where you can easily run into a PS overflow (fields in the wrong color or asterisks on the screen) when the background is overwritten with text. In all cases, the data stream to the PS screen is much longer with background, so you will get a longer period of flicker.

You can get round this by using a FORCE command (see Chapter 10, “Creating dynamic sequences” on page 151) to force the background on the screen, and then place the text, possibly using another FORCE command:

```
1 )ops reset; backgr 3031
  )ops force
  )ops ss admuksf; size 25mm; start 30 40%
  Here is the text
  ...
```

drawing frames

Background set by an infile command (as in the example) applies only to the page where it is put. Scrolling up and down are not disturbed by a BACKGR set in an infile page. Thus, in the following example:

```
1 )ops reset; backgr blu

      WELCOME
1 )ops gdf mygdf23
1 )ops chart mychart2
1 )ops backgr red
      AND HELLO
1 ...
```

both the blue and the red backgrounds are seen only on pages that include the BACKGR commands.

Drawing frames

Use the FRAME command to draw a single rule between the command field and the graphics field (RULE), or a full frame around the graphics field (FULL).

RULE is the default with hardware or PS loaded symbol sets (SS, HW, or VEF), avoiding a confusing mixture of text and frame lines along the edges (the hardware symbol set deletes any underlying graphics on PS-type terminals).

FRAME has the same operands as DRAW: line color, line width, and line style, in that order. This example:

```
frame red 2
```

draws a red, thick (width 2) frame of the kind determined by the current symbol set.

You can get a list of possible colors with:

```
frame ?
```

Use the keywords RULE and FULL to specify the type of frame, as in:

```
frame red 2 full
frame pink rule
```

Remove a frame with:

```
frame off
```

This also removes any “paper limits” or “page boundaries” (the dotted yellow lines used in NORMAL mode to signify paper and page limits).

Like the BACKGR command, FRAME as an infile command is valid only in the page; FRAME as a direct command makes an established frame.

Chapter 9. Displaying pictures

As well as text and line art, OPS can display pictures from different sources, either alone or with something else on the screen, including:

- Charts created with the GDDM Interactive Chart Utility (ADMCDATA and ADMCFORM files)
- Pictures stored in ADMGDF format
- Images stored in ADMIMG or PSEG format

You can save an OPS screen in ADMGDF format, and display it without accessing the source, the ADMOPS-type file. You can also make it part of another picture.

This chapter tells you how to save and display pictures.

Note: Under CMS, you can display ADMCDATA and ADMGDF pictures on secondary dialed screens, as described in Chapter 16, “Using dialed terminals under CMS” on page 191.

Displaying charts created by the Interactive Chart Utility

OPS displays charts made by the GDDM Interactive Chart Utility (ICU), which lets you create all important chart types: bar, curve, surface, pie charts, and so on. OPS gives you direct access to ICU with the ICU command, described in Chapter 13, “Executing functions outside OPS” on page 171.

You can store an ICU chart in ADMCDATA files (containing the data to build the chart), and ADMCFORM files (containing the format—information about chart type, color of axes, and so on). The ADMCDATA and ADMCFORM files usually have the same filename.

Adding an ICU chart to the current picture

The CHART command adds an ICU chart to the current picture. The chart combines with any other graphics in the same position. You must specify the name of the chart: the name or names of the ADMCDATA and ADMCFORM files that together describe the chart. You can omit the name of the ADMCFORM part if it is the same as the ADMCDATA part. For example, the command:

```
chart mypie
```

loads the files MYPIE ADMCDATA and MYPIE ADMCFORM and displays the chart in its maximum size within the graphics field (two lines smaller than the full screen).

You can specify a separate name of the ADMCFORM part after the first name *if the ADMCFORM name begins with a letter*, which is always the case in TSO. For example:

```
chart mypie myform
```

displays the chart from MYPIE ADMCDATA and MYFORM ADMCFORM. OPS does not support a name beginning with a number.

adding a chart

You can get an overview of all available ADMCDATA files by specifying ? or * as the filename. You can get a limited selection by making * a part of the name, as described in “Displaying file lists” on page 197. Here are some examples:

```
chart ?
chart ch87*
```

You can also see the whole list by pressing PF6 from the OPS help menu.

If you want to display a chart in a reduced size, specify a box into which to put the chart. Use the same syntax as for a simple BOX (see “Drawing boxes” on page 119), as in:

```
chart mypie 30 30 80mm 56mm      ;* center alignment
```

This displays the same chart as above, but reduced to 80mm by 56mm, and with the center at (30,30).

Note: Ensure that any chart that you want to reduce uses only vector symbol sets, or the text will be distorted.

Here’s an example with corner-alignment:

```
chart mychrt1 co 10 10 40
```

If you don’t specify the height of the chart, OPS sets it to .71 times the width, corresponding to the aspect ratio of most screens.

If you use CHART as a direct command, OPS prompts you for the position (the “box”), unless you add the keyword FULL immediately after the chart name:

```
chart mychrt          ;* if direct, OPS will ask for position
chart mychrt full     ;* fill the screen without prompting
```

You can easily frame or add background color to an ICU chart; for example:

```
1 ...
  )ops backgr from darkblue to blue
  )ops box      40 30 80mm 55mm, draw whi 2, fi n
  )ops chart myicu 40 30 80mm 55mm
1 ...
```

The BOX command draws a white frame around the chart, and erases the background (FI N). You can also erase the area behind the chart using ERASE:

```
1 ...
  )ops backgr purple 5
  )ops chart myicu 40 30 80mm erase
1 ...
```

However, no frame is drawn this way.

The following example shows you how to make an online management information system without any commands:

```
1 )ops dark; reset; prefix %
  %pfk clear, 3 'end' 4 'page 2' 5 'p 3' 6 'p 4' ...
  %* Pressing Pf keys 4, 5, etc, now locates pages 2, 3, etc.
  %pfk 12 'top' color tur
  %pfk text Press Enter to move forward. PF3=End. PF12=Menu.
  %enter down
  %ss hw; col t
```

Key indicators for Company X.
Select chart using a PF key:

- | | |
|----------------|----------------------|
| PF 3 - Exit | PF 8 - Finances |
| PF 4 - Sales | PF 9 - Market-shares |
| PF 5 - Debtors | PF10 - Forecasts |

```
...
1 %chart sales87 ; * page 2 - sales
1 %chart debtor2 ; * page 3 - debtors
...
```

Figure 57. Sample management information system

Displaying a chart on a clear screen

The CHARTX command clears the current screen, and displays the ICU chart you name. The clear screen has no command field; no matter which key you press, you will return to the OPS main panel. Figure 58 illustrates the difference between CHART and CHARTX.

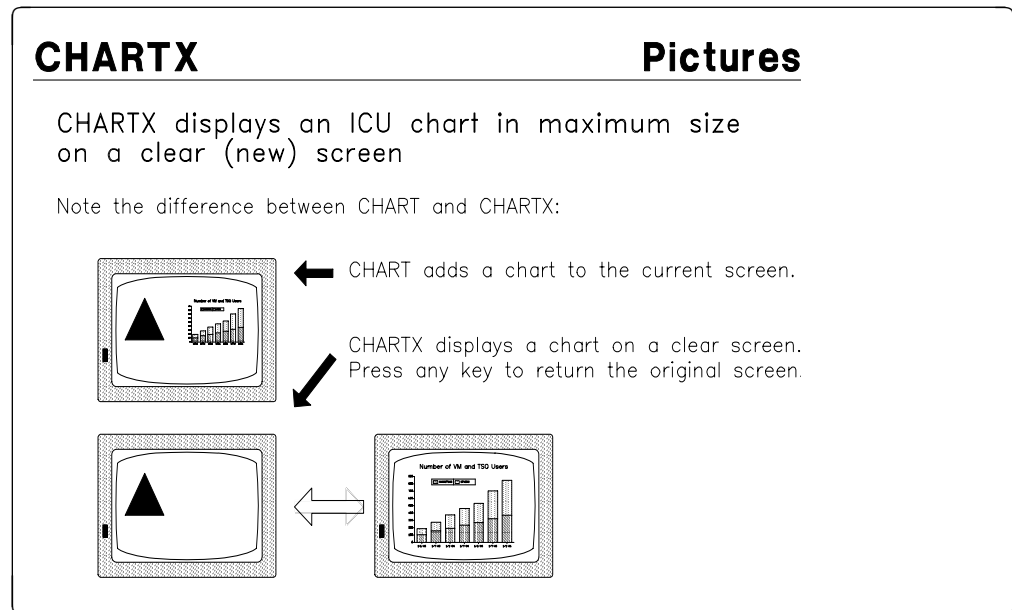


Figure 58. CHART versus CHARTX

using GDFs with OPS

Use CHARTX to insert unplanned charts during a presentation; for example, to answer a question from the audience. The display affects the presentation in no other way.

You specify the chart name as for CHART:

```
chartx mychart
```

or, if the ADMCFORM part has a separate name:

```
chartx mychart myform
```

You can get a list of the available charts using ? as the name or * as part of the name.

Under CMS, CHARTX can also display the picture on a secondary dialed terminal. In this case, the syntax is:

```
chartx chart-name device-id
```

See Chapter 16, “Using dialed terminals under CMS” on page 191 for more detail.

Using GDFs with OPS

Graphics Data Format (GDF) is the recommended format in GDDM for storing pictures. In GDF, a picture is stored as a detailed description of how it is structured, in a way that allows GDDM to reproduce the picture on all devices (screens, printers, and plotters) supported by GDDM.

Under CMS, the filetype of a GDF file must be ADMGDF. Under TSO, user GDFs are usually stored and retrieved using the *ddname* ADMGDF. However, OPS uses the *ddname* ADM7GDF to store GDFs.

An ADMGDF file (referred to simply as a GDF) can depict anything from the simplest line art to complete screen images. Very often, installations develop quite large collections of GDFs depicting various objects useful in presentations (people, symbols of all kinds, company products, and so on). Putting a GDF on a shared disk or in a shared dataset makes it available to all users of OPS.

GDFs can be generated by OPS, ICU, ADMUSP4 (a programming sample included with GDDM), conversion of PIF files, DrawMaster, and other programs.

OPS includes a number of GDFs and the OPS files that produced them. Figure 59 on page 139 shows some of these GDFs.

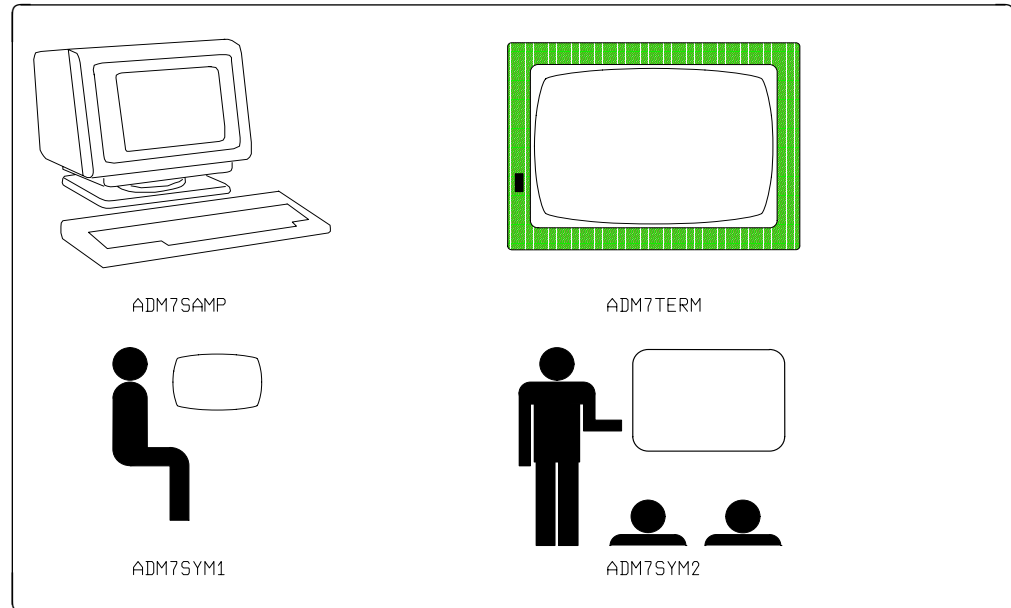


Figure 59. Some GDFs included in OPS

Transforming GDFs

A GDF can be **transformable**, non-transformable, or both.

If a GDF is transformable, you can scale it up or down, turn, shear, and move it. If a GDF is non-transformable, you can display it only in full format or in reduced form within the graphics field. If a GDF contains transformable and non-transformable parts, OPS transforms what it can and displays the rest in full format.

In order to be transformable, a GDF cannot contain **unnumbered graphics segments** and it must have been created explicitly to be transformable. An ICU chart is **never** transformable. A stored OPS page **is** transformable, as long as it does not contain non-transformable parts from a GDF or an ICU chart.

Handling symbol sets with GDFs

If a GDF contains text written with a named symbol set, this symbol set must be available when the picture is displayed. The definition of the symbol set is not included in the GDF. When a GDF is to be displayed, GDDM will load **all** symbol sets loaded at the time when the GDF was made, even if the picture uses no symbol sets at all.

This has two important consequences:

- When you send a GDF to other users, they can see it correctly only if they have access to the symbol sets used. Missing symbol sets are replaced by the default symbol sets, VSS or HW, and GDDM displays an error message. So remember to release all superfluous symbol sets before you store an OPS page as a GDF, by:

```
q ss all
release ss-number ...
```

displaying GDFs

- When you display a GDF together with other text or another GDF, GDDM does not consider already loaded symbol sets. GDDM assigns a number between 65 and 250 to loaded symbol sets. These numbers are stored with the symbol set names in the GDF. When displaying a GDF, GDDM loads the original symbol sets into the original numbers, so that already loaded symbol sets of the same number are overwritten.

OPS monitors this and, when a symbol set being used on the page is overwritten, gives you the message:

"Symbol Set conflict. Refresh may help"

If GDDM itself reproduces the page (for example if a segment is deleted using UNDO), the text on the screen is written with the last-loaded symbol set. Use REFRESH to make OPS find an available number for the overwritten symbol set and then reload it.

Displaying GDFs

The GDF command displays files of filetype ADMGDF under CMS, or files in the data sets allocated for input GDFs under TSO.

The GDF command has the same syntax as the CHART command:

```
gdf gdf-name
```

Here's an example:

```
gdf adm7samp
```

Figure 60 shows the result.

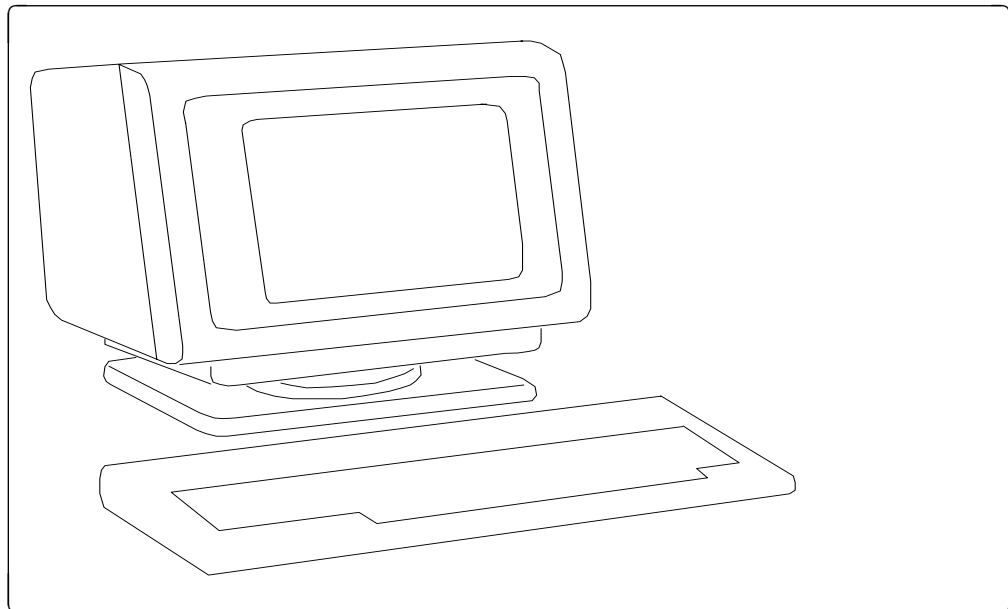


Figure 60. Display from ADM7SAMP ADMGDF

If you use the command directly, add FULL to avoid prompting for size and position, as in:

```
gdf mygdf full ;* display full size without prompting
```

You can get a list of files by typing GDF ? or GDF IBM*.

Just as with CHART, you can get a reduced display by specifying a box into which to put the GDF. For example:

```
gdf adm7samp 40 30 40mm      ;* default alignment is the center
gdf adm7samp co 40 30 40mm
```

both display the GDF file ADM7SAMP reduced to a width of 40mm. The first puts the center at (40,30). The second puts the lower left-hand corner at (40,30). The GDF appears in the original aspect ratio, with circles remaining circles.

If you omit the height (as in the examples), OPS chooses a default height of .71 times the width (the same aspect ratio as with charts).

You can also specify the height of the box following the width, as in:

```
gdf mygdf 40 30 40mm 40mm
```

The GDF still appears in the original aspect ratio. If you create it in a square graphics field (ASPECT 1 1), it becomes larger than the height of .71*40mm that it would be if you omitted the fourth operand (see Figure 61).

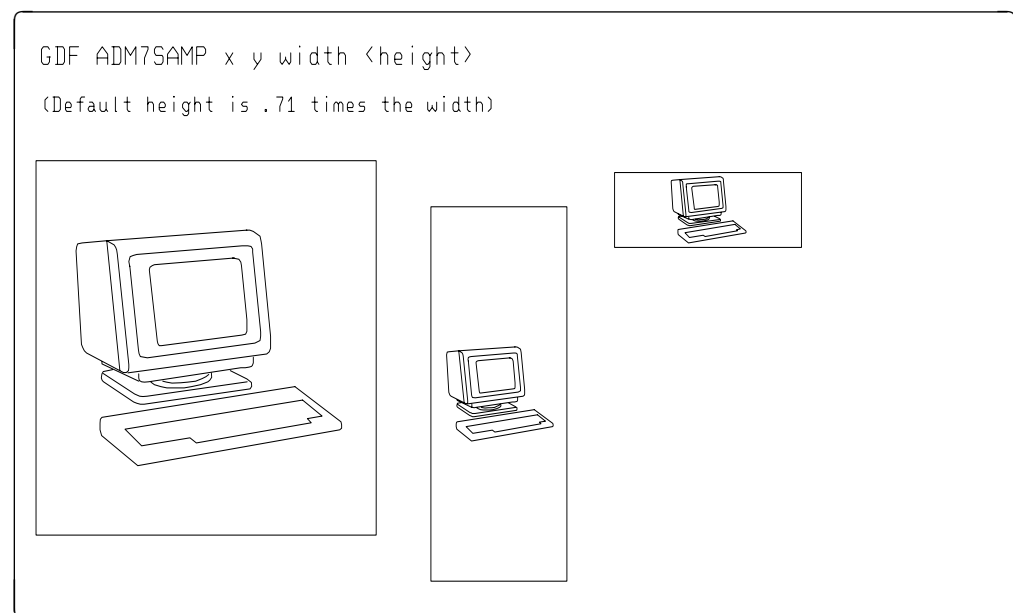


Figure 61. Displaying a GDF at reduced size

displaying GDFs

If the GDF is transformable, you can manipulate it, as shown in the examples below:

Command	Effect
GDF ADM7SAMP 50 50% 60mm SCALE 1 .5	Squeeze the picture of the GDF ADM7SAMP.
GDF ADM7SAMP 50 50% 60mm TURN 45	Turn with the original aspect ratio 45 degrees counterclockwise.
GDF ADM7SAMP 50 50% 60mm SHEAR -30	Shear the GDF 30 degrees to the left.
GDF ADM7SAMP FULL SCALE 1.5	Magnify the GDF 1.5 times.

See Figure 62 for an illustration.

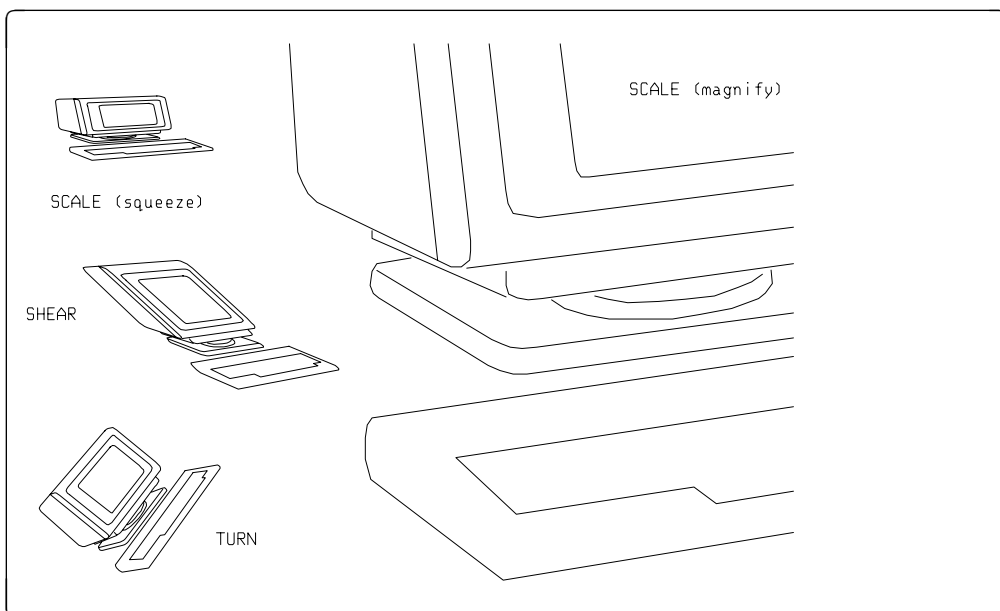


Figure 62. Displaying GDF files with transformation

If the GDF is non-transformable, it gets smaller as you move it closer toward the edge of the graphics field. See Figure 61 on page 141.

You can make sure that OPS won't try to transform a GDF by adding the keyword NOTRANS (NO), as in:

```
gdf adm7samp 80 50 70mm notrans
```

If necessary, OPS reduces the window in size to make it fit within the graphics field.

Note: You cannot scale or turn image symbol sets. If you use such symbol sets in a GDF, a transformation may give you unwanted results.

Recoloring primitives while loading GDFs

You can change **unspecific** (also called **default**) values of graphics attributes, such as unspecific color or line type, in a GDF to be loaded. An unspecific attribute has a value of 0, set using COLOR, DRAW, FILL, or WFILL in any object saved as a GDF, letting you recolor selected parts of an object loaded as a GDF.

You change attributes using DRAW and FILL, either as primary commands or as subcommands on the GDF command. You can apply only one color for both DRAW and FILL.

Whether you specify FILL or not, the object is filled if the original object was filled. To control a fill pattern, however, you must use FILL.

For example, assume that you have created a box using:

```
box 50 50% 30 fi 0 0 draw blu width 3
```

and saved it in a GDF called MYBOX. You can then vary its appearance when you load the GDF, as shown in the following examples:

GDF MYBOX	The center is white, the border blue.
GDF MYBOX FILL YEL	The center is solid yellow, the border blue.
GDF MYBOX FILL GREEN 11	The center is shaded green, the border blue.

Merging graphics segments while loading GDFs

An OPS picture consists of many graphics segments, typically one for each line art or text object. When a picture is saved as a GDF, the internal structure is kept.

OPS keeps track of which segments belong to which command, so that, when you load a GDF with several segments into the picture, OPS treats the segments as a unbreakable group.

Letting a GDF consist of many segments has the advantage of making it editable. You can use a segment editor such as GDFEDIT (described in “Editing GDFs without the OPS source” on page 145), to move segments around or remove unwanted parts.

However, when you work with GDFEDIT, you may want to treat a GDF as an unbreakable unit. To achieve this, load the GDF with the MERGE keyword, merging all its segments into one.

For example:

```
gdf adm7sym2 20 50 100mm merge
```

Displaying GDFs on a clear screen

The GDFX command displays the specified GDF on a clear screen, independent of the current panel. There is no command field on the clear screen. Pressing Enter or a PF key will bring you back to the primary page.

Use GDFX to display GDFs in the middle of a presentation when you want the primary panel to remain unchanged after the display. OPS reloads any overwritten symbol sets before showing the primary page again.

You specify the name of the GDF as for the GDF command:

```
gdfx adm7samp
```

You can use * to get a list of files, as in:

```
gdfx ibm*8*
```

The picture is **always** displayed at maximum size. You cannot make any transformations.

In normal mode, GDFX displays any description stored in the GDF in the primary message field of OPS.

Under CMS, GDFX can also display the picture on a secondary dialed terminal. In this case, the syntax is:

```
gdfx gdf-name device-id
```

See Chapter 16, “Using dialed terminals under CMS” on page 191 for more detail.

Saving the graphics field as a GDF

GDFSAVE saves the current graphics field in GDF format. The picture is saved as transformable if it does not contain an ICU chart or external, non-transformable GDFs. The picture name must be a maximum of eight characters.

For example, to generate the file MYGDF ADMGDF on your A-disk, use:

```
gdfsave mygdf
```

If the file already exists, specify REPLace:

```
gdfsave mygdf repl
```

You can use GDFSAVE as an infile command as well as a direct command. If you use it directly, OPS prompts you for a description, which is saved in the GDF itself. You can display the description with GDFX in normal mode.

You can also specify the description in the command using the keyword DESCR:

```
gdfsave mygdf repl descr 'This is my first GDF'
```

Finally, you can suppress prompts in direct mode by typing:

```
gdfsave mygdf repl nodescription
```

You can also use the GDFSAVE command to save selected graphics segments (see “Editing GDFs without the OPS source” on page 145).

Use the SEGMENT keyword to request a limited save:


```
gdfsavemy sketch segment
gdfsavemy segm seg 34
```

The first example saves the last drawn segment (which might be a segment added by the SKETCH command). The second example saves segment 34.

You can query the segment number of detectable objects by the QUERY command:

```
q segment
```

This prompts you to pick an object. The segment number is displayed in the message field.

Note that segments may be saved even when they are not detectable.

The AUTO command also supports GDFSAVE during automatic browsing:

```
auto gdfsavemy base-name
```

where *base-name* is replaced by a base name having a maximum of five characters, as in:

```
auto gdf mygdf
```

The AUTO command saves all pages, starting with the current page, and assigns them the names MYGDF001, MYGDF002, and so on. Any existing files are replaced.

Editing GDFs without the OPS source

GDFEDIT, which you can execute only directly, sets a new environment from which you can:

- Save the current picture as a GDF
- Copy or move any detectable segment
- Delete any detectable segment
- Issue any OPS commands (except scroll commands)

You can do the first three completely by mouse, cursor, or PF key.

Starting GDFEDIT

Use GDFEDIT on its own to start editing the current picture. You can also load a GDF before editing:

```
GDFEDIT MYGDF
```

The screen is not cleared before loading the GDF.

When you are editing a named GDF, the SAVE function of GDFEDIT automatically saves and replaces the original GDF, with the description (if any) found within it.

No prompt is issued.

When you edit the current picture, the SAVE function prompts for a name, a description, and for permission to overwrite an existing GDF of the same name. Subsequent saves under the same name do not prompt for the description or permission to replace.

Working with GDFEDIT

When you invoke GDFEDIT, the first line of your screen displays messages telling you what to do.

The second line shows the PF key assignments, as follows:

1	Help
2	Save
3	End
4	Command
5	Copy
6	Move
9	Delete
11	Draft on

You select the action you want using the PF key, then continue as instructed by the message field.

If you select COPY, MOVE, or DELETE, the cursor jumps to center screen, and you are prompted to pick an object.

If you select COPY or MOVE, a green cross appears at the point you picked. Pick a new position and the original object is copied or moved to that point.

If you select DELETE, the picked object disappears immediately.

You can then continue with the same function by picking a new object, or you can select a new function using the PF keys.

If you want to add new objects to the picture, select the CMD function (PF4). This provides you with a normal command field, from which you can issue any OPS command, except a scroll command. After issuing one or more commands, return to the GDFEDIT menu by pressing Enter.

Displaying images

Images are pictures defined in picture elements (pels). OPS lets you display images defined in one of the following formats:

ADMIMG	GDDM's internal standard for images
PSEGxxxx	Page segments (PSEGs) for IBM 38xx or 4250 page printers. PSEGs are most commonly used with DCF (Document Composition Facility) documents and AFPDS printers. PSEGs can be created by the OPS PSEG command.

Note: OPS displays only PSEGs containing IM type images. The PSEG command creates these. You cannot use OPS to display images containing IOCA or GOCA images.

Basic image display

Under TSO, images are kept in partitioned data sets allocated to certain *ddnames*; under CMS, images are kept in files with defined filetypes and filemodes.

To display an image, use the IMAGE command to tell OPS which one you want. The simplest use requires just an image name, as in:

```
image houses
```

OPS assumes ADMIMG as the TSO data set *ddname* or CMS filetype.

If your image has a filetype or *ddname* other than ADMIMG, or you want to input page segments (PSEGxxxx or ADMIMAGE), either change the default image data set using the IMGLIB command, or put the data set name you want on the IMAGE command.

Under CMS, the syntax of IMGLIB is:

```
imglib filetype
```

which sets the default image library to *filetype* (ADMIMG or PSEGxxxx).

Under TSO, the syntax of IMGLIB is:

```
imglib ddname 'data set name'
```

which sets the default image data set to *ddname* (ADMIMG or PSEGxxxx). If you have already allocated a data set to *ddname*, you can omit 'data set name'.

Similarly, under CMS, the syntax of IMAGE is:

```
image filename filetype filemode
```

which searches for the file identified, with *filetype* and *filemode* defaulting to ADMIMG and A.

Under TSO, the syntax of IMAGE is:

```
image member ddname
```

which searches for *member* in the data set allocated to *ddname*, which defaults to ADMIMG, or whatever you have set in IMGLIB.

For page segments, the *filetype* (CMS) or *ddname* (TSO) must be PSEGxxxx (where xxxx is any character or blank).

Scaling images

An image is a collection of display points or **picture elements (pels)** with:

1. A built-in size, which differs from device to device
2. A **resolution ratio**: the vertical resolution divided by the horizontal resolution, which may differ from that of the device

To help you deal with the size differences, OPS provides scaling.

Defining the position and size of images

In general, images are positioned like GDFs: by specifying a box into which to put the image.

Because many images look best at their built-in size, OPS gives you that as the default, as in:

```
image admu5img          ;* display at built-in size
```

If you try the above command from the command field, you will be prompted for center and size. You can avoid the prompting by telling OPS what you want:

```
image admu5img asis
image admu5img 30 20 asis
```

In the first example, OPS will give you the built-in size, but prompt for centering. In the second example, the image is centered around (30,20) and shown without scaling.

As with GDFs, you can also request that the picture fills the screen:

```
image admu5img full
```

If you want the image to fit within a box, use the same syntax as for BOX, GDF, or CHART; for example:

```
image myimg 40 30 30mm      ;* center at (40,30)
image myimg cc 40 30 30mm   ;* the same
image abc co 20 20 40 30    ;* lower-left at (20,20)
image abc co 20 20 -10 10   ;* lower-right at (20,20)
```

Refer to “Drawing boxes” on page 119 for more information.

Notes:

1. By default, underlying objects show through the image. To avoid this, use ERASE as a subcommand on the IMAGE command.
2. You cannot use the GDDM user control “zoom in” function on an image displayed by OPS. OPS displays images as **graphics** images, and you cannot zoom in on such images.
3. When an image is scaled to fit within a box, it is scaled evenly in all directions so that no distortion takes place. If you **want** to distort it, use SCALE, as described below.

If you add ASIS, the image does not fit the box, but is displayed at its built-in size. It is centered in the box only if you use center-syntax; otherwise the corner indicated by your syntax is fixed at the fix point. This gives you full control of the image position without knowing the built-in size. In the case of ADMU5IMG, the built-in size fills most of the screen, as shown in Figure 63 on page 149.

IMAGE ADMU5IMG 20 50 ASIS

JOB APPLICATION FORM	
1. Position applied for	Computer Programmer
2. Name	Geraldine Moore
3. Date of Birth	28 August 1966
4. Place of Birth	Hallfield, England
5. Home Address	"Chilcote" Ringside Way Winchester
6. Telephone Number	01963 4433
7. <input type="checkbox"/> Male <input checked="" type="checkbox"/> Female	
8. <input type="checkbox"/> Married <input checked="" type="checkbox"/> Single	
9. Do you possess a valid driving licence?	<input checked="" type="checkbox"/> Yes <input type="checkbox"/> No
10. Educational Qualifications	First Class degree in Image Manipulation from Oxbridge University, 1986.
11. Present Employer	Winchester Software Consultants
12. Recent Job Experience	Working as an applications programmer in the computer department.
13. Do you need relocation assistance?	<input type="checkbox"/> Yes <input checked="" type="checkbox"/> No
14. Can you work overtime and/or shift?	<input checked="" type="checkbox"/> Yes <input type="checkbox"/> No
15. Signature	G. Moore Date 12 January 1987




Figure 63. Image using ASIS

Using the SCAle keyword you can scale the image unevenly, and mirror it. Scaling is done on top of your initial position and size request. Here are some examples:

```
image admu5img scale 2
image admu5img co 10 10 scale 2.1 asis
image admu5img 30 20 30mm 30mm sca 0.8 0.6
```

The first two load the image ASIS, and then scale it. In the third example the image is first fitted into the box and then reduced unevenly.

To mirror an image, use a negative scale value. A negative x-value mirrors the image about a vertical line; a negative y-value mirrors about a horizontal line:

```
image admu5img asis sca -1 1 ;* mirror in x
image admu5img sca 2 -.5 ;* stand on your head
```

Controlling the color of images

Use COLOR to set the display color of either a monochrome image or a layer of a multilayered image. Color value may be in the full range of basic GDDM colors: -2 to 16, or abbreviations of the English color names (see "Coloring text" on page 69).

Use SELECT to select a specific layer of a multi-layered image.

Here are some examples:

```
image admu5img id* select 4
image admu5img id3279 sel green col whi
```

Use REVERSE to reverse the on/off status of the pels in an image; that is, to display it positive or negative. Here's an example:

```
image admu5img 23 45 reverse, color 13
```

Chapter 10. Creating dynamic sequences

OPS contains a number of commands to produce dynamic, or even animated, sequences.

Dynamic means that the picture changes under control of the user or presenter, typically by making new objects (text or drawing) appear when the user scrolls forward in the presentation.

In an **animated** sequence, the picture changes without any interaction from the user or presenter. Typically, objects change their shape (as they do in cartoons).

Your ability to provide animated sequences depends on the type of terminal in use. On a PS type (such as a 3279), the screen is presented in a single step (or a few steps, depending on the buffer size), so the user does not see the order in which the objects are placed, and animation is quite effective. On a vector-type screen (such as a 3192 G), the picture is drawn line by line, so the user can clearly see how the picture is created, reducing its value as an animation.

Inserting pauses in presentations

Use the WAIT command (infile only) to insert breaks in a presentation. For example:

```
wait
```

displays the picture and then waits for input. The kind of input it expects is:

- You press Enter or issue the command DOWN (usually by pressing PF8). OPS continues reading the infile from the point where it stopped.
- You enter a command; OPS executes it. You can issue any command including scroll commands; for example DOWN PAGE (normally P + PF8), which makes OPS go to the top of the next page.

You can use the AUTO command (see Chapter 14, "Scrolling" on page 177) with the keyword PAGE (OPS presses Enter for you), in association with WAIT:

```
auto p
```

This scrolls each page without your having to do anything. When OPS executes the WAIT command, it shows the picture as far as it has been completed, breaks for one second, then continues the reading. When it reaches a page break, OPS stops and waits for the next command.

AUTO P stays active on all pages until you enter:

```
auto off
```

You can also use the WAIT command to force a break of a fixed period, by adding the time, in seconds:

```
wait 5
```

OPS presents the picture, breaks for five seconds, and then continues the reading. During the breaks, the user cannot issue commands, or cancel OPS.

forcing the graphics field

Use breaks of a fixed period to add effect to a picture. You get more impact from a slogan or header if you present it word by word with an interval of one second between each word:

```
1 ...
  )ops ss admuwksf; h 25mm; col red
  )ops ce on
  USE
  )ops wait 1
  MORE
  )ops wait 1
  GRAPHICS
  )ops ce off
  ...
```

Forcing the graphics field to the screen

Use the FORCE command to force the current graphics field to the screen and then return control to OPS without waiting for new input.

On a PS-type terminal, FORCE lets you display intermediate levels of the completed picture on the screen. On a vector-type terminal, you always see the intermediate levels, but FORCE makes a clearer division of the sequence.

A typical use of the FORCE command is to delete the content of a window before replacing the rest of the content. Here's an example:

```
1 ...
  )ops fill neutral
  )ops box 50 30 50 36; * a black window
  )ops force;          * make it appear on the screen
  )ops gdf slambam 50 30 50
  )ops wait;          * wait for Enter or DOWN
  )ops box 50 30 50 36; * delete the content of the window
  )ops force;          * make it appear on the screen
  )ops gdf nextone 50 30 50
  )ops wait;          * wait for Enter or DOWN
```

Without the FORCE command in the above example, you would see the old picture "crackle" while the new picture is created. Using FORCE, you ensure a floating change between the pictures with as little flicker as possible (PS terminals only).

You can also use FORCE to make animations on a PS type terminal. The next example shows you a box moving across the screen:

```
1 ...
  )ops draw off
  )ops box 30 30 20 20 fill r
  )ops force; * make the box appear and continue immediately
  )ops box 30 30 20 20 fill n
  )ops box 35 30 20 20 fill r
  )ops force
  )ops box 35 30 20 20 fill n
  )ops box 40 30 20 20 fill r
  )ops force
  )ops box 40 30 20 20 fill n
  )ops box 45 30 20 20 fill r
  ...
```

Creating animated sequences

The MOVIE command is equivalent to a FORCE command inserted after each input text line in the infile, and after each graphics command that creates screen output. It displays text line-by-line, and draws figures sequentially on the screen.

MOVIE has two keywords: ON or OFF. ON is the default, so the command:

```
movie
```

switches **movie mode** on. Here's an example with a word presented stepwise ("New deals:") and an arrow:

```
1 ...
)ops ss admuwtrp; h 6; col y; start 20 40
)ops movie on
)ops arrow 20 10 20 11 w 1 h 1
)ops text 'New '
)ops arrow 20 10 20 20 w 1 h 1
)ops text 'd'
)ops arrow 20 10 20 30 w 1 h 1
)ops text 'e'
)ops arrow 20 10 20 30 35 30 w 1 h 1
)ops text 'a'
)ops arrow 20 10 20 30 50 30 w 1 h 1
)ops text 'l'
)ops arrow 20 10 20 30 65 30 w 1 h 1
)ops text 's'
)ops arrow 20 10 20 30 65 30 75 40 w 1 h 1
)ops text ':'
)ops arrow 20 10 20 30 65 30 76 41 w 1 h 2 sc 2
)ops wait 1
)ops movie off
)ops arrow 20 10 20 30 65 30 76 41 w 1 h 2 sc 2 dr n 2
)ops arrow 20 10 20 30 65 30 76 41 w 1 h 2 sc 2 curve
```

Displaying messages in dynamic sequences

When you use OPS for online information or self-tuition classes, you may want to tell the users about the PF keys they can use, and possibly the commands they can execute. You can do this with the PFK text, set by the PFK command.

However, in dynamic connections, you'll also want to use inserted messages such as "Wait a second." For this, you use the MSG command:

```
msg 'This is an important message'
```

OPS displays the message with the current attributes in the message field, either as initiated with DARK or NORMAL, or as set by the MSGFLD command (see "Controlling alphanumeric fields" on page 55). You can set temporary attributes, valid only for this message, with the same operands as on MSGFLD:

```
msg 'Press Enter now' color red, blink
```

You can also use ALARM to sound the built-in alarm when the message is shown:

```
msg 'Do NOT press the keys' blink, alarm
```

dynamic sequence examples

If you have no operand other than the message text, you can omit the quotation marks.

The message is displayed at the next screen update (after FORCE, WAIT, or full screen), when OPS is waiting for input from the user. MSG has a higher level of priority than display of the PF keys (PF ON), because it is used for more important messages. (You can always use QUERY PF if you need some information about the PF keys.)

You remove the message with an empty MSG command or RESET:

```
msg
reset
```

While a MSG command is active, OPS displays messages with the default attributes set by DARK, NORMAL, or MSGFLD. An error message is deleted by any command or when Enter is pressed. Then the active MSG is shown once more.

You might want to have online users see messages, and a presentation audience not see them. For this, you use:

```
msg off
```

and, of course:

```
msg on
```

Sounding the alarm

You can make the terminal's alarm sound when OPS updates the screen, using:

```
alarm
```

This sounds the alarm after FORCE, WAIT, or end-of-page.

Examples of dynamic sequences

Here are a few examples to illustrate use of the different dynamic facilities:

```
1 )ops dark; reset
   )ops pfk text 'Press PF8 to move forward.'
   )ops pfk on, col t
   )ops alarm
   )ops h 5
```

```
      This online information is
      about the weather
```

```
1 )ops msg "Don't press any key." col red
   )ops gdf clouds 30 50% 45
   )ops wait 1
   )ops gdf sun    70 50% 45
   )ops wait 1
   )ops clear; reset; * now MSG is deleted
   )ops h 5
```

And now a few facts:

...

The following example is an extended version of the example in Figure 57 on page 137.

```

1 )ops dark; prefix %
  %pfk clear, 3 'end' 4 'page 2' 5 'p 3' 6 'p 4' ...
  %pfk 12 'top' color tur
  %pfk text 'Press Enter to move forward. PF3=End. PF12=Menu.'
  %msg 'Management Information System - Company X' col red
  %**** RESET on the subsequent pages will remove this MSG,
  %**** and then the PF keys will be shown.
  %enter down
  %ss hw; col t

Key indicators of Company X.
Select chart using a PF key:

      PF 3 - Exit           PF 8 - Finances
      PF 4 - Sales         PF 9 - Market-shares
      PF 5 - Debtors       PF10 - Forecasts
      ...
1 %* page 2 - Sales
  %reset; chart sales87
1 %* page 3 - debtors
  %reset; chart debtor2
  ...

```

Figure 64. Sample Management Information System (extended)

dynamic sequence examples

Chapter 11. Controlling the sequence of execution

OPS lets you control the sequence using IF-THEN-ELSE constructs and labels to process defined parts of a presentation or picture. You'd use this, for example, to:

- Support different colors or patterns for different devices
- Exclude or include selected material depending on the audience
- Allow the users to control the flow by mouse or PF key

You'll find examples of the functions involved in the rest of this chapter, as well as in ADM7OPDA and ADM7OPNA.

Flow control commands

The primary control command is the IF command, as in:

```
if &c = 2 then color red
if 'This is fine' = &v
if &x < sqrt(&xx) then locate page 1
```

The IF command sets a TRUE-FALSE flag, of which only one exists. You test the flag, and act on the result using THEN and ELSE commands. You can have more than one THEN and ELSE associated with each IF command.

Here are some examples:

```
)ops msg 'Type 1 for screen, 2 for plotter' alarm
)ops readcmd prompt 'Enter 1 or 2:' white
)ops if &$rcmd = 1
)ops then fill 3211
)ops then draw off
)ops else fill red 2
)ops else draw white
```

Note: In the above example, and in several examples in this chapter, the character \$ is used. The actual character is the currency symbol, and will vary according to the language in which you are working.

To see what character applies to you, execute:

```
q sysvar
```

The character in front of the variables shown is the one that you must use.

If &\$rcmd equals 1, the two THEN commands are executed. Otherwise the two ELSE commands are executed.

You can achieve a more conventional select-construct using ELSE-IF cascades, as in:

```
)ops .again
)ops readtag
)ops if &$rtag = 0 then locate &$page+1
)ops else if &$rtag = 1 then exec p1
)ops else if &$rtag = 2 then exec p23
)ops else if &$rtag = 3 then end
)ops else if &$rtag = 4 then locate page 1
)ops else goto again
```

Using labels

You define labels either directly (from the command field) or infile (using a label command).

To define a direct label, enter a single alphabetic character prefixed with a period (.), for example:

```
Command ==> .a
```

A direct label is always associated with the top of the page. Transferring to a direct label redisplay the associated page.

You can redefine the setting of a direct label at any time. The new position overwrites the old.

To define an infile label, have a line begin with a word of from 2 to 10 characters, prefixed with a period (.). For example:

```
)ops .L2      ;* The way you define a label.
```

An infile label must begin with an alphabetic character, and be unique. Duplicate infile labels are considered an error.

To jump to a label, use either LOCATE or GOTO. As you'll see from the examples below, they're very similar, but with some slightly different defaults. Here are some LOCATE examples:

```
locate 2      => locate page 2 (if the file has pages)
l p 2        => locate page 2
l l 2        => locate line 2
l .abc       => locate label abc (similar to PDF Edit)
```

Here are some GOTO examples:

```
goto 2        => goto line 2 (different from Locate 2)
goto p 2     => goto page 2
goto .abc    => goto label abc
goto abc     => goto label abc (not legal with LOCATE)
```

Note: In order for OPS to locate a label, it must be already known (that is, the flow must have passed the label), **or** it must be the first command in the record. For example:

```
)ops prefix %
%.a1; box 20 30 4 3      ;* Well defined label
%box 20 30 2 4 5 ;.a2   ;* NOT well defined
```

Tagging objects

OPS lets you **tag** drawn objects, such as boxes and text. You can then retrieve the tag value of an object by pointing to it with a cursor or mouse. This lets you create mouse-controlled presentations or applications.

Here's the TAG command in action:

```
%box 40 30 20 tag 2 fill red
%text 'Pick me' &x+10 tag 879
%tag 34
%image adm7samp 70 20 30
%box 70 20 30
%tag 0 ;* no more tagging
```

Valid tag values are -1, 0, and any number between 1 and 1000, interpreted as follows:

- 1 The object is explicitly untagged.

This is a parallel to the unspecific graphics attributes: if you save the object with GDFSAVE, you can change the tag to a specific value when you later retrieve it with GDF. Having done so, you can read the tag using READTAG, enabling you to create taggable icons and similar objects.

Note: TAG -1 creates an undetectable object: you cannot COPY, MOVE, TEST, or UNDO it until you reload it with GDF and tag it with a positive value.

- 0 You cannot pick the tag using READTAG (see "Getting tag values" on page 160). This is the default, and the value set by RESET.

You can use COPY, MOVE, and so on with objects drawn with TAG 0, but you can't add a tag value during a GDF load.

1 to 1000

You can pick the tag with READTAG, using this specific tag value.

Any number of objects can have the same tag. Use the TAG main command to assign the same tag to several objects, or to tag infile text, formatted or unformatted. Here's an example:

```
)ops tful
)ops tfli More information about
)ops tag 2
subject 1 and subject 2
)ops tag 0
may be found by clicking on any of the words.
)ops tfli ...
...
```

Any ADMGDF you save from OPS will contain the tag values set in the picture; if they're positive, READTAG can read them. Here's an example of using TAG -1 and changing the tag in the GDF load command:

getting tag values

```
% * create an object to be tagged when loaded

% tag -1                ;* unspecific tag
% box ....
% poly ...

% box ... ; line ...
% gdfsavemytag repl descr 'This ADMGDF has no specific tag'
% clear                ;* remove all objects from the screen
% tag 0                ;* reset tag to the default
% box .... tag 1       ;* box tagged 1
% gdf mytag ... tag 2 ;* will now be tagged "2"
% readtag nopf
% if &$rtag=0 then end
% * tag value will be 1 or 2 at this point.
```

Note: The CHART command does not support TAG.

Getting tag values

Use the READTAG command to retrieve the tag value of an object by pointing with a mouse or cursor. The following example illustrates the principle:

```
)ops set &button = str(box 1.3* fill turq col w)
)ops text 'These are your options:' 10 20 ss admuutrp h 5
)ops text 'ICU'      10 10 &button tag 9
)ops text 'End'     30 10 &button tag 3
)ops text 'Backward' 50 10 &button tag 7
)ops text 'Forward' 70 10 &button tag 8
)ops text 'Top'     90 10 &button tag 1
)ops .again
)ops msg 'Make your choice...'
)ops readtag
)ops if &$rtag = 0 then locate &$page+1
)ops if &$rtag = 1 then locate p 1
)ops if &$rtag = 3 then end
)ops if &$rtag = 7 then locate &$page-1
)ops if &$rtag = 8 then locate &$page+1
)ops if &$rtag = 9 then icu
)ops ..
)ops goto again
```

The first part of the example draws five buttons with various contents. Each box is tagged with a unique number using TAG.

The label .again marks the beginning of a processing loop. When the READTAG command executes, OPS displays the message Make your choice..., and places the cursor at the center of the screen.

Control returns to the presentation when the user presses any PF key or picks a tagged item.

In the first case, the system variable &\$RTAG is set to 0; in the second case, &\$RTAG holds the tag value of the picked object.

If the user presses a PF key, READTAG verifies that the PF key is defined, and schedules the associated command. For example, if PF8 is defined as DOWN,

READTAG executes DOWN, and the command following READTAG is executed as a result.

Likewise, if a presentation is run in AUTO mode, READTAG automatically issues a DOWN command with &\$RTAG=0 set.

To handle these situations, **always** follow a READTAG command with a processing option for the case &\$RTAG=0.

You can also disable the PF keys:

```
readtag nopf
```

But be careful: if the picture contains no tagged object, you can't escape.

By default, READTAG positions the cursor at the last pick position, or at center screen the first time. You can select another cursor position using one or two coordinates immediately after the READTAG keyword:

```
readtag 10 5 nopf
```

Note: PS/2 with Graphics Workstation Program may require you either to move the mouse or to press the mouse button twice before it acknowledges the click.

Reading the command field

Use READCMD to retrieve the contents of the command field into two variables: &\$RCMD and &\$RCMDU. The first contains the string without translation, the second the string in uppercase. Using IF-THEN-ELSE, you can alter the flow of your presentation based on the contents of the command field.

The parameters on READCMD are the same as on CMDFIELD; for example:

```
)ops readcmd prompt 'Enter Y or N:' pink, reverse
)ops if &$rcmdu = Y then goto page 17
```

READCMD restores the command field to its previous status after the read process.

reading command field

Chapter 12. Using symbolic variables

Use **symbolic variables** to store and retrieve sequences that you use often, or sequences that you want to vary. This chapter tells you how to set up and use symbolic variables.

Defining and retrieving variables

Use the SET command to assign values or expressions to symbolic variables, as in:

```
set i = 0
set c = red 2
set descr = 'This is the description'
```

You can then use the symbolic variables, identified by the ampersand (&) prefix:

```
set i = &i+1
draw &c
gdfsav pict01 desc &descr
```

Variable names must be no longer than 10 characters, begin with an alphabetic character in the range A to Z, and contain only alphabetic characters, numbers, and underscore. OPS translates all names to uppercase internally. OPS also reserves a few names for its own use; see "Using system variables" on page 167.

OPS evaluates variables whenever it finds a word preceded by an ampersand, even when the word is within a quoted string. The only exception is the single & (**current point**), which, for convenience, is not evaluated in quoted strings or in infile text. Thus:

```
text "Donald Duck & Company"
```

displays what you expect.

If you want OPS to interpret an ampersand as an ampersand, put two:

```
text "The year is available in the variable &&$year"
```

This displays &&\$year without resolving.

Note: The above example includes the system variable \$year. The \$ character is the currency symbol, and will vary according to the language in which you are working. To see what character applies to you, execute:

```
q sysvar
```

The character in front of the variables shown is the one that you must use.

You can also disable variable substitution with the VARSUBST command:

```
varsubst off
text "The year is available in the variable &&$year"
varsubst on
```

In the example, the variable \$year will not be resolved.

interpreting numeric values

You can use variables anywhere in a command, even to symbolize a complete command, as in:

```
set f1 = ss admuusrp
&f1
```

However, a variable cannot contain a semicolon (;), and cannot span more than a single command. The following example is therefore invalid:

```
set font1 = 'ss admuusrp; col red'
```

Variables can affect more than a single command, as the following example shows:

```
)ops set cond1 = *
)ops set cond2 =
)ops &cond1 color red ; ss admuusrp
)ops &cond2 color yellow ; ss admuutrp
```

Here the full line prefixed with &cond1 is a comment (following the usual conventions); the line prefixed with &cond2 is executable.

How values are interpreted and handled

Variables are not declared, and have no fixed type. However, at any instance, OPS classifies a variable as one of:

- Integer
- Floating-point
- Word
- Quoted string
- General phrase

Here are some examples:

12	Integer
3.23	Floating-point
admuusrp	Word
'This is the header'	Quoted string
red 2 1 &c	General phrase

Interpreting numeric values

Numeric values are kept as integers as long as no decimal point appears in an expression, and as long as no floating-point function is invoked.

Floating-point numbers are stored internally with approximately six digits accuracy (full word floating-point). When presented as text, for example using the TEXT command, the precision is controlled by the OGRAIN command. The default setting is 0.01; that is, numbers are presented with two decimal places. For example:

```
set k = 13.26
ograin = .1
text 'There are &k km to Copenhagen from here'
```

displays:

```
There are 13.3 km to Copenhagen from here
```

Here are some more examples of the effect of OGRAIN rounding:

Value	OGRAIN	Output
28.346	0.01	28.35
28.346	0.5	28.5
28.346	5.0	30

Interpreting quoted strings

Quotation marks are used only around text parameters, and must not be used around general phrases. Variables are resolved within quoted strings as well as in infile text, unless, of course, you explicitly set variable substitution off. However, arithmetic expressions are **not** resolved within quotation marks, or in infile text.

Here are some examples:

```
)ops set &y = &$year+1
)ops set &y = 'is &y'
Next year &y, and then we'll see something...
```

```
)ops set &t = 'Note that the quotation marks are part of the value'
)ops text &t color red
)ops msg &t blink
```

```
)ops set c = 'color red'
)ops line & 20 30 &c
```

**gives syntax error because
of the quotation marks**

In the first example, the quotation marks are not required, but are accepted for consistency with text parameters.

The last example gives a syntax error, because quotation marks are not accepted in the line command.

Interpreting general phrases

Phrases are sequences of words or arithmetic expressions. A phrase cannot contain a quoted string. When assigning a phrase, OPS substitutes variables and evaluates expressions. For example:

```
set c = 3
set a = color &c+2
```

assigns the phrase `color 5` to the variable `a`.

To stop OPS substituting variables and evaluating expressions, use the `STR` function. In the following example:

```
set &attr = str(draw &col, fill &col+1)
set col = 2; box 10 20 5 &attr
set col = 6; box 30 20 4 &attr
```

concatenating variables

STR assigns the string in parentheses to *attr*. &col is not resolved, and the expression &col+1 is not evaluated. When executed the two BOX commands become:

```
box 10 20 5 draw 2 fill 3
box 30 20 4 draw 6 fill 7
```

To address a single word within a phrase, use a colon (:) as an indexing mechanism, as in:

```
set gdfs = ibm3179 ibm3827 ibm3090
gdf &gdfs:3      ;* display IBM3090
```

The index can also be an integer variable:

```
set i = 2
gdf &gdfs:&i     ;* display IBM3827
```

OPS resolves multiple indexes from right to left, for example:

```
set gdfs = ibm3179 ibm3827 ibm3090
set ix = 1 2 3
gdf &gdfs:&ix:2  ;* display IBM3827
```

You can't use arithmetic expressions on the index:

```
gdf &gdfs:&i-1      is invalid
```

The expression would resolve to IBM3827-1, which is invalid syntax. To overcome this, use an intermediate variable, as in:

```
set k = &i-1      ;* introduce intermediate variable
gdf &gdfs:&k
```

Concatenating variables

To concatenate text to the value of a variable, use a period (.) between the variable name and the text string. For example:

```
set c = IBM
gdf c.3827      ;* results in gdf IBM3827
```

The text to concatenate can come from a variable:

```
set c = IBM
set v = 3800 3820 3827
gdf &c.&v:2      ;* results in gdf IBM3820
```

This also shows that indexes and concatenation operators are evaluated strictly from right to left.

Another concatenation operation involves quoted strings. As a rule, an expression involving a quoted string becomes a quoted string. As arithmetic expressions are not resolved within quoted strings, you'll find concatenations useful when mixing arithmetic expressions with text:

```
text "Next year is "&$year+1", and then we'll see it."
```

Note: There must be *no* blanks between the quotation marks and the arithmetic expression.

Following the rule above, the arithmetic expression can also be concatenated in front of any text, as in:

```
text &$year+1' is next year'
```

Finally, here is how you display the coordinate values of the current point:

```
set cp = &          ;* OPS does not resolve & within text
text '&cp'         ;* so you need to use this form
```

Using system variables

OPS provides system (built-in) variables of three types:

Status values Contain all the status values you can set with OPS commands

Miscellaneous Contain year, date, and other common variables

Coordinates Contain the current point, and similar

With the exception of the current point variables (with references, &, &x, and &y), *all* system variables have names beginning with \$, for example \$color, and \$year.

You *cannot* assign a value to a system variable using SET. Status variables are set only by their associated status-setting command, miscellaneous variables by miscellaneous means, and the coordinate variables by graphics commands.

Status values

The status set by any status-setting OPS command is available in a variable of the same name as the command but prefixed by \$. For example, the current colors are available by referring to \$color.

The following example shows how to define and use a highlight color depending on the current color:

```
* normal color sequence: 1 2 3 4 5 6 7
set hcol =                5 3 6 6 7 2 2
text 'Hilites' color &hcol:&$color:1
```

Miscellaneous

OPS defines the following variable references:

\$line Current line (record) number within the file.

\$page Current page number.

\$pageline Line number within the file of the first line of the current page.

\$mouse The number of available mouse buttons. If there is no mouse, the value is 0.

\$rcmd Command field contents following a READCMD.

\$rcmdu The same as \$rcmd, but in uppercase.

\$readtag The tag value of the last item picked by READTAG.

\$rtag The same as \$readtag.

\$term The IBM product number of your terminal, for example 3179. If the terminal is not recognized, the value is UNKNOWN.

sharing variables

\$year	Current year in four digits, for example 1996.
\$month	Current month in two digits, for example 03.
\$day	Current day within a month in two digits, for example 12.
\$time	Current time as HHMMSS.
\$rc	The return code set in the external EXEC called in response to the EXEC command.

Coordinates

OPS defines the following variable references:

&x, &y x- and y-values of the current point.

Note: These two names are reserved. To change their values, use the START command.

\$xmax, \$ymax
Maximum x- and y-values.

The BOX command sets the following values:

\$xc, \$yc Center of the box.

\$xl x-value of the left-hand side of box.

\$xr x-value of the right-hand side of box.

\$yt y-value of top of the box.

\$yb y-value of bottom of box.

Sharing variables with ISPF

Use the ISPEXEC command to pass variables to and from the ISPF variable pools. The syntax follows ISPF:

```
ispexec vget variable-list
ispexec vput variable-list
```

where *variable-list* takes the form:

```
(variable-name, variable-name, ...)
```

Here are some examples:

```
ispexec vget zuser      ;* retrieve the user's userid
ispexec vput (a, b, $color)
```

The VGET service defines or replaces OPS variables of the specified names, which may be referred to in the usual way:

```
)ops ispexec vget (zuser,zenvir) ;* User/ISPF version
)ops text 'Hello &zuser. You are running &zenvir' 20 40 h 5
```

In keeping with ISPF conventions, don't include the ampersand (&) in variable names on the ISPEXEC VPUT service. If an ampersand precedes a name, variable substitution takes place before calling the PUT-operation.

Other ISPEXEC services, such as DISPLAY or SELECT must, for security reasons, be invoked using the OPS EXEC facility, described in “Executing functions” on page 171.

Querying variables

Use the QUERY command to display the value of any variable, for example:

```
query &a
q    &$color
```

The value of *a* or *\$color* is displayed in the message field:

```
&a = 12
&$color = 5 2
```

You can query **all** system-defined variables in one command, and get the values displayed in a scrollable window:

```
q sysvars
```

You can also display all user-defined variables, using:

```
q vars
```

You can also change variables, using MODIFY. It works the same way as QUERY, but puts the result of the query in the command field in a form suitable for modification.

For example, if you enter:

```
modify &a
```

OPS shows, in the command field, something like:

```
set &a = 12
```

You can then change the value, pressing Enter to implement it.

If you want to modify a system status variable, MODIFY gives you the command you need. For example, when you enter:

```
mod &$color
```

OPS responds:

```
color 2 5
```

ready to modify and reenter.

querying variables

Chapter 13. Executing functions outside OPS

This chapter describes commands that address functions outside OPS:

BROWSE	Browse an external file.
EDIT	Edit your infile.
EXEC	Execute an external function.
ICU	Access the GDDM Interactive Chart Utility.
ISE, VSE	Access the GDDM Symbol Set Editors.
CMS	Execute CMS command (VM only).
TSO	Execute TSO command (MVS only).
ISPF	Invoke ISPF function or panel (MVS only).

Browsing files

Use the BROWSE command to browse an external file:

```

browse profile exec           (CMS)
browse 'ispf.local.clist'    (TSO)

```

OPS invokes either ADM7SCAN EXEC (CMS) or ADM7SCAN CLIST (TSO). You can change this to another browse program by editing ADM7SCAN.

Editing files

Use the EDIT command to edit the infile, or the source to the infile, with either XEDIT (CMS) or ISPF/PDF (TSO):

```
edit
```

When you finish editing, OPS starts again.

The EDIT command has no operand, so you can't edit files other than the preselected one. If you need to, use the commands XEDIT or ISPF to start an editor with a free choice of file.

EDIT is assigned to PF12.

The EDIT command starts text editing of a preselected file called the **edit file**. This file is selected by the OPS startup procedure, which makes the selection open to local modifications.

Executing functions

Use the EXEC command to execute functions external to OPS. For security reasons, *all* external calls are invoked using an EXEC (or CLIST) of the same name as the OPS file. By modifying that EXEC, you can control which external functions are allowed. If the EXEC is renamed, for example, no external functions are available.

using the ICU

Any parameters on the EXEC command are passed unchanged to the external EXEC. For example:

```
)ops * This &OPS is named "TEST5 &OPS."  
...  
)ops set a = charlie  
)ops exec p1 25.7 &a
```

executes:

```
'exec test5 p1 25.7 charlie'          (CMS)  
%test5 p1 25.7 charlie                (TSO)
```

If the external exec is a CLIST (TSO), make sure that you supply the proper number of positional parameters. The actual call syntax is determined by the CLIST.

The return code from the EXEC is available in variable \$RC. Using the return code is the simplest way of sending messages from the EXEC to the presentation. Here's an example:

```
...  
)ops exec p1 25.7 &a  
)ops &$rc = 1 then goto page 23  
)ops &$rc = 2 then goto page 37
```

The sample ADM7OPDA shows practical examples of the EXEC command.

Using the Interactive Chart Utility

Use the ICU command to start the Interactive Chart Utility (ICU):

```
icu
```

This starts ICU and displays the ICU home panel.

You can also specify the name of a chart you want to restore at once, as in:

```
icu mychart
```

which starts ICU and restores MYCHART ADMCDATA and MYCHART ADMCFORM. The chart is shown at once and at its full size. Press PF12 to move to the ICU home panel, or press Enter to see the function of the PF keys. Return to OPS by pressing PF9.

You can also get a file list by putting * in the name:

```
icu *  
icu bx87*
```

The first command lists all charts; the second lists all charts beginning with BX87.

Use PF12 (edit-item) from the file list panel to enter ICU.

Note the difference between the ICU and CHART or CHARTX commands:

- ICU** Goes directly to the Interactive Chart Utility. You can produce and save new charts. You can only use it as a direct command.
- CHART** Adds an existing ICU chart to the graphics field of OPS. You can mix the chart with any other content in the picture.
- CHARTX** Displays an ICU chart on an empty screen. OPS keeps control. The chart will become larger than when using CHART (where two lines of the screen are reserved).

Remember that you can call ICU from the file list you get using the command CHART ?.

The ICU command makes it easy to produce OPS presentations using the GDDM Interactive Chart Utility. You can also produce your text foils using ICU, which includes many useful interactive facilities.

A clean ICU presentation could look like this:

```
1 )ops chart chart1
1 )ops chart chart2
1 )ops chart chart3
1 ...
```

When you scroll the presentation and want to make a change on a page, press PF6 (LASTcmd) to get the last command and thereby the name of the chart. On page 2 you will see in the command field:

```
chart chart2
```

If you change CHART into ICU:

```
icu chart2
```

and press Enter, you can make your corrections to the chart while under control of ICU.

Return to OPS by pressing PF9 (EXIT from ICU). Then press PF2 (REFResh) to get the picture updated with the changed chart.

Using symbol editors

VSE and ISE are parallels to the ICU command; they give direct access to the GDDM utilities: Vector Symbol Editor and Image Symbol Editor.

You can use VSE or ISE to inspect or change symbol sets. Remember that “symbols” can be drawings of anything; it’s up to you to decide whether an “A” should look like an A or a cat.

You can specify the name of a symbol set you want to change or inspect by:

```
vse admuwtrp
```

This starts the Vector Symbol Editor with the symbol set ADMUWTRP loaded, ready for edit.

If you want a file list, use the SS command:

executing TSO commands

```
ss admu*
```

From the file list you can enter the appropriate editor by pressing PF12 (edit-item). OPS selects either VSE or ISE depending on the symbol set type.

When you have changed a symbol set, you can use it immediately by selecting it with the SS command. However, if the symbol set has already been used, you need to ask OPS to release the old version. To save resources and speed up graphics, OPS does not load a symbol set that is already loaded.

Use the RELEASE command to release the old symbol set (see “Releasing symbol sets” on page 72):

```
release admuwtrp
```

Executing CMS commands

Use the CMS command to execute CMS commands belonging to the CMS SUBSET without leaving OPS. In this category are such commands as XEDIT, RDRLIST, and FILELIST. However, some are lacking, such as COPYfile.

To execute a command such as RDR (RDRLIST), type:

```
cms rdr
```

To execute CP commands you must explicitly add CP:

```
cms cp def graf 61
```

If you type CMS without specifying any operands, OPS enters CMS SUBSET, where you can enter commands to CMS as in the normal CMS mode. You return to OPS by typing:

```
return
```

Be careful when using CMS commands. Do **not** start other graphics programs while in SUBSET. When you return to OPS, the result is unpredictable (but seldom pleasant).

Executing TSO commands

Use the TSO command to execute any TSO command directly from OPS, for example:

```
tso send 'Ready for lunch?' u(charlie)
```

Do **not** start up any other graphics programs, especially OPS. When you return to the original instance of OPS, the status will be unpredictable.

If you run other user programs or CLISTs, be sure that they do not deallocate any files allocated to OPS. OPS *ddnames* begin with ADM.

Using ISPF functions

When you are running OPS in an ISPF environment, use the ISPF command to access ISPF functions. To access the PDF main panel (panel name ISR@PRIM), enter:

```
ispf
```

To access a particular option on the ISPF main panel, enter ISPF with the option number; for example:

```
ispf 2          (to get the PDF Edit panel)
```

To access a particular selection panel (menu), enter:

```
ispf panel panel-name
```

There are no restrictions other than the ones mentioned above under the TSO command.

using ISPF functions

Chapter 14. Scrolling

This chapter introduces the commands for scrolling backward and forward in a file with OPS.

The commands described here are commands you would typically issue from the command field. Chapter 11, “Controlling the sequence of execution” on page 157 tells you how to control scrolling from the infile.

Using scroll commands

You can scroll up and down in the file using the commands UP and DOWN, defined in PF7 and PF8, respectively. OPS doesn't let you specify the scroll size; it uses a fixed unit of a full screen, or a full document page if this is defined and if the page fits on a single screen. When scrolling up, OPS tries to place the top of a page in the first line of the data field.

You can specify a scroll amount together with the UP and DOWN commands: either a number of lines or a number of pages. The following commands scroll down 12 lines:

```
down 12 lines
d 12 l
```

The following commands scroll down 12 pages:

```
down 12 pages
d 12 p
```

The following command scrolls down to the next page:

```
down p
```

If you have PF8 defined as DOWN, just type the scroll amount in the command field, and press PF8 to execute the command.

If you do *not* specify the scrolling unit (LINE or PAGE), the unit is determined from the current value of CC (Carriage Control). If the infile is divided into pages (that is, CC is either A or M), the default unit is PAGE; otherwise it is LINE. In practice this means that when scrolling lists or OPS presentations, the default scroll unit is always PAGE.

You can scroll to the first or last page using:

```
up m
down m
```

or you can use TOP or BOTTOM.

You can also scroll using the LOCATE (L) command. LOCATE takes an argument of either a line number or page number, or a relative line amount or page amount. Here are some examples:

sideways scrolling

```
locate p 5          ;* scroll to page 5
locate l 342        ;* scroll to line 342
locate p +2         ;* scroll two pages on
locate l -120       ;* scroll backward 120 lines
```

You can put the operands in any order:

```
locate +2 pages     ;* same as third example above
locate -120 lines   ;* same as fourth example above
```

If you do not specify the unit in scroll commands, PAGE is assumed for infiles divided into pages.

To make scrolling easier, the LOCATE command is assumed where possible. The following are all valid LOCATE commands:

```
342 l
+20
+20 page
-5 p
p -5
```

In page-divided files, such as OPS presentations, you can scroll by typing numbers with or without sign, and pressing Enter.

You can use this for producing menu-like panels in an OPS presentation. As an example, look at the first page of the OPS tutorial. The page number referenced in UP, DOWN, and LOCATE is the internal page number in OPS. At startup, OPS initiates the page number to 0 and then updates it by 1 each time a page break is passed.

Synchronizing the page number

Use PAGESET to synchronize the page number in OPS with the current page number in a document, where you will often find page 1 somewhat after the beginning (for example, after pages i, ii, and so on). Here's an example:

```
pageset 1
```

sets the page number to 1.

Sideways scrolling

OPS doesn't include any commands for sideways scrolling. If the file is wider than the screen, adjust the size of the symbol set so you get enough space. However, many formatted documents are supplied with a left margin, having room for holes. Scrolling with the hardware symbol set can mean that you can't see some characters at the extreme right. Use the BIND command to adjust the document horizontally. For example:

```
bind 3
```

moves the document three positions to the left, cancelling the same command given during formatting with SCRIPT/VS. You can set BIND using an OPS startup operand.

"Indenting text" on page 80 describes how to implement left and right commands using the more advanced text facilities in OPS.

Setting and using labels

A label is a name you associate with a given position in the infile. You can set labels directly as well as from the infile. Only the first use is described here; infile labels are described in “Using labels” on page 158.

A direct label is a **single** alphabetic character prefixed with a period (.). You define it by entering it, as in:

```
Command ==> .a
```

You can enter the label character in upper or lower case; both are treated the same.

To go to a label, use the LOCATE command with the label as parameter:

```
locate .a
```

A direct label is always associated with top-of-page, so transfer to a direct label redisplay the associated page.

You can redefine the setting of a direct label (such as .a) at any time. The new position overwrites the old.

Refreshing the page

A series of variables determines the layout of a page on the screen (symbol set, color, indentation, and so on). When one of these variables is changed, the page must be refreshed to make the change visible, using the REFRESH command, as follows:

```
refresh
```

REFRESH can be abbreviated to REFR, and is assigned to PF2.

You can request that OPS refreshes the page automatically whenever a relevant variable is changed by typing:

```
refresh on
```

You cancel this with:

```
refresh off
```

REFRESH ON is the default mode for OPS presentations.

You can specify a status-setting command, such as COLOR or BIND, as an argument to REFRESH. For example:

```
refresh color red
refresh color red; bind 3
```

Note that **all** commands in the line must set status. The following command is not allowed:

```
refresh col red; down 5
```

It doesn't even make sense, because REFRESH refreshes the current screen, conflicting with the request to scroll down five lines.

Searching the infile

To search the infile, use the FIND and RFIND commands. The following:

```
find string
```

searches the infile for text containing the text string `string`. The string must be enclosed in quotes only if it contains imbedded blanks; for example:

```
find 'the needle'
```

You can abbreviate FIND to F.

OPS searches forward from the bottom of the screen. If it finds the string, OPS tries to show *all* the page where the string was found; the top of the page is placed in the top line of the screen and the page is displayed using the current symbol set and character size. The string is shown in a unique color if possible. If the string is not found on the screen (when the page cannot be contained in one single screen), OPS scrolls forward one screen at a time until the string is displayed.

To search repetitively for the same text string, use the RFIND command, assigned to PF5. RFIND starts from the bottom of the screen and searches forward in the file.

You can supply RFIND with search criteria, making the function exactly like FIND. This is most useful when you have assigned RFIND to a PF key.

A search for a text string is made after translation into uppercase, but no other translation is made of the infile (see translation tables in “Basic text handling” on page 65).

OPS searches only text (ordinary text lines in the infile), or text strings in the graphics text commands TEXT, HEAD, and FOOT. FIND COLOR will not find any COLOR commands in the infile.

Searching an infile containing symbolic variables

When an OPS presentation uses symbolic variables, searching for text with the FIND command becomes more complicated.

When you execute a FIND, OPS scans for the search string, from the current page-bottom forward, all infile text lines and all TEXT, HEAD, and FOOT commands. This process fails when the string is hidden in a symbolic variable, as shown by the example below:

```
1 %*page 7
  %set n1 = 'Charlie'; set n2 = 'Betty'
  %text &n1 10 30 col red
  ...
1 %*page 8
  %text &n2 10 30 col pink
  ...
```

The string Betty does not appear in any text context.

OPS therefore scans SET commands for the search string, and when successful starts playing the presentation from that page onwards, looking for the string to appear in a write-to-screen operation. Nothing is shown on the screen until the string actually appears, but as all commands are processed, the FIND takes longer.

Automatic scrolling

To scroll a file or part of a file without any user interaction, use the AUTO command:

```
auto
```

AUTO without any operands scrolls through the file until end-of-file, pausing for two seconds between each page. At end-of-file, OPS exits AUTO mode and returns control to the user.

Note: It is *not* possible to interrupt OPS during automatic scrolling.

AUTO-scrolling can be limited to a certain number of pages, and you can also specify a pause length in seconds. For example:

```
auto 5 10
```

specifies scrolling five pages with a pause of 10 seconds between each page. You can also specify the pause as a keyword with value, so you do not have to specify a page count, as in:

```
auto pause 10
```

Any WAIT commands without time specification in the infile (see Chapter 10, “Creating dynamic sequences” on page 151) are replaced by pauses of one second so that they do not stop the automatic scrolling.

You can specify AUTO as an operand at startup.

Note: You can make a non-stop presentation by ending the infile with a TOP command and then starting the display using an AUTO command. The TOP command prevents OPS from reaching end-of-file, so that the automatic scrolling continues. The only way to cancel is to switch off the panel.

During automatic scrolling, OPS can output all pages it finds as ADMPRINTs, GDFs or PSEGs. This is described in Chapter 15, “Printing and plotting” on page 183.

automatic scrolling

Chapter 15. Printing and plotting

The print and plot commands of OPS are:

PRINT	Create ADMPRINT for queued printers and plotters
PLOT	Plot on directly connected plotter
PSEG	Create page segment

The connection between the OPS commands and external programs and devices is illustrated in Figure 65.

The figure uses a convention to show connections, as follows:

- The filled arrows indicate a direct connection between OPS commands, print or plot files, and devices. These connections are fully explained in this chapter.
- The hollow arrows indicate that OPS acts only as an indirect agent. This applies to GL plot files and PostScript print files. These connections are introduced here; for more information, see the referenced GDDM books.

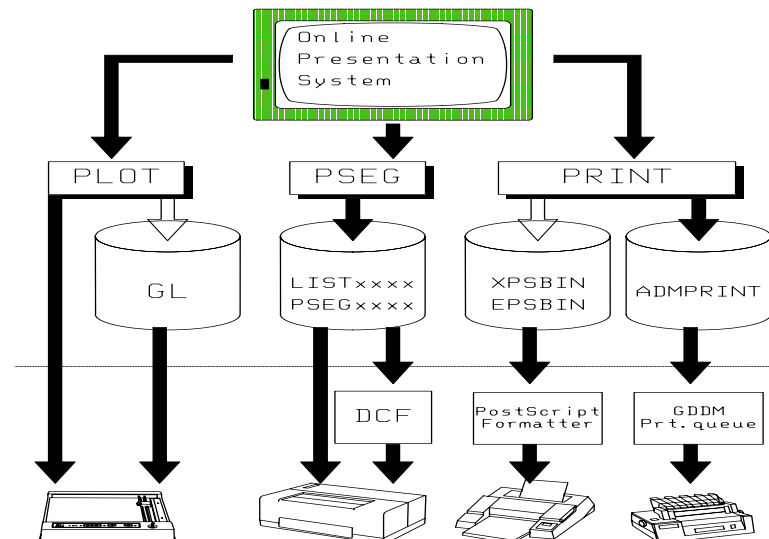


Figure 65. Print and plot from OPS

Printing panels

PRINT copies the current panel, both graphics and alphanumeric fields, into an ADMPRINT file, which you can print on a color matrix printer such as the IBM 3287, 3268 or 4224 printer. If you have GDDM Central Slide and Plot Facility (CSPF) or a similar product installed, ADMPRINT files can also be used to drive queued plotters such as the IBM 6182.

How the actual printing or plotting is done depends on the operating environment:

- Under CMS, OPS creates a file of type ADMPRINT; use your local procedures to print it.
- Under TSO, you can address a printer directly using the print file name. If you are familiar with printing from Interactive Chart Utility, you can apply the

procedures used for that product; they will be the same for OPS. Otherwise ask your local expert for advice.

The name of the ADMPRINT file (or the name of the printer under TSO) must be specified the first time you use the PRINT command in an OPS session:

```
print myprint
```

Subsequent use of PRINT without any operands will append the current screen image to the last specified ADMPRINT file, which contains the screen images without any separator pages.

If you set a PF key to PRINT, you can append screen images to an open print file by pressing that PF key.

If you want more than one copy, specify the number after the file name:

```
print myprint 2
```

The default width of the print is 80 columns on the printer, approximately the width of A4 vertical format (210mm).

If you use vector symbol sets only, you can enlarge the print without distortion. You specify the width as a number of print columns *after* specifying the number of copies. For example:

```
print myprint 1 132
```

This gives one copy with print width 132, which is full paper width.

The default depth is 80 rows, which is the largest value accepted by GDDM. In practice this means that the print or plot size is controlled by the width alone, since the depth is not restrictive.

For certain devices you may get a GDDM error saying "INVALID SIZE," referring to the depth of 80. In this case specify a lower value for the depth immediately after the width:

```
print mypict 1 92 46
```

OPS supports automatic scrolling and generation of an ADMPRINT file at the same time, creating contiguous hardcopy of a presentation. You can do this using PRINT as a keyword on the AUTO command:

```
auto 5 print myprint
```

The example automatically scrolls five pages forward and copies the five screen images into a single ADMPRINT file named MYPRINT.

Note: Dynamic sequences with overpainting and so on cannot be shown correctly on a printer. You can print, however, and use it as a kind of support material for the presenter.

Creating PostScript output

You can create PostScript output from OPS, but not directly. You use the PRINT command to output to a nickname that you have set up in an ADMDEFS profile file.

For details on how to do this, see *GDDM Base Application Programming Reference*.

Plotting presentations

Use PLOT to copy the current screen to an attached plotter such as the IBM 7371/7372 or 6180 color plotter. (If your plotter is centrally controlled, use the PRINT command for plotting.)

The PLOT command with no operands creates a plot with a pen velocity of 10cm per second, which is suitable on transparencies. You can get a faster plot (50cm per second) using the keyword FAST:

```
plot fast
```

If you have any plotter definitions in your nickname file (PROFILE ADMDEFS), you can refer directly to one using its nickname, as in:

```
plot myplot
```

GDDM controls plotting using the PLOT command, which makes sure that hidden lines on the screen (from overpainting) remain hidden on the plot. In other words, dynamic sequences with overpainting can be reproduced as a sequence of plots (use PLOT at each WAIT); the generated plots are identical to the screens.

However, plotting has the following technical limitations:

- You cannot scale image symbol sets, so they will seldom be shown in the correct size.
- Image symbols as well as images are drawn point by point and could damage your plotter pens. **Avoid IMAGE symbol sets and IMAGES.**
- Dotted lines (line style 1) are also hard on the plotter. If you plot from NORMAL mode, remove the dotted page limits with the command "MARGIN OFF."
- Only fillings made with the default patterns of GDDM (not the general colors in OPS) are reflected on a plotter. Furthermore, the patterns on the plot will be different from what you see on the screen.
- 7371/7372 plotters have only six colors, while most terminals have seven. Yellow and white share pen number 6, which is normally set to black. Thus, white words in a yellow text (which look fine on the screen) cannot be distinguished on a plot.
- Wide lines are very time-consuming to plot. A wide frame may take 10-15 minutes. Be aware of this when planning your work.
- Likewise, filled (shaded) symbol sets are very time-consuming. Examples are ADMUUKSF and ADMUUARP.

Creating GL files

You can create GL files from OPS, but not directly. You use the PLOT command to output to a nickname that you have set up in an ADMDEFS profile file.

For details on how to do this, see *GDDM Base Application Programming Reference*.

Creating page segments (PSEGs)

The PSEG command creates **page segments** (PSEGs) for inclusion in documents to be formatted by IBM Document Composition Facility (DCF) and subsequently printed on page printers such as the IBM 3812, 3820-family, and 4250 printers.

A page segment is a copy of the graphics field, rasterized to yield a picture of a given size on a specific device. For example, a picture 100mm wide on a 4250 printer requires many more pels than a similarly sized picture on a 3820 printer, because the pels density is much higher on the 4250 (600 pels per inch versus 240).

Optionally you can create a document with a single page segment, ready to print, or you can create an unformatted black-and-white image of the screen.

Note: PSEGs can be included in your pictures using the IMAGE command described in “Basic image display” on page 147.

Controlling output under CMS

Under CMS you must specify the filename of the output file, and you may optionally specify the filetype and filemode. If you specify only the filename, OPS selects a default filetype governed by the device token. The default filetypes are displayed along with the aliases in response to PSEG ?. If you are using a true GDDM device token (rather than an OPS alias), the default filetype is ADMIMAGE. Here are some examples:

```
pseg a4      mypseg,    100mm
pseg img600x seg01,   100mm
pseg a4      mypseg pseg38pp t, 100mm
```

The first example creates MYPSEG PSEG3820 A if the supplied alias A4 is unmodified. The second creates SEG01 ADMIMAGE A. The third creates MYPSEG PSEG38PP T.

PSEGs require a lot of disk space, so you might want to put them somewhere other than your A-disk. You can do this by:

- Using a filemode parameter on the PSEG command
- Using the PSEGLIB command

To specify a filemode on the PSEG command, give the full name of the page segment to be created, as in:

```
pseg a4 myseg1 pseg38pp b 100mm
```

which writes the page segment MYSEG1 PSEG38PP on your B-disk.

The same effect is achieved by the following commands:

```
pseglib b
pseg a4 myseg1 100mm
```

The latter approach assigns the B-disk as the output target for *all* subsequent page segments.

You can find out which disk is being used for the page segment library using:

```
q pseglib
```

Controlling output under TSO

Under TSO, first decide whether you want to output the PSEG to a sequential data set, or as a member of a partitioned data set.

For sequential data sets, specify the data set name on the PSEG command. If you don't use quotes round the name, OPS constructs it using the TSO profile prefix value as the high-level qualifier. Here are some examples:

```
pseg a4      mypseg.pseg  100mm
pseg a4      'abc.seg01.pseg38pp',  8.3cm
pseg a4      betty 3.5i
```

The first example creates a data set named *userprefix.MYPSEG.PSEG*. The second example creates the data set named in the command. The third example creates a data set called *userprefix.BETTY.ADMIMAGE*. You can change the qualifier ADMIMAGE by setting the filetype value in the OPS device token.

In all three cases, if the data set named already exists, the contents are overwritten.

Note: Set PSEGLIB *off* to use sequential data set output.

For partitioned data set members, first define the data set name, either in the ADMOPSLx EXEC, or using the PSEGLIB command. If you don't use quotation marks round the name, OPS constructs it using the TSO profile prefix value as the high-level qualifier. Here are some examples:

```
pseglib 'p1234.test.pseglib'
pseglib pseg3820
```

The second example defines *userprefix.PSEG3820* as the PSEGLIB.

You can clear the PSEGLIB allocation using:

```
pseglib off
```

You can query the PSEGLIB allocation using:

```
q pseglib
```

Once you've used the PSEGLIB command to define the PSEG partitioned data set, you can specify the member name directly on the PSEG command, as in:

```
pseglib test.pseglib      ;* can be done at startup
pseg a4  charlie 150mm
```

This creates the member CHARLIE in *userprefix.TEST.PSEGLIB*.

Page segment formats

A page segment is usually a copy of the graphics field in rasterized (uncompressed image) format. However, you can also create page segments using the graphics defined by GOCA graphics orders, and the images defined by IOCA image orders. You do this by choosing appropriate GDDM device tokens or processing options, as described in *GDDM Base Application Programming Reference*.

You can't input page segments of this format into OPS using the IMAGE command; you can only input page segments containing uncompressed images into OPS.

Converting colors into gray shades

OPS can convert any color automatically into one of 33 levels of gray when a screen is saved as a page segment. The conversion takes color as well as pattern into account, making the hardcopy look as much as possible like a black-and-white photograph of the colored screen.

You can explicitly ask for no conversion by specifying NORecolor on the PSEG command:

```
pseg a4 mypseg1 100mm nor
```

No conversion means that all solid colors and all general colors are reproduced as solid black, and that all patterns become independent of color.

Previewing the result

You can preview the result of the color conversion on your screen if your terminal lets you load pattern sets. Pattern set loading is supported by all non-programmable terminals, the 3270 PC/G(X), and PS/2s running GWSP.

To preview a PSEG, specify PREview as a parameter following the size:

```
pseg a4 myseg2 190 pre
```

If you preview a PSEG on a terminal without support for pattern loading (for example, GDDM OS/2 Link), all general colors appear black.

Automatic scaling of images

When you create PSEGs, images are scaled.

If images were copied directly from the screen to the page segment, the size would appear correct only if the page segment had the same number of pels as the screen. OPS gets round this by rescaling all **known** images in the picture during PSEG creation, using the original image source. A **known** image is one created by an IMAGE command, or COPYed from such an object. Images added using ADMGDFs are **unknown**, and not rescaled.

Creating PSEGs with AUTO

The AUTO command lets you create PSEGs during auto scroll. The parameters are the same as for the PSEG command, for example:

```
auto pseg 3812 myops 150
```

This creates PSEGs for a 240 pels/inch device. The page segments created have names built by the base-name (MYOPS), followed by a three-character number starting from 001. So, the PSEGs are called MYOPS001, MYOPS002, and so on.

Note: This limits the base-name to five characters.

The print width of each segment is 150mm.

Chapter 16. Using dialed terminals under CMS

Under CMS, OPS lets you transfer pictures to a maximum of nine terminals dialed to your virtual machine, which is called the **primary terminal**. The others are called **secondary terminals**.

This lets you show presentations using several terminals. You can, for example, use a separate terminal on which to show the agenda or other important figures, or copy screens from one terminal to another. The secondary terminals do *not* have to be in the same place as the primary terminal, so you can create a video conference using OPS.

Establishing secondary terminals

You establish a secondary terminal in two steps: first define it from the primary terminal, then DIAL it from the secondary terminal.

A secondary terminal need not be the same type of terminal as the primary, but it must be a graphics terminal.

Defining a secondary terminal

Define secondary terminals as GRAF devices on the primary virtual machine using addresses in the range 061 to 069. You do this using CP DEFINE commands:

```
cp def graf 61
cp def graf 62
...
```

You can add these commands to your PROFILE EXEC to establish them permanently. You can also execute them from the command field in OPS:

```
cms cp def graf 61
...
```

You need to make the definition only once per session. If you repeat the definition, OPS tells you that a particular GRAF is already defined.

You detach a defined GRAF using:

```
cp det 61
```

from outside OPS, or:

```
cms cp det 61
```

within it.

Dialling from a secondary terminal

A secondary terminal is attached to the primary terminal by using DIAL instead of LOGON. At the point in the logon process on the **primary nodeid** where you normally enter LOGON *userid*, (or L *userid*), type:

```
dial primuser
```

where primuser is the primary user's userid.

issuing commands

Note: You do *not* need to have your own userid in order to dial the primary user.

DIAL is answered by:

```
dialed to primuser as 061
```

for the first attached terminal, "062" for the next, and so on.

If the primary user does not have any available GRAF device defined, you get this message:

```
LINE(S) NOT AVAILABLE ON primuser
```

The DIAL process must take place on the primary nodeid, which does not have to be your own nodeid. It can even be in another country, but of course it must be "visible" on the network.

Figure 66 summarizes the process.

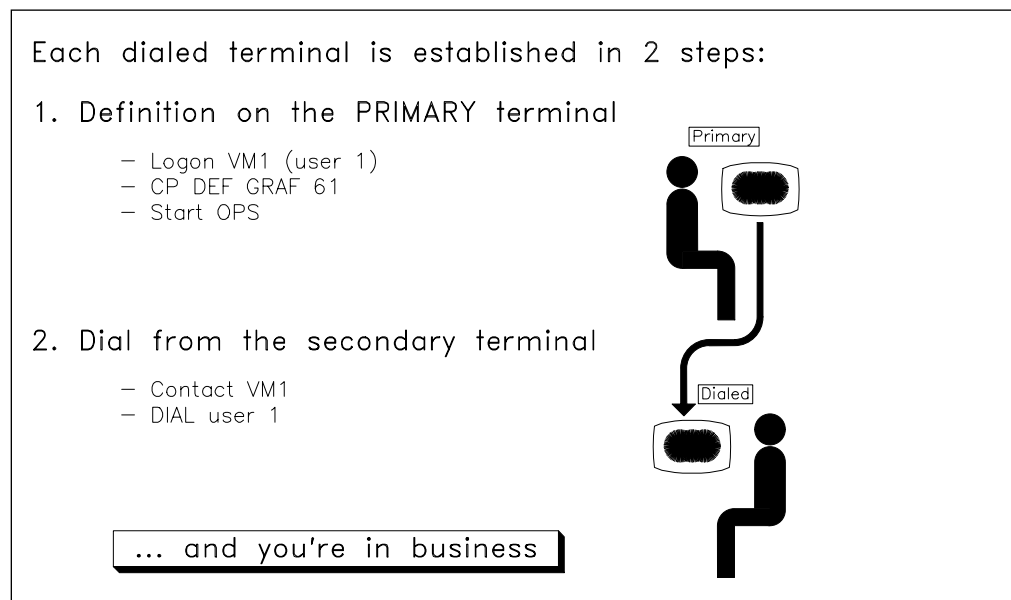


Figure 66. Establishing a dialed terminal under VM

Issuing commands from a secondary terminal

When you have established a secondary terminal, you can immediately use it from OPS on the primary terminal. Secondary terminals are supported by the following commands:

CLEAR Clear the screen
SCOPY Copy the screen
CHARTX Show an ICU chart
GDFX Show a GDF

All these commands, except SCOPY, are described elsewhere. On any of them, you can give an address or "id" of a secondary terminal as the last operand (for CLEAR and SCOPY, it is the only operand). For example:


```
clear 61
scopy 61
chartx mychart 62
```

The address is the address of the virtual GRAF device. As an alternative you can use an "id," where "id" is a number from 1 to 9 (the address less 60).

```
scopy 1
gdfx adm7samp 2
```

CLEAR can also clear all secondary screens simultaneously:

```
clear all
```

Copying the graphics field to a secondary terminal

Use the SCOPY command to copy the graphics field of the primary screen to a secondary screen. The graphics field of the primary screen does not have to be displayed. It exists internally in OPS as soon as a command is drawing in the graphics field. Here's an example:

```
1 )ops reset; ss admuwxsf; h 25; col y
  )ops text '1' 50 50% cc
  )ops scopy 1 ;* Display a 1 on the address 061
  )ops clear ;* Clear the graphics field
  )ops text '2' 50 50% cc
  )ops scopy 2 ;* Display a 2 on the address 062
  )ops clear ;* Clear the graphics field
  )ops text 'Primary' 50 50% cc
1 ...
```

The primary screen displays the word Primary; the two secondary screens display 1 and 2, respectively. 1 and 2 are never displayed on the primary screen.

SCOPY makes it easy to show text or graphics to remote colleagues. Start OPS against the file of interest, and set, for example, PF4 to SCOPY 1:

```
pfk 4 'scopy 1'
```

The file is scrolled in the usual way with PF7 and PF8. You can use FIND to locate anything you want to find. The current screen is copied if you press PF4.

copying the graphics field

Chapter 17. Getting help

You can get help in using OPS in six ways:

1. OPS tutorial
2. OPS help file
3. Command syntax help
4. File lists
5. Graphics overviews
6. Query commands

Using the OPS tutorial

The OPS tutorial is a graphics introduction to OPS, aimed at the new user. It is also an OPS presentation, so you can use it as an example for your own presentation.

You can call the OPS tutorial by typing:

```
amops1x adm7opta
```

where *x* is your language version.

The first page is a menu from which you can go directly to any topic of interest. Each topic is headed by a submenu allowing direct access to any individual page. You can press Enter to go through the presentation sequentially.

Displaying the OPS help file

Press PF1 twice when the command field is empty, or once as step 2 in the HELP command (see “Getting command syntax help” on page 196), to get to the OPS help file.

The help file is a flat file including the following information:

1. How to use the help facilities
2. Command overview—all commands on one screen
3. Commands arranged according to functions, one per line
4. Command reference—commands in alphabetic order with explanation
5. Basic rules for an OPS presentation
6. Coordinate system and how to express coordinates and dimensions
7. Symbolic variables
8. Arithmetic expressions
9. How to develop a presentation
10. Using dialed screens under CMS

Finding your way round the help file

You can find your way round the help file using the following commands, which look like the equivalent OPS commands:

- | | |
|-----------------------|--|
| n | Scroll directly to section n . n is a number between 1 and 8. |
| Down <n> | Scroll down n lines. If you omit n , OPS scrolls down one screen. The argument M is equivalent to BOTTOM . DOWN is assigned to PF8. |

getting help on commands

Up <n>	Scroll up n lines. If you omit n , OPS scrolls up one screen. The argument M is equivalent to TOP . UP is assigned to PF7 .
TOP	Scroll to the top. TOP is assigned to PF10 .
BOTtom	Scroll to the bottom. BOTTOM is assigned to PF12 .
Find string	Searches the help file from the top for string . The file is positioned with the current line next to the top line on the screen. The string is shown in yellow. The cursor is placed below the first character of the found string.
RFind <string>	If you specify a string , this string is searched for. Alternatively, the search is continued for the string specified on the last FIND or RFIND command. The search is made from the second line and forward. RFIND is assigned to PF5 .

You can find a command explanation in one of the following ways:

- Enter the command word and press **PF1**.
- Place the cursor below any letter in a command word anywhere in the help file and press **Enter** (or **PF1**).

The help file is positioned with the explanation of the command at the top of the picture. The command word is displayed in yellow.

Getting command syntax help

To get help on a particular command:

1. Write the command word, such as **PRINT**, in the command field and press **PF1**. OPS displays a compact syntax example in the command field; for example:

```
* print <name <copies <width>>>
```

* in front shows that this is a comment. Remove it, and modify the command to suit your needs.

2. Press **PF1** once more to get to the place in the help file where the command **PRINT** is described.

Step 1 is skipped if the command field contains anything besides the command word, `print myprint` for instance.

This is useful when you want help while entering a command, and when you have entered an incorrect command and need help to understand the error message.

For commands with positional operands, the compact command syntax is also used as an error message when you enter the command word by itself. Type:

```
arrow
```

and press **Enter**. OPS shows the following message:

```
Syntax is---> ARROW x y ... <W .> <CU | R .> (and so on)
```

You can use this instead of **PF1**, but only when the command requires operands (otherwise the command is executed).

Bypassing the help menu and command syntax help

For special applications of OPS, such as for management information systems, where the user only needs PF keys to scroll through charts and text, you don't need all the help available.

For such applications, you can produce a dedicated, short help file, perhaps only one screen, to display when the user presses PF1, or writes HELP or ?, regardless of what is in the command field.

To bypass the help menu and the command syntax help, do one of the following:

1. In the presentation, place the command:

```
helpmode short
```

2. Start OPS with the operand HELPMODE SHORT.

See Appendix A, "OPS startup procedures" on page 201 for more detail.

Displaying file lists

You can make queries about things like available symbol sets from the Help menu or by putting a question mark where the symbol set name is required. For example:

```
ss ?
ss *
```

You can display the available ADMCDATA (from the Interactive Chart Utility), ADMGDF, and image files by:

```
chart ?
gdf ?
image ?
```

where you can also replace ? by *.

You can limit the search to a certain subset of the files by specifying a pattern that the file name has to fit. The pattern is formed exactly as with CMS LISTFILE. Examples where ? is understood are:

```
gdf ibm*
ss *tr*
```

In the first example, all GDF files beginning with IBM are selected. In the second, all symbol sets with TR somewhere in the name are selected.

You get a list of file names shown on a panel looking like this:

displaying graphics overviews

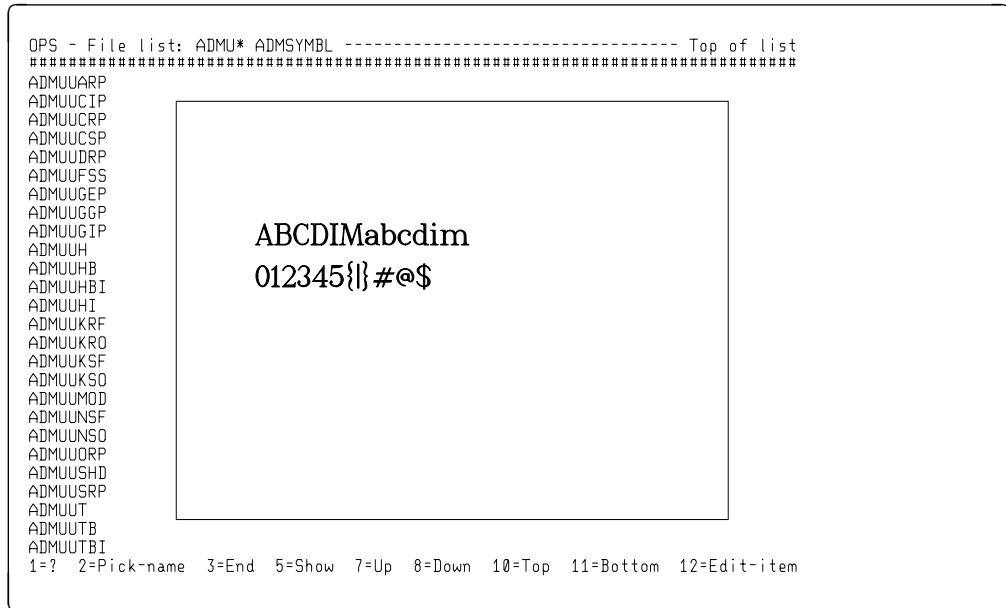


Figure 67. The file list panel

You can look at a sample symbol set, a chart, or a picture by placing the cursor at the name and then pressing Enter or PF5.

You can edit symbol sets and ICU charts (using GDDM Image, Vector Symbol Editor, or Interactive Chart Utility) by pointing at the file name you want and pressing PF12.

If you want to pick a file name and bring it into the original query, point at the name and press PF2. This query:

```
ss admu*
```

followed by selecting ADMUVTIP in the file list panel results in:

```
ss admuvtip
```

in the command field on the main panel.

You can also get a list of symbol sets if you use SS as a **subcommand** on, for example, the TEXT command. An example is:

```
text 'hello' color red ss admuw*
```

If you pick a symbol set name from the list, the TEXT command is changed accordingly:

```
text 'hello' color red ss admuwtip
```

Displaying the graphics overviews

You can display the options of color, line styles, and patterns, as well as arrow heads and tails, either from the help menu or in the same way as for file list (that is by specifying a ? instead of an operand value). For example:

```
color ?
```

The options are displayed on a picture that covers almost the entire screen. You can modify the command while the help is being displayed.

Help for patterns covers two pictures. You can jump between these using PF7 and PF8 (UP and DOWN).

The help disappears from the screen when you execute a command or press Enter.

You can get help from any command with a graphics operand:

```
Colors :    COLOR
Lines  :    DRAW    FRAME
Patterns:    FILL    WFILL    BACKGR
Arrows :    ARROW
```

You can also get help when the command in question is used as a subcommand, as in:

```
text 'hello' color ?
box 50 50 40 draw ? fill ?
```

In the help file and the command syntax help, the help option is shown by <?> at the end of the syntax.

Querying the value of system variables

You can find the value of system variables using a QUERY command or the parallel command MODIFY as shown below:

```
query ss
q ss

modify ss
mod ss
```

Both commands show you which symbol sets are currently active. QUERY displays the information in the message field; MODIFY uses the command field, ready to use again after modifying.

You can make queries of all variables you can set yourself (such as COLOR, SS). Use ? QUERY for a complete list.

QUERY can also tell you about variables that you cannot modify (you can't do this with MODIFY). Here are some examples:

```
q infile      ;* shows the name of the infile
q line        ;* shows the record number of the first line on the screen
q ss all      ;* displays all loaded symbol sets
q segment     ;* prompts you to pick an object in the picture, and then
               ;* displays the graphics segment number of the picked primitive
q pf          ;* shows the value of all 12 PF keys in a window
```

querying system variables

Appendix A. OPS startup procedures

This appendix describes the startup procedures for OPS under TSO or CMS.

Starting OPS under TSO

The ADMOPSLx EXEC is used to start OPS under TSO. The value of “x” depends on which language version of the EXEC you are using. Throughout this book, we use ADMOPSLA, the US English version of the EXEC. For a list of other EXECs, see “Choosing the EXEC for your language” on page 5.

The ADMOPSLx EXEC allocates the files required by OPS: the infile, the help file, and any customization file; and data sets for symbol sets, images, ADMGDF, ADMCDATA, ADMCFORM, and graphics output. Customize the EXEC to suit your own environment before you use it, as described in “Customizing the ADMOPSLx EXEC under TSO” on page 204.

For ADMSYMBL, ADMGDF, ADMCDATA, ADMCFORM, and ADMIMG input, OPS considers three levels:

1. Project (read only—created at your enterprise)
2. User (read/write—your personal data)
3. OPS (read only—supplied by IBM)

The data set prefix for the user level takes the value of the current TSO userid, or whatever you specify when you customize the EXEC.

You provide the data set prefix for the project level, using two EXEC parameters: PROJECT and LEVEL. If you omit them, the project level is ignored. If you specify them, you are asked whether you want to continue if any of the input datasets are missing. The EXEC does *not* create them.

The OPS data sets are the GDDM system data sets, and must be specified when you customize the EXEC.

For graphics output, only the user level is considered. You can output to any of the four data set types: ADMGDF, ADMSYMBL, ADMCDATA, and ADMCFORM. If you specify the EXEC parameter ALLOC, OPS prompts you to allocate any missing output data set. If you don't do so, OPS disables the associated output functions if the output data set is missing.

If you specify BROWSE, OPS allocates no output data sets.

Note: You can concatenate data sets only if they have similar values for attributes such as RECFM and LRECL. To avoid excessive overhead, the ADMOPSLx EXEC does *not* verify data set attributes. Your local standards should ensure that similar data sets have similar attributes. While GDDM is able to handle a mixture of blocked and unblocked data sets, ISPF, which is used to create file lists, is not. An ISPF error may result if such a mixture is attempted.

```

The syntax of the ADMOPSLx EXEC is:

ADMOPSLx infile ( < options >

Options:
    <AUto>
    <ALloc>
    <Blnd n>
    <BRowse>
    <CHeckd>
    <CMd (command-string)>
    <COLor color | ('color,color<,...>')>
    <CUst custom-file>
    <DARk | NOrmal>
    <HElpmode Short | Full >
    <LEvel level>
    <MArgin x | OFF>
    <MSg>
    <PROJect project>
    <SS ss-name>
    <TRC>
    
```

Figure 68. ADMOPSLx EXEC syntax under TSO

ADMOPSLx EXEC parameters under TSO

The parameters for ADMOPSLx EXEC are described below. All of them are optional. The notation is as follows:

- Keywords (such as **PROJect**) begin with an uppercase character. Any abbreviation down to the uppercase part (**PROJ**) is accepted.
- Values such as *project* must be substituted by something appropriate.
- The vertical bar “|” denotes a choice.
- Optional items are enclosed in <>.
- Defaults are underlined, for example Normal.

infile

The identification of the infile. The record format can be fixed or variable, with a maximum record length of 255 bytes or less.

infile is an unqualified name (maximum eight characters).

If you omit *infile*, OPS assumes ADMOPF0A, which must exist in the system OPS data set SADMOPS.

If you specify an *infile*, OPS searches for it using the following sequence:

1. project.level.ADMOPS (if you specify project or level)
2. userprefix.ADMOPS (if the data set exists)
3. The system data set SADMOPS

If OPS fails to find *infile* in any of these, it prompts you to create a new one, as follows:

1. Create a userprefix.ADMOPS partitioned data set (if it doesn't already exist).
2. Create a new member in this data set.
3. Copy the skeleton presentation, ADM7OPXA, or create an empty presentation.

AUto

Scroll through the infile without user intervention.

ALloc

Prompt for allocation of any missing user data set for graphics output. User data sets have a first qualifier equal to the userid, or whatever you specified in customizing ADMOPSLx EXEC for your installation. Output may occur in any of the four ADM data set types: ADMSYMBL, ADMGDF, ADMCDATA, and ADMCFORM. (Your installation may have changed the naming convention.)

Blnd *n*

Adjust text lines *n* columns to the left.

BRowse

Do not allocate any data sets for graphics output. Use this option to reduce overhead at startup, when you know that no output will be created.

CHeckd

Check the device for graphics capability *quietly*. If graphics cannot be shown, return without a message, and with a return code of 20. Use this to select an alternative display program and file, when using OPS in an information system.

CMd (*command-string*)

The initial OPS command, executed after opening the file. You can specify any direct command.

Note: The total length of the parameter string passed to OPS must not exceed 100 characters.

COlor *color* | ('*color,color<,...>*')

The initial colors for text. You can specify up to four colors, in the same way as you do with the COLOR command. You need parentheses only when you specify more than one color.

CUst *custom-file*

Use the customization file *custom-file* (maximum eight characters). A customization file must be a member of the system data set SADMDAT. If you omit CUST, OPS looks for ADM7OPC.

DARk | **NOrmal**

Run in DARK or NORMAL display mode.

HElpmode Short | **FuLL**

Set HELPMODE to SHORT or FULL,

LEvel *level*

The second-level qualifier of the "project" data sets for presentations (last qualifier ADMOPS), symbol sets, GDFs, and ICU charts.

MArgin *x* | **OFF**

Specify a non-default margin display. *x* is any valid OPS x-coordinate, such as 45m.

MSg

Show the initial message (“Use PF1 for help”), even in DARK mode.

PROJect *project*

The high-level qualifier of the project data sets for presentations (last qualifier ADMOPS), symbol sets, GDFs, and ICU charts.

If you specify L`Level`, the default for PROJect is the TSO userid. If you omit L`Level` and PROJect, OPS ignores these data sets.

SS *ss-name*

Initial symbol set to load. This symbol set is also the default symbol set (the one specified by RESET), unless you modify it with a DEFAULTS command.

TRC

Assume TRC codes in column 2.

Customizing the ADMOPSLx EXEC under TSO

Before you run ADMOPSLx EXEC, customize it to suit your installation. There are two types of customization: things you **must** do (essential customization) and things you **can** do (optional customization).

Essential customization

You must define a high-level qualifier for the system datasets.

To run the sample presentations, all the system data sets must be present, because they contain the necessary presentations, GDFs, and so on.

To run user presentations, the minimum system data sets you need to define are:

- systemprefix.SADMDAT for help, customization, and profiles
- The system load data set (SADMMOD)
- The user data sets

Optional customization

Here are other things you might choose to do:

- Add extra common input data sets for symbol sets, GDFs, charts, and images
- Change the default user data set prefix from the TSO userid to your installation convention
- Include a common PSEG data set in which to save page segments

Note: The user output data sets for GDFs, charts, and images will only be allocated if they exist.

Starting OPS under CMS

Use the ADMOPSLx EXEC to start OPS under CMS. The value of “x” depends on which language version of the EXEC you are using. Throughout this book, we use ADMOPSLA, the US English version of the EXEC. For a list of other EXECs, see “Choosing the EXEC for your language” on page 5.

ADMOPSLx EXEC follows the usual CMS conventions, so you can do things like use it from FILELIST. You could also create a panel interface to the ADMOPSLx EXEC using, for example, the ISPF Dialog Manager.

Normally you don’t need to specify any options; they are mainly for use when invoking OPS from another EXEC as part of an application.

The syntax of the ADMOPSLx EXEC is:

```
ADMOPSLx <fn <ft <fm>>>< ( options>
```

Options:

```
<AUTO>
<BIND n>
<CHECKd>
<CMD ( command-string )>
<COLor color | ( color ... )>
<DARK | NORMAL>
<Edit edit-filetype>
<Helpmode Short | Full>
<MARGin x | OFF>
<MSG>
<OPS>
<SS ss-name>
<TRC>
```

Figure 69. ADMOPSLx EXEC syntax under CMS

ADMOPSLx EXEC parameters under CMS

The parameters for ADMOPSLx EXEC are described below. They are all optional. The notation follows the rules:

- Keywords (such as **COLor**) begin with uppercase characters. Any abbreviation down to the uppercase part (for instance **COL**) is accepted.
- Values (such as *color*) must be substituted by something appropriate.
- The vertical bar “|” denotes a choice.
- Optional items are enclosed in <>.
- Defaults are underlined, for example Normal.

fn ft fm

The identification of the **infile**. The record format may be fixed or variable, with a maximum record length less than 256 bytes.

Defaults are:

fn: ADMOPF0A

ft: ADMOPS

fm: *

If you omit all the parameters, these defaults mean that OPS will display its sample presentations

AUTO

Scroll through the infile without user intervention.

BIND *n*

Adjust text lines *n* columns to the left.

CHECKd

Check the device for graphics capability *quietly*. If graphics cannot be shown, return without a message and with a return code of 20. Use this to select an alternate display program and file when using OPS in an information system.

CMD (*command-string*)

The initial OPS command,executed after opening the file. You can specify any direct command.

Note: The total length of the parameter string passed to OPS must not exceed 100 characters.

COLOr *color* | (*color ...*)

The initial colors for text. You can specify up to four colors, as you can with the COLOR command. You need parentheses only if you specify more than one color.

DARK | NORMAL

Run in DARK or NORMAL display mode.

Edit *edit-filetype*

The filetype for the edit file. The full name of the file is:

fn edit-filetype fm

Helpmode Short | Full

Set HELPMODE to SHORT or FULL.

MARGIn *x* | OFF

Specify a non-default margin display. *x* is any valid OPS x-coordinate, such as 45m.

MSG

Show the initial message ("Use PF1 for help"), even if in DARK mode.

OPS

Treat the infile as an OPS presentation, even when filetype is not ADMOPS.

SS *ss-name*

The initial symbol set to load. This is also the default symbol set (the one specified by RESET), unless you modify it with the DEFAULTS command.

TRC

Assume TRC codes in column 2.

ADMOPSLx EXEC return codes under CMS

The return code from ADMOPSLx EXEC may be one of the following values:

RC	Explanation
RC = 0	No errors detected by EXEC
RC = 8	Error in the EXEC; the module was not loaded
RC = 12	Erroneous argument to ADMOPSLx EXEC
RC = 20	Graphics cannot be shown
RC = 28	File not found

Appendix B. Runtime customization

At runtime, OPS lets you customize aliases for GDDM page-printer device tokens. To do so, you create a file called ADM7OPC DATA under CMS, or set up ADM7OPC as a member of the system data set (systemprefix.SADM DAT) under TSO.

ADM7OPC contains page-printer device token records (lines beginning with P) and comments (lines beginning with *). Each device token record has the following format:

```
P alias device_token filetype
```

where:

alias The alias name you want to use.

device_token

A "true" GDDM device token. See *GDDM Base Application Programming Reference* for a list of available tokens.

filetype

The last qualifier (TSO) or filetype (CMS) of the generated page segments using this alias.

If you create a customization file, the OPS default values are ignored. Here's what they are:

```
P A4      A4      PSEG3820
P 240X    IMG240X PSEG3820
P 240     IMG240  PSEG3820
P LEGAL   LEGAL   PSEG3820
P FINE240 FINE240  PSEG3820
P 300A4   IMG300A4  PSEG4028
P 300Q    IMG300Q  PSEG4028
P 300L    IMG300L  PSEG4028
P 600X    IMG600X  PSEG4250
P FINE600 FINE600  PSEG4250
```

Note: You can specify a maximum of 10 aliases.

Appendix C. OPS sample presentation (ADM7OPXA ADMOPS)

```

1 )ops *****
)ops *           Sample presentation for *
)ops *           GDDM Online Presentation System (OPS) *
)ops * 5684-168, 5695-167 (C) Copyright IBM Corporation 1988, 1996 *
)ops *****
)ops * NOTE: Edit MACRO ADMPAGE locates the page you saw *
)ops *           Edit MACRO ADMGET gets any commands PUT. *
)ops *****
)ops set copyr = '5684-168,5695-167 (C) Copyright IBM Corp 1988,1996'
)ops *****
)ops * Throw away the parts you don't need. *
)ops * Each page (foil sample) is self contained, except for: *
)ops * - DARK is needed to get DARK mode (no visible frame, etc.) *
)ops * - ASPECT is needed to get proper aspect ratio when your *
)ops * presentation is supposed to run on different terminal types.*
)ops *****
)ops * Remember 4 simple rules: *
)ops * --- Column 1 is reserved for page control: *
)ops * - "1" means "new foil". *
)ops * - "+" means "overwrite the last text line". *
)ops * --- OPS command lines must begin with a specific string *
)ops * (termed command prefix). Default is ")ops", but you can *
)ops * change it with the PREFIX command. *
)ops * --- Use ";" to separate commands on a single line. *
)ops * --- "*" means "comment". *
)ops *****
)ops *
)ops * Let's first change the Command Prefix:
)ops prefix % ; * It's easier to write "%" than ")ops"
% dark ; * To run without visible command field.
% * * Remove the command to get a visible command field,
% * * or use the CMDFLD command.
% aspect 100 67; * Set aspect ratio of graphics field to suit most
% * * terminals. Needed only if you intend to run your
% * * presentation on different types of terminals.
% frame; * Add a yellow frame on this page.
%*** Some important commands are:
% reset; * Reset indentation, strange attributes, etc.
% ss admuksf; * Select symbol set named "admuksf".
% color red; * Set text color. "col r" would be enough.
% h 7%; * Set text height to 7% of screen - many other units.
% space 0.6t; * Insert space = 0.6 times the text size.
% ce; * Begin text centering.
% shear 15; * Begin text shearing 15 degrees.
G D D M
% sh 0; * Stop text shearing.
% color purple; * Only some devices knows this color
% h .9t; * 90% of current height
Online Presentation System (OPS)
% ce off; * Stop text centering
% ss admuudrp; * Select symbol set named "admuudrp".
% col y; * Change to yellow.
% h 3.5

```

OPS sample presentation

```
% lsp 0.2t;      * Leading SSpace should be 0.2 times the text size.
%               * (normally 0).
% in 5;         * Indentation 5 out of 100 units in x-direction.
```

On the following pages are shown various samples of foils, etc.

```
% space 0.5t;   * Insert space = 0.5 times the text size.
Use them as skeletons.
```

```
% ss admuusrp; lsp 0; h 3.3; col t
% colattr ¢ red > yel; * Set one or more color attribute characters -
%                   * precede any word by color-attrib to change color
% in 15
```

Use:

```
% arrow 5 &y+.5t 12, w .7, draw tur ;* Arrow at y-level = &y+.5t, means
%                               * bottom of text + .5 text height
+   ¢PF1 for Help
    ¢PF8 to scroll forward
    ¢PF12 to Edit.
```

Or type a command on the bottom line.

```
% arrow 5 &y+.5t 12, w .7, draw tur ;* Same comment
```

Type a number to locate a page.

```
% text © ri 99 0.5 size 1.1 2 col w ss vss ;* Write copyright
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A typical text foil. The symbol set ADMUUSRP is good for plotting
%* Text flow is not used in this sample. It not really necessary...
%*****
% reset;                * The recommended way to start a new page
% ss admuusrp; h 4; col r; * These 3 commands define the text
%*** First a classical foil header
% head 'A text foil sample--OPS' draw r 2
%*
%*** Then the body
% ss admuudrp; h 4.5; col b
% space 1.5t; in 5
Some advice regarding symbol sets:
% ss admuusrp; h 3.5; col yel; lsp 0.3t
% colattr ¢ w _ keep
```

```
* Don't use too many different sets
*¢GDDM standard sets begin with¢"ADM"
* This is¢ADMUUSRP. Good for plotting.
*¢ADMUUDRP is used in the blue header
* Filled symbol sets take a long time to plot
* Take care with¢ADMUARP. Very demanding.
```

```
%*
%*** And finally the foot (using color-attributes):
% foot 'Feb. 1996-¢Company_Use_Only' dr r 2, ss admuusrp h 4 col r
% *****
```

```
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
% *** This foil uses text flow functions: TFxxx ***
% *****
% reset
% ss admuudrp; h 4.5; color red
% head 'Which kind of infiles?' draw off ; * No underlining...
```

```

% box 49 32 98 60 2 -45 dr red 2; *... rather a box as a fancy frame
% tf xmax 95; in 5
% col t; colattr ¢ r ! g
% tfh Files OPS!can read
% ss admuusrp; h 3; col w; lsp 0.1t
% tfp Any flat file with a maximum record length of 255 characters.
Typical files are: Formatted listings for 3800 and 6670,
program listings, and OPS presentations.
% ss admuudrp; h 4.5; col p
% tfh Files OPS¢cannot read
% ss admuusrp; h 3; col w; lsp 0.1t
% tfp Files wider than 255 characters, for instance listings for page
printers such as IBM 4250 and 3820.
% col t; h 2.3

% tfp (This foil was produced using OPS text flow functions. The
frame is drawn by BOX).
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A more complicated text foil using text flow functions and TRC
%*****
% reset; * The recommended way to start a new page
% trc on; * TRC means: column 2 controls the fonts
%*** Define 3 fonts for TRC use:
% ss admuudrp admuucip admuusrp
% h 3.5 3 2.3
% col tur yel whi
% lsp 0 0 .1t
%*** now 0 in column points to 1st font, 1 points to 2nd font, etc.
%*** Do NOT add further (global) ss, h, color, or lsp commands
%*** Local commands, like color on the head-command, are OK:
0% head 'Text flow functions' dr r 2 color r
0% head '→OPS' ss admuksf, col t dr off; * Even local SS is OK
%*
%*** Start text flow.
% tf xmax 96; in 4
0% tfh Here are the OPS text flow commands:
1% tful
1% tfli TF - Text flow start
2% tfp May be used to set Xmax and paragraph spacing (psp keyword)
1% tfli TFH and TFP - Header and Paragraph
2% tfp For all the bulk text. TFH spaces 1.5 paragraph spaces -
2thats the only difference.
1% tfli TFUL, TFOL, TFEUL, TFEOL, TFLI - for lists
2% tfp This foil uses an unordered list. You can nest and mix
2unordered and ordered lists to any depth
1% tfli TFXMP, TFXEMP
2% tfp Use these around
1examples
2or text with
1tabs
% tfeul
% trc off
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A text foil using tabs to put text into columns
%*****
%reset

```

OPS sample presentation

```
%**** First a header with two symbol sets:
% ss admuutrp; h 4.5
% head 'A foil with tabs' col r dr r 2
% head '→OPS' ss admuuksf, col t dr off
%*
%**** The tabs definition. "-" is the default tab character
% tabs 5 20 60 r, 65; * "r" means "Right" adjust at 60.
% h 3.5

→Product summary division A
% ss admuudrp; col y; h 3.5

→Item→Description→Price→Notes
% line 5 &y-3mm 90 dr y 2
% ss admuusrp; col w; h 3.5; lsp 0.3t; colattr c pink ! t 0

-1234→Flower pot-12.34
-3456→Roses      -1.23
-5678→Clean water→23.45→!Stiff price...
-6789→Clean air→3.45
% ss admuutrp; col r; h 4; * Overwrite in new font
+→→→Special offer
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A foil combining text with an ICU chart
%*****
%reset
%****
% ss admuutrp; h 4.5; col r
% head 'Imbedding an ICU chart' dr off
% line 0 &y-2mm 100, w 2mm, dr red ;* a fat line
%*
%**** Then the chart
%**** The last 2 numbers are width and height. Try your way.
%chart adm7smp3 50 45% 82 56
%*
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* Now to a more fancy foil
%*****
%*** A nice background in bluish green
% reset
% backgr 3031; * 031= RGB-mixture = 0/3 Red, 3/3 Green, 1/3 Blue.
% msg "Don't press any key now..." col r; * Fingers off ...
% force; * force it to the screen and go on immediately
% ss admuuksf; h 8
%*** Text with shadows - simply write it twice:
% text 'It's easy with OPS...' 50% 86% center, color neutral
% text 'It's easy with OPS...' 50.7% 87% center, color yellow
% wait 1; * Wait 1 second
%*** A chart in a black box
% box 36 45% 70 50 fill n, draw w 2; * a black hole with frame
% force; * force it to the screen and go on immediately
% chart adm7smp3 36 45% 70 ; * 70% of full screen width
% wait 2; * wait 2 seconds
% gdf ADM7TERM 16 50% 50mm; * Add a terminal symbol within the chart
% force
% chart adm7smp3 16 49% 25mm
```

```

% wait 3; * wait 3 seconds
%*** OPS's logo - directly from Hollywood
% box 75 30% 48 34 fi n dr w; * a black hole
% force
% gdf ADM7LOGO 75 30% 48
% msg
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* Some text experiments
%*****
% reset; ss admuutrp
% text 'Some text experiments' 50 61 ce, ss admuksf col y h 5.5
% box 50 30.5 99 59 r 1cm width 2mm dr t ;* frame with thickness
%h 7%; col y
%start 8 75%
A word of many colors:
%colattr c r 0, ! g 0, / b 0 ; * colattr chars with 0-width
%text ' 0!P/S' ss admuksf h 9% ; * note absence of coordinates
%start 50 60%; * Sets the starting point for next text
%col y
We can start
where we want
%wait 1
%start 8 60%; col w
% lsp 0.5t; * leading space = 0.5 * current text height
Written with
leading space
LSP 0.5t

%force
%lsp 0; col t
Written with
leading space
LSP 0
%force; h 5%
%start 70 35%
%center 70
Centering
about
any
line
%* The TEXT command does it all (if you have space enough)
% text 'OPS' 50 20 cc, h 10 ss admuksf col r turn 90 box 1.2 fi b, r 6mm
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* Examples on simple figures
%* Use the numbers in black squares to identify the code section below
%*****
% reset; ss admuksf; h 5.5; col y
% head 'OPS Line Art' dr y
%* Save the text box attributes in variable "txtbox" for ease of use
% set txtbox = str(cc ss admuutrp, h 3, col w, box 4 fi n)
%*****
%*** Boxes - marked "1" ****
%*****
% draw white; * draw lines in white
% fill red; * Fill areas with red
% box 15 45 20 20

```

OPS sample presentation

```
% fill 3311; * 311 = RGB-mixture: 3/3 Red, 1/3 Green, 1/3 Blue
% box 25 46 16 16 turn 45
% box 35 49 12 12 fill 3322, shear 45
% text '1' 06 50 &txtbox
% force
%*****
%*** Box - marked "2" ****
%*****
% box 20 30 20 13 5, r 7mm, w 3mm, draw w, wfill 3230, fill 3310
% text '2' 20 30 &txtbox
% force
%*****
%*** Line - marked "3" ****
%*****
% line 5 88% 95 88% 95 10% w 3mm r 2cm wf 3220 dr w
% text '3' 90 55 &txtbox
% force
%*****
%*** Polygon - marked "4" ****
%*****
% poly 50 40 70 55 90 35 fill 3120, dr y, width 2mm, r 1cm ;* Wow
% text '4' 70 45 &txtbox
% force
%*****
%*** Disk - marked "5" ****
%*****
% draw w; * Draw white, line width 1
% dasd 65 30% 20 20 fi 3012; * a DASD
% dasdf 65 39% 20 3 fi tur v1 50 v2 150 ; * A DASD file
% dasdx 65 35% 20; * a file separator in the DASD
% dasdx 65 28% 20
% text '5' 76 21 &txtbox
% force
%*****
%*** Arrow - marked "6" ****
%*****
% arrow 22 21,22 5,65 5,65 20,w 4mm h 3 t 2 curve wf 3130 dr y
% text '6' 43 07 &txtbox
% force
%***
%*****
1) ops prefix %;* Assure prefix is known if the page is "cut out" ****
%* A fancy sample using the AREA command
%*****
% aspect 100 67; reset
% ti ! 15 ;* New line = x'15'
% text 'Use the!AREA command:' 2 60 ss admuusrp h 3
% text 'for INTERNAL!area filling' 2 49 ss admuusrp h 3
% offset 15 15
%*** A capital 0 ****
% area fill red 12
% arc 25 40 20 v2 180
% line 15 40 15 30
% arc 25 30 20 v1 180 v2 360
% line 35 30 35 40
% arc 25 40 10 v2 180
% line 20 40 20 30
% arc 25 30 10 v1 180 v2 360
```



```

% line 30 30 30 40
% area end
%*** A capital P *****
% area fill green 14
% line 48 30 45 30 45 20 40 20 40 50 48 50
% arc 48 40 20 v1 90 v2 -90
% line 48 35 45 35 45 45 48 45
% arc 48 40 10 v1 90 v2 -90
% area end
%*** A capital S *****
% area fill blue 12
% arc 70 41.5 18 v2 270
% arc 70 28.5 8 v1 90 v2 -180
% line 66 28.5 61
% arc 70 28.5 18 v1 -180 v2 90
% arc 70 41.5 8 v1 -90 v2 -360
% line 74 41.5 79
% area end
% offset 15 -18
%*box co 14 19 66 32 dr off fi w 5
% gdf ADM7LOGO 46 35 68 32 scale 1.6
% area fill n
% box co 14 19 66 32 dr off
%*** A capital O *****
% arc 25 40 20 v2 180
% line 15 40 15 30
% arc 25 30 20 v1 180 v2 360
% line 35 30 35 40
% arc 25 40 10 v2 180
% line 20 40 20 30
% arc 25 30 10 v1 180 v2 360
% line 30 30 30 40
%*** A capital P *****
% line 48 30 45 30 45 20 40 20 40 50 48 50
% arc 48 40 20 v1 90 v2 -90
% line 48 35 45 35 45 45 48 45
% arc 48 40 10 v1 90 v2 -90
%*** A capital S *****
% arc 70 41.5 18 v2 270
% arc 70 28.5 8 v1 90 v2 -180
% line 66 28.5 61
% arc 70 28.5 18 v1 -180 v2 90
% arc 70 41.5 8 v1 -90 v2 -360
% line 74 41.5 79
% area end
% offset 0
% text 'for MASKING' 3 17 ss admuusrp h 3
%*****
1 )ops prefix %;* Assure prefix is known if the page is "cut out" *****
%* A final frame sample
%* Suggestion: Use WFill OFF on the ARROW when plotting (takes time)
%*****
% reset; * Never forget it...
% msg 'No more samples. PF: 1=? 7=Up 12=Edit' col red
% ss admuksf
% arrow 34 4, 1 4, 1 60, 99 60, 99 4, 66 4, head 3, d, w 1, r 5, draw r
% box 50 4 30 6 dr r 2, r 3mm
% text 'The Frame of Presentations' 50 61 ce, h 6, col t

```

OPS sample presentation

```
% force; mix on
% rays 36 33 37 dv 5 dr r 1
% rays 50 33 37 dv 5 dr g 1
% rays 64 33 37 dv 5 dr b 1
% mix off; ss admuuksf; h 7.5; col n; sh 15; lsp -.5
% start 26.5 35.5
% text 'Online' 27.5 35.5
% text 'Online' 27 36 col w
% text 'Presentation' 27.5 28.5
% text 'Presentation' 27 29 col w
% text 'System' 27.5 21.5
% text 'System' 27 22 col w
% force; col r; sh 0; h 5.5
% text 'Color' 50 4, cc, box 30 6 dr r 2, r 3mm, fi n
% wait 1
% text 'Graphics' 50 4, cc, box 30 6 dr r 2, r 3mm, fi n
% wait 1
% text 'Dynamics' 50 4, cc, box 30 6 dr r 2, r 3mm, fi n
% wait 1
% text 'Online' 50 4, cc, box 30 6 dr r 2, r 3mm, fi n
% wait 1
% box 50 4 32 8 dr n 2, r 3mm, fi n
% force
% arrow 30 4 34 4 head 3, w 1, r 5, draw n
% arrow 70 4 66 4 head 3, w 1, r 5, draw n
% line 30 4 70 4 w 1 draw r
%* The end *****
```

Appendix D. Messages

Messages provide information or report errors.

OPS displays messages in response to direct commands in the message field, using the current field attributes as set by default or by the MSGFLD command.

OPS displays messages in response to infile commands on a special panel showing the erroneous command as well as the original input line. If a parameter error occurs before the main panel has been displayed, the message is written directly to the screen in line mode.

Most information messages are suppressed in DARK mode; some are also suppressed if the cause is an infile command.

Messages are classified as:

- I** These are for information. They tell you about status changes, such as loading a symbol set.
- W** These indicate that something might be wrong, such as loading a GDF, and overwriting a symbol set in use.
- E** These are issued in response to syntax errors, missing external files, and so on. The requested action cannot be accomplished. In most cases the original status is preserved.
- S** These are errors that either prevent OPS from continuing, or that occur before the primary page is created. In the latter case, execution is discontinued, because the reason for the error might be outside the user's control (such as an error in a local OPS startup procedure or menu function) and the user would not otherwise be aware of this fact.

With the exception of severe error messages, messages are by default displayed without classification or number. You can display both with the command:

MSG LONG

You can also query the last message, using the QUERY command:

QUERY MSG

which displays the last system message in LONG form.

Message listing

Because OPS always displays messages from infile commands within the context of the error that has generated them, the meaning of most such messages, and the action you need to take, is clear from the context in which the messages appear.

If an error occurs in response to a direct command, press PF1 to display the command explanation in the help file.

For most OPS messages, therefore, it's clear, from the context and the help provided, what the message means, and what you should do, if anything. The partial list of OPS messages here provides further information only where the meaning of the message might not be clear from the context in which it is issued.

In the message text, the following four symbolic names are used:

&COMMAND	The command verb causing the error
&KEYWORD	The keyword or operand causing the error
&STRINGV	A character string causing the error, or the name of a keyword getting an incorrect value
&INT	An integer value related to the specific error; for example, a limit value of an operand which has been exceeded

000I No message issued

Explanation: This message is issued in response to QUERY MSG before any system message has been issued.

001S Unknown parameter: &KEYWORD

Explanation: The OPS invocation includes the displayed parameter, which is unknown to OPS.

User Response: If you invoked OPS by the original ADMOPSLx EXEC in CMS, this error should not occur. In any other case, consult your local support if you don't understand what caused the error. It could be an error in the local OPS interface.

002S Syntax error for parameter: &KEYWORD

Explanation: The displayed parameter has improper syntax, such as a missing right parenthesis.

User Response: Refer to message 001.

003S Length of the value for &KEYWORD exceeds maximum: &INT

Explanation: Each parameter value has a maximum length determined by its type. The actual maximum value is displayed.

User Response: Check the length of the parameter value. For example, a symbol set name should have a maximum of 8 characters. Otherwise, refer to message 001.

004S Error occurred during parameter evaluation

Explanation: This message accompanies another error message, and tells you that the other error occurred when some OPS parameter was being processed, such as a symbol set being loaded.

User Response: If you entered the parameter, correct it. Otherwise, refer to message 001.

005S Graphics cannot be shown

Explanation: Either your terminal or your current connection cannot display host graphics.

008S Syntax error &INT in customization file. Erroneous line:

Explanation: The displayed line from the OPS customization file is erroneous. The integer reason codes mean:

1. The leading character must be: * or P.
2. There are more than 10 APA printer records.
3. A name is missing or is longer than eight characters on a page printer device token record.

012E PF-key not defined

Explanation: You are using a PF key without a defined value.

User Response: On the main panel, issue Q PF to see which PF keys are currently defined. On other panels this information is displayed either at the top or at the bottom of the screen.

013E Unknown command: &COMMAND

Explanation: The displayed command is not an OPS command.

014E &COMMAND not allowed instream

Explanation: The displayed command is not supported as an infile command. It may only be used as a direct command.

User Response: "Where to find information on commands" on page 43 lists all commands, and indicates which are valid as infile commands.

015E &COMMAND not allowed direct

Explanation: The displayed command is not supported as a direct command. It may only be used as an infile command.

User Response: "Where to find information on commands" on page 43 lists all commands, and indicates which are valid as direct commands.

016E Syntax is-->

Explanation: The message is issued in response to invalid or incomplete command syntax.

017E Invalid operand: &KEYWORD

Explanation: The displayed operand is not valid on the command or is in an invalid position.

018E Too many operands

Explanation: You have supplied more operands to the command than it expects. (This condition is not checked by all commands; some deliberately ignore superfluous operands.)

019E Required library not allocated. Command prohibited: &COMMAND

Explanation: This occurs under TSO only.

A request was issued to display or save a graphics file, or to display a file list. The request failed because the required library (*ddname*) was not allocated, or because the libraries allocated to the required *ddname* have different and (for ISPF/PDF services) inconsistent

attributes. For example, they may have inconsistent record formats.

If the request was for a file list, the ISPF/PDF error message is logged in the user's ISPF log.

Note: Creating page segments with the PSEG command requires an output library allocated for GDFs.

020E Name missing

Explanation: The command requires the name of a file.

021E Name too long: &KEYWORD

Explanation: Names (of files or GDDM nicknames) must be no longer than eight characters long.

022E Invalid character "&STRINGV" in name

Explanation: The cursor points at a character that is not valid in the name of a file.

030E Coordinate missing. (At least &INT required.)

Explanation: The command requires the displayed number of coordinates as a minimum. For example, a LINE requires at least three coordinate values.

031E Coordinate was expected at: "KEYWORD"

Explanation: In the indicated position, a coordinate value is the only thing allowed.

035E Invalid unit: "&STRINGV"

Explanation: The displayed coordinate unit is not supported.

User Response: "Using the coordinate system" on page 60 tells you what units OPS supports. This error may also occur when a numeric operand and an alphabetic keyword get concatenated in the input field because of a missing blank.

036E Invalid sign: "&STRINGV"

Explanation: The displayed operand must not be signed.

040E Missing value for "&STRINGV"

Explanation: The displayed operand is of type "keyword with value." The value has not been supplied.

041E Invalid numeric value: &STRINGV

042E Value for "&STRINGV" must be integer: &KEYWORD

043E Value for "&STRINGV" must be positive: &KEYWORD

044E Value for "&STRINGV" must not be positive: &KEYWORD

045E Value for "&STRINGV" must be negative: &KEYWORD

046E Value for "&STRINGV" must not be negative: &KEYWORD

047E Value for "&KEYWORD" less than minimum: &INT

048E Value for "&KEYWORD" greater than maximum: &INT

Explanation: The errors 041 - 048 each display a keyword and an invalid value supplied for that keyword.

050E "?" not supported as operand on this command

Explanation: Syntax help requires a "?" in front of the command word. "?" after the command word is supported only for some commands (yielding overviews of valid operands). See the help file, where "?" is shown in the syntax wherever supported.

052E "&COMMAND" must be selected among |@!%/(=+-<>|

Explanation: The commands SEPCH, TABCH, and COLATTR allow as operands only the special characters shown.

100E PFK number not in range 1-12

Explanation: OPS supports definition of PF keys only in the range 1 to 12.

101E PFK function exceeds 75 characters

Explanation: The function assigned to a PF key is limited to 75 characters.

102W PFK text truncated to 79 chars. Use "Q PF" for full text

Explanation: Unless you specify the TEXT or NOText keyword, OPS composes a text string of the active PF functions. Each key function is truncated to eight characters, and the total string is truncated to 79 characters.

User Response: Compose your own string via the TEXT keyword, or just use "Q PF" when you want to see the PF key functions.

110E HELP not installed. Inform your local support.

Explanation: No help file is allocated.

User Response: If you started OPS via a supported startup facility like the ADMOPSLx EXEC, inform your local support of the error.

111E Help file truncated. Repeat your query.

Explanation: The help file must not be bigger than 1500 lines.

User Response: Inform the person responsible for the help file, if it is not your own file.

112I Press PF1 again for more information

Explanation: This message is issued in response to PF1 + a command word in the command field. In the command field you should have a syntax sample. If you press PF1 again (keeping the original command verb unchanged), you'll enter the help file at the command explanation.

113W No command details found. Try clean PF1

Explanation: You are asking for syntax help on a command verb not found in the help file in position 4.

User Response: If the help file is a locally written file, this may be an error in the file. Otherwise, use PF1 on an empty command field to get into the help file, where you may subsequently use FIND to search for commands and topics.

**114E Invalid argument to &STRINGV:
&KEYWORD**

Explanation: You have supplied an invalid argument to one of the help file commands: Up, Down, Top, Bottom, Locate, Find or RFind. Type "?" and press ENTER to see the valid arguments.

116E Section numbers must be in the range 0 to 99

Explanation: Typing a number and pressing ENTER will take you to the numbered section in the help file. The number must be in the range 0 to 99.

118E Not a proper word (topic): &KEYWORD

Explanation: You did not point at a word of alphanumeric characters when you pressed ENTER or PF1.

130E Unable to create filelist

Explanation: The filelist is created on your A-disk using the CMS LISTFILE command. This message indicates that the list could not be written, probably because your A-disk is nearly full. Normally you will get an error message from CMS when this occurs.

134I Description: &STRINGV

Explanation: The displayed ADMGDF file contains the displayed textual description.

140E Unsupported Query

Explanation: You are querying a variable or facility not supported by the QUERY command. The help file contains an overview of supported queries.

142E Severe ISPF error

Explanation: If the error occurred in response to a file list request, the ISPF/PDF error message is logged in the user's ISPF log. A probable reason is inconsistent library attributes, such as inconsistent record formats.

150E Error in RESET: &STRINGV

Explanation: RESET executes SS, COLOR, and SIZE commands for the default symbol set (unless explicitly requested not to). One of these commands has caused the displayed error.

User Response: Use the DEFAULTS command to specify a correct default symbol set.

152E WAIT arg not a positive integer

Explanation: WAIT supports pause only for an integer number of seconds.

181E Specify target line/page/label

Explanation: On the LOCATE command, you must specify an integer as target line or page number.

182E Invalid scroll unit: &KEYWORD

Explanation: Valid scroll units are only Page and Line.

185E Invalid argument to REFRESH

Explanation: REFRESH must be followed only by a command of the status setting type, such as COLOR.

186E Invalid Page number

Explanation: The PAGESET command requires an integer page number.

**202E Too many symbol sets loaded.
RELEASE some...**

Explanation: The internal symbol set table of OPS supports a maximum of 40 symbol sets at a time. Of these 40 positions, three are occupied by HW, VSS, and ISS, leaving 37 positions for further symbol sets.

User Response: To load further symbol sets, you must release some of the 37 other symbol sets, using the RELEASE command. Note that you may also use RELEASE as a file command, and that no error message will be issued even when you attempt to release symbol sets that are not loaded. You can put RELEASE commands in the infile (RELEASE 65 66 67 68 ..., and so on) when you need room for many symbol sets.

205E Symbol set not loaded: &STRINGV

Explanation: You have tried to RELEASE of a symbol set which was not loaded.

Note: This message is issued only when the RELEASE command is direct.

217E &STRINGV value outside screen

Explanation: The START command does not allow X > XMAX (100) or Y > YMAX.

**232E TABS not allowed when text flow active.
Use TFXMP.**

Explanation: Use of tab stops and text flow is not supported at the same time. It is legal, however, to have TABS defined (remain defined) during text flow. This error message is issued only when the TABCH character (usually “~”) is encountered in the text during text flow, and TABS are defined.

244E Invalid length: &KEYWORD

Explanation: Length of color attribute characters may be specified as “0” or “1” only. “1” is default.

250E Invalid MARGIN value: &KEYWORD

Explanation: The MARGIN value must be a valid coordinate expression.

251E Shearing (close to) infinite

Explanation: The SHEAR angle is (or is close to) 90 degrees, which yields infinite shearing.

280E No active &STRINGV ordered list

Explanation: You have issued a TFEUL or TFEOL command, but no list is active.

281E No List is active

Explanation: You have issued a TFLI command, but no list is active.

**282E BW must be positive or a string:
&KEYWORD**

Explanation: The bullet width may be specified as a positive coordinate value or as a string only.

283E Text paragraph underflows the screen

Explanation: Although a page may occupy several screens, OPS does not support text flow of a single paragraph over screen boundaries.

User Response: Split the paragraph into several paragraphs, or decrease the text size or leading space to keep the text within a single screen.

**301E Help-GDF: &STRINGV not available.
Inform local support**

Explanation: The graphics help you requested was not found by OPS.

User Response: Inform your local OPS support, stating what help you requested (such as DRAW attributes).

304E Invalid palette pattern: &KEYWORD

Explanation: Palette patterns must be composed as “3RGB” where R, G, and B take integer values in the range 0-3.

305E Invalid color: &KEYWORD

Explanation: Color must be a number in the range 0-8 or any abbreviation, after the first letter, of Blue, Red, Pink, Green, Turquoise, Yellow, White, or Neutral, or a palette pattern “3RGB” where R, G and B take values in the range 0-3.

**310E Draw/Fill are all OFF. Set either to some
color.**

Explanation: You have requested DRAW OFF / 8 / Neutral, as well as no FILL.

313E V1 to V2 spans more than &INT degrees

Explanation: The difference between V2 and V1 is more than the displayed number of degrees, which is the maximum allowed by the command.

320E AREA already open

Explanation: You can have only one open AREA at a time.

321E FILL is OFF. Specify FILL color/pattern for the AREA

Explanation: AREA is used to fill a group of objects seen as a single object. AREA does not make sense without FILL enabled.

324E Open AREA has been closed (invalid function within AREA)

Explanation: Valid functions within an AREA are line art functions only (TEXT, for example, is not allowed). It is also not allowed to have an AREA open when the picture is displayed, for example at WAIT, FORCE, or end-of-page. OPS has closed the open AREA.

350E FORM name MUST begin with alphabetic: &KEYWORD

Explanation: ICU has been called with an ADMCFORM name beginning with a non-alphabetic character.

351E Coordinates or size less than 0

Explanation: ICU charts cannot be translated outside the screen boundary, or mirrored. Accordingly, center position must be within the screen and sizes must be positive.

352W Area has been reduced

Explanation: The size you have specified for a chart would extend outside the screen. This is not possible, so OPS has reduced the size to make the chart fit the screen.

360E Zero or negative size invalid

Explanation: GDF size must be positive. To mirror a GDF, use negative SCALE value(s).

361E SCALE/TURN/SHEAR are not allowed with NOTRANS

Explanation: You have requested “no transformation,” which is in conflict with the request for SCALE, TURN, and/or SHEAR.

362E GDF contains non-transformable segment. Specify “NOTrans”

Explanation: OPS has detected a non-transformable segment in the GDF. Your GDF command, however, requires some transformation that cannot be satisfied (for example, translation through a screen boundary).

User Response: Either modify your size requirement in order to make the GDF fit within the screen, or add “NOTrans” to the GDF command, which will cause OPS to reduce the size for you.

363W Symbol set conflict. Refresh may help...

Explanation: A GDF has overwritten a symbol set already in use. This may or may not cause trouble in your picture.

User Response: If the picture shows an erroneous symbol set, try REFRESH, which will make OPS reload any overwritten symbol set and rewrite all text. However, if two GDFs are overwriting the same symbol set, nothing helps.

365E &STRINGV ADMGDF already exists. Specify REPLACE option.

Explanation: This message is issued only in response to an infile SAVE command.

390E Device number not in range 1-9 or 61-69: &KEYWORD

Explanation: A secondary terminal must be addressed by a number as shown.

391E Device &KEYWORD not available

Explanation: Either you have not defined the secondary terminal from your own userid, or the secondary terminal has not been dialed to yours.

400E Proper syntax (first time): PRINT filename

Explanation: You must specify a name of a print file (printer ID) the first time you issue the PRINT command in a session.

401E Invalid width of print: &KEYWORD

Explanation: Print width must be a positive integer.

406I No current print file open

Explanation: Issued in response to “Q PRINT” before any PRINT command has been issued.

410I Plotter not available

Explanation: OPS cannot detect any plotter.

User Response: If a plotter is locally connected to your terminal, make sure the power is switched on (also on the graphics extension to your terminal). If the problem doesn’t resolve, ask your local expert for help.

420E Units allowed are “mm,” “cm,” and “i” (or none)

Explanation: Page segment size must be specified without unit, or with “mm,” “cm,” or “i” only.

421E **Width must be integer or have unit:
“mm,” “cm” or “i”**

Explanation: Default unit is “mm.” Maximum precision is 1mm (fractional mm values are not supported).

422E **Page segment size greater than device
token allows**

Explanation: Each device token is associated with a maximum paper size. The size you are requesting exceeds the paper size.

User Response: Either reduce the size required, or select another device token.

500E **No dataset defined for edit**

Explanation: OPS has been started without any dataset defined for the EDIT function.

User Response: This should not occur when OPS is started using ADMOPSLX EXEC. If OPS is started via a customized EXEC, this will be the cause. It may be an error. If in doubt, call your local support.

502E **Error while re-allocating infile**

Explanation: An error occurred when trying to reallocate the infile after EDIT. A system message (VM/MVS) may be displayed before this message appears.

User Response: If you don't understand immediately why this has happened, call your local support for help.

510W **Already undone**

Explanation: You tried to UNDO an object which OPS has marked as undone.

511E **Cannot be undone**

Explanation: The command string following UNDO is not associated with an object that can be undone (such as a CHART command), or the command is not found in the OPS command list (last 100 kept).

512I **Nothing to undo**

Explanation: No more commands in the command list (last 100 commands) are associated with an object that can be undone.

513I **TEST active**

Explanation: The last command executed was a TEST command. If the next command you execute is also a TEST command, the last generated object is deleted (allowing you to TEST a command).

514E **Nothing picked**

Explanation: You did not pick any pickable object from the screen (you did not get a detectable primitive within the cursor aperture). Remember that ICU charts are not pickable.

515E **Command creating the object not found
among last 100.**

Explanation: The object you picked is not associated with a command in the OPS command list (last 100 commands).

User Response: You may clean up the command list by issuing REFRESH. Unless the current screen uses more than 100 commands, they will all be in the command list and the problem should be solved.

520I **Command field added to PUT file**

Explanation: The contents of the command field has been added to the PUT file. The command has been prefixed with the current file command prefix, for example)OPS.

999E **No description found for error/message
&INT**

Explanation: Internal error in OPS. which should not occur.

User Response: Inform your local support of this problem.

Glossary

This glossary defines technical terms used in GDDM documentation that apply to OPS, or used only in OPS. 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

advanced function printing. The ability of licensed programs to use the all-points-addressable concept to print text and illustrations on a printer.

alphanumerics. Pertaining to alphanumeric fields.

APA. All points addressable.

API. Application programming interface.

application programming interface (API). The formally defined programming interface used by an application programmer to pass commands to, and get responses from, an IBM system control program or licensed program.

area. A shaded shape, such as a solid rectangle. It is created by opening the area, defining its outline, and closing the area.

aspect ratio. The width-to-height ratio of an area, symbol, or shape.

attributes. Characteristics or properties that can be controlled, usually to obtain a required appearance; for example, the color of a line.

axis. In a chart, a line that is drawn to indicate units of measurement against which items in the chart can be viewed.

A3. A paper size, more common in Europe than in the U.S. It measures 297mm by 420mm, and is twice the size of A4. See also *A4*.

A4. A paper size, more common in Europe than in the U.S. It measures 210mm by 297mm, and is half the size of A3. Compare with *quarto*. See also *A3*.

B

background color. Black on a display, white on a printer. The initial color of the display medium. Contrast with *neutral color*.

base GDDM color. One of 19 colors, available for text and graphics, and selectable either by number, or by a valid abbreviation of the name of the color in English.

bi-level image. An image in which each pixel is either black or white (value 0 or 1). Contrast with *gray-scale image* and *halftone image*.

C

CECP. Country-extended code page.

CGM. Computer Graphics Metafile. A file that contains information about the content of a picture, and conforms to the International Standard, ISO 8632, or is of a similar format.

character. A letter, digit, or other symbol.

character box. 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 can be program-controlled.

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' through X'FE', or its EBCDIC character equivalent.
- For the Vector Symbol Editor, a hexadecimal constant in the range X'00' through 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 graphics screen. The size of the grid determines the basic size of the characters in all text constructed by presentation graphics routines. It is the fundamental measurement in chart layout, governing the spacing of

glossary

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 interspacing. The space inserted between consecutive character boxes in a text string.

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*.

CMS. Conversational Monitor System. A time-sharing subsystem that runs under VM.

code page. Defines the relationship between a set of code points and graphic characters. This relationship covers both the standard alphanumeric characters and the national language variations. GDDM supports a set of code pages used with typographic fonts for the 4250 page printer.

color attribute character. In OPS, a special character which you can associate with a color, and then use to highlight characters or words in that color.

command prefix. In OPS, the character or character string that precedes command words, identifying the command as an *infile command*. The default command prefix is)OPS.

command separator character. In OPS, the character used to separate multiple commands on one line of input. The default command separator character is a semicolon (;).

command word. In OPS, the first element of a command, which defines the main function being invoked. For example, ARROW, BOX, and WAIT.

coordinate system. In OPS, the means of addressing individual points on the screen. The coordinate system uses an x-coordinate to specify the horizontal position, and a y-coordinate to specify the vertical position. See also *percentage coordinate system*.

country-extended code page (CECP). An extension of a normal EBCDIC code page that includes definitions of all code points in the range X'41' through X'FE'. Each code page contains the same 190 characters, but

the mapping between code points and graphics characters depends on the country for which the code page is defined. This is a method of marking a GDDM object so that the environment in which it was created can be identified. It enables automatic translation to a different environment.

current point. In OPS, the end of the previously-written text line. Unless the current point is explicitly changed, this is also where the next text line will be written.

cursor. A physical indicator that can be moved around a display screen.

D

dark mode. In OPS, a way of displaying presentations that minimizes the presence of OPS itself, allowing users to concentrate on the content of the presentation. Contrast with *normal mode*.

DASD. Direct access storage device.

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. The value of an attribute chosen by GDDM when no value is explicitly specified by the user. For example, the default line type is a solid line. The default value is sometimes device-dependent.

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).

direct command. In OPS, a command executed from the screen's command field rather than from the OPS source file (*infile*). Contrast with *infile command*.

display device. Any output unit that gives a visual representation of data, such as a screen or printer. More commonly, the term is used to mean a screen and not a printer.

display terminal. An input/output unit by which a user communicates with a data processing system or subsystem. It usually includes a keyboard and always provides a visual presentation of data. For example, a 3179 display.

dot matrix. In computer graphics, a two-dimensional pattern of dots used for constructing a display image. This type of matrix can be used to represent characters

by dots. Synonymous with *character matrix* and *display-point matrix*.

double-byte characters. Characters belonging to the *double-byte character set (DBCS)*.

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; for example, *Kanji* or *Hangeul*.

drawing default. The value of a graphics attribute chosen by GDDM when no value is explicitly specified by the user. The drawing default may be altered by the user.

E

EBCDIC. Extended binary coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

edit. To enter, modify, or delete data.

editing grid. In the GDDM Image and Vector Symbol Editors, a grid used as a guide for editing a symbol. In the Image Symbol Editor, it is a dot matrix. In the Vector Symbol Editor, it is a grid of lines.

enterprise. An organization or company that undertakes local, national, or international business ventures.

F

field. An area on the screen or the printed or plotted page. See *graphics field*.

fillable area. In OPS, an area on the graphics screen limited by closed lines, such as the area inside a circle or box. Fillable areas can be filled with colors and patterns.

fix point. In OPS, the point around which a piece of line art, such as a line or curve, is drawn. By default, the fix point is the geometric center of the object.

flat file. A file that contains only data; that is, a file that is not part of a hierarchical data structure. A flat file can contain fixed-length or variable-length records.

foil. A transparency for overhead projection.

font. A particular style of typeface (for example, Gothic English). In GDDM, a font can exist as a programmed symbol set.

formatted document. A type of file containing text, images, and graphics.

four-button cursor. A hand-held device, with cross-hair sight, used on the surface of a *tablet* to indicate position on a screen. Synonymous with *puck*.

G

GDDM. Graphical Data Display Manager. A series of IBM licensed programs, running in a host computer, that manage communications between application programs and display devices, printers, plotters, and scanners for graphics applications.

GDDM-OS/2. A licensed program that enables IBM Personal System/2 and other personal-computer systems with OS/2 installed to run GDDM application programs in the host computer.

GDDM-PGF. GDDM-Presentation Graphics Facility. A member of the GDDM family of licensed programs. It is concerned with business graphics, rather than general graphics.

GDF. Graphics data format.

general color. In OPS, the color set (by the COLOR command) for all line art for which no other color is defined.

GIF. Graphics Interchange Format.

GL. Graphical Language.

Graphical Data Display Manager. See *GDDM*.

graphics. A picture defined in terms of *graphics primitives* and *graphics attributes*.

graphics area. Part of a mapped field that is reserved for later insertion of graphics.

graphics attributes. The color selection, color mix, line type, line width, graphics text attributes, marker symbol, and shading pattern definitions.

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 field. A rectangular area of a screen or printer page, used for graphics.

glossary

graphics text attributes. The symbol (character) set to be used, character-box size, character angle, character mode, character shear angle, and character direction.

gray-level. A digitally encoded shade of gray, normally (and always in GDDM) in the range 0 through 255. See also *gray-scale image*.

gray-scale image. An image in which the gradations between black and white are represented by discrete gray-levels. Contrast with *bi-level image* and *halftone image*.

H

halftone image. A bi-level image in which intermediate shades of gray are simulated by patterns of adjacent black and white pixels. Contrast with *gray-scale image*.

Hangeul. A character set of symbols used in Korean ideographic alphabets.

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*.

hexadecimal. Pertaining to a numbering system with base sixteen.

host. See *host computer*.

host computer. The primary or controlling computer in a multiple-computer installation.

I

ICU. Interactive Chart Utility.

image. Synonym for *digital image*.

image field. A rectangular area of a screen or printer page, used for image. Contrast with *alphanumeric field* and *graphics field*.

image symbol. A character or symbol defined as a dot pattern.

Image Symbol Editor (ISE). A GDDM-supplied interactive editor that enables users to 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)*.

infile. In OPS, the file containing OPS commands used as the source of the OPS presentation.

infile command. In OPS, a command issued from within the OPS *infile*. Each infile command is prefixed by the *command prefix*. Contrast with *direct command*.

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 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 can also be conversational, implying a continuous dialog between the user and the system.

ISE. Image Symbol Editor.

ISO. International Organization for Standardization.

ISPF. Interactive System Productivity Facility.

ISS. Image symbol set.

J

joystick. A lever that can pivot in all directions in a horizontal plane, used as a *locator* device.

K

Kanji. A character set of symbols used in Japanese ideographic alphabets.

Katakana. A character set of symbols used in one of the two common Japanese phonetic alphabets; Katakana is used primarily to write foreign words phonetically. See also *Kanji*.

key. In a legend, a symbol and an associated data-group name. A key might, for example, indicate that the blue 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 one of the various data groups.

L

landscape. The arrangement of text and/or graphics on a page or screen so that its width is greater than its length. Contrast with *portrait*.

Latin. Of or pertaining to the Western alphabet. A synonym for *single-byte character set*.

leading space. In OPS, the space inserted between vertically consecutive character boxes, and controlled by the LSP command.

legend. A set of symbolic keys used to identify the data groups in a business chart.

line attributes. Color, line type, and line width.

line spacing. In OPS, the sum of leading space and character box height.

logical input device. A concept that allows application programs to be written in a device-independent manner. The logical input devices to which the program refers may be subsequently associated with different physical parts of a terminal, depending on which device is used at run-time.

M

marker. A symbol centered on a point. Line graphs and polar charts can use markers to indicate the plotted points.

menu. A displayed list of logically grouped functions from which the user can make a selection. Sometimes called a menu panel.

menu-driven. Describes a program that is driven by user response to one or more displayed menus.

mixed character string. A string containing a mixture of *Latin* (one-byte) and *Kanji* or *Hangeul* (two-byte) characters.

mouse. A device that a user moves on a flat surface to position a pointer on a screen.

MVS. IBM Multiple Virtual Storage. A system under which GDDM can be used.

MVS/XA. Multiple Virtual Storage/Extended Architecture. A subsystem under which GDDM can be used.

N

National Language Support (NLS). A special feature that provides translations of the ICU panels and some of the GDDM messages into a variety of languages, including U.S. English.

neutral color. White on a display, black on a printer. Contrast with *background color*.

nickname. In GDDM, a means of referring to a device, the characteristics and identity of which have been already defined.

NLS. National Language Support.

normal mode. In OPS, a way of working in which OPS features, such as the command and message field, are visible. This is the mode in which presentations are usually developed. Contrast with *dark mode*.

null character. An empty character represented by X'00' in the EBCDIC code. Such a character does not occupy a screen position. Contrast with *blank character*.

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.

page printer. A printer, such as the 3820 or 4250, to which the host computer sends data in the form of a succession of formatted pages. Such devices can print pictorial data and text, and can position all output to pixel accuracy. The pixel density and the general print quality both often suffice as camera-ready copy for publications. Also known as *composed-page printer*.

page segment. An object that can contain text and images, and that can be included on any addressable point on a page or electronic overlay.

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*.

pel. Picture element. See *pixel*.

percentage coordinate system. In OPS, a form of the coordinate system in which the x- and y-coordinates are both expressed as percentages of the screen, rather

glossary

than as points on the screen. See also *coordinate system*.

PGF. Presentation Graphics Facility.

pick. The action of the operator in selecting part of a graphics display by placing the graphics cursor over it.

picture interchange format (PIF) file. In graphics systems, the type of file, containing picture data, that can be transferred between GDDM and an IBM 3270-PC/G, /GX, or /AT workstation.

PIF. Picture interchange format.

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.

pointings. Pairs of x-y coordinates produced by an operator defining positions on a screen with a locator device, such as a *mouse*.

portrait. The arrangement of text and/or graphics on a page or screen so that its length is greater than its width. Contrast with *landscape*.

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.

preview chart. A small version of the current chart that can 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.

print utility. A subsystem-dependent utility that sends print files from various origins to a queued printer.

processing option. Describes how a device's I/O is to be processed. It is a device-family-dependent and subsystem-dependent option that is specified when the device is opened (initialized). An example is the choice between CMS attention-handling protocols.

profile. A file that contains information about how GDDM is to process requests for services to devices or other functions.

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.

PSEG. See *page segment*.

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.

PS/2. Personal System/2.

puck. Synonym for *four-button cursor*.

put file. In OPS, the intermediate file in which OPS commands are collected before being saved in the infile.

Q

quarto. A paper size, more common in the U.S. than in Europe. It measures 8.5 inches by 11.0 inches. Also known as A size. Compare with A4.

R

raster device. A device with a display area consisting of dots. Contrast with *vector device*.

rastering. The transforming of graphics primitives into a dot pattern for line-by-line sequential use. In GDDM PS devices, this is done by transforming the primitives into a series of programmed symbols (PS).

resolution. In graphics and image processing, the number of pixels per unit of measure (inch or meter).

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.

REXX. Restructured Extended Executor Language. One of the programming languages supported by GDDM.

Roman. Relating to the *Latin* type style, with upright characters.

S

SBCS. Single-byte character set.

scanner. A device that produces a digital image from a document.

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.

shear. The action of tilting graphics text so that each character leans to the left or right while retaining a horizontal baseline.

single-byte character set (SBCS). A set of characters in which each character occupies one byte position in internal storage and in display buffers. Used for example, in most non-Oriental symbols. Contrast with *double-byte character set*.

source image. An image that is the data input to image processing or transfer.

stand-alone (mode). Operation that is independent of another device, program, or system.

standard default. The value of a graphics attribute chosen by GDDM when no value is explicitly specified by the user. The standard default cannot be altered by the user, although it may be over-ridden by the user.

stylus. A pen-like pointer used on the surface of a tablet to indicate position on a screen.

symbol. Synonymous with *character*. For example, the following terms all have the same meaning: vector symbols, vector characters, vector text.

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 that the user has 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 use of system spooling facilities.

T

table reference code. In OPS, a numeric character (0, 1, 2, or 3) corresponding to the order in which attributes such as symbol sets and colors have been specified. If used, the character appears in the second column of the infile.

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.

target position. In the GDDM Vector Symbol Editor, the grid coordinates of a point on the editing grid to which a vector is to be drawn.

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*.

test symbol. In the GDDM Image and Vector Symbol Editors, an area on the Symbol Edit panel in which the currently chosen symbol is displayed.

text. Characters or symbols sent to the device. GDDM provides alphanumeric text and graphics text.

text attributes. See *graphics text attributes*.

transform. (1) The action of modifying a picture for display; for example, by scaling, rotating, or displacing. (2) The object that performs or defines such a modification; also referred to as a *transformation*. (3) In GDDM image processing, a definition of three aspects of the data manipulation to be done by a projection:

1. A transform element or sequence of transform elements
2. A resolution conversion or scaling algorithm
3. A location within the target image for the result.

Only the third item is mandatory.

transformable. A segment must be defined as transformable if it will subsequently be moved, scaled, or rotated.

transparency. A document on transparent material suitable for overhead projection.

TRC. See *table reference code*.

glossary

TSO. Time sharing option. A subsystem of MVS under which OPS can be used.

U

User Control. A GDDM function that enables the terminal or workstation to perform some functions without the need for application programming. The actions include: moving and zooming graphics; manipulating windows; printing, plotting, and saving pictures.

V

vector. (1) In computer graphics, a directed line segment. (2) In the GDDM-PGF Vector Symbol Editor, a straight line between two points.

vector device. A device capable of displaying lines and curves directly. Contrast with *raster device*.

vector symbol. A character or shape composed 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.

VM/ESA. IBM Virtual Machine Enterprise Systems Architecture.

VM/SP CMS. IBM Virtual Machine/System Product Conversational Monitor System. A system under which OPS can be used.

VM/XA. IBM Virtual Machine Extended Architecture. A system under which OPS can be used.

VSS. Vector symbol set.

W

window. In GDDM, the term window has three distinct meanings:

1. The “graphics window” is the coordinate space used for defining the primitives that make up a graphics picture. By default, both x and y coordinates run from 0 through 100. The graphics window can be regarded as a set of coordinates that are overlaid on the viewport.
2. An “operator window” is an independent rectangular subdivision of the screen. Several can exist at the same time, and each can receive output from, and send input to, either a separate GDDM program or a separate function of a single GDDM program.
3. The “page window” defines which part of a page which is deeper or wider than its partition should currently be displayed.

workstation. A display screen together with attachments such as a local copy device or a tablet.

world coordinates. The user application-oriented coordinates used for drawing graphics. See also *window*.

Index

Special Characters

; (command separator) 37

? command 196

\$

See dollar (\$)

* command 40

&

See ampersand (&)

Numerics

3-D effect on boxes 120

A

A4 keyword on ASPECT command 59

accuracy of cursor reading 25

adding an ICU chart 135

adding objects 24

ADM7OPTA (tutorial) 195

ADM7OPXA sample file

listing of 211

using 22

ADM7SCAN 171

ADMCDATA files

CHART command, used with 135

description of 135

ICU command, used with 172

libraries for, under TSO 6

TSO library allocation 201

ADMCFORM files

CHART command, used with 135

description of 135

ICU command, used with 172

libraries for, under TSO 6

TSO library allocation 201

ADMCOLSx pattern

definition of 103

FILL command 105

ADMDEFS (profile) file 185

ADMGET, edit macro 31

ADMIMAGE files 186

ADMIMG files 146

ADMOPS

data set assumed under TSO 6

default for presentations 19

filetype assumed under CMS 5

ADMOPSLx EXEC

CMS options 205

customizing (TSO) 204

return codes (CMS) 207

selecting the language with 5

ADMOPSLx EXEC (*continued*)

starting OPS (CMS) 5

starting OPS (TSO) 6

TSO startup options 201

ADMPAGE, edit macro 23

ADMPRINT files 183

ADMSYMBL files

GDDM symbol sets 65

libraries for, under TSO 6

symbol editors 173

TSO library allocation 201

advanced text handling 73—101

ALARM command 154

alignment

boxes 119

corner, on CHART command 136

text 81, 87

allocating

libraries under TSO 201

output data sets under TSO 6

alphanumeric fields, controlling 55

ampersand (&) 163

animation

creating sequences 153

overview 151

ARC command 124

SET keyword 124

area

attributes 105

color 105

pattern 105

AREA command 128

arithmetic expressions

basic rules 39

rules for coordinates with units 63

ARROW command

DOUBLE keyword 117

drawing arrows 115

heads 116

HSCALE keyword 117

rounded corners 115

tails 116

ASA control characters 20

ASIS keyword on IMAGE command 148

ASPECT command

A4 keyword 59

controlling aspect ratio 58

default setting 58

landscape 59

portrait 59

use with coordinate system 61

index

- assigning
 - Enter key to commands 41
 - PF keys to commands 41
- attributes
 - box, controlling 120
 - text, setting 21
 - unspecific 105
 - unspecific (FILL command) 108
- AUTO command
 - dynamic sequences 151
 - GDFSAVE subcommand 145
 - printing, used in 184
 - PSEG command, used with 189
 - scrolling, used in 181
- automatic scrolling 181
 - GDFSAVE command, used with 145
- B**
- BACKGR command 133
- background, filling 133
- basic GDDM colors
 - definition of 69
 - DRAW command 104
 - FILL command 106
- basic text handling 65—73
- BIND command 178
- BLINK keyword on field commands 56
- BOTTOM command 177
- BOX command
 - 3-D effect 120
 - controlling alignment 119
 - coordinate prompting 25
 - drawing boxes 119
 - rounded corners 120
 - shearing 122
 - system variables set by 168
 - TEXT command, subcommand on 92, 122
 - turning 122
 - width 120
- BREAK command 93
- BROWSE command 171
- browsing files 171
- BULLET keyword on TFUL command 98
- C**
- carriage control 20
- CC—object center alignment
 - BOX command 119
 - center text on line (TEXT command) 87
 - IMAGE command 148
 - LINE command 114
- CE—line center alignment
 - BOX command 119
 - center text on box (TEXT command) 87
- CENTER command 81
- center of line, changing 114
- centering text 81, 87
- changing
 - command prefix 39
 - infile 31
- character box
 - controlling 67
 - interspacing 74
 - line spacing 76
 - variation in size 68
- character interspacing 74
 - DEFAULTS command with 75
 - TRC codes with 74
- CHART command
 - adding an ICU chart 135
 - coordinate prompting 25
 - corner alignment 136
 - file list 136
 - sample 137
- charts (ICU)
 - adding to current picture 135
 - displaying 135
 - displaying on clear screen 137
- CHARTX command
 - dialed terminals 138
 - displaying a chart on a clear screen 137
- Chinese startup EXEC (ADMOPSLC) 5
- choosing a sample presentation 8
- CIRCLE command 123
- CLEAR command 192
 - clearing graphics field 59
 - clearing PF key definitions 41
- clearing graphics field 59
- CLISTs, creating presentations with 34
- CMD keyword on CMDFLD command 56
- CMDFLD command
 - controlling the command field 56
 - prompting for 56
- CMS
 - ADMOPSLx EXEC 205
 - command 174
 - executing commands 174
 - OPS startup options 205
 - page segment output 186
 - sample presentations under 9
- CMYK color model 108
- CO—corner alignment
 - BOX command 119
 - CHART command 136
 - IMAGE command 148
- COLATTR command
 - coloring words and characters 84
 - KEEP keyword 85
- color
 - areas (pattern) 105

- color (*continued*)
 - background 133
 - basic GDDM 69, 105
 - CMYK model 108
 - command field 56
 - fading 109
 - field commands, values on 55
 - frame 134
 - general (OPS) 105
 - with FILL command 108
 - graphics overviews 70, 198
 - grid 24
 - images 149
 - individual words 84
 - lines 104
 - margin indicators 134
 - message field 56
 - mixing 109
 - of line art 105
 - of text 69
 - on plotters 70
 - PFK field 57
 - recoloring of GDF 143
 - recoloring of PSEs 188
 - RGB model 108
 - values on field commands 55
 - wide lines 110
- color attribute characters 84
 - language considerations xiii
- COLOR command
 - IMAGE command, subcommand on 149
 - set text color 69
 - setting defaults for 73
 - subcommand of field commands 55
 - TEXT command, subcommand on 90
- column 1, carriage control in 20
- columns, writing text in 82, 83
- command
 - adding to presentations 30
 - assigning Enter key to 41
 - assigning PF keys to 41
 - CMS, executing 174
 - comments within 40
 - direct, definition of 37
 - external 171
 - help on 43
 - infile prefix 39
 - infile, definition of 37
 - issuing from dialed terminal 192
 - length 37
 - list of 44
 - operands 38
 - recalling 42
 - reference information for 43
 - retrieving 28
 - rules for infile 40
- command (*continued*)
 - separator character (;) 37
 - START, retrieving 28
 - structure 37
 - testing 27
 - TSO, executing 174
 - words 37
- command field
 - controlling 56
 - dark mode 54
 - highlighting keywords 56
 - initializing with a command 54
 - normal mode 54
 - reading, in flow control 161
- comment 40
 - samples of usage 56
- compact list 98
- concatenating symbolic variables 166
- context sensitive unit 62
- controlling OPS display 53–64
- controlling the message field 56
- coordinate
 - evaluating 63, 75
 - evaluation 78
 - offsetting 64
 - prompting for 25
 - reading with cursor 26
 - system variables containing 168
 - system, definition of 60
 - valid units 62
- coordinate system
 - adjusting 64
 - ASPECT command 61
 - context sensitive unit 62
 - definition of 60
 - displaying 24
 - GRID command 24
 - STR function with 63
 - units in 62
- COPY command 29
- creating graphics commands 27
 - summary (figure) 30
- CSP command
 - DEFAULTS command with 75
 - set character interspacing 74
- currency characters, language considerations xiii
- current point
 - definition of 79
 - moving 111
 - TEXT command 86
- cursor
 - pointing 28, 198
 - reading coordinates 26
 - set accuracy of 25
- CURVE command
 - LINE command, subcommand on 113

index

- CURVE command (*continued*)
 - POLYGON command, subcommand on 114
- curves, drawing 113
- customization
 - GDDM device token 209
- D**
- Danish startup EXEC (ADMOPSLD) 5
- DARK command
 - default colors set by 55
 - selecting dark mode 53
 - using dark mode 54
- dark mode
 - fields in 54
 - selecting 53
 - using 54
- DASD command
 - DRAW subcommand 127
 - drawing DASD symbol 126
 - FILL subcommand 127
 - shearing 127
 - turning 127
- DASDF command
 - DRAW subcommand 127
 - drawing DASD file symbols 126
 - FILL subcommand 127
 - shearing 127
 - turning 127
- DASDX command
 - DRAW subcommand 127
 - drawing DASD file separator symbols 126
 - shearing 127
 - turning 127
- data sets, allocating output 6
- defaults
 - aspect ratio 58
 - setting text 21
 - text, setting 73
- DEFAULTS command
 - character interspacing 75
 - LSP command 77
 - rules for using 21
 - setting text defaults 73
 - TRCs with 73
- DELETE command 30
- deleting an object 26
- DESCR keyword on GDFSAVE command 144
- device token 186
- dialed terminal (CMS)
 - copy screen to 193
 - establishing 191
 - issuing commands 192
 - overview 191
 - primary nodeid 191
 - SCOPY command 193
- dialed terminal (CMS) (*continued*)
 - secondary, defining 191
 - using
 - CHARTX command 138
 - GDFX command 144
- direct command, definition 37
- displacing objects 131
- display control 53—64
- display modes 53
- dollar (\$) 163
- DOUBLE keyword on ARROW command 117
- DOWN command 177
- DRAFT command 27
- DRAW command
 - BOX command, subcommand on 120
 - DASD commands, subcommand on 127
 - default colors 104
 - LINE command, subcommand on 111
 - POINT command, subcommand on 110
 - setting line attributes 104
 - unspecific attributes 105
- drawing
 - arcs 124
 - arrows 115
 - boxes 119
 - circles 123
 - curves 113
 - frames 134
 - lines 111
 - pie slices 125
 - points 110
 - polygons 114
 - rays 125
 - sectors 125
 - shadows behind fillable objects 130
 - wide lines 110
- dynamic sequence
 - AUTO command 151
 - forcing graphics field to screen 152
 - messages in 153
 - overview 151
 - pauses, inserting 151
 - samples 154
 - sounding the alarm 154
 - WAIT command 151
- E**
- EDIT command
 - external command 171
 - invoking text editor 23
- edit macros 23, 31
- editing GDFs 145
- editing the infile 171
- ellipse (line art) 123

ELSE command 157
 END command 31
 English startup EXEC (ADMOPSLA) 5
 ENTER command 41
 ERASE command 107
 erasing areas before filling 107
 erasing fillable objects 130
 error messages 219—225
 displaying 42
 displaying numbers of 42
 evaluating coordinates 63
 EXEC command 171
 EXECs, creating presentations with 34
 exporting from OPS
 ADMPRINT 183
 GDFs 144
 PSEGs 186
 expressions, arithmetic 39
 external commands
 BROWSE 171
 CMS 174
 EDIT 171
 EXEC 171
 ICU 172
 ISE 173
 ISPF 175
 overview 171
 symbol editors, using 173
 TSO 174
 VSE 173

F

fading colors 109
 file
 browsing 171
 GDF 138
 file lists
 CHART command 136
 displaying 197
 GDFs 141
 GDFX command 144
 ICU files 172
 infiles 31
 SS command 173
 symbol set 65
 FILL command
 basic GDDM colors 106
 BOX command, subcommand on 120
 CMYK color model 108
 DASD commands, subcommand on 127
 ERASE subcommand on 107
 general colors with 108
 help panel for 106
 RGB color model 108
 setting area attributes 105

FILL command (*continued*)
 unspecific attributes 108
 fillable area
 definition of 103
 drawing shadows beyond 130
 erasing 130
 setting 105
 filling of grouped objects 128
 filling the background 133
 FIND command 180
 flow control
 command field, reading 161
 commands 157
 getting tag values 160
 labels in 158
 overview 157
 tagging objects 159
 FOOT command
 creating footers 93
 multiple 94
 footer (on foil) 93
 FORCE command 152
 forcing a line break 93
 format, page segment 188
 FRAME command 134
 French startup EXEC (ADMOPSLF) 5
 FULL keyword
 GDF command 140
 IMAGE command 148

G

GDDM
 basic colors 69
 device token 186
 FILL command basic colors 106
 ISS, default image symbol set 66
 patterns available 106
 symbol sets 65
 user control mode 60
 VSS, default vector symbol set 66
 GDDM Internet home page xiii
 GDF command
 coordinate prompting 25
 displaying GDFs 140
 FULL keyword 140
 merging graphics segments 143
 recoloring GDFs 143
 shearing 142
 specifying size of GDF 141
 turning 142
 GDF files
 displaying 140
 displaying on clear screen 144
 editing 145
 libraries for, under TSO 6

index

- GDF files (*continued*)
 - merging graphics segments 143
 - overview 138
 - recoloring 143
 - saving pictures as 144
 - shearing 142
 - specifying size of 141
 - symbol sets with 139
 - transforming 139
 - TSO library allocation 201
 - turning 142
 - GDFEDIT command 145
 - GDFSAVE command 144
 - GDFX command
 - displaying GDFs on clear screen 144
 - file list 144
 - general phrases in symbolic variables 165
 - GETC command 26
 - getting to know OPS 7—11
 - GL files 186
 - global, nameless translation table 71
 - glossary 227
 - GOTO command 158
 - GRAIN command 25
 - Graphics Data Format
 - See GDF files
 - graphics field
 - aspect ratio 58
 - clearing 59
 - dark mode 54
 - forcing to screen 152
 - in normal mode 54
 - graphics overview 70, 198
 - gray shades in page segments 188
 - GRID command 24
 - grouping objects (AREA) 128
- ## H
- HEAD command
 - creating headers 93
 - multiple 94
 - HEAD keyword on ARROW command 116
 - header (on foil) 93
 - heads, controlling arrow 116
 - HEIGHT command
 - controlling character box height 67
 - setting defaults for 73
 - TEXT command, subcommand on 90
 - TRC codes with 68
 - height, character box 67
 - HELP command 196
 - help facilities
 - file lists 197
 - graphics overviews 70, 198
 - help file 195
 - help facilities (*continued*)
 - query commands 199
 - syntax help 196
 - tutorial 195
 - help file 195
 - FILL command 106
 - HELMODE command 197
 - HIGH keyword on PFK command 57
 - highlighting
 - command field 56
 - individual words 84
 - message field 56
 - PFK field 57
 - home page for GDDM xiii
 - how to use this book xiii
 - HSCALE keyword on ARROW command 117
- ## I
- ICU command 172
 - IF command 157
 - IMAGE command
 - ASIS keyword 148
 - color of images 149
 - COLOR subcommand 149
 - coordinate prompting 25
 - displaying images 147
 - FULL keyword 148
 - IMGLIB command 147
 - SCALE keyword 149
 - SELECT keyword 149
 - Image Symbol Editor 173
 - image symbol set, default GDDM 66
 - images
 - ADMIMG files 146
 - basic display 147
 - color of 149
 - displaying 146
 - IMAGE command 147
 - IMGLIB command 147
 - page segments, scaling 188
 - picture elements 147
 - resolution ratio 147
 - scaling 148
 - IMGLIB command 147
 - IN command
 - indenting text 80
 - scrolling with 81
 - indentation 80
 - infile
 - ADMGET macro 31
 - changing 31
 - command prefix, changing 21
 - command, definition of 37
 - copying put file to 31
 - for presentation sample 10

- infile (*continued*)
 - INFILE command 31
 - PREFIX command 21
 - prefix, changing 39
 - rules for commands 40
 - searching 180
 - text in 76
 - INFILE command 31
 - input
 - ADMSYMBL files 65
 - GDF files 138
 - ICU charts 135
 - images 146
 - inserting space between text 79
 - Interactive Chart Utility
 - displaying charts 135
 - ICU command 172
 - Internet home page for GDDM xiii
 - interspacing, character 74
 - ISE command 173
 - ISPEXEC command 168
 - ISPF
 - command 175
 - sharing variables with 168
 - Italian startup EXEC (ADMOPSLI) 5
 - italicizing text 78
- J**
- Japanese Kanji startup EXEC (ADMOPSLK) 5
- K**
- Kanji (Japanese) startup EXEC (ADMOPSLK) 5
 - KEEP keyword on COLATTR command 85
 - keywords 38
- L**
- labels
 - using directly 179
 - using from the infile 158
 - languages, national xiii
 - selecting startup EXEC for 5
 - LASTCMD command 42
 - leading space 76
 - leaving OPS 31
 - left scrolling 81, 178
 - library allocation under TSO 201
 - line art 103–134
 - samples of 103
 - line attributes
 - definition of 103
 - LINE command, setting with 111
 - setting 104
 - line break, forcing 93
 - line color
 - defaults 104
 - definition of 103
 - setting 104
 - LINE command
 - changing center of line 114
 - creating rounded corners 112
 - CURVE subcommand 113
 - DRAW subcommand 111
 - drawing curves 113
 - drawing lines 111
 - filling wide lines 112
 - moving current point 111
 - RADIUS keyword 112
 - REFP keyword 114
 - SET keyword 111
 - setting line attributes 111
 - setting line width 112
 - shearing 113
 - turning 113
 - WFILL subcommand 112
 - WIDTH keyword 112
 - line spacing (text) 76
 - line style
 - definition of 103
 - setting 104
 - line width
 - definition of 103
 - LINE command, setting with 112
 - setting 104
 - list item 97
 - list of commands 44
 - LOCATE command
 - flow control, used in 158
 - scrolling, used in 177
 - LOW keyword on PFK command 57
 - LSP command
 - controlling line spacing 76
 - DEFAULTS command used with 77
- M**
- macros, edit 23, 31
 - margin
 - left margin for text 82
 - with text formatting 95
 - MARK command 101
 - MERGE keyword on GDF command 143
 - merging graphics segments 143
 - message field
 - controlling 56
 - dark mode 54
 - highlighting keywords 56
 - in normal mode 54

index

messages
 displaying numbers of 42
 error messages 42, 219—225
 field in normal mode 54
 user messages 153

MIX command 109

mixing colors 109

MOD keyword on TI command 71

mode, selecting display 53

MODIFY command
 modify option setting 199
 symbolic variables 169

MOVE command 29

MOVIE command 153

moving the current point 111

MSG command
 displaying messages 42
 dynamic sequences 153

MSGFLD command
 controlling the message field 56
 dynamic sequences 153

N

named translation table 71

nameless global translation table 71

national languages xiii
 selecting startup EXEC for 5

nested lists 97

nicknames (GDDM)
 GL files, used for 186
 plotting, used in 185
 PostScript output, used for 185

NOPROMPT keyword on CMDFLD command 56

NORMAL command
 default colors set by 55
 keyword on field commands 56
 selecting normal mode 53
 using normal mode 53

normal mode
 command field in 54
 graphics field in 54
 message field in 54
 selecting 53
 using 53

Norwegian startup EXEC (ADMOPSLN) 5

NOTEXT keyword on PFK command 57

numbers
 definition 38
 symbolic variable interpretation of 164

O

object picking (TEST) 28

objects
 copying 29

objects (*continued*)
 deleting 26, 30
 displacing 131
 exercise 32
 modifying (TEST) 28
 moving 29
 OFFSET command 131
 tagging 159

OFFSET command
 adjusting coordinate system 64
 local 131
 of line-art 131
 primary 131
 TEXT command, subcommand on 89

online presentations, advantages of 3

operands, OPS command 38

OPS
 CMS startup 205
 display control 53—64
 END command for leaving 31
 GDFs, using with 138
 getting to know 7
 leaving 31
 overview 3
 requirements for 4
 sample presentations for 7
 startup procedures 201
 TSO startup 201
 tutorial 195

ordered list 97

output data sets, allocating 6

output options
 ADMPRINT files 183
 GDF files 138
 matrix printer 183
 page segment creation 186
 plotter 185

overview, graphics 70

P

page
 first in a presentation 22
 presentation 20
 refreshing 179
 synchronizing page count 178

page printer 186

page segment
 AUTO command 189
 coloring 188
 creating 186
 displaying 146
 formats 188
 IMGLIB command 147
 previewing 188
 PSEG command 186

- page segment (*continued*)
 - scaling images 188
- PAGESET command 178
- panels, printing 183
- paragraph 96
- paragraph spacing 95
 - in lists 98
- pattern
 - defining 106
 - definition of 103
 - setting 105
- PDF editor
 - ADMGET used with 31
 - EDIT command 171
 - TSO, editor under 4
- pel 147
- period (.) 179
- PF keys
 - assigning to commands 41
 - default setup 22
 - highlighting keywords 56
- PFK command
 - assigning PF keys to commands 41
 - controlling PF keys 57
 - HIGH keyword 57
 - LOW keyword 57
 - NOTEXT keyword 57
 - TEXT subcommand 57
- picking an object (TEST) 28
- picture
 - adding ICU charts to 135
 - elements 147
 - ICU charts 135
 - images, displaying 146
 - saving as GDFs 144
 - types of 135
- pie slices, drawing 125
- PLOT command 185
 - color translation 70
- plotting 183
 - GL files 186
 - overview 183
 - PLOT command 185
 - presentations 185
 - sample presentation 11
- POINT command 110
 - DRAW subcommand 110
- points, drawing 110
- POLYGON command
 - CURVE subcommand 114
 - drawing polygons 114
 - rounded corners 114
 - shearing 115
 - turning 115
- positional operands 38
- positioning text exactly 82
- PostScript 185
- PREFIX command
 - changing the default prefix 39
 - guidance on using 40
 - rules for using 21
- presentation
 - adding commands to 30
 - advantages of online 3
 - basic principles 20
 - creating 22
 - using EXEC or CLIST 34
 - editing 23, 171
 - first page 22
 - how to create 35
 - new page 21
 - pages 20
 - private 19
 - public 19
 - sample 193
 - CHART command 137
 - choosing 8
 - CMS 9
 - dynamic sequences 154
 - listing of 211
 - listing of source 10
 - printing/plotting 11
 - saving 11
 - TSO 8
 - working with 8
 - starting 22
 - types of 19
- previewing page segment output 188
- primary nodeid (dialed terminals) 191
- primary terminal 191
- PRINT command 183
- printing
 - overview 183
 - page segments 186
 - panels 183
 - sample presentation 11
- private presentations 19
- procedures, startup 201
- programmed symbols (PS) 133
- PROMPT keyword on CMDFLD command 56
- prompting 56
- prompting for coordinates 25
- proportional spacing 67
- PS overflow 133
- PSEG
 - See page segment
- PSEG command 186
- PSEGLIB command 186, 187
- PSP keyword on TF command 96
- public presentations 19

index

PUT command 30

put file

- ADMGET macro 31
- copying to infile 31
- creating 30

Q

QUERY command

- Enter key definition 41
- message numbers 42
- PF key settings 41
- query status 199
- symbol sets 72
- symbolic variables 169

quotation marks 86

quoted strings in symbolic variables 165

R

RADIUS keyword

- BOX command 120
- LINE command 112

ratio, aspect 58

RAYS command 125

READCMD command 161

reading coordinates via cursor 26

READTAG command 160

recalling commands 42

recoloring of GDF 143

REFP keyword on LINE command 114

REFRESH command 179

RELEASE command 174

- GDFs, used with 139
- releasing symbol sets 72

removing objects 26

requirements for OPS 4

RESET command

- rules for using 21
- setting defaults for 73

resolution ratio of images 147

retrieving commands 28

REVERSE keyword on field commands 56

RFIND command 180

RGB color model 108

right alignment

- LINE command 114
- right justify text on box (TEXT command) 87

right scrolling 81, 178

rounded corners

- arrows 115
- boxes 120
- lines 112
- polygons 114

S

SADMDAT, filetype 203

sample

- CHART command 137
- dynamic sequences 154
- line art 103

sample file (ADM7OPXA)

- listing of 211
- using 23

SCALE keyword on IMAGE command 149

scaling

- arrow heads and tails 117
- GDFs 142
- images 148
- PSEGs, images in 188

SCOPY command 193

screen

- reading coordinates from 26
- simplified update (DRAFT) 27

scrolling

- commands for 177
- GDFSAVE command, automatic with 145
- labels, using 179
- left 81
- principles 177
- refreshing the page 179
- right 81
- searching the infile 180
- sideways 178
- synchronizing page number 178

secondary terminal 191

SECTOR command 125

SEGMENT keyword on GDFSAVE command 144

SELECT keyword on IMAGE command 149

selecting display mode 53

separating commands 37

SEPCH command 37

SET command 163

SET keyword

- ARC command 124
- LINE command 111

setting line attributes 104

SHADOW option 130

SHEAR command

- BOX command, subcommand on 122
- italicizing text 78
- LINE command, subcommand on 113
- POLYGON command, subcommand on 115
- TEXT command, subcommand on 91

shearing

- boxes 122
- circles 123
- dasd symbols 127
- GDFs 142
- lines 113

- shearing (*continued*)
 - polygons 115
 - text 78, 91
 - simplified Chinese startup EXEC (ADMOPSLC) 5
 - simplified screen update 27
 - size
 - character box 67
 - variation in character box 68
 - SIZE command
 - controlling character box size 67
 - setting defaults for 73
 - TEXT command, subcommand on 90
 - TRC codes with 68
 - source listing of presentation sample 10
 - SPACE command 79
 - Spanish startup EXEC (ADMOPSLS) 5
 - SS command
 - selecting symbol sets 65
 - setting defaults for 73
 - TEXT command, subcommand on 90
 - START command
 - columns, writing 82
 - positioning text exactly 82
 - retrieving 28
 - starting OPS
 - language selection 5
 - under CMS 5, 205
 - under TSO 6, 201
 - status values 167
 - STR function 165
 - sample of use 63
 - string, specifying text 86
 - subcommands 38
 - suspending processing 151
 - Swedish startup EXEC (ADMOPSLV) 5
 - switching PF key display 57
 - symbol sets
 - character box
 - controlling 67
 - line spacing 76
 - conflict 140
 - default 66
 - editing 173
 - GDFs 139
 - initial symbol set (CMS) 206
 - initial symbol set (TSO) 204
 - querying 72, 139
 - releasing 72, 139
 - selecting 65
 - with TRC codes 66
 - types of 65
 - symbolic variables
 - concatenation 166
 - defining 163
 - general phrases 165
 - interpretation 164
 - symbolic variables (*continued*)
 - modifying 169
 - numeric values 164
 - OGRain used with 165
 - querying 169
 - quoted strings 165
 - searching infile containing 180
 - SET command 163
 - sharing with ISPF 168
 - system variables 167
 - TEXT command 86
 - system variables
 - coordinates 168
 - status values 167
 - types of 167
- ## T
- tab character 83
 - in headers and footers 93
 - TABCH command 83
 - Table Reference Code (TRC)
 - character interspacing 74
 - COLOR command 70
 - HEIGHT command 68
 - selecting symbol sets 66
 - setting default value for 73
 - SIZE command 68
 - text formatting 99
 - table, translation 71
 - TABS command 83
 - TAG command 159
 - TAIL keyword on ARROW command 116
 - tails, controlling arrow 116
 - terminal, dialed 191
 - TEST command
 - modifying an object 28
 - testing commands 27
 - using 27
 - text
 - advanced text handling 73
 - alignment 81, 87
 - attributes setting 21
 - basic handling 65—73
 - box around 92
 - centering 81, 87
 - character interspacing 74
 - color of 69
 - setting defaults for 73
 - defaults, setting 21
 - formatting 95—101
 - bullets, changing 98
 - compact lists 98
 - examples in 98
 - nested lists 97
 - ordered lists 97
 - paragraph 96

index

text (*continued*)

- formatting (*continued*)
 - paragraph spacing 96
 - start 95
 - text flow header 96
 - TRCs in 99
 - unordered lists 97
 - writing markers 101
- height of 67
- in presentations 101
- indentation 80
- infile 76
- inserting space between 79
- left margin for 82
- mirroring 90
- shearing 78, 91
- size of 67
 - setting defaults for 73
- spacing 76
- starting point 79, 82
- string, specifying 86
- symbol set selection 65
 - setting defaults for 73
- TEXT command 90
- transforming 90
- turning 91

TEXT command

- aligning text 87
- alignment codes on 87
- BOX subcommand 92, 122
- coordinate prompting 25
- manipulating text strings 86
- OFFSET subcommand 89
- PFK command, subcommand on 57
- SHEAR subcommand 91
- specifying fonts 90
- symbolic variables with 86
- transforming text 90
- TURN keyword 91

TF command 95

TFEOL command 97

TFEUL command 97

TFEXMP command 98

TFH command 96

TFLI command 97

TFOL command 97

TFP command 96

TFUL command 97

- BULLET keyword 98

TFXMP command 98

THEN command 157

three-dimensional effect on boxes 120

TI command

- defining nameless global translation table 71
- MOD keyword 71

TOP command 177

transforming GDFs 139

translation tables

- introduction to 71
- named 71
- nameless global 71
- types of 71

TRC command

- character boxes 68
- COLOR command 70
- selecting symbol sets 66
- setting default value for 73
- text formatting 99

TRTDEF command 71

TRTDROP command 71

TRTUSE command 71

TSO

- allocating output data sets under 6
- command 174
- customizing ADMOPSLx 204
- executing commands 174
- OPS startup 201
- page segment output 187
- sample presentations under 8

turning

- boxes 122
- circles 123
- dasd symbols 127
- GDFs 142
- lines 113
- polygons 115
- text 91
- TEXT command 91

tutorial 195

U

underscore 56

- in variable names 163

UNDERSCORE keyword on field commands 56

UNDO command 26

units in coordinate system 62

unordered list 97

unspecific attributes

- DRAW command 105
- FILL command 108

UP command 177

US English startup EXEC (ADMOPSLA) 5

user storage needed for OPS 4

USERCTL command 60

V

value, keywords with 38

VARSUBST command 163

Vector Symbol Editor 173
vector symbol set
 default GDDM 66
VGET and VPUT keywords on ISPEXEC
 command 168
VSE command 173

W

WAIT command 181
 AUTO command used with 151
 pauses in presentations 151
WFILL command
 BOX command, subcommand on 120
 drawing wide lines 110
 LINE command, subcommand on 112
wide lines 110
wide lines, filling 112
WIDTH command
 BOX command, subcommand on 120
 LINE command, subcommand on 112
words, command 37
writing markers 101

Z

zooming pictures 60

Sending your comments to IBM

GDDM-PGF

OPS

User's Guide

SC33-1776-00

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., after your international access code use 44 1962 870229
 - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
 - IBM Mail Exchange: GBIBM2Q9 at IBMAIL
 - 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

OPS

User's Guide

SC33-1776-00

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	

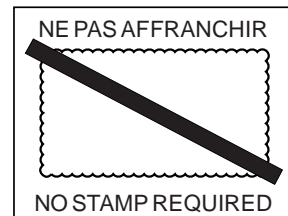
If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

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 _____

1 Cut along this line

4 Fasten here with adhesive tape



Printed in the United States of America
on recycled paper containing 10%
recovered post-consumer fiber.

SC33-1776-00

