GDDM

System Customization and Administration

Version 3 Release 2

SC33-0871-02

GDDM

System Customization and Administration

Version 3 Release 2

SC33-0871-02

Note!

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

Third Edition (December 2001)

This edition applies to these IBM GDDM licensed programs:

Program number	Program name	Version	Release	Modification
5695-167	GDDM/MVS	3	2	0
5684-168	GDDM/VM	3	2	0
5686-057	GDDM/VSE	3	2	0

GDDM/MVS as an element of OS/390 (program number 5645-001)

and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

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

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

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

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

Contents

Figures

Tables

Notices

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

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

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

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

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

Programming interface information

This book is intended to help you to customize and manage the day-to-day requirements of a GDDM installation.

This manual also documents General-use Programming Interface and Associated Guidance Information and Product-sensitive Programming Interface and Associated Guidance Information provided by GDDM Version 3 Release 2.

General-use programming interfaces allow the customer to write programs that obtain the services of GDDM.

General-use Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

General-use programming interface

General-use Programming Interface and Associated Guidance Information...

End of General-use programming interface

Product-sensitive programming interfaces allow the customer installation to perform tasks such as diagnosing, modifying, monitoring, repairing, tailoring, or tuning of GDDM. Use of such interfaces creates dependencies on the detailed design or implementation of the IBM software product. Product-sensitive programming interfaces should be used only for these specialized purposes. Because of their dependencies on detailed design and implementation, it is to be expected that programs written to such interfaces may need to be changed in order to run with new product releases or versions, or as a result of service.

Product-sensitive Programming Interface and Associated Guidance Information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking:

Product-sensitive programming interface

Product-sensitive Programming Interface and Associated Guidance Information...

End of Product-sensitive programming interface

Trademarks and service marks

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

Advanced Function Printing	AFP	APL2
AT	CICS	GDDM
IBM	InfoWindow	MVS/XA
OS/2	Personal System/2	Proprinter
PS/2	Quietwriter	VM/ESA
VM/XA	VTAM	3090
OS/390		

The following terms, used in this publication, are trademarks of other companies:

DEC	Digital Equipment Corporation
Hewlett Packard	Hewlett Packard Company
Helvetica	Allied Corporation
HP	Hewlett Packard Company
Monotype Times New Roman	Monotype Corporation Limited
PostScript	Adobe Systems Corporation
Tektronix	Tektronix Inc.
Univers	Allied Corporation

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

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

C-bus is a trademark of Corollary, Inc.

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

Preface

This book provides the information needed to customize these IBM GDDM licensed programs:

GDDM/MVS, program number 5695-167
GDDM/VM, program number 5684-168
GDDM/VSE, program number 5686-057
GDDM Presentation Graphics Facility (GDDM-PGF), program number 5668-812
GDDM-GKS, program number 5668-802
GDDM Interactive Map Definition (GDDM-IMD), program number 5668-801
GDDM Image View Utility (GDDM-IVU), program number 5668-723.

and GDDM/MVS as an element of OS/390 (program number 5645-001).

GDDM/MVS, GDDM/VM, and GDDM/VSE are known collectively as “GDDM Base.” GDDM Base includes GDDM-PCLK and GDDM-OS/2 Link. GDDM Base also includes GDDM-REXX in the VM/CMS and MVS/TSO environments.

Who this book is for

This book is for system-support personnel who need to tailor GDDM to meet local requirements, both immediately after GDDM is installed and as often as required thereafter.

What you need to know

It is assumed that users of this book have some experience of the subsystem under which GDDM is being run.

How to use this book

This book describes the following key topics:

- How to set up and maintain device support for GDDM users
- The effects of GDDM’s *external defaults*, and how to tailor them
- Code-page, font, and symbol-set support, and how it can be tailored
- Factors that affect GDDM performance, and how to tune them
- How to replace modules.

Use the book for the topics that interest you. You do not need to read the whole book.

The books of the GDDM libraries (GDDM Base, GDDM-PGF, GDDM-IVU, GDDM-GKS, and GDDM Interactive Map Definition) are listed in “GDDM publications” on page xxiii.

Summary of changes

This softcopy-only update corrects references to Taiwan. Here are the main changes for GDDM 3.2:

- The book has been divided into five Parts, and the chapter order has been rearranged.
- Information from the *GDDM Installation: Planning, Testing, and Servicing* books has been moved into this book. For example:
  - Chapter 9, “Reviewing the telecommunication network” on page 91
  - Chapter 24, “Working with GDDM-OS/2 Link” on page 271
  - Chapter 25, “Working with GDDM-PCLK” on page 279
  - Chapter 20, “Resource and capacity planning” on page 221.
- The Print Queue Manager (PQM) is described in Chapter 3, “The GDDM/MVS Print Utilities, ADMOPUT and ADMOPUJ” on page 15.
- New code pages are described in Chapter 16, “GDDM’s code-page support” on page 159.
- The new FRCETYPE processing option is described in Appendix B, “Processing options” on page 333.
- New device tokens are described in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Chapter 21, “Understanding GDDM performance” on page 227 has been revised.

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.

Books from related libraries

You might need to refer to some of these books, in addition to those from the GDDM libraries:

AFPDS	*AFPDS Data Stream Reference*, S544-3202
APL2	*APL2 Installation and Customization under CMS*, SH21-1062 *APL2 Installation and Customization under TSO*, SH21-1055 *APL2 Messages and Codes*, SH21-1059 *APL2 Diagnosis*, LY27-9601
CICS/ESA 4.1	*System Definition Guide*, SC33-1164 *Operations and Utilities Guide*, SC33-1167 *Resource Definition Guide*, SC33-1166
CICS/ESA 3.3	*System Definition Guide*, SC33-0664 *Operations Guide*, SC33-0668 *Resource Definition (Online)*, SC33-0666 *Resource Definition (Macro)*, SC33-0667
CICS/VSE 2.1, 2.2, 2.3	*System Definition and Operations Guide*, SC33-0706 *Resource Definition (Online)*, SC33-0708 *Resource Definition (Macro)*, SC33-0709
Composed Document Printing Facility (CDPF)	*Installation and Operations*, SC33-6135
DOS	*DOS 5.00 User's Guide and Reference*, Z84F-9779
GDDM/graPHIGS	*Installing GDDM/graPHIGS*, SC33-8101 *Understanding graPHIGS*, SC33-8102 *Messages and Error Codes for graPHIGS*, SC33-8105 *Problem Diagnosis for graPHIGS*, SC33-8108
GOCA	*Graphics Object Content Architecture Reference*, SC31-6804
IBM-GL	*IBM-GL Programming Manual (Graphics Language) for the IBM 6182, 6184, 6185, 6186, and 6187 Color Plotters*, SH23-0092.
IOCA	*Image Object Content Architecture Reference*, SC31-6805
IPDS	*Intelligent Printer Data Stream Reference*, GA34-2082 *IBM 3812 and 3816 Page Printers: IPDS Handbook*, GA34-2082 *IBM 3112 Page Printer and IBM 3116 Page Printer User's Guide*, G544-5253 *IBM 3912 and 3916 Page Printers Handbook*, S544-3901
JES/328X	*JES/328X Print Facility Program Description and Operations Manual*, SH20-7174
MO:DCA	*Mixed Object Document Content Architecture Reference*, SC31-6802
MVS	*MVS/XA Initialization and Tuning Guide*, GC28-1149 *MVS/XA Supervisor Services and Macro Instructions*, GC28-1154
Networking	*Network Program Products Samples: VM SNA*, SC30-3309

Operating System/2	Use the following generic titles that apply to the level of OS/2 you are using: *OS/2 Information and Planning Guide* *OS/2 System Administrator's Guide for Communications* *OS/2 Getting Started* *OS/2 User's Guide* *OS/2 EHLLAPI Programming Reference*
Print Services Facility	*System Programmer's Guide for MVS*, SH35-0091 *System Programmer's Guide for VM*, S544-3511 *System Programmer's Guide for VSE*, S544-3103 *Data Stream Reference, PSF/VM, PSF/MVS, PSF/VSE, and OS/400*, S544-3202
PTOCA	*Presentation Text Object Content Architecture Reference*, SC31-6803
TSO	*APL2 Installation and Customization under TSO*, SH20-9222 *MVS/XA TSO Guide to Writing a Terminal Monitor Program or a Command Processor*, GC28-1295
VM/ESA	*VM/ESA CP Planning and Administration*, SC24-5521 *VM/ESA CMS Planning and Administration*, SC24-5445 *VM/ESA Procedures Language VM/REXX Reference*, SC24-5466 *VM/ESA CP Command and Utilities Reference*, SC24-5519 *VM/ESA CMS Command Reference*, SC24-5461
VSAM	*Using VSE/VSAM Commands and Macros*, SC24-5144 *VSE/VSAM Messages and Codes*, SC24-5146 *MVS/XA VSAM Catalog Administration Access Method Services Reference*, GC26-4136
VSE/ESA	*VSE/ESA Planning*, SC33-6503 *VSE/ESA Installation*, SC33-6504 *VSE/ESA System Control Statements*, SC33-6513
3117 scanner	*IBM 3117 Scanner and IBM 3117 PC Adapter Guide to Operations*, GA18-2477 *IBM 3117 Scanner and Extension Unit Guide to Operations*, GA18-2478 *IBM 3117 Scanner Hardware Maintenance and Service*, SY18-2159 *IBM 3117 Scanner Technical Reference*, SC18-2105
3118 scanner	*Scanner Guide to Operations*, GA18-2475 *High Speed Adapter Guide to Operations*, GA18-2476 *IBM 3118 Scanner Hardware Maintenance and Service*, SY18-2158 *High Speed Adapter Hardware Maintenance and Service*, SY18-2167 *Scanner Technical Reference*, SC18-2104 *High Speed Adapter Technical Reference*, SC18-2117
3174 establishment controller	*Data Stream Programmer's Reference*, GA23-0059 *Functional Description*, GA23-0218 *Terminal User's Reference for Expanded Functions*, GA23-0332 *Planning Guide Configuration Support B Release 2*, GA27-3862
3179-G, 3192-G	*3179-G and 3192-G Color Graphics Display Station Description*, GA18-2589

3193 display station	*Description*, GA18-2364 *Setup Instructions*, GA18-2366 *Operator's Guide*, GA18-2365 *Problem Solving Quick Check Guide*, GA18-2443 *Problem Solving Guide*, GA18-2444
3270-family devices	*3270 Information Display System Configurator*, GA27-2849 *3270 Information Display System Data Stream Programmer's Reference*, GA23-0059 *8775 Display Terminal: Component Description*, GA33-3044
3270-PC/G and 3270-PC/GX workstations	*IBM 3270 Information Display System: Color and Programmed Symbols*, GA33-3056 *Introducing the IBM 3270 Personal Computer/G and /GX Ranges of Workstations*, GA33-3157 *3270-PC/G Personal Computer/G and /GX Ranges of Workstations; Planning Guide*, GA33-3158 *3270-PC/G Guide to Operations*, SA33-3155 *3270-PC/GX Guide to Operations*, SA33-3156
3274 control unit	*3274 Control Unit Description and Programmer's Guide*, GA23-0061 *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827
3472-G display	*3472-G User's Guide*, GA18-7026
3812 printer	*IPDS NDS Attachment Feature Installation and Programming Instructions*, S544-3101 *Guide to Operations*, S544-3267
3816 page printer	*IBM 3816 Page Printer Operating Instructions*, GA34-2075
3820 page printer	*IBM 3820 Page Printer Operator's Guide*, S544-3080 *IBM 3820 Page Printer Reference Manual*, S544-3175
3825 page printer	*3825 Page Printer Operator's Guide*, G544-3481 *3825 Page Printer Product Description*, G544-3482
3827 page printer	*3827 Page Printer Operator's Guide*, G544-3189 *3827 Page Printer Product Description*, G544-3194
3828 advanced function MICR printer	*IBM 3828 Advanced Function MICR Printer Operator's Guide* , S544-3360 *IBM 3828 Advanced Function MICR Printer Product Description*, S544-3361
3835 page printer	*3835 Page Printer Product Description*, G544-3498 *3835 Page Printer Operator's Guide*, G544-3208
3900 advanced function printer	*IBM 3900 Advanced Function Printer Product Description*, GA32-0135 *IBM 3900 Advanced Function Printer Operator's Guide*, GA37-0210
4028 printer	*IBM LaserPrinter 4028 Introduction and Planning Guide*, S544-4258 *IBM LaserPrinter 4028 IPDS Handbook*, S544-4260 *3270 Programming Guide and Reference Manual for the IBM LaserPrinter 4028 Model NS1*, S544-4262 *IBM LaserPrinter 4028 Model NS1 Guide to Operations*, S544-4263
4224 printer	*Printer Product and Programming Description Manual*, GC31-2551 *Operating Instructions*, GC31-2546 *Guide to Operations*, GC31-3621

4230 printer	*4230 Models 102/202 Printer Product and Programming Description*, GC40-1701 *4230 Models 102/202 User's Guide*, SA40-0564 *4230 Models 102/202 Operator's Panel Instructions*, SA40-0565
4234 printer	*4234-11,12,13 Product Description and Programming Manual*, GC31-3879 *4234-11 Operating Instructions*, GC31-3736
4250 printer	*Operator's Guide*, GA33-1551
5550 multistation (available in Japanese only)	*5550 Japanese 3270-PC User's Guide*, N:SC18-2059 *How To Use 5550 Japanese 3270-PC*, N:SC18-2060 *5550 Japanese 3270-PC/G User's Guide*, N:SC18-2071 *How To Use 5550 Japanese 3270-PC/G*, N:SC18-2072 *5550 Small Cluster User's Guide*, N:SC18-2092 *How To Use 5550 Small Cluster*, N:SC18-2091 *5550 Small Cluster/Graphics User's Guide*, N:SC18-2107 *How To Use 5550 Small Cluster/Graphics*, N:SC18-2108 *5550 3270 Kanji Emulation Description*, N:SC18-2020 *5550 3270 Kanji Emulation Operator's Guide*, N:SC18-2021
6180 color plotter	*Guide to Operations*, GA66-0500
6182 color plotter	*Guide to Operations*, SA37-0100
6186 plotter	*Guide to Operations*, SH23-0093
6187 plotter	*Guide to Operations*, SH52-0279

Part 1. GDDM output

Chapter 1. An introduction to GDDM output families

This chapter introduces some of the key concepts that you need to understand to be able to set up and maintain devices on which GDDM output can be displayed and printed.

GDDM output processes are considered to belong to one of four groups or *families*. These groupings do not reflect any single feature of an output process, such as the print device, but are expressions of many factors, including:

- The type of data being printed or displayed (alphanumerics, graphics, image, or a combination of these)
- The subsystem under which GDDM is running
- The type of output device being used
- The way in which the output device is attached.

Each of the four output families is described below, beginning with family-1.

Family-1 output

A family-1 output destination is a 3270-data-stream terminal, or any 3270-data-stream printer or plotter directly attached to that terminal. Plotters are usually family-1 devices. Family-1 devices can include the IBM 3816 and 4224 printers, and the IBM 6187 plotter. In the VM environment only, a printer directly attached to a user ID can be accessed as a family-1 printer from that user ID. Family-1 output is written *directly* to the device by GDDM. (Printers attached via GDDM-OS/2 Link or GDDM-PCLK can be used as family-1 devices, regardless of the host subsystem under which GDDM is running.)

Family-2 printing

Family-2 printing, which is also known as *queued printing*, is a two-step process:

1. Data to be printed is written to an intermediate file or data set in a GDDM-specific format.
2. The contents of the intermediate file or data set are printed by the *GDDM print utility*.

The GDDM print utility executes independently of the user. It writes directly to the output device (which it accesses as a family-1 device), and one instance of the utility can manage multiple printers.

A request for family-2 output (on a DSOPEN call, whether from a locally written application or from a GDDM utility, such as the GDDM-PGF ICU or User Control) results in the creation of a GDDM print file or data set. Like family-1 output, family-2 output can be printed on 3270-data-stream devices only. It can go to a 3270-family printer (such as the 3816) or (except under IMS) to a plotter attached to a workstation running GDDM-PCLK or GDDM-OS/2 Link, for example.

How the two steps of the family-2 printing process work in practice is largely dependent on the subsystem under which GDDM is running. The name of the GDDM print utility also varies from one subsystem to another. Family-2 printing under each subsystem is described below.

Family-2 printing in the VM environment

In the VM environment, a request for family-2 printing causes a file of type ADMPRINT to be created on the user's A-disk. The filename is the name of a printer that was supplied on input to the DSOPEN call. After the ADMPRINT file has been created, it can be printed on the target device by the GDDM/VM print utility, ADMOPUV, as follows:

- ADMOPUV is run in the user's VM virtual machine. The printer to which ADMOPUV is writing must be directly attached to the same virtual machine. All other processing is suspended until output is complete.
- ADMOPUV is run in a separate virtual machine to which the printer is directly attached and to which the ADMPRINT file is spooled automatically by the ADMQPOST EXEC.
- ADMOPUV is used to send the ADMPRINT data as a punch file to RSCS for printing.

If ADMOPUV runs in the user's virtual machine, it can be invoked explicitly by the user, or it can be invoked automatically by means of a *nickname* statement. The nickname mechanism enables you to set this up for all users. ADMOPUV is described in more detail in Chapter 2, "The GDDM/VM Print Utility, ADMOPUV" on page 9. Nicknames are described in Chapter 18, "Nickname user-default specifications" on page 205.

Family-2 printing in the CICS environment

A request for family-2 printing in the CICS environment results in the creation of a data set in CICS temporary storage (DFHTEMP) and an entry in the CICS temporary storage queue. The queue name begins with the characters "ADMT," though this can be changed by the CICSTSPX *external default*. (External defaults are described in Chapter 17, "GDDM user-default specifications" on page 197.) The GDDM print utility is known as ADMOPUC in the CICS environment. It runs as a CICS transaction and does not have to be invoked explicitly. That is, family-2 print requests automatically schedule ADMOPUC using the CICS interval control facility. Such print requests are retrieved from temporary storage by ADMOPUC and written directly to the printer.

If the CICS temporary storage is defined as recoverable, the ADMOPUC transaction is not started until a synchronization point (SYNCPOINT) is reached. Printing might therefore be delayed until the display task terminates, or until a user SYNCPOINT is reached. The GDDM-PGF ICU does not contain any explicit SYNCPOINTs.

Family-2 printing in the TSO environment

In the TSO environment, a request for family-2 printing creates a temporary print data set `userid.ADMPRINT.REQUEST.#nnnnn`, where "nnnnn" is a sequence number. When the temporary data set is created, an entry is made in the *GDDM master print queue* that links the VTAM LUID of the target printer with the temporary print data set.

The GDDM print utility in the TSO environment, ADMOPUT, is a VTAM application that runs independently of the user. It can run continuously, or be invoked as a standard batch job. ADMOPUT scans the GDDM Master Print Queue for print-request entries, acquires the appropriate printer, prints the data set, deletes the data set, and releases the printer if no further print data sets are queued for output to that printer. Print requests for each device are processed in the order in which they are received.

Alternatively, a family-2 print request under TSO can be written to the Job Entry Subsystem (JES) spool and printed by ADMOPUJ.

For more information about family-2 printing under TSO, see Chapter 3, "The GDDM/MVS Print Utilities, ADMOPUT and ADMOPUJ" on page 15.

Family-2 printing in the IMS environment

The GDDM print utility in the IMS environment is called ADMOPUI. It can run as a message-processing program or as a batch message program (BMP). When it runs as a message-processing program, no explicit invocation is required. When it runs as a BMP, standard JCL is required to start it. ADMOPUI reads temporary data sets from the message queue and writes them to the printer device. ADMOPUI cannot direct output to plotters.

The family-2 header page

The GDDM print utility prints a header page at the beginning of each file it sends to a printer, and performs a page-eject at the end of each file. The header page records the date and time of the file's creation and identifies its origin. The origin information for the print file depends on the subsystem and is one of the following:

CICS	The transaction identifier
IMS	The user ID, the logical terminal, or asterisks if neither value is available.
TSO	The user ID
VM/CMS	The user ID

A header page is not produced at the start of each file written to a plotter. However, the ORIGINID processing option (which is described in Appendix B, "Processing options" on page 333) can be used to superimpose an origin-identification string containing similar information on each page.

Family-2 plotting

To cause the GDDM print utility to send a print file to a plotter, it is usually necessary to specify the DSOPEN processing option, STAGE2ID, when the print file is created.

To ensure that control information for the plotter is honored by GDDM, nickname statements are required in the defaults file.

For plotters that do not support page feed, the GDDM print utility begins to plot on a device as soon as it receives a print file for that device. At the same time, it sends a status message to the associated 3179-G, 3192-G, or 3472-G color display station, 3270-PC/G or 3270-PC/GX work station, or device supported by GDDM-PCLK.

The GDDM print utility pauses at the end of each page that is plotted to give the operator an opportunity to reload the plotter device. GDDM sends another status message at that time and prompts the operator to press any attention key to cause plotting to continue, or to complete the processing of the print file.

At any time, the plotting of the current page may be canceled by pressing the CLEAR key on the associated workstation.

If any errors are detected during the plotting process, the error messages are added to the status messages on the associated 3179-G, 3192-G, or 3472-G color display station, 3270-PC/G or 3270-PC/GX work station, or device supported by GDDM-PCLK. These messages are accumulated; each error message is prefixed by the number of the page that was being plotted when the error was detected and, if possible, by the function that GDDM was running at the time the error was detected.

If multiple copies are plotted, the process is repeated for each copy.

Note: Any nickname processing for the associated 3179-G, 3192-G, or 3472-G color display station, or 3270-PC/G or 3270-PC/GX work station, is suppressed during the time that plotting takes place.

Error reporting by the GDDM print utility

If the GDDM print utility detects errors while printing, it produces an error page that lists a maximum of 19 errors. Each error message is prefixed by the number of the page that was being printed when the error was detected and, where possible, with an indication of the function GDDM was running at the time. The page count begins with the header page (if one has been produced).

If multiple copies of a file are being printed, an error page is usually printed following each copy during which errors were detected. GDDM attempts to optimize the output of error messages for multiple copies of a file, particularly where the print file itself is a single page. In such cases, some errors will be detected during output of the first copy only.

If the error messages cannot be printed (for example, if errors occur during initialization of the print utility), they are written to a system-dependent destination:

CICS	The error log
IMS	The Master Terminal Operator (via broadcast)
TSO	The system operator
VM/CMS	The terminal operator

Family-3 printing

Family-3 printing, which is also known as *system printing*, is for alphanumeric data only. Output is written to external storage as a text file that is formatted for a particular device type. Family-3 output can also be directed to the system spool for later printing, without saving the output in intermediate storage.

If output is written to intermediate storage:

- Under CMS, the file is of type ADMLIST by default. The default filetype can be altered using the CMSSYSP external default.
- Under CICS, output is written to the system printer output queue defined in the DCT. The name of the default queue is ADMS, though this can be changed using the CICSYSP external default.
- Under TSO, output is written to ddname ADMLIST by default. This name can be changed using the TSOSYSP external default.
- Under IMS, output is written to output destination ADMLIST by default. This name can be changed using the IMSSYSP external default.

Printers that can be used for family-3 output include the IBM 4228, the IBM 3262 Models 1, 5, and 11, and all IBM 38xx advanced-function printers in nonpage mode.

Family-4 printing

Family-4 printing is available in the VM and TSO environments.[1] Family-4 printing is submitted to advanced-function printers, such as the IBM 3825 and the IBM 3900, which are usually managed by the IBM Print Service Facility (PSF). Family-4 printing can also be directed to the IBM 4250 printer via the Composite Document Printing Facility (CDPF). Output directed to PSF-managed printers is in Advanced Function Presentation Data Stream (AFPDS) format. The output is written by default to a file of type ADMIMAGE or to a ddname of ADMIMAGE before being passed to either PSF or CDPF.

In the VM environment, such output can be spooled automatically to PSF/VM if the CPSPOOL processing option is set. (For more information, see Appendix B, “Processing options” on page 333.)

In the TSO environment, the output can be spooled automatically to PSF/MVS via JES. Family-4 output can also use the POSTPROC processing option to postprocess the output file (using a procedure or program).

[1] It is also supported in the VSE/Batch environment, as described in Chapter 4, “The GDDM/VSE Batch Print Utility, ADMUPRTC” on page 31.

Family-4 output can be a document of one or more pages containing a mixture of alphanumerics, graphics, and image; or a page segment (which can be included in a DCF document to be printed). Alphanumeric data can also be printed on advanced-function printers under VM/CMS (via the ADMOPUV or ADMOPUJ print utility) and under TSO (via the ADMOPUT print utility) if appropriate nickname statements (specifying TOFAM=4) are in effect. Nicknames are described in Chapter 18, “Nickname user-default specifications” on page 205.

Chapter 2. The GDDM/VM Print Utility, ADMOPUV

In the VM/CMS environment, a request from a GDDM application program or from the GDDM-PGF Interactive Chart Utility (ICU) for printing on a 3270-family printer (that is, a family-2 print request) causes a print file to be created on the user's A-disk. If the default operation of the GDDM/VM Print Utility ADMOPUV is not altered, the user has to invoke ADMOPUV explicitly to cause the file to be printed on a specified, directly attached, printer. ADMOPUV runs in the user's virtual machine and all other processing is suspended until printing is complete.

This chapter describes how you can improve on the default scenario. In particular, it describes:

- How the user invokes ADMOPUV explicitly
- How to make invocation of ADMOPUV automatic
- How to print family-2 output via RSCS
- How to enable automatic transmission of print files from the user's virtual machine to a disconnected virtual machine, from where ADMOPUV prints to a directly attached printer

Invoking ADMOPUV explicitly

To start ADMOPUV from the CMS command line, enter the following:

```
ADMOPUV filename
        [filetype[filemode]]
        [ON cuu]
        [([CC | NOCC]          [DEV device token])]
```

filename
The name of the file to be printed. It must be specified.

filetype
The filetype of the file to be printed. This is usually ADMPRINT. If it is omitted, ADMPRINT is assumed unless it has been overridden by the CMSPRNT external default. CMSPRNT is described in Appendix A, "External defaults" on page 307.

filemode
The filemode of the file to be printed. "*" is the default.

cuu
The printer-device name or address. This identifies a non-VTAM-attached printer that must be attached to this user ID (virtual machine) using the CP ATTACH command. (CP ATTACH is not available to the general user.)

If no *cuu* value is specified, the device is identified by the STAGE2ID processing option that was specified when the print file was created. If no STAGE2ID value was specified, *cuu* defaults to 061.

CC | NOCC
CC causes the first character of each record to be interpreted as a carriage-control character. NOCC causes the first character of each record to be interpreted as part of the data. (Carriage-control characters are processed as described in the description of the FSLOGC call in the *GDDM Base Application Programming Reference* book.)

DEV *device token*
Passes a device token to ADMOPUV. This allows the device characteristics that GDDM would usually infer to be overridden. Device tokens are described in Appendix C, “Device tokens supplied by GDDM” on page 367.

Invoking ADMOPUV automatically

The processing option INVKOPUV causes ADMOPUV to be invoked automatically whenever a family-2 print file is created. You can specify INVKOPUV for your enterprise by adding suitable nickname statements to the GDDM external-defaults module, ADMADFV. For example, the following nickname statement applies the INVKOPUV option for a printer attached as 062:

```
[label] ADMMNICK FAM=2,NAME=062,PROCOPT=((INVKOPUV,YES))
```

The next statement applies the option for *all* 3270-family printers in your enterprise:

```
[label] ADMMNICK FAM=2,PROCOPT=((INVKOPUV,YES))
```

When INVKOPUV is set, family-2 printing requests cause a temporary print file to be created. The printer name specified by the application program or the ICU user is taken to identify a directly attached printer on which the file should be printed. GDDM processing equivalent to that performed by ADMOPUV occurs automatically and immediately (in the user’s virtual machine) to cause the print file to be printed on the named printer. The temporary file is then deleted.

If the printer to which you are sending output cannot be queried, you need to ensure that the device token is associated with the temporary print file when it is created and with the print device when this file is sent to print. You can do this by specifying an additional nickname statement such as the following one:

```
[label] ADMMNICK NAME=062,DEVTOK=X4224SE
```

Sending output to an RSCS destination

If the Remote Spooling Communication Subsystem (RSCS) Version 2 (or later) is available at your location, output can be directed to a printer attached to RSCS.[2] This is usually more efficient for printing large files than using ADMOPUV directly. To cause the print file to be sent to the virtual punch rather than to a real 3270-data-stream printer, you specify “PUNCH” as the printer address on input to ADMOPUV. You must issue suitable CP SPOOL and CP TAG commands before entering the ADMOPUV command. If you do this, the command:

```
ADMOPUV filename ON PUNCH (DEV devtok
```

can be used to direct a printer data stream to a suitable virtual machine capable of printing such data streams. In the above command, `devtok` is a device token that provides GDDM with a description of the intended printer device. (Device tokens are described in Appendix C, “Device tokens supplied by GDDM” on page 367.)

You can specify the device token, the CP SPOOL and CP TAG commands, and the printer address of PUNCH for your enterprise by adding suitable nickname

[2] For printing on IPDS printers, RSCS Version 2.2 or later is required.

statements to the GDDM/VM external-defaults module ADMADFV. For example, if you are using RSCS, the following nickname statement:

```
[label] ADMMNICK FAM=1,NAME=3287N03,TONAME=PUNCH,DEVTOK=L87,
             PROCOPT=((CPSPOOL,TO,RSCS),
             (CPTAG,CHICAGO,3287N03,50,PRT=GRAF))
```

causes the command:

```
ADMOPUV filename ON 3287N03
```

to punch a 3287 data stream to RSCS node ID `CHICAGO` with a tag of "`3287N03 PRT=GRAF`" applied, and with a transmission priority of 50.

This nickname statement could be combined with an INVKOPUV nickname statement to make this invocation of ADMOPUV automatic.

Note: If the print file goes through any node in the network where the RSCS system does not support PRT=GRAF or strips TAG options from print files (RSCS V1, for example), the file prints incorrectly.

Transmitting print files to a disconnected virtual machine using ADMQPOST

GDDM/VM supplies a facility (the ADMQPOST EXEC) for transmitting print files from a user's virtual machine to a disconnected virtual machine for printing. This might be of interest to you if your enterprise does not have a virtual machine capable of printing data streams generated by the PUNCH facility. It could also be used, for example, to transmit family-2 output automatically to JES for printing via MVS.

You are recommended to create an ADMQPOST EXEC and make it available to your users, such that it is invoked automatically by GDDM whenever a family-2 print file is created. You also need to attach printers to the disconnected machine, and cause the virtual machine to invoke ADMOPUV as required. This enables you to centralize printing of GDDM print files, and frees your users from having to wait while the print files are being processed.

Figure 1 on page 12 shows an example ADMQPOST EXEC. Note that GDDM enters CMS subset mode to invoke the EXEC, which limits the range of functions you can perform.

```
/*********************************************************************/
/*                                                                   */
/* Example ADMQPOST EXEC                                             */
/* This EXEC is designed to be executed when a GDDM print file is    */
/* closed.                                                           */
/*                                                                   */
/* Required parameters are the print file FILENAME FILETYPE FILEMODE */
/* The filename is assumed to be the userid of the disconnected      */
/* virtual machine which will print the file.                        */
/*                                                                   */
/*********************************************************************/
ADDRESS 'COMMAND'
PARSE UPPER ARG filename filetype filemode . /* get parameters       */

'CP SPOOL PUN CLOSE NOHOLD NOCONT'
'CP SPOOL PUN TO' filename
'PUNCH' filename filetype filemode '(NOHEADER'

'ERASE' filename filetype filemode

'CP SPOOL PUN CLOSE'
```

Figure 1. An example ADMQPOST EXEC

You also need to provide an EXEC to control the operation of the disconnected virtual machine. An example EXEC for this purpose is shown in Figure 2 on page 13. Note, however, that the example EXEC fulfils only the basic requirements and you are likely to want to produce something more sophisticated.

If your EXECs require your users to name their print files in any special way (as do the sample EXECs shown), remember to inform your users of these requirements.

```
/*********************************************************************/
/*                                                                   */
/* Example EXEC to run in a disconnected Virtual Machine to read in  */
/* ADMPRINT files and print them using ADMOPUV.                      */
/*                                                                   */
/* Required parameters are PRINTER_NAME and SLEEP_SECONDS            */
/* The EXEC will print all files currently in the reader and then    */
/* wait for an interval of SLEEP_SECONDS seconds.                    */
/*                                                                   */
/*********************************************************************/
ADDRESS 'COMMAND'

/*********************************************************************/
/*                                                                   */
/*         Read the input parameters and check them                  */
/*                                                                   */
/*********************************************************************/
PARSE UPPER ARG prtname sleepsecs .   /* get parameters              */
IF DATATYPE(prtname,ALPHANUMERIC) ¬= 1 | ,
   DATATYPE(sleepsecs,WHOLE NUMBER) ¬= 1 THEN DO
  SAY 'Required parameters are PRINTER_NAME and SLEEP_SECONDS'
  SAY 'A syntax error has been detected, please respecify parameters'
  EXIT                              /* exit if incorrect parameters  */
END

/*********************************************************************/
/*                                                                   */
/*         Read in files from the virtual reader and print them.     */
/*         If a file fails to print it is erased.                    */
/*                                                                   */
/*********************************************************************/
'ERASE PRTFILE ADMPRINT A'          /* Make sure file doesn't exist  */

DO FOREVER

  'READCARD PRTFILE ADMPRINT A'     /* Try reading in a file         */

  IF rc = 0 THEN DO                 /* If a file was read in then    */
    'ADMOPUV PRTFILE ON' prtname    /* Print file using GDDM         */
    IF rc ¬= 0 THEN SAY 'Print failed, RC =' rc  /* Message if failed */
    'ERASE PRTFILE ADMPRINT A'      /* Always erase the print file   */
  END
  ELSE 'CP SLEEP' sleepsecs 'SEC'   /* Sleep when no more to print   */

END
```

Figure 2. Example EXEC for printing from a disconnected virtual machine

Directing VM family-2 output to family-4 devices

By specifying an appropriate `FAM=2,TOFAM=4` nickname statement, you can enable alphanumeric family-2 output to be directed automatically to a server machine running PSF/VM or some other AFPDS processor. For example:

```
ADMMNICK FAM=2,NAME=PSFPRINT,TOFAM=4,TONAME=PRINTER,DEVTOK=A3820Q,
              PROCOPT=((CPSPOOL,CLASS,n,DEST,destname,FORM,formname))
```

The `TONAME=PRINTER` value is a special extension of the VM namelist parameters and is described in Appendix D, “Name-lists” on page 383. For a description of the CPSPOOL processing option, see Appendix B, “Processing options” on page 333.

Chapter 3. The GDDM/MVS Print Utilities, ADMOPUT and ADMOPUJ

In the TSO environment, there are three possible routes for a family-2 print request to follow:

- By default, output data is written to a temporary print data set `userid.ADMPRINT.REQUEST.#nnnnn`, where "nnnnn" is a sequence number. When the temporary data set is created, an entry is made in a print-request queue (called the "GDDM Master Print Queue") that links the VTAM LUID of the target printer with the temporary print data set. (The VTAM LUID of the target printer is specified by the user when the print request is issued.)

  The GDDM/MVS Print Utility ADMOPUT, a VTAM application that runs independently of the user, scans the GDDM Master Print Queue for print-request entries. Print requests are processed in the order in which they are received. ADMOPUT acquires the appropriate printer, prints the data set, deletes the data set, and releases the printer if no further print data sets are queued for output to that printer.

  The status of the Master Print Queue can be ascertained by the GDDM Print Queue Manager, which manages the request queue independently of ADMOPUT.

- Print data sets can be written directly to the JES spool. JES directs each request to the JES/328X Print Facility, a Program Offering that extends support of remote-job entry (RJE) devices to 3270-data-stream printers. JES/328X directs the output to a special version of the GDDM/MVS Print Utility, ADMOPUJ, which in turn sends the output to the printer.

  Alternatively by using the PRINTDST processing option to specify certain JES spool parameters, the output on the JES spool can be directed to an external print application for printing.

- Output can be written to a DD destination, or data set other than the temporary print data set, for later processing (either via batch JCL or via the JES/328X DSPRINT command.)

Which of these routes for family-2 output is used is specified on the PRINTDST processing option, which is described in Appendix B, "Processing options" on page 333.

This chapter describes in more detail the two print utilities, ADMOPUT and ADMOPUJ.

The TSO Print Utility, ADMOPUT

This section describes:

- How to set up the GDDM Master Print Queue
- The ways in which ADMOPUT can be started
- How to manage the GDDM Master Print Queue by using the GDDM Print Queue Manager (PQM)
- How to cancel a job submitted to ADMOPUT
- How to print alphanumeric files via ADMOPUT.

Setting up the GDDM Master Print Queue

The GDDM Master Print Queue data set contains information about the jobs that are waiting to print and about the printers for which they are destined. By default, the GDDM Master Print Queue is called `ADMPRINT.REQUEST.QUEUE`. You can specify an alternative to the `ADMPRINT` component of the name on the TSOPRNT external default. (External defaults are described in Appendix A, “External defaults” on page 307.)[3]

GDDM/MVS provides a sample job stream for creating and initializing the GDDM Master Print Queue.[4] This sample job stream is supplied as member ADMQFMT in the GDDM sample library. You need to edit ADMQFMT to include details of the printers that ADMOPUT is to service. An extract from ADMQFMT is shown in Figure 3.

```
//*****************************************************************
//* BUILD THE PRINTER DEFINITIONS
//*****************************************************************
//BUILDDEF EXEC PGM=IEBGENER,REGION=1024K
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DUMMY
//SYSUT2   DD DSN=&&TEMPIN,UNIT=SYSDA,DISP=(NEW,PASS),
//             SPACE=(2480,(1000,1000))
//SYSUT1   DD   *
*********************************************************************
*                                                                   *
*                    CONTROL  STATEMENTS                            *
*                                                                   *
*********************************************************************
* TIMER DEFINES THE POLL INTERVAL IN TENS OF SECS
* EXTRA DEFINES THE NUMBER OF PRINTERS THAT CAN BE DYNAMICALLY ADDED
         HEADER   TIMER=4,EXTRA=30
*
* ADD MORE PRINTER STATEMENTS HERE AS REQUIRED
         PRINTER  PRINTER1,SIZE=1920 PRINTER NAME,BUFFER SIZE
         PRINTER  PRINTER2,SIZE=1920
         PRINTER  PRINTER3,SIZE=1920
         PRINTER  PRINTER4,SIZE=1920
         PRINTER  PRINTER5,SIZE=1920
         PRINTER  PRINTER6,SIZE=1920
*
* PQECNT DEFINES THE MAXIMUM NUMBER OF PRINT REQUESTS WHICH CAN BE
* ACTIVE AT ONE TIME. INCREASE THE NUMBER IF REQUIRED
         TRAILER  PQECNT=186         ACTIVE PRINT REQUEST LIMIT
         END
```

Figure 3. An extract from ADMQFMT, which creates and initializes the GDDM Master Print Queue

---

[3] Any value specified on the TSOPRNT external default applies to both the name of the GDDM Master Print Queue data set and the name of each temporary print data set created by a print request.

[4] If you are migrating to GDDM 3.2, you do not have to recreate the GDDM Master Print Queue data set unless you want to define additional devices, for example.

Notes:

1. By default, the interval at which ADMOPUT searches the GDDM Print Master Queue (for files with output addresses defined to VTAM) is set to 40 seconds. You can change this by editing the line:

   ```
   HEADER   TIMER=4
   ```

   Note that the TIMER= value is specified in tens of seconds.

   If a print request is submitted for a printer that is inactive or offline (GDDM message ADM0415), ADMOPUT attempts to submit the job for printing at the interval specified on the TIMER= value until the printer is either reactivated or removed from the Master Print Queue. If the polling interval is too short, there could be a performance impact from ADMOPUT's polling of inactive devices. If the value specified on TIMER is too great, the long polling interval can lead to delays when shutting down the print utility after replying stop to the outstanding reply. Files with addresses not known to VTAM (such as "HOLD") are ignored.

2. The EXTRA parameter specifies the number of spare slots to be formatted in the Master Print Queue to allow printer definitions to be added dynamically to the print queue by the Print Queue Manager (PQM). The default value is 30.

3. The PRINTER entries identify the printers on which ADMOPUT is going to print. Any printer not defined here cannot be used by ADMOPUT unless it is added by the Print Queue Manager. There must be one entry for each printer. The printer name is the name by which the printer is known to VTAM (its LUID).

   The SIZE value is the hardware-buffer size. It must be specified, even though it is overridden on queriable printers by the size given on execution-time bind parameters.

   Thus, if you have two printers, one with an LU name of GPRINT1 and the other with an LU name of GPRINT2, and you expect a buffer size of 1920 to be used, you would define them in ADMQFMT as:

   ```
   PRINTER  GPRINT1,SIZE=1920
   PRINTER  GPRINT2,SIZE=1920
   ```

   All other PRINTER lines must be deleted. You can specify any number of printers, up to the maximum allowed by the subsystem or access method.

   If you need to add or remove printers at a later stage, stop ADMOPUT while you rerun ADMQFMT or use the Print Queue Manager.

4. Specify on the PQECNT value the maximum number of print data sets (that is, requests for printing) that you will allow at any one time. You can afford to be generous with this value, because each entry in the GDDM Master Print Queue requires approximately 80 bytes of direct-access storage only. By default, this value is set to 186. For example, if you estimate that you will never have more than 200 data sets queued for printing at the same time, you would specify PQECNT=200.

5. By default, ADMQFMT creates the GDDM Master Print Queue data set on unit SYSDA. If you want to locate the data set more explicitly, change the UNIT=SYSDA value in step CREATEQ of ADMQFMT to specify volume and unit information.

6. When you have finished editing ADMQFMT, submit the job stream to create and initialize the GDDM Master Print Queue. ADMQFMT uses the IEBDG utility to initialize the print queue. The step invoking IEBDG can take a noticeable period of time to complete.

The MVS macros RESERVE and DEQ can be used to reserve the DASD device on which the Master-Print-Queue data set resides whenever the data set is updated, and to release it when the update is complete. This protects the Master Print Queue from multiple updates, and is of particular benefit where print queues are shared. To request this protection, you set the TSORESV external default to "YES." TSORESV is described in Appendix A, "External defaults" on page 307.

TSORESV=YES reserves the whole of the DASD device on which the Master-Print-Queue data set resides. If your enterprise has multiple processors, TSORESV must have the same value on all of them to ensure full protection of the data set. If TSORESV=YES when an MVS system fails, the RESERVE is released when MVS is restarted. If GDDM itself fails, all resources are automatically released during task termination: there is no need to restart MVS to free resources when this happens.

How to invoke ADMOPUT

When the GDDM Master Print Queue data set has been created and initialized, ADMOPUT can be started by submitting the job-control statements shown in Figure 4, modified as necessary for your enterprise. For example, you can modify this JCL to respecify the library on which the utility is kept. The job-control statements are usually held in SYS1.PROCLIB.

ADMOPUT can be started either as a submitted batch job or as a started task. Figure 4 provides statements for both approaches to the task.

JOB STREAM FOR STARTING AS A BATCH JOB

```
//applname  EXEC    PGM=ADMOPUT,DYNAMNBR=n,REGION=mK,
//                  PARM='NAME=xxxx,MAXPRTRS=nnnn'
//STEPLIB   DD      DSN=GDDM.SADMMOD,DISP=SHR
//ADMSYMBL  DD      DSN=GDDM.SADMSYM,DISP=SHR
//ADMGGMAP  DD      DSN=GDDM.SADMMAP,DISP=SHR           <optional>
//ADMPRNTQ  DD      DSN=ADMPRINT.REQUEST.QUEUE,DISP=SHR <optional>
//ADMDEFS   DD      DSN=YOUR.ADMDEFS,DISP=SHR           <optional>
//SYSABEND  DD      SYSOUT=A
```

JOB STREAM FOR STARTING AS A STARTED TASK

```
//          PROC    PGM=ADMOPUT,DYNAMNBR=n,REGION=mK,
//                  PARM='NAME=xxxx,MAXPRTRS=nnnn'
//STEPLIB   DD      DSN=GDDM.SADMMOD,DISP=SHR
//ADMSYMBL  DD      DSN=GDDM.SADMSYM,DISP=SHR
//ADMGGMAP  DD      DSN=GDDM.SADMMAP,DISP=SHR           <optional>
//ADMPRNTQ  DD      DSN=ADMPRINT.REQUEST.QUEUE,DISP=SHR <optional>
//ADMDEFS   DD      DSN=YOUR.ADMDEFS,DISP=SHR           <optional>
//SYSABEND  DD      SYSOUT=A
```

Figure 4. Suggested job stream for starting the GDDM print utility, ADMOPUT

An explanation of the key values in these job streams follows:

applname
: The VTAM application name used by ADMOPUT. It must be defined by the APPL macro in the VTAM network definition. The name is usually ADMPRINT, but this can be changed if required. The *applname* value is derived as follows:

- For a batch job that does not use a cataloged procedure, *applname* is taken from the job-step name.
- For a batch job that uses a cataloged procedure, *applname* is taken from the procedure-step name.
- For a started task with no task identifier specified on the START command, *applname* is taken from the member name in the procedure library of the started-task JCL.
- For a started task with a task identifier specified on the START command, *applname* is taken from the task identifier.

Include the `applname` value in SYS1.VTAMLST with ACQUIRE authorization. For example: `applname APPL AUTH=(ACQ)`

DYNAMNBR
The DYNAMNBR parameter must be set to a number greater than the maximum number of printers defined by the MAXPRTRS parameter that are to print concurrently using ADMOPUT. This number is not necessarily the same as the number of printers defined in the request queue because, if this is a large number, they are unlikely all to be in use at the same time. If DYNAMNBR is not large enough, ADMOPUT fails. As a recommended value, set DYNAMNBR to 30, or the value you have specified for MAXPRTRS plus one, whichever is the larger.

REGION
The virtual storage requirement of the GDDM TSO print utility depends on the size of your print queue, the maximum number of active printers you specify (using the MAXPRTRS parameter), and the size of your GDDM external defaults module or external defaults file. Because the GDDM default information is loaded in 31-bit mode (if possible), it is usually worth specifying a REGION size greater than 16 MB, to take advantage of this and to avoid running out of storage.

NAME=*name*
Uniquely identifies an instance of ADMOPUT for use in messages to the operator. If the NAME parameter is not specified, the VTAM application name is used in such messages.

AUTO
Indicates that no outstanding reply to the operator is generated, and that ADMOPUT will terminate automatically when there are no further print requests that can be processed. AUTO is optional.

MAXPRTRS=*nnn*
Is the maximum number of printers that the print utility tries to operate concurrently. ADMOPUT processes work for printers that would otherwise be beyond the MAXPRTRS limit after work for other printers has completed.

Use this parameter to limit the maximum number of concurrent print jobs if ADMOPUT is running out of storage or if your network is being slowed down because too many printers are active at one time.

If you do not specify MAXPRTRS, the value 32767 is used.

The `STEPLIB DD` statement specifies the data set on which the GDDM load library resides. This is not required if the library has been included in the link list.

The ADMSYMBL DD statement specifies the data set on which symbol sets reside. If, at the time the print request is made, named symbol sets are identified for use on one or more of the GDDM calls statements GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC, the TSO user requesting the printer must ensure that the symbol sets are in the data set identified by the ADMSYMBL DD statement.

The ADMGGMAP DD statement specifies the data set on which GDDM-IMD-generated mapgroups reside. This statement is required if the user's application programs issue FSCOPY against mapped pages.

The ADMPRNTQ DD identifies the GDDM Master Print Queue data set. The statement is optional. If supplied, it overrides any value specified on the TSOPRNT external default. If you omit the ADMPRNTQ DD statement, and do not set the TSOPRNT external default, the data set name is ADMPRINT.REQUEST.QUEUE by default.

The ADMDEFS DD statement is optional. If supplied, it identifies an external-defaults file. (External-defaults files are described in Appendix A, "External defaults" on page 307.) For the print utility, you are recommended to use a data set as an external-defaults file rather than inline data (that is, not //ADMDEFS DD *). In some operating environments, the implementation of inline data is such that it might be read and acted upon by only the first of the many subtasks that ADMOPUT uses.

Activating printers used by ADMOPUT

Printers can be activated before ADMOPUT is started. Under VTAM, this is achieved by the following command:

```
VARY NET,ACT,ID=printername
```

If a device is activated *after* ADMOPUT is started, ADMOPUT needs to be notified that the device is available. You can do this explicitly by using the LOGON operand when the device is activated, as follows:

```
VARY NET,ACT,ID=printername,LOGON=applname
```

where applname is ADMOPUT's VTAM application name. Alternatively, the VTAM network can be defined such that ADMOPUT is automatically notified when a device is activated. This is done by nominating *applname* as the controlling application on the LOGAPPL parameter on network terminal-definition macros.

Running multiple instances of ADMOPUT

You might want to run multiple instances of ADMOPUT so that you could segregate the output of different groups of users for security reasons, for example. To do this, you must:

- Have a different GDDM Master Print Queue data-set name for each group. The name can be assigned in the job stream that starts each instance of ADMOPUT, or on the TSOPRNT external default. TSOPRNT would have to be specified for each group of users (typically, in an external defaults file) rather than in the external-defaults module for the enterprise.
- Ensure that each instance of ADMOPUT uses a different VTAM applname and that each name is authorized in SYS1.VTAMLST if all instances of ADMOPUT are running in a single domain. This applies even if the instances are on different processors.

Stopping ADMOPUT

If AUTO is not specified in the job stream to start ADMOPUT, this message is received at the system console when ADMOPUT starts:

```
ADM2000 I ADMOPUT(instance-name). TO TERMINATE,
          REPLY 'STOP', 'STOPQ', OR 'STOPS'
```

The system operator can stop ADMOPUT printing by replying STOP, STOPQ, or STOPS to this message. The effect of each of these replies is:

STOP ADMOPUT terminates when all requests in process have been completed.

STOPQ ADMOPUT terminates immediately. Current requests are restarted when ADMOPUT is next initialized.

STOPS GDDM issues message ADM2019, which gives the operator the choice of either

- Entering STOPQ, causing ADMOPUT to stop immediately, or
- Ignoring this message, in which case ADMOPUT continues to run until all requests in process have been completed.

Thus, the utility terminates when VTAM is halted, when a request is made to terminate by a reply to message ADM2000, or if AUTO is specified and there are no more active print subtasks running.

Using the Print Queue Manager (PQM)

The GDDM Print Queue Manager (PQM) is an ISPF panel-driven, stand-alone utility that enables a system administrator or controller to dynamically manage the GDDM Master Print Queue while it is being serviced by the GDDM print utility ADMOPUT. The facilities it offers are:

- List all the print requests for a particular printer.
- List all the printers defined to the Master Print Queue.
- List all print requests on the Master Print Queue and their destination printers.
- List all print requests submitted by a particular user and their destination printers.
- Put a printer into HOLD status so that printing is temporarily suspended to that printer. Requests can still be queued for this printer.
- Add a new printer to the Master Print Queue.
- Delete a printer from the Master Print Queue.
- Delete print requests from the Master Print Queue (this also deletes the print data set).
- Reroute print requests from one printer to another.

The PQM is started by the CLIST ADMPQM, which can be found in the SADMSAM data set. This must be customized for your particular installation. Here is an extract from the CLIST:

```
/*********************************************************************/
/*     Dynamically concatanate PQM libraries: message library        */
/*                                            panel library          */
/*                                            load  library          */
/*********************************************************************/
   ISPEXEC LIBDEF ISPMLIB DATASET ID('GDDM.SADMMSG') UNCOND

   ISPEXEC LIBDEF ISPPLIB DATASET ID('GDDM.SADMPNL') UNCOND

   ISPEXEC LIBDEF ISPLLIB DATASET ID('GDDM.SADMMOD') UNCOND
```

Change the three library names to those created for your installation.

Access to the GDDM Print Queue Manager may represent a security risk at your location because it gives users the ability to redirect output. To restrict the use of this utility, use the RACF RDEFINE command to create a resource called ADM.ADMPQM of class FACILITY, as follows:

```
RDEFINE FACILITY ADM.ADMPQM UACC(NONE)

PERMIT ADM.ADMPQM CLASS(FACILITY) ID(userid)
```

When this has been defined, only users who are authorized to this resource can use the Print Queue Manager.

To dynamically add printers to the Master Print Queue after the Queue has been formatted, spare slots must be formatted into the Master Print Queue. This is done automatically by the format job ADMQFMT. The user can choose the number of these spare slots. When the spare slots have been used, no more printers can be added until the queue is reformatted.

Note: If the queue has to be reformatted, any dynamic printer additions or deletions are undone by running ADMQFMT. You should therefore keep a list of all dynamic printer changes and add them to your ADMQFMT job.

The PQM is an ISPF application, driven through ISPF panels. Each panel has help text associated with it. Panel and message text is available in US English only. There is a full tutorial on the PQM, obtained by entering **T** from the main PQM panel.

How to cancel a job submitted to ADMOPUT

You can cancel a print request without using the Print Queue Manager by deleting the associated temporary print data set (`userid.ADMPRINT.REQUEST.#nnnnn`), provided that the job has not yet started printing. When you do this, ADMOPUT prints a diagnostic message noting that the request was deleted. You can use the TSO LISTC command to identify the names and order of pending print requests for a specific user ID.

Some types of error prevent a request from proceeding to the point where it can be canceled in this way, though they might continue to allow ADMOPUT to retry the request at regular intervals. If this happens, stop the utility, delete the appropriate data set, and restart the utility.

When a request has started to print, it might not be possible to delete the corresponding data set because this would require exclusive (DISP=OLD) access. In this case, the output can be canceled at the printer by switching the printer off

and on at least three times during the printing of a single page. After each power-on, ADMOPUT tries to reprint the interrupted page. The printer must be powered off during the reprinting of this page to maintain the cancelation sequence.

For a printer operating in SCS mode, the PA1/PA2 and CANCEL PRINT switches can be used to cancel a job in progress. These switches allow limited communication with ADMOPUT, and are used with the Hold Print/Enable Print switch, as described in the Component Description and Operator's Guide for the appropriate printer. The effect of each switch is as follows:

PA1 Sending a PA1 switch code to ADMOPUT causes it to restart printing of the current request at the page after the header page. For a multiple-copy request, printing is resumed at the start of the copy being processed at the time of the interrupt.

PA2 Sending a PA2 switch code to ADMOPUT causes it to restart printing of the current page.

CANCEL PRINT Sending a CANCEL PRINT switch code to ADMOPUT causes it to cancel printing of the current request.

Plotter output can be canceled by pressing the CLEAR key on the associated 3179-G, 3192-G, or 3472-G color display station, 3270-PC/G or 3270-PC/GX workstation, or personal-computer system running GDDM-PCLK or GDDM-OS/2 Link.

Printing alphanumeric files

The GDDM Sequential File Print Program ADMOPRT allows files to be printed that contain alphanumeric data. ADMOPRT converts a sequential file into a graphics print file so that ADMOPUT can print it. Alternatively, if a current nickname UDS specifies `TOFAM=4`, ADMOPRT can convert alphanumeric data for output on family-4 devices.

The syntax of the command that calls the GDDM Sequential File Print Program is:

```
CALL 'data-set-name(ADMOPRT)' 'file-name ON
     printer-name [ (NOCC ]'
```

where:

data-set-name
is the name of the data set into which ADMOPRT was installed.

file-name
is either a ddname allocated to the data set to be printed or the actual name of the data set to be printed.

If *file-name* is a data-set name, it must be entered using normal TSO naming conventions. In this case, ADMOPRT does not support a data-set name that represents a member of a partitioned data set.

Note: ADMOPRT uses the values given by the PRINTCTL processing option for the number of characters per line and the number of lines per page.

ON
is a required keyword that must be specified before the name of the printer.

printer-name
is the name of the queued printer on which the file is to be printed.

NOCC
indicates that any existing carriage-control characters are to be ignored. If NOCC is not specified, the presence or absence of carriage-control characters is determined by ADMOPRT according to the record format of the input file.

Messages

Messages issued by ADMOPUT are numbered from ADM2000. These messages are described in the *GDDM Messages* book.

The TSO Print Utility, ADMOPUJ, with JES/328X

ADMOPUJ is a special version of the GDDM Print Utility whose purpose is to provide an interface to the JES/328X Print Facility. JES/328X is a Program Offering that extends the support of Remote Job Entry (RJE) devices provided by MVS JES2 and JES3 to 3270-data-stream printers, including Intelligent Printer Data Stream (IPDS) printers.

Note: Installation and operation of JES/328X is described in the *JES/328X Print Facility Program Description and Operators Manual*, SH20-7174. You are recommended to read this manual carefully before you attempt to alter existing JES/328X definitions to accommodate GDDM.

Print data sets are written directly to the JES spool rather than to a separate, temporary print data set, and no entry is made in the GDDM Master Print Queue. JES directs each print request to JES/328X, which passes it in turn to ADMOPUJ. ADMOPUJ writes the data (alphanumerics or graphics) to the printer device.

This method gives complete control of the GDDM output print data sets to the JES operator for as long as they remain in the JES Spool. JES operator commands can be used to reroute, reorder, and cancel print jobs, for example. JES/328X also provides Interactive System Productivity Facility (ISPF) panels to enable the user to issue some JES2, JES3, or JES/328X commands.

Installing JES/328X

JES/328X is supplied in SMP/E-installable format on standard-label, 9-track tapes. When JES/328X is installed, you need to:

- Define remote workstations to JES

  JES considers JES/328X printers to be remote workstations. Therefore, when JES/328X printers are defined to JES, both a printer and a punch must be specified, and the JES class for the remote punch must be a punch class. GDDM output is directed to the punch. Failure to define the punch causes invalid output.

- Define LOGMODE for use with the printer

  The LOGMODE used by the JES/328X printer, whether the standard DLOGMOD for the printer or the LOGMODE specified on the JES/328X LOGMODE= parameter, *must* be queriable. (For information about defining such LOGMODES, see “Checking a VTAM network” on page 91.) Note that, for IPDS printers, only LU0 and LU1 printing is supported. Also, for printing on

nongraphic printers (such as the 6262), SELECT=BASIC must be included in the JES definition.

If the printer is defined in a VTAM cross-domain environment, the LOGMODE must be defined in the printer-owning domain.

- Define remote workstation to VTAM

  Because JES communicates with JES/328X as a VTAM application, its remote-workstation name must be defined to VTAM as an application.

- Define JES/328X to VTAM

  Because JES/328X uses ADMOPUJ to perform the printing task, ADMOPUJ requires a previously defined VTAM application for all sessions between itself and the printers. JES/328X passes the step name as the name of the ACB.

- Define DSPRINT to VTAM for GDDM

  The JES/328X DSPRINT command allows a user to bypass the JES spool and print direct on the requested printer. You must therefore define either a pool of VTAM applications for use with DSPRINT, or specific names to be used.

- Set up the JES/328X definition

  JES/328X initialization reads a definition file for all the printers it is likely to drive. These definitions relate the JES remote-workstation definitions to the VTAM LU names of the printers, and also specify the type of installation exit to be used. This enables GDDM to take over support of IPDS printers, and also of those printers that will use JES to route GDDM data to them.

  Please note the following:

  – A PASSTHRU value (PASSTHRU=INTERNAL or PASSTHRU=JSXGDDM) must be coded. Use PASSTHRU=INTERNAL for IPDS printers, and PASSTHRU=JSXGDDM for 3270 printers.

  – If used, the LOGMODE= entry must specify the correct logmode. If DLOGMOD is used, the default LOGMODE must be queriable.

  – The LUTYPE= parameter must match the TYPE= parameter specified in the MODEENT for the LOGMODE being used.

- Define a start-up procedure for JES/328X

  DD definitions are required for DD ADMSYMBL and ADMGGMAP. An ADMDEFS DD statement might also be required, depending on specific output applications.

Some examples of print requests directed via JES/328X

The PRINTDST processing option allows you to send a print request either to the JES spool for processing by JES/328X or to a file. When the print request is sent to a file, it can be:

- Transferred to VM for processing there
- Sent to JES by means of either the JES/328X DSPRINT command or JCL for JES/328X processing
- Sent directly to the printer by means of the JES/328X DSPRINT command.

The following examples illustrate some of the ways in which JES/328X and ADMOPUJ can be used to send output to printers. They assume that:

- JES/328X has been installed and is running.
- The JES/328X DSPRINT command has superseded the TSO DSPRINT command.
- Remote destinations RMT1 and RMT2 have been defined to JES with the printers serving Class P and the punches serving Class G.
- Remote destinations RMT1 and RMT2 have been defined to VTAM.
- Remote destinations RMT1 and RMT2 have been defined to JES/328X, as follows:
  - RMT1 is a 3287 device with an LUNAME of L870. GDDM is called to process all requests for CLASS G.
    (PASSTHRU=JSXGDDM and CLASS=G specified.)
  - RMT2 is an IPDS device with an LUNAME of L890. (Data destined for IPDS devices automatically causes GDDM to be invoked, provided PASSTHRU=INTERNAL is specified.)
- The TSO user's external-defaults file or module contains these nickname statements:

```
NICKNAME NAME=R187,FAM=2,
      PROCOPT=((PRINTDST,G,RMT1))
NICKNAME NAME=R2IP,FAM=2,DEVTOK=S4224SE,
      PROCOPT=((PRINTDST,G,RMT2))
NICKNAME NAME=DA87,FAM=2,
      PROCOPT=((PRINTDST,*,DD87))
NICKNAME NAME=DAIP,FAM=2,DEVTOK=S4224SE,
      PROCOPT=((PRINTDST,*,DDIP))
```

- Invocations of the Interactive Chart Utility use these nickname statements, and have DDNAMES DD87 and DDIP allocated to data sets USER.GDDMPRT.DATA and USER.IPDSPRT.DATA respectively.
- The user has a data set USER.PRINT.DATA containing alphanumeric data. (Specifically, the data set does not have carriage-control set.)

Example 1: The GDDM-PGF Interactive Chart Utility (ICU) is invoked and a print is created for R187. This sends the print request (formatted for a 3287) directly into the JES Spool, from where it is passed to JES/328X. JES/328X, in turn, passes the request to ADMOPUJ for printing.

Example 2: The ICU is invoked and a print is created for R2IP. This sends the print request (formatted for an IPDS printer) directly into the JES spool, from where it is passed to JES/328X. JES/328X, in turn, passes the request to ADMOPUJ for printing.

Example 3: The ICU is invoked and a print is created for DA87. This outputs the print request (formatted for a 3287) to the data set USER.GDDMPRT.DATA.

Example 4: The ICU is invoked and a print is created for DAIP. This outputs the print request (formatted for an IPDS printer) to the data set USER.IPDSPRT.DATA.

In examples 1 and 2, the print requests are sent automatically to ADMOPUJ for printing. In examples 3 and 4, the user can:

- Transfer the files to VM for processing.

Because the format of the print request is subsystem-independent, the print request can be processed on either VM or TSO.

- Submit the files to JES for printing using JES/328X and ADMOPUJ by issuing the following commands:

  ```
  DSPRINT 'USER.GDDMPRT.DATA' RMT1 CLASS(G) NONUM
  DSPRINT 'USER.IPDSPRT.DATA' RMT2 CLASS(G) NONUM
  ```

  These commands result in message DSP012, indicating that the print requests have been sent to JES for subsequent processing.

- Submit the files directly to JES/328X, bypassing the JES spool.

  This facility is particularly useful where data of a confidential nature is being printed and the user does not want the print request to be sent by way of the "public" spool facility. Typically, the user is not far from the printer and can ensure that the print is collected as soon as it is finished.

  The following commands do this:

  ```
  DSPRINT 'USER.GDDMPRT.DATA' L870 GDDM
  DSPRINT 'USER.IPDSPRT.DATA' L890 GDDM
  ```

  These commands result in a foreground print operation, the end of which is signaled by message DSP040. Because this is a foreground process, the user's terminal is "locked out" for the duration of the print request. Also, any symbol sets required for printing have to be allocated by the user before the JES/328X DSPRINT command is issued.

Changing the program that processes your spooled print output

If you want to send a print file to the JES spool to be printed by an external program, specify the WRITER parameter of the PRINTDST processing option. If you have an external writer program called MYWTR, you can place a nickname statement such as the following in the user defaults module:

```
ADMMNICK  FAM=2,PROCOPT=((PRINTDST,G,,MYWTR)),
          NAME=R31P,DEVTOK=S4224SE
```

Whenever users create a print file for R31P, the print request goes to the JES spool where it is processed by MYWTR.

Selecting different paper types for your spooled print output

If you want to have different types of paper loaded in different printers at your location or if you want to delay the printing of spooled output until suitable paper is loaded in the printer, you can define different FORMS types in JES. To take advantage of this, specify the FORMS parameter of the PRINTDST processing option. The following example contains two nickname statements that a user could place in the user defaults file to specify the JES FORMS and WRITER parameters:

```
ADMMNICK  FAM=2,PROCOPT=((PRINTDST,A,,MYWTR,A4)),
          NAME=R31P,DEVTOK=S4224SE,
          DESC="Print file spooled to MYWTR & queued to 4224 with A4.  "

ADMMNICK  FAM=2,PROCOPT=((PRINTDST,A,,GENWTR,STANDARD)),
          NAME=R31F,DEVTOK=S4224SE,
          DESC="Print file spooled to GENWTR & queued to 4224 with std. fanfold."
```

Whenever users create a print file for R31P, the print request goes to the JES spool where it is processed by MYWTR. It then is queued to the printer with A4 paper loaded.

If a user creates a print file for R31F, the print request is processed in exactly the same way but is spooled to a printer loaded with standard fanfold paper. If there is only one printer the output for R31F remains in the queue until the paper is changed to standard fanfold.

Printing alphanumeric files via JES/328X

There are two ways of printing files containing alphanumeric data:

1. You can issue the following commands to submit the files to JES for printing via JES/328X (and GDDM if the printer is an IPDS device):

   ```
   DSPRINT 'USER.PRINT.DATA' RMT1 CLASS(P) NONUM
   DSPRINT 'USER.PRINT.DATA' RMT2 CLASS(P) NONUM
   ```

   Message DSP012 confirms that the print requests have been sent to JES for processing.

2. You can issue the following commands to submit the files directly to JES/328X:

   ```
   DSPRINT 'USER.PRINT.DATA' L870 GDDM NONUM
   DSPRINT 'USER.PRINT.DATA' L890 GDDM NONUM
   ```

   These commands result in a foreground print operation, the end of which is signaled by message DSP040.

The message ADM0244 E INVALID PRINT RECORD SEQUENCE appears on the print request if GDDM has been called to process invalid data. This happens if you send alphanumeric data via JES to JES/328X using the class defined for GDDM files.

```
DSPRINT 'USER.PRINT.DATA' RMT1 CLASS(G) NONUM
```

The above command sends the alphanumeric data through JES to JES/328X as CLASS G output. JES/328X requires that GDDM data has carriage-control set on, and ignores all records without carriage-control. If this should happen, the message `JSX209 - NON-GRAPHICS RECORDS IGNORED` is sent to the operator console and an empty file is sent to GDDM. GDDM then sends message ADM0244 to the printer.

The interface between GDDM and JES

GDDM uses Dynamic Allocation to send a print request directly to the JES Spool with the CLASS and DEST parameters from the PRINTDST processing option “class” and “destname” values respectively.

This is the one case where GDDM creates a print request without carriage-control. This allows the data to be passed to GDDM correctly. However, when sending the print request to a data set, GDDM sets the carriage-control indicator on in the data set, even if it is a preallocated data set. DSPRINT accommodates this automatically, but if the user wants to route the data through JES by some other means, such as an IEBGENER process, the RECFM needs to be overridden:

```
//*
//*  ** SEND GDDM DATA THROUGH JES TO JES/328X
//*
//GENER1   EXEC PGM=IEBGENER
//SYSPRINT DD  SYSOUT=A
//SYSUT2   DD  SYSOUT=G,DEST=RMT1,
//             DCB=(RECFM=FB,LRECL=80,BLKSIZE=1600)
//SYSUT1   DD  DSN=USER.GDDMPRT.DATA,DISP=SHR
//SYSIN    DD  DUMMY
```

jesgddm.The interface between JES/328X and GDDM

ADMOPUJ is called from the JES/328X special exit JSXGDDM, either to process a print request or, at initialization and termination time, to OPEN and CLOSE the VTAM ACB.

JSXGDDM passes this parameter list, the address of which is in Register 1:

`JXEACBN` address of an 8-byte area containing the VTAM ACB name.

`JXEDEST` address of an 8-byte area containing the VTAM LU name of the printer destination.

`JXEDSN` address of a 44-byte area containing the ddname of the GDDM input print file. The JXEDSN ddname has two special values, OPEN and CLOSE, which identify initialization and termination requests respectively.

`JXECOMM` address of a 4096 common area. The common area is used as a work area by GDDM.

`JXESHUT` address of the full-word JES Shutdown ECB. The JES shutdown ECB enables GDDM to detect a shutdown request.

`JXEMSG` address of 160-byte message area. The message area enables GDDM to inform JES/328X of any errors detected in printing.

Chapter 4. The GDDM/VSE Batch Print Utility, ADMUPRTC

Family-4 output (that is, documents or page segments) can be printed on advanced-function printers, such as the IBM 3820 or 3900, and on the IBM 4250 printer in the VSE/Batch environment.

This chapter provides:

- A brief explanation of the ways in which documents and page segments are generated for printing on advanced-function printers and on the IBM 4250
- A description of the GDDM/VSE print-job utility ADMUPRTC, and of the ways in which you can tailor it.

Printing documents under VSE/Batch

There are two ways of printing documents under VSE/Batch:

- You can run an application program that produces family-4 output.

  If the output is destined for an advanced-function printer, when the program issues an output statement (such as FSFRCE) the print file is sent to VSE/POWER spool space.

  If the output is destined for a 4250 printer, an output statement such as FSFRCE causes the print file to be sent to a VSAM variable-length (SAM) file named on the `namelist` specified on input to DSOPEN.

- You can use the VSE print utility ADMUPRTC to print:
  - ICU chart-format (ADMCFORM) files
  - ICU chart-data (ADMCDATA) files
  - GDDM graphics-data-format (ADMGDF) files
  - GDDM image (ADMIMG) files.

  These files must be stored in the VSAM ADMF data set, or in a private VSAM data set. (Private VSAM data sets are described in Chapter 6, “GDDM objects and other files” on page 47.) ADMUPRTC, for which the default CICS transaction name is ADMB, prompts the user for more information, such as the name of the file to be printed, the number of copies, and accounting information.[5] You can change the transaction name of ADMUPRTC by setting the CICPRNT external default. CICPRNT is described in Appendix A, “External defaults” on page 307.

  What happens next depends on whether the output is destined for an advanced-function printer or for an IBM 4250 printer.

  If the output is destined for an advanced-function printer, ADMUPRTC invokes the ADMUCDSD print program to submit a batch job that sends the print file to VSE/POWER spool space. PSF, running in another VSE partition, takes the file from VSE/POWER spool space and prints it on an advanced-function printer.

  If the output is destined for a 4250 printer, ADMUPRTC submits a batch job that sends the print file to a VSAM variable-length (SAM) file. When the print

[5] ADMUPRTC uses the CICS Report Controller Feature

file has been generated, CDPF must be run as a separate job to print the file on the 4250 printer.

Customizing the printing process for documents under VSE/Batch

As an alternative to ADMUPRTC, you can provide your own JCL to send documents to VSE/POWER spool space. Figure 5 shows some example JCL that submits a primary data stream (a document) for output on an advanced-function printer. You can also provide some JCL to invoke CDPF to send documents to a 4250 printer. Figure 6 on page 33 shows some example JCL that performs this task.

Figure 5. VSE JCL for submitting documents to VSE/POWER spool space. The numbers at the beginning of each record identify comments following the JCL.

```
1)  * $$ JOB JNM=jobname,CLASS=x,DISP=y
2)  * $$ LST CLASS=x,DISP=y,DEST=(node,userid),JSEP=1
3)  * $$ LST CLASS=x,DISP=y,DEST=(,psfid),LST=cuu,JSEP=1
4)  // JOB jobname
5)  // DLBL libname,'name.of.a.library'
5)  // EXTENT ,volid
6)  // LIBDEF *,SEARCH=(l1.s1,l2.s2,...,li.si)
7)  // DLBL IJSYSUC,'user.catalog.name',,VSAM
8)  // DLBL ADMF,'gddm.objects.file.name',,VSAM
9)  // ASSGN SYSyyy,cuu
10) // EXEC ADMUCDSD,SIZE=ADMUCDSD,                                    *
        PARM='filename filename 99 4 token (procopts) (SYSyyy)'
    /*
    /&
    * $$ EOJ
```

1. The POWER JOB statement for the GDDM batch job.
2. The POWER LST statement for the SYSLST output.
3. The POWER LST statement for the primary data stream destined for the advanced-function printer at address `cuu`.
4. Job name card.
5. DLBL and EXTENT statements for every library named in the LIBDEF statement. If the library is not VSAM controlled, use EXTENT only.
6. Search chain for all libraries to be read from.
7. DLBL for the user catalog (which contains ADMF).
8. DLBL for GDDM objects file.
9. Assignment statement linking the programmer logical unit `SYSyyy` to the device `cuu`, the spooled printer.
10. EXEC card for ADMUCDSD.

Figure 6. VSE JCL for submitting documents to 4250 printers via CDPF. The numbers at the beginning of each record refer to the notes following the JCL.

```
1)  * $$ JOB JNM=jobname,CLASS=x,DISP=y
2)  * $$ LST CLASS=x,DISP=y,DEST=(node,userid),   *
          JSEP=1
3)  // JOB jobname
4)  // DLBL lib1,'cdpf.library.name'
4)  // EXTENT ,volid1
4)  // DLBL lib2,'font.library.name'
4)  // EXTENT ,volid2
5)  // LIBDEF *,SEARCH=(lib1.sublib1,lib2.sublib2)
6)  // DLBL usercat,'user.catalog.name',,VSAM
7)  // DLBL INPUT,'print.file.name',,VSAM,        *
          CAT=usercat,DISP=(OLD,DELETE)
8)  // EXEC BFUCDPF,SIZE=AUTO,                    *
          PARM='PRINT (BRACKET vtam_printer_name)'
    /*
    /&
    * $$ EOJ
```

1. The POWER JOB statement for the GDDM batch job.
2. The POWER LST statement for the SYSLST output.
3. Job name card.
4. DLBL and EXTENT statements to define the libraries containing CDPF and the FONTLIB.
5. Search chain for the libraries containing CDPF and the fonts.
6. DLBL statement for the user catalog.
7. DLBL statement for the print file.
8. EXEC card for CDPF.

Note the DELETE option on the DLBL statement for the print file, which causes the input file to be deleted after it is printed. Use the option KEEP if you want to keep the print file.

Printing page segments under VSE/Batch

Unlike documents, page segments are directed to a VSE phase library rather than to a VSAM data set or VSE/POWER spool space. Though they can be submitted for printing using the ADMUPRTC utility (invoked via the ADMB transaction), you need to supply some JCL for the batch job started by ADMUPRTC. In particular, you need to identify the VSE phase library in which page segments are stored. The JCL for the batch job started by ADMUPRTC is held in the CICS external-defaults file, ADMADFC. Instructions for updating it can be found in “Tailoring the batch job that ADMUPRTC initiates” on page 34. To identify the phase library, you need to add a statement like this to ADMADFC:

```
// EXEC ADMUCDSD PARM='fn fn 99 4 token (procopts) (phase-library)'
```

The device token indicates the type of page segment (for a 3800 or 4250) to be generated.

You can include the page segment in a file that is formatted using a suitable program, and print it using either PSF or CDPF.

Note that there is a limit of 40K bytes on the size of a primary data stream for input to CDPF. If you are formatting a page segment for the 4250 printer, you should run the supplied ADMUP2VD program to copy the page segment from the phase library to a VSAM entry-sequenced data set (ESDS).

Here is an example job to copy the page segment P38SEG01 to the VSAM ESDS file P38SEG01.MYLIB.ESDS.

```
* $$ JOB JNM=ADMUP2V0,CLASS=0,DISP=D
* $$ LST CLASS=x,DISP=y,DEST=(node,userid),JSEP=1
// JOB ADMUP2V0
// LIBDEF *,SEARCH=(VSEDEVT.WORKLIB)
// DLBL IJSYSUC,'GDEV2.UCAT',,VSAM
// DLBL P38SEG0,'P38SEG01.MYLIB.ESDS',,VSAM
// EXEC IDCAMS,SIZE=AUTO
 DELETE (P38SEG01.MYLIB.ESDS)   -
        CLUSTER
 DEFINE CLUSTER              -
          (NAME(P38SEG01.MYLIB.ESDS)   -
           NONINDEXED                  -
           RECORDFORMAT(V)             -
           RECORDSIZE(4000 8202)       -
           TRACKS(5 5)                 -
           VOL(PAC371))                -
           DATA                        -
           ( NAME(P38SEG01.MYLIB.DATA) )
/*
// EXEC ADMUP2VD,SIZE=ADMUP2VD,PARM='P38SEG01'
/*
/&
* $$ EOJ
```

Tailoring the batch job that ADMUPRTC initiates

The JCL for the batch job initiated by ADMUPRTC is stored as an extension to the CICS external-defaults file, ADMADFC, which is described in Chapter 17, “GDDM user-default specifications” on page 197.

Figure 7 on page 35 shows an example of the JCL in the ADMADFC external-defaults file. Some explanatory notes follow the figure.

```
ADMADFC  CSECT
         ADMMDFT START       normal defaults start
         ADMMDFT . . .
         ADMMDFT DFTXTNA=VSEJCLA new extension default
         ADMMDFT . . .
         ADMMDFT END         normal defaults end
*
*********************************************************************
*   JCL FOR PRINTING ON 3800
*
*   REPLACE THE FOLLOWING SYMBOLIC NAMES:
*     PR3800      IS THE PARM TO BE SUPPLIED TO PSF.
*     VSAMUSERCAT IS THE VSAM USER CATALOG FOR THE GDDM OBJECTS FILE
*     VSAMVOL     IS THE VOLUME WHERE TEMPORARY FILES SHOULD BE CREATED
*     PRINTLOG    IS THE USERID TO RECEIVE THE JOB SYSLST OUTPUTS
*********************************************************************
VSEJCLA  ADMMDFTX TYPE='VSEJCL  ',NAME='3800    '
         ADMMDFTX DATA='* $$ JOB JNM=%JOBNAME%,CLASS=0,DISP=D'
         ADMMDFTX DATA='* $$ LST CLASS=A,DISP=D,DEST=(,PRINTLOG),JSEP=1C
               ,RBS=500'
         ADMMDFTX DATA='* $$ LST CLASS=A,DISP=D,DEST=(,PR3800),LST=1A0,C
               JSEP=2,COPY=%COPIES%'
         ADMMDFTX DATA='// JOB %JOBNAME% %ACCOUNT%'
*
*    DEFINE LOCATION OF ADMF
*
         ADMMDFTX DATA='// DLBL IJSYSUC,''vsamusercat'',,VSAM'
         ADMMDFTX DATA='// DLBL ADMF,''ADMF'',,VSAM'
*
*    DELETE SPILL FILE
*
         ADMMDFTX DATA='// DLBL ADM0001,''%JOBNAME%.SPILL.FILE'',,VSAM'
         ADMMDFTX DATA='// EXEC IDCAMS,SIZE=AUTO'
         ADMMDFTX DATA=' DELETE (%JOBNAME%.SPILL.FILE) CLUSTER'
         ADMMDFTX DATA='/*'
*
*    CREATE SPILL FILE
*
         ADMMDFTX DATA='// EXEC IDCAMS,SIZE=AUTO'
         ADMMDFTX DATA=' DEFINE CLUSTER -'
         ADMMDFTX DATA='   (NAME(%JOBNAME%.SPILL.FILE) -'
         ADMMDFTX DATA='    NONINDEXED RECORDSIZE(1000 2000) -'
         ADMMDFTX DATA='    REUSE -'
         ADMMDFTX DATA='    RECORDS(400 400) VOLUME(VSAMVOL))'
         ADMMDFTX DATA='/*'
         ADMMDFTX DATA='IF $RC > 4 THEN'
         ADMMDFTX DATA='GOTO $EOJ'
```

Figure 7 (Part 1 of 2). Example of the extension to ADMADFC

```
*
*     INVOKE GDDM TO PRINT OUTPUT
*
         ADMMDFTX DATA='// ASSGN SYS020,1A0'
         ADMMDFTX DATA='// EXEC ADMUCDSD,SIZE=ADMUCDSD,PARM=''%DATANAMEC
               % %FORMAT% %CODES% * () (%NICKNAME%)'''
         ADMMDFTX DATA=' ADMMNICK FAM=4,PROCOPT=((HRISWATH,%SWATHES%))'
         ADMMDFTX DATA=' ADMMNICK FAM=4,PROCOPT=((COLORMAS,6))'
         ADMMDFTX DATA=' ADMMNICK FAM=4,DEVTOK=%NICKNAME%,TONAME=SYS020C
               '
         ADMMDFTX DATA='/*'
         ADMMDFTX DATA='/&&'
*
*     DELETE SPILL FILE CREATED BY GDDM TO CONSERVE SPACE
*
         ADMMDFTX DATA='// JOB %JOBNAME% %ACCOUNT%'
         ADMMDFTX DATA='// DLBL IJSYSUC,''vsamusercat'',,VSAM'
         ADMMDFTX DATA='// EXEC IDCAMS,SIZE=AUTO'
         ADMMDFTX DATA=' DELETE (%JOBNAME%.SPILL.FILE) CLUSTER'
         ADMMDFTX DATA='/*'
         ADMMDFTX DATA='/&&'
         ADMMDFTX DATA='* $$ EOJ'
*
* ... and so on for other printing environments
*
         ADMMDFTX END
```

Figure 7 (Part 2 of 2). Example of the extension to ADMADFC

Notes:

1. The external default DFTXTNA specifies the label of the extension to ADMADFC. In this example, the label is `VSEJCLA`.

   The ADMMDFTX macro is used to define the JCL to be used for batch printing. All ADMMDFTX macros must follow the ADMMDFT END macro in the external-defaults module, ADMADFC. The special form ADMMDFTX END must appear as the last ADMMDFTX macro in the sequence.

   The syntax of ADMMDFTX is as follows:

   ```
   name  ADMMDFTX TYPE=,           VSEJCL  |VSEJCLIM      *
                 NAME=,           3800 |4250             *
                 DATA=            lines of JCL
   ```

   name
   The *name* field of the first ADMMDFTX macro in the external-defaults module is used in the ADMMDFT DFTXTNA=name macro. For other ADMMDFTX macros, it has no significance.

   TYPE=VSEJCL|VSEJCLIM
   Identifies the external-defaults extension area as containing the JCL for the VSE batch printing job.

   NAME=3800|4250
   Identifies the type of device on which printing is to be done.

   DATA=*data*
   Data statements that define the JCL to be used for batch printing.

2. Following label `VSEJCLA` are some lines of JCL included by ADMMDFTX macros with the keyword DATA. The lines are grouped, with each group introduced by a heading giving a name and a type. The type value can be one of these two:

   VSEJCL Identifies JCL for printing ADMCFORM, ADMCDATA, or ADMGDF files

   VSEJCLIM Identifies JCL for ADMIMG files

   The example shows one group of type VSEJCL with the name "3800." ADMUPRTC accesses the JCL by name and type.

3. Within the DATA operand there are modifiable fields. These are identifiers enclosed between two % characters.

   These identifiers are used by ADMUPRTC:

   JOBNAME Job name (8 characters)

   ACCOUNT Accounting information (16 characters)

   DATANAME Chart-data name (8 characters)

   FORMAT Chart-format name (8 characters)

   CODES Options for ADMUCDSD (3–4 characters)

   NICKNAME Device nickname (8 characters)

   JOBTYPE Skeleton selected (8 characters)

   SWATHES Number of swathes (1–2 characters)

   COPIES Number of copies (1–2 characters).

   ADMUPRTC recognizes these fields by their identifiers, so you must not change them. You can remove an identifier and its "%" marks altogether, thereby making the JCL nonmodifiable. If you need to insert a % in the JCL generated, code it as two % marks.

4. The parameter list for the job stream is restricted to 100 characters after substitution. A nickname is used to refer to the device. The definition of the nickname can be given in ADMADFD, the VSE/Batch external-defaults file. If the user needs detailed control of the processing options, these can be defined as inline data, which will be in the job stream. This file is read as SYSIPT and acts to GDDM as the external defaults file.

5. Because the batch job might not run until long after the JCL is submitted, there is no feedback to the user of successful job completion. Messages about errors during printing are sent to the batch console log.

Instructions for replacing ADMADFC when you have edited it are in Chapter 19, "Updating GDDM default modules" on page 213.

Tracing

If you enable GDDM tracing for batch printing, you need to define an entry-sequenced data set (ESDS) to receive trace records. Here is some sample JCL for defining an ESDS:

```
// DLBL MYTRACE,'MYTRACE',,VSAM,CAT=USERCAT
// EXEC IDCAMS,SIZE=AUTO
 DEFINE CLUSTER NAME(MYTRACE) -
        NONINDEXED -
        RECORDSIZE(66 132) -
        TRACKS(1 1) -
        VOL(xxxxxx)
```

In this example, the name of the ESDS is “MYTRACE.” The GDDM-supplied default ESDS name is ADMTRCE. You use the VSETRCE external default to specify a different name. VSETRCE is described in Appendix A, “External defaults” on page 307.

In the batch job started by ADMUPRTC, you would need to include a statement naming the ESDS. For example:

```
ADMMDFTX DATA=' ADMMDFT VSETRCE=MYTRACE'
```

or for other inline job streams:

```
ADMMDFT VSETRCE=MYTRACE
```

To print the trace data set, use AMS REPRO statements to copy it to SYSLST. A suitable job would be based on these statements:

```
// DLBL MYTRACE,'MYTRACE',,VSAM,CAT=USERCAT
// EXEC IDCAMS,SIZE=AUTO
   REPRO INFILE(MYTRACE) OUTFILE(SYSLST)
```

Editing the ADMUPRTC panels

GDDM supplies the source of the ADMUPRTC panels as Z-type sublibrary members called ADMUPx, where x is the National Language letter. These can be edited using GDDM-IMD by first loading them into the ADMX staging file and then invoking GDDM-IMD to import the files into an MSL.

Chapter 5. Printing and viewing composite documents

A composite document contains any combination of formatted alphanumeric text, graphics, and image. A LIST3820 file in the VM environment, for example, is a composite document. A composite document can be in one of two formats:

- Advanced Function Printing Data Stream (AFPDS)
- Composite Document Presentation Data Stream (CDPDS).

GDDM provides a *composite document print utility* (CDPU) that supports both display and printing of composite documents in AFPDS and CDPDS formats.[6]

The CDPU can be invoked:

1. From a locally written application, by means of the CDPU call. The CDPU call is described in the *GDDM Base Application Programming Reference* book.
2. From the GDDM-supplied program ADM4CDU*x* (where *x* is subsystem-dependent).

This chapter describes how the supplied program ADM4CDUx is invoked so that you can make it available to the users in your enterprise. Use of the CDPU is described in the *GDDM User's Guide*.

How to invoke ADM4CDUx

ADM4CDUx can be invoked:

- To print AFPDS and CDPDS documents, page segments, and overlays under CMS, TSO, CICS, MVS batch, and VSE batch
- To display AFPDS and CDPDS documents, page segments, and overlays under CMS, TSO, and CICS

However, you need to supply any necessary commands or JCL to invoke ADM4CDUx, and you need to make those instructions available to your users. Two sample procedures, both of which call ADM4CDUx to browse a file at the terminal, are provided with GDDM 3.1. These are:

- ADMUBCDV, a CMS REXX-language procedure
- ADMUBCDT, a TSO command list (CLIST).

Other CMS example procedures are shown in Figure 8 on page 41 and Figure 9 on page 43. The first of these two examples sends a document to a 4224 printer, and the second sends it to a 38xx AFPDS printer. The first is similar to the supplied sample ADMUBCDV, except that the device token and namelist variables are set to printer values.

---

[6] AFPDS is a Print Services Facility (PSF) data stream and is defined in the publication *AFPDS Data Stream Reference*, S544-3202. GDDM supports the AFPDS enhancements provided by PSF Version 2.1 in the VM and MVS environments. These enhancements include support for GOCA graphics orders and IO compressed image.

For a description of the structure of a CDPDS document, and for a list of the AFPDS structured fields supported by the CDPU, see the *GDDM Base Application Programming Reference* book.

Values passed to ADM4CDUx

The syntax of any call to ADM4CDUx is as follows:

```
ADM4CDUx (group-1) (group-2) (group-3) (group-4) (group-5) (group-6)
```

where *x* is C (for CICS), D (for VSE Batch), T (for MVS Batch and TSO), or V (for VM/CMS).

In the CICS environment, the default transaction identifier is ADM4. The parameter list is specified on the *from* option of the CICS START call.

In all environments, the parameter list comprises six groups of data separated by a blank or a comma. All groups are optional, and most have default values. Note that the groups must be specified in the order described below.

The parameter groups are:

group-1
An array of 8-byte character tokens specifying the name of the CDPDS or AFPDS document, page segment, or overlay:

CICS	temporary queue name
VSE Batch	DLBL file name (7 characters)
TSO	DD name
CMS	*filename* (*filetype*|**LISTCDP**) (*filemode*|⋆)

Each name part is left-justified.

group-2
An array of fullword integers specifying printing options. Options not supported by the target printer are ignored. The elements of the array are:

1. Number of collated copies of document. This option applies only to family-1 IPDS printers. The default value is one copy.
2. Duplex control:

   1 Simplex (the default)
   2 Normal duplex
   3 Tumble duplex
3. View control

   Applies only when the document is viewed.

0	View the entire document (the default)
+n	Draw page n, with images included
-n	Draw page n, with images indicated by boxes

   If +n or -n is specified, the CDPU creates a GDDM page containing the required output, but leaves it to the invoking application program to display the page.
4. Deletion control

   0 Same as 1 (the default)
   1 The input file is not deleted
   2 The input file is deleted
   3 The input file is deleted only if processing has completed successfully
   4 The input file is deleted only if processing has not been completed successfully.

When the CDPU call is included in a locally written application, document copies are uncollated. They are collated when printing is requested via ADM4CDUx.

group-3
Device name list.

Possible values are the same as those of the name-list parameter of the DSOPEN call. (Name lists are described in Appendix D, “Name-lists” on page 383.) There is no default value, unless the program is running under CMS and family-4 is specified (or defaulted) in parameter 6. In that case, the default name is the same as the input file name.

group-4
Device processing options (procopts), specified as a sequence of processing-option group codes and numeric processing-option group codes. These are described in Appendix B, “Processing options” on page 333. The parameters of the processing options must be specified in encoded form. Processing options with parameters that can be specified only in character form, such as HRIDOCNM or POSTPROC, cannot be specified for this utility. By default, no processing options are specified.

group-5
Device token.

The default value is S4224QE (SNA-attached 4224) under CICS, and A4 (38xx AFPDS printer) under other systems.

group-6
Device family.

Identifies the device family of the target device (1, 2, 3, or 4). The default is 1 under CICS, and 4 under the other subsystems.

For example, a simple parameter list to print a CDPDS file called DOC, on a 38xx page printer defined by the device token IMG240, using 1 swathe, would be:

```
(DOC) ( ) ( ) (7 1) (IMG240)
```

Invoking ADM4CDUx — example for 4224 printer

```
/* Name : CD42SAMP - example exec  (4224 printer)                    */
/* This is an example user exec that takes a Composite Document      */
/* Presentation Data Stream (CDPDS) file and prints the document on  */
/* a 4224 printer.                                                   */

 Arg fn ft fm .
/* Check invocation parameters                                       */
 If fn = '?' | fn = ''             /* If parameters are incorrect    */
    then signal prompt             /* prompt user .....              */
/* Substitute default value for filetype & filemode if non-specified */
 Parse Value ft "LISTCDP"  With ft . /* I/P filetype                 */
 Parse Value fm "*"        With fm . /* I/P filemode                 */
```

Figure 8 (Part 1 of 2). REXX procedure for printing a composite document on a 4224 printer under CMS

```
/* Set default parameters for ADM4CDUV                              */
 copies   = "1"                   /* number of copies              */
 duplex   = "1"                   /* 1 = simplex                   */
                                  /* 2 = normal duplex             */
                                  /* 3 = tumble duplex             */
 procopts = ""                    /* Processing options            */
 devtok   = "X4224QE"             /* Device token                  */
 family   = "1"                   /* GDDM Family                   */
 namelist = "061"                 /* Namelist entry (print address) */

/* Check that the specified CDPDS file exists                       */
 address command 'STATE' fn ft fm  /* Look for specified file      */
 If rc ¬= 0 then                  /* If not, issue error message   */
    do;                           /* and exit with CMS return code */
      say                         /*                               */
      say fn ft fm 'NOT FOUND'    /*                               */
      say                         /*                               */
      exit rc                     /*                               */
    end;                          /*                               */

/* Start the GDDM Composite Document Print Utility                  */
 address command 'ADM4CDUV ' fn ft fm '(' copies duplex ')',
           '(' namelist ')(' procopts ')(' devtok ')(' family ')'
 exit rc

/* Provide a description of the invocation parameters for this exec  */
prompt : parse source . . execname .
say 'This exec reads a Composite Document Presentation Data Stream (CDPDS)'
say 'file and prints the composite document on a 4224 printer.            '
say '                                                                     '
say '   Format :-                                                         '
say '                                                                     '
say '          'execname' filename filetype filemode                      '
say '                                                                     '
say '          where filename is the input filename                       '
say '                filetype is the input filetype                       '
say '                filemode is the input filemode                       '
say '                                                                     '
exit 0
```

Figure 8 (Part 2 of 2). REXX procedure for printing a composite document on a 4224 printer under CMS

Invoking ADM4CDUx — example for 38xx AFPDS printer

```
/* Name : CD38SAMP - example exec  (38xx AFPDS printer)              */
/* This is an example user exec that takes a Composite Document      */
/* Presentation Data Stream (CDPDS) file and creates a LIST38PP      */
/* file for printing by a 38xx AFPDS printer.                        */

 Arg fn ft fm .

/* Check invocation parameters                                       */
 If fn = '?' | fn = ''             /* If parameters are incorrect    */
    then signal prompt             /* prompt user .....              */
/* Substitute default value for filetype & filemode if non-specified */
 Parse Value ft "LISTCDP"  With ft . /* I/P filetype                 */
 Parse Value fm "*"        With fm . /* I/P filemode                 */
/* Set default parameters for ADM4CDUV                               */
 copies   = "1"                    /* number of copies               */
 duplex   = "1"                    /* 1 = simplex                    */
                                   /* 2 = normal duplex              */
                                   /* 3 = tumble duplex              */
 procopts = "9 1 7 20"             /* GDDM processing options        */
                                   /*  9 = 1 Formatted output        */
                                   /*  7 = 20 Swathes                */
                                   /* 32 = 0 no inline resources     */
 devtok   = "IMG240"               /* Device token                   */
 family   = "4"                    /* GDDM Family                    */
 postproc = "PRT3812"              /* Post processing                */
                                   /* PRT3812 - print on 3812        */
                                   /* PSF     - print on 3800-3      */
 outfn    = fn                     /* O/P filename                   */
 outft    = "LIST38PP"             /* O/P filetype                   */
 namelist = outfn outft            /* output filename                */

/* Check that the specified CDPDS file exists                        */
 address command 'STATE' fn ft fm  /* Look for specified file        */
 If rc ¬= 0 then                   /* If not, issue error message    */
    do;                            /* and exit with CMS return code  */
      say                          /*                                */
      say fn ft fm 'NOT FOUND'     /*                                */
      say                          /*                                */
      exit rc                      /*                                */
    end;                           /*                                */

/* Erase the output file if it already exists                        */
 address command 'STATE' outfn outft 'A'
 If rc = 0 then
   address command 'ERASE' outfn outft 'A'
/* Start the GDDM Composite Document Print Utility                   */
 address command 'ADM4CDUV ' fn ft fm '(' copies duplex ')',
           '(' namelist ')(' procopts ')(' devtok ')(' family ')'
```

Figure 9 (Part 1 of 2). REXX procedure for printing a composite document on a 38xx AFPDS printer under CMS

```
/* Post processing                                                  */
 Select
                                  /*Invoke PRT3812 to print the file*/
   When postproc = 'PRT3812' then
   'PRT3812 ' namelist ' ( COPIES ' copies

                                  /* Invoke PSF to print the file   */
   When postproc = 'PSF'     then
     Do
       'SPOOL PRINTER CLASS B FORM PAGEQUAR NOHOLD'
       'PSF'  namelist ' ( COPY ' copies

/* If procopt 32 = 1 use the following line instead                */
/*     'PSF'  namelist ' ( COPY ' copies 'FORMDEF (F1ADM001))'      */
     End

   Otherwise;
 End

 exit rc

/* Provide a description of the invocation parameters for this exec  */
prompt : parse source . . execname .
say 'This exec reads a Composite Document Presentation Data Stream (CDPDS)'
say 'file and creates a LIST38PP file for printing on a 38xx printer.     '
say '                                                                     '
say '   Format :-                                                         '
say '                                                                     '
say '          'execname' filename filetype filemode                      '
say '                                                                     '
say '          where filename is the input file name                      '
say '                filetype is the input file type                      '
say '                filemode is the input file mode                      '
say '                                                                     '

exit 0
```

Figure 9 (Part 2 of 2). REXX procedure for printing a composite document on a 38xx AFPDS printer under CMS

Displaying and printing double-byte character data

The CDPU supports display and printing of both single-byte character set (SBCS) and double-byte character set (DBCS) text. The text can be alphanumeric text or graphics text.

When DBCS data is displayed by the CDPU, the character set currently defined as the GDDM installation default is used. Mode-1 graphics text is not supported. If you intend to install DBCS fonts other than those supplied as defaults by GDDM, and want them to be emulated by the CDPU, you must update some of the emulation tables supplied by GDDM in support of the CDPU. These are described in Chapter 11, “GDDM font-emulation and conversion tables” on page 123.

SBCS data can be printed and displayed by the CDPU, with the following restrictions:

- SBCS code page 290 (GDDM Katakana) is not supported.
- SBCS code page 1027 (GDDM Japan (Latin) Extended) is supported for mode-3 graphics characters only.

Printers that support composite documents

The CDPU can direct AFPDS and CDPDS documents to AFPDS (page) printers and Intelligent Printer Data Stream (IPDS) printers.

AFPDS printers include:

3800 Model 3 and Model 8
3812
3816
38xx printers
3112 and 3116 printers
3912 and 3916 printers
4028
4224 Models 2xx
4234 Model 11

IPDS printers include:

3812 Model 2 with 3270 Attachment Feature
3816 with 3270 Attachment Feature
3112 and 3116 printers
3912 and 3916 printers
4028
4230
4224 Models 2xx
4234 Model 11

For a complete list of IPDS and AFPDS printers that support printing of composite documents, and for details of any restrictions on their use, see the *GDDM General Information* book.

GDDM does not check the suitability of the code pages or the fonts for the target printer. The application program that generates the CDPDS or AFPDS document must ensure that suitable code pages and fonts are used. The application program must also ensure that the text is formatted to fit within the page.

GDDM does not issue messages when undefined characters are encountered. Checks for such characters are handled by the printer (for IPDS output) or by PSF (for AFPDS output).

Color masters from CDPDS documents

General-use programming interface

Only one color master is allowed from a CDPDS document. Therefore, the only valid value for the MASTERS parameter of the ADMMCOLT macro is 1. The IBM-supplied color table ADM00006 uses this value, and is suitable for translating colors in CDPDS documents into a gray scale.

The ADMMCOLT macro is described in Chapter 14, “Color-separation master tables” on page 145.

Inline resources in AFPDS and CDPDS documents

AFPDS and CDPDS documents can contain information that controls how they should be printed. Such information is called *inline resources*, and specifies page offsets, overlay names, duplex control, and paper source, for example.

When a CDPDS document is directed to an AFPDS printer via the CDPU, you can use the INRESRCE processing option to specify whether the CDPU is to transfer inline resources from the CDPDS input to the AFPDS output. (Note, however, that not all AFPDS printer drivers support inline resources.)

For example, this nickname statement causes inline resources to be transferred to AFPDS output:

```
NICKNAME FAM=4,PROCOPT=((INRESRCE,YES))
```

The CDPU also supports inline resources in CDPDS output, unless the processing option OFFORMAT is set to GRIMAGE or GRCIMAGE.

Inline resources in AFPDS input files are ignored.

End of General-use programming interface

CDPU error reporting

Errors arising from CDPU printing that do not cause the printer to stop are generally written to an error report. The error report is printed on a separate page at the end of the document, and GDDM message ADM2779 is displayed on the screen.

Chapter 6. GDDM objects and other files

A *GDDM object* is a special type of file that is created by GDDM and has a GDDM-internal format. All GDDM objects are held as collections of fixed-length, 400-byte records in all GDDM environments. Each record contains a 20-byte "source key," which includes the name that GDDM last used to store the object. The structure of a GDDM object is described in more detail in the *GDDM Base Application Programming Reference* book.

The GDDM objects are:

Object description	*Object type*
Chart-data files	ADMCDATA
Chart-definition files	ADMCDEF
Chart-format files	ADMCFORM
GDDM-IMD tutorial pages	ADMGIMP
Generated map groups	ADMGGMAP
GKS metafiles	ADMGKSM
Graphics-data-format files	ADMGDF
Image data files	ADMIMG
Image projection definition files	ADMPROJ
Saved-picture files	ADMSAVE
Symbol sets	ADMSYMBL

Each of the GDDM objects is described below.

Chart-format, chart-data, and chart-definition files

Chart-format and chart-data files are produced by the GDDM-PGF ICU (when a chart is saved) or by the GDDM-PGF call CSSAVE. The chart-format file defines the structure of the chart, and the chart-data file holds the content of the chart. This simplifies the production and storage of multiple charts that have the same format.

In the VM and MVS environments only, chart-data-definition files define how imported data in flat files is to be interpreted by the ICU. They are most useful when files of a common format are regularly imported to the ICU.

GDDM-IMD tutorial pages

GDDM-IMD has an online tutorial that can be used for initial instruction in the use of GDDM-IMD and for help while using GDDM-IMD to define maps. These tutorial pages are GDDM objects of type ADMGIMP.

Generated map groups

Maps are screen-layout definitions. They are produced by the GDDM-IMD licensed program in three forms:

- Editable (source) form, of type ADMMSL, held in libraries known as Map Specification Libraries (MSLs).
- Run-time or *generated* form, of type ADMGGMAP, held in map groups as GDDM objects.
- Exportable form, of type ADMIFMT, for use on other systems.

Although GDDM-IMD cannot be used under the IMS subsystem, generated map groups can be created under TSO and imported to run under IMS, using the IMS Import/Export Utility, which is described in Appendix F, "The GDDM Object Import/Export utility (IMS)" on page 393. For more information about generated map groups, see the *GDDM Interactive Map Definition* book.

GKS metafiles

GKS metafiles (ADMGKSM) are sequential files that can be used for storage, transmittal, and transfer of graphical information. Metafile contents can be interpreted to provide graphical output identical to that produced by direct invocation of the relevant GDDM-GKS functions. GKS metafiles are supported in the VM and MVS environments only.

Graphics data format (ADMGDF) files

Graphics data format (ADMGDF) files contain GDF orders, which are encoded forms of GDDM calls that define pictures. ADMGDF files, which are device-independent, can be produced by the GSSAVE call or by the GDDM-PGF ICU. They are retrieved from storage by the ICU or by the GSLOAD call. GDF orders are described in the *GDDM Base Application Programming Reference* book.

In the VM and MVS environments only, ADMGDF files can also be produced by GDDM-GKS users. Such files contain encoded versions of the GDDM call statements used by GDDM-GKS to create a picture. They are device-independent. ADMGDF files are produced by use of the GDF workstation in GDDM-GKS.

Image data files

Image data files, which are of type ADMIMG, contain compressed image data that is interpreted by GDDM. They can be created and accessed by GDDM-IVU, or by the GDDM Base calls IMASAV and IMARST. For information about the image-file format, see the *GDDM Base Application Programming Reference* book.

Image projection definition files

Image projection definitions, which are of type ADMPROJ, contain editing instructions that define how image data is to be moved from a source image to a target image during an image-transfer operation. Because the editing instructions are held separately, they can be applied to any number of images. Image projection definitions can be created and accessed using GDDM-IVU or the GDDM Base calls IMPSAV and IMPRST. For more information about image projection definition files, see the *GDDM Image View Utility* book.

Saved-picture files

GDDM pictures can be saved in a data-stream format using the FSSAVE call. Such pictures, which are of type ADMSAVE, can be reproduced only on a device that has the characteristics of the saved file (for example, the same usable area, or the same PS requirements, if these are used).

Symbol sets

GDDM supplies a large number of image and vector symbol sets, which are of type ADMSYMBL, and you can also produce your own using the Image Symbol Editor or the GDDM-PGF Vector Symbol Editor. For

more information about symbol-set objects, see Chapter 12, “The GDDM default symbol sets” on page 131.

GDDM users can create and save GDDM objects using both locally written GDDM application programs and GDDM interactive utilities, such as the GDDM-PGF ICU and the symbol-set editors. Depending on the volume of objects created at your enterprise, and on the use to which such objects are put, you might find it necessary to create a system to manage them and to prevent unauthorized access to confidential objects, for example.

The GDDM-PGF Interactive Chart Utility (ICU) and GDDM-IMD both provide a facility for listing objects and other files that can be used as part of a housekeeping scheme. The ICU lists chart-format and chart-data files, symbol sets, saved pictures, ADMGDF files, images and projections, and generated map groups. GDDM-IMD can be used to provide a list of generated map groups.

Where GDDM objects are stored

A default destination, which is environment-dependent, is defined for all GDDM objects:

Under CMS GDDM objects are usually written to the A-disk of the user who creates them. Their default filetypes are as defined in the list of objects on page 47.

Under IMS GDDM objects are held in an object database, which is set up during installation. By default, the database DBD name of the GDDM objects is ADMOBJ1.

Under TSO GDDM objects are held as members of partitioned data sets. The ddname allocated before an object is saved or stored identifies the data set to be used. The default ddnames are as defined in the list of objects on page 47.

Under CICS GDDM objects are held in VSAM data sets that are defined in the CICS file control table (FCT). By default, data set ADMF is used for all objects.

Changing the default destination for GDDM objects

In all environments, the default destination for GDDM objects is controlled by the OBJFILE external default, which is described in Appendix A, “External defaults” on page 307. OBJFILE allows you to nominate different destinations for different types of objects (so that you can store symbol sets separately from ADMGDF files, for example).

In the CICS, IMS, and TSO environments, additional destinations (VSAM data sets, database DBD names, or ddnames, as appropriate) can be specified for GDDM objects by means of the GDDM call ESLIB. Any such destination must be defined and made available to the application before any attempt is made to load or save objects.

The symbol-set editors operate on the default data set or data base. They cannot operate on nondefault data sets or databases, unless an application is provided that issues ESLIB followed by the ISSE call (to call the Image Symbol Editor) or the VSSE call (to call the Vector Symbol Editor). ESLIB and ISSE are described in the

GDDM Base Application Programming Reference book. VSSE (a GDDM-PGF call) is described in the *GDDM-PGF Programming Reference* book.

Using private VSAM data sets

In the CICS environment, GDDM objects can be stored in private VSAM data sets. You might want to use private data sets for security reasons (because some GDDM objects are confidential, for example) or to prevent access to the default data sets. (You might find that space is used up rapidly if general access is given to the default data sets. Note, however, that you can use the OBJFILE external default to nominate different data sets for different types of object.)

Private data sets must be defined and initialized before they can be used. A job stream is supplied with GDDM to perform this task.

How to prepare private VSAM data sets

1. Find the supplied job stream:
   - In the CICS/MVS environment, the job stream is stored as member ADMUJC04 in the SADMSAM sample library.
   - In the CICS/VSE environment, the job stream ADMUJD03 is stored as a source statement member of type Z in the installation library.
2. Edit the job stream to include the correct names and other values for the private data sets.
3. Run the job, which should complete with a return code of 0.
4. Add the private data sets to your CICS startup job stream, and define them in your FCT or CSD file.

GDDM-supplied CLISTs and JCL

The following CLISTS and JCL, which you may want to provide for your users, are supplied by GDDM in the SADMSAM data set:

Table 1. CLISTs and JCL supplied by GDDM

Name	Type	Component	Function
ADMUBCDT	CLIST	GDDM/MVS	Generate family-4 printer files from a chart or GDF file
ADMUFILE	CLIST	GDDM/MVS	Transfer files between host and workstations
ADMUPCFT	CLIST	GDDM/MVS	Convert GDF files to ADMGDF format
ADMUJT10	JCL	GDDM/MVS	Print an image file by calling ADMUIMPT
ADMUCIMT	CLIST	GDDM-PGF	Browse or print composite documents

Moving GDDM objects to another system or subsystem

In general, GDDM objects can be moved easily between subsystems. Some considerations that apply to the movement of objects in specific subsystems are listed below:

1. Under CMS, a GDDM object is retrieved using its file name. The internal source key of the object is checked only for consistency: the name in the source key does not have to match the external file name.

   Moving a GDDM object between systems involves transporting the 400-byte records using whatever method is available to your computer system.

2. Under IMS, GDDM supplies an IMS Import/Export Utility, which can be used to transfer objects between an IMS database and a partitioned data set (as might be used on TSO systems). For more information on this utility, see Appendix F, “The GDDM Object Import/Export utility (IMS)” on page 393.

   Note that ADMSAVE files generated under IMS cannot be shown on other subsystems. Similarly, ADMSAVE files generated on subsystems other than IMS cannot be shown under IMS.

3. Under CICS, the 20-byte source key in each record of the object is used to identify the records that constitute the object. The REPRO function of VSAM Access Method Services can be used to move the records representing a particular object between the VSAM data set and a sequential data set on the underlying system. When using REPRO to move an object out of a VSAM data set, you must specify the object name (and possibly the object type) in FROMKEY and TOKEY operands.

4. Under TSO, a GDDM object is stored as a member of a PDS and is retrieved by specifying the member name. The internal source key of the object is checked only for consistency: the name in the source key does not have to match the external member name.

   Moving a GDDM object between systems involves transporting the 400-byte records using whatever method is available to your computer system.

5. When VSE is being used under VM, you might have a problem importing or exporting files across the link. This is because the connection between VM and another system is effected by punching a file into 80 byte-fixed length records. One solution to this problem is to develop a routine that converts 400-byte records to 80-byte records, sends them between the systems, and blocks them at 400 bytes at the other end of the link. Figure 10 on page 52 shows some sample code for exporting a file to the punch. Figure 11 on page 52 shows some sample code for importing objects from the reader.

* $$ JOB JNM=OBJOUT,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=D
* $$ PUN CLASS=A,DISP=D
// JOB OBJOUT - EXPORT OBJECT TO PUNCH
// DLBL DISK,'XXXXXX',0,SD
// EXTENT SYS000,VOLSER,1,0,390,60
// ASSGN SYS000,DISK,VOL=VOLSER,SHR
// DLBL ADMF,'ADMF',,VSAM,CAT=USERCAT
// EXEC IDCAMS,SIZE=AUTO
	Defines DASD

REPRO - INFILE (ADMF) - OUTFILE - (DISK ENV (BLKSZ(400) RECFM(F))) - FROMKEY(objname) - TOKEY(objname) /*	Duplicates on DASD ADMF file 400 FB with a FROMKEY and a TOKEY to select required object from general ADMF file

// DLBL DISK,'XXXXXX',0,SD
// EXTENT SYS000,VOLSER,1,0,390,60
// ASSGN SYS000,DISK,VOL=VOLSER,SHR
// EXEC REBLOCK,SIZE=64K
OBJOUT
/*
/&
* $$ EOJ
	Reblocks 400-byte records on DASD to 80-byte records suitable for PUNCH file. SENDS file to PUNCH

Figure 10. Sample program for exporting a file to the punch

* $$ JOB JNM=OBJIN,CLASS=0,DISP=D
* $$ LST CLASS=A,DISP=D
* $$ PUN CLASS=A,DISP=D
// JOB OBJIN - IMPORT OBJECT FROM READER
// DLBL DISK,'XXXXXX',0,SD
// EXTENT SYS000,VOLSER,1,0,390,60
// ASSGN SYS000,DISK,VOL=VOLSER,SHR
// EXEC REBLOCK,SIZE=64K
OBJIN
	Define DASD

gddm objects in 80-byte form /* *	Receive objects in 80-byte DASD records

// DLBL DISK,'XXXXXX',0,SD
// EXTENT SYS000,VOLSER,1,0,390,60
// ASSGN SYS000,DISK,VOL=VOLSER,SHR
// DLBL ADMF,'ADMF',,VSAM,CAT=USERCAT
// EXEC IDCAMS,SIZE=AUTO
	Define disk area where objects are supplied

REPRO - INFILE - (DISK ENV (BLKSZ(400) RECFM(F))) - OUTFILE (ADMF) - REPLACE /* /& * $$ EOJ	Copy objects in 80-byte records into object file with 400-byte records

Figure 11. Sample program for importing objects from reader

Converting GDDM objects and other files

GDDM supplies several conversion utilities. These support:

- Conversions between some object formats.
- Conversions between some object formats and Picture Interchange Format (PIF) or Computer Graphics Metafile (CGM) format. Such conversions allow interchange with other, non-GDDM applications.
- Conversions from some object formats to formats suitable for page-printer output. Such page-printer formats include PSEG38PP, PSEG3820, PSEG4250, LIST38PP, LIST3820, and LIST4250.

This section identifies GDDM-supported conversions and introduces the GDDM utilities that you use to perform them. For a more comprehensive description of conversions supported by IBM products, see the International Technical Support Centers (ITSC) Printing Library *Document Transforms Cookbook*, GG24-3530.

Table 2 summarizes which conversions are possible. Not every conversion identified here is accomplished in one step. Figure 12 on page 54 is a "route map" that shows the steps involved in the conversions identified as possible in Table 2.

Table 2. Summary of GDDM-supported format conversions

From ↓ to →	*ADMGDF*	*ADMIMG*	*CGM*	*LIST38PP*	*LIST3820*	*LIST4250*	*PIF*	*PSEG38PP*	*PSEG3820*	*PSEG4250*	*GIF*
ADMCDATA	●	●	●	●	●	●	●	●	●	●	
ADMCFORM	●	●	●	●	●	●	●	●	●	●	
ADMGDF		●	●	●	●	●	●	●	●	●	●
ADMIMG				●	●	●		●	●	●	
CGM	●	●		●	●	●	●	●	●	●	
PIF	●	●	●	●	●	●		●	●	●	
PSEG3820		●		●	●	●		●		●	

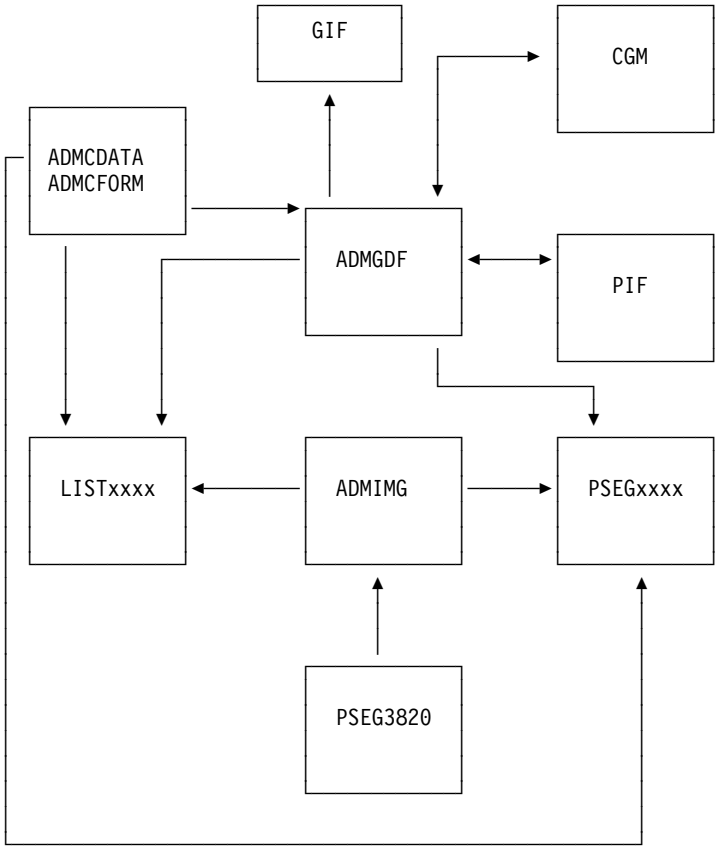


Figure 12. A route map for GDDM-supported conversions. LISTxxxx includes LIST3820, LIST38PP, and LIST4250. PSEGxxxx includes PSEG3820, PSEG38PP, and PSEG4250. PSEG3820 is also shown separately: it is the only one of the PSEGxxxx formats that can be converted to ADMIMG format.

Table 3 identifies the GDDM-supplied utilities that support direct ("one-step") format conversions. In most cases, more than one method is listed. Each of the methods is described briefly following Table 3.

Table 3 (Page 1 of 2). GDDM-supplied conversion utilities

Input	Output	Using
ADMCDATA ADMCFORM	ADMGDF	GDDM-PGF ICU
ADMCDATA ADMCFORM	PSEGxxxx LISTxxxx	ADMUCDSO GDDM-PGF ICU
ADMGDF	PSEGxxxx LISTxxxx	ADMUCDSO GDDM-PGF ICU
ADMGDF	CGM	ADMUGC
ADMGDF	PIF	ADMUPCx
ADMIMG	PSEGxxxx LISTxxxx	ADMUIMPx GDDM-IVU

Table 3 (Page 2 of 2). GDDM-supplied conversion utilities		
Input	**Output**	**Using**
CGM	ADMGDF	ADMUCG
PIF	ADMGDF	ADMUPCx
PSEG3820	ADMIMG	GDDM-IVU
ADMGDF	GIF	ADMUGIF

ADMUCDSO

ADMUCDSO, which is supported in the CMS and TSO environments, invokes the GDDM-PGF ICU noninteractively to format ADMCDATA and ADMCFORM or ADMGDF input for output to a page printer. Output can be in LIST3820, LIST38PP, LIST4250, PSEG3820, PSEG38PP, or PSEG4250 format. The output format is determined by parameters passed to ADMUCDSO.

A sample EXEC (ADMUCIMV EXEC) and a sample CLIST (ADMUCIMT CLIST) that invoke ADMUCDSO are supplied with GDDM. ADMUCDSO, ADMUCIMV, and ADMUCIMT are described in the *GDDM-PGF Programming Reference* book.

ADMUCG Utility ADMUCG, which is supported in the CMS and TSO environments, converts CGM files to ADMGDF format. For more information about ADMUCG, see the *GDDM Base Application Programming Reference* book.

ADMUCG can also be invoked from ADMUPCFx, which is described on page 56.

ADMUGC Utility ADMUGC, which is supported in the CMS and TSO environments, converts ADMGDF files to CGM format. ADMUGC is described in the *GDDM Base Application Programming Reference* book.

ADMUGC can also be invoked from ADMUPCFx, which is described on page 56.

ADMUGIF Utility ADMUGIF, which is supported in the CMS and TSO environments, converts ADMGDF files to GIF format. For more information about ADMUGIF, see the *GDDM Base Application Programming Reference* book.

ADMUIMPx The image print utility ADMUIMPx (where “x” is V under CMS, T under TSO, and D under VSE) formats ADMIMG data for output to a page printer. Sample code is provided to invoke ADMUIMPx as follows:

- ADMUIMP, a REXX-language CMS procedure that invokes ADMUIMPV.
- ADMUJT10, sample MVS JCL that invokes ADMUIMPT.
- ADMUJD10, sample VSE JCL that invokes ADMUIMPD.

These samples can be amended to suit your requirements. ADMUIMPx is described in Chapter 7, “The Image Print Utility, ADMUIMPx” on page 57.

ADMUPCx ADMUPCx (which is ADMUPCT under TSO and ADMUPCV under CMS) converts PIF files to ADMGDF format, and ADMGDF files to PIF format. ADMUPCx is described in the *GDDM Base Application Programming Reference* book. In the TSO environment only, a CLIST (ADMUFILE) that invokes ADMUPCT is provided in data set SADMSAM.

ADMUPCx can also be invoked from the ADMUPCFx utility.

ADMUPCFx

ADMUPCFx (which is ADMUPCFT in data set SADMSAM under TSO, and ADMUPCFV under CMS) is a multipurpose utility that performs both file transfer (between workstation and host) and file conversion. (Refer to the information on transferring files in the *GDDM User's Guide*. It can:

- Convert ADMGDF data to PIF or CGM format, and transfer the results to the workstation
- Transfer PIF or CGM data to the host and convert it to ADMGDF format

On installation, ADMUPCFV is renamed to IND$FILE EXEC, and ADMUPCFT is renamed to IND$FILE CLIST.

GDDM-PGF ICU

The GDDM-PGF ICU (Interactive Chart Utility) can be used to convert:

- ADMCDATA and ADMCFORM objects to ADMGDF format
- ADMCDATA, ADMCFORM, or ADMGDF objects to PSEGxxxx or LISTxxxx format.

The ICU is described in the *GDDM-PGF Interactive Chart Utility* book.

GDDM-IVU The GDDM Image View Utility (IVU) can be used to convert:

- ADMIMG objects to PSEGxxxx or LISTxxxx format
- PSEG3820 objects to ADMIMG format

The GDDM-IVU is described in the *GDDM Image View Utility* book.

Chapter 7. The Image Print Utility, ADMUIMPx

The Image Print Utility, ADMUIMPx, creates page printer files from GDDM image (ADMIMG) objects. It can run under CMS, TSO, and VSE (in batch mode). The entry name and parameter format depend on the environment:

CMS
ADMUIMPV `('name')('scale')('prtname')('procopts') ('token')('field')`
TSO
ADMUIMPT `'name(scale)prtname(procopts)token(field)'`
VSE
ADMUIMPD `'name(scale)prtname(procopts)token(field)'`

The meanings of the six parameter groups are shown below. All parameters except groups 1 (`name`) and 3 (`prtname`) are optional. The utility assigns default values to empty groups.

name
Name of image file to be printed:

Under CMS: `filename filetype filemode`
Under TSO: `image name`
Under VSE: `image name`

scale
Scaling control:

0 Field to fit image: dimensions in `field` ignored. Forced if `fieldh=0` or `fieldv=0`.

1 Original image size: image may be clipped or spaced (default).

2 Scale to fit field: image may be distorted.

3 Rescale with same aspect ratio: image may contain blank space.

4 Same as 0 but rotated 90 degrees counterclockwise.

5 Same as 1 but rotated 90 degrees counterclockwise.

6 Same as 2 but rotated 90 degrees counterclockwise.

7 Same as 3 but rotated 90 degrees counterclockwise.

prtname
Name to be assigned to print file:

Under CMS: `filename filetype filemode`
Under TSO: `ddname`
Under VSE: `dlbl`

procopts
Processing options (procopts) to be used on the DSOPEN call for the printer.

token
Printer device token. Default is `IMG240`.

field

Image field size, in this format:

Under CMS: `fieldh fieldv fields`
Under TSO: `fieldh,fieldv,fields`
Under VSE: `fieldh,fieldv,fields`

where:

`fieldh` = Horizontal dimension (default 0)
`fieldv` = Vertical dimension (default 0)
`fields` = Units for above:
`0` = tenths of inches (default)
`1` = millimeters.

Sample code invoking these modules is supplied with the GDDM Base programs, as follows:

- ADMUIMP, a REXX-language CMS procedure that invokes ADMUIMPV. It produces a file that can be printed on a 3800 Printing Subsystem Model 3 as a page segment to fit an area 6 inches wide and 4 inches deep. The procedure can be amended to suit your installation and users. Normally, you should choose the device token and type (document or page segment), and select the most appropriate output file type (by assigning the required value to `outft`).
- ADMUJT10, sample MVS JCL that invokes ADMUIMPT.
- ADMUJD10, sample VSE JCL that invokes ADMUIMPD.

ADMUIMPx can be used to convert ADMIMG files to AFPDS format for printing on an advanced function printer. An IOCA device token must be specified on input to ADMUIMPx.

Part 2. GDDM devices

Chapter 8. Preparing devices for use with GDDM

GDDM can be used in conjunction with a wide variety of printers, plotters, and display devices. (A complete list of these devices can be found in the *GDDM General Information* book.) The *General Information* book for your system provides some information about ensuring that your existing telecommunication network is able to accommodate GDDM's use of devices. In particular, VTAM users are recommended to provide a LOGMODE table, and to ensure that the LU, TERMINAL and LOCAL macros in the system definition nominate the LOGMODE table and the MODEENT entry in the table.

This chapter takes particular devices as examples, and gives some guidance on defining them so that they are accessible by GDDM. In view of the number and variety of devices supported by GDDM, the information supplied here cannot be comprehensive. However, it does give an indication of what you need to consider when defining devices for use with GDDM. As a general point, you should always refer to the documentation supplied with the device for guidance on device setup. "Books from related libraries" on page xxiv identifies the documentation for many such devices.

Preparing printers for use with GDDM

This section begins with some general advice about using the following 3270-data-stream printers and Intelligent Printer Data Stream (IPDS) printers with GDDM:

- 3270-data-stream printers:
  - IBM 3268 Printer Model 2C
  - IBM 3287 Printer
- IPDS printers
  - IBM 4224 Printer
  - IBM 4234 Dot Band Printer Model 11
  - IBM 3812 Pageprinter Model 2
  - IBM 3816 Printer
  - IBM 3112 Printer
  - IBM 3116 Printer
  - IBM 3912 Printer
  - IBM 3916 Printer
  - IBM 4028 Printer

Following the section of general advice (see "Defining 3270-data-stream printers and IPDS printers" on page 62) each of these printers is discussed in turn.

The 3270-data-stream printers and the IPDS printers can be used for family-1 or family-2 printing. System printers, which are used for family-3 printing, are discussed in "System printers" on page 70. Family-4 printers (advanced-function printers driven by Print Services Facility (PSF) or the IBM 4250 Printer driven by the Composite Document Print Facility (CDPF)) are not driven by GDDM. They must be defined to PSF or CDPF as appropriate.

Defining 3270-data-stream printers and IPDS printers

The following notes apply to all of these printers:

- 3270-data stream printers:
  - 3268-2C
  - 3287
- IPDS printers
  - 4224
  - 4234-11
  - 3812-2
  - 3816
  - IBM 3112 Printer
  - IBM 3116 Printer
  - IBM 3912 Printer
  - IBM 3916 Printer
  - 4028

1. If you are using a 3274 Controller:
   - It must be configured for graphics.
   - It must have sufficient storage. 96KB[7] is adequate for the printers discussed here. (Any 3274 Model 31 has sufficient storage.)
   - The 3274 microcode level must be correct:
     - For the 3286-2C and the 3287, the minimum microcode level is Configuration Support D Release 64.0.
     - For the IPDS printers, the minimum microcode level is Configuration Support D Release 65.1.
   - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Diagnosis* book.
2. If you are using a 3174 Controller with IPDS printers, the minimum microcode release level is Release 2 or higher.
3. If the printer is installed under VM/CMS without VTAM, it must be defined as a 3287 in DMKRIO.
4. IPDS output can be rotated via the IPDSROT processing option (procopt). For family-2 printing, two nickname statements, both of which specify rotation, are required. These two example statements can be used to request rotation in the MVS environment:

[7] 1KB equals 1024 bytes

```
ADMMNICK NAME=IPDSROT,
         TONAME=luname,
         FAM=2,
         PROCOPT=((IPDSROT,90)),
         DEVTOK=X3812Q

ADMMNICK NAME=luname,
         FAM=1,
         PROCOPT=((IPDSROT,90))
```

Device tokens are not required on the second statement because GDDM can query the device.

Other processing options that affect output to IPDS printers are IPDSBIN, IPDSCPI, IPDSIMSW, IPDSLPI, IPDSQUAL, and IPDSTRUN. For more information, see Appendix B, “Processing options” on page 333.

5. The CICS terminal definitions must define correctly the features of the printers that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry.

   - Where VTAM is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or the ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in the CSD entry.
   - All entries for queriable printers must include TRMSTAT=TRANSCEIVE in their DFHTCT entry, or ATI(YES) and TTI(YES) in their CSD TYPETERM entry. (This is required because of the “query device” that is performed automatically by GDDM.)
   - For IPDS printers, set TRMTYPE=SCSPRT in the DFHTCT entry or DEVICE=SCSPRINT in the CSD TYPETERM entry.
   - VTAM SNA printers must have BUFFER=1536 in the DFHTCT entry or SENDSIZE(1536) in the CSD TYPETERM entry. Do not allow this value to default.

   For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.

The 3268-2C printer

The 3268-2C is a 3270-data-stream color printer that supports printing of composite documents, alphanumerics, graphics, and images.

- To enable use of the triple-plane programmed symbols (PS), which is required for graphics printing, the language-switch 1 on the front panel of the printer must be ON.
- Suitable device tokens are L68, L68S, and L68Q. These are described in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Suitable MODEENT macros are:

LU1 SNA / SCS (if SNA character string (SCS) is supported)

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'85C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU3 SNA (if SCS is not supported)

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'8587',
PSERVIC=X'0380000000001850xxxx7F00'
        where xxxx is the buffer size
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 3287 printer

The 3287 is a 3270-data-stream color printer that supports printing of composite documents, alphanumerics, graphics, and images.

- Suitable device tokens are L87 and L87S (for a locally attached 3287) and R87 and R87S (for a remotely attached 3287). These device tokens are listed in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Suitable MODEENT macros are:

LU1 SNA / SCS (if SCS is supported)

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'85C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU3 SNA (if SCS is not supported)

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'8587',
PSERVIC=X'0380000000001850xxxx7F00'
        where xxxx is the buffer size
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 4224 IPDS printer

The 4224 IPDS printer is a desk-top printer that supports printing of composite documents, alphanumerics, graphics, and images.

- Ensure that the printer switch settings, paper size, and device tokens match. If device tokens are being used, the page size defined to the printer must not be smaller than that defined to GDDM by the device token. (However, the printer page size can be the same as, or larger than, that defined by the device token.) For information about the printer switch settings, see the 4224 printer manual *Operating Instructions*, GC31-2546.
- GDDM requires the printer pitch to be set to 10 characters per inch, and the number of lines per inch to 8. You can set these parameters using the 4224 operator panel.
- Suitable device tokens for 4224 printers are described in Appendix C, “Device tokens supplied by GDDM” on page 367.
- The AFPDS-to-IPDS conversion table ADMDKFNT does not define all of the font types supported by the 4224 printer, so you might need to update

ADMDKFNT. For more information, see Chapter 11, “GDDM font-emulation and conversion tables” on page 123.

- Alphanumeric data, except for APL characters, is printed in near letter quality (NLQ) mode. APL characters are printed in Data Processing (DP) mode.
- GDDM cannot load user-defined pattern sets to the 4224 printer, If a pattern from such a set is referenced by an application, a warning message is issued. The 4224 uses its internal default shading pattern (solid) instead.
- The 4224 does not print a marker selected from the system marker set if any part of the enclosing marker box falls outside the graphics field. Markers selected from a user-defined marker set are not subject to this restriction.
- The 4224 supports a graphics-mix mode of overpaint only. GDDM issues a warning message if the application requests any other form of mix mode.
- Symbol sets referenced by alphanumeric fields can be loaded only into extended-storage models of the 4224. GDDM issues an error message if the application uses loadable symbol sets when the printer does not have extended storage. Affected characters are printed using the 4224 resident default font.
- Using the 4224 extended-storage (512KB) model reduces the possibility of graphics output causing printer-storage overflow. If overflow is unavoidable, GDDM issues a warning message and sends only an amount of picture data that fits in the available printer storage.
- Suitable MODEENT macros are:

LU1 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'86C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 4234-11 IPDS printer

The 4234 Model 11 IPDS printer supports printing of composite documents, alphanumerics, graphics, and image.

- There is no storage overflow with this printer: when its storage capacity is exceeded, the 4234 rewinds the paper to continue printing. Alternatively, the data stream can be truncated if the processing option IPDSTRUN is specified. IPDSTRUN is described in Appendix B, “Processing options” on page 333.
- Ensure that any device tokens used match the paper size specified for the printer. If device tokens are being used, the page size defined to the printer must not be smaller than that defined to GDDM by the device token. (However, the printer page size can be the same as, or larger than, that defined by the device token.)
- Suitable device tokens for 4234 printers are described in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Suitable MODEENT macros are:

LU1 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'86C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 3812-2 IPDS printer

The 3812 Model 2 Printer supports printing of composite documents, alphanumerics, graphics, and images. It is supported as an IPDS printer when it has the 3270 Attachment Feature (number 3190).

- Ensure that any device tokens used match the paper size specified for the printer. (The paper size is determined by the paper sensors, located in the paper bins, on the 3812-2.) If device tokens are being used, the page size defined to the printer must not be smaller than that defined to GDDM by the device token. (However, the printer page size can be the same as, or larger than, that defined by the device token.)
- When an “intervention required” condition exists (for example, when the printer runs out of paper or there is a paper jam), the condition is reported to GDDM

after a delay of one minute. GDDM's response is to terminate printing of the current document and issue an error message. If you prefer the 3812 to handle such conditions, change the setting of the printer's configuration switch C17 to "SUPPRESS TIMEOUT=YES" (value 001). This change allows printing to continue when an "intervention required" problem is fixed.

- GDDM-supplied device tokens for the 3812-2 are documented in Appendix C, "Device tokens supplied by GDDM" on page 367.
- Suitable VTAM MODEENT macros are:

LU1 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'86C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 3816 IPDS printer

The 3816 supports printing of composite documents, alphanumerics, graphics, and images. It is supported as an IPDS printer when it has the 3270 Attachment Feature (number 3190).

- Ensure that any device tokens used match the paper size specified for the printer. The printer page size is checked and altered using the printer operator panel. If device tokens are being used, the page size defined to the printer must not be smaller than that defined to GDDM by the device token. (However, the printer page size can be the same as, or larger than, that defined by the device token.)
- GDDM-supplied device tokens for the 3816 IPDS printer are defined in Appendix C, "Device tokens supplied by GDDM" on page 367.
- The IBM 3816-D duplex printer is supported by GDDM for duplex printing.
- When an "intervention required" condition exists (for example, when the printer runs out of paper or there is a paper jam), the condition is reported to GDDM after a delay of one minute. GDDM's response is to terminate printing of the current document. If you prefer the 3816 to handle such conditions, change the setting of the printer's configuration switch C17 to "SUPPRESS

TIMEOUT=YES" (value 001). This change allows printing to continue when an "intervention required" problem is fixed.

- Suitable MODEENT macros are:

LU1 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'86C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

The 4028 IPDS printer

The 3112, 3116, 3912, 3916, and 4028 IPDS printers support printing of composite documents, alphanumerics, graphics, and images.

- Ensure that any device tokens used match the paper size specified for the printer. (To check and alter the printer page size, use the printer operator panel.) If device tokens are being used, the page size defined to the printer must not be smaller than that defined to GDDM by the device token. (However, the printer page size can be the same as, or larger than, that defined by the device token.)
- GDDM-supplied device tokens for the these printers are defined in Appendix C, "Device tokens supplied by GDDM" on page 367.
- Suitable VTAM MODEENT macros are:

LU1 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'7080',
RUSIZES=X'86C7',
PSNDPAC=X'01',
SRCVPAC=X'01',
PSERVIC=X'01000001E100000000000000'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'0080000000001850yyyy7F00'
        where yyyy is the printer ALT BUFFER size
```

System printers

System printers, which are supported by GDDM as family-3 devices, are used for printing alphanumeric data only. A nickname user-default specification (UDS) is required to route data to the system printer. For example:

```
ADMMNICK NAME=DSQPRINT,
         FAM=2,
         TOFAM=3,
         DEVTOK=ADMKSYSP
```

GDDM-supplied device tokens for system printers are listed in Appendix C, “Device tokens supplied by GDDM” on page 367.

pclkdis.GDDM-PCLK displays, printers and plotters

The GDDM-PCLK component of the GDDM Base products enables you to run GDDM applications from host-attached IBM Personal System/2 (PS/2) computers and other personal-computer systems that have IBM PC DOS 2.1 (or later) and a suitable IBM 3270 terminal emulator installed. For more information about the personal-computer systems on which GDDM-PCLK is supported, and about the auxiliary devices that can be attached to them, see the *GDDM General Information* book.

Please note the following:

- If you are using a 3274 Controller:
  - It must be configured for graphics.
  - The minimum microcode level for the 3274 Controller is Configuration Support D Release 64.0.
  - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Diagnosis* book.
  - 3274 Controller Models 31x and 51C must have the integrated diskette-drive enhancement and extended-function store.
- If you are using a 3174 Controller, the minimum microcode release level is A2.0.
- If the personal-computer system is installed under VM/CMS without VTAM, define it as a 3279-3 in DMKRIO.

- Suitable device tokens for GDDM-PCLK-attached workstations, printers, and plotters are defined in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Printers attached directly to a workstation running GDDM-PCLK require a nickname to be defined. For example:

  ```
  ADMMNICK NAME=PCPRINT,TOFAM=1,TONAME=(*,ADMPCPRT)
  ```

- Plotters attached directly to a workstation running GDDM-PCLK require a nickname to be defined. For example:

  ```
  ADMMNICK NAME=PCPLOT,TOFAM=1,TONAME=(*,ADMPLOT)
  ```

- The CICS terminal definitions must define correctly the features of the personal-computer system that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry. Where VTAM is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in your CSD entry.

  For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.

- Suitable VTAM MODEENT macros are:

  LU2 SNA / SCS

  ```
  TYPE=1,
  FMPROF=X'03',
  TSPROF=X'03',
  PRIPROT=X'B1',
  SECPROT=X'90',
  COMPROT=X'3080',
  RUSIZES=X'86C7',
  PSNDPAC='X'01',
  SRCVPAC=X'01',
  PSERVIC=X'028000000000185020507F00'
  ```

  LU0 Non SNA

  ```
  TYPE=1,
  FMPROF=X'02',
  TSPROF=X'02',
  PRIPROT=X'71',
  SECPROT=X'40',
  COMPROT=X'2000',
  PSERVIC=X'008000000000185020507F00'
  ```

os2ldsp.GDDM-OS/2 Link displays, printers, and plotters

The GDDM-OS/2 Link component of the GDDM Base programs enables you to run GDDM applications from host-attached IBM Personal System/2 (PS/2) computers and other personal-computer systems that have OS/2 Extended Edition 1.2 (or later version of OS/2) installed. For more information about the personal-computer systems on which GDDM-OS/2 Link is supported, and about the auxiliary devices that can be attached to them, see the *GDDM General Information* book.

Please note the following:

- If you are using a 3274 Controller:
  - It must be configured for graphics.
  - The minimum microcode level for the 3274 Controller is Configuration Support D Release 64.0.
  - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Diagnosis* book.
  - 3274 Controller Models 31x and 51C must have the integrated diskette-drive enhancement and extended-function store.
- If you are using a 3174 Controller, the minimum microcode release level is A2.0.
- If the personal-computer system is installed under VM/CMS without VTAM, define it as a 3279-3 in DMKRIO.
- A printer or plotter directly attached to the workstation running GDDM-OS/2 Link must be defined to both GDDM (via a nickname statement) and OS/2. Defining a plotter to OS/2 typically requires you to:
  - Set up the communications port
  - Install the presentation driver for the plotter
  - Associate the driver with the plotter
  - Add a plotter queue
  - Set up the plotter (usually via the plotter display window)

  For examples of nickname statements, see Chapter 18, “Nickname user-default specifications” on page 205.

  When plotting via the GDDM print utility to an OS/2-attached plotter, to achieve the maximum remote-output plotting size you must:
  - Set the output dimensions to 47 × 88 on the ICU output panel, or
  - Create a device token with MAXPAGE=(47,88). For more information about creating device tokens, see Chapter 10, “Creating your own device tokens” on page 101.
- Suitable device tokens for GDDM-OS/2-Link-attached workstations, printers, and plotters are defined in Appendix C, “Device tokens supplied by GDDM” on page 367.
- The CICS terminal definitions must define correctly the features of the personal-computer system that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry. Where VTAM

is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in your CSD entry.

For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.

- Suitable VTAM MODEENT macros are:

LU2 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'86C7',
PSNDPAC='X'01',
SRCVPAC=X'01',
PSERVIC=X'028000000000185020507F00'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'008000000000185020507F00'
```

3270 terminals

The 3270 terminals, which comprise the IBM 3278 Display Stations and the IBM 3279 Color Display Stations, use the 3270 data stream and can support the display of composite documents, alphanumerics, graphics and images. This section begins with information common to both the 3278 and 3279 display stations; each group is then considered separately.

- If you are using a 3274 Controller:
  - It must be configured for graphics if graphics are to be displayed.
  - It must have sufficient storage. 96KB is adequate for the displays discussed here. (Any 3274 Model 31 has sufficient storage.)
  - The minimum microcode level is Configuration Support C or D Release 64.0.
  - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Diagnosis* book.
- The *GDDM Diagnosis* book contains details of diagnostic tests that determine:
  - Whether the terminal supports the display of graphics.
  - Whether the device is operating with an extended device control block (EDCB). (3270 devices that have been configured for extended functions,

also known as “structured-field and attribute processing,” should normally operate with an EDCB allocated by the control unit and defined during customization of the control unit.)

- The CICS terminal definitions must define correctly the features of the displays that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry. Where VTAM is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in your CSD entry.

  For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.

3278-2, 3278-3, and 3278-4 display stations

- If the 3278 display station is installed under VM/CMS without VTAM, it must be defined as a 3278 in DMKRIO.
- Support for graphics and image requires:
  - 5790 Programmed Symbols
  - 3620 Character Set Extension
- Suitable VTAM MODEENT macros are:

LU2 SNA

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'8587',
PSERVIC=X'028000000000185018507F00'
        (Model 2 - 24 x 80 Alt screen size)
PSERVIC=X'028000000000185020507F00'
        (Model 3 - 32 x 80 Alt screen size)
PSERVIC=X'02800000000018502B507F00'
        (Model 4 - 43 x 80 Alt screen size)
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'008000000000185018507F00'
        (Model 2 - 24 x 80 Alt screen size)
PSERVIC=X'008000000000185020507F00'
        (Model 3 - 32 x 80 Alt screen size)
PSERVIC=X'00800000000018502B507F00'
        (Model 4 - 43 x 80 Alt screen size)
```

3279-3B, 3279-S3B, 3279-S3G, and 3279-03X color display stations

- If a 3279 display station is installed under VM/CMS without VTAM, it must be defined as a 3279-3 in DMKRIO.
- Support for graphics and image on all displays except the 3279-S3G requires:
  - 5790 Programmed Symbols
  - 3650 Extended Function
- The 3279-03X supports the 8750 Video Output feature, which allows graphics to be copied from the terminal to another device such as a camera or TV screen.
- Suitable VTAM MODEENT macros are:

LU2 SNA

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'8587',
PSERVIC=X'028000000000185020507F00'
        (Model 3 - 32 x 80 Alt screen size)
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'008000000000185020507F00'
        (Model 3 - 32 x 80 Alt screen size)
```

Distributed-function terminals (3179-G, 3192-G, 3472-G, 3472-M)

The “distributed-function” terminals perform many of the functions of the Controller at the device. That is, a downstream-load diskette that contains some of the Controller’s microcode functions is installed in the Controller. The microcode is loaded directly into the terminal when the terminal is powered on.

Terminals in this category include the IBM 3179 Color Display Station Model G, the IBM 3192 Color Display Station Model G, and the IBM 3472 InfoWindow Model G and Model M Displays.

- If you are using a 3274 Controller:
  - It must be configured for graphics if graphics are to be displayed.
  - 3274 Controller Models 31x and 51C must have the integrated diskette-drive enhancement extended-function store.
  - The minimum microcode level for the 3274 is Configuration Support D Release 64.0. The downstream load diskette must include feature 9301 to support the 3179-G and 3192-G.

  - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Diagnosis* book.

- If you are using a 3174 Controller, the minimum microcode level is A2.0.
- The 3179-G and 3192-G support a keyboard definition utility that allows users to define code points associated with keys when the keyboard is in native mode. The default GDDM character set may cause translation of some incoming non-EBCDIC code points, so native mode is supported only for GDDM applications that specify ASTYPE(1) to suppress translations.
- If a distributed-function terminal is installed under VM/CMS without VTAM, define it as a 3279-3 in DMKRIO.
- Suitable device tokens are defined in Appendix C, “Device tokens supplied by GDDM” on page 367.
- The CICS terminal definitions must define correctly the features of the displays that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry. Where VTAM is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in your CSD entry.

  For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.

- Suitable VTAM MODEENT macros are:

LU2 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'86C7',
PSNDPAC='X'01',
SRCVPAC=X'01',
PSERVIC=X'028000000000185020507F00'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'008000000000185020507F00'
```

Auxiliary devices attached to DFT terminals

Auxiliary devices (such as plotters) can be attached to the distributed-function terminals.

Please note the following:

- These plotters can be attached to distributed-function terminals:
  - IBM 6180 Color Plotter (8-pen)
  - IBM 6182 Color Plotter (8-pen, sheet feed)
  - IBM 6184 Color Plotter (8-pen)
  - IBM 6185 Color Plotter (8-pen)
  - IBM 6186 Color Plotter (8-pen)
  - IBM 6187 Color Plotter (8-pen)
  - IBM 7371 Color Plotter (2-pen)
  - IBM 7372 Color Plotter (6-pen)
  - IBM 7374 Color Plotter (8-pen)
  - IBM 7375 Color Plotter (8-pen)

  The 6186 Model 2 and the 6187 Model 2 support roll-feed plotting.
- There are two types of plotter connection, IEEE and RS-232C. IEEE devices can be attached to the 3179-G and the 3192-G. RS-232C plotters can be attached to the 3472-G and 3472-M.
- A 3979 I/O expansion unit is required to support auxiliary devices on the 3179-G and the 3192-G.
- AN IEEE plotter attached to a 3179 or 3192 terminal must have the address on the plotter set to 5.
- When a non-IBM plotter is attached to a 3472-G or a 3472-M, the plotter number of a compatible IBM plotter (for example, 6186) must be entered on the terminal's customization panel. When an IBM plotter is used, you can either enter the correct IBM plotter number or leave the field blank.
- When the 6187 plotter is attached via the RS-232C connection, the following serial interface conditions must be set:

Dataflow	REMOTE/STANDALONE
Baud rate	9600 or 4800 (dependent on terminal)
Parity	0
Handshake	Hardwire: ON XON/XOFF: ON

- Device tokens for attached plotters are described in Appendix C, "Device tokens supplied by GDDM" on page 367.
- Other auxiliary devices are listed in the *GDDM General Information* book.

3193 display station and attached scanner

Images can be scanned using an IBM 3117 Image Scanner (flat-bed) or an IBM 3118 Image Scanner (sheet-feed), and displayed on an IBM 3193 Display Station to which the scanner is attached via the IEEE-488 interface. The 3193 display station, with any attached scanner, is a GDDM family-1 device. Please note the following:

- If you are using a 3274 Controller:

  - It must be configured for graphics.
  - The minimum microcode level for the 3274 Controller is Configuration Support D, Level 64.
  - During setup of the 3274 Controller, specific questions relating to the support of graphics devices must be answered. For more information, see the *3274 Control Unit Planning, Setup and Customization Guide*, GA27-2827, and the *GDDM Base Application Programming Reference* book.
  - 3274 Controller Models 31x and 51C must have the integrated diskette drive enhancement and extended-function store.
- If you are using the 3174 Controller, the minimum microcode level is Release A2.0.
- If the display is installed under VM/CMS without VTAM, define the 3193 as a 3279-3 in DMKRIO.
- The CICS terminal definitions must define correctly the features of the display that GDDM uses, unless you are using a queriable device and have specified QUERY in your CSD terminal entry or FEATURE=QUERYALL or QUERYCOLD in your DFHTCT entry. Where VTAM is in use, the terminal entries must match the VTAM definition. In particular, check the ALTSCRN, COLOR, EXTDS, HILIGHT, and PS operands of the DFHTCT entry, or ALTSCREEN, COLOR, EXTENDEDDS, HILIGHT, and PROGSYMBOLS in your CSD entry.

  For more information about CICS terminal definitions, see the CICS Resource Definition books for your environment.
- Device tokens suitable for the 3193 are defined in Appendix C, “Device tokens supplied by GDDM” on page 367.
- Suitable VTAM MODEENT macros are:

LU2 SNA / SCS

```
TYPE=1,
FMPROF=X'03',
TSPROF=X'03',
PRIPROT=X'B1',
SECPROT=X'90',
COMPROT=X'3080',
RUSIZES=X'8587',
PSNDPAC='X'01',
SRCVPAC=X'01',
PSERVIC=X'028000000000185020507F00'
```

LU0 Non SNA

```
TYPE=1,
FMPROF=X'02',
TSPROF=X'02',
PRIPROT=X'71',
SECPROT=X'40',
COMPROT=X'2000',
PSERVIC=X'008000000000185020507F00'
```

Customizing the 3193 for use with GDDM-IVU

If you are planning to use the 3193 display station with GDDM-IVU, you are recommended to customize it as follows:

1. Power-on the 3193. After the 3193 has completed its power-on tests, the data at the left side of the operator-information area should contain these symbols:

2. Hold down the `Alt` key and press the `Setup` key. This switches the 3193 into level-1 setup mode.
3. The line of information at the bottom of the screen in the operator-information area should be similar to this:

   ```
   <1>                              Aa  LP  [ ][A][I]  CL0  AL1  <1>
   ```

4. If any of the fields `[ ]`, `[A]`, or `[I]` is in reverse video, move the cursor under each one in turn and press the `Cursr Sel` key. This action causes the fields to alternate between normal and reverse video.
5. Move the cursor under the `<1>` at the right-hand end, overtype it with a `2` and press the `Cursr Sel` key. This action puts the terminal into level-2 setup mode. There should be a row of boxes numbered 1 through 6 across the lower part of the screen. In each of the boxes 3 through 5, two sets of letters and numbers are shown. You can ignore the lower set because only the upper set affects the logical terminal used by GDDM-IVU.
6. Perform the following steps. The cursor-move keys take the cursor from one box to the next, and from one item to the next within a box.
   a. In box 1, check that two partially overlapping rectangles are shown, indicating that the 3193's two logical terminals both occupy the full screen. If the rectangles do not overlap, move the cursor into box 1 and press the `Cursr Sel` key repeatedly until one rectangle is shown partially overlapping the other.
   b. In box 2, check that the model number in the upper rectangle is `48R`. If it is something different, move the cursor under the rectangle and press the `Cursr Sel` key until `48R` appears.
   c. In box 3, check that the upper entry has `¬EAB` selected. If it shows `EAB` without the `¬`, move the cursor under it and press the `Cursr Sel` key. The field will alternate between `EAB` and `¬EAB`.
   d. In box 4, check that the upper entry shows the number of partitions as `2`. If it does not, overtype it with a `2`.
7. Press the `Alt` and `Setup` keys again to leave setup mode.
8. Hold down the `Alt` key and press the `Jump` key if necessary to ensure that logical terminal 1 `LT-1` is selected.

For information about tailoring the GDDM-IVU panels, see the *GDDM Image View Utility* book.

3270-PC/G and 3270-PC/GX workstations

If you are using GDDM with an IBM 3270 Personal Computer/G or /GX workstation (with IBM 5279 Color Display or IBM 5379 Display), you need to customize the workstation via the Graphics Control Program (GCP). In particular, you need to specify:

- The name to be used for any plotter attached to the workstation. If you are using a plotter attached to the IEEE-488 port of a 3270-PC/G or /GX, you supply a plotter name during the customization of the device (as described in the *Graphics Control Program User's Guide*). This plotter name corresponds to the "auxiliary-device-name" referenced by GDDM on a DSOPEN call or on a nickname statement.

  GDDM recognizes the reserved name ADMPLOT as a reference to the first plotter attached to a workstation.

- How many host sessions you will be using and their identifiers. This depends on how the 3274 Controller has been set up. If you are going to use complex graphics, you are recommended to restrict the number of host sessions so that performance is not impaired.

- An amount of storage for each session (host or workstation). Storage sizes depend on the type of application you plan to run. Simple alphanumeric pictures require significantly less storage in the workstation than complex graphics.

  To exploit the capabilities of the 3270-PC/G and /GX workstations, such as dragging or local pan and zoom, GDDM must be able to operate in retained mode to these devices. This is the default case, when sufficient segment storage has been allocated to the GDDM host session in the workstation.

  In addition to the increased function that retained mode operation offers, there are potential performance advantages, particularly in the area of data-stream reduction.

  Table 4 and its following notes should be used as a starting point for estimating 3270-PC/G and /GX segment-storage allocation.

Table 4. 3270-PC/G and /GX segment storage requirements

GDDM picture type	Segment storage size
1. ICU business charts	5-15KB
2. Simple GDDM Base graphics	10KB
3. Scientific and engineering graphics	5-30KB
4. Complex cartographic or seismological applications	30-60KB
5. Image applications	200-350KB

Notes:

1. The above figures do not include the requirements for characters from any nondefault GDDM symbol sets that might be used in the picture. Allow 8K bytes for each GDDM-supplied symbol set. User-defined symbol sets may need more than this.
2. The range quoted for complex pictures (type 4) is thought to be reasonable. You should be aware, however, that there is no limit to the size of picture

that GDDM can create for display on these workstations, excepting those imposed by the computing power and storage available in the host processor.

3274 configuration

When using a remote non-SNA (BSC) attachment, the 3270-PC/G and /GX also require that 3274 WACK support is configured. If this is not done, you may experience line timeouts. To configure WACK support, specify **1** in reply to 3274 customization question 176: BSC Enhanced Communication Option (Distributed Function Terminals).

ASCII graphics displays

The following Tektronix and DEC ASCII graphics terminals are supported via the 3174 Controller with the Asynchronous Emulator Adapter (AEA):

Tektronix: 4105, 4205, 4207, 4208, 4209

DEC: VT240, VT241, VT330, VT340

Please note the following:

- GDDM supplies device tokens in support of these devices. The device tokens are listed in Appendix C, “Device tokens supplied by GDDM” on page 367. GDDM selects one of these device tokens according to query-reply data returned from the 3174 Controller. If you specify a device token for an ASCII device, it takes precedence over that selected by GDDM. Device tokens for the ASCII devices are produced by the ADMM3270 macro and are held in the table ADMLSYSA. ADMM3270 is described in Chapter 10, “Creating your own device tokens” on page 101.
- Translation of graphics characters from EBCDIC to ASCII is controlled by GDDM-supplied translation tables. These are described in “ASCII code-page tables” on page 194.
- Translation of incoming ASCII codes to 3270-attention-identifier (AID) and mouse-button codes is controlled by the ADMDGAIT module, which is described on page 85.
- The device characteristics and communications settings of ASCII graphics displays must be configured correctly in the 3174 Controller. For details, see *3174 Establishment Controller Planning Guide Configuration Support B Release 2* and *3174 Establishment Controller Terminal User’s Reference for Expanded Functions*. Note that:
  - Tektronix graphics displays require XON/XOFF flow control to avoid corruption of graphics data. Configure this using question 731 in the 3174 AEA station set and in the display.
  - DEC VT330 and VT 340 terminals should be set to VT300 mode, with the status line set to “host writable.”
- For terminals other than the DEC VT241 and Tektronix 4205, User Defined Tables (UDTs) are required in the 3174 AEA configuration. The Graphics Query Reply field in the UDT must contain the name of one of the GDDM-supplied device tokens for ASCII graphics terminals. These are listed in Appendix C, “Device tokens supplied by GDDM” on page 367.

So that the full screen can be used for GDDM graphics, set the alternate screen size in the 3174 AEA UDTs as follows:

DEC displays	0	(24 row only)
Tektronix 4105/4205	1	(30 row)
Tektronix 4207/4208/4209	2	(32 row)

The graphics-input timeout for Tektronix terminals should be set to 400 milliseconds.

Sample UDTs are shown in Figure 13 on page 83 and Figure 14 on page 84.

A sample UDT for the Tektronix 4208

```
                  User-Defined Terminal (UDT) Selection

 Type the new UDT number and name,
 Then type the Model ID for the initial UDT defaults.


UDT Number U1               U1-U6

Name . . .Tektronix 4208  14 Characters

Model  . .X4                Terminal Table Options (Model ID)
                            U1. UDT-1               V1. DEC VT100
                            U2. UDT-2               V2. DEC VT241
                            U3. UDT-3               V5. DEC VT52
                            U4. UDT-4               V6. DEC VT220
                            U5. UDT-5               H2. HP 2621B
                            U6. UDT-6               L3. LS ADM 3A
                            I1. IBM 3101            T1. Televideo 912
                            I3. IBM 3151/61/62/63   T7. Televideo 970
                            I4. IBM 3164            W1. WYSE 50/60
                            FC. FTTERM Color        X4. Tektronix 4205


 PF: 3=Quit                      8=Fwd
```

```
                   U1 User-Defined Terminal Attributes

Tab to the Selection field, then type the Option.

                                 Selection         Options
                                 _______      ________

Last Line Reserved for Status     N            Y=Yes      N=No
Status Line Character Set . .     0            0-2
Status Line Clear Option          0            0-3
Use Cursor Seq on Status Line     N            Y=Yes      N=No
Scrolling On  . . . . . . . .     Y            Y=Yes      N=No
Cursor Wraps at End of Line .     N            Y=Yes      N=No
Color Supported . . . . . . .     Y            Y=Yes      N=No
Cursor Class  . . . . . . . .     1            0-5
Cursor Sequence . . . . . . .     1B5B3B48  ASCII hex codes
Alternate Screen Size . . . .     2            0-2
Graphics Query Reply  . . . .     TEK4208M  8 Characters
Graphics Input Wait Time  . .     4            0-99 (100 milliseconds)
Graphics Input Ending Seq . .     _________     ASCII hex codes
Graphics Input Length . . . .     128          1-128 Bytes

PF: 3=Quit       4=Default                     8=Fwd
```

Figure 13. User-defined terminal table for Tektronix 4208 with mouse

A sample UDT for the DEC 340

```
                  User-Defined Terminal (UDT) Selection

 Type the new UDT number and name,
 Then type the Model ID for the initial UDT defaults.


UDT Number U2               U1-U6

Name . . .DEC VT340        14 Characters

Model  . .V2                Terminal Table Options (Model ID)
                            U1. UDT-1               V1. DEC VT100
                            U2. UDT-2               V2. DEC VT241
                            U3. UDT-3               V5. DEC VT52
                            U4. UDT-4               V6. DEC VT220
                            U5. UDT-5               H2. HP 2621B
                            U6. UDT-6               L3. LS ADM 3A
                            I1. IBM 3101            T1. Televideo 912
                            I3. IBM 3151/61/62/63   T7. Televideo 970
                            I4. IBM 3164            W1. WYSE 50/60
                            FC. FTTERM Color        X4. Tektronix 4205


 PF: 3=Quit                      8=Fwd
```

```
                  U2 User-Defined Terminal Attributes

Tab to the Selection field, then type the Option.

                                 Selection        Options
                                 _______      ________

Last Line Reserved for Status     Y            Y=Yes      N=No
Status Line Character Set . .     0            0-2
Status Line Clear Option          0            0-3
Use Cursor Seq on Status Line     Y            Y=Yes      N=No
Scrolling On  . . . . . . . .     N            Y=Yes      N=No
Cursor Wraps at End of Line .     N            Y=Yes      N=No
Color Supported . . . . . . .     N            Y=Yes      N=No
Cursor Class  . . . . . . . .     1            0-5
Cursor Sequence . . . . . . .     1B5B3B48   ASCII hex codes
Alternate Screen Size . . . .     0            0-2
Graphics Query Reply  . . . .     DEC340M    8 Characters
Graphics Input Wait Time  . .     2            0-99 (100milliseconds)
Graphics Input Ending Seq . .     5D0D       ASCII hex codes
Graphics Input Length . . . .     128          1-128 Byte s

PF: 3=Quit        4=Default                    8=Fwd
```

Figure 14. User-defined terminal table for DEC 340 with mouse

ASCII-graphics-input translation

The ASCII-graphics-input translation-table module, ADMDGAIT, contains a set of tables used by GDDM to translate incoming ASCII codes to 3270-attention-identifier (AID) and mouse-button codes. These tables are used only when an ASCII graphics device is in graphics-input mode (with the device cross-hair cursor visible).

Structure of module ADMDGAIT

Module ADMDGAIT contains an index table, followed by multiple translation tables for supported ASCII graphics device types, as shown in Figure 15.

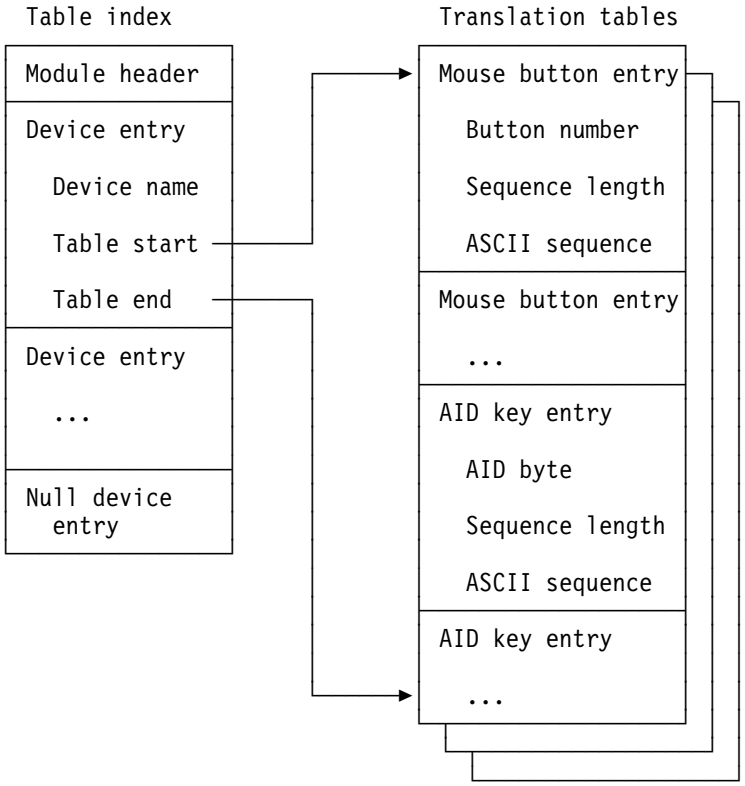


Figure 15. The structure of module ADMDGAIT

The translation tables contain optional mouse-button and AID-key entries, of which the format is:

```
<button number or AID byte>  <length of ASCII sequence>  <ASCII sequence>
```

How to use the ASCII-graphics-input translation table

The GDDM-supplied version of ADMDGAIT contains tables to support ASCII graphics devices with keyboards that are supported by the IBM 3174 Asynchronous Emulation Adapter (AEA).

You might need to change the tables in ADMDGAIT, or add tables to it:

- If you have an ASCII graphics device that is not supported by GDDM
- If you have a keyboard that is not supported by the 3174 AEA
- If you are using a keyboard that is supported via a user-defined table (UDT) in the 3174 AEA

Here is an example of a suitable entry for a Tektronix 4205 graphics terminal:

```
* Table index entry:
         DC    CL8'TEK4205'             Tektronix 4205
         DC    A(TEK4205)                 head pointer
         DC    A(TEK4205E)                tail pointer
         .
         .
         .
* Translation table entry:
TEK4205  DS    0C
*
*        Mouse button ASCII key sequences:
*              Num  Length ASCII sequence
         DC    X'01',X'01',X'31'        Left
         DC    X'02',X'01',X'32'        Middle
         DC    X'03',X'01',X'33'        Right
*
*        AID key ASCII sequences:
*              AID  Length ASCII sequence
         DC    X'7D',X'01',X'0D'        Enter
         DC    X'7D',X'03',X'1B4F4D'    Enter
         DC    X'6D',X'01',X'03'        Clear
         DC    X'F1',X'03',X'1B4F71'    PF1
         DC    X'F2',X'03',X'1B4F72'    PF2
         DC    X'F3',X'03',X'1B4F73'    PF3
         .
         .
         .
TEK4205E DS    0C
```

GDDM selects an appropriate translation table from ADMDGAIT on the basis of the device-token AUXDEV parameter. (AUXDEV is described in Chapter 10, “Creating your own device tokens” on page 101.)

Note: More than one entry can be supplied for an AID key or mouse button. In the example above, two different ASCII sequences are mapped to the ENTER key.

For information on how to replace a GDDM module, see Chapter 19, “Updating GDDM default modules” on page 213.

Saving plotter output in graphics language (IBM-GL) files

In the CMS, TSO, and MVS/Batch environments, family-1 plotter output can be saved on auxiliary storage in IBM-GL (graphics language) format for later output to an IBM or non-IBM plotter. To request that plotter output be saved in an intermediate IBM-GL file or data set, you use the TOFILE processing option.

Please note the following:

- When a TOFILE procopt is in effect, plotter output is directed to auxiliary storage and named as specified by current namelist parameters (specified on either DSOPEN or a nickname user-default specification).
- The POSTPROC processing option enables you to specify a program that is to perform postprocessing on the GL file created. You can specify a program that performs the download using this procopt on the DSOPEN call.

- A plotter device token (that is, a token defined explicitly for a plotter) must be specified on the DSOPEN call or on a nickname user-default specification. A device token defined for a terminal that has a plotter attached as an auxiliary device cannot be used. Plotter device tokens are listed in Appendix C, “Device tokens supplied by GDDM” on page 367.
- The format of the IBM-GL file or data set is subsystem-dependent.

  In the VM environment

  the file-format is V (variable) and the maximum record length is 1020.

  In the MVS environment

  if the data set is preallocated, GDDM determines the file format from the DCB of the data set. If the data set is dynamically allocated, the format is V (variable) and the maximum record length is taken from the IOBFSZ external default. The DCB characteristics of the sequential or partitioned data set are:

  ```
  RECFM     LRECL     BLKSIZE
  -----     -----     -------
  F         <=8000    LRECL
  FB                  LRECL*n
  V                   LRECL+4
  VB                  >=LRECL+4
  ```

  If output is directed to a preallocated data set with a fixed record format, a record length (LRECL) of 1020 is recommended. Any records not completely filled are padded with semicolons (X'3B').
- Output is saved in the format required by the target plotter. If the output is for a plotter attached to a personal computer system, download it as binary data.
- The plotter processing options PLTFORMF, PLTPENV, PLTPENW, PLTPENP, PLTAREA, PLTPAPSZ, PLTROTAT, PLTDELAY can be specified. Of these, PLTDELAY has no effect. All others cause appropriate instructions to be stored in the IBM-GL file.
- The plotter output stored in the file is reverse-clipped. When it is drawn on a plotter, it looks as it would have if GDDM had drawn it directly to the plotter.
- Long plots can be stored in IBM-GL files.
- IBM-GL is defined in the manual *IBM-GL Programming Manual (Graphics Language) for the IBM 6182, 6184, 6185, 6186, and 6187 Color Plotters*, SH23-0092. IBM-GL is a subset of Hewlett Packard (HP) GL.

Producing long plots

GDDM supports the production of long plots on roll-feed plotters, such as the 6186-2 and the 6187-2. The maximum possible length of a plot depends on the width of the paper or other medium (such as vellum or film) being used:

Paper width	*Maximum length of output*
914.4 mm (36 in.)	11.862 m (38.91 ft.)
609.6 mm (24 in.)	10.582 m (34.71 ft.)
279.4 mm (11 in.)	4.389 m (14.40 ft.)

Please note the following:

- Long plots can be produced only when the plotter is driven by GDDM as a family-1 or family-2 device. Long plots are not supported on plotters driven by GDDM-OS/2 Link.
- In the CMS, TSO, and MVS/Batch environments, long plots can be saved in IBM-GL format. (For more information about GDDM support for IBM-GL, see "Saving plotter output in graphics language (IBM-GL) files" on page 86.) Note that such files can contain GL plotter instructions that are not supported by all plotters, or that do not have the desired effect on roll-feed plotters when they are not loaded with a roll-feed medium. On the 6184 and 6185 plotters, such instructions are treated as No Operation (NOP) Instructions. On other plotters, an error condition is raised.
- GDDM-supplied device tokens for use with roll-feed plotters are listed in Appendix C, "Device tokens supplied by GDDM" on page 367. Each supplied device token has a MAXPAGE value of (128,320), which corresponds to the largest possible single-page size for a plotter. This can be altered as described in the documentation of the ADMM3270 macro in Chapter 10, "Creating your own device tokens" on page 101.
- Long plots are supported for family-2 output when the output is spooled to a device with an attached plotter. In this case, the processing option STAGE2ID must be specified at the time the plot file is created. Also, if other processing options are specified that affect the plotted output, they must be specified on the same nickname user-default specification as the STAGE2ID procopt. The nickname for the deferred (stage 2) device must be for a roll-feed capable plotter. The size and position of spooled family-2 plotter output can be further controlled by use of the PRINTCTL procopt.
- Using the PLTDELAY processing option, you can specify a delay between successive frames to allow for media stabilization. If you specify a delay of more than 30 seconds, an informational message is displayed on the console when plotter output begins. The default is no delay. PLTDELAY is ignored if the plotter does not support roll-feed plotting, or if single-sheet output is being produced. The PLTDELAY value is not recorded in IBM-GL files.
- You must ensure that the plotter is loaded with a roll-feed medium that is long enough to accommodate the plot. For the 6187-2 plotter, the front panel option "Frames" should be set OFF (it is ON by default), and the "LongAxis-1.6M" option should be OFF (the default setting). If the plotter runs out of paper while a long plot is being drawn, plotting stops and a message is displayed on the plotter's front-panel display. Plotting continues when a new roll of the plotting medium is loaded.
- GDDM usually draws the x-axis of the output parallel to the longer side of the paper. However, if plot rotation is in effect (either the PLTROTAT procopt has been specified or the plotter front-panel options have been set for rotation) the x-axis is drawn along the direction of the paper width, and the y-axis is drawn in the direction of the roll feed. Frame advance usually occurs in the negative X direction (that is, the right-most frame is drawn first). If plot rotation is in effect, however, frame advance is in the positive y direction (that is, the bottom-most frame is drawn first).
- A black 0.3 mm fiber-tip pen (for paper) or 0.35 mm drafting pen (for vellum or film) must be loaded in stall 8 of the plotter. If the wrong pen is used, gaps can be left between frames.

- For information about writing applications that can produce long plots, see the *GDDM Base Application Programming Guide*.

For information about setting up these plotters for long plots, see the IBM 6186 *Guide to Operations*, SH23-0093, or the IBM 6187 *Guide to Operations*, SH52-0279, as appropriate.

Using non-IBM plotters

Data saved in IBM-GL files can be directed to both IBM and non-IBM plotters. Table 5 lists suitable GDDM-supplied device tokens and the IBM and Hewlett Packard plotters with which the generated GL output is compatible.

Table 5. IBM plotters, their non-IBM equivalents, and suitable device tokens

Device token	IBM plotter	HP plotter
L3179G80 L6180	6180	7440 ColorPro
L3179G82 L6182	6182 6182-031	7550A or B –
L3179G84 L6184	6184	7570 DraftPro
L3179G85 L6185	6185-1 6185-2	7575 DraftPro DXL 7576 DraftPro EXL
L3179G86 L6186	6186-1	7595A Draftmaster
L3G862 L61862	6186-2	7596A Draftmaster
L3179G87 L6187	6187-1	7595B Draftmaster SX
L3G872 L61872	6187-2	7596B Draftmaster RX
L3179G71 L7371 L5550G71	7371	7470
L3179G72 L7372 L5550G72	7372	7475
L7374	7374	7580
L7375	7375-1 7375-2	7585B 7585B

Chapter 9. Reviewing the telecommunication network

This chapter describes how to review the telecommunication network of your enterprise to ensure that it is compatible with GDDM Base.

Checking a VTAM network

If GDDM is to be used with VTAM, you should review the definition of the VTAM network. Pay particular attention to the Bind images that are set in the PSERVIC operand of the MODEENT macro. The Bind image is used by GDDM to determine the type of device to which it is sending the data stream. If the Bind image is wrong, GDDM cannot work.

GDDM requires that a logmode table should exist and that the LU, TERMINAL, or LOCAL macros in the system definition should nominate the logmode table and the MODEENT entry within the table.

For details about the use of these VTAM macros, see the appropriate ACF/VTAM installation book. However, the instructions that follow should be enough to enable you do the work necessary to establish the telecommunication network for GDDM.

Besides checking these macros, you can also consider using VPACING, and enlarging the IOBUF area to improve performance.

Queriable terminals and printers

The term “queriable” is used to describe those devices that support the Write Structured Field (WSF) and Read Partition (Query) Structured Field commands, within the 3270 Data Stream.

The 3270 Data Stream is described in the *IBM 3270 Information Display System Data Stream Programmer's Reference* book. Typical devices within this category, such as any IBM 3270-PC workstation, IBM 3279, IBM 3287, 5550 multistation, or 8775 display terminal, support some of the following:

7-color attributes
Extended highlighting
Graphics (using GDDM)
Image
IPDS-mode operation
Partitions
Programmed symbols (PS)

Instructions for checking a VTAM network

This is what you do:

1. Check whether you are already using a LOGMODE table. If you are not, you must create one with the MODETAB macro. It takes the form:

   ```
   tabname  MODETAB
   ```

   where `tabname` is the name of the LOGMODE table.

   If you do not create a logmode table, the VTAM default table, ISTINCLM, is used, and this is not adequate for many of the operations that can be done by GDDM.

2. Check the MODEENT macros in the logmode table (or create them if you do not already have a logmode table). Examples of MODEENT macros are given in Table 6 on page 93, and instructions on setting up the PSERVIC operand are given in "The PSERVIC operand of the MODEENT macro" on page 95. You are strongly recommended to use the given values; if you do not, your computer system may not work correctly.

   When checking or writing your MODEENT macros, note these important points:

   a. The examples are samples only. If the given PSERVIC values are followed exactly, GDDM will work. However, you may have problems if you change the values. In particular, note the RUSIZES (see next point).

   b. The outgoing request unit (RU) size for SNA 3270 terminals (the second byte in the RUSIZES operand) should be checked carefully.

      - X'87' (indicating 1024 decimal) is suggested as an outgoing RU size. This may be too large for some subsystems.
      - For display terminals attached through a 3274-1A controller, the outgoing RU size must not exceed X'C7' (indicating 1536 decimal).

   c. The SRCVPAC (secondary receive pacing) value must be specified for the terminal, workstation, or multistation as follows (where n is the SRCVPAC value, and m is the outgoing maximum RU size):

      For SNA 3179-G1, 3179-G2, 3192-G, or 3472-G terminals:

      $(2 \times n - 1) \times m \leq 7680$

      For SNA 3270-PC/G and /GX workstations, and 5550 multistations:

      $(2 \times n - 1) \times m \leq 3584$

      For SNA 3193 terminals:
      The fixed size of buffer (7680 bytes) is shared for the pacing buffer by the two LTs that the 3193 supports.

      The maximum RU length that the PLU can send is defined in byte 11 of the BIND, and the pacing count is specified in byte 9.

      The 3193 accepts a maximum RU size and a pacing count within the following limits:

      The BIND to one (LT-1 or LT-2) is accepted if the following formula is met:

      $(2 \times n - 1) \times m \leq 6144$

      The BIND to the other LT is accepted if:

      $(2 \times n - 1) \times m \leq 7680 - buffer$

      where `buffer` is the buffer size already allocated to the first LT.

   d. On nonqueriable printers, the GDDM TSO print utility uses the existence of an alternate screen size to determine which APL character set feature to assume.

      If APL is to be used in the TSO print utility, you must specify an alternate size on any nonqueriable printer on which this is valid, *even when it is the same as the primary size*. The presence of an alternate size is used by GDDM to differentiate between 3268 and 3287 printers.

      For example, on a 3274-attached 3287 printer with a 2KB buffer specification of 1920, an alternate size of 1920 should be given.

e. On nonqueriable display terminals for TSO, GDDM relies on the default and alternate screen sizes being different to establish the type of APL character set that will be sent to the terminal.

   If you use APL and need to have the default and alternate screen size the same for nonqueriable display (non-printer) terminals, see the sections on APL and GDDM in the *GDDM System Customization and Administration* book.

3. Check that the MODETAB and DLOGMOD operands have been included in your system definition table and that they point to the correct logmode tables and MODEENT entries for the terminals.

   You should specifically ensure that the DLOGMOD operands are always *explicitly* included, and not allowed to default.

   The MODETAB and DLOGMOD operands may be coded for individual terminals in the LU, LOCAL, or TERMINAL macros, or for many terminals in the GROUP, LINE, or CLUSTER macros. Examples of the use of these macros are shown in Table 7 on page 95.

4. For queriable non-SNA terminals, FEATUR2=EDATS must be included in the TERMINAL macro for individual terminals, or at a higher level for many terminals in the GROUP, LINE, or CLUSTER macros.

5. Consider the use of VPACING to improve performance because of the large data streams generated by GDDM.

6. If non-SNA displays are going to be used, consider the size of the I/O buffer pool as specified by the IOBUF parameter in VTAM. The large size of the data streams generated by GDDM and the mode in which they are sent by GDDM may require an increase in the value of this parameter.

7. You must define to the host computer as “queriable,” all personal computer systems you intend to use with GDDM-PCLK or GDDM-OS/2 Link to enable file transfer between the host computer system and the personal computer system to work.

Table 6 (Page 1 of 2). Examples of MODEENT macros

The MODEENT macro for non-SNA 3270 display terminals and printers:

```
MODEENT  LOGMODE=modname,                    C
         TYPE=1,                             C
         FMPROF=X'02',                       C
         TSPROF=X'02',                       C
         PRIPROT=X'71',                      C
         SECPROT=X'40',                      C
         COMPROT=X'2000',                    C
         PSERVIC=X'.....
```

Table 6 (Page 2 of 2). Examples of MODEENT macros

The MODEENT macro for SNA 3179-G, 3192-G1, 3192-G2, 3472-G, 3193 display terminals; 3270-PC/G, 3270-PC/GX, and PS/2 workstations; and 5550 multistations:

```
MODEENT  LOGMODE=modname,                    C
         TYPE=1,                             C
         FMPROF=X'03',                       C
         TSPROF=X'03',                       C
         PRIPROT=X'B1',                      C
         SECPROT=X'90',                      C
         COMPROT=X'3080',                    C
         RUSIZES=X'8687',                    C
         PSNDPAC=X'01'                       C
         SRCVPAC=X'01'                       C
         PSERVIC=X'.....
```

The MODEENT macro for SNA 3270 display terminals and printers excluded from the above:

```
MODEENT  LOGMODE=modname,                    C
         TYPE=1,                             C
         FMPROF=X'03',                       C
         TSPROF=X'03',                       C
         PRIPROT=X'B1',                      C
         SECPROT=X'90',                      C
         COMPROT=X'3080',                    C
         RUSIZES=X'8587',                    C
         PSERVIC=X'.....
```

The MODEENT macro for SCS and IPDS printers with FMH-1 support (queriable SCS printers):

```
MODEENT  LOGMODE=modname,                    C
         TYPE=1,                             C
         FMPROF=X'03',                       C
         TSPROF=X'03',                       C
         PRIPROT=X'B1',                      C
         SECPROT=X'90',                      C
         COMPROT=X'7080',                    C
         RUSIZES=X'86C7',                    C
         PSNDPAC=X'01',                      C
         SRCVPAC=X'01',                      C
         PSERVIC=X'.....
```

The MODEENT macro for SCS printers without FMH-1 support (nonqueriable SCS printers):

```
MODEENT  LOGMODE=modname,                    C
         TYPE=1,                             C
         FMPROF=X'03',                       C
         TSPROF=X'03',                       C
         PRIPROT=X'B1',                      C
         SECPROT=X'90',                      C
         COMPROT=X'3080',                    C
         RUSIZES=X'8585', (3276-attached)    C
           or
         RUSIZES=X'8587', (3174/3274-attached)C
         PSNDPAC=X'01',                      C
         SRCVPAC=X'01',                      C
         PSERVIC=X'.....
```

Table 7. Examples of LU, TERMINAL, and LOCAL macros

The LU macro for SNA 3270 devices:

```
LU       LOCADDR=addr,                    C
         MODETAB=tabname,                 C
         DLOGMOD=modname,                 C
         ................
         other parameters
```

The MODETAB and DLOGMOD operands can also be specified on the PU macro.

The LOCAL macro for non-SNA 3270 devices:

```
LOCAL    CUADDR=addr,                     C
         MODETAB=tabname,                 C
         DLOGMOD=modname,                 C
         ................
         other parameters
```

The TERMINAL macro for non-SNA 3270 devices:

```
TERMINAL ADDR=addr,                       C
         MODETAB=tabname,                 C
         DLOGMOD=modname,                 C
         FEATUR2=EDATS,                   C
         ................
         other parameters
```

Notes:

1. Specify FEATUR2=EDATS only for queriable devices.
2. Verify that FEATUR2=MODEL1 is not implicitly specified for devices that have a default buffer size larger than 480 bytes.
3. You can also specify the FEATUR2, MODETAB, and DLOGMOD operands on the GROUP, LINE, and CLUSTER macros.
4. Always specify the DLOGMOD operand explicitly; you should not let it default.

The PSERVIC operand of the MODEENT macro

This section provides more information about the PSERVIC operand.

Bytes 1 through 6, and 12

These bytes define the display terminal or printer type and should be set as follows:

Nonqueriable devices

`X'000000000000.....00'` for a non-SNA display terminal or printer (LU0)
`X'01000000E100.....00'` for an SNA character string (SCS) or IPDS printer (LU1)
`X'020000000000.....00'` for an SNA display terminal (LU2)
`X'030000000000.....00'` for an SNA printer (LU3)

Queriable devices

`X'008000000000.....00'` for a non-SNA display terminal or printer (LU0)
`X'01000001E100.....00'` for an SNA character string (SCS) or IPDS printer (LU1)
`X'028000000000.....00'` for an SNA display terminal (LU2)
`X'038000000000.....00'` for an SNA printer (LU3)

Under VM SNA, the following must be coded for non-SNA (LU0) display terminals that are *not* 3277s:

`X'..40...................'` nonqueriable display terminals and printers
`X'..C0...................'` queriable display terminals and printers

Bytes 7 through 11

These define the “screen sizes”. They also apply to printers.

For IMS display terminals and printers
`X'............185000007E..'` for size 1920 (24x80) – Model 2
`X'............205000007E..'` for size 2560 (32x80) – Model 3
`X'............2B5000007E..'` for size 3440 (43x80) – Model 4
`X'............1B8400007E..'` for size 3564 (27x132) – Model 5

*For non-IMS display terminals and printers, and for*display terminals and printers
`X'............0000000001..'` for size 480 (12x40)
`X'............0000000002..'` for size 1920 (24x80)
`X'............0C280C507F..'` for dual sizes 480,960 (12x40,12x80)
`X'............185018507F..'` for dual sizes 1920,1920 (24x80,24x80) – Model 2
`X'............185020507F..'` for dual sizes 1920,2560 (24x80,32x80) – Model 3
`X'............18502B507F..'` for dual sizes 1920,3440 (24x80,43x80) – Model 4
`X'............18501B847F..'` for dual sizes 1920,3564 (24x80,27x132) – Model 5

For 3290 Information Display Panel, 5279 and 5379 Displays

The 3290, 5279, and 5379 displays (the latter two of which are part of the 3270-PC/G and /GX workstations), may be set up to have any of a large number of screen sizes. Their PSERVIC operands should be:

For IMS systems

`X'............rdcd00007E..'`

where:

(rd,cd)	is the default screen size in rows and columns

For non-IMS systems

`X'............rdcdraca7F..'`

where:

(rd,cd)	is the default screen size in rows and columns
(ra,ca)	is the alternate screen size in rows and columns
rd, cd, ra, and ca	are single-byte hexadecimal numbers

A typical configuration for such a display (and the standard configuration for IBM 5279s and 5379s) would specify a default screen size of 24x80 and an alternate screen size of 32x80.

Particular subsystem levels may restrict the values that can be supported. (This depends on how the device was customized.) For an SNA device, the value here overrides a customized value. For non-SNA devices, the values must match.

For SNA character string (SCS) and IPDS printers

`X'............0000000000..'`

For other printers

Screen sizes should be set that fit within the buffer values, and that correspond to the “character print operation” specify feature. You should set both primary and alternate screen sizes for any nonqueriable printers that are to be used with the GDDM TSO Print Utility, or with GDDM.

For example, for a queriable IBM 3287 printer, or a nonqueriable 3287 printer with the APL/Text feature, with a 2KB buffer, a primary and alternate screen size of 1920 (24 rows, 80 columns) should be given, thus:

For IMS systems

```
X'............185000007E..'
```

For non-IMS systems

```
X'............185000007F..'
```

For non-SCS printers

```
X'............185018507F..'
```

The primary and alternate screen sizes for a queriable IBM 3287 printer with a 4KB buffer, or a nonqueriable 3287 printer with the APL/Text Feature and with a 4KB buffer, should be given as follows:

For IMS systems

`X'............185000007E..'` for size 1920 (24x80) – Model 2
`X'............205000007E..'` for size 2560 (32x80) – Model 3
`X'............2B5000007E..'` for size 3440 (43x80) – Model 4
`X'............1B8400007E..'` for size 3564 (27x132) – Model 5

For non-IMS systems

`X'............185018507F..'` for dual sizes 1920,1920 (24x80,24x80) – Model 2
`X'............185020507F..'` for dual sizes 1920,2560 (24x80,32x80) – Model 3
`X'............18502B507F..'` for dual sizes 1920,3440 (24x80,43x80) – Model 4
`X'............18501B847F..'` for dual sizes 1920,3564 (24x80,27x132) – Model 5

Default logmodes

The specification of the logmode on the macros provides only the *default* logmode. VTAM provides a mechanism to override the default. For example, the terminal operator may be able to specify the logmode on the LOGON command, or a logmode may be added to the LOGON command as a result of command conversion driven by the USSTAB definition table. Therefore, you should ensure that a suitable logmode name is selected, whatever the source of the name.

Part 3. GDDM default values

Chapter 10. Creating your own device tokens

General-use programming interface

GDDM supplies device-characteristics tokens (device tokens) for most uses of most devices. These are described in Appendix C, “Device tokens supplied by GDDM” on page 367. You can define additional device tokens, or alter the supplied tokens, to cater for an unusual device or for an unusual device setup.

Device tokens are defined in device-characteristics tables. Entries in these tables are generated by these macros:

ADMM3270 Generates entries in table ADMLSYS1 (for 3270-family devices) and in table ADMLSYSA (for ASCII graphics terminals)

ADMMSYSP Generates entries in table ADMLSYS3 (for system printers)

ADMMIMAG Generates entries in table ADMLSYS4 (for high-resolution image files).

ADMMAFP Generates entries in table ADMLSYS4 (for AFPDS printers).

The general form of the assembler input to generate the tables is:

```
ADMLSYSx  CSECT
          ADMMxxxx  START
T1        ADMMxxxx  .....          DEFINITION FOR TOKEN T1
T2        ADMMxxxx  .....          DEFINITION FOR TOKEN T2
          ........
          ADMMxxxx  END
          END
```

The CSECT name must be the name of the GDDM default module you are updating (ADMLSYS1, ADMLSYS3, ADMLSYS4, or ADMLSYSA). ADMMxxxx is the name of the macro required to generate entries in the module named on the CSECT entry. Between the ADMMxxxx START and ADMMxxxx END entries there must be at least one device-token definition. The last macro in the list must contain the single parameter `END`. For information on how to replace an updated GDDM default module, see Chapter 19, “Updating GDDM default modules” on page 213.

Each macro is described in more detail below.

The ADMM3270 macro

The ADMM3270 macro is used to define the characteristics of 3270-family and ASCII graphics terminals. The properties of 3270 devices are described in the manual *3270 Information Display System: 3274 Control Unit Description and Programmer’s Guide*, which is referred to throughout this chapter as the “Control Unit Description” manual.

A device token is a chain of query-reply structured fields, organized as follows:

- At least one non-IPDS structured field (all devices)
- A sense type and model (STM) header field, followed by at least one STM structured field (IPDS devices only)

- At least three request-printer-information (RPI) structured fields (IPDS devices only).

The syntax of the macro invocation is as follows:

```
name  ADMM3270  dev,
unit-code | DISPLAY | PRINTER |
                    PLOTTER | START | PLOTNAO | END
          type,     BASE | EXTENDED
          APL=,     YES | NO
          ANOMALY=, reply-field
          ASCIIGL=, DEC|TEK
          ASCIIDR=, 1|2|3
          ASCIISZ=, (xpels,ypels)
          AUXDEV=,  reply-field
          AUXONLY=, YES | NO
          BGTRAN=,  reply-field
          BUFFER=,  (rows,columns)
          COLOR=,   MONO | NO | reply-field
          COMPRES=, YES | NO
          DBCS=,    reply-field
          DDM=,     reply-field
          FLDOUT=,  reply-field
          FMH=,     YES | NO
          GCOLEX=,  reply-field
          GCOLOR=,  reply-field
          GSSETnn=, sdp-field (for nn=00 to 19)
          GSSHDR=,  reply-field
          GSSPSK=,  sdp-field
          HILITE=,  NO | reply-field
          IMAGE=,   reply-field
          IMAUX=,   reply-field
          IMPART=,  reply-field
          KANJI=,   YES | NO
          LCID=,    (lcid-1,lcid-2,....)
          LINETYP=, reply-field
          MAXPAGE=, (rows,columns)
          OEMAUXn=, reply-field (n=0 to 9)
          OUTBND=,  YES | NO
          PART=,    reply-field
          PDEST=,   plotter-destination/origin-id
          PORTn=,   reply-field (for n=0 to 9)
          PNAME=,   plotter-user-name (PLOTTERn)
          PROC=,    reply-field
          PTYPE=,   IBM737x|IBM618x
          PS=,      NO| 0 | 2 | 4 | 6 | reply-field
          REPLY=,   FIELD | EFIELD | CHAR
          REPLYn=,  general-reply (for n=0 to 9)
          SCS=,     YES | NO
          SCSSET=,  (<BASE|FULL>,<BEL|NOBEL>)
          SEGMENT=, reply-field
          STGPOOL=, reply-field
          TEXT=,    YES | NO
          UAREA=,   reply-field
          UNSPEC=,  unspecified-hex-string
          VALID=,   reply-field
```

The following parameters are relevant only to device tokens for IPDS devices:

```
DATCHN=,     reply-field
DSTXBL=,     reply-field
IPDS=,       stm-model
STMBC=,      stm-field
STMDC=,      stm-field
STMDR=,      stm-field
STMIM=,      stm-field
STMIO=,      stm-field
STMLF=,      stm-field
STMOL=,      stm-field
STMPS=,      stm-field
STMPT=,      stm-field
RPIFA=,      rpi-field
RPIFC=,      rpi-field
RPIFI=,      rpi-field
RPIFR=,      rpi-field
RPIPA=,      rpi-field
RPIPAn=,     rpi-field (for n=0 through 3)
RPIPQ=,      rpi-field
RPIRFnn=,    sdp-field (for nn=01 through 19)
RPISPD=,     sdp-field
RPISPn=,     sdp-field (for n=1 through 9)
RPISSF=,     sdp-field
RPISSV=,     sdp-field
```

The keyword parameters can be arranged in any order. The last keyword parameter must not be delimited with a comma.

The explanations of the operands of the ADMM3270 macro instruction are in the order:

1. reply-field
2. general-reply
3. sdp-field (self-defining parameters field)
4. stm-model (sense type and model)
5. stm-field
6. rpi-field (request printer information)
7. name
8. dev
9. type

They are followed by explanations of the keywords in alphabetical order.

Operands of the ADMM3270 macro

reply-field

This is the value of a query-reply structured field of the appropriate type from the fifth byte onward. Its format is either a single data constant, or a list of data constants enclosed within parentheses. These are used as operands to one or more assembler DC instructions. The content of the structured field is described in the Control Unit Description manual. Some examples are shown in Figure 16 on page 115.

A query-reply structured field consists of:

- A 2-byte length field
- A code byte, X'81', identifying it as a query reply
- A type byte, referred to in the descriptions of the parameters that follow
- The reply-field, as specified for the parameter

Use a reply-field to specify the values to be assumed by GDDM for devices that GDDM itself cannot query (dummy devices, nonqueriable devices, or devices under IMS).

general-reply
This is the value of a query-reply structured field from the fourth byte onwards. It is similar to a reply-field, but also includes the type byte.

sdp-field
This is a more general form of reply-field, which is intended for unusual query-reply structured fields that cannot be defined using one of the supplied keywords. The reply-field for some non-IPDS keywords and the rpi-field for some IPDS keywords contain one or more self-defining parameters (SDP). These consist of:

- A 1-byte length field
- A 1-byte type field, referred to in the descriptions of parameters given below
- The sdp-field as specified for the parameter

stm-model
This is a 3-byte data constant, specifying the IPDS product and model numbers. It forms part of the STM header field, which comprises:

- A header byte X'FF'
- A 2-byte product number
- A 1-byte model number
- A reserved field, X'0000'

stm-field
This is the value of an STM query-reply structured field of the appropriate type from the sixth byte onwards. Its format is a single-digit level number, optionally followed by a list of 16-bit data constants. The whole value must be enclosed within parentheses, unless the parameter comprises only the level number. An STM structured field consists of:

- A 2-byte length field
- A 2-byte type field, referred to in the descriptions of the STM parameters below
- A 2-byte level field of the form X'FFn0' where “n” is the single-digit level number
- A sequence of 2-byte property fields

rpi-field
This is the value of an RPI query-reply structured field of the appropriate type from the fifth byte onwards. An RPI structured field consists of:

- A 2-byte length field
- A 2-byte type field, referred to in the descriptions of the RPI parameters below
- The rpi-field, as specified for the parameter

name
This is the name of the device-characteristics token. It is from 1 through 8 characters long and must follow the rules for assembler labels. It must be present on all invocations of the macro except those with START and END, from which it must be omitted. The name must not start with the letters “ADM.”

dev
This indicates the type of device for which the token is defined. The type of device can be:

| `unit-code`, `PRINTER`, `PLOTTER`, `PLOTNAO`, or `DISPLAY`.
`END` is coded when the list of device-token definitions is terminated.
`START` can be specified the first time the macro is invoked.

The most common case is “unit-code.” Unit-code contains two parts separated by a hyphen. For example, 3278-2.

“unit” is one of

3275	3277	3279	3284	8775
3276	3278	3283	3287	

and “code” is a single digit from 1 through 5. It is a code for the character buffer size of the device, although it can be overridden by the BUFFER parameter. For displays, it has the following meanings:

Code	Rows	Columns
1	12	40
2	24	80
3	32	80
4	43	80
5	27	132

The hyphen and “code” can be omitted only when the device is a printer to which SCS data streams are to be sent. In this case, the SCS option described below must be used.

For displays, the buffer size defined by “code” determines the size of the default page created by GDDM.

For printers, the character buffer specification is used by GDDM to determine how much of a page can be transferred to the printer at any one time.

If applicable to a device, it is assumed that the default and alternate character buffer sizes are equal. For details of default and alternate character buffer sizes, see the Control Unit Description manual.

Most of the information associated with a device-characteristics token is held in the table in the form of a QUERY REPLY, whose content is defined for each of the “unit” numbers in the list above. If the device being defined requires a QUERY REPLY that does not match that for one of the unit types defined above and the remaining options cannot override it, the DEV parameter should be coded accordingly as a PLOTTER, DISPLAY, or PRINTER.

If the device is a plotter, GDDM treats it as a separate device and much of the query-reply data from the session device is not relevant. To simplify the construction of tokens for plotters, the DEV parameter can specify PLOTTER, and only those keywords relevant to plotters (PTYPE, PNAME, and PDEST) need be specified. Such tokens can be used for auxiliary devices only.

The PLOTNAO device type specifies a plotter that is not an auxiliary-only plotter. This device type is provided for the circumstances where incompatibities between the primary device and the attached auxiliary-device plotter prevent GDDM from successfully querying the plotter's characteristics. This means that a device token must be specified. The normal PLOTTER tokens cannot be used for this purpose on MVS because the AUXONLY=YES setting prevents the display terminal (to which the plotter is attached) from being opened for messages.

If the device is a display or a printer, the DEV parameter should specify PRINTER if it is a printer and DISPLAY if it is a display, and the rest of the information for the device should be built up by specifying values for the options described below.

Some device tokens can be defined using only the "unit-code," BUFFER, and COMPRES options. If only these options are coded, the remainder of the options default to the maximum function or capability. For example, for 3279 devices, it is assumed by default that all the extended attributes are supported, and that the maximum number of loadable PS stores is available. Thus, one of the GDDM-supplied tokens for a 3279 Model 3 is:

```
L79A3    ADMM3270 3279-3
```

type
This is a gross qualification of the function provided by the device. If this option is specified as BASE, the device supports no extended attributes. If specified as EXTENDED, the device supports at least one extended attribute. By default, all attributes are assumed to be supported, but this can be overridden by the COLOR, HILITE, and PS options described below.

Keywords of the ADMM3270 macro (all 3270 devices)

APL=YES|NO
Indicates whether the device contains the APL character set and has the APL keyboard. APL=NO indicates that the device does not support APL characters.

This parameter can be specified for any device. The default value of the parameter is NO for all devices except the 3279 and 3287, for which the default is YES. You can specify either APL=YES or TEXT=YES, but not both.

When PS=*reply-field* is also specified, you should ensure that it is consistent with respect to "graphic-escape supported."

ANOMALY=*reply-field*
Specifies the reply-field for type X'9D' (anomaly implementation) query-reply structured fields. It is valid in VM and MVS systems only.

ASCIIGL=DEC|TEK
Indicates that the device is an ASCII graphics display that recognizes the DEC ReGIS (DEC Remote Graphics Instruction Set) graphics data stream (ASCIIGL=DEC), or that the device is an ASCII graphics display that recognizes the Tektronix 4100/4200 data stream (ASCIIGL=TEK).

ASCIIDR=1|2|3
Indicates the driving level of an ASCII graphics display.

For Tektronix displays:

1 Tektronix 4105 with microcode level 4.1 or later
2 Tektronix 4200
3 Tektronix 4200 with microcode level 11 or later

For DEC displays:

1 DEC 240 level
2 DEC 340 level

ASCIISZ=(*xpels,ypels***)**
Specifies the screen size, in horizontal and vertical pixels, of a Tektronix graphics display.

AUXDEV=*reply-field*
Specifies the reply-field for type X'99' (auxiliary device) query-reply structured fields.

AUXONLY=YES|NO
Specifies whether the device is an auxiliary-only device (a plotter). AUXONLY does not need to be specified if the *dev* value is PLOTTER or PLOTNAO (see explanation on page 106).

BGTRAN=*reply-field*
Specifies the reply-field for type X'A8' (background transparency) query-reply structured fields.

BUFFER=(*rows,columns***)**
Specifies the character-buffer size in rows and columns. This parameter can be specified for devices of type EXTENDED only. It is required for the generic DISPLAY or the generic non-SCS PRINTER devices. It is optional for generic IPDS PRINTER devices, and invalid for generic SCS PRINTER devices.

COLOR=MONO|NO|*reply-field*
Indicates whether the device supports the color attribute.

COLOR=MONO indicates that Set-Attribute orders can be sent to the device, but that each color is displayed or printed in the default color.
COLOR=NO indicates that the device does not support Set-Attribute orders for color.

If neither of the above two options is coded, the value specified is assumed to be a reply-field for type X'86' (color) query-reply structured fields.

COMPRES=YES|NO
Indicates the load PS data format that can be sent to the device.

COMPRES=YES indicates that the device accepts formats in which the definition of each character has been compressed to reduce line traffic. COMPRES=YES is valid for EXTENDED devices only.
COMPRES=NO indicates that compressed character definitions cannot be sent to the device. This is the default setting.

DBCS=*reply-field*
Specifies the reply-field for type X'91' (DBCS-Asian mode) query-reply structured fields.

DDM=*reply-field*
Specifies the reply-field for type X'95' (distributed data management) query-reply structured fields.

FLDOUT=*reply-field*
Specifies the reply-field for type X'8C' (field outlining) query-reply structured fields.

FMH=YES|NO
Indicates whether Function Management Headers Type 1 (FMH-1) may be sent to an SCS device. If this parameter is not specified for an SCS device, the default value is YES if the device supports the SCS data structured field, and NO if it does not.

GCOLEX=*reply-field*
This parameter can be used as an extension of the GCOLOR parameter, when the GCOLOR specification exceeds the assembler limit of 455 characters for macro operands.

GCOLOR=*reply-field*
Specifies the reply-field for type X'B4' (graphics color) query-reply structured fields.

GSSETnn=*sdp-field*
Specifies the sdp-field for the type X'01' (symbol store definition) self-defining parameter for the graphics symbol sets query-reply structured field. The value “nn” is a number in the range 00 through 19.

The value is from the third byte onwards.

GSSHDR=*reply-field*
Specifies the first part of the reply-field for type X'B6' (graphics symbol sets) query-reply structured fields. The remainder of the reply-field can be specified by one or more GSSET00 through GSSET19 operands and a GSSPSK operand.

GSSPSK=*sdp-field*
Specifies the sdp-field for the type X'02' (proportional spacing and kerning) self-defining parameter for the graphics symbol sets query-reply structured field. Like the GSSETnn operand, the value is from the third byte onwards.

HILITE=NO|*reply-field*
Indicates support for the Set Attribute (extended highlighting) order by the device. HILITE=NO indicates that the device does not support the order. Any other value is assumed to be a reply-field for type X'87' (highlighting) query-reply structured fields.

IMAGE=*reply-field*
Specifies the reply-field for type X'82' (image device) query-reply structured fields.

IMAUX=*reply-field*
Specifies the reply-field for type X'AA' (image auxiliary device) query-reply structured fields.

IMPART=*reply-field*
Specifies the reply-field for type X'A6' (implicit partitions) query-reply structured fields.

This option is required if UAREA does not define the buffer size explicitly (for example, for the IBM 3290).

KANJI=YES|NO
Indicates whether the device supports the KANJI character set. This parameter may be coded only if the *dev* value is the unit code 3278-2, 3283-2 or (if SCS=YES) 3283.

LCID=(*lcid-1,lcid-2,...***)**
This is a list of symbol-set identifiers, one for each loadable symbol set. Each identifier must be two hexadecimal digits. See the Control Unit Description manual for valid values of these identifiers.

LINETYP=*reply-field*
Specifies the reply-field for type X'B2' (line types) query-reply structured fields.

MAXPAGE=(*rows,columns***)**
This is an explicit specification, in rows and columns, of the page size that is used by GDDM if the default page is created. (The default page is created as page 0 by GDDM if no explicit FSPCRT call is used.)

The rows and columns values must be positive. For display devices, they must be less than or equal to the rows and columns values of the character-buffer size respectively.

For both displays and printers, rows × columns × 2 must be less than or equal to 32 000.

If the MAXPAGE parameter is not explicitly specified in the device token, GDDM applies the following default values for different types of device:

Displays
: The default values are determined by the character buffer size.

Printers with SCS=YES
: GDDM selects defaults appropriate to the contents of the UAREA field of the device token, and the limits defined above.

IPDS printers
: GDDM selects defaults appropriate to the contents of the RPIPA field of the device token, and the limits defined above.

nonIPDS, nonSCS printers
: GDDM applies a default value of (80,132).

For plotters, the paper size loaded on the plotter or the PLTPAPSZ Procopt, if specified, determines the size of the page in rows and columns. The MAXPAGE values are used only if the plotter is not directly attached, such as when Family-2 ADMPRINT files are spooled to a remote plotter. In this case GDDM uses the MAXPAGE values to select the paper size for the eventual plotter output. The paper size selected is the largest that can contain the number of rows and columns specified. If even the largest paper size supported by the plotter type is not large enough for MAXPAGE rows and columns, the largest available paper size is selected and the size for the default page is set to the number of rows and columns associated with that size of paper.

The number or rows and columns associated with each size of paper is given in the description of the FSPCRT call in the *GDDM Base Application Programming Reference* book.

OEMAUXn=*reply-field*
The values OEMAUX0 through OEMAUX9 specify the reply-fields for type X'8F' (OEM auxiliary device) query-reply structured fields.

OUTBND=YES|NO
Indicates support for the outbound 3270-structured-field for displays or, for SCS printers, the support of the SCS data structured field. OUTBND=NO indicates that the device does not support the structured field.

This option can be coded for devices of type EXTENDED only, and must not be coded for the generic devices PRINTER and DISPLAY. The default is YES where the *dev*value is a unit code and the *type* value is “EXTENDED.”

Devices attached to the IMS subsystem and supported by GDDM should specify OUTBND=YES. If NO is specified, GDDM cannot generate graphics for the device.

PART=*reply-field*
Specifies the reply-field for type X'84' (partitions) query-reply structured fields. If this parameter is not specified, the device is assumed to have no partition capability.

PDEST=*plotter-destination|origin-id*
This is the IEEE address of an attached plotter.

PORTn=*reply-field*
The values PORT0 through PORT9 specify the reply-fields for type X'B3' (port) query-reply structured fields.

PNAME=*plotter-user-name*
This is the name of an attached plotter. It must match the name specified when you customize your workstation. Suggested names are of the form PLOTTERn.

PROC=*reply-field*
Specifies the reply-field for type X'B1' (procedure) query-reply structured fields.

PTYPE=IBM737x|IBM618x
The type of an attached plotter. Supported values are:

IBM6180
IBM6182
IBM6184
IBM6185
IBM6186
IBM61862
IBM61872
IBM7371
IBM7372
IBM7374
IBM7375

Only the values listed here are recognized. When the device token is used, GDDM uses the value of PTYPE to infer various plotter characteristics, such as the paper sizes and the number of pens available.

PS=NO|0|2|4|6|*reply-field*
Indicates the number of programmed-symbols-set stores in the device. NO means the same as 0. The value “6” can be coded only for displays.

If the value is not NO, 0, 2, 4, or 6, it is assumed to be a reply-field for type X'85' (character sets) query-reply structured fields.

REPLY=FIELD|EFIELD|CHAR
Indicates the reply modes supported by the device. REPLY=FIELD indicates support for Field-Mode, REPLY=EFIELD indicates support for Extended-Field-Mode, and REPLY=CHAR indicates support for Character-Mode incoming data streams. These modes are described in the Control Unit Description manual.

REPLYn=*general-reply*
Parameters REPLY0 through REPLY9 specify query-reply structured field values from the fourth byte onward.

SCS=YES|NO
Indicates whether SCS data streams are accepted by the device. If SCS=YES, a type X'A2' (data stream) query-reply structured field is generated. The “reply-field” defaults to either X'00' (for non-IPDS tokens) or X'0002' (for IPDS tokens).

This parameter can be coded for PRINTER devices only. The default is NO.

SCSSET=BASE|FULL, (BEL|NOBEL)
Identifies the subset of SCS data streams accepted by a device for which SCS=YES has been specified. SCSSET=FULL indicates that the FULL BASE subset is supported., SCSSET=BASE indicates that the BASE subset is supported. The content of these two subsets of SCS is defined in the Control Unit Description manual.

The BEL value indicates that BEL can be sent to the device; NOBEL indicates that it can not.

SCSSET is valid only for devices for which SCS=YES. The default is “FULL,BEL” for EXTENDED devices, and “BASE,NOBEL” for BASE devices.

SEGMENT=*reply-field*
Specifies the reply-field for type X'B0' (segment) query-reply structured fields.

STGPOOL=*reply-field*
Specifies the reply-field for type X'96' (storage pools) query-reply structured fields.

TEXT=YES|NO
Specifies whether the device contains the TEXT character set and has a Data Entry keyboard.

This option can be coded for any device. The default is NO for all devices. You can specify either APL=YES or TEXT=YES, but not both.

If “PS=reply-field” is also specified, ensure that it is consistent with respect to “graphic-escape supported.”

UAREA=*reply-field*
Specifies the reply-field for type X'81' (usable area) query-reply structured fields.

UNSPEC=*unspecified-hex-string*
Specifies a complete query-reply structured field from the first byte onward.

VALID=*reply-field*
Specifies the reply-field for type X'8A' (validation) query-reply structured fields. If this parameter is not coded, the device is assumed to have no validation feature.

Keywords of the ADMM3270 macro (IPDS devices only)

DATCHN=*reply-field*
Specifies the reply-field for type X'98' (data-chaining) query-reply structured fields. This parameter must be specified if SCS data streams are not accepted.

DSTXBL=*reply-field*
Specifies the reply-field for type X'9A' (LU-0 data-stream type, and transfer-buffer limit) query-reply structured fields. This parameter must be specified if SCS data streams are not accepted.

IPDS=*stm-model*
Identifies an IPDS token and therefore:

Validates the inclusion of STM and RPI keywords.
Defines the product and model numbers. These are stored in the STM reply-list header.

STMBC=*stm-field*
Specifies the stm-field for type X'C2C3' (bar code) STM query-reply structured field.

STMDC=*stm-field*
Specifies the stm-field for type X'C4C3' (IPDS device control) STM query-reply structured field.

STMDR=*stm-field*
Specifies the stm-field for type X'E5C7' (vector graphics) STM query-reply structured field.

STMIM=*stm-field*
Specifies the stm-field for type X'C9D4' (Advanced Function Presentation Data Stream – AFPDS – image) STM query-reply structured field.

STMIO=*stm-field*
Specifies the stm-field for type X'C9D6' (IDF image) STM query-reply structured field.

STMLF=*stm-field*
Specifies the stm-field for type X'C3C6' (loaded font) STM query-reply structured field. Three levels are supported with the following dependencies:

LF/1 — coded fonts only
LF/2 — loaded symbol sets only
LF/3 — coded fonts and loaded symbol sets

STMOL=*stm-field*
Specifies the stm-field for type X'D6D3' (overlay) STM query-reply structured field.

STMPS=*stm-field*
Specifies the stm-field for type X'D7E2' (page segment) STM query-reply structured field.

STMPT=*stm-field*
Specifies the stm-field for type X'D7E3' (presentation text) STM query-reply structured field.

RPIFA=*rpi-field*
Specifies the rpi-field for type X'0007' (features available) RPI query-reply structured field.

RPIFC=*rpi-field*
Specifies the rpi-field for type X'0005' (foreground colors) RPI query-reply structured field.

RPIFI=*rpi-field*
Specifies the rpi-field for type X'0006' (features installed) RPI query-reply structured field.

RPIFR=*rpi-field*
Specifies the rpi-field for type X'0003' (font resolution) RPI query-reply structured field.

RPIPA, RPIPAn=*rpi-field*
Specify the rpi-field for type X'0001' (printable area definition) RPI query-reply structured field. **n** is the bin number, for multiple bin printers.

RPIPQ=*rpi-field*
Specifies the rpi-field for type X'0009' (print quality) RPI query-reply structured field.

RPIRFnn=*sdp-field*
Specifies the sdp-field for type X'01' (resident font-code-page support) self-defining parameter. A number of these parameters are concatenated together to form the rpi-field for type X'0008' (resident font list) RPI query-reply structured field.

RPISPD=*sdp-field*
Specifies the sdp-field for type X'01' (storage pool) self-defining parameter. This default parameter always forms the first or only part of a type X'0004' (storage pool list) RPI query-reply structured field.

RPISPn=*sdp-field*
Specifies the sdp-field for type X'01' (storage pool). These supplementary parameters are concatenated with the default parameter to form the rpi-field of a type X'0004' (storage pool list) RPI query-reply structured field.

RPISSF=*sdp-field*
Specifies the sdp-field for type X'01' (symbol set, fixed block size) self-defining parameter. This parameter forms the first or only part of the RPI field for a type X'0002' (symbol set) RPI query-reply structured field.

RPISSV=*sdp-field*
Specifies the sdp-field for type X'02' (symbol set, variable block size) self-defining parameter. This parameter forms the second or only part of the RPI field for a type X'0002' (symbol set) RPI query-reply structured field.

Notes relating to IPDS tokens:

1. The IPDS keyword may be coded only for PRINTER devices.
2. The IPDS device supports LU-1 mode (SCS=YES), or LU-0 mode (SCS=NO), LU-0 being the default.
3. A data-stream query-reply structured field is generated either (1) as described by the SCS keyword for LU-1 mode tokens, or (2) as described by the DSTXBL keyword for LU-0 mode tokens.
4. The BUFFER keyword can be omitted. This is equivalent to BUFFER=(0,0).
5. All objects that are not specified in supplementary storage pools are stored in the default storage pool.
6. If a null parameter is coded for features available, the default is a copy of the features-installed parameter.
7. The keywords STMDC, RPIFR, RPIPA, and RPISDP must be specified. All the other STM or RPI keywords are optional.

An example of the ADMM3270 macro

An example of the coding for a device-characteristics token is shown in Figure 16.

```
*********************************************************************
*                                                                   *
*      EXAMPLE OF FULL SPECIFICATION OF DEVICE TOKEN.               *
*                                                                   *
*      (The following hexadecimal Reply-fields specify Query        *
*       Reply structured fields from the fifth byte onwards.)       *
*                                                                   *
*********************************************************************
ADMK9060 ADMM3270 DISPLAY,EXTENDED,BUFFER=(62,160),REPLY=CHAR,APL=YES, X
               UAREA=(X'03A003C002EF010005000E0005000E060C0000060C101F'X
               ),                                                    X
               HILITE=(X'0400F0F1F1F2F2F4F4'),                       X
               PART=(X'10600080',X'07020200000000'),                 X
               IMPART=(X'0000',X'0B01000050001800A0003E',X'0B0200000600X
               0C0006000C'),                                         X
               PS=(X'B80009106000000005',X'0000000910',X'0000000910',X'X
               0100F10910',X'0100F10910',X'0280FF0910',X'0380FF0910',X'X
               0480FF0910',X'0580FF0910',X'0680FF0910',X'0780FF0910'), X
               COLOR=(X'000800FAF100F200F300F400F500F600F700')

*********************************************************************
```

For detailed descriptions of each of the above reply-fields, see the Control Unit Description manual. The query-reply structured fields generated by the macro are as follows:

```
ADM1011A EQU     *
         DC      AL2(ADM1011B-ADM1011A)
         DC      X'8181'          USABLE AREA
         DC      X'03A003C002EF010005000E0005000E060C0000060C101F'
ADM1011B EQU     *
ADM1014A EQU     *
         DC      AL2(ADM1014B-ADM1014A)
         DC      X'8184'          PARTITIONS
         DC      X'10600080',X'07020200000000'
ADM1014B EQU     *
ADM1015A EQU     *
         DC      AL2(ADM1015B-ADM1015A)
         DC      X'8185'          CHARACTER SETS
         DC      X'B80009106000000005',X'0000000910',X'0000000910',X'01X
               00F10910',X'0100F10910',X'0280FF0910',X'0380FF0910',X'04X
               80FF0910',X'0580FF0910',X'0680FF0910',X'0780FF0910'
ADM1015B EQU     *
ADM1016A EQU     *
         DC      AL2(ADM1016B-ADM1016A)
         DC      X'8186'          COLOR REPLY
         DC      X'000800FAF100F200F300F400F500F600F700'
```

Figure 16 (Part 1 of 2). Example device token for inclusion in ADMM3270

```
ADM1016B EQU     *
ADM1017A EQU     *
         DC      AL2(ADM1017B-ADM1017A)
         DC      X'8187'          HIGHLIGHTING
         DC      X'0400F0F1F1F2F2F4F4'
ADM1017B EQU     *
ADM1018A EQU     *
         DC      AL2(ADM1018B-ADM1018A)
         DC      X'8188'          REPLY MODES
         DC      X'000102'
ADM1018B EQU     *
ADM1016E EQU     *
         DC      AL2(ADM1016F-ADM1016E)
         DC      X'81A6'          IMPLICIT PARTITIONS
         DC      X'0000',X'0B01000050001800A0003E',X'0B02000006000C0006X
               000C'
ADM1016F EQU     *
```

Figure 16 (Part 2 of 2). Example device token for inclusion in ADMM3270

The ADMMSYSP macro

The ADMMSYSP macro defines the device characteristics of a system printer (a family-3 printer) that is accessed using GDDM functions. The syntax of the macro is as follows:

```
name  ADMMSYSP  type,            3800 | * | omitted
                MAXPAGE=,        (rows,columns)
                SPACING=         (lines/inch,columns/inch)
```

name
The name of the device-characteristics token. It can be up to 8 characters long, must follow the rules for assembler labels, and must not begin with the letters “ADM.” It must be present on all invocations of the macro except the last, from which it must be omitted.

type
Determines whether one- or two-character carriage-control sequences are to be used. If no type value is specified, or if an asterisk * is coded, a single carriage-control character is used. If “3800” is specified, the additional table reference characters for 3800 printers are used.

MAXPAGE=(*rows,columns*)
The maximum number of rows and columns that can be created on a GDDM page. Each value is an integer in the range 1 through 32767. The default value is (66,132).

As the number of columns does not include space required for carriage control, the output print file will be 1 or 2 characters wider than the size given.

SPACING=(*lines/inch*, *columns/inch*)
The number of lines per inch followed by the number of columns per inch on the device. Each is an integer in the range 1 through 32767. The default value is (6,10).

The ADMMIMAG macro

To generate device tokens for family-4 (page-printer) devices, you can use the ADMMIMAG macro instruction. However, for AFPDS devices you are recommended to use the ADMMAFP macro in preference to the ADMMIMAG macro.

The syntax of the macro invocation is as follows:

```
name   ADMMIMAG  type,              device number
                 PPI=value,         pixels per inch
                 LINE=value,        pixels per unit line width
                 PAPER=(width,      default paper width
                        depth,      default paper depth
                        units)      units (1/10 inch or mm)
```

name
The name of the device-characteristics token. It can be up to 8 characters long, must follow the rules for assembler labels, and must not begin with the letters “ADM.” It must be present on all invocations of the macro except the last, from which it must be omitted.

type
The device number. This value is required if the image data is to be formatted according to specific device requirements. It can be:

⋆	4250 (the default value)
4250	IBM 4250 Printer
3800	IBM 3800 Printing Subsystem
42xx	IBM 4224 Printer and IBM 4234 Dot Band Printer
4028	IBM 4028 Printer
38xx	IBM 38xx (3812, 3816), 3112, 3116, 3912, 3916 printers
UNF	Unformatted (bit-map data)

PPI=*pixels per inch*
The factor to be used in converting the image size from inches to pixels.

LINE=*pixels-per-unit line width*
The factor to be used in calculating the widths of graphics lines. The value indicates the number of pixels per unit line width. It also affects the pitch of a line type: dashes or dots increase in number-per-inch for a thinner line. Note that approximations due to pixel boundaries can sometimes give unexpected results when output from different devices is compared. For example, there could be variations in the number of dashes or dots on a given length of line.

PAPER=(*width, depth, units***)**
Defines the default usable size of the medium on which the image is to be drawn. The width and depth values are specified in tenths of an inch or in millimeters according to the units parameter.

```
units = 0       tenths of an inch
        1       millimeters
```

Note: When a graphics or image field is defined, the size of the field is rounded down to a multiple of 32 pixels in both directions. This can cause the picture to appear slightly smaller than the specified size.

Example of the ADMMIMAG macro

```
IMG85    ADMMIMAG 4250,
                  PPI=600,
                  LINE=6,
                  PAPER=(85,110,0)
```

The ADMMAFP macro

To generate device tokens for AFPDS devices, you are recommended to use the ADMMAFP macro. ADMMAFP supports alphanumeric output, vector graphics (GOCA), and compressed image (IOCA).

The syntax of the macro invocation is:

```
name   ADMMAFP  TYPE,              device type
                RES=,              (xres, yres)
                SIZE=,             (width, depth, units)
                CELLS=,            (cols, rows)
                FONTNME=,          (name,xsize,ysize)
                CODEPGE=,          name
                FONTCHR=,          (name,xsize,ysize)
                DBCSFNT=,          (name,xsize,ysize)
                GOCA=,             level
                IM=,               level
                IOCA=,             level
                PTOCA=,            level
                LINEW=             pels
```

name
This is the name of the device token. It is 1 through 8 characters long and must follow the rules for assembler labels.

TYPE
A comment parameter used to indicate the model number of the target printer (for example, 3816).

RES=(*xres, yres***)**
Defines the horizontal (xres) and vertical (vres) resolution of the device. Units are pixels per inch (ppi). Default is 240 ppi.

SIZE=(*width, depth, units***)**
Defines the horizontal and vertical dimensions of the usable area. This value is usually smaller than the paper size to allow for limitations of the printer and any margins added during PSF processing. Valid units are:

PELS Pixels
TENTHS Tenths of an inch
MM Millimeters.

This parameter is mandatory; there is no default value.

CELLS=(*cols, rows***)**
Specifies the usable area in terms of character columns and rows to allow positioning of alphanumerics on the page. There is no default value. If this parameter is omitted, alphanumerics are not written.

FONTNME=*(name,xsize,ysize)*

name is the coded-font name to be used for alphanumerics. The font can also be defined by code-page name and character-set name (see below). *xsize* and *ysize* are the dimensions of the font character box. If they are not specified, a default value is calculated using the values specified on the SIZE and CELLS parameters.

GDDM-supplied tokens use a 10 characters-per-inch (cpi) Serif Text font, which is the default if the FONTNME parameter is not specified. If you define your own tokens, you must ensure that the font you choose is available at the destination printer.

CODEPGE=*name*

Defines the code-page name, which can be used together with the font character-set name as an alternative to the coded-font name. There is no default value.

The cell-based, family-4 device tokens listed in Appendix C, “Device tokens supplied by GDDM” on page 367 specify code page 500 for use with PSF, or code page 361 for use with VM3812. You might need to change the code page to suit your local printers. An additional set of sample tokens that use a variety of CECP and APL code pages is provided in ADMLSYS4. These sample tokens have names BCxxxQ or BCxxxA4, where xxx is the code page identifier.

FONTCHR=*(name,xsize,ysize)*

name defines the font-character-set name, which can be used together with the code-page name as an alternative to the coded-font name. There is no default value. *xsize* and *ysize* are the dimensions of the font character box. If they are not specified, a default value is calculated using the values specified on the SIZE and CELLS parameters.

DBCSFNT=*(name,xsize,ysize)*

name defines the coded-font name to be used for DBCS alphanumerics. There is no default value. *xsize* and *ysize* are the dimensions of the font character box. If they are not specified, a default value is calculated using the values specified on the SIZE and CELLS parameters. This font can be defined only as a coded-font name. Users defining their own tokens must ensure that the font they choose is actually available at the destination printer.

GOCA=0|2

Defines the level (0 or 2) of Graphics Object Content Architecture (GOCA) graphics supported. The default value is 0, which means GOCA is not supported.

IM=0|1

Defines the level (0 or 1) of IM (uncompressed image) supported. The default value is 1. A value of 0 means no support. The IOCA parameter overrides any setting of this parameter.

IOCA=0|1

Defines the level (0 or 1) of Image Object Content Architecture (IOCA) supported. The default value is 0, which means no support. This parameter overrides any setting of the IM parameter.

PTOCA=0|1|2
This defines the level (0, 1, or 2) of Presentation Text Object Content Architecture (PTOCA) supported. The default value is 2. A value of 0 means no support. PTOCA level 2 provides more function than PTOCA level 1.

LINEW=*pels*
The rastered line width in pixels. The default is calculated as the closest value to 1/100 inch. For example, this is 2 pixels wide for 240 ppi printers. This parameter applies to rastered graphics only.

Example of the ADMMAFP macro

This example defines an AFPDS device token for an IBM 4028 laser printer. It supports the printing of GDDM alphanumerics, graphics, and image on a paper area of 8 × 10.5 inches.

```
LASER    ADMMAFP 4028,
                 RES=300,
                 SIZE=(80,105,TENTHS),
                 CELLS=(80,84)
                 GOCA=2,
                 IOCA=1,
                 PTOCA=2
```

Creating your own PostScript device tokens with the ADMMPSCR macro

The ADMMPSCR macro is used to define device tokens suitable for PostScript printers in the ADMLSYS4 system definition module. The operands are:

Type
Device type.

Possible values are:

PS1M	PostScript level 1 monochrome
PS2M	PostScript level 2 monochrome
PS1C	PostScript level 1 color
PS2C	PostScript level 2 color

Res
Resolution (xres,yres)

Units are pixels per inch (ppi).

The default resolution is 300 x 300 ppi

Size
Default media size (width, depth, units)

Valid units are:

- PELS (Pixels)
- TENTHS (tenths of an inch)
- MM (millimeters)

The default size is ISO A4 size: 210 x 297 mm.

cells
Alphanumeric page size (cols,rows)

Specifies the approximate number of character columns and rows to be used for positioning alphanumerics on the page.

Default according to paper size assuming 10 columns per inch and 6 lines per inch.

The values specified define the size for any alphanumeric characters output on the page. The actual number of alphanumeric rows and columns is then calculated from the character size and the actual output area size.

GDDM uses the defined character size irrespective of the cell sizes of image symbol sets which might be used for output by the application. If the symbol set cell size does not match the character size the symbol set characters are scaled to fit.

PNAME
Printer name (default the same as Type)

Not used.

PDEST
Printer destination/origin id (default 0)

Not used.

LINEW
PostScript output line width (default 2)

Affects the “boldness” of the PostScript output. The value can be in the range 1 through 9. GDDM resets the value to the default (2) if the value specified is outside this range.

End of General-use programming interface

Chapter 11. GDDM font-emulation and conversion tables

General-use programming interface

In Chapter 5, “Printing and viewing composite documents” on page 39, GDDM’s composite document print utility (CDPU) is described. GDDM supplies several conversion tables to support the workings of the CDPU. This chapter (the one you are reading) describes each table, giving an explanation of its purpose and instructions for altering it. For information on replacing GDDM default modules that you have altered, see Chapter 19, “Updating GDDM default modules” on page 213.

GDDM font-emulation table, ADM4FONT

The GDDM font-emulation table contains information about the fonts that GDDM uses to emulate composite documents on display devices. The font-emulation table defines:

- The font to be emulated
- Which GDDM symbol set is to be used
- The character width and height
- The font weight (interpreted as color on display devices)
- Whether the spacing of the font is proportional or fixed
- The degree of shear to be applied
- Whether the font is a single-byte or double-byte character set

The GDDM font-emulation table is a module that is link-edited with GDDM. The source of the table is supplied so that you can add or change entries if necessary.[8]

Changing the ADM4FONT table

ADM4FONT contains an invocation of the ADMMFONT macro for each font that is to be used for emulation of composite documents on display devices. The syntax of the macro varies, depending on whether the font in question is identified by the name of a CDPDS font, the name of a coded font, the name of a code page, or a numeric code-page identifier.

For a CDPDS font, a coded font, or the name of a code page, the syntax of the macro is:

```
fontname ADMMFONT  SSETNAM=,       a GDDM vector symbol set name
                   CWIDTH=,        in units of 1/1440th inch
                   CHEIGHT=,       in units of 1/1440th inch
                   WEIGHT=,        MEDIUM | BOLD | 1 through 9
                   SPACING=,       FIXED | PROPORTIONAL
                   SHEAR=,         float between 0.0 and 1.0
                   CECP=,          CECP identifier
                   DBCS=           DBCS indicator
```

8 If you are migrating to GDDM 3.2 and modified an earlier version of the font-emulation table, you should reapply any local modifications to the GDDM 3.2 version of ADM4FONT.

For a numeric code-page identifier, the syntax of the macro is:

```
ADMMFONT  CODEID=,        a numeric code-page identifier
          SSETNAM=,       a GDDM vector symbol set name
          CWIDTH=,        in units of 1/1440th inch
          CHEIGHT=,       in units of 1/1440th inch
          WEIGHT=,        MEDIUM | BOLD | 1 through 9
          SPACING=,       FIXED | PROPORTIONAL
          SHEAR=,         float between 0.0 and 1.0
          CECP=,          CECP identifier
          DBCS=           DBCS indicator
```

fontname
An 8-character label that identifies the CDPDS font, coded font, or code page being defined.

Notes:

1. Invocations of ADMMFONT should be coded in ascending order of *fontname* label or CODEID parameter.
2. To enable both the unrotated and rotated versions of a font to be specified with a single macro invocation, the *fontname* label can have a wild card as its second character. For example, the value

   ```
   C#A05500  ADMMFONT
   ```

   defines font-emulation-table entries for C1A05500, C2A05500, C3A05500, and C4A05500.

   It is possible to override some or all of the entries specified with a wild card by explicitly defining a given font. In the GDDM-supplied module ADM4FONT, this is done for fonts C1Z05640, C2Z05640, C3Z05640, and C4Z05640.
3. Use either the *fontname* label or the CODEID parameter, but not both.

CODEID=*'code-page identifier'*
This is provided as an alternative syntax to the *fontname* syntax to enable fonts with a numeric name to be specified. The GDDM-supplied ADM4FONT contains an example of its use for fonts 163, 84, 86, and 87.

SSETNAM=ADMDVSS|*'vector-symbol-set name'*
Name (up to 8 characters) of a GDDM vector-symbol set to be used for this font. For DBCS, if blank the default DBCS vector symbol set is used.

CWIDTH=200|*character-box width*
The width of the character box for the font. The default is equivalent to 10-point text.

CHEIGHT=200|*character-box height*
The height of the character box for the font. The default is equivalent to 10-point text.

WEIGHT= MEDIUM|BOLD|1|2|3|4|5|6|7|8|9
The weight of the font. This is interpreted as a GDDM color when the text is reproduced on display devices. **MEDIUM** displays as turquoise, and **BOLD** displays as neutral. For both types of invocation, the `WEIGHT=` values correspond to GDDM colors 1 through 9.

SPACING= PROPORTIONAL| FIXED
The spacing of the font. This attribute, whether specified explicitly or by default, overrides the characteristics of the named GDDM symbol set.

SHEAR= 0.0
The shear (specified as a quoted floating point value) to be applied to the character. For a description of how GDDM interprets shear, see the description of the GSCH call in the *GDDM Base Application Programming Reference* book. The shear specified forms the *dx* component of the shear. The *dy* component is always 1.0.

CECP=*CECP identifier*
The code-page identifier corresponding to the coded-font or code-page name specified for the fontname label or the CODEID value.

DBCS=0|1
The DBCS indicator, which must be set to 1 for DBCS fonts. The indicator defaults to 0, which represents SBCS symbol sets.

GDDM AFPDS-to-IPDS conversion table, ADMDKFNT

GDDM uses the AFDPS-to-IPDS conversion table to convert information from the AFPDS font, code page, or coded-font format into a suitable IPDS format, so that the CDPU can print AFPDS documents on IPDS printers. The conversion table supplied with GDDM is suitable for most applications. You need not change it unless you have special requirements.

The conversion table identifies:

- The AFPDS font of the source data
- The model number of the IPDS printer for which this table is valid
- The numeric identifier of the target IPDS font
- The IPDS code-page number
- IPDS-font variables, such as font weight and width, and whether the font is wide, italic, or double-struck.

The AFPDS-to-IPDS font conversion table is a module (ADMDKFNT) that is link-edited with GDDM. The source of this module is supplied so that you can add or change entries if necessary. The comments in the source of ADMDKFNT include some guidance on how to obtain the closest approximation to AFPDS output on an IPDS printer. In principle, it is necessary to use only those AFPDS fonts, code pages, and coded fonts that can be converted into exact IPDS equivalents. Where this is not possible, some characters might not print correctly.

Changing the ADMDKFNT table

ADMDKFNT contains an invocation of the ADMMKFNT macro for each AFPDS font. The syntax of the macro varies, depending on whether the APFDS font is identified by the name of a font, a code page, or coded font, or by a numeric code-page identifier.

For named fonts, coded fonts, or code pages, the syntax of the macro is:

```
afpds ADMMKFNT  MODEL=,         IPDS printer model number
                FONT=,          IPDS font number
                FWIDTH=,        IPDS font width
                CDPG=,          IPDS code page number
                BOLD=,          YES|NO
                ITALIC=,        YES|NO
                WIDE=,          YES|NO
                DOUBLE=         YES|NO
```

For numeric code-page identifiers, the syntax of the macro is:

```
      ADMMKFNT  CODEID=,        a numeric code-page identifier
                MODEL=,         IPDS printer model number
                FONT=,          IPDS font number
                FWIDTH=,        IPDS font width
                CDPG=,          IPDS code page number
                BOLD=,          YES|NO
                ITALIC=,        YES|NO
                WIDE=,          YES|NO
                DOUBLE=         YES|NO
```

afpds
The name of the AFPDS font, code page, or coded font.

Notes:

1. Invocations of ADMMKFNT must be coded in ascending order of the *afpds* label or the CODEID parameter.
2. To enable both the unrotated and rotated versions of an AFPDS resource to be specified with a single macro invocation, the *afpds* label can have a wild card as its second character. For example, the value

   ```
   C#A05500  ADMMKFNT
   ```

   defines conversion-table entries for C1A05500, C2A05500, C3A05500, and C4A05500.

   It is possible to override some or all of the entries specified with a wild card by explicitly defining a given AFPDS resource.
3. Use either the *afpds* label or the CODEID parameter, but not both.

CODEID=*'code-page identifier'*
This is provided as an alternative syntax to enable fonts with a numeric name to be specified. The GDDM-supplied ADMDKFNT contains an example of its use for fonts 163, 84, 86, and 87.

MODEL=3812|*model number*
The IPDS printer model number for which this table entry is valid (for example, 3812, 4224). MODEL=3812 is valid for the 3816 printer.

FONT=85|*font number*
The IPDS font number (decimal) that is to be used instead of the AFPDS name. The default IPDS font is Courier 12.

FWIDTH=*font width*
A number in the range 1 through 32766 that is passed to the IPDS printer to define the font width. The default value depends on the font used. Units are 1440ths of an inch.

CDPG=500|*code-page number*
The IPDS code-page number (decimal) that is to be used instead of the AFPDS name.

WIDE=NO|YES
Specifies whether the IPDS font is printed using wide characters.

ITALIC=NO|YES
Specifies whether the IPDS font is printed using italic characters.

BOLD=NO|YES
Specifies whether the IPDS font is printed using bold characters.

DOUBLE=NO|YES
Specifies whether the IPDS font is printed twice (that is, whether each character is double-struck).

GDDM code-page-identifier conversion table, ADM4CPID

GDDM uses the code-page-identifier conversion table to convert character-set and code-page identifiers in the coded-font name (GRID) in CDPDS documents into PSF-format code-page identifiers. This table is used only when GDDM is printing CDPDS documents on family-4 printers.

For DBCS fonts, ADM4CPID does not return a PSF-format code-page identifier, but indicates that the character-set and code-page combination is a DBCS font, and that ADM4CFID is to be used.

The GDDM code-page-identifier conversion table is a module, ADM4CPID, that is link-edited with GDDM. The source of this module is supplied so that you can add or change entries if necessary.

Changing the ADM4CPID table

ADM4CPID contains an invocation of the ADMMCFNT macro for each code page to be converted. The syntax of the macro is:

```
label  ADMMCFNT  CODEID=,       character set and code page
                 NAME=,         PSF codepage name
                 DBCS=          Optional DBCS indicator
```

label
This can be any valid macro label. It is ignored.

CODEID=*'character set / code page'*
An 8-character string representing 4 bytes of hexadecimal data. The first 2 bytes are the graphics-character-set global identifier (GCSGID). The remaining 2 bytes are the code-page global identifier (CPGID). GDDM matches this against bytes 0 through 3 of the coded font name in the MCF/2 structured field in the CDPDS document.

For a description of the MCF/2 structured field, see the *GDDM Base Application Programming Reference* book.

NAME=*PSF-code-page name*
The 8-character name of the PSF code page to be used by GDDM for family-4 output. If this value identifies a DBCS code page, the name of the font is in the table ADM4CFID.

DBCS=*DBCS indicator*
The DBCS indicator, which must be set to 1 for DBCS code-page combinations.

GDDM font-global-identifier conversion table, ADM4FGID

GDDM uses the font-global-identifier conversion table to convert the font global identifiers (FGIDs) in CDPDS documents into PSF-format font global identifiers. This table is used only when GDDM is printing CDPDS documents on family-4 printers.

The GDDM font-global-identifier conversion table is a module, ADM4FGID, that is link-edited with GDDM. The source of this table is supplied so that you can change or add entries if necessary.

Changing the ADM4FGID table

ADM4FGID contains an invocation of the ADMMCFNT macro for each font global identifier to be converted. Entries in the table are generated by coding invocations of the ADMMCFNT macro. The syntax of the macro is:

```
label  ADMMCFNT  CODEID=,       font global identifier and width
                 NAME=          PSF font member name
```

label
This can be any valid macro label. It is ignored.

CODEID=*'font global identifier and width'*
An 8-character string representing 4 bytes of hexadecimal data. The first 2 bytes are the font global identifier (FGID). The remaining 2 bytes give the character width. GDDM matches this value against bytes 4 through 7 of the coded font name in the MCF/2 structured field.

For a description of the MCF/2 structured field, see the *GDDM Base Application Programming Reference* book.

NAME=*PSF font-member name*
The 8-character name of the PSF font member to be used by GDDM for family-4 output.

GDDM DBCS code-page and font-conversion table, ADM4CFID

GDDM uses the font- and code-page-conversion table, ADM4CFID, to translate a coded-font name (GRID) in a CDPDS document into an 8-byte, PSF-coded-font name. This table is used, together with ADM4FONT, when emulating an AFPDS file that contains a PSF coded-font name that identifies a DBCS font. However, this table is used only if the code-page and character-set combination that indicates that the font is DBCS has been added to the code-page-conversion table ADM4CPID.

Each 16-byte entry is as follows:

Field name	Field offset	Field length	IBM default value
CFNAME	0	8	X'0172012CD1370078'
PSFNAME	7	8	X0M32F

CFNAME The coded font name, comprising 8 bytes of hexadecimal data:

Bytes	Description
0–1	Graphic-character-set global identifier (GCSGID)
2–3	Code-page global identifier (CPGID)
4–5	Font global identifier (FGID)
6–7	2-byte character-width field.

GDDM matches this value against bytes 0 through 7 of the coded-font name in the MCF/2 structured field when translating a coded-font name into a PSF-coded-font name.

For a description of the MCF/2 structured field, see the *GDDM Base Application Programming Reference* book.

PSFNAME The 8-character, PSF-coded-font name that is to be used by GDDM for family-4 output. Alternatively, if the font is being emulated, this name is used to map to a GDDM font via the table ADM4FONT.

GDDM also matches this name against the PSF-coded-font name in an AFPDS file when emulating AFPDS documents on the display device.

Adding new DBCS fonts

If you want to define DBCS fonts that can be recognized by the CDPU (in addition to those supplied by GDDM 3.2), you must update some of the tables described in this chapter:

1. In ADM4CFID, add the coded-font name (GRID) from a CDPDS document's Map Coded Font/2 (MCF/2) and the associated coded-font name used in an AFPDS document's Map Coded Font (MCF/1).

   If the coded-font name appears in an AFPDS file and is to be used for emulation on the screen only, a correct GRID entry is not necessary. The entry must, however, be unique, and the character-set and code-page combination must be one that is recognized in ADM4CPID.

2. Add the character set and code page (part of a GRID) to ADM4CPID if necessary, and set the DBCS indicator.

3. If the font is to be emulated on the screen, add the AFPDS document's coded-font name to ADM4FONT, and set the DBCS indicator. The GDDM symbol-set name can either be set to blank, in which case the GDDM default DBCS vector symbol set is used, or the required DBCS vector symbol-set name can be specified (for example, SSETNAM=ADMVQ).

End of General-use programming interface

Chapter 12. The GDDM default symbol sets

General-use programming interface

GDDM supplies a large number of sample image and vector symbol sets, which are used to display characters, to shade areas, and to draw markers. A complete list of the supplied sample symbol sets can be found in the *GDDM Base Application Programming Reference* book.

Symbol sets can be explicitly loaded by an application, or the *default symbol sets* can be used: GDDM uses a subset of the supplied sample symbol sets as defaults. GDDM uses the default symbol sets from load modules (MVS), phases (VSE), or text decks in ADMGLIB TXTLIB or the GDDM saved segment (VM). Those symbol sets that GDDM uses as defaults are also supplied in source (ADMSYMBL) form, so that you can edit them or use them as the basis for a new symbol set.

This chapter identifies which symbol sets are used as defaults by GDDM, and explains how you can replace them. Table 8 lists the names of the default symbol sets, specifies whether they are vector or image sets, and identifies whether they define text, markers, or shading patterns. Note that, in many cases, the name of the source symbol set is not the same as its load module or phase version.

As a general rule, the last letter of a symbol-set name identifies the size of the character cell. Table 9 on page 133 provides a list of cell sizes in display points, and identifies the devices for which they are suitable.

Table 8 (Page 1 of 3). The default symbol sets: usage, type, and naming conventions

Name within load module or phase	Source name	Type	Usage and size
ADMDHHVN	ADMDVIH *	Vector	Mode 3 text (8 by 16)
ADMDHHVR	ADMDVIH *	Vector	Mode 3 text (12 by 20)
ADMDHIIA	ADMDHHIA	Image	Mode 2 text (9 by 16)
ADMDHIIC	ADMDHHIC	Image	Mode 2 text (9 by 12)
ADMDHIIG	ADMDHHIG	Image	Mode 2 text (10 by 8)
ADMDHIIK	ADMDHHIK	Image	Mode 2 text (20 by 18)
ADMDHIIN	ADMDHHIN	Image	Mode 2 text (8 by 16)
ADMDHIIQ	ADMDHHIQ	Image	Mode 2 text (24 by 30)
ADMDHIIR	ADMDHHIR	Image	Mode 2 text (12 by 20)
ADMDHIIT	ADMDHIIT	Image	Mode 2 text (12 by 16)
ADMDHIMA	ADMDHIMA	Image	Markers (9 by 16)
ADMDHIMC	ADMDHIMC	Image	Markers (9 by 12)
ADMDHIMG	ADMDHIMG	Image	Markers (10 by 8)
ADMDHIMK	ADMDHIMK	Image	Markers (20 by 18)
ADMDHIMN	ADMDHIMN	Image	Markers (8 by 16)
ADMDHIMQ	ADMDHIMQ	Image	Markers (24 by 30)

Table 8 (Page 2 of 3). The default symbol sets: usage, type, and naming conventions

Name within load module or phase	Source name	Type	Usage and size
ADMDHIMR	ADMDHIMR	Image	Markers (12 by 20)
ADMDHIMT	ADMDHIMT	Image	Markers (12 by 16)
ADMDHIMV	ADMDHIMV	Vector	Markers
ADMDHIPA	ADMDHIPA	Image	Patterns (9 by 16)
ADMDHIPC	ADMDHIPC	Image	Patterns (9 by 12)
ADMDHIPG	ADMDHIPG	Image	Patterns (10 by 8)
ADMDHIPJ	ADMDHIPJ	Image	Patterns (32 by 32)
ADMDHIPM	ADMDHIPM	Image	Patterns (32 by 32)
ADMDHIPO	ADMDHIPO	Image	Patterns (32 by 32)
ADMDHIVA	ADMDVSS	Vector	Mode 3 text (9 by 16)
ADMDHIVC	ADMDVSS	Vector	Mode 3 text (9 by 12)
ADMDHIVG	ADMDVSS	Vector	Mode 3 text (10 by 8)
ADMDHIVJ	ADMDHIVJ	Vector	Modes 1, 2 and 3 text
ADMDHIVK	ADMDVSS	Vector	Mode 3 text (20 by 18)
ADMDHIVM	ADMDHIVM	Vector	Modes 1, 2 and 3 text (See Note)
ADMDHIVN	ADMDVSS	Vector	Mode 3 text (8 by 16) (See Note)
ADMDHIVQ	ADMDVSS	Vector	Mode 3 text (24 by 30)
ADMDHIVR	ADMDVSS	Vector	Mode 3 text (12 by 20)
ADMDHIVT	ADMDVSS	Vector	Mode 3 text (12 by 16)
ADMDHIVU	ADMDVSS	Vector	Mode 3 text (plotters)
ADMDHJIA	ADMDHIIA	Image	Mode 2 text (9 by 16)
ADMDHJIC	ADMDHIIC	Image	Mode 2 text (9 by 12)
ADMDHJIG	ADMDHIIG	Image	Mode 2 text (10 by 8)
ADMDHJIK	ADMDHIIK	Image	Mode 2 text (20 by 18)
ADMDHJIN	ADMDHIIN	Image	Mode 2 text (8 by 16)
ADMDHJIQ	ADMDHIIQ	Image	Mode 2 text (24 by 30)
ADMDHJIR	ADMDHIIR	Image	Mode 2 text (12 by 20)
ADMDHJVA	ADMDVECP	Vector	Mode 3 text (9 by 16)
ADMDHJVC	ADMDVECP	Vector	Mode 3 text (9 by 12)
ADMDHJVG	ADMDVECP	Vector	Mode 3 text (10 by 8)
ADMDHJVJ	ADMDHJVJ	Vector	Modes 1, 2 and 3 text
ADMDHJVK	ADMDVECP	Vector	Mode 3 text (20 by 18)
ADMDHJVM	ADMDHJVM	Vector	Modes 1, 2 and 3 text (See Note)
ADMDHJVN	ADMDVECP	Vector	Mode 3 text (8 by 16) (See Note)
ADMDHJVQ	ADMDVECP	Vector	Mode 3 text (24 by 30)

Table 8 (Page 3 of 3). The default symbol sets: usage, type, and naming conventions

Name within load module or phase	Source name	Type	Usage and size
ADMDHJVR	ADMDVECP	Vector	Mode 3 text (12 by 20)
ADMDHJVT	ADMDVECP	Vector	Mode 3 text (12 by 16)
ADMDHJVU	ADMDVECP	Vector	Mode 3 text (plotters)

Note: * means equivalent to a symbol set built into the device

Those load modules or phases whose names begin with the letters ADMDHI contain symbol sets for non-CECP processing; those whose names begin with the letters ADMDHJ are CECP sets and are tagged with country extended code page 00037. (For a description of GDDM code-page support, see Chapter 16, “GDDM’s code-page support” on page 159.)

Table 9 (Page 1 of 2). Symbol sets: cell sizes and suitable devices

Last letter	Matrix size	Monochrome or multicolor	Suitable devices
A	9 x 16	Monochrome	3278 Display Station Models 2 and 3 8775 Display Terminal Models 1 and 11 8775 Display Terminal Models 2 and 12 (can be either 9 x 16 or 9 x 12) 3270 PC Workstation (symbol size is 9 x 14) 3270 PC/G Workstation 3290 Information Panel display (all models) 3472-G InfoWindow Display
C	9 x 12	Monochrome	3278 Display Station Model 4 8775 Display Terminal Models 2 and 12 (can be either 9 x 16 or 9 x 12) IBM PC EGA high-resolution display adapter IBM Personal System/2 display adapter
		Multicolor	3179-G Color Display Station 3192 Color Graphics Display Station, Model G
D	9 x 12	Multicolor	3179 Color Graphics Display Station, Model G 3279 Color Display Station (all models) 3192 Color Graphics Display Station, Model G 3472 InfoWindow, Model G 3472 InfoWindow, Model M IBM PC CGA display adapter IBM PC EGA low-resolution display adapter
E	9 x 10	Monochrome	3270 PC/G alphanumerics only
G	10 x 8	Monochrome	3268, 3287 Printers (all models) 3112 Printer 3116 Printer 3912 Printer 3916 Printer
H	10 x 8	Multicolor	3268, 3287 Printers (all models)
J	32 x 32	Monochrome	4250 Printer (high-resolution symbol sets, 400 pixels per inch or greater)
K	20 x 18	Monochrome	4224 Printer

Table 9 (Page 2 of 2). Symbol sets: cell sizes and suitable devices

Last letter	Matrix size	Monochrome or multicolor	Suitable devices
L	32 x 32	Monochrome	3800 Printing Subsystems (medium-resolution symbol sets, less than 400 pixels per inch)
M	32 x 32	Monochrome	3800 Printing Subsystems (medium-resolution symbol sets, less than 400 pixels per inch)
N	8 x 16	Multicolor	3270 PC/G, all models (graphics only)
O	32 x 32	Monochrome	3800 Printing Subsystems (low-resolution symbol sets, less than 100 pixels per inch)
Q	24 x 30	Monochrome	3812 Pageprinter Model 2 3816 Printer
R	12 x 20 12 x 24	Multicolor	3270 PC/GX (all models) 5550 Multistation IBM Personal System/2 8514/A display adapter
S	9 x 21	Monochrome	3472-G InfoWindow Display
T	12 x 16	Monochrome	4234 Dot Band Printer

If a device has a cell size that is not included in Table 9 on page 133, GDDM selects a set whose cell size is the smallest that could contain that of the device in question. For example, for a device with a cell size of 9 by 14, such as the 3270 PC, GDDM selects an image symbol set with a cell size of 9 by 16 (last letter A). However, this means that the 3270 PC can suffer from bars across shaded patterns, which causes a “venetian blind” effect.

Editing symbol sets

To edit any vector symbol set, you use the GDDM-PGF Vector Symbol editor, which is described in the *GDDM-PGF Vector Symbol Editor* book. For illustrations of the supplied vector symbol sets, see the *GDDM Base Application Programming Reference* book. To edit any image symbol set, you use the Image Symbol Editor, which is described in the *GDDM Using the Image Symbol Editor* book.

Before using the Vector or Image Symbol Editor, ensure that the current application code page is the same as that of the symbol set being edited. If a new symbol set is created, or if an existing symbol set with a code page equal to the installation code page is edited, the application code page can be allowed to default. However, if the symbol set to be edited is not defined with the current installation code page, the application code page must be set explicitly before the editor is invoked. To set the application code page, use the APPCPG external default:

```
ADMMDFT APPCPG=nnnnn
```

where `nnnnn` is the code-page identifier of the symbol set to be edited. If you fail to do this, incorrect symbols are likely to be displayed during editing and, when the symbol set is saved, it is tagged with the wrong code page.

Note: All GDDM CECP symbol sets are supplied in the order of code page 00037: before you edit any of the supplied CECP symbol sets, set the application code page to 00037.

Where symbol sets are held

In the VM environment, the source (editable) versions of the symbol sets have the filetype ADMSYMBL. The source of a symbol set can be accessed by entering its name on invoking the appropriate symbol set editor. The deck formats generated by the symbol set editors have a default filetype of ADMDECK. The default symbol sets used by GDDM are TEXT decks contained in the ADMGLIB TXTLIB or the GDDM saved segment. If any default symbol set is to be changed, its TXTLIB or saved-segment version must be replaced.

In the MVS and VSE environments, the load module or phase versions are held in the load modules or phases identified in Table 10. The source of a symbol set can be accessed by entering its name on input to the appropriate symbol set editor. If any default symbol set is to be changed, its load module or phase version must be replaced.

Table 10 (Page 1 of 2). MVS and VSE: load modules and phases containing the default symbol sets

Name of symbol set	Load module or phase containing the symbol set
ADMDHHVN	ADMDHI1N
ADMDHHVR	ADMDHI1R
ADMDHIIA	ADMDHI0A, ADMDHI0B
ADMDHIIC	ADMDHI0C, ADMDHI0D, ADMDHI1C
ADMDHIIG	ADMDHI0G, ADMDHI0H
ADMDHIIK	ADMDHI0K
ADMDHIIN	ADMDHI0N, ADMDHI1N
ADMDHIIQ	ADMDHI0Q
ADMDHIIR	ADMDHI0R, ADMDHI1R
ADMDHIIT	ADMDHI0T
ADMDHIMA	ADMDHI0A, ADMDHI0B
ADMDHIMC	ADMDHI0C, ADMDHI0D, ADMDHI1C
ADMDHIMG	ADMDHI0G, ADMDHI0H
ADMDHIMK	ADMDHI0K
ADMDHIMN	ADMDHI0N, ADMDHI1N
ADMDHIMQ	ADMDHI0Q
ADMDHIMR	ADMDHI0R, ADMDHI1R
ADMDHIMT	ADMDHI0T
ADMDHIMV	ADMDHI0J, ADMDHI0L, ADMDHI0O, ADMDHI0U
ADMDHIPA	ADMDHI0A, ADMDHI0B
ADMDHIPC	ADMDHI0C, ADMDHI0D, ADMDHI0N
ADMDHIPG	ADMDHI0G, ADMDHI0H
ADMDHIPJ	ADMDHI0J
ADMDHIPM	ADMDHI0L
ADMDHIPO	ADMDHI0O

Table 10 (Page 2 of 2). MVS and VSE: load modules and phases containing the default symbol sets

Name of symbol set	Load module or phase containing the symbol set
ADMDHIPU	ADMDHI0U
ADMDHIVA	ADMDHI0A, ADMDHI0B
ADMDHIVC	ADMDHI0C, ADMDHI0D, ADMDHI1C
ADMDHIVG	ADMDHI0G, ADMDHI0H
ADMDHIVJ	ADMDHI0J
ADMDHIVK	ADMDHI0K
ADMDHIVM	ADMDHI0O
ADMDHIVN	ADMDHI0N
ADMDHIVQ	ADMDHI0Q
ADMDHIVR	ADMDHI0R
ADMDHIVT	ADMDHI0T
ADMDHIVU	ADMDHI0U
ADMDHJIA	ADMDHI0A, ADMDHI0B
ADMDHJIC	ADMDHI0C, ADMDHI0D, ADMDHI1C
ADMDHJIG	ADMDHI0G, ADMDHI0H
ADMDHJIK	ADMDHI0K
ADMDHJIN	ADMDHI0N, ADMDHI1N
ADMDHJIQ	ADMDHI0Q
ADMDHJIR	ADMDHI0R, ADMDHI1R
ADMDHJVA	ADMDHI0A, ADMDHI0B
ADMDHJVC	ADMDHI0C, ADMDHI0D, ADMDHI1C
ADMDHJVG	ADMDHI0G, ADMDHI0H
ADMDHJVJ	ADMDHI0J
ADMDHJVK	ADMDHI0K
ADMDHJVM	ADMDHI0L, ADMDHI0O
ADMDHJVN	ADMDHI0N
ADMDHJVQ	ADMDHI0Q
ADMDHJVR	ADMDHI0R
ADMDHJVT	ADMDHI0T
ADMDHJVU	ADMDHI0U

Replacing or editing the default symbol sets

If you are using GDDM's country-extended code page (CECP) support, you can replace the national language default symbol set by using a GDDM nickname user-default specification. To replace any other default symbol set, you must overwrite the executable copy of the symbol set.

The remainder of this section discusses how to replace the default national language symbol set, and how to edit and replace any of the default symbol sets.

Replacing the default national language vector symbol set for CECP processing

GDDM's CECP support simplifies the provision of default vector symbol sets for national language use. If you specify a CECP as your installation and application code page, your application programs, including the GDDM utility programs, produce the correct national-use characters, without any further action on your part.

The external defaults you use to specify the installation and application code pages are:

```
ADMMDFT INSCPG=nnnnn,APPCPG=nnnnn
```

where `nnnnn` is the code-page number. For more information about setting the INSCPG and APPCPG external defaults, see Chapter 16, "GDDM's code-page support" on page 159.

Replacing the default national language vector symbol set for non-CECP processing

A national-language vector symbol set can be made the default vector symbol set for non-CECP processing by using the Vector Symbol Editor to generate a deck version of the symbol set and replacing the default version.

An example of how to do this is shown in "Instructions for editing and replacing a default symbol set" on page 138.

A list of the non-CECP, national-language symbol sets is shown in Table 11.

Table 11 (Page 1 of 2). National language vector symbol sets

Name	Description
ADMDVSSB	Brazilian
ADMDVSSD	Danish
ADMDVSSE	English (U.K.)
ADMDVSSF	French
ADMDVSSG	German
ADMDVSSI	Italian
ADMDVSSJ	Japanese (Latin) Extended
ADMDVSSK	Japanese (Katakana)
ADMDVSSN	Norwegian
ADMDVSSS	Spanish

Table 11 (Page 2 of 2). National language vector symbol sets

Name	Description
ADMDVSSV	Swedish

The following image symbol sets also support Katakana:

Name	Cell size	Description
ADMDISJN	8 × 16	Japanese (Latin) Extended for 16 by 16 5550 (OS/2-J)
ADMDISJR	12 × 24	Japanese (Latin) Extended for 24 by 24 5550 (OS/2-J)
ADMDISKA	9 × 16	Katakana
ADMDISKC	9 × 12	Katakana
ADMDISKG	10 × 8	Katakana
ADMDISKN	8 × 16	Katakana for 16 by 16 5550
ADMDISKR	12 × 24	Katakana for 24 by 24 5550

Note that there are source versions, but no load module or phase versions, of the supplied double-byte character sets. The default DBCS sets are ADMIKxx (image) and ADMVKxx (vector). To change the default DBCS symbol sets, use the DBCSDNM external default, which is described in Appendix A, “External defaults” on page 307. For information on how to construct a double-byte data stream using the default DBCS symbol sets, see the *GDDM Base Application Programming Guide*.

End of General-use programming interface

Instructions for editing and replacing a default symbol set

This example shows how to make the French vector symbol set the default, non-CECP vector symbol set. However, these instructions can be followed for any of the symbol sets, including the remaining non-CECP national-language sets.

Note that the ADMDVSS vector symbol set has nine load module or phase format names, but only one editable format name. All of the load module or phase formats are identical.

1. Start the GDDM-PGF Vector Symbol Editor. How you do this varies according to subsystem.
2. Start editing the French symbol set by entering its name in the symbol-set field:

   ```
   Symbol Set ==>  ADMDVSSF
   ```

3. Save the symbol set in deck format.

   Note that the ADMDVSS vector symbol set has nine different load-module or phase names, but a single name (ADMDVSS) for its source format. Therefore, to save the symbol set in deck format, you:

   a. Rename the set to ADMDHIVA using the command `RENAME ADMDHIVA`
   b. Save the deck ADMDHIVA using the command `SAVE DECK`
   c. Repeat steps a and b for the names ADMDHIVC, ADMDHIVG, ADMDHIVK, ADMDHIVN, ADMDHIVQ, ADMDHIVR, ADMDHIVT, and ADMDHIVU

d. Exit the vector symbol editor by pressing PF3

4. You must now replace the existing versions of the symbol sets. For more information, see Chapter 19, “Updating GDDM default modules” on page 213.
5. To ensure that GDDM uses the changed default symbol set, specify the LOADDSYM processing option:

```
ADMMNICK PROCOPT=((LOADDSYM,YES))
```

Defining additional pattern sets for GDDM-PGF ICU users

General-use programming interface

If you want to set up an extended pattern set to be generally available to ICU users in an installation, create a user-defined pattern set with the name ADMICUPx, where “x” can be either “D” or “C,” depending on the device type in use.

For example, the sample 64-color shading pattern set supplied on the distribution tape under the name ADMCOLSD need only be copied under the name ADMICUPD to become available to ICU users with pattern values 65 through 128. The copying can be done with the Copy command of the ICU Directory panel. For further information on setting up your own pattern sets, see the *GDDM-PGF Programming Reference* book.

End of General-use programming interface

Chapter 13. Translation of user-defined shading patterns

General-use programming interface

The GDDM Base API call GSPAT sets the current shading pattern. (The “current shading pattern” is the one used to fill graphics areas at the time they are drawn.) The only parameter to GSPAT is a number that identifies the shading pattern to be used:

- If a number in the range 1 through 16 is specified, one of the GDDM-supplied shading patterns becomes the current shading pattern. The GDDM-supplied patterns are supported on all devices supported by GDDM. (Note, however, that the supplied patterns might not appear the same on all devices.)
- If a number in the range 65 through 254 is specified, a *user-defined* pattern is used. A user-defined pattern must have been defined in a pattern set and loaded before it can be used.

User-defined patterns are supported only on devices that support loadable pattern sets. Some devices, such as IPDS printers, ASCII graphics displays, and devices running GDDM-PCLK or GDDM-OS/2 Link, do not support user-defined patterns and show such areas as solid black.

The ADMDGTRN default module

The GDDM default module ADMDGTRN contains three tables that substitute GDDM-supplied shading patterns for user-defined shading patterns. This is provided for devices that do not support loadable pattern sets. ADMDGTRN contains a default table, ADM00001, and two sample tables, ADM00002 and ADM00003. You specify which of the tables in ADMDGTRN is to be used for translation of user-defined shading patterns on the PATTRAN processing option, as follows:

```
ADMMNICK PROCOPT=((PATTRAN,1,1))
```

This nickname UDS selects the default table, ADM00001. PATTRAN is described in Appendix B, “Processing options” on page 333. If the PATTRAN processing option is not specified, or if a table identifier of “0” is specified, no translation occurs.

The default table (ADM00001) is shown on page 142. If another table is used and that table does not specify equivalents for all user-defined patterns, a default equivalent is taken from table ADM00001.

Table ADM00001

The default table is structured as follows:

Table 12. GDDM-supplied translation table for user-defined shading patterns, ADM00001

User-defined shading patterns	Target shading pattern
65, 81, 97, 113, 129, 145, 161, 177, 193, 209, 225, 241	1
66, 82, 98, 114, 130, 146, 162, 178, 194, 210, 226, 242	2
67, 83, 99, 115, 131, 147, 163, 179, 195, 211, 227, 243	3
68, 84, 100, 116, 132, 148, 164, 180, 196, 212, 228, 244	4
69, 85, 101, 117, 133, 149, 165, 181, 197, 213, 229, 245	5
70, 86, 102, 118, 134, 150, 166, 182, 198, 214, 230, 246	6
71, 87, 103, 119, 135, 151, 167, 183, 199, 215, 231, 247	7
72, 88, 104, 120, 136, 152, 168, 184, 200, 216, 232, 248	8
73, 89, 105, 121, 137, 153, 169, 185, 201, 217, 233, 249	9
74, 90, 106, 122, 138, 154, 170, 186, 202, 218, 234, 250	10
75, 91, 107, 123, 139, 155, 171, 187, 203, 219, 235, 251	11
76, 92, 108, 124, 140, 156, 172, 188, 204, 220, 236, 252	12
77, 93, 109, 125, 141, 157, 173, 189, 205, 221, 237, 253	13
78, 94, 110, 126, 142, 158, 174, 190, 206, 222, 238, 254	14
79, 95, 111, 127, 143, 159, 175, 191, 207, 223, 239	15
80, 96, 112, 128, 144, 160, 176, 192, 208, 224, 240	16

Table ADM00002

The sample table ADM00002 translates user-defined patterns 65 through 128 in the symbol set ADMPATTC to similar system patterns. Where two or more user-defined patterns are translated to the same system pattern, a different color translation is used to distinguish between them. User-defined patterns 129 onwards are translated to system patterns, as in table ADM00001, without color translation.

Table 13. GDDM-supplied sample table for translation of user-defined shading patterns, ADM00002

User Pattern	System Pattern	Color	User Pattern	System Pattern	Color	User Pattern	System Pattern	Color
65	1	1	87	13	1	108	2	1
66	3	1	88	14	1	109	1	7
67	8	1	89	14	2	110	9	7
68	1	2	90	14	3	111	2	3
69	3	2	91	7	1	112	6	3
70	3	3	92	5	2	113	8	4
71	5	1	93	6	5	114	8	5
72	1	3	94	4	1	115	8	6
73	6	1	95	4	2	116	8	7
74	1	4	96	13	3	117	7	2
75	8	2	97	1	6	118	10	5
76	1	5	98	6	2	119	7	3
77	9	1	99	10	4	120	7	4
78	2	4	100	8	3	121	7	5
79	10	1	101	4	3	122	5	5
80	10	2	102	5	3	123	4	4
81	10	3	103	11	1	124	5	6
82	9	3	104	12	4	125	4	5
83	9	4	105	11	5	126	6	4
84	9	5	106	2	6	127	2	5
85	9	6	107	5	4	128	4	6
86	13	2						

Table ADM00003

Table ADM00003 translates user-defined patterns 65 through 254 to the default system pattern with a change in color. This table is intended for use with GDDM-OS/2 Link. User-defined patterns 65 through 254 are translated to system pattern 0.

Editing the contents of module ADMDGTRN

The source of module ADMDGTRN contains one or more invocations of the ADMMTRAN macro for each table. The syntax of macro ADMMTRAN is:

```
name ADMMTRAN user-pattern, target-pattern, target-color
```

name

Is a user-supplied name to mark the start of a new table. If no name is specified, this invocation of ADMMTRAN is taken to be a continuation of the previous table. The name must be of the form ADMnnnnn, where nnnnn is any number in the range 00001 through 99999.

user-pattern
Is the number of a user-defined pattern in the range 65 through 254.

target-pattern
Is the pattern number to which the user-defined pattern is to be translated:

0 the drawing default.

1-16 the GDDM-defined pattern.

target-color
Is the color number to which the user-defined pattern is to be translated. Possible values are included in the description of the GSCOL parameter in the *GDDM Base Application Programming Reference* book.

A color specified in the translation table overrides any color specified on a GSCOL call. If no color is specified in the translation-table, the application-defined color is used.

At the end of the current area, the color and pattern specified by the application become current again.

For information about replacing updated modules, see Chapter 19, “Updating GDDM default modules” on page 213.

End of General-use programming interface

Chapter 14. Color-separation master tables

General-use programming interface

GDDM page-printer support allows you to create a set of output files that represent the components in a subtractive or additive color-separation process. Several output files are created for every picture: each file represents one of the component colors of the picture and can be used to create the relevant printing plate. The separation process is controlled by a module, ADMDJCOL, (which is constructed using a supplied macro, ADMMCOLT) and an image-pattern set, (which can be constructed using the Image Symbol Editor, ADMISSE). ADMDJCOL contains multiple instances of the ADMMCOLT macro, each of which describes the patterns to be used to represent each color.

This chapter describes the ADMMCOLT macro, and illustrates the contents of the supplied version of the ADMDJCOL module. If you decide to alter ADMDJCOL, refer to Chapter 19, “Updating GDDM default modules” on page 213 for instructions on how to replace the module.

The ADMMCOLT macro

The syntax of the ADMMCOLT macro is:

```
name  ADMMCOLT codes,    START | END | Pattern codes
              SETID=,   Set ID
              PATTERN=, Pattern set
              SETS=,    Number of sets defined
              COLORS=,  Number of colors
              MASTERS=  Number of masters per color
```

codes
The *codes* value must be “START” for the initial invocation of ADMMCOLT and “END” for the final invocation of ADMMCOLT. For all intermediate invocations of ADMMCOLT, *codes* is `(x1,x2,x3,xi,...,xn)`, where `xi` is a 2-digit hexadecimal code that identifies the pattern in the pattern-symbol set to be used by color master `i`, and `n` is the total number of masters.

SETID=*set id*
Name of set. This can be up to 8 characters of the form ADMnnnnn, where nnnnn is in the range 00001 through 99999. GDDM supplies seven sets (ADM00001, ADM00002, ADM00003, ADM00004, ADM00005, ADM00006, ADM00007).

PATTERN=*pattern set*
Name of pattern set.

This must be a monochrome, image-character set having a cell size of 32 × 32 pixels. GDDM supplies a sample called ADMDHIPK.

SETS=*number of sets*
The number of sets defined.

COLORS=*number of colors*
The number of colors defined for this set.

MASTERS=*number of masters per color*
The number of masters to be created for this set.

The ADMDJCOL module

The ADMDJCOL module supplied by IBM provides seven color-master tables. These are shown below. You specify which of these is to be used by setting the COLORMAS processing option. COLORMAS is described in Appendix B, “Processing options” on page 333.

```
ADMDJCOL CSECT  ,
*********************************************************************
*                                                                   *
* DESCRIPTIVE NAME:  GDDM HIGH-RESOLUTION IMAGE GENERATOR           *
*                    DEFAULT COLOR TABLES                           *
*                                                                   *
*                5664-200,5665-356,5666-328                         *
*                (C) COPYRIGHT IBM CORP. 1979, 1986.                *
*                LICENSED MATERIALS - PROPERTY OF IBM               *
*                                                                   *
* FUNCTION:                                                         *
*                                                                   *
*   THIS MODULE GENERATES THE SAMPLE COLOR TABLES FOR THE COLOR     *
*   SEPARATION PROCESS IN FAMILY-4 DEVICE SUPPORT.                  *
*                                                                   *
*   ADMDHIPK PATTERN CODES HAVE THE FOLLOWING MEANING:              *
*   ------------------------------------------------                *
*   CODE X'41' =   0 % (NO COLOR)                                   *
*   CODE X'42' = 100 % (SOLID COLOR)                                *
*   CODE X'43' =  50 % (1ST HALF COLOR)                             *
*   CODE X'44' =  50 % (2ND HALF COLOR)                             *
*   CODE X'45' =  25 % (1ST QUARTER COLOR)                          *
*   CODE X'46' =  25 % (2ND QUARTER COLOR)                          *
*   CODE X'47' =  25 % (3RD QUARTER COLOR)                          *
*   CODE X'48' =  25 % (4TH QUARTER COLOR)                          *
*                                                                   *
*   ADMDHIPL PATTERN CODES HAVE THE FOLLOWING MEANING:              *
*   ------------------------------------------------                *
*                                                                   *
*   THERE ARE 33 SHADES OF GRAY.  THE PATTERN CODES START AT X'41'  *
*   (NO COLOR) AND FINISH AT X'61' (ALL BLACK).  EACH SHADE HAS     *
*   APPROXIMATELY 3% MORE PIXELS SET ON THAN ITS PREDECESSOR.       *
*                                                                   *
*********************************************************************
ADMDJCOL AMODE ANY
ADMDJCOL RMODE ANY
         ADMMCOLT START,SETS=7
```

```
*
*************************************************************************
* TABLE 1.  SUBTRACTIVE COLORS FOR PRINTERS                             *
*************************************************************************
*
ADM00001 ADMMCOLT PATTERN=ADMDHIPK,COLORS=10,MASTERS=4,SETID=ADM00001

*
*               *-------------*
* COLOR MASTER:  * 1  2  3  4  *
* COLOR SEPS:    * YY MM CC BB * (YELLOW, MAGENTA, CYAN, BLACK)
*               *-------------*
*
DEFAULT  ADMMCOLT (41,41,41,42)
BLUE     ADMMCOLT (41,43,44,41)
RED      ADMMCOLT (43,44,41,41)
PINK     ADMMCOLT (41,42,41,41)
GREEN    ADMMCOLT (43,41,44,41)
TURQSE   ADMMCOLT (41,41,42,41)
YELLOW   ADMMCOLT (42,41,41,41)
NEUTRAL  ADMMCOLT (41,41,41,42)
BACKGRD  ADMMCOLT (41,41,41,41)
ALLBLK   ADMMCOLT (42,42,42,42)
*
*************************************************************************
* TABLE 2.  ADDITIVE COLORS FOR DISPLAYS                                *
*************************************************************************
*
ADM00002 ADMMCOLT PATTERN=ADMDHIPK,COLORS=9,MASTERS=3,SETID=ADM00002

*
*               *----------*
* COLOR MASTER:  * 1  2  3  *
* COLOR SEPS:    * RR BB GG * (RED, BLUE, GREEN)
*               *----------*
*
DEFAULT  ADMMCOLT (42,42,42)
BLUE     ADMMCOLT (41,42,41)
RED      ADMMCOLT (42,41,41)
PINK     ADMMCOLT (42,42,41)
GREEN    ADMMCOLT (41,41,42)
TURQSE   ADMMCOLT (41,42,42)
YELLOW   ADMMCOLT (42,41,42)
NEUTRAL  ADMMCOLT (42,42,42)
BACKGRD  ADMMCOLT (41,41,41)
```

```
*
*********************************************************************
* TABLE 3.  GENERAL COLOR MASTER TABLE                              *
*********************************************************************
*
ADM00003 ADMMCOLT PATTERN=ADMDHIPK,COLORS=256,MASTERS=8,SETID=ADM00003

*
*               *-------------------------*
* COLOR MASTER:  * 1  2  3  4  5  6  7  8  *
* COLOR SEPS:    * ** ** ** ** ** ** ** ** *
*               *-------------------------*
*
COL00    ADMMCOLT (41,41,41,41,41,41,41,41)
COL01    ADMMCOLT (42,41,41,41,41,41,41,41)
COL02    ADMMCOLT (41,42,41,41,41,41,41,41)
COL03    ADMMCOLT (42,42,41,41,41,41,41,41)
COL04    ADMMCOLT (41,41,42,41,41,41,41,41)
COL05    ADMMCOLT (42,41,42,41,41,41,41,41)
COL06    ADMMCOLT (41,42,42,41,41,41,41,41)
COL07    ADMMCOLT (42,42,42,41,41,41,41,41)
COL08    ADMMCOLT (41,41,41,42,41,41,41,41)
COL09    ADMMCOLT (42,41,41,42,41,41,41,41)
COL0A    ADMMCOLT (41,42,41,42,41,41,41,41)
COL0B    ADMMCOLT (42,42,41,42,41,41,41,41)
COL0C    ADMMCOLT (41,41,42,42,41,41,41,41)
COL0D    ADMMCOLT (42,41,42,42,41,41,41,41)
COL0E    ADMMCOLT (41,42,42,42,41,41,41,41)
COL0F    ADMMCOLT (42,42,42,42,41,41,41,41)
*
 .
 .  The first and last 16 values of a binary progression are shown
 .
*
COLF0    ADMMCOLT (41,41,41,41,42,42,42,42)
COLF1    ADMMCOLT (42,41,41,41,42,42,42,42)
COLF2    ADMMCOLT (41,42,41,41,42,42,42,42)
COLF3    ADMMCOLT (42,42,41,41,42,42,42,42)
COLF4    ADMMCOLT (41,41,42,41,42,42,42,42)
COLF5    ADMMCOLT (42,41,42,41,42,42,42,42)
COLF6    ADMMCOLT (41,42,42,41,42,42,42,42)
COLF7    ADMMCOLT (42,42,42,41,42,42,42,42)
COLF8    ADMMCOLT (41,41,41,42,42,42,42,42)
COLF9    ADMMCOLT (42,41,41,42,42,42,42,42)
COLFA    ADMMCOLT (41,42,41,42,42,42,42,42)
COLFB    ADMMCOLT (42,42,41,42,42,42,42,42)
COLFC    ADMMCOLT (41,41,42,42,42,42,42,42)
COLFD    ADMMCOLT (42,41,42,42,42,42,42,42)
COLFE    ADMMCOLT (41,42,42,42,42,42,42,42)
COLFF    ADMMCOLT (42,42,42,42,42,42,42,42)
```

```
*
*********************************************************************
* TABLE 4.  SUBTRACTIVE COLORS FOR PRINTERS WITH CLUSTER PATTERNS     *
*********************************************************************
*
ADM00004 ADMMCOLT PATTERN=ADMDHIPL,COLORS=17,MASTERS=1,SETID=ADM00004

*                             *-------*
*                             * GDDM  *
*                             * COLOR *
*                             *-------*
*
DEFAULT  ADMMCOLT (43)            0
BLUE     ADMMCOLT (59)            1
RED      ADMMCOLT (51)            2
PINK     ADMMCOLT (4A)            3
GREEN    ADMMCOLT (55)            4
TURQ     ADMMCOLT (4E)            5
YELLOW   ADMMCOLT (45)            6
NEUTRAL  ADMMCOLT (41)            7
BACKGRD  ADMMCOLT (61)            8
DKBLUE   ADMMCOLT (59)            9
ORANGE   ADMMCOLT (55)            10
PURPLE   ADMMCOLT (51)            11
DKGREEN  ADMMCOLT (59)            12
TURQSE   ADMMCOLT (45)            13
MUSTARD  ADMMCOLT (4E)            14
GRAY     ADMMCOLT (44)            15
BROWN    ADMMCOLT (57)            16
```

```
*
************************************************************************
* TABLE 5.  SUBTRACTIVE COLORS FOR PRINTERS WITH CLUSTER PATTERNS     *
* (THIS IS THE BVBTSO DEFINITION WITH APPROX 6% BETWEEN SHADES)       *
************************************************************************
*
ADM00005 ADMMCOLT PATTERN=ADMDHIPL,COLORS=17,MASTERS=1,SETID=ADM00005

*                              *-------*  *--------*
*                              * GDDM  *  * PIXELS *
*                              * COLOR *  *   %    *
*                              *-------*  *--------*
*
DEFAULT  ADMMCOLT (42)             0         3.1
BLUE     ADMMCOLT (5C)             1        84.3
RED      ADMMCOLT (58)             2        71.8
PINK     ADMMCOLT (54)             3        59.3
GREEN    ADMMCOLT (51)             4        50.0
TURQ     ADMMCOLT (47)             5        18.7
YELLOW   ADMMCOLT (4C)             6        34.3
NEUTRAL  ADMMCOLT (41)             7        00.0
BACKGRD  ADMMCOLT (61)             8       100.0
DKBLUE   ADMMCOLT (56)             9        65.6
ORANGE   ADMMCOLT (4F)            10        43.7
PURPLE   ADMMCOLT (4A)            11        28.1
DKGREEN  ADMMCOLT (53)            12        56.2
TURQSE   ADMMCOLT (45)            13        12.5
MUSTARD  ADMMCOLT (49)            14        25.0
GRAY     ADMMCOLT (43)            15         6.2
BROWN    ADMMCOLT (5A)            16        78.1
```

```
*
**********************************************************************
* TABLE 6. COLOR TONING SET FOR 3800/3820 PRINTERS                   *
**********************************************************************
*
ADM00006 ADMMCOLT PATTERN=ADMDHIPL,COLORS=17,MASTERS=1,SETID=ADM00006

*                             *-------*
*                             * GDDM  *
*                             * COLOR *
*                             *-------*
*
DEFAULT  ADMMCOLT (61)             0
BLUE     ADMMCOLT (5A)             1
RED      ADMMCOLT (54)             2
MAGENTA  ADMMCOLT (4A)             3
GREEN    ADMMCOLT (4F)             4
CYAN     ADMMCOLT (46)             5
YELLOW   ADMMCOLT (42)             6
NEUTRAL  ADMMCOLT (61)             7
BACKGRD  ADMMCOLT (41)             8
DKBLUE   ADMMCOLT (5D)             9
ORANGE   ADMMCOLT (4C)            10
PURPLE   ADMMCOLT (51)            11
DKGREEN  ADMMCOLT (56)            12
TURQSE   ADMMCOLT (47)            13
MUSTARD  ADMMCOLT (49)            14
GRAY     ADMMCOLT (44)            15
BROWN    ADMMCOLT (58)            16
```

```
*
*************************************************************************
* TABLE 7. COLOR TONING SET FOR 4250 PRINTER                            *
*************************************************************************
*
ADM00007 ADMMCOLT PATTERN=ADMDHIPL,COLORS=17,MASTERS=1,SETID=ADM00007

*                              *-------*
*                              * GDDM  *
*                              * COLOR *
*                              *-------*
*
DEFAULT  ADMMCOLT (61)             0
BLUE     ADMMCOLT (55)             1
RED      ADMMCOLT (4D)             2
MAGENTA  ADMMCOLT (47)             3
GREEN    ADMMCOLT (49)             4
CYAN     ADMMCOLT (44)             5
YELLOW   ADMMCOLT (42)             6
NEUTRAL  ADMMCOLT (61)             7
BACKGRD  ADMMCOLT (41)             8
DKBLUE   ADMMCOLT (59)             9
ORANGE   ADMMCOLT (48)            10
PURPLE   ADMMCOLT (4B)            11
DKGREEN  ADMMCOLT (4F)            12
TURQSE   ADMMCOLT (45)            13
MUSTARD  ADMMCOLT (46)            14
GRAY     ADMMCOLT (43)            15
BROWN    ADMMCOLT (51)            16
*
         ADMMCOLT END
         END
```

End of General-use programming interface

Chapter 15. Customizing user defaults for PostScript printing

To specify the mapping of GDDM colors onto PostScript colors, use the ADMMCLTB UDS.

To specify the mapping of GDDM symbol sets or presentation-text fonts onto PostScript typefaces, use the ADMMTYPF UDS.

Specifying the color mapping using the ADMMCLTB UDS

This user default specification enables you to modify the color table used by GDDM when applications generate PostScript output. Each GDDM color number is defined in terms of RGB and CMYK values.

The syntax is

```
ADMMCLTB CMYK=((col_no,C,M,Y,K),(col_no,C,M,Y,K),(.....))
```

or

```
ADMMCLTB RGB=((col_no,R,G,B),(col_no,R,G,B),(....))
```

where *col_no* is a value in the range 0 through 255 and the RGB or CMYK color values are floating-point values in the range 0.nnn through 1.

The synonym COLORTAB is provided for use in external defaults files

The colors in the default color table are shown in Table 14.

Table 14 (Page 1 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
1 blue	0.000	0.000	1.000
2 red	1.000	0.000	0.000
3 magenta	1.000	0.000	1.000
4 green	0.000	1.000	0.000
5 cyan	0.000	1.000	1.000
6 yellow	1.000	1.000	0.000
7 white	1.000	1.000	1.000
8 black	0.000	0.000	0.000
9 dark blue	0.000	0.000	0.667
10 orange	1.000	0.502	0.000
11 purple	0.667	0.000	0.667
12 dark green	0.000	0.573	0.000
13 dark turquoise	0.000	0.573	0.667
14 mustard	0.753	0.627	0.125
15 gray	0.514	0.514	0.514
16 brown	0.565	0.188	0.000
17 shadowy green	0.000	0.141	0.000
18 shadowy blue	0.000	0.141	0.333
19 dreary blue	0.000	0.141	0.667
20 happy blue	0.000	0.141	1.000
21 drab green	0.000	0.286	0.000
22 shadowy cyan	0.000	0.286	0.333
23 twilight blue	0.000	0.286	0.667
24 fresh blue	0.000	0.286	1.000
25 muddy green	0.000	0.427	0.000
26 muddy cyan	0.000	0.427	0.333

Table 14 (Page 1 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
27 numb cyan	0.000	0.427	0.667
28 light blue	0.000	0.427	1.000
29 stormy green	0.000	0.573	0.333
30 gloomy cyan	0.000	0.573	1.000
31 dusky green	0.000	0.714	0.000
32 murky green	0.000	0.714	0.333
33 twilight cyan	0.000	0.714	0.667
34 wintry cyan	0.000	0.714	1.000
35 ominous green	0.000	0.859	0.000
36 mysterious green	0.000	0.859	0.333
37 faded cyan	0.000	0.859	0.667
38 mysterious cyan	0.000	0.859	1.000
39 glowing green	0.000	1.000	0.333
40 scintillating green	0.000	1.000	0.667
41 shadowy red	0.169	0.000	0.000
42 shadowy pink	0.169	0.000	0.333
43 deep blue	0.169	0.000	0.667
44 hot blue	0.169	0.000	1.000
45 shadowy yellow	0.169	0.141	0.000
46 muddy blue	0.169	0.141	0.333
47 dense blue	0.169	0.141	0.667
48 glowing blue	0.169	0.141	1.000
49 drab yellow	0.169	0.286	0.000
50 drab cyan	0.169	0.286	0.333
51 wintry blue	0.169	0.286	0.667
52 warm blue	0.169	0.286	1.000

Table 14 (Page 2 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
53 dull green	0.169	0.427	0.000
54 stagnant cyan	0.169	0.427	0.333
55 sombre cyan	0.169	0.427	0.667
56 brilliant blue	0.169	0.427	1.000
57 sombre green	0.169	0.573	0.000
58 twilight green	0.169	0.573	0.333
59 dim cyan	0.169	0.573	0.667
60 scintillating blue	0.169	0.573	1.000
61 gloomy green	0.169	0.714	0.000
62 wintry green	0.169	0.714	0.333
63 evening cyan	0.169	0.714	0.667
64 mellow cyan	0.169	0.714	1.000
65 mellow green	0.169	0.859	0.000
66 happy green	0.169	0.859	0.333
67 sweet green	0.169	0.859	0.667
68 faint cyan	0.169	0.859	1.000
69 perky green	0.169	1.000	0.000
70 bright green	0.169	1.000	0.333
71 shining green	0.169	1.000	0.667
72 light cyan	0.169	1.000	1.000
73 drab red	0.333	0.000	0.000
74 drab pink	0.333	0.000	0.333
75 faded blue	0.333	0.000	0.667
76 stunning blue	0.333	0.000	1.000
77 dull red	0.333	0.141	0.000
78 muddy pink	0.333	0.141	0.333
79 flat blue	0.333	0.141	0.667
80 bright blue	0.333	0.141	1.000
81 muddy yellow	0.333	0.286	0.000
82 dull pink	0.333	0.286	0.333
83 faint blue	0.333	0.286	0.667
84 vibrant blue	0.333	0.286	1.000
85 dull yellow	0.333	0.427	0.000
86 dreary green	0.333	0.427	0.333
87 cheerful blue	0.333	0.427	0.667
88 dazzling blue	0.333	0.427	1.000
89 deep green	0.333	0.573	0.000
90 dense green	0.333	0.573	0.333
91 deep cyan	0.333	0.573	0.667
92 shining blue	0.333	0.573	1.000
93 faded green	0.333	0.714	0.000
94 flat green	0.333	0.714	0.333
95 murky cyan	0.333	0.714	0.667
96 flat cyan	0.333	0.714	1.000
97 faint green	0.333	0.859	0.000
98 fresh green	0.333	0.859	0.333
99 vibrant green	0.333	0.859	0.667
100 bleary cyan	0.333	0.859	1.000
101 stunning green	0.333	1.000	0.000
102 dazzling green	0.333	1.000	0.333
103 pastel green	0.333	1.000	0.667
104 dazzling cyan	0.333	1.000	1.000
105 stagnant red	0.502	0.000	0.000
106 stagnant pink	0.502	0.000	0.333
107 sombre pink	0.502	0.000	0.667
108 dusky pink	0.502	0.000	1.000
109 sombre red	0.502	0.141	0.000
110 numb pink	0.502	0.141	0.333
111 dreary pink	0.502	0.141	0.667
112 radiant blue	0.502	0.141	1.000
113 muddy orange	0.502	0.286	0.000

Table 14 (Page 2 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
114 dim red	0.502	0.286	0.333
115 stormy pink	0.502	0.286	0.667
116 sparkling blue	0.502	0.286	1.000
117 stagnant yellow	0.502	0.427	0.000
118 numb yellow	0.502	0.427	0.333
119 perky blue	0.502	0.427	0.667
120 soft blue	0.502	0.427	1.000
121 drab gray	0.004	0.004	0.004
122 dull gray	0.031	0.031	0.031
123 stagnant gray	0.063	0.063	0.063
124 sombre gray	0.094	0.094	0.094
125 stormy gray	0.125	0.125	0.125
126 dim gray	0.161	0.161	0.161
127 dusky gray	0.192	0.192	0.192
128 evening gray	0.224	0.224	0.224
129 gloomy gray	0.255	0.255	0.255
130 murky gray	0.290	0.290	0.290
131 foggy gray	0.322	0.322	0.322
132 ominous gray	0.353	0.353	0.353
133 mellow gray	0.384	0.384	0.384
134 dowdy gray	0.416	0.416	0.416
135 mysterious gray	0.451	0.451	0.451
136 bleary gray	0.482	0.482	0.482
137 hot gray	0.545	0.545	0.545
138 perky gray	0.580	0.580	0.580
139 glowing gray	0.612	0.612	0.612
140 sweet gray	0.643	0.643	0.643
141 light gray	0.675	0.675	0.675
142 bright gray	0.706	0.706	0.706
143 vibrant gray	0.741	0.741	0.741
144 radiant gray	0.773	0.773	0.773
145 dazzling gray	0.835	0.835	0.835
146 sparkling gray	0.871	0.871	0.871
147 soft gray	0.902	0.902	0.902
148 pastel gray	0.933	0.933	0.933
149 faded gray	0.965	0.965	0.965
150 faint gray	0.996	0.996	0.996
151 dreary yellow	0.502	0.573	0.333
152 sweet blue	0.502	0.573	0.667
153 pale blue	0.502	0.573	1.000
154 dim yellow	0.502	0.714	0.000
155 bleary green	0.502	0.714	0.333
156 foggy cyan	0.502	0.714	0.667
157 gentle blue	0.502	0.714	1.000
158 hot green	0.502	0.859	0.000
159 light green	0.502	0.859	0.333
160 sparkling green	0.502	0.859	0.667
161 glowing cyan	0.502	0.859	1.000
162 brilliant green	0.502	1.000	0.000
163 soft green	0.502	1.000	0.333
164 hazy green	0.502	1.000	0.667
165 soft cyan	0.502	1.000	1.000
166 evening red	0.667	0.000	0.333
167 faded pink	0.667	0.000	1.000
168 stormy red	0.667	0.141	0.000
169 foggy red	0.667	0.141	0.333
170 deep pink	0.667	0.141	0.667
171 ominous pink	0.667	0.141	1.000
172 dusky red	0.667	0.286	0.000
173 mellow red	0.667	0.286	0.333
174 twilight pink	0.667	0.286	0.667

Table 14 (Page 3 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
175 flat pink	0.667	0.286	1.000
176 sombre orange	0.667	0.427	0.000
177 bleary red	0.667	0.427	0.333
178 gloomy pink	0.667	0.427	0.667
179 pastel blue	0.667	0.427	1.000
180 stormy yellow	0.667	0.573	0.000
181 deep yellow	0.667	0.573	0.333
182 murky pink	0.667	0.573	0.667
183 hazy blue	0.667	0.573	1.000
184 twilight yellow	0.667	0.714	0.000
185 gloomy yellow	0.667	0.714	0.333
186 warm green	0.667	0.714	0.667
187 misty blue	0.667	0.714	1.000
188 faded yellow	0.667	0.859	0.000
189 radiant green	0.667	0.859	0.333
190 pale green	0.667	0.859	0.667
191 vibrant cyan	0.667	0.859	1.000
192 mellow yellow	0.667	1.000	0.000
193 faded green	0.667	1.000	0.333
194 misty green	0.667	1.000	0.667
195 faded cyan	0.667	1.000	1.000
196 murky red	0.835	0.000	0.000
197 mysterious red	0.835	0.000	0.333
198 evening pink	0.835	0.000	0.667
199 mysterious pink	0.835	0.000	1.000
200 ominous red	0.835	0.141	0.000
201 hot red	0.835	0.141	0.333
202 dense pink	0.835	0.141	0.667
203 bleary pink	0.835	0.141	1.000
204 dowdy red	0.835	0.286	0.000
205 fresh red	0.835	0.286	0.333
206 foggy pink	0.835	0.286	0.667
207 blunt pink	0.835	0.286	1.000
208 happy red	0.835	0.427	0.000
209 light red	0.835	0.427	0.333
210 mellow pink	0.835	0.427	0.667
211 fresh pink	0.835	0.427	1.000
212 dusky orange	0.835	0.573	0.000
213 vibrant red	0.835	0.573	0.333
214 sparkling red	0.835	0.573	0.667
215 bright pink	0.835	0.573	1.000
216 murky yellow	0.835	0.714	0.000
217 foggy yellow	0.835	0.714	0.333
218 pale red	0.835	0.714	0.667
219 steaming blue	0.835	0.714	1.000
220 wintry yellow	0.835	0.859	0.000
221 flat yellow	0.835	0.859	0.333
222 faint yellow	0.835	0.859	0.667
223 blizzard blue	0.835	0.859	1.000
224 mysterious yellow	0.835	1.000	0.000
225 blunt yellow	0.835	1.000	0.333
226 blizzard green	0.835	1.000	0.667
227 blizzard cyan	0.835	1.000	1.000
228 sweet red	1.000	0.000	0.333
229 wintry pink	1.000	0.000	0.667
230 perky red	1.000	0.141	0.000
231 bright red	1.000	0.141	0.333
232 dowdy pink	1.000	0.141	0.667
233 sweet pink	1.000	0.141	1.000
234 stunning red	1.000	0.286	0.000
235 scintillating red	1.000	0.286	0.333

Table 14 (Page 3 of 3). PostScript Red, Green, and Blue values for GDDM colors

Color	Red	Green	Blue
236 faint pink	1.000	0.286	0.667
237 radiant pink	1.000	0.286	1.000
238 warm red	1.000	0.427	0.000
239 dazzling red	1.000	0.427	0.333
240 faded red	1.000	0.427	0.667
241 sparkling pink	1.000	0.427	1.000
242 dense orange	1.000	0.573	0.000
243 shining orange	1.000	0.573	0.333
244 hazy red	1.000	0.573	0.667
245 pale pink	1.000	0.573	1.000
246 ominous orange	1.000	0.714	0.000
247 pastel orange	1.000	0.714	0.333
248 faint red	1.000	0.714	0.667
249 hazy pink	1.000	0.714	1.000
250 dowdy orange	1.000	0.859	0.000
251 bleary orange	1.000	0.859	0.333
252 blizzard red	1.000	0.859	0.667
253 blizzard pink	1.000	0.859	1.000
254 sparkling yellow	1.000	1.000	0.333
255 blizzard yellow	1.000	1.000	0.667

Notes:

1. Do not assume that the color names give any clear indication of the appearance of the colors.
2. Take care not to change the mapping of the color value 8 to any color other than black. The GRAYLINE processing option maps lines onto color 8, which it requires to be black.
3. The first ADMMCLTB specified determines how the colors are output. For instance, if the first ADMMCLTB is CMYK, all colors are output as CMYK. Subsequent definitions that use RGB are converted. If no ADMMCLTB specification is made, RGB values are used to output colors.

Specifying symbol-set and font mapping using the ADMMTYPF UDS

This user default specification enables you to specify how applications that use GDDM symbol sets and presentation-text fonts map those symbol sets to typeface names when a PostScript file is generated.

Mapping GDDM symbol sets to PostScript fonts

To map GDDM symbol sets to PostScript fonts, specify the SYMMAP parameter of the ADMMTYPF UDS. This is the syntax of the statement:

```
ADMMTYPF SYMMAP=((SS_name,Typeface),(SS_name,Typeface),(..))
```

Note: TYPEFACE can be used as a synonym for ADMMTYPF in external defaults files.

SS_name The GDDM symbol set for which a printer font is to be substituted.

Typeface The PostScript printer font to be used.

This name must be in the form recognized by the PostScript printer. You must pay attention to the case of letters and any imbedded special characters such as dashes or hyphens.

The default table maps each of the 12 Core Interchange symbol sets to the closest matching typeface supported by PostScript.

Table 15. The PostScript typefaces that most closely match the GDDM Core Interchange symbol sets

GDDM Core Interchange Symbol Set	Closest matching PostScript typeface
ADMUUT	Times-Roman
ADMUUTI	Times-Italic
ADMUUTB	Times-Bold
ADMUUTBI	Times-BoldItalic
ADMUUH	Helvetica
ADMUUHI	Helvetica-Oblique
ADMUUHB	Helvetica-Bold
ADMUUHBI	Helvetica-BoldOblique
ADMUVC	Courier
ADMUVCI	Courier-Oblique
ADMUVCB	Courier-Bold
ADMUVCBI	Courier-BoldOblique

Notes:

1. A symbol set is mapped only if the mapping is established *before* it is loaded with a GSLSS call.
2. Mapping defined using ESSUDS after GSLSS is not effective.
3. The default vector symbol set is not mapped unless loaded by GSLSS.
4. Image symbol sets are mapped to `Courier` by default.

5. Characters from vector symbol sets that are not mapped are drawn as lines and areas in the PostScript output.

Mapping IBM presentation-text fonts onto PostScript typefaces

The default table described in Appendix G, “Conversion table for GDDM and PostScript typefaces” on page 395 maps each 3820 and 38pp presentation font onto the closest matching PostScript font. If you want to change this mapping or add mapping for another font, use the FONTMAP parameter of the ADMMTYPF UDS to alter the table.

This is the syntax of the statement:

```
ADMMTYPF FONTMAP=((IBM_font,Typeface,Pointsize),(...),(...))
```

Note: TYPEFACE can be used as a synonym for ADMMTYPF in external defaults files.

IBM_font
The IBM presentation-text font name. See Appendix G, “Conversion table for GDDM and PostScript typefaces” on page 395 for information about how this font specification is used.

Typeface
The name of the PostScript printer font to be used.

The font name must appear here exactly as the PostScript printer knows it, paying attention to the case of letters and any imbedded special characters such as dashes or hyphens.

Pointsize
The point size of the IBM font.

Each IBM font is defined in a particular point size. This must be specified so that the PostScript printer font can be scaled appropriately.

The default table is shown in Appendix G, “Conversion table for GDDM and PostScript typefaces” on page 395.

Specifying nicknames for PostScript output

This section describes how to code name_lists for PostScript output and gives some examples for the CMS and TSO subsystems. For more details, see Appendix D, “Name-lists” on page 383.

Suppose an application opens a device using the following call:

```
        /* ID  DEV_FAMILY  DEV_TOKEN  PROCESSING OPTIONS  DEV_NAME */
CALL DSOPEN(5,      4,        '*',    0,PROCOPT_LIST,   1,QUALPRT   );
```

This call generates output for a printer with the name `QUALPRT`. End users at your enterprise, who want to print the output from their applications on a PostScript printer, can specify QUALPRT in nickname statements in their user default files.

Note: It is important to have the application code page set to a CECP when generating PostScript output with GDDM. The code points of EBCDIC country-extended code pages can be mapped onto code page 37, which GDDM uses for PostScript printing. If you use an EBCDIC code page that is not a CECP, the code points are not translated.

Coding nicknames for PostScript on VM/CMS

You can code a nickname statement in the external defaults module, ADMMDFV to associate a PostScript printer with the name-list QUALPRT. Here is an example of a possible nickname statement:

```
ADMMNICK FAM=4,NAME=QUALPRT,DEVTOK=PPS2CA4,
         TONAME=(QUALPRT,PSBIN,),
         PROCOPT=((POSTPROC,DOWNLOAD)),
         DESC="Generic level-2 color PostScript output downloaded to W.Stn."
```

Any output generated for the device called “QUALPRT” is written in ASCII format to a PostScript print file formatted for A4 paper. The TONAME parameter retains the name-list(1) parameter of the DSOPEN call but changes the name-list(2) parameter to PSBIN. This gives the PostScript file a filetype of PSBIN instead of ADMIMAGE, which is the default filetype for family-4 output. The POSTPROC processing option specifies that postprocessing of the PostScript output be performed by a user-written procedure called DOWNLOAD. The DOWNLOAD procedure must be capable of operating in CMS SUBSET mode.

Coding nicknames for PostScript on TSO

The name-list must specify a DDNAME or DSNAME for the generated file. Details are as for current Family-4 MVS name lists. You can code a nickname statement in the external defaults module, ADMMDFT to associate a PostScript printer with the name-list QUALPRT. Here is an example of a possible nickname statement:

```
ADMMNICK FAM=4,NAME=QUALPRT,DEVTOK=PPS2CA4,
         PROCOPT=((POSTPROC,DOWNLOAD)),
         DESC="Generic level-2 color PostScript output downloaded to W.Stn."
```

Any output generated for the device called “QUALPRT” is placed in a PostScript print file formatted for A4 paper. Datasets to be used for PostScript output should be allocated with the following DCB characteristics

```
Record format         Record length              Block size
  V or VB                 1028                     >=LRECL+4
```

The POSTPROC processing option specifies that post processing of the PostScript output be performed by a user-written procedure called DOWNLOAD. This procedure must be a TSO command, a CLIST, or a REXX exec. GDDM uses the TSO Service facility (IKJEFTSR) to invoke it.

Chapter 16. GDDM's code-page support

General-use programming interface

This chapter describes the code pages supported by GDDM, explains how country extended code pages work, and describes the structure of the alphanumeric defaults module ADMDATRN, so that you can alter it if necessary.

Extended Binary Coded Decimal Interchange Code (EBCDIC)

Both alphanumeric text characters and graphics text characters are represented internally as hexadecimal codes. A *code page* is a mapping between these hexadecimal codes and the characters they represent. The hexadecimal codes in a code page are usually referred to as *code points*.

By default, IBM devices attached to host computers use the Extended Binary Coded Decimal Interchange Code (EBCDIC) to represent single-byte characters. The EBCDIC codes for the Latin letters (A through Z) in uppercase and lowercase, and for Arabic numerals (0 through 9), are consistent across devices. However, EBCDIC allows the characters represented by some other codes, designated as being for national use, to vary from one device to another. For example, X'5B' is a national-use code: on terminals made for the U.S.A., X'5B' usually represents the dollar sign, $, but U.K. terminals normally use this code to represent the pound sign, £. In the U.S.A., X'4A' usually represents the cent sign, ¢, whereas in the U.K. it usually represents the dollar sign. So, if you enter this data on a U.S.A. terminal:

```
Price: $1 large or 50¢ small
```

store it in a file, and redisplay it on a U.K. terminal, you will probably see:

```
Price: £1 large or 50$ small
```

Furthermore, if the source and target code pages do not have the same number of code points, a code generated by one device may have no corresponding character defined for it on another device. In other words, data can contain characters that are nondisplayable or nonprintable on some devices.

Country extended code pages (CECPs)

To address the problems caused by variations in the EBCDIC national-use codes, GDDM supports extensions to EBCDIC called *country extended code pages* (CECPs). All GDDM CECPs contain 190 characters and 190 code points, ranging from X'41' through X'FE'. (X'40' is not defined in the CECPs, but always represents a blank.) The only difference between one CECP and another is in the order in which the 190 characters are mapped to the 190 code points. The 190 CECP characters are illustrated in Figure 17 on page 160.

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	¢	.	<	(	+	|
5	&	é	ê	ë	è	í	î	ï	ì	ß	!	$	*	)	;	¬
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	^	£	¥	·	©	§	¶	¼	½	¾	[	]	¯	¨	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 17. 190 CECP characters

The CECP character set includes all standard EBCDIC characters and all country-specific national-use characters. The 80 standard EBCDIC characters have the same code points in all CECPs, and each country's CECP uses the same code points for its 14 national-use characters as in the EBCDIC scheme. So, for example, the capital letter A is represented by X'C1', and the numeral 1 by X'F1', in both EBCDIC and CECP representations. The dollar sign is represented by X'5B' and the cent sign by X'4A' in both the U.S.A. CECP and the EBCDIC national-use assignments for the U.S.A.. In summary, CECPs are generally supersets of each country's older EBCDIC codes, which means that existing EBCDIC data is usually displayed or printed correctly on CECP devices.

Other code pages supported by GDDM

GDDM supports a number of extended, non-CECP code pages such as Latin 2 and Japanese. As well as the ones in the list in the next section, GDDM also supports Arabic character sets if a change is made to the alphanumeric-defaults module.
See "The alphanumeric-defaults module, ADMDATRN" on page 181.

In general, code page conversion occurs only for CECP code pages. There are three exceptions to this:

1. Translation occurs between Japanese code pages 290 and 1027, using character set 1172.
2. Translation occurs between Turkish code pages 1026 and 905. When a required character does not exist, it is replaced by the DUP character.
3. Translation occurs between CECP, Latin 2, and Cyrillic code pages using character-set folding. Where an accented character is not available, it is replaced by the non-accented character. Cyrillic characters are represented by their phonetic equivalent in the other code pages.

No translation occurs when symbol sets are used in alphanumeric fields or graphics text strings. Mode 1 graphics text is also not translated.

Code pages supported by GDDM

GDDM supports these EBCDIC, CECP, Katakana, and other non-CECP code pages:

Code-page name	*Code-page identifier*
CECP U.S.A./Canada/Portugal/Netherlands/Brazil	00037
CECP Austria/Germany	00273
CECP Denmark/Norway	00277
CECP Finland/Sweden	00278
CECP Italy	00280
CECP Japan (Latin)	00281
CECP Spain/Latin America	00284
CECP United Kingdom/Ireland	00285
GDDM Katakana	00290
GDDM Katakana (extended)	00290
CECP France	00297
GDDM default EBCDIC	00351
CECP multilingual page (MLP)/Switzerland/Belgium	00500
CECP Iceland	00871
GDDM Japan (Latin) Extended	01027
GDDM Latin 2	00870
GDDM Greece	00875
GDDM Cyrillic	00880 and 1025
GDDM Turkey	00905 and 1026
GDDM Baltic multilingual	01112
GDDM Estonia	01122
GDDM ISO/ANSI Multilingual	00819*
GDDM ASCII USA	00437*
GDDM ASCII Multilingual	00850*

Code pages marked * are provided for CGM interchange only. See “CGM code pages” on page 175.

In addition, code pages 00282 (Portugal), 00274 (Belgium), and 00275 (Brazil) are supported, but for compatibility only. They can be specified for applications that must run with CECP devices or data that employ these CECPs. Otherwise, they should not be specified.

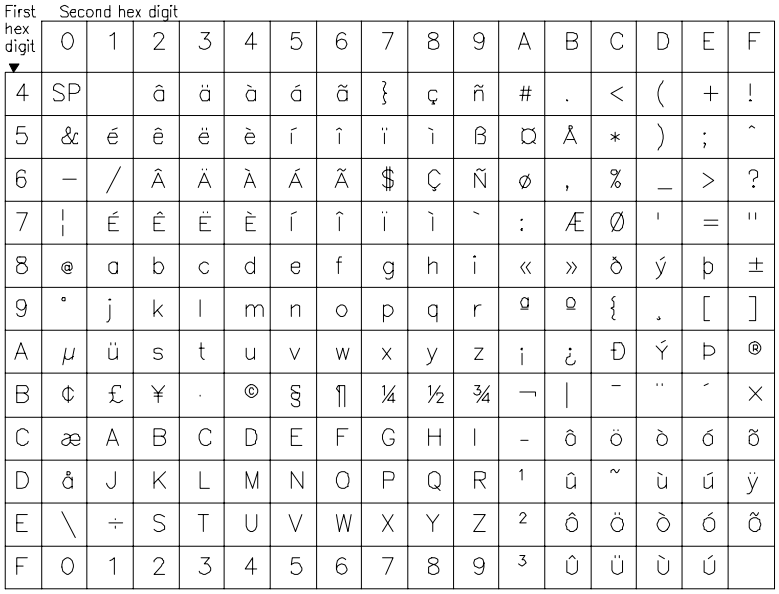
Some of the GDDM-supported code pages are illustrated in the figures that follow.

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	¢	.	<	(	+	|
5	&	é	ê	ë	è	í	î	ï	ì	ß	!	$	*	)	;	¬
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	^	£	¥	·	©	§	¶	¼	½	¾	[	]	¯	¨	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 18. Code page 00037 (U.S.A, Canada, Portugal, Netherlands, Brazil)

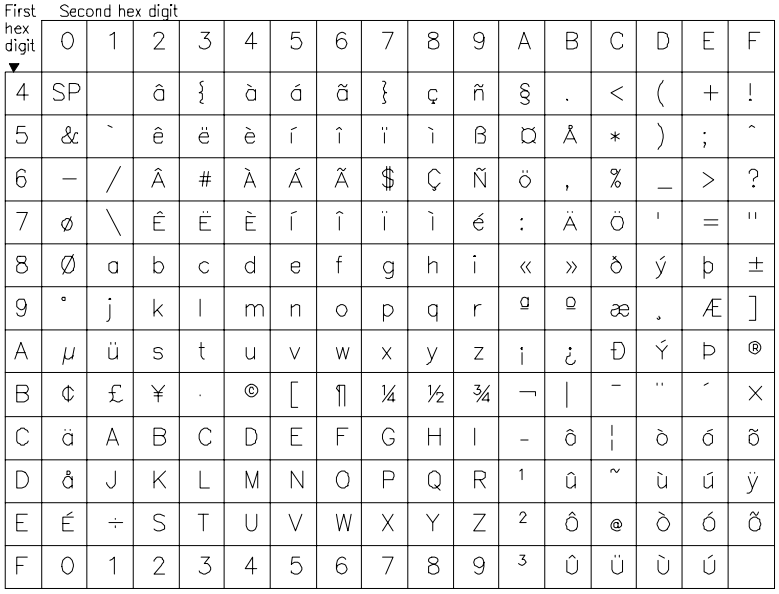
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	{	à	á	ã	å	ç	ñ	Ä	.	<	(	+	!
5	&	é	ê	ë	è	í	î	ï	ì	~	Ü	$	*	)	;	^
6	–	/	Â	[	À	Á	Ã	Å	Ç	Ñ	ö	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	§	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	ß	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	£	¥	·	©	@	¶	¼	½	¾	¬	|	¯	¨	´	×
C	ä	A	B	C	D	E	F	G	H	I	-	ô	¦	ò	ó	õ
D	ü	J	K	L	M	N	O	P	Q	R	¹	û	}	ù	ú	ÿ
E	Ö	÷	S	T	U	V	W	X	Y	Z	²	Ô	\	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	]	Ù	Ú	

Figure 19. Code page 00273 (Austria, Germany)



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	}	ç	ñ	#	.	<	(	+	!
5	&	é	ê	ë	è	í	î	ï	ì	ß	¤	Å	*	)	;	^
6	–	/	Â	Ä	À	Á	Ã	$	Ç	Ñ	ø	,	%	_	>	?
7	¦	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	Æ	Ø	'	=	"
8	@	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	{	,	[	]
A	µ	ü	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	£	¥	·	©	§	¶	¼	½	¾	¬	|	¯	¨	´	×
C	æ	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	å	J	K	L	M	N	O	P	Q	R	¹	û	~	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 20. Code page 00277 (Denmark, Norway)



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	{	à	á	ã	}	ç	ñ	§	.	<	(	+	!
5	&	`	ê	ë	è	í	î	ï	ì	ß	¤	Å	*	)	;	^
6	–	/	Â	#	À	Á	Ã	$	Ç	Ñ	ö	,	%	_	>	?
7	ø	\	Ê	Ë	È	Í	Î	Ï	Ì	é	:	Ä	Ö	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	,	Æ	]
A	µ	ü	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	£	¥	·	©	[	¶	¼	½	¾	¬	|	¯	¨	´	×
C	ä	A	B	C	D	E	F	G	H	I	-	ô	¦	ò	ó	õ
D	å	J	K	L	M	N	O	P	Q	R	¹	û	~	ù	ú	ÿ
E	É	÷	S	T	U	V	W	X	Y	Z	²	Ô	@	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

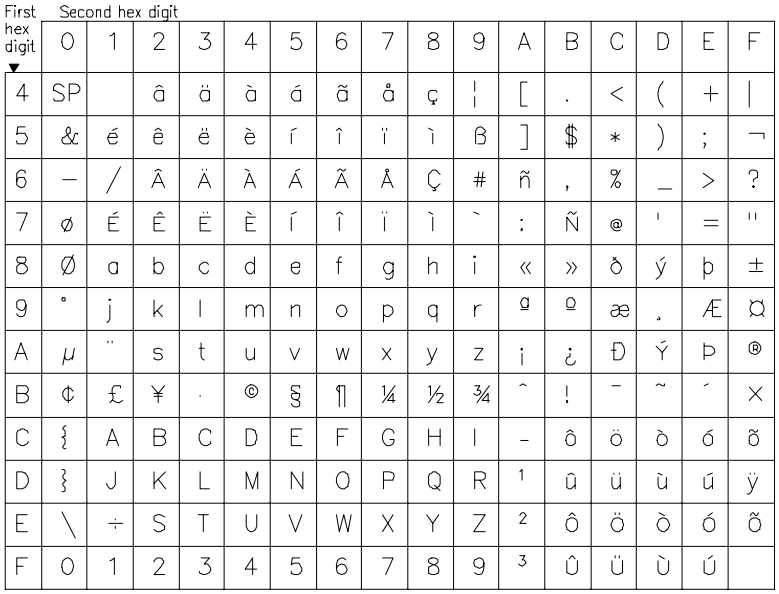
Figure 21. Code page 00278 (Finland, Sweden)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	{	á	ã	å	\	ñ	°	.	<	(	+	!
5	&	]	ê	ë	}	í	î	ï	~	ß	é	$	*	)	;	^
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	ò	,	%	_	>	?
7	ø	É	Ê	Ë	È	í	Î	ï	ì	ù	:	£	§	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	[	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	ì	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	#	¥	·	©	@	¶	¼	½	¾	¬	|	¯	¨	´	×
C	à	A	B	C	D	E	F	G	H	I	-	ô	ö	¦	ó	õ
D	è	J	K	L	M	N	O	P	Q	R	¹	û	ü	`	ú	ÿ
E	ç	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 22. Code page 00280 (Italy)

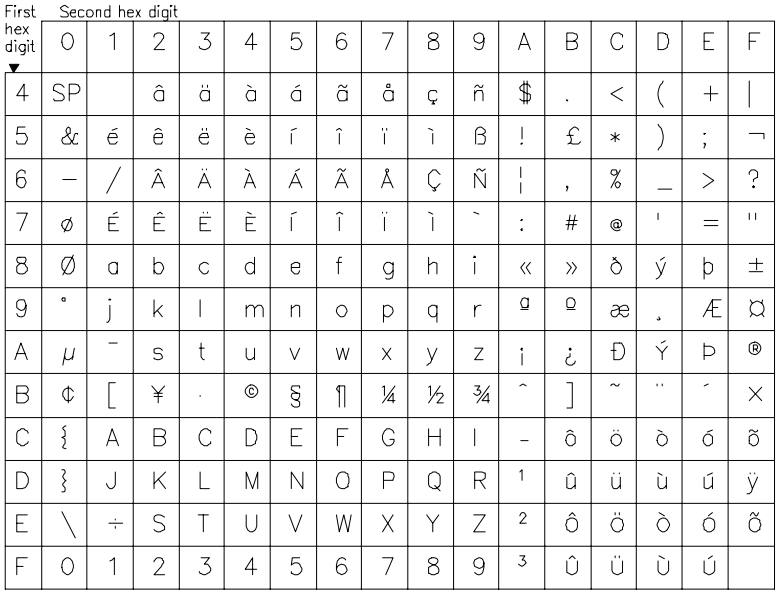
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	£	.	<	(	+	|
5	&	é	ê	ë	è	í	î	ï	ì	ß	!	¥	*	)	;	¬
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	í	Î	ï	ì	`	:	#	@	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	¯	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	[	\	·	©	§	¶	¼	½	¾	^	]	~	¨	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	$	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 23. Code page 00281 (Japan (Latin))



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	¦	[	.	<	(	+	|
5	&	é	ê	ë	è	í	î	ï	ì	ß	]	$	*	)	;	¬
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	#	ñ	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	Ñ	@	'	=	''
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	¨	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	£	¥	·	©	§	¶	¼	½	¾	^	!	¯	~	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 24. Code page 00284 (Spain, Latin America)



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	$	.	<	(	+	|
5	&	é	ê	ë	è	í	î	ï	ì	ß	!	£	*	)	;	¬
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	''
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	¯	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	[	¥	·	©	§	¶	¼	½	¾	^	]	~	¨	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 25. Code page 00285 (United Kingdom, Ireland)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	@	á	ã	å	\	ñ	°	.	<	(	+	!
5	&	{	ê	ë	}	í	î	ï	ì	ß	§	$	*	)	;	^
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	ù	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	µ	:	£	à	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	[	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	`	¨	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	#	¥	·	©	]	¶	¼	½	¾	¬	|	¯	~	´	×
C	é	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	è	J	K	L	M	N	O	P	Q	R	¹	û	ü	¦	ú	ÿ
E	ç	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 26. Code page 00297 (France)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	[	.	<	(	+	!
5	&	é	ê	ë	è	í	î	ï	ì	ß	]	$	*	)	;	^
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	`	:	#	@	'	=	"
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ð	ý	þ	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	~	s	t	u	v	w	x	y	z	¡	¿	Ð	Ý	Þ	®
B	¢	£	¥	·	©	§	¶	¼	½	¾	¬	|	¯	¨	´	×
C	{	A	B	C	D	E	F	G	H	I	-	ô	ö	ò	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ô	Ö	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 27. Code page 00500 (Multilingual, Switzerland, Belgium)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4	SP		â	ä	à	á	ã	å	ç	ñ	Þ	.	<	(	+	!
5	&	é	ê	ë	è	í	î	ï	ì	ß	Æ	$	*	)	;	Ö
6	–	/	Â	Ä	À	Á	Ã	Å	Ç	Ñ	¦	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	Ì	ð	:	#	Ð	'	=	''
8	Ø	a	b	c	d	e	f	g	h	i	«	»	`	ý	{	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	}	¸	]	¤
A	µ	ö	s	t	u	v	w	x	y	z	¡	¿	@	Ý	[	®
B	¢	£	¥	·	©	§	¶	¼	½	¾	¬	|	¯	¨	\	×
C	þ	A	B	C	D	E	F	G	H	I		ô	~	ò	ó	õ
D	æ	J	K	L	M	N	O	P	Q	R	¹	û	ü	ù	ú	ÿ
E	´	÷	S	T	U	V	W	X	Y	Z	²	Ô	^	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 28. Code page 00871 (Iceland)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		â	ä	ţ	á	ă	č	ç	ć	[	ą	<	ˇ	+	!
5	&	é	ę	ë	ů	í	î	ť	ĺ	Ł	]	Ś	*	)	ş	ˆ
6	–	/	Â	Ä	´´	Á	Ă	Č	Ç	Ć	|	ą	˛	___	>	?
7	˘	Ě	Ę	Ë	Ů	Í	Î	Ľ	Ĺ	˜	;	#	@	'	=	¨
8	˘	a	b	c	d	e	f	g	h	i	ś	ň	ð	˝	ř	ş
9	°	j	k	l	m	n	o	p	q	r	ł	ń	š	¸	˛	¤
A	ą	~	s	ţ	u	v	w	x	y	z	Ś	Ň	Đ	Ý	Ř	Ş
B	°	Ą	ž	Ţ	Ž	§	ž	ź	Ž	Ź	Ł	Ń	Š	¨	´	×
C	{	A	B	C	D	E	F	G	H	I		ô	ö	ŕ	ó	ő
D	}	J	K	L	M	N	O	P	Q	R	Ě	ú	ü	ť	ú	ę
E	\	÷	S	T	U	V	W	X	Y	Z	ą	ô	ö	Ř	ó	ő
F	Ø	1	2	3	4	5	6	7	8	9	Ď	Ú	Ü	Ţ	Ú	

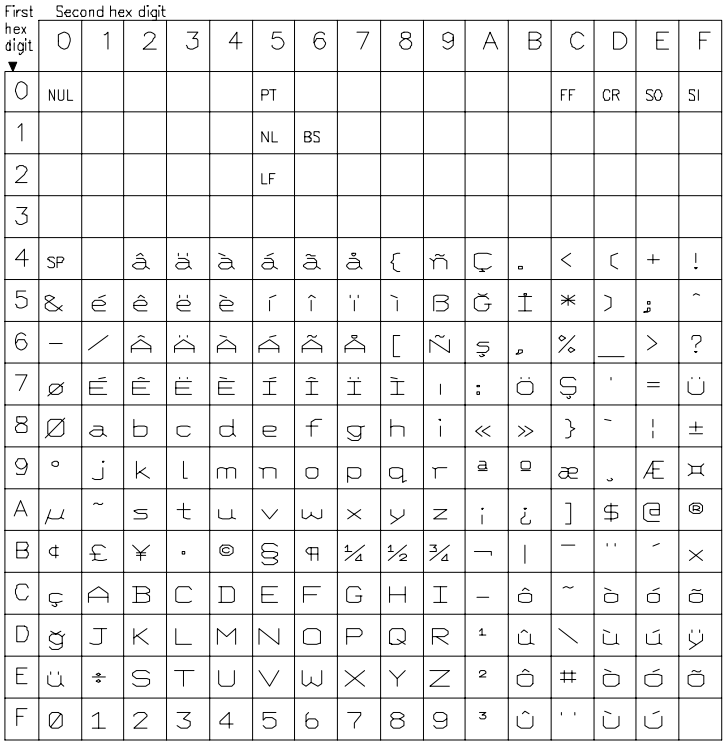
Figure 29. Code page 00870 (Latin 2)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	[	.	<	(	+	!
5	&	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	]	$	*	)	;	^
6	-	/	Τ	Υ	Φ	Χ	Ψ	Ω	Ϊ	Ϋ	|	,	%	_	>	?
7	¨	Ά	Έ	Ή		Ί	Ό	Ύ	Ώ	`	:	#	@	'	=	‘
8	΅	α	β	χ	δ	ε	φ	γ	η	ι	ξ	κ	λ	μ	ν	ο
9	°	j	k	l	m	n	o	p	q	r	π	θ	ι	κ	χ	μ
A	´	~	σ	τ	υ	ν	ω	χ	υ	ζ	ν	ΐ	ο	π	ρ	σ
B	£	ά	έ	ή	ϊ	ί	ό	ύ	ϋ	ώ	ς	τ	υ	φ	χ	ψ
C	{	Α	Β	C	D	Ε	F	G	Η	Ι	-	ω	ί	ϋ	΄	―
D	}	J	K	L	M	N	O	P	Q	R	±	½		΅	΄	¦
E	\	÷	S	T	U	V	W	X	Y	Z	²	§			«	¬
F	0	1	2	3	4	5	6	7	8	9	³	©			»	

Figure 30. Code page 00875 (Greece)

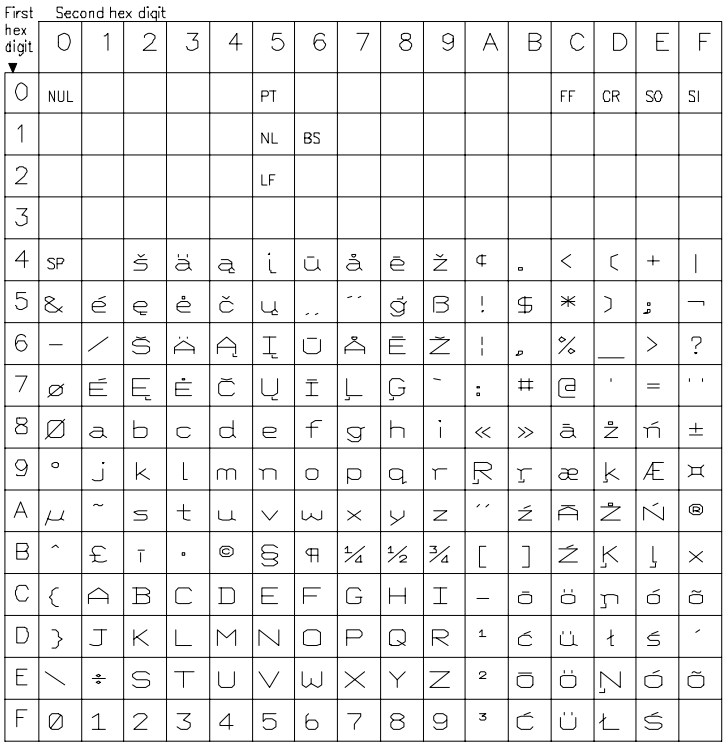
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		Ђ	ѓ	ё	є	ѕ	і	ї	ј	[	.	<	(	+	!
5	&	љ	њ	ћ	ќ	ў	џ	ъ	№	Ъ	]	$	*	)	;	^
6	-	/	Ѓ	Ё	Є	Ѕ	I	Ї	J	Љ	|	,	%	_	>	?
7	Њ	Ћ	Ќ	-	Ў	Џ	ю	а	б	`	:	#	@	'	=	‘
8	ц	a	b	c	d	e	f	g	h	i	д	e	ф	г	х	и
9	й	j	k	l	m	n	o	p	q	r	к	л	м	н	о	п
A	я	~	s	t	u	v	w	x	y	z	р	с	т	у	ж	в
B	ь	ы	з	ш	э	щ	ч	ъ	Ю	А	Б	Ц	Д	Е	Ф	Г
C	{	А	В	С	D	Е	F	G	Н	I	Х	И	Й	К	Л	М
D	}	J	K	L	M	N	O	P	Q	R	Н	О	П	Я	Р	С
E	\	§	S	T	U	V	W	X	Y	Z	Т	У	Ж	В	Ь	Ы
F	0	1	2	3	4	5	6	7	8	9	З	Ш	Э	Щ	Ч	

Figure 31. Code page 01025 (Cyrillic)



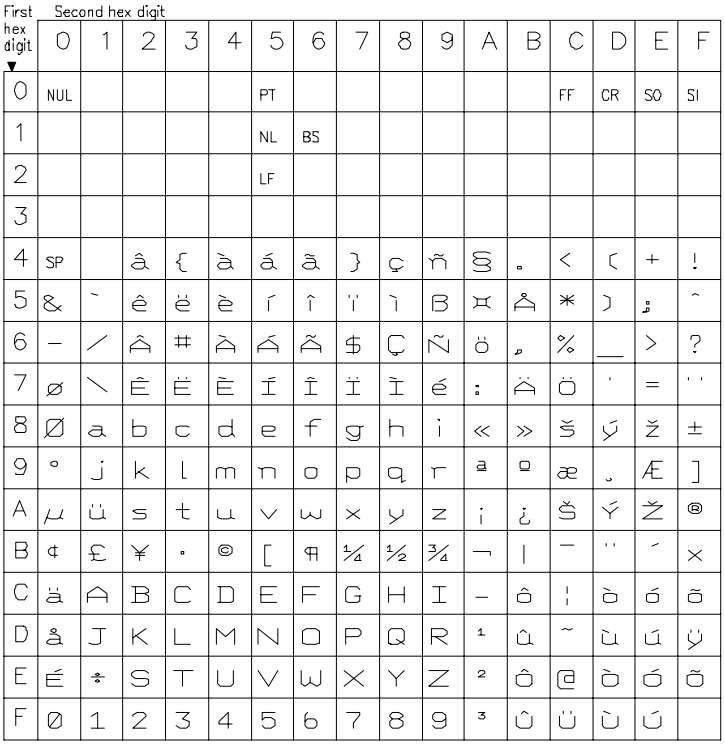
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		â	ä	à	á	ã	å	{	ñ	Ç	.	<	(	+	!
5	&	é	ê	ë	è	í	î	ï	ì	B	Ğ	İ	*	)	;	^
6	-	/	Â	Ä	À	Á	Ã	Å	[	Ñ	Ş	,	%	_	>	?
7	ø	É	Ê	Ë	È	Í	Î	Ï	İ	:	Ö	Ş	'	=	Ü	
8	Ø	a	b	c	d	e	f	g	h	i	«	»	}	`	¦	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	¤
A	µ	~	s	t	u	v	w	x	y	z	¡	¿	]	$	®	®
B	¢	£	¥	·	©	§	¶	¼	½	¾	¬	|	¯	¨	´	×
C	ç	A	B	C	D	E	F	G	H	I	-	ô	~	ò	ó	õ
D	ğ	J	K	L	M	N	O	P	Q	R	¹	û	\	ù	ú	ÿ
E	ü	÷	S	T	U	V	W	X	Y	Z	²	Ô	#	Ò	Ó	Õ
F	Ø	1	2	3	4	5	6	7	8	9	³	Û	`	Ù	Ú	

Figure 32. Code page 01026 (Turkey)



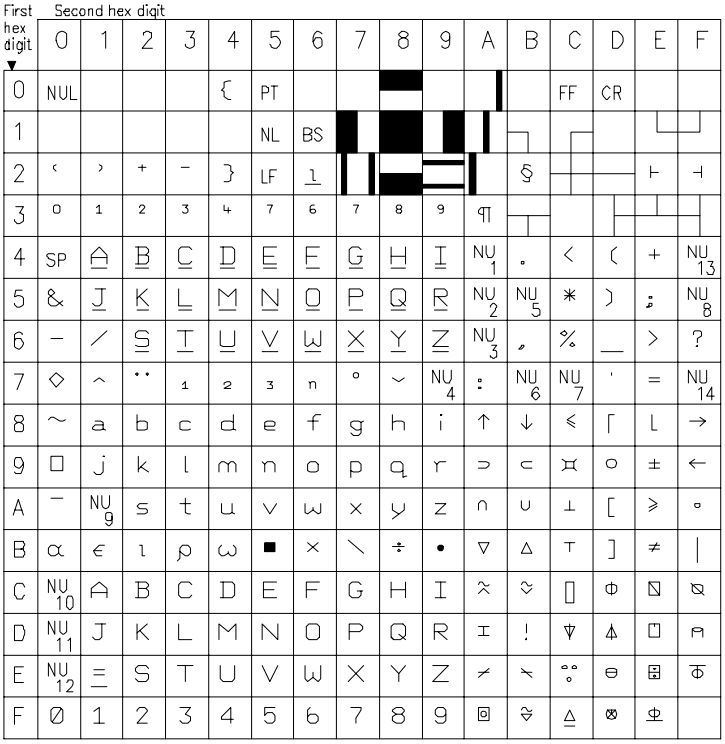
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		š	ä	ą	¡	ū	å	ē	ž	¢	.	<	(	+	|
5	&	é	ę	ė	č	ų	¸	´	ģ	B	!	$	*	)	;	¬
6	-	/	Š	Ä	Ą	Į	Ū	Å	Ē	Ž	¦	,	%	_	>	?
7	ø	É	Ę	Ė	Č	Ų	Ī	Ļ	Ģ	`	:	#	@	'	=	¨
8	Ø	a	b	c	d	e	f	g	h	i	«	»	ą	ž	ń	±
9	°	j	k	l	m	n	o	p	q	r	Ŗ	ŗ	æ	ķ	Æ	¤
A	µ	~	s	t	u	v	w	x	y	z	´	ź	Ą	Ž	Ń	®
B	^	£	ī	·	©	§	¶	¼	½	¾	[	]	Ź	Ķ	Ļ	×
C	{	A	B	C	D	E	F	G	H	I	-	ō	ö	ņ	ó	õ
D	}	J	K	L	M	N	O	P	Q	R	¹	ć	ü	ł	ś	´
E	\	÷	S	T	U	V	W	X	Y	Z	²	Ō	Ö	Ņ	Ó	Õ
F	Ø	1	2	3	4	5	6	7	8	9	³	Ć	Ü	Ł	Ś	

Figure 33. Code page 01112 (Baltic multilingual)



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		â	{	à	á	ã	}	ç	ñ	§	.	<	(	+	!
5	&	`	ê	ë	è	í	î	¨	ì	ß	¤	Å	*	)	;	^
6	–	/	Â	#	À	Á	Ã	$	Ç	Ñ	ö	,	%	_	>	?
7	ø	\	Ê	Ë	È	Í	Î	Ï	Ì	é	:	Ä	Ö	'	=	' '
8	Ø	a	b	c	d	e	f	g	h	i	«	»	š	ý	ž	±
9	°	j	k	l	m	n	o	p	q	r	ª	º	æ	¸	Æ	]
A	µ	ü	s	t	u	v	w	x	y	z	¡	¿	Š	Ý	Ž	®
B	¢	£	¥	·	©	[	¶	¼	½	¾	¬	|	¯	¨	´	×
C	ä	Ä	B	C	D	E	F	G	H	I	–	ô	¦	ò	ó	õ
D	å	J	K	L	M	N	O	P	Q	R	¹	û	~	ù	ú	ÿ
E	É	÷	S	T	U	V	W	X	Y	Z	²	Ô	@	Ò	Ó	Õ
F	0	1	2	3	4	5	6	7	8	9	³	Û	Ü	Ù	Ú	

Figure 34. Code page 01122 (Estonia)



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL				{	PT							FF	CR		
1						NL	BS									
2	‹	›	+	–	}	LF	1					§			⊢	⊣
3	0	1	2	3	4	7	6	7	8	9	¶					
4	SP	Â	B	C	D	E	F	G	H	I	NU 1	.	<	(	+	NU 13
5	&	J	K	L	M	N	O	P	Q	R	NU 2	NU 5	*	)	;	NU 8
6	–	/	S	T	U	V	W	X	Y	Z	NU 3	,	%	_	>	?
7	◇	^	··	1	2	3	n	°	ˇ	NU 4	:	NU 6	NU 7	'	=	NU 14
8	~	a	b	c	d	e	f	g	h	i	↑	↓	⩽	⌈	⌊	→
9	□	j	k	l	m	n	o	p	q	r	⊃	⊂	¤	○	±	←
A	¯	NU 9	s	t	u	v	w	x	y	z	∩	∪	⊥	[	⩾	°
B	α	∈	ı	ρ	ω	■	×	\	÷	•	▽	△	⊤	]	≠	|
C	NU 10	Â	B	C	D	E	F	G	H	I	⩽	⩾	[]	Φ	⊠	⊗
D	NU 11	J	K	L	M	N	O	P	Q	R	⊥	!	∀	⍋	⍞	⍈
E	NU 12	=	S	T	U	V	W	X	Y	Z	⍲	⍱	⍤	⊖	⊞	⍟
F	0	1	2	3	4	5	6	7	8	9	⍰	⩾	⍙	⊗	⍦	

Figure 35. Code page 00351 (Default EBCDIC)

How CECPs work

All of the code pages supported by GDDM are defined in *translation tables*, which are in the alphanumeric defaults module, ADMDATRN. By means of pointers, the characters in one translation table can be mapped to those in any other table. So, for example, a word containing é entered in French-language data is displayed correctly as é on a German-language terminal when CECP code-page conversion is enabled.

Module ADMDATRN can be modified. The structure of the module, and some reasons for updating it, are provided beginning on page 181.

The remainder of this section describes the possible GDDM code-page conversions, and explains what you have to do to enable code-page conversion for your enterprise.

Code-page conversion

A code-page value can be specified for an installation, for an application, for a GDDM object, and for a device. When data moves from one of these entities to another (for example, when an application writes to a device), that data is guaranteed to display correctly only if the code page of the application and the device are the same. If they are not, *code-page conversion* is required to ensure correct display. By default, no code-page conversion occurs: you have to take special action to enable this. When it is enabled, however, code-page conversion applies to all character data. For example, if the application code page is USA CECP 00037 and the device code page is UK CECP 00285, and the application passes code point `X'5B'` for display, GDDM converts it to `X'4A'` before transmitting it to the device, so that the dollar sign displays as a dollar sign rather than as a pound sign.

Note: The three GDDM code pages (default EBCDIC 00351 and Katakana 00290 and 01027), are not CECPs, so do not support code-page conversion. Therefore, if either the source or the target code page is a GDDM code page, *no conversion takes place*.

Application code page

The application code page is the code page in which all character data moving between an application and GDDM is interpreted. You can specify a code page for all applications running under GDDM using the APPCPG external default. An application can also specify the APPCPG external default on an ESEUDS, ESSUDS or SPINIT call. If no APPCPG external default is specified, the GDDM default EBCDIC code page 00351 is used. This effectively prevents code-page conversion for all data passed to and from the application program.

Installation code page

The installation code page is that used for character data passing between GDDM and the operating system: it determines principally the names of files or data sets. The installation code page is specified on the GDDM external default, INSCPG. If you do not specify an installation code page, CECP 0037 (U.S. English) is used. This is the CECP for USA, Canada, Portugal, the Netherlands and Brazil.

Device code page

The device code page is that used by a terminal, printer, or plotter for data input and output. GDDM queries the code-page support of devices whenever an explicit or implicit DSOPEN call is executed. The CECP identifier returned by the device is used as the device code page, unless a DEVCPG processing option has been specified for the device, in which case the processing option overrides the query reply. (DEVCPG is described in Appendix B, “Processing options” on page 333.) If the device does not return a CECP identifier and no DEVCPG processing option has been specified for the device, the installation code page is used as the device code page.

Object code page

The object code page applies to data stored in GDDM objects. The GDDM objects and their file-types are:

GDDM object	*File type*
Chart-data files	ADMCDATA
Chart-definition files	ADMCDEF
Chart-format files	ADMCFORM
Graphics-data-format (GDF) files	ADMGDF
Generated mapgroups	ADMGGMAP
GDDM-IMD tutorial pages	ADMGIMP
GKS metafiles	ADMGKSM
Image-data files	ADMIMG
Image-projection-definition files	ADMPROJ
Saved pictures (FSSAVE) files	ADMSAVE
Symbol sets	ADMSYMBL

GDDM objects are described in Chapter 6, “GDDM objects and other files” on page 47.

All GDDM objects are tagged with the application code page that is current at the time the object is created. When an existing object is loaded, GDDM inspects its code-page tag and uses the identified code page to interpret the object’s contents. If an object does not have a valid tag (usually this is because it was created under a release of GDDM before Version 2 Release 2), GDDM uses the current application code page as the object code page.

Please note the following:

Notes:

1. ADMGDF object files converted from picture interchange format (PIF) using GDDM’s conversion utility are tagged with the application code page that is current at the time of conversion. However, the object’s contents are not converted.
2. ADMPRINT files are tagged with an object code page when they are created. The GDDM Print Utility converts any CECP data in the file from the object code page to the device code page.
3. Symbol sets are converted only if they contain a character in every CECP code point. That is, an image symbol set must have characters for X'41' through

X'FE', and a vector symbol set must have characters for X'42' through X'FE'.

What you should do about code-page conversion

1. Specify your local, national CECP as the installation and application code pages for GDDM. For example, if you are working in the French language, add these ADMMDFT statements to your external defaults module:

   ```
   ADMMDFT INSCPG=00297
   ADMMDFT APPCPG=00297
   ```

   These statements enable code-page conversion.

   For more information about specifying external defaults, see Chapter 17, “GDDM user-default specifications” on page 197.

2. In some circumstances, you might want applications to operate without code-page conversion. If this is the case, specify EBCDIC code page 00351 as the application code page. Note, however, that explicit conversions using the FSTRAN call are unaffected by the current application code page. FSTRAN is described in the *GDDM Base Application Programming Reference* book.

3. For Katakana applications, there are code pages 00290 (GDDM Katakana) and 01027 (Japan (Latin) extended). When either is specified as the application or device code page, no CECP-type code-page conversion takes place. However, data transmitted to and from devices is converted as if an ASTYPE call with a parameter value of 3 had been executed.

4. Some terminals, such as the 3179-G, allow you to specify whether keyboard input of all 190 CECP code points is to be allowed. You might want to do this in order to protect existing applications from the input of new code points. When full CECP input is prevented, only a base set of (typically) 94 code points can be entered. Attempting to enter one of the new CECP code points puts the terminal into the input-inhibit state.

   To inhibit full CECP input, include the following ADMMDFT statement in your external defaults module:

   ```
   ADMMDFT CECPINP=NO
   ```

   The inclusion of this statement does not affect the display and printing of the full range of CECP characters.

5. The CECP character set does not include APL characters. Applications requiring APL characters can use GDDM default EBCDIC (code page 00351) as the application code page. Alternatively, applications can use the ASCSS or GSCS call to specify an alternative, nonloadable symbol set. If code page 00351 is specified as the application code page, but national-language characters are also required, the GDDM default symbol sets should be replaced. This process is described in Chapter 12, “The GDDM default symbol sets” on page 131.

6. The 4250 printer code-page function, in which a code page is specified using a GSCPG call or CPN4250 external default parameter, applies only when the current device is a 4250 printer. If a 4250 (type 5) symbol set is specified in a GSLSS call, the specified symbol set is loaded using the code page defined on the GSCPG call or on the CPN4250 external default. It is not affected by the

current GDDM device or application code page. This function therefore remains independent of the code page functions.

CECP support for users of extended character set RPQs

This information is for users in multilingual environments who have an RPQ solution from IBM to help them use an extended set of national characters. Such solutions are available to customers with Version 2 Release 2 (or later) of GDDM in Latin America and in countries such as Belgium, Canada, Portugal, and Switzerland.

If such an RPQ is in use at your installation, you need to take one or more of the following actions to ensure correct CECP translation:

1. Attach the devices, whose character sets you want to expand, via a 3174 controller with the CECP RPQ 8Q0556.
2. Modify the ADMDATRN module to move the correct entry ahead of entry `00101 00037` in the the second part of the CECP look-up table. See "Altering the alphanumeric defaults module" on page 181 and the comments in the module itself for instructions on how to do this.
3. Specify the DEVCPG processing option in the user-default specifications accessed by users of the devices supported by the RPQ. Appropriate values for the procopt in each country are as follows:

Belgium	`PROCOPT=((DEVCPG,73007604))`
Canada bilingual	`PROCOPT=((DEVCPG,59310117))`
New Spanish	`PROCOPT=((DEVCPG,42598684))`
Portugal	`PROCOPT=((DEVCPG,73007141))`
Switzerland	`PROCOPT=((DEVCPG,59507188))`

Adding code-page tags to GDDM objects

If you are migrating from a release of GDDM earlier than GDDM Version 2 Release 2, you are likely to have many untagged objects in your system. You should consider adding tags to these objects, as GDDM does not attempt code-page conversion of untagged objects. In the VM/CMS and TSO environments, a module is supplied to allow users to add a tag to a GDDM object. In the VM environment, it is invoked as follows:

```
ADMUOTV object-name object-type cpgid
```

In the MVS environment, it is invoked as follows:

```
CALL 'GDDM.SADMMOD(ADMUOTT)' 'object-name object-type cpgid'
```

In both cases, `object-name` and `object-type` (a number from the list in Table 16 on page 175) identify the object, and `cpgid` is the number of the code page.

Table 16. Tagging GDDM objects using ADMUOTx		
Description	**Name**	**GDDM object-type number**
Chart data	ADMCDATA	5
Chart definition	ADMCDEF	9
Chart format	ADMCFORM	4
GDF files	ADMGDF	7
Generated mapgroups	ADMGGMAP	2
GKS metafiles	ADMGKSM	8
Image data	ADMIMG	11
Image projection	ADMPROJ	10
Saved pictures	ADMSAVE	3
Symbol sets	ADMSYMBL	1

GDDM objects that you import from another country and that have code-page tags are managed automatically by GDDM, with the exception of ADMCDATA and ADMCFORM chart files. A PL/I sample program, ADMUSP7, is supplied with GDDM-PGF to enable you to import these. If you use ADMUSP7, you need to link-edit the supplied text deck, in much the same way as other PL/I programs are link-edited. For more information about ADMUSP7, see the *GDDM-PGF Programming Reference* book.

CGM code pages

The CGLOAD and CGSAVE API calls, and the ADMUCG and ADMUGC utilities, allow interchange between Computer Graphics Metafile (CGM) and ADMGDF formats.

Generally, text strings in a CGM are in one of the ASCII code pages and so translation of character strings is necessary when GDDM imports (CGLOAD or ADMUCG) or exports (CGSAVE or ADMUGC) CGMs.

Limited support is provided for three ASCII-based CGM code pages, catering for the majority of CGM interchange situations:

- Code page 437 - ASCII USA
- Code page 850 - ASCII Multilingual
- Code page 819 - ISO/ASCII Multilingual

These are shown in the following figures.

All these code pages use the same code points for the “core” characters (such as uppercase and lowercase Latin letters and numerics). They differ in both the quantity and code points of other characters. Only code page 819 contains all the 190 CECP characters (but not at the EBCDIC/CECP code points).

You can set the CGM code page to be used as a parameter of the API calls or utilities mentioned above, in the CGM conversion profile, or you can let it default to code page 850. Refer to the CGM information in the *GDDM Base Application Programming Reference*.

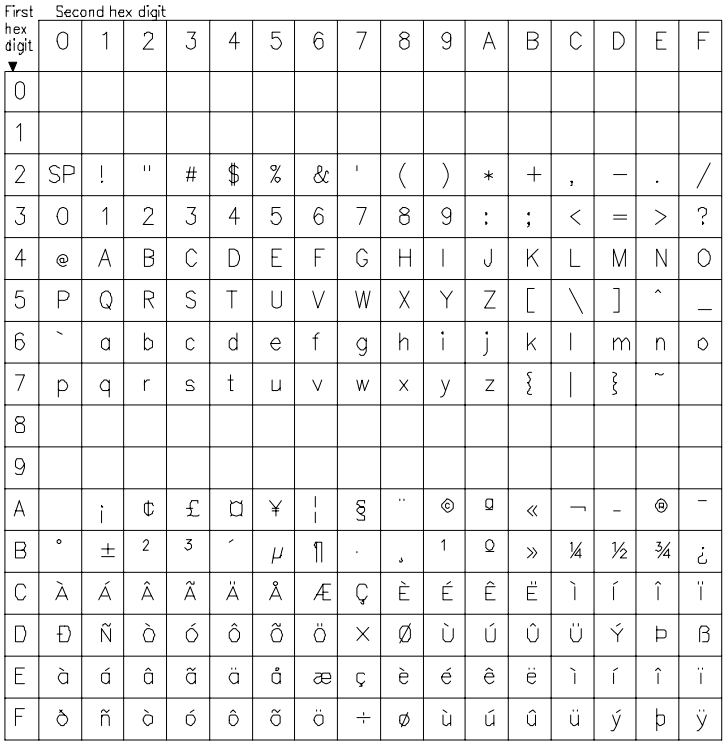
You may need to experiment to discover which CGM code page suits your non-GDDM application that generates or interprets CGM.

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								'								¤
1					¶	§	–									
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	¢	£	¥		f
A	á	í	ó	ú	ñ	Ñ	ª	º	¿		¬	½	¼	¡	«	»
B				|											¬
C						+										
D																
E		ß					µ		Ø			ð		ø		
F		±					÷		°	·	·		n	²		

Figure 36. Code page 00437 (ASCII USA)

First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0								'								¤
1					¶	§	–									
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	–	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8	Ç	ü	é	â	ä	à	å	ç	ê	ë	è	ï	î	ì	Ä	Å
9	É	æ	Æ	ô	ö	ò	û	ù	ÿ	Ö	Ü	ø	£	Ø	×	f
A	á	í	ó	ú	ñ	Ñ	ª	º	¿	®	¬	½	¼	¡	«	»
B				|		Á	Â	À	©					¢	¥	¬
C					–	+	ã	Ã								¤
D	ð	Ð	Ê	Ë	È	ı	Í	Î	Ï					¦	Ì	
E	Ó	ß	Ô	Ò	õ	Õ	µ	þ	Þ	Ú	Û	Ù	ý	Ý	¯	´
F		±	=	¾	¶	§	÷	¸	°	¨	·	¹	³	²		

Figure 37. Code page 00850 (ASCII Multilingual)



First hex digit ▼	Second hex digit															
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2	SP	!	"	#	$	%	&	'	(	)	*	+	,	−	.	/
3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
6	`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
7	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~	
8																
9																
A		¡	¢	£	¤	¥	¦	§	¨	©	ª	«	¬	-	®	¯
B	°	±	²	³	´	µ	¶	·	¸	¹	º	»	¼	½	¾	¿
C	À	Á	Â	Ã	Ä	Å	Æ	Ç	È	É	Ê	Ë	Ì	Í	Î	Ï
D	Ð	Ñ	Ò	Ó	Ô	Õ	Ö	×	Ø	Ù	Ú	Û	Ü	Ý	Þ	ß
E	à	á	â	ã	ä	å	æ	ç	è	é	ê	ë	ì	í	î	ï
F	ð	ñ	ò	ó	ô	õ	ö	÷	ø	ù	ú	û	ü	ý	þ	ÿ

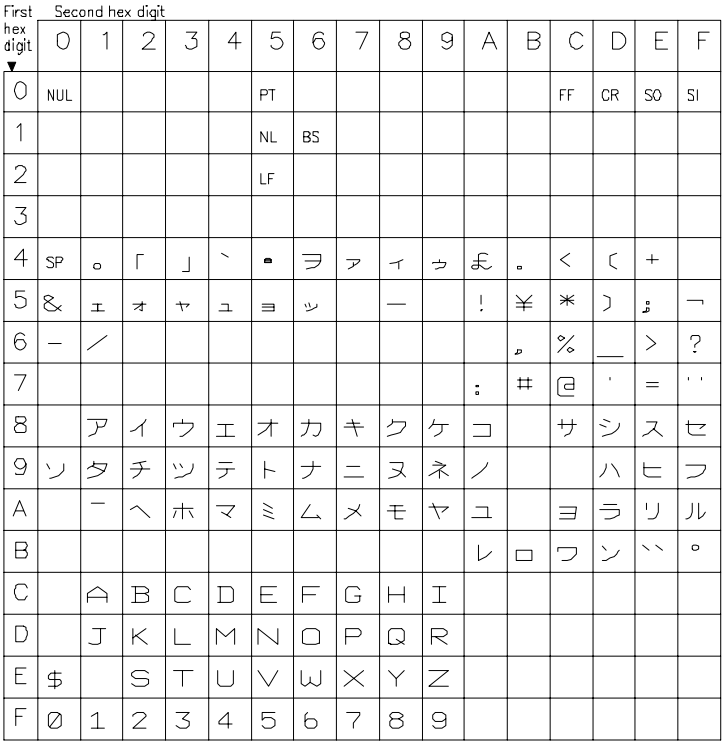
Figure 38. Code page 00819 (ISO/ANSI Multilingual)

Enabling translation between Japanese code pages 290 (extended) and 1027

Whenever a GDDM application opens a device, it issues an internal query of that device's characteristics. One characteristic queried is the device code page. Every application has a code page associated with it. If the device code page and application code page are different and no mapping has been established between them, meaningful user input to the application is impossible.

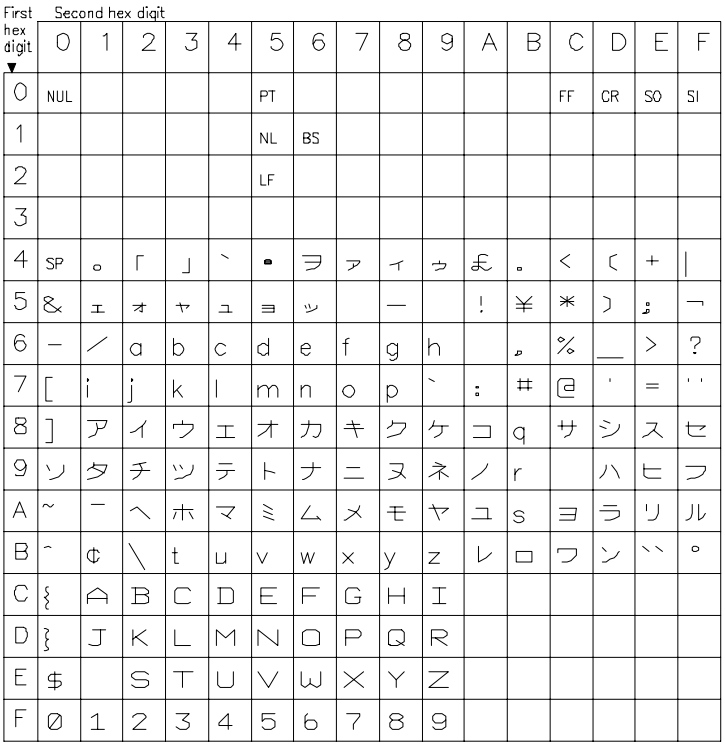
Because Japanese (Latin) Extended code page 1027 and Japanese (Katakana) Extended code page 290 are based on character set 1172, GDDM can map all the code points of one onto the other. Either can be the device code page and function perfectly with the other as the application code page.

You can compare the characters and their code points in figures 39, 40, and 41.



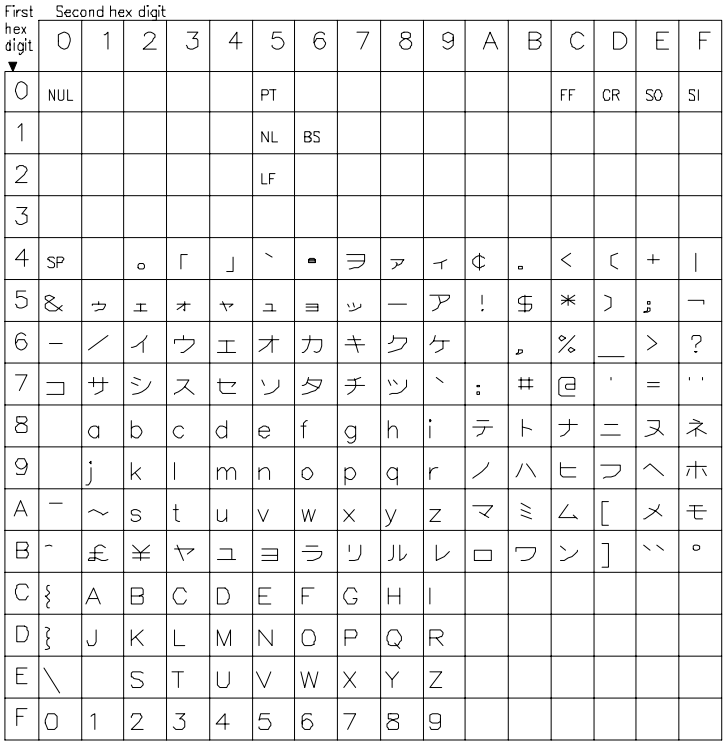
First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP	。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	
5	&	ッ	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ
6	−	／										ﾟ	％	_	>	?
7											:	#	@	'	=	"
8		ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ		サ	シ	ス	セ
9	ソ	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ			ハ	ヒ	フ
A		¯	ヘ	ホ	マ	ミ	ム	メ	モ	ヤ	ユ		ヨ	ラ	リ	ル
B											レ	ロ	ワ	ン	゛	°
C		A	B	C	D	E	F	G	H	I						
D		J	K	L	M	N	O	P	Q	R						
E	$		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

| *Figure 39. Code page 00290 Katakana*



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP	。	「	」	、	・	ヲ	ァ	ィ	ゥ	ェ	ォ	ャ	ュ	ョ	|
5	&	ッ	ー	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	サ	シ	ス	セ
6	−	／	a	b	c	d	e	f	g	h		ﾟ	％	_	>	?
7	[	i	j	k	l	m	n	o	p	`	:	#	@	'	=	"
8	]	ア	イ	ウ	エ	オ	カ	キ	ク	ケ	コ	q	サ	シ	ス	セ
9	ソ	タ	チ	ツ	テ	ト	ナ	ニ	ヌ	ネ	ノ	r		ハ	ヒ	フ
A	~	¯	ヘ	ホ	マ	ミ	ム	メ	モ	ヤ	ユ	s	ヨ	ラ	リ	ル
B	^	¢	\	t	u	v	w	x	y	z	レ	ロ	ワ	ン	゛	°
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	$		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

| *Figure 40. Code page 00290 Katakana (extended)*



First hex digit ▼	Second hex digit 0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	NUL					PT							FF	CR	SO	SI
1						NL	BS									
2						LF										
3																
4	SP		。	「	」	、	・	ヲ	ァ	ィ	¢	。	<	(	+	|
5	&	ゥ	ェ	ォ	ャ	ュ	ョ	ッ	ー	ア	!	$	*	)	;	¬
6	-	/	イ	ウ	エ	オ	カ	キ	ク	ケ		゛	%	_	>	?
7	コ	サ	シ	ス	セ	ソ	タ	チ	ツ	ヽ	:	#	@	'	=	‘ ’
8		a	b	c	d	e	f	g	h	i	テ	ト	ナ	ニ	ヌ	ネ
9		j	k	l	m	n	o	p	q	r	ノ	ハ	ヒ	フ	ヘ	ホ
A	‾	～	s	t	u	v	w	x	y	z	マ	ミ	ム	[	メ	モ
B	¯	£	¥	ヤ	ユ	ヨ	ラ	リ	ル	レ	ロ	ワ	ン	]	゛	゜
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

Figure 41. Code page 01027 (Japan (Latin) Extended).

NUL = null
PT = program tab
FF = form feed
CR = carriage return
SO = shift out
SI = shift in
NL = new line
BS = backspace
LF = line feed
SP = space

You can enable automatic translation between the Japanese extended Katakana and extended Latin code pages by assigning the identifier of a code-page based on character set 1172 to the device code page, application code page, or installation code page. Japanese (Latin) Extended code page 1027 and Japanese (Katakana) Extended code page 290 are both based on character set 1172. You can, for example, enter statements such as these in the user defaults module.

```
ADMMDFT APPCPG=1027
ADMMDFT INSCPG=1027
```

If you don’t set one of these defaults explicitly, code page 0351 is used, which disables all translation.

Reconfiguring devices to enable translation

Some devices can be configured with Japanese (Katakana) code page 290 as the associated code page. Because this code page is based on character set 332, no mapping can be established between its code points and those of Japanese (Latin) Extended code page 1027. A partial mapping can be established between the Japanese (Katakana) code page 290 and Japanese (Katakana) Extended code page 290.

Because many applications use Japanese (Latin) Extended code page 1027 as the application code page, any users at your enterprise with devices using the Japanese (Katakana) code page 290, can have problems with input.

You can avoid these problems by reconfiguring the device to use a suitable code page. This is the best solution but is possible only with PS/55 workstations.

If nonconfigurable devices are in use, you may need to consider overwriting the character-set information returned by the device query with character set 1172. See “Overwriting the device’s code-page and character-set information” for more information.

Overwriting the device’s code-page and character-set information

You can force GDDM to treat user input from Japanese (Katakana) code page 290 with character set 332 as if it came from Japanese (Katakana) Extended code page 290 with character set 1172. Since the characters in Japanese (Katakana) code page 290 occupy the same positions in Japanese (Katakana) Extended code page 290, the input is just as if it came from a device configured with the extended code page. If the application code page is Japanese (Latin) Extended code page 1027, GDDM converts the input to the equivalent code points on that code page.

Problems may arise however, if the application sends output to the user’s screen. Because character set 1172 contains characters (such as lowercase Latin characters) that are not in character set 332, such characters cannot be displayed correctly by the device.

You can overwrite the device’s code-page and character-set information in the following ways, though it is probably best to let the users of affected devices do this themselves:

1. Set the device code page to 290 with the DEVCPG processing option. GDDM searches in the code-page tables of the ADMDATRN module and uses the character set referenced in the first entry for 290 in the tables. If you ensure the first entry for 290 in the tables refers to character set 1172, Japanese (Katakana) Extended code page 290 is used. If the first entry refers to character set 332, the basic 290 code page is used and translation with Japanese (Latin) Extended code page 1027 is not possible.

   You can override the character-set reference in the code-page tables without changing the ADMDATRN module by specifying the DEVCSET processing option on a nickname in the user defaults module. You can specify a nickname in the following way:

   ```
   ADMMNICK FAM=0,PROCOPT=((DEVCSET,1172))
   ```

2. You can set the device code page to 290 and the character set to 1172 simultaneously using the DEVCPG processing option. This procopt has the

syntax (DEVCPG,n) in which the input parameter n is derived from the code-page identifier, cp and the character-set identifier cs in the following way.

n = cs (2¹⁶) + cp

To specify Japanese (Katakana) Extended code page 290, n is calculated as follows:

n = 1172 (2¹⁶) + 290 = 76808482

The nickname statement to specify the DEVCPG procopt is:

```
ADMMNICK FAM=0,PROCOPT=((DEVCPG,76808482))
```

The alphanumeric-defaults module, ADMDATRN

The alphanumeric-defaults module, ADMDATRN, is a set of 256-character translation tables used by GDDM to ensure that any of the character codes used by an application appears the same on any device, regardless of the national language being used. Pointers enable the characters in one translation table to be mapped to those in any other table. ADMDATRN also plays a part in enterprises that use APL, and in those that use Personal Services/CICS (PS/CICS).

Altering the alphanumeric defaults module

The supplied module ADMDATRN handles code-page conversion in most circumstances and does not have to be altered. However, you might want to alter the module to:

- Set up special translation tables of your own that can support a nonstandard device or a group of nonstandard devices.
- Make the default translation table Katakana rather than EBCDIC, so that IBM 3270-family devices (including workstations that use IBM 3270 data-stream architecture) are assumed to be Katakana rather than EBCDIC where there is such a choice.

  You can make this change simply by specifying a Katakana code page on the INSCPG and APPCPG external defaults. Alternatively, you can:

  1. Locate the assembler label T0 in ADMDATRN.
  2. On the next line change F‘2’ to F‘3’.
  3. Reassemble and link-edit the defaults module as described in Chapter 19, “Updating GDDM default modules” on page 213.

If you decide to make changes to ADMDATRN, you need to understand the structure of the module. ADMDATRN is described in the remainder of this chapter.

Altering ADMDATRN to support the Arabic character set

There are two ways of modifying and using the ADMDATRN table.

1. The table contains TRNTYPES fields that correspond to the characteristics of supported terminals. These are used when the application or device is not CECP. If the application program is modifiable by you, you should create a new TRNTYPE for use by the Arabic terminals and specify this with an ASTYPE call. The new type may be modeled on T5279 (non-APL) and the six translate tables changed to pass through all characters from X'40' through X'FF' without change. This is done by changing locations X'40' through X'FF' to have the values X'40' through X'FF'.

2. If the application cannot be changed, the translate tables for all the device types that you use need modifying as described above. Also, if APL is supported, it has to be disabled by changing the TRNFMASK at location X'000008' from B'11111011' to B'11011011'. (This turns off the TRNAPLM bit.)

The structure of the alphanumeric defaults module

The alphanumeric defaults module consists of sets of tables for various national languages and devices. *Control information* in the module governs how the tables are to be used and how they are to be associated with a particular language and type of device. The format of the alphanumeric defaults module is shown in Figure 42. As the figure shows, the module comprises:

- Basic control information
- Translation-type descriptor blocks for each supported translation-type value
- The translation tables themselves.

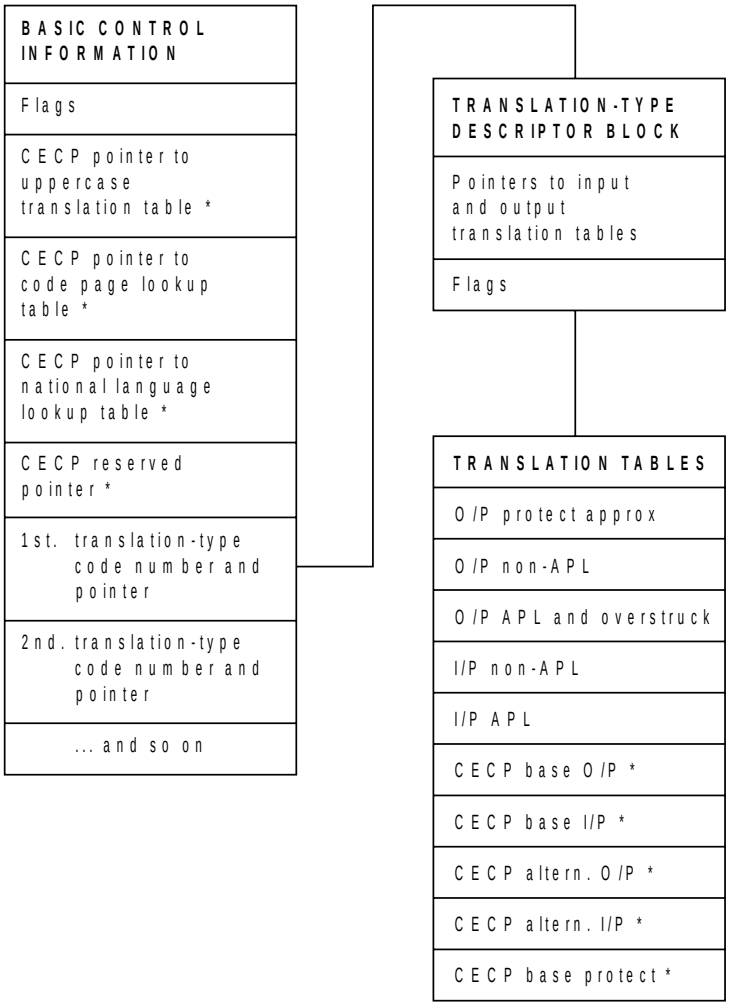


Figure 42. Format of the alphanumeric defaults module

The contents of the fields shown in Figure 42 are described in the remainder of this chapter. If you intend updating the module, you are recommended to refer to the additional information included in the source of ADMDATRN itself.

The structure described in the following pages uses TRNxxx labels. These labels do not appear in the supplied version of ADMDATRN, but you could use them to define overlays representing the structure.

Basic control information

This is the first part of the alphanumeric defaults module.

Table 17. Alphanumeric defaults module, basic control information

Field name	Field offset	Field length (in bytes)	IBM default value (where applicable)
TRNUCTP	0	4	Address
TRNPTP	4	4	Address
TRNFMASK	8	3	–
TRN3277M	x.......	Bits in TRNFMASK	1
TRNDISPM	.x......		1
TRNAPLM	..x.....		1
TRNNLCM	...x....		1
TRNSYSPM	....x...		1
TRNIGM	......x.		1
TRN293M	x		1
Reserved	xxxxxxxx		–
Reserved	xxxxxxxx		–
TRNFCECP	11	1	–
TRNCECP	x.......	Bits in TRNFCECP	1
Reserved	.xxxxxxx		–
TRNNTYPE	12	4	54
TRNCECPP			–
TRNMUCTP	16	4	Address
TRNCPLTP	20	4	Address
TRNLLTP	24	4	Address
TRNEXTP	28	4	Address
TRNTYPES(*)	32+		–
TRNTYPE		4	0
TRNTYPEP		4	Address

TRNUCTP
The address of the uppercase translation table. The GDDM-supplied table translates lowercase EBCDIC characters to uppercase, but leaves all other characters unchanged. The address must not be zero.

TRNPTP
The address of the invalid-character translation table. The GDDM-supplied table translates the character codes X'01' through X'3F' and X'FF' to the character code X'40' on output. The address must not be zero.

TRNFMASK
A set of flags that correspond to device-characteristic flags in the translation-type descriptor blocks. If these flags are set on, the corresponding flags in the translation-type descriptor blocks are tested when searching for a translation table set for a particular device.

TRN3277M Older 3270 devices, or those that support highlighting, color, and PS.

TRNDISPM	Display or printer.
TRNAPLM	APL or non-APL devices.
TRNNLCM	Device displays lowercase characters in recognizable form or does not.
TRNSYSPM	System printers or other devices. (System printers are characterized by the use of carriage control.)
TRNIGM	Interactive graphics devices or others.
TRN293M	Device supports APL code page 293.
Reserved	Eight unused bits.
Reserved	Eight unused bits.

TRNFCECP
A set of flags relating to country extended code pages (CECPs)

TRNCECP	Country extended code page available.
Reserved	Seven unused bits.

TRNNTYPE
A fullword integer (or string of four characters) specifying the number of entries in TRNTYPES. Each entry consists of one TRNTYPE and one TRNTYPEP. The list is scanned at GDDM device-initialization time to determine which translation type to use. There must be at least one entry.

The translation types supplied with GDDM are listed in “Translation tables in ADMDATRN.”

TRNCECPP
An array of four pointers defined over the TRNTYPES array for use by CECP:

TRNMUCTP	Pointer to CECP uppercase translation table.
TRNCPLTP	Pointer to CECP lookup table.
TRNLLTP	Pointer to national-language-to-CECP lookup table.
TRNEXTP	Pointer to extension table. This contains:
	TRN79XP Pointer to 3279 adjust table.
	TRNASCP Pointer to ASCII device code page index table.
	TRNDTRNP Pointer to code page direct translation table.

TRNTYPES(n)
Array of device types:

TRNTYPE	A fullword integer (or string of four characters) naming the translation-type identifier for this entry. This value must be different from all other values of TRNTYPE in the list.
TRNTYPEP	The address of a translation-type descriptor block. Each TRNTYPEP in the list should address a different descriptor block. This address must not be zero.

Translation tables in ADMDATRN

When GDDM is selecting a translation type, the search starts at group 0 in the group selections, and goes to the point specified by group 0. By default, this is group 2, unless it has been changed to 3 (Katakana). GDDM then moves to the point specified by group selection 2 or 3, and searches until a translation-type number that matches the device in use is located.

Group selections: define the points in the table at which the search starts.

Table 18. Group selections

Translate type number	Device
0	Points to 2 by default (can be altered to be same as Katakana or other group)
2	Points to EBCDIC group
3	Points to Katakana group
293	Points to APL2 group

Specific selections: can be specified only on an ASTYPE call.

Table 19. Specific selections

Translate type number	Device	"Invalid" character
1	No translation except output protection using TRNPTP.	—
3288	3288	ASTERISK (*)
32881	3288-TN	ASTERISK (*)
3289	3289	ASTERISK (*)
32891	3289-TN	ASTERISK (*)

Katakana group: (defaulting to EBCDIC group) – one of these values is selected by GDDM if group selection 0 is set to point to 3 and a suitable match is found.

Table 20. Katakana group

Translate type number	Device	"Invalid" character
Note: DUP(*) means unprintable character		
32772	Katakana 3277	DUP(*)
32792	Katakana 3279 or 5550	DUP(*)
32793	Katakana 3279 – APL	DUP(*)
3179	Katakana 3179-G	DUP(*)
31791	Katakana 3179-G – APL	DUP(*)
5553	Katakana 5553 or 5557	DUP(*)
55531	Katakana 5553 or 5557 – APL	DUP(*)
4224	Katakana 4224	DUP(*)
42241	Katakana 4224 – APL	DUP(*)
3812	Katakana 3812	DUP(*)
38121	Katakana 3812 – APL	DUP(*)
5550	Katakana 5550 – Japanese (Latin) Extended	DUP(*)
55501	Katakana 5550 – Japanese (Latin) Extended – APL	DUP(*)

EBCDIC group: (may be selected by GDDM by default) – one of these values is selected by GDDM if group selection 0 is set to point to 2, or if it is set to point to 3 but no suitable match is found in the Katakana group.

Note: For 3290 and 8775 terminals, GDDM selects translation types as it does for 3278 terminals.

Table 21. EBCDIC group

Translate type number	Device	"Invalid" character
Note: DUP(⋆) means unprintable character		
5279	3270-PC/G, 3270-PC/GX, 3179-G	DUP(⋆)
52791	3270-PC/G, 3270-PC/GX, 3179-G – APL2	DUP(⋆)
3277	3277	DUP(⋆)
3279	3278, 3279	DUP(⋆)
32771	3277-APL/DE	DUP(⋆)
32791	3278-APL/DE, 3279-APL/DE	DUP(⋆)
713287	3271/3287	DUP(⋆)
743287	3271/3287	DUP(⋆)
7132871	3271/3287-APL	DUP(⋆)
7432871	3274/3287-APL	DUP(⋆)
1403	1403, 3211	DUP(⋆)
744224	3274/4224	DUP(⋆)
7442241	3274/4224-APL	DUP(⋆)
743812	3274/3812	DUP(⋆)
7438121	3274/3812-APL	DUP(⋆)

APL2 group: APL2 group – Tables in this group act to produce an Application Code Page of 293 on devices that can support it.

Table 22 (Page 1 of 2). APL2 group

Translate type number	Device	"Invalid" character
Note: DUP(⋆) means unprintable character		
52792	3270-PC/G, 3270-PC/GX, 3179-G	DUP(⋆)
527912	3270-PC/G, 3270-PC/GX, 3179-G – APL2	DUP(⋆)
3277	3277	DUP(⋆)
3279	3278, 3279	DUP(⋆)
327712	3277-APL/DE	DUP(⋆)
327912	3278-APL/DE, 3279-APL/DE	DUP(⋆)
713287	3271/3287	DUP(⋆)
743287	3274/3287	DUP(⋆)
7132872	3271/3287-APL	DUP(⋆)
7432872	3274/3287-APL	DUP(⋆)

Table 22 (Page 2 of 2). APL2 group

Translate type number	Device	"Invalid" character
1403	1403, 3211	DUP(*)
74424	3274/4224	DUP(*)
74422412	3274/4224-APL	DUP(*)
743812	3274/3812	DUP(*)
74381212	3274/3812-APL	DUP(*)

Translation-type descriptor block

The translation-type descriptor block is the second part of the alphanumeric defaults module.

Table 23. Alphanumeric defaults module, translation-type descriptor block

Field name	Field offset	Field length (in bytes)	IBM default value (where applicable)
TRNDEFRT	0	4	number
TRNBOP	0	4	Address or 0
TRNBIP	4	4	Address or 0
TRNAOP	8	4	Address or 0
TRNAIP	12	4	Address or 0
TRNBPOP	16	3	—
TRNFLAGS	20	Bits	—
TRN3277	x.......	within	—
TRNDISP	.x......	TRNFLAGS	—
TRNAPL	..x.....		—
TRNUCOT	...x....		—
TRNSYSP	....x...		—
TRNDEFR	.....x..		—
TRNIG	......x.		—
TRN293	x		—
			—
TRNNONU	x.......		—
TRN1027	.x......		—
Reserved	..xxxxxx	1	—
TRNIIC	23	4	Address or 0
TRNCBOP	24	4	Address or 0
TRNCBIP	28	4	Address or 0
TRNCAOP	32	4	Address or 0
TRNCAIP	36	4	Address or 0
TRNCPOP	40	4	Address or 0

TRNDEFRT

Referred to only if TRNDEFR=1. Contains the type number at which search should continue. It is used to bypass standard defaults with nonstandard defaults such as Katakana.

TRNBOP

The address of the output symbol set 0 (non-APL) translation table. If TRNBOP is zero, this translation does not occur.

TRNBIP

The address of the input symbol set 0 (non-APL) translation table. If TRNBIP is zero, this translation does not occur.

TRNAOP
The address of the output symbol set 1 (APL) translation table. It is also used as overstruck-character translation table. If TRNAOP is zero, this translation does not occur.

TRNAIP
The address of the input symbol set 1 (APL) translation table. If TRNAIP is zero, this translation does not occur and character codes revert automatically to symbol store 0.

TRNBPOP
The address of the output-protected approximation translation table. If TRNBPOP is zero, this translation does not occur.

TRNFLAGS
A set of flags that specify the combination of device characteristics for which the translate table is used. Some flags may be ignored, depending on the setting of TRNFMASK in the basic control information.

TRN3277 Indicates the type of display:

1 Device is a 3277
0 Device is not a 3277

TRNDISP Indicates the type of device:

1 Device is a display
0 Device is a printer

TRNAPL Indicates whether the device has an APL character set:

1 Device has an APL character set
0 Device does not have an APL character set

TRNUCOT Indicates whether the device displays lowercase characters:

1 Device must be one that does not display lowercase characters in a recognizable form (for example, a Katakana device). This setting affects the operation of the ASCGET call when ASFTRN has been used to specify uppercase translation of alphanumeric input.
0 Device must display lowercase characters in a recognizable form. This setting is valid for those devices that display lowercase characters as uppercase.

This flag is not used in device-characteristic matching.

TRNSYSP Indicates whether the device is a system printer:

1 Device is a system printer (for example, 1403 or 3211).
0 Device is not a system printer

TRNDEFR Indicates whether this type number is processed:

1 Do not process this type number. Go to the translation-type number specified on TRNDEFRT and continue searching from there.
0 Process in the normal way.

This flag is not used in device-characteristic matching.

TRNIG Indicates whether the device supports the APL2 character set:

1 Device is an interactive graphics device that supports the APL2 character set (for example, a 3179-G or a 3270-PC/G)
0 Device does not support the APL2 character set

TRN293 Indicates whether the device supports APL code page 293:

1 Device supports APL code page 293
0 Device does not support APL code page 293

TRNNONU Indicates whether the device has national-use characters:

1 Device has no national-use characters
0 Device has national-use characters

TRN1027 Indicates whether the device has Japanese (Latin) SBCS characters:

1 Device has Japanese (Latin) SBCS characters
0 Device does not have Japanese (Latin) SBCS characters

TRNIIC
Invalid-character replacement

TRNCBOP
The address of the CECP base output table. It may be zero when TRNDEFR=0.

TRNCBIP
The address of the CECP base input table. It may be zero. It is used when TRNDEFR=0, and must be 0 when TRNDEFR=1.

TRNCAOP
The address of the CECP alternate output table. It may be zero. It is used when TRNDEFR=0, and must be 0 when TRNDEFR=1.

TRNCAIP
The address of the CECP alternate input table. It may be zero. It is used when TRNDEFR=0, and must be 0 when TRNDEFR=1.

TRNCPOP
The address of CECP base protected output table. It may be zero. It is used to approximate protected field output when TRNDEFR=0, and must be 0 when TRNDEFR=1.

Translation tables

The translation tables are the third part of the alphanumeric defaults module. The object of GDDM translation of alphanumeric fields is to:

- Give consistency of interpretation of hexadecimal character codes between different languages
- Give consistency of interpretation of hexadecimal character codes between different devices
- Give an approximation of undisplayable codes where this is likely to help the user
- Allow for APL characters.

ADMDATRN contains two sets each of five tables: three for output and two for input, for each device type. These tables are used to translate alphanumeric fields.

One set of tables handles non-CECP applications. The other set is for CECP applications running on non-CECP devices. Figure 43 on page 190 shows how the tables are used. Both sets of tables work in the same way. The tables and their uses are described in the remainder of this chapter.

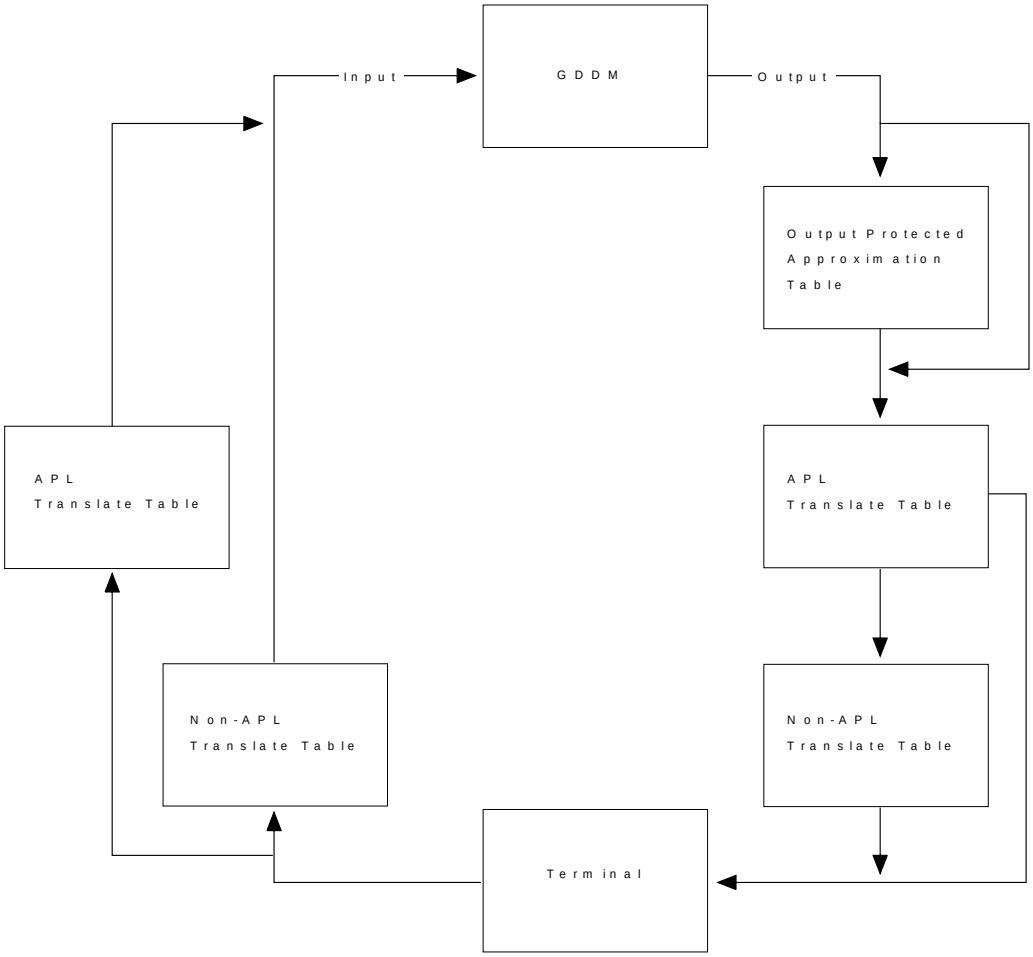


Figure 43. Translation tables used for GDDM alphanumeric fields

The non-CECP translation tables are used only for the read-only symbol sets supplied in the display terminal hardware. No translation occurs if you use other symbol sets in alphanumeric fields.

Translation is performed on one-byte characters that use ASCPUT only. Double-byte characters using ASGPUT are not translated.

Setting up a new translation table

Instructions for setting up a new translation table to be accessed by the ASTYPE call are given below. You should not try to do this before you have studied ADMDATRN and the comments it contains.

1. Set up a new translation group by copying either group 3 (for Katakana terminals) or group 2 (for other terminals). Make this group number 4.
2. Change the translate-type numbers. Any number can be used, provided it is unique within the module.
3. Set up the appropriate translate tables, using those in ADMDATRN as a model.

4. Set the entries in group 4 so that they point to the appropriate translate tables.
5. If you want the new table to become the default, change group 0 so that it points to group 4. If you do not want it as the default, you can specify it explicitly by using ASTYPE(4).
6. Refer to Chapter 19, “Updating GDDM default modules” on page 213 for information on how to replace the supplied module ADMDATRN.

Device-dependent translation tables

Device-dependent translation tables process both CECP and non-CECP alphanumeric fields. The pointers to the tables are in the translation-type descriptor block.

Output-protected approximation table

For protected fields only. This table translates characters that cannot be shown on the device to the nearest possible equivalent. (For example, superscript values may be shown as ordinary values.) The table is used only for protected fields because the data stream could be corrupted if the translated value were returned. If the character cannot be approximated, a blank is used.

The table is addressed by TRNBPOP for non-CECP, and by TRNCPOP for CECP.

Output symbol set 0 (non-APL) translation table

Translates the hexadecimal code sent by GDDM to the code that results in the required character appearing on the terminal. The table is addressed by TRNBOP for non-CECP, and by TRNCOP for CECP.

Output symbol set 1 (APL) translation table

Detects and translates APL characters for interactive devices. APL characters are known to GDDM as hexadecimal values (for example, A underscore is represented as X'41'). On some devices, this value is represented as X'81' in symbol set 1. The output symbol set 1 translation table translates characters in symbol set 1 to their symbol set 1 value (for example, X'41' to X'81'), and translates characters in symbol set 0 to zero, in order to distinguish them from the symbol set 1 characters.

The same table is used for output of APL2 characters to 3287-type printers with the appropriate APL2 feature.

APL translation is done after output-protect translation and before non-APL translation. As shown in Figure 43 on page 190, characters translated by the APL translation table are not passed through the non-APL translation table.

This table is addressed by TRNAOP for non-CECP, and by TRNCAOP for CECP.

Input symbol set 0 (non-APL) translation table

Translates the hexadecimal code sent by the device to the code that will be expected by GDDM, which is the reverse of the equivalent output table.

This table is addressed by TRNBIP for non-CECP, and by TRNCIP for CECP.

Input symbol set 1 (APL) translation table

Translates APL characters back into the format recognized by GDDM.

This table is addressed by TRNAIP for non-CECP, and by TRNCAIP for CECP.

In addition to this translation, GDDM removes invalid characters that might corrupt the device, and replaces them with the Invalid Device Character. A record is kept of the position of these characters and they are replaced in the field on input if they have not been changed.

CECP lookup table

Table 24. Alphanumeric defaults module, CECP lookup table

Field name	Field offset	Field length (in bytes)	IBM default value (where applicable)
TRNCSID	0	4	–
TRNCPID	4	4	–
TRNCPFG	8	4	–
TRNCECPF	x.......	Bits	1
TRNACPF	.x......	within	1
TRNAROSF	..x.....	TRNCPFG	1
Reserved	...xxxxx		–
Reserved	xxxxxxxx	1	–
TRNASCII	10	1	–
TRNADJCO	12	4	Address
TRNFMLP	16	4	Address
TRNTMLP			Address

TRNCSID
Character-set identifier

TRNCPID
Code-page identifier

TRNCPFG
Code-page flags:

TRNCECPF	CECP flag. Set if there is a CECP-coded character set.
TRNACPF	Application code-page flag. Set if the code page is supported as an application or installation code page. If TRNACPF is not set, the coded character set is used as a device code page for input/output operations.
TRNAROSF	ROS translation flag. Set for coded character sets on devices supporting 94 EBCDIC characters.
Reserved	Five unused bits.
Reserved	Eight unused bits.
TRNASCII	Index into ASCII code-page index table.
TRNADJCO	NLS table adjust code. Used to match to a code in the adjust table. Set if 3279 devices in specific countries support more than 94 characters.

TRNFMLP
Pointer to the “translate-from” multilingual page 500 table.

TRNTMLP
Pointer to the “translate-to” multilingual page 500 table.

There is no set order for the entries in this table, except that:

- Entries for possible application code pages must be placed first.

- The last entry must have a zero TRNCSID and TRNCPID.

Other entries can be arranged to optimize performance.

This table is used to validate and interpret code-page identifiers. It has an entry for each coded character set recognized by GDDM. These correspond to:

- Pages that can be supported as the application or installation code page.
- Additional device code pages.

Each entry contains character-set and code-page identifiers, flags, and pointers to translation tables.

The table is addressed by TRNCPL in the basic control information at the start of module ADMDATRN.

Code-page-to-code-page translation tables

Translation tables are supplied for translation from code page 500 to every other supported code page, and the converse. (It is also possible to translate directly from one code page to another, without going via code page 500. More information about this is provided in “Direct code-page translation” on page 195.)

The table is addressed by TRNFMLP in the CECP lookup table for translating *from* code page 500, and by TRNTMLP in the CECP lookup table for translating *to* code page 500.

National-language-to-CECP lookup table

Table 25. Alphanumeric defaults module, national language lookup table

Field name	Field offset	Field length (in bytes)	IBM default value (where applicable)
TRNNATL	0	1	–
Reserved	1	3	–
TRNNLCP	4	4	–

TRNNATL
National-language identifier.

TRNNLCP
Code-page identifier.

Valid combinations of national language identifier and code page identifier are shown in Table 26 on page 194.

Table 26. National-language and code-page identifiers for TRNNATL and TRNNLCP

National language	Code page	Language
A	00037	U.S.-English
B	00037	Brazilian Portuguese
C	00037	PRC Chinese
D	00277	Danish
F	00297	French
G	00273	German
H	00037	Korean (Hangeul)
I	00280	Italian
K	00037	Japanese (Kanji)
N	00277	Norwegian
Q	00037	Canadian French
S	00284	Spanish (Latin America)
T	00037	Taiwan
V	00278	Swedish
0	00000	End of table

The national-language-to-CECP lookup table is used to access the message data set. It is addressed by TRNNATP in the basic control information at the start of ADMDATRN.

CECP uppercase translation table

The CECP uppercase translation table for code page 500 is used by ASFTRN to translate characters to uppercase on input. The table is addressed by TRNMUCTP in the basic control information at the start of ADMDATRN.

ASCII code-page tables

The ASCII code-page tables are a set of EBCDIC-to-ASCII translation tables used for translation of graphics characters output on ASCII graphics terminals. Translation of alphanumeric characters to and from ASCII is performed by the 3174 AEA controller.

ASCII code-page index table:

Table 27. ASCII code-page index table

Field name	Field offset	Field length (in bytes)	IBM default value (where applicable)
TRNADEC7	0	4	Address
TRNADEC8	4	4	Address
TRNATEK7	8	4	Address
TRNATEK8	12	4	Address

TRNADEC7

Pointer to table for translation from code page 500 to DEC 7-bit ASCII.

TRNADEC8
Pointer to table for translation from code page 500 to DEC 8-bit ASCII.

TRNATEK7
Pointer to table for translation from code page 500 to Tektronix 7-bit ASCII.

TRNATEK8
Pointer to table for translation from code page 500 to Tektronix 8-bit ASCII.

This table is indexed by TRNASCII in the CECP lookup table.

ASCII output translation tables: Tables that support translation from code page 500 to supported ASCII code pages are addressed by TRNADEC7, TRNADEC8, TRNATEK7, and TRNATEK8 in the ASCII code-page lookup table. Unsupported code points are translated to the nearest approximation or to blanks.

Direct code-page translation

FSTRAN can translate from one code page to another without going via the intermediate code page 500. This provides better translation of some ASCII-to-EBCDIC code pages used by GDDM during CGM input and output. The mechanism is not restricted to this use. Any code-page pair can be translated in this way if you provide a suitable translation table.

TRNEXTP in the basic control information is the address of the ADMDATRN extension table. TRNDTRNP in the extension table is the address of the direct-translation lookup table.

Table 28. Direct-translation lookup table

Field name	Field offset	Field length (in bytes)	Description
TRNDCSFR	0	4	Character set of “from” code page
TRNDCPFR	4	4	Code page id of “from” code page
TRNDCSTO	8	4	Character set of “to” code page
TRNDCPTO	12	4	Code page id of “to” code page
TRNDTA	16	4	Address of 256-byte translate table

The direct translation lookup table contains one entry for each translation table and is terminated by “–1” in the TRNDCSFR position. The translation tables supplied are 850 to 351 and the reverse, and 437 to 351 and the reverse.

End of General-use programming interface

Chapter 17. GDDM user-default specifications

General-use programming interface

During its operation, GDDM uses many values that you can alter to customize GDDM for your installation. These values are known as *user-default specifications* (UDSs). GDDM uses five types of UDS:

ADMMDFT The ADMMDFT statement defines an **external default**. External defaults define a variety of system values, such as buffer sizes, conventions for time and date expressions, the national language of your installation, and the code page to be used. Many of the external defaults are subsystem dependent. Appendix A, “External defaults” on page 307 describes all of the external defaults. You will also find them referred to as appropriate throughout this manual.

ADMMNICK The ADMMNICK statement defines a **nickname**. The nickname is a powerful mechanism for defining how devices are to be used. Nicknames are used to specify **processing options** (also known as “procopts”) and **device tokens** to GDDM. Nicknames are described in Chapter 18, “Nickname user-default specifications” on page 205.

ADMMEXIT The ADMMEXIT statement defines a **user exit**. User exits allow a system program to trap specific events whenever an application program uses a GDDM resource or a system resource. For more information about user exits, see the *GDDM Base Application Programming Reference* book.

ADMMCLTB This UDS enables you to modify the color table used by GDDM when applications generate PostScript output to ensure that GDDM colors are represented accurately. It can also be used to change the color table used by the ADMUGIF utility. Each GDDM color number is defined in terms of RGB and CMYK values.

For more information about using this UDS definition, see “Specifying the color mapping using the ADMMCLTB UDS” on page 153.

ADMMTYPF This UDS enables you to specify how applications that use GDDM symbol sets and presentation-text fonts are to map those symbol sets and fonts to typeface names when a PostScript file is generated.

For more information about using this UDS definition, see “Specifying symbol-set and font mapping using the ADMMTYPF UDS” on page 156.

This chapter describes how to specify UDSs to GDDM.

Ways of specifying user-default specifications

There are several ways of specifying a UDS to GDDM: which of these methods you use depends on the intended scope of the specification.

- If the UDS is to apply within an application only, it can be coded on an ESSUDS, ESEUDS, or SPINIT call. You can code your own access routine for GDDM applications containing these calls.
- If the UDS is to apply to a single user or group of users, it can be coded in an *external defaults file*. This mechanism is not available to IMS users, and its use is restricted under CICS. Although you can use an external defaults file to specify UDSs for all users, this is not recommended: it can be overridden accidentally, and performance is better if an *external defaults module* is used.
- If the UDS is to apply to all users, create an external defaults module. Because the effects of the external defaults module are usually system-wide, its customization and maintenance are considered to be system-support tasks. You should aim to have an external defaults module that meets the needs of the majority of your users in most circumstances. This will minimize the need for one-off enhancements, and allow the operation of your GDDM system to be largely transparent to your users.

Note that not all external defaults can be specified using all of these methods. Appendix A, “External defaults” on page 307 identifies, for each external default, any restriction on the ways in which it can be specified.

When a particular value is set using more than one of these methods, GDDM assigns these priorities to the methods to determine which of the settings is implemented:

1. ESSUDS or ESEUDS
2. SPINIT
3. External-defaults file
4. External-defaults module

That is, an ESSUDS or ESEUDS call overrides an SPINIT call, SPINIT overrides an external defaults file, and an external defaults file overrides the external defaults module.

Formats of user-default specifications

The UDS can be specified in three different formats. These formats vary according to the method (API call, external defaults file, or external defaults module) being used.

The ESSUDS call and the external defaults file
On the ESSUDS call and in the external defaults file, you supply a *source-format* UDS. A source-format UDS is in an English-like form.

The external defaults module
UDSs in the external defaults module are in executable format, which results from assembling source-format UDSs.

The ESEUDS call and the SPINIT call
On the ESEUDS and SPINIT calls, you specify an *encoded* UDS. Encoded format is a string of hexadecimal characters that can be generated manually or by processing a file in executable format.

The rules governing how a source-format UDS is coded vary slightly, depending on whether the UDS is to be assembled in an external defaults module. The next section describes the requirements for a UDS that is to be assembled. "Coding UDSs for an external defaults file" on page 202 describes the ways in which the syntax rules can be relaxed for a UDS that is to be used in source format only.

Coding UDSs for an external defaults module

The assembler statements for the UDS must be in the following format:

```
[label] type value [comment]
```

If the length of *value* is greater than 72 characters, the format becomes:

```
[label] type value-part-1, [comments] x
             value-part-2, [comments] x
                   .
                   .
             value-part-n  [comments]
```

label
is optional. If specified, it must start in column one and be no more than 8 characters. It is ignored by the assembler. Any text that starts in column one is assumed to be part of a label.

type
identifies the type of UDS. This is one of ADMMDFT, ADMMNICK, and ADMMEXIT. The type value must be preceded by at least one blank, so cannot start in column one.

value
is the parameter you want to change. It must not contain embedded blanks (except when it is within quotation marks), it must be preceded by at least one blank, and can be coded in columns 2 through 71. If *value* is too long to fit on one line, you can split it at the commas that appear in the value string because these are used as continuation characters within the value string.

comments
is optional additional text that can be coded to explain why the UDS has been specified, for example, or to clarify its effects. The comment text must be separated from the value information by at least one blank.

x is an assembler continuation character that must be coded in column 72 if *value* is to occupy more than one line.

Comment records can be included. They must begin with an asterisk (*) in column 1 and are ignored by GDDM.

Creating an external defaults module

The name of the external defaults module depends on the subsystem under which GDDM is running. GDDM supplies assembler-language samples that you can use to create your own external defaults module. The names of the external defaults modules, and the location of the assembler samples, are as follows:

Table 29. External defaults: module names and locations

Module name	Subsystem	Where to find assembler sample
ADMADFV	VM/CMS	ADMADFV ASSEMBLE file
ADMADFT	TSO	Member ADMADFT in the SADMSAM data set
ADMADFC	CICS/MVS	Member ADMADFC in the SADMSAM data set
	CICS/VSE	A-type member ADMADFC
ADMADFI	IMS	Member ADMADFI in the SADMSAM data set
ADMADFD	VSE batch	A-type member ADMADFD

The ADMADFC sample contains some examples that can be used for batch printing from CICS/VSE. These lines should be deleted if you use the sample for CICS/MVS.

When you are creating your external defaults module, ensure that you create a new copy on your own storage. If you update the supplied sample in the system version, it could be overwritten by servicing or upgrading GDDM and any customized copy will be lost. Either copy the supplied assembler statements, or create your own external defaults module in the following format:

```
ADMADFx CSECT

        ADMMDFT START

[label] type value     [comments]
[label] type value     [comments]

⋮
[label] type value     [comments]

        ADMMDFT END

        END
```

Figure 44. Structure of an external defaults module. `ADMADFx` *is the subsystem-dependent name of the external defaults module. It must be one of ADMADFV, ADMADFT, ADMADFC, ADMADFI, and ADMADFD.*

Each UDS in the file must adhere to assembler-language conventions. In particular (assuming no ICTL instruction has altered the assembler conventions):

- Continuations must begin before column 17.
- Continuations must be flagged by a nonblank character in column 72.
- The source-format UDSs must be entered in uppercase. GDDM does not convert lowercase characters to uppercase.

The order of the UDSs within the module is not significant.

For information on how to assemble an external defaults module and use it to replace the supplied external defaults module, see Chapter 19, “Updating GDDM default modules” on page 213.

An example external defaults module

```
ADMADFx CSECT
     ADMMDFT START
************************************************************************
* The following ADMMDFT statement changes the default national        *
* language to French; note the inline comment text:                   *
*                                                                     *
     ADMMDFT NATLANG=F  CHANGE THE NATIONAL LANGUAGE
*                                                                     *
************************************************************************
* The following ADMMDFT statement changes the default filetype        *
* of page printer output in the VM environment from ADMIMAGE to       *
* LIST3820:                                                           *
*                                                                     *
     ADMMDFT CMSMONO=LIST3820
*                                                                     *
* The same function is available to TSO users (using the TSOMONO      *
* external default) and to VSE users (using the VSEMONO external      *
* default).                                                           *
************************************************************************
* The following ADMMDFT statement sets the country-extended code page *
*                                                                     *
     ADMMDFT APPCPG=285
*                                                                     *
************************************************************************
* The following ADMMNICK sets up PA2 as the key to access User Control*
* and, for VM systems, PA1 to be the CMS interrupt key:               *
*                                                                     *
     ADMMNICK FAM=1,PROCOPT=((CTLMODE,YES),(CTLKEY,4,2),(CMSINTRP,PA1))
*                                                                     *
************************************************************************
* The following ADMMNICK lets you print or plot from User Control:    *
*                                                                     *
     ADMMNICK FAM=1,PROCOPT=((CTLPRINT,YES))
*                                                                     *
************************************************************************
* The following ADMMNICK statements allow the use of GDDM-PCLK.  Any  *
* request for printing or plotting that uses the name PCPLOT is       *
* directed to the first (or only) plotter that is attached to the     *
* workstation.  A similar request using the name PCPRINT goes to the  *
* attached printer:                                                   *
*                                                                     *
     ADMMNICK FAM=1,PROCOPT=((PCLK,YES))
     ADMMNICK FAM=0,NAME=PCPLOT,TOFAM=1,TONAME=(*,ADMPLOT)
*                                                                     *
* The next nickname is split to show how continuation works:          *
*                                                                     *
     ADMMNICK FAM=0,NAME=PCPRINT,                                    X
               TOFAM=1,TONAME=(*,ADMPCPRT)
*                                                                     *
************************************************************************
* The following ADMMNICK statement sets the nickname for a 5080       *
* display:                                                            *
*                                                                     *
     ADMMNICK PROCOPT=((SPECDEV,IBM5080,MY5080))
*                                                                     *
************************************************************************
     ADMMDFT END
     END
```

Figure 45. Some example external defaults and nicknames. For more information about nicknames, see Chapter 18, "Nickname user-default specifications" on page 205.

Coding UDSs for an external defaults file

An external defaults file contains one or more source-format, user-default specifications. It can be used without restriction under VM/CMS and TSO. However, it cannot be used under IMS, and is intended to be used only for testing or diagnosis purposes under CICS.[9]

Under MVS, you allocate the DD name ADMDEFS to the sequential data set that is to contain the external defaults data. Under VM, the external defaults file is called PROFILE ADMDEFS. These names can be altered using the TSODFTS and CMSDFTS external defaults respectively. (TSODFTS and CMSDFTS can be specified in encoded UDSs only.) The external defaults file is usually held in storage local to the user for whom it has been provided.

The external defaults file can be in fixed or variable record format and must have a record length no greater than 255 bytes. On TSO, it can be either a sequential data set or a member of a partitioned data set. Also, on TSO, the external defaults file must not be a null data set: it must contain at least one record.

The format of an external defaults file is the same as that of an external defaults module, but with some relaxations of the syntax rules. See “Coding UDSs for an external defaults module” on page 199 for the rules that apply to the external defaults module. The external defaults file differs from an external defaults module in the following ways:

- The assembler CSECT and END op codes must not be supplied.
- The ADMMDFT START and ADMMDFT END UDSs must not be supplied.
- The continuation character in column 72 is not required. A comma is sufficient to indicate that a value continues on the next line.
- Text can be entered in mixed case. GDDM converts lowercase characters to uppercase before processing, unless they are within quotation marks. Values within quotation marks remain as entered.
- The type ADMMDFT can be specified as DEFAULT, and the type ADMMNICK can be specified as NICKNAME.

Create a file or data set for your external defaults file and enter the UDSs you require.

Accessing an external defaults file

On VM, the UDSs are accessed by anyone who links to the disk that contains the ADMDEFS PROFILE file you have created. Therefore, ensure that you restrict access to this file to those users for whom the UDSs are intended. Also, remember that, if you have just created the file, users must reaccess the disk to find the file.

On MVS, the UDSs are accessed only by allocating the ddname ADMDEFS to point to the data set you have created. Remember that, if you have a security product that restricts access to data sets, other users might need read access if

[9] The role of the external defaults file in the diagnosis task is described in the *GDDM Diagnosis* book.

you intend them to use your UDSs. You can allocate the ddname for TSO users:

1. In the logon procedure
2. In a CLIST executed by the logon procedure
3. As part of an application CLIST

Coding a UDS for an ESSUDS call

The ESSUDS call passes to GDDM a single UDS in the same format as the assembler source for an external defaults module. See “Coding UDSs for an external defaults module” on page 199 for a description of this format. To supply multiple UDSs using the ESSUDS call, code one call for each UDS. The ESSUDS call is recommended rather than ESEUDS or SPINIT as it is easier to code. However, there are times when the SPINIT call must be used rather than ESSUDS. Appendix A, “External defaults” on page 307 identifies whether ESSUDS can be used for particular UDSs.

Here is an example that shows how to code ESSUDS to set the default national language to French:

```
ESSUDS (17,'ADMMDFT NATLANG=F')
```

where `'ADMMDFT NATLANG=F'` is the required UDS, and 17 is its length in bytes. For more information about the format of the ESSUDS call, see the *GDDM Base Application Programming Reference* book.

Coding a UDS for an SPINIT or ESEUDS call

The ESEUDS and SPINIT calls include a list of UDSs in encoded format. Encoded format is also used in the object deck of an external defaults module. Therefore, if you are going to code a list of UDSs on SPINIT or ESEUDS, you are recommended to create a file in the format of an external defaults module, assemble it, and use the assembler listing to get the encoded format information. For more information about the ESEUDS and SPINIT calls, see the *GDDM Base Application Programming Reference* book.

End of General-use programming interface

Chapter 18. Nickname user-default specifications

The GDDM Base API call DSOPEN defines to GDDM the device for which the issuing application is creating its output. The device definition supplied on DSOPEN can be complete, in which case the application is device dependent: if anything about the device changes, the application itself has to be altered. Alternatively, a partial device definition can be supplied on input to DSOPEN. A partial definition would include a device identifier, a family identifier, and a device name.

A nickname, which is defined in a user-default specification (usually in the external defaults module or in an external defaults file, as described in Chapter 17, "GDDM user-default specifications" on page 197) supplies all the other information about the device. In particular, the nickname can provide a *device token* and some *processing options*. It can also redirect the application's output to a different family and device from those specified on DSOPEN, thereby extending the range of devices supported by the application. This mechanism allows you to alter the definition and usage of any device, without requiring updates to be made to applications that use it.

You are recommended to add a set of nickname UDSs to the external defaults module that describes all printers and plotters in your enterprise, for use by all GDDM users. This simplifies the tasks of the application programmer and end user, without depriving them of the ability to override such definitions.

The format of the nickname UDS

General-use programming interface

The syntax of a nickname UDS is as follows:

```
[label] ADMMNICK [APPEND|REPLACE]
                 [,FAM=family]
                 [,NAME=name-list]
                 [,TOFAM=to-family]
                 [,TONAME=to-name-list]
                 [,DEVTOK=device token]
                 [,PROCOPT=processing-option list]
                 [,DESC='description'|"description"]
```

label
The *label* value is optional and is ignored by GDDM. Any word starting in column one is assumed to be a label.

APPEND|REPLACE
Specifies whether this nickname UDS is to replace or be added to other nickname UDSs with the same FAM= and NAME= values. (For more information about the effects of multiple nickname UDSs, see "Which nickname is implemented?" on page 208.)

FAM=0|1|2|3|4
Identifies the GDDM printing family (1,2,3, or 4) to which this nickname statement applies. For example, a value of `FAM=2` specifies that the other information supplied on this nickname statement applies only when family-2

output has been requested on the DSOPEN call. (Note that the GDDM-PGF ICU and User Control both produce family-2 output. If you want to define nicknames that affect the output from the ICU and User Control, you must set FAM=2.)

For a nickname statement to be usable by a DSOPEN call, the FAM= value on the nickname statement and on the DSOPEN call must be the same. The default setting is FAM=0, which means that this nickname statement can be applied to *any* family specified on the DSOPEN call. The same effect is achieved by specifying FAM=**,**, or omitting the FAM= parameter.

NAME=*name-list*

Identifies the physical destination of the output. This varies according to both the subsystem and the GDDM printing family. For example, the destination of family-2 output in the VM/CMS environment is an intermediate file (of type ADMPRINT, by default), whereas the destination of family-4 output in the TSO environment is a print data-set. For more information about possible name-list values, see Appendix D, “Name-lists” on page 383.

The *name-list* parameter comprises one or more name-parts, each of which is a string of 0 through 8 nonblank characters. The following are valid name-lists:

NAME=name1**,**	— one nonblank name-part
NAME=(name1)**,**	— one nonblank name-part
NAME=()**,**	— one blank name-part
NAME=(name1**,**name2**,**name3)**,**	— three nonblank name-parts
NAME=(name1**,,**name3)**,**	— two nonblank name-parts and one blank name-part
NAME=**,**	— null name.

For a nickname statement to be usable by a DSOPEN call, the name-list values on the DSOPEN call and on the nickname UDS must match, or that specified on the nickname statement must be null. The two name-lists are considered to match if the corresponding name-parts are the same (after left-justification, translation to uppercase, and padding with blanks). If the name-lists do not contain the same number of name-parts, the shorter of the two name-lists is extended with “*” name-parts for the purpose of comparison. A value of NAME=* identifies the default device, which is the user console. In each of these three examples, the two name-lists are considered to match:

1. 'FRED' and '(FRED)'
2. 'FRED' and '(FRED**,***)'
3. 'FRED' and '(FRED**,***,***)'

'FRED' and '(FRED**,**ADMPRINT)' do not match.

A name-part can contain a leading and a trailing “?” character, which is considered to match any combination of characters in the same position. For example:

'abc?'	— matches any name-part starting with ‘abc’
'?'	— matches *any* name-part

Embedded “?” characters are invalid. The default is null (that is, NAME=**,**), which is considered to match any name-list specified on the DSOPEN call.

TOFAM=<u>0</u>|1|2|3|4

Allows output assigned to one GDDM printing family (on the DSOPEN call and on the FAM= parameter of this nickname statement) to be assigned to a

different family. In effect, this allows rerouting of output from one printer to another. For example, if a nickname statement specifies FAM=2 and TOFAM=4, family-4 output is created instead of family-2 output.

TOFAM=0, which is the default value, leaves the FAM= value unaltered.

TONAME=*to-name-list*

Replaces the name-list specified on NAME= with a different name-list. If a nickname statement contains a TOFAM= value that changes the print family, a TONAME= value is also likely to be be required.

The *to-name-list* has the same format as the *namelist* value, except that "?" characters are not supported. If *to-name-list* is null, the NAME= name-list value is left unaltered.

DEVTOK=*device token*

A string of 0 through 8 nonblank characters that identifies some of the characteristics of the target printer or plotter, such as its model number and paper size. GDDM supplies a large set of device tokens, which are listed in Appendix C, "Device tokens supplied by GDDM" on page 367. You can also define your own. How to do this is described in Chapter 10, "Creating your own device tokens" on page 101.

The device token must be suitable for the TOFAM= and TONAME= values. If you specify a device token on the DEVTOK= parameter, it takes effect only if the device-token value on the DSOPEN call is set to "*" or blank. If a device token is explicitly named on the DSOPEN call, it cannot be overridden.

Device tokens tell GDDM the characteristics of the device, thereby suppressing the usual "device query." In general, it is better to let GDDM query a device at the time output is produced than to provide a device token on the DSOPEN call. However, there are circumstances in which GDDM cannot query a device and in which a device token should therefore be supplied. These are:

- If GDDM is running under IMS
- If GDDM does not have direct access to the device (usually a printer, typically managed by some other program, such as PSF)
- If a dummy device (one not yet physically present) is used

Even in these cases, it is still better to specify the device token on a nickname statement than on the DSOPEN call.

PROCOPT=*processing-option list*

Processing options (procopts) control the details of an output process. For example, you can use a processing option to specify whether swathes are to be used to process a high-resolution image (HRISWATH), to specify the speed of plotter pens (PLTPENV), and to enable user control (CTLMODE). For a full list of processing options and their values, see Appendix B, "Processing options" on page 333.

PROCOPT= specifies a list of procopts as follows:

```
PROCOPT=((procopt-spec),(procopt-spec),...)
```

Each `procopt-spec` comprises a keyword followed by a number of arguments valid for that processing option:

```
PROCOPT=((keyword,argument,argument),
         (keyword,argument), ...)
```

Procopts specified on the DSOPEN call cannot be overridden.

DESC=*'description'|"description"*

A string of 0 through 72 bytes that describes the printer or plotter. The string can contain mixed double-byte (DBCS) and single-byte (SBCS) characters. SBCS characters must be encoded using the installation code page. The string should be enclosed in double quotation marks if there are single quotation marks within the string itself. By entering "?" in the "Name of printer or plotter" field of the User-Control output panel or in the ICU Printer Selection Panel, users can display the description of any printer or plotter that has a namelist of 1 through 8 characters. When displayed on the ICU Printer Selection Panel, descriptions are truncated to 60 bytes, and are listed in the reverse of the order in which they are defined in the external defaults module or file.

You are recommended to supply a description of any printer or plotter to which end users are likely to submit output.

Which nickname is implemented?

It is possible for multiple nickname UDSs to be specified for the same family (FAM=) or device name (NAME=), or both. GDDM maintains a "nickname list" containing all current nickname UDSs in the following order:

1. Those defined on ESEUDS and ESSUDS calls, in the order in which the calls were made
2. Those defined on SPINIT calls
3. Those defined in an external defaults file
4. Those defined in an external defaults module

ESEUDS and ESSUDS calls override SPINIT calls, SPINIT calls override external defaults files, and external defaults files override the external defaults module.

When an application issues a DSOPEN call, GDDM constructs a "source" parameter list and a "target" parameter list, each of which contains these values as specified on the DSOPEN call:

Printing family
Name-list
Device token
Procopt list

In other words, GDDM constructs two identical lists using values specified on DSOPEN.

GDDM then scans the "nickname list" (the list of current nickname UDSs), looking for nicknames whose FAM= and NAME= parameters match those stored in the source and target parameter lists. When a match is found, GDDM updates the *target* parameter list (but not the source parameter list) with the TOFAM, TONAME, DEVTOK, and PROCOPT parameters of the matching nickname UDS according to the rules given in "The format of the nickname UDS" on page 205.

When GDDM has worked through the entire nickname list, the source parameter list is updated with the values from the target parameter list. GDDM then scans the nickname list again, looking for nickname UDSs that match the modified source parameter list, and updates the target parameter list when a match is found.

Any nickname UDSs that are found to match on any scan or rescan of the nickname list are excluded from subsequent rescans for this DSOPEN call. This is

so, even if a nickname UDS was ignored because the REPLACE parameter was specified on a later nickname UDS.

When GDDM completes a scan of the nickname list without finding a match, the final version of the target parameter list is implemented.

A note about processing options: As the scan proceeds, procopt specifications are inserted in the target parameter list such that they follow any procopt lists added so far during the current scan, but are ahead of any that were added to the list during a previous scan. In general, where specifications for identical procopts are added to the list, the one closest to the top of the list is implemented. The exceptions to this rule are the PRINTCTL and ORIGINID procopts, whose effects are cumulative. Any procopt specifications that do not apply to the current device family and name are ignored.

Directing family-2 output to family-4 devices

It is possible (by specifying `FAM=2,TOFAM=4`) to redirect family-2 output to family-4 devices. This enables you to migrate existing applications (and their users) to the more sophisticated printing capabilities of advanced-function printers, without requiring changes to be made to applications or to the way in which they are used. It also minimizes the use of intermediate files and operations involved in family-2 printing.

Family-4 devices allow objects to be positioned on the page in arbitrary units down to pel accuracy. (That is, they are all-points addressable.) All other device families (1,2, and 3) position objects by rows and columns of alphanumeric cells. To cater for this difference, GDDM provides some family-4 device tokens that specify, in addition to the device resolution and line width:

- The font and the alphanumeric grid to be used for emulation of alphanumerics
- Support for graphics output
- Support for image output
- The size of the usable area (to allow for printing margins)

Thus, if the device token defines the page size in cells, the output is positioned in cells.

These tokens, which are listed in Appendix C, “Device tokens supplied by GDDM” on page 367, facilitate production of AFPDS output from existing family-2 applications.

Most paper sizes are catered for by these device tokens. If you want to use a size of paper that the supplied tokens do not support, you can define additional tokens. How to do this is described in Chapter 10, “Creating your own device tokens” on page 101.

Some example nickname UDSs

This section provides an assortment of example nickname statements to give you an understanding of the ways in which they can be used.

1. `ADMMNICK FAM=1,NAME=PLOT90,PROCOPT=((PLTROTAT,YES))`

   Effect: If anyone opens a directly attached (family-1) device called PLOT90, apply this processing option to rotate the plot.

2. `ADMMNICK NAME=*,FAM=1,PROCOPT=((CTLKEY,1,11))`

   Effect: On all user consoles (NAME=*,) that are family-1 devices, the key to activate User Control is PF11. (By default, the key to activate User Control is PA3.)

3. ```
   ADMMNICK NAME=PLOT9,FAM=2,TONAME=(*,ADMPLOT),
                       TOFAM=1,PROCOPT((PLTROTAT,NO),(PLTPENV,50))
   ```

   Effect: If anyone opens a family-2 device called PLOT9 (for example, if a user enters the device name “PLOT9” in the GDDM-PGF ICU Output Panel), change the device to (*,ADMPLOT) and the family to 1. A file of output called PLOT9, that would otherwise have been created on the user’s A-disk for printing by one of the GDDM print utilities, is now directed to a plotter called ADMPLOT attached to the user console (*). The output is plotted in landscape (horizontal) orientation at a velocity of 50 centimeters per second.

4. `ADMMNICK NAME=MYPRINT,FAM=2,DEVTOK=X4224SE`

   Effect: If anyone opens a family-2 device (that is, creates an intermediate file of output) called MYPRINT, apply the device token “X4224SE.” (For a description of the effects of this device token, see Appendix C, “Device tokens supplied by GDDM” on page 367.)

5. `ADMMNICK FAM=1,PROCOPT=((PCLK,YES))`

   Effect: Enables output from GDDM applications to displays attached via GDDM-PCLK for all users with access to this nickname statement. If this processing option is not set, GDDM-PCLK cannot be used.

6. `ADMMNICK NAME=PCPRINT,FAM=0,TOFAM=1,TONAME=(*,ADMPCPRT)`

   Effect: If any GDDM application directs output to device PCPRINT, send it to the printer attached to the user console. Provided that PROCOPT=((PCLK,YES)) has been specified, this nickname statement enables output to PCLK-attached plotters from GDDM graphics applications.

7. ```
   ADMMNICK FAM=0,NAME=DEFPRT,TOFAM=1,
            TONAME=(*,ADMPMOP),DESC='Default OS/2-attached printer'

   ADMMNICK FAM=0,NAME=MYPRINT,TOFAM=1,
            TONAME=(*,PRINTER1),DESC='OS/2 printer 1'
   ```

   Effect: These two nickname statements enable output from GDDM applications to printers and plotters attached via GDDM-OS/2 Link.

   `(*,ADMPMOP)` directs output to the default, directly attached printer or plotter (that is, to the printer or plotter attached to the workstation).

   `(*,PRINTER1)` directs output to the printer or plotter (not the port name) defined via OS/2. “PRINTER1” is merely an example name for a printer. The DESC value “OS/2 printer 1” is the description of the printer that would be displayed to the user from the

User-Control output panel or from the ICU Printer Selection Panel.

8. ```
   ADMMNICK PROCOPT=((IMGINIT,BACKGND))
   ```

   Effect: Initializes the image field for displays and printers to the background color (black on a display device and white on a printer). By default, image fields are initialized to BLACK, which can result in the printing of large, unwanted areas of black on a printer.

9. ```
   ADMMNICK NAME=PSFPRINT,FAM=2,TONAME=PRINTER,
                     TOFAM=4,DEVTOK=A3820Q,

   PROCOPT=((CPSPOOL,CLASS,n,,DEST,destname,FORM,formname)),
                     DESC='The 3812 printer in Room S480'
   ```

   Effect: When an application (such as the GDDM-PGF ICU) requests family-2 output to destination PSFPRINT, the output is spooled automatically to a PSF/VM server for printing. The DESC= value is displayed when the user enters "?" from the ICU or from User Control to request a list of printers.

10. ```
    ADMMNICK NAME=,FAM=1,TONAME=PR7,TOFAM=2
    ```

    Effect: An attempt to send output to *any* family-1 device, regardless of the name of the device, results in the output being directed to the family-2 destination (an intermediate file) PR7.

11. ```
    ADMMNICK NAME=TO68,FAM=2,DEVTOK=L68,
                     DESC='Your local 3268 printer'
    ```

    Effect: Assigns the device token L68 to destination TO68 when it is used for family-2 output.

12. ```
    ADMMNICK NAME=3812LAND,FAM=2,DEVTOK=S3812Q,
                     PROCOPT=((IPDSROT,90),(IPDSCPI,12),(IPDSLPI,6))
    ```

    Effect: Alphanumeric characters are usually printed on IPDS printers (such as the IBM 4224 or 3816) at 10 characters per inch and 8 lines per inch, giving 85 columns and 88 rows on U.S Letter (U.S.Quarto) paper (8.5 inches by 11 inches). This nickname statement gives 132 columns and 51 rows, and produces output in landscape rather than the more usual portrait orientation.

13. ```
    ADMMNICK NAME=PLOT1,TONAME=(*,PLOTTER1),FAM=2,TOFAM=1

    ADMMNICK NAME=PLOT2,TONAME=(*,PLOTTER2),FAM=2,TOFAM=1
    ```

    Effect: If you are using a 3270 PC/G or /GX workstation, or a PS/2 with the Graphics Work Station Program (GWSP), you can have more than one plotter attached. These two statements allow two directly attached plotters to be differentiated and explicitly selected.

14. ```
    ADMMNICK FAM=1,DEVTOK=TEK4205M
    ```

    Effect: Allows the Tektronix 4205 ASCII terminal to be used with a mouse (assuming that the 3174 controller has been set up for an ASCII terminal with no mouse).

15. ```
    ADMMNICK FAM=1,PROCOPT=((GINKEY,1,24))
    ```

    Effect: Identifies PF24 as the graphics-input key on ASCII graphics displays.

16. ```
    ADMMNICK NAME=*,FAM=1,PROCOPT=((SPECDEV,IBM5080,ddname))
    ```
    ```

Effect: Enables an IBM5080 or 6090 graphics system to be used. The value *ddname* is the name you choose to describe the device. The name can be up to 8 characters, and must be specified on the command you issue before using the 5080 or 6090. Under CMS, the command is `FILEDEF ddname GRAF cuu`. Under TSO, the command is `ALLOC FILE (ddname) UNIT (cuu)`. In both cases, *cuu* is the address of the control unit for the display.

17.
```
ADMMNICK NAME=QPLOT,TOFAM=2,TONAME=luname,DEVTOK=L7372,
              PROCOPT=((STAGE2ID,PLOTTER))

ADMMNICK NAME=PLOTTER,FAM=1,TONAME=(*,ADMPLOT),
              PROCOPT=((PLTPENV,10))
```

    Effect: Causes the GDDM TSO print utility to send a file of output to a plotter. The STAGE2ID processing option specifies a name-list for the target device. Both nickname statements must be in effect at the time the file is created; the second of the two statements, which in this example also specifies the pen velocity for the plotter, must be in effect at the time the data is plotted. When the GDDM print utility is directing output to a plotter, any device token you specify must be a plotter device token, rather than a token for a terminal that has a plotter attached as an auxiliary device. In this example, `luname` is the name by which the device is known to VTAM.

18.
```
ADMMNICK NAME=AB?,FAM=2,PROCOPT=((PRINTDST,Q,=))
```

    Effect: Causes a family-2 file named “AB123” to be routed to a printer with luname “AB123.”

19.
```
ADMMNICK NAME=PRT?,FAM=2,PROCOPT=((PRINTDST,G,=))
```

    Effect:

    This nickname matches any device with a name beginning PRTxxx opened for family-2 output. The ‘=’ specified for the destname parameter of the PRINTDST processing option causes the name with which the device was opened to be used both for the name of the family-2 print file and for the LUNAME passed to JES.

    The greater the number of uses you have for the different output devices at your enterprise, the more nickname statements you need to place in the user-defaults module. Enterprises with many devices may find a large proportion of the user-defaults module taken up with nickname statements.

    To keep the size of your assembled module within the allowed limit (320 KB), consider coding nicknames that match several device names.

    Nickname statements that match all devices opened for a particular output family are one option but may be too general. By coding a leading or trailing ‘?’ as part of the *namelist*, the nickname can match several devices whose names begin with the same prefix or end with the same suffix.

End of General-use programming interface

Chapter 19. Updating GDDM default modules

This chapter provides instructions for replacing GDDM-supplied default modules that you have updated. The modules you can replace using these instructions are:

Module	Description	Page
ADMADFx	External-defaults module	197
ADMDATRN	Alphanumeric-defaults module	181
ADMDGAIT	ASCII-graphics-input translation table	85
ADMDGTRN	User-defined shading patterns translation tables	141
ADMDJCOL	Color-separation master tables	145
ADMDKFNT	AFPDS-to-IPDS conversion table	125
ADMLSYSA	ASCII device-tokens table	101
ADMLSYS1	3270 device-tokens table	101
ADMLSYS3	System printers device-tokens table	101
ADMLSYS4	Family-4 device-tokens table	101
ADM4CFID	DBCS code-page and font-conversion table	129
ADM4CPID	Code-page-identifier conversion table	127
ADM4FGID	Font-global-identifier conversion table	128
ADM4FONT	Font-emulation table	123
	Default symbol sets	131

Page references indicate where in this book you can find more information about the module.

GDDM supplies these default modules in both source (assembler or symbol set) and executable forms. If you are changing the source of one of the default modules, you should create a new copy on your own storage. If you update the supplied sample in the system version, it could be overwritten when GDDM is serviced or upgraded and any customized copy will be lost. Also, if you decide later to restore the original defaults, you will be able to recreate them from the system version of the source.

The source of the GDDM default modules is supplied as part of GDDM Base as follows:

GDDM/VM: Files of type ASSEMBLE
GDDM/MVS: As members of the SADMSAM data set
GDDM/VSE: As A-type members in the GDDM sublibrary

The source of the default symbol sets is supplied as part of GDDM Base as follows:

GDDM/VM: Files of type ADMSYMBL
GDDM/MVS: As members of the SADMSYM data set
GDDM/VSE: As Z-type members in the GDDM sublibrary

Updating GDDM default modules in the VM environment

1. Copy the GDDM source file to your own storage.
2. Edit the source file

   For the default modules, if you are not using an assembler that supports the AMODE and RMODE instructions, remove any such instructions from the file.

   For the default symbol sets, load the symbol set using the GDDM Image Symbol Editor or Vector Symbol Editor, as appropriate. For more information about editing the symbol sets, see Chapter 12, “The GDDM default symbol sets” on page 131.
3. Create the TEXT deck for your updated source.

   For the default modules:

   Assemble the file using the system assembler. You need to specify the following macro library:

   ```
   GLOBAL MACLIB ADMLIB
   ASSEMBLE ADMxxxxx
   ```

   This generates a text deck.

   For the default symbol sets, save the symbol set from the editor as a text deck. For the Image Symbol Editor, select option 3 from the home panel to generate an object deck. For the Vector Symbol Editor, type SAVE DECK on the command line and press ENTER. You must change the filetype of the output file from ADMDECK to TEXT.
4. Replace the text deck in the ADMGLIB TXTLIB library

   ```
   TXTLIB DEL ADMGLIB ADMxxxxx
   TXTLIB ADD ADMGLIB ADMxxxxx
   ```
5. If you have created a GDDM/VM saved segment, you must rebuild it in order to make the updated modules available. (For more information about rebuilding the saved segment, see “Shared segment” on page 248.)

If service is installed for any of the GDDM default modules, the service EXEC issues a warning and backs up the existing copy of the source and text versions with file types OASSEM and OTEXT. You must merge your changes with the new source and repeat steps 3 through 5.

Accessing ADMADFV, the GDDM/VM external defaults module

If you have created your own version of ADMADFV, the GDDM/VM external defaults module, you may want to test it before you make it available to users or before you recreate the GDDM saved segment. Alternatively, you may have several GDDM applications running on your system, with conflicting requirements for GDDM customization, which can only be satisfied by using separate copies of ADMADFV. The methods for accessing ADMADFV described here can be used for any of these reasons.

First create one or more suitable external defaults modules. If you are creating more than one, you may want to keep them in separate TXTLIBs, in order to be able to distinguish them. Remember that files of type TEXT are always accessed before the `GLOBAL TXTLIB` list. If you are using TXTLIBs, the appropriate library

must be added to the list of libraries required for your application and before ADMGLIB. For example,

```
GLOBAL TXTLIB appn-txtlib admadfv-txtlib ADMGLIB ADMHLIB
```

Method 1

Load ADMADFV with your application at execution time.

```
LOAD appn-name ADMADFV (RESET appn-entry-name START
```

Method 2

Create a load module that includes ADMADFV, using the stub module ADMUX0AV.

```
LOAD appn-name ADMUX0AV (load-options
GENMOD appn-name MODULE A (FROM appn-entry-name
```

Updating GDDM default modules in the MVS environment

All of the steps involved in replacing the default modules in the MVS environment can be carried out under TSO. Under CICS and IMS, however, only the symbol sets can be edited and saved. Therefore, you are recommended to use TSO for all of these tasks as it removes the need for you to set up the mechanism to generate symbol set object decks and you do not have to change subsystem to complete the task.

1. Copy the GDDM source member from the SADMSAM or SADMSYM data set to your own storage.
2. Edit the source file.

   For the default modules, if you are not using an assembler that supports the AMODE and RMODE instructions, remove any AMODE and RMODE instructions from the file.

   If you are editing the source of ADMADFC (the CICS external defaults module), delete the ADMMDFTX macros and the ADMMDFT DFTXTNA= statement from the file. (These entries are required in ADMADFC in the VSE environment only.)

   For the default symbol sets, ensure that you have set up the output file to save a symbol set object deck. For TSO, this is a sequential FB80 data set allocated to the ddname ADMDECK. Load the symbol set using the GDDM Image Symbol Editor or Vector Symbol Editor, as appropriate. For more information, see Chapter 12, “The GDDM default symbol sets” on page 131.
3. Create the text deck for your updated source.

   For the default modules, assemble the source. You must add your SADMSAM data set to the SYSLIB data set ddname.

   For the default symbol sets, save the symbol set from the editor as a text deck. For the Image Symbol Editor, select option 3 from the home panel to generate an object deck. For the Vector Symbol Editor, type SAVE DECK on the command line and press ENTER.
4. Link-edit the changed default module in the SADMMOD data set.

   Create an SMP/E usermod to link-edit your changes with the GDDM source. This ensures that you are warned if the default module is changed by installing GDDM service. SMP/E automatically links the default module into all of the load modules in which it resides. Base your usermod on the following example:

```
++USERMOD(MYUMOD1) .
++VER(Z038)  FMID(HGD3100) .
++VER(C150)  FMID(HGD3100) .
++VER(P115)  FMID(HGD3100) .
++MOD(ADMxxxxx)  DISTLIB(AADMMOD) LEPARM(RENT,REFR) .

      INCLUDE YOUR OBJECT DECK HERE
```

You can change the name of the usermod 'MYUMOD1' to any 7-character alphanumeric name beginning with a letter. You must change ADMxxxxx to match the name of the default module you are changing.

Receive and apply the usermod. Do not accept it. It is very important that you do not accept the usermod because you will not be able to RESTORE it if you need to install service for the default module at a later date.

If you later try to install service for the default module, the APPLY will fail with RC=12 and with message:

```
GIM31901I SYSMOD ptfname DOES NOT SPECIFY myumod1 ON THE PRE OR SUP OPERAND.
```

To install the service, first do an SMP/E RESTORE for your usermod and then APPLY the GDDM service. You must merge any changes that have been made to the source file with your customized version. When you have completed the updates, repeat steps 3 on page 215 and 4 on page 215, but change the version statements on your usermod as follows:

```
++VER(Z038)  FMID(HGD3100) SUP(ptfname) .
++VER(C150)  FMID(HGD3100) SUP(ptfname) .
++VER(P115)  FMID(HGD3100) SUP(ptfname) .
```

For more information about the SMP/E commands used here, see the *SMP/E Reference*, SC28-1107, and the *SMP/E User's Guide*, SC28-1302.

Accessing ADMADFC, ADMADFI, and ADMADFT, the GDDM/MVS external defaults modules

If you have created your own version of one of the GDDM/MVS external defaults modules, you may want to test it before you make it available to users. Alternatively, you may have several GDDM applications running on your system, with conflicting requirements for GDDM customization, which can only be satisfied by using separate copies of these modules. The methods for accessing the external defaults modules described here can be used for either reason.

First create one or more suitable external defaults modules. Keep them in private data sets in order to be able to distinguish them.

Method 1

Link-edit the external defaults module with your application, in the same place as you include the GDDM stub modules such as ADMASNT or ADMASNC. For example, if you were link-editing a nonreentrant TSO application module, your link-edit statements might look like this. This method overrides method 2.

```
INCLUDE (appn-dataset)       appn-name
INCLUDE (SADMMOD)            ADMASNT
INCLUDE (private-dataset)    ADMADFT
NAME appn-name (R)
```

Method 2

Create a single-CSECT load module for the external defaults module and place this higher in the data set access list than the GDDM SADMMOD data set.

Notes:

1. The access order is STEPLIB, followed by LPA, followed by LINKLIST.
2. If you are using TSO CALL to execute your application, the library on the CALL statement overrides the normal access list.

Updating GDDM default modules in the VSE environment

1. Copy the GDDM source file to your own storage.
2. Edit the source file.

   For the default modules, if you are not using an assembler that supports the AMODE and RMODE instructions, remove any such instructions from the file.

   For the default symbol sets, ensure that you have created the ADMD object deck output file, defined it to CICS, and included it in your CICS startup job stream. Load the symbol set using the GDDM Image Symbol Editor or Vector Symbol Editor, as appropriate. For more information, see Chapter 12, “The GDDM default symbol sets” on page 131.
3. Create the text deck for your updated source.

   For the default modules, assemble the source. You must add your GDDM sublibrary to the LIBDEF search chain.

   For the default symbol sets, save the symbol set from the editor as a text deck. For the Image Symbol Editor, select option 3 from the home panel to generate an object deck. For the Vector Symbol Editor, type SAVE DECK on the command line and press ENTER.
4. Link-edit the changed default module in your GDDM sublibrary. Copy your new default module to replace the OBJ type member supplied by GDDM. Relink-edit GDDM/VSE. Copy the link-edit statements from the Z-type member ADMLNKB in the GDDM sublibrary. You might notice the following error message in the output from the link-edit job:

   ```
   UNRESOLVED EXTERNAL REFERENCES   WXTRN  ADMxxxxx
   ```

   This is caused by the way in which the GDDM products are packaged and can be ignored.

If you later install service for the modules you have changed, your changes will be lost. You must merge any changes to the new GDDM source with your copy before repeating steps 3 and 4.

Accessing ADMADFC and ADMADFD, the GDDM/VSE external defaults modules

If you have created your own version of one of the GDDM/MVS external defaults modules, you may want to test it before you make it available to users. Alternatively, you may have several GDDM applications running on your system, with conflicting requirements for GDDM customization, which can only be satisfied by using separate copies of these modules. The methods for accessing the external defaults modules described here can be used for either reason.

First create one or more suitable external defaults modules. Keep them in private sublibraries in order to be able to distinguish them.

Method 1

Link-edit the external defaults module with your application in the same place as you include the GDDM stub modules such as ADMASNB or ADMASLC. Add your private sublibrary to the LIBDEF search chain ahead of the GDDM library. For example, if you were link-editing a nonreentrant CICS application module, your link-edit statements might look like this. This method overrides method 2.

```
INCLUDE appn-name
INCLUDE ADMASNB
INCLUDE ADMASLC
INCLUDE ADMADFC
PHASE appn-name,*
```

Method 2

Create a single-CSECT PHASE for the external defaults module and place this higher in the LIBDEF search chain or shared virtual area list than the GDDM sublibrary.

Part 4. GDDM performance and tuning

Chapter 20. Resource and capacity planning

This section tells you how to estimate how much of the various host computer system resources your GDDM users will require. Read it to find out how to calculate the amount of:

- Virtual storage required by GDDM
- Workstation storage required by GDDM
- DASD space required by GDDM
- DASD space required for user-created GDDM objects

Virtual storage required by GDDM

The GDDM virtual storage requirement comprises two elements:

- The GDDM code size
- The additional dynamic storage requirement for each GDDM user.

In the operating system environments that support 31-bit addressing, most of the GDDM code and dynamic storage can be above the 16MB line.

The GDDM code virtual storage requirement depends on which GDDM components are being used. Table 30 gives you an approximate idea of how much storage is needed when various GDDM components are running.

Table 30. GDDM storage estimates

GDDM component being run	**Storage**
GDDM Base: alphanumerics only	1200KB
GDDM Base: graphics	2000KB
GDDM Base: image	2300KB
GDDM Base: Image Symbol Editor	2100KB
GDDM Base: print utility	2100KB
GDDM Base: CDPU	2100KB
GDDM Base: run-time mapping	1200KB
GDDM Base: High Performance Alphanumerics	1200KB
GDDM-PGF: graphics	2200KB
GDDM-PGF: Interactive Chart Utility	2900KB
GDDM-PGF: Vector Symbol Editor	2400KB
GDDM-IVU	2500KB
GDDM-GKS	2300KB
GDDM-IMD	1550KB

The dynamic storage required by each user depends on which GDDM functions are being called, which device is being used, and the complexity of any picture being displayed. The normal range for this requirement is from about 20KB for a simple alphanumerics-only picture, through 260KB for a typical graphics picture, to about 2MB for a complex high-resolution image. In general, a contemporary

distributed-function terminal needs slightly less storage than an older non-programmable terminal. You must make an estimate of the amount of virtual storage needed, based on the likely usage you are going to make of GDDM.

Reducing the amount of virtual storage needed by GDDM

The amount of virtual storage needed for the GDDM code in a user address space can be reduced by:

- On MVS, putting GDDM into the pageable link pack area (LPA). This is described in "TSO tuning—things you can do" on page 247.
- Repackaging by:
  - On VM, putting GDDM into a saved segment. This is described in "Shared segment" on page 248.
  - On VSE, using the supplied packaging stubs to load only the required subset of GDDM code.

  Repackaging is described in Chapter 23, "Repackaging for performance" on page 251.

Workstation storage requirements for GDDM-PCLK

GDDM-PCLK runs in a DOS session provided by an IBM terminal emulator. For a list of suitable emulators, see "Setting up terminal emulators for GDDM-PCLK" on page 279.

The minimum storage size required in the workstation for this session depends partly on the configuration being used.

Typical storage requirements for various configurations are shown below.

Basic GDDM-PCLK system: 182KB.

Add for display

	CGA	MCGA	EGA (640x200)	EGA (640x350)	VGA	8514/A
For display:	+48KB	+76KB	+51KB	+60KB	+73KB	+14KB
Save/restore graphics:	+20KB	+40KB	+65KB	+114KB	+154KB	+0KB

Add for printing and plotting:

Immediate plot +16KB

Immediate print +80KB

In addition to the above storage requirements, more storage is needed in the personal computer system session for DOS, and the mouse driver or 8514/A, XGA, or Image Adapter/A display driver, if present.

You must also check that the personal computer system on which you intend to install GDDM-PCLK has a minimum of one 360KB diskette drive, double-sided.

If you are installing on a 360KB double-sided disk drive, you need to do the following:

1. Copy the display driver (either GQDXX10A.EXE or GQDXX06A.EXE onto another diskette and delete it from the PCLK11 subdirectory.
2. Before running GDDM-PCLK, preload the display driver by inserting the diskette (see above) on which the display driver was stored and type `GQDXX10A` or `GQDXX06A`, as appropriate.

GDDM-PCLK can now be installed. The display driver remains resident until the workstation is next rebooted. When GDDM-PCLK finds a display driver preloaded, it does not check that the driver exists in the PCLK11 subdirectory.

Workstation storage requirements for GDDM-OS/2 Link

GDDM-OS/2 Link runs on any personal computer system that supports OS/2 Extended Edition 1.2 or later, subject to storage availability. The following figures indicate the minimum amount of free space required:

- 375KB of hard-disk storage
- 270KB of RAM.

DASD space required by GDDM

The space requirements for GDDM and its components are given in the following tables. These figures include an additional 25 percent for service. Note that, on MVS, space for the SMP/E temporary data sets, equivalent to the space for the GDDM distribution data sets, is also required during installation.

These figures show the space required in DASD cylinders using 4KB blocks on VM and 1KB blocks on VSE, and fixed block architecture (FBA) 1KB storage blocks.

Be aware that these figures are for planning purposes only; for more accurate figures, see the appropriate *GDDM Program Directory*.

GDDM/VM can be installed in an SFS directory.

Table 31. GDDM/MVS space requirements

Data sets	3380 cylinders	3390 cylinders
SMP/E		
SMP/E data sets (if new SMP/E environment)	60	54
Target data sets		
GDDM/MVS Base	35	31
GDDM/MVS NLS per language	+3	+3
GDDM/MVS NLS Japanese (see note)	+6	+6
GDDM/MVS-PGF	+4	+3
GDDM/MVS-IVU	+2	+2
GDDM/MVS-GKS	+2	+1
GDDM/MVS-IMD	+4	+4
Distribution data sets		
GDDM/MVS Base	40	36
GDDM/MVS NLS per language	+4	+3
GDDM/MVS NLS Japanese (see note)	+6	+6
GDDM/MVS-PGF	+6	+5
GDDM/MVS-IVU	+3	+2
GDDM/MVS-GKS	+4	+3
GDDM/MVS-IMD	+5	+5
Note: GDDM/MVS NLS Japanese now includes the Kanji symbol sets.		

Table 32. GDDM/VM space requirements

GDDM component	IBM 3350	IBM 3375	IBM 3380	IBM 3390	FBA storage blocks
GDDM/VM Base	36	45	29	24	15000
GDDM/VM Base NLS	+3.4	+4.0	+3.0	+2.3	+1500
GDDM/VM Base NLS Japanese (see note)	+9.0	+12.0	+7.0	+6.0	+4100
GDDM-PGF	+5.0	+6.0	+4.0	+3.0	+1900
GDDM-IVU	+1.3	+1.6	+1.3	+1.0	+550
GDDM-GKS	+2.2	+2.7	+1.8	+1.5	+900
GDDM-IMD	+6.0	+8.0	+5.0	+4.0	+2500
Note: GDDM/VM NLS Japanese now includes the Kanji symbol sets.					

Table 33. GDDM/VSE space requirements

GDDM component	IBM 3350	IBM 3375	IBM 3380	IBM 3390	FBA storage blocks
GDDM/VSE Base	51	76	49	46	23000
GDDM/VSE Base NLS per language	+5	+8	+5	+5	+2250
GDDM/VSE Base NLS Japanese (see note)	+15	+23	+15	+14	+7000
GDDM-PGF	+7	+11	+7	+7	+3200
GDDM-IVU	+3	+4	+3	+3	+1250
GDDM-IMD	+6	+9	+6	+6	+2900

Note: GDDM/VSE NLS Japanese now includes the Kanji symbol sets.

DASD space required for user-created GDDM objects

GDDM users can produce several objects (such as saved charts) from application programs that use GDDM or from the GDDM interactive utilities. These objects require direct-access storage over and above that required by GDDM itself.

Table 34 on page 226 shows the objects and their contents, the features of GDDM with which they are associated, how many you may expect your users to produce, and their likely size. Be aware that the size and number of these factors can vary enormously according to the use made of GDDM, and are therefore necessarily approximate. However, they should give you a value that you can use for estimating.

Table 34. Contents and sizes of user-created GDDM objects

Object type (default name)	Contents	Produced by	How many	Typical size of records for each object (bytes)
Page printer image files (ADMIMAGE)	GDDM pictures held in device-dependent data stream format	GDDM Base programs	Not many: 0.5 per user	500×137 (IBM 3800/3820) 100×2000 (IBM 4250)
Saved pictures (ADMSAVE)	GDDM pictures held in device-dependent data stream format	GDDM Base programs	Not many: 0.5 per user	40×400 (variable)
Image data (ADMIMG)	Image information	GDDM Base programs	Many: 10 per user	(150—350)×400 for documents
Image projections (ADMPROJ)	Projection information	GDDM Base programs	Not many: 0.5 per user	(2—5)×400
Saved GDF files (ADMGDF)	GDDM pictures held in device-independent Graphic Data Format	GDDM Base programs and ICU	Many: 10 per user	20×400
Symbol sets: image	Dot-pattern symbols for logos and special characters (see note 1)	GDDM Base Image Symbol Editor	Not many: 0.5 per user	10×400 (variable)
Chart data (ADMCDATA)	Data values for ICU Charts	GDDM-PGF ICU	Many: 20 per user	4×400
Chart definitions (ADMCDEF)	Data extraction rules	GDDM-PGF ICU	Many: 5 per user	3×400
Chart formats (ADMCFORM)	Formats for ICU charts	GDDM-PGF ICU	Many: 15 per user	4×400
Symbol sets: vector	Line symbols for logos and special characters (see note 1)	GDDM-PGF Vector Symbol Editor	Not many: 0.5 per user	20×400 (variable)
GKS metafiles (ADMGKSM)	Metafile input/output	GDDM-GKS programs	8 per user	500×400
Maps: generated ADS (COPY)	Data structures that correspond to maps	GDDM-IMD	Not many: 5 per mapping program	50×80
Maps: generated mapgroups (ADMGGMAP)	Maps in usable form for execution	GDDM-IMD	Not many: 1+ per mapping program	4×400
Maps: MSLs (ADMMSL)	Library of maps in a form for editing	GDDM-IMD	Not many: 1 per IMD user +2 or 3 common. Many maps per MSL	8×256 per map

Notes:

1. This applies to user-defined symbol sets only.
2. The last column indicates the number of records multiplied by the record size.

Chapter 21. Understanding GDDM performance

This chapter explains how GDDM works, and what the performance implications are. It gives you enough background to help you make informed judgments about the subsystem-dependent customizing suggestions discussed in Chapter 22, “Tuning and customizing by subsystem” on page 235.

GDDM application-program design is not discussed in this chapter, but it has the potential to be the biggest factor in causing poor performance. For information about how GDDM application-program design affects performance, see the *GDDM Base Application Programming Guide*.

What governs GDDM resource usage

Three major factors influence the amount of system resources that GDDM requires to display a picture on a graphics device or an image device:

- The type of device

  Some devices are capable of a high level of processing so GDDM can offload some of the picture building to the device, while others need the complete picture to be built by GDDM. Similarly, the device resolution plays a part in the resource required to build the data stream.

- Picture complexity

  For graphics, picture complexity is related to the number of elements (lines, arcs, areas, and so on) that define the picture. For images, complexity is related to the total number of pixels in the picture, how many are on or off, and how often they alternate between the two states.

  Complexity is also related to the device on which the picture is displayed. Graphics pictures that appear fairly modest to GDDM when it is producing them for a display device may well appear very complex when the device is a plotter.

- Tuning

  There are often tuning parameters that can be changed to make GDDM more efficient or which can trade different resources within the system as a whole or within GDDM. This is discussed in Chapter 22, “Tuning and customizing by subsystem” on page 235.

This chapter discusses how GDDM works and how this is related to the devices which GDDM supports.

How GDDM draws pictures

GDDM is made up of several layers of function. Some of the layers use other layers. The Interactive Chart Utility, for example, uses the Presentation Graphics Routines (known as the PG Routines). If you draw a picture using the Interactive Chart Utility, the panel options you select are translated into GDDM PG Routine calls, like CHPIE to produce a pie chart. The PG Routine calls themselves get converted into GDDM Base calls, such as GSLINE to draw a line, or GSARC to produce an arc.

The GDDM Base calls are converted into the device-dependent data stream necessary to display the picture. For devices such as programmable workstations, little work is necessary to do this. Image devices on the other hand, require a significant amount of effort from GDDM.

GDDM operates on several subsystems, and the way it produces graphics is the same in all of them. Therefore, most of the GDDM code is independent of the subsystem that it runs under. All requests for subsystem services (such as device reads), are channeled through an environmental layer that is different for each subsystem.

What happens in a GDDM program

Understanding the sequence of events in a GDDM program helps to identify some of the areas where performance problems can occur.

- The first GDDM call is usually FSINIT. The primary purpose of this call is to establish the environment in which GDDM will operate. It starts GDDM processing by:
  - Loading the GDDM code and resolving routine addresses
  - Creating work areas and initializing GDDM default values
- The next calls concern the characteristics of the device on which you are going to draw the picture. These tell GDDM things like:
  - Which device type you are going to use
  - The device characteristics available
  - Which part of the display area you are going to use to draw on
  - What coordinate system you are going to use.

  These calls generally start with a DSOPEN call to open the device to GDDM. GDDM nickname processing is done at this stage. If GDDM has access to the device and the device is queriable, a 3270 Read Partition (Query) is generated to ask the device what its characteristics are.

  The DSOPEN can tell GDDM things that have a bearing on performance, such as what mode the output device is working in.
- Next follows a series of calls that build the picture.

  For graphics or alphanumeric pictures these calls define the picture as a series of primitives such as lines, arcs, or areas. These elements are kept in a table in main storage in *graphics data format (GDF)*. There is a choice for using floating-point GDF or fixed-point GDF as the method for storing pictures and this can have a significant influence on performance. This is discussed in the *GDDM Base Application Programming Guide*.

  GDDM cannot create images via its own application calls, but images previously output from GDDM or created by other products may be loaded into GDDM. If the output image device can handle the image directly, it is kept in the same format as it is generated or previously stored. Where the device does not support direct transmission, the images are held in a GDDM-internal format in dynamic storage.

  Note that these calls do not cause the picture to be sent to the device.
- When the picture is complete, a call is issued to send it the the device. Examples of GDDM output calls are: DSFRCE, FSFRCE, ASREAD, GSREAD, MSREAD, or WSIO.

These calls cause GDDM to convert the internal representation of the picture into a device dependent data stream and to send it to the device. This is where device type and picture complexity have a strong influence on the time taken to do this conversion.

- The final call in a GDDM program is FSTERM. This clears the GDDM environment by releasing all the storage that GDDM has acquired for the application and deleting all the loaded modules.

GDDM hardware – how it works

A device can work in one of two ways to produce graphics or image. These are known as the *raster-scan* and *random-scan* technologies.

For raster-scan devices, the picture area is defined as a series of regularly spaced dots. The drawing implement (like an electron beam or the print head of a matrix printer) moves along each row, marking those dots that form the picture.

For random-scan devices (plotters) the path of the pen that actually draws the picture moves along the lines that make up the picture.

All the devices that GDDM supports rely on one of those two basic technologies to produce graphics or images. These technologies in themselves have implications for GDDM performance, but there are other device characteristics, the prime one being device programmability, that play an even larger role.

GDDM supports six classes of devices:

- Order-driven devices – programmable
- Order-driven devices – nonprogrammable with random scan
- Order-driven devices – nonprogrammable with raster scan
- Image devices
- Character-driven devices
- Page printers

In general, devices in the classes at the top of this list have the highest internal processing capability, requiring the shortest data stream and hence the least work by GDDM to display a picture.

Of the six classes, only the nonprogrammable, order-driven devices use the random-scan technique to produce pictures. All the others use raster-scan technology.

Order-driven devices

All GDDM graphics calls that a program issues eventually get translated to commands like “draw a line from A to B” or “future items will be colored blue.” The order-driven devices obey just such orders. All GDDM has to do is encode the program calls into the language that the device recognizes and to send them.

At the device, the orders are decoded and the drawing is made.

Programmable devices

Workstations such as an IBM PS/2, IBM PS/55, and IBM 5550 running a host emulator are examples of a programmable order-driven device. One of the key features of these devices is that the work of activating the dots that represent the picture is done in the terminal itself. That process is called *vector-to-raster conversion*. Besides converting lines or vectors to dots, the devices can also do other things like fill areas.

GDDM has two possible modes of operation for communicating with workstations. These modes are known as *retained* and *nonretained* modes. In some cases, the emulator allows the user to choose the mode used; in others, it is fixed.

In retained mode, the vector definition of the picture is written to *segment storage* as the terminal starts the vector-to-raster process onto an *all-points-addressable* (APA) bit map of the screen. In nonretained mode, each vector is drawn on the bit map as it is received and then discarded.

You can see that in nonretained mode it is possible to display very complex pictures, regardless of the number of vector definitions that they contain. However, there are some other important performance advantages of retained mode which must be considered: (Some older workstations support nonretained mode only.)

- Segment manipulation

  The workstations can perform certain actions on pieces of a picture. These pieces of the picture are properly known as *segments* and are defined by the graphics application.

  Workstation functions performed on segments include:

  – Dragging

    The ability to move a segment around the screen under graphics cursor control. Its new position can be signalled back to the host.

  – Picture update

    Specific segments can be deleted by sending a “delete segment” order to the workstation. When adding a segment to a picture, only the new segment has to be sent, not the entire picture. Similarly, highlighting and visibility changes can be made by sending a small amount of data.

- Character expansion

  The workstation contains a set of graphics-character definitions, more properly known as a *symbol set* or *font*. When a picture is sent from the host, it contains references to characters in the symbol set. The workstation takes the required character definitions from the set, performs any scale, shear, or rotation necessary, and adds them to the picture definition. GDDM can download additional symbol sets to the workstation when working in retained mode which means that they can be processed directly by the workstation.

- Panning and zooming

  Under user control, the workstations can locally scroll pictures up and down or left and right (panning), or enlarge pictures (zooming). (For other graphics displays, and for the workstations if the local capability is not activated, GDDM performs the panning and zooming in the host.) This avoids the necessity of sending a new picture for every new viewing operation.

- Clipping

  If the picture is larger than the graphics field of the display, the picture needs to be clipped to fit the graphics field. Workstations running in retained mode do the clipping themselves so if the graphics field size is changed or the window is partially overlaid, the picture does not have to be resent.

Nonprogrammable devices

In many ways, devices such as the IBM 3179-G, 3192-G, and 3472-G are equivalent to a workstation running in non-retained mode. GDDM sends individual vector commands to the device and the device performs the vector to raster conversion to determine which pixels to activate to show the picture.

The device, however, has only one default vector-symbol set and has only four image symbol sets, so symbols have to be rastered out by GDDM. It also has no local clipping capability, so GDDM may have to do extra clipping in the host (for example, when a page is scrolled or an operator window is sized). GDDM tries to minimize the amount of the picture that needs to be updated and will only redraw a small section of the screen if this is possible.

Dot-matrix printers such as the IBM 4224 and IBM 4234 are also in the nonprogrammable device category. The vector-to-raster conversion process is performed in the printer, and so a 4224 printer generally requires less host-processor resource than, for example, a 3287 printer.

Because the paper moves through the printer in only one direction, all the GDDM data that describes the picture must be sent to the printer before the rastering process can begin. The 4224 has no resident vector symbol sets, so all graphics text referencing such symbol sets must be expanded into its constituent lines and arcs by GDDM. GDDM does not send arcs to the device. They are split into short vectors, resulting in more host processing and longer data streams.

Plotters such as the IBM 737x and 618x are also in the nonprogrammable, order-driven class.

They are similar to the programmable devices in that they can translate orders into pictures. However, the set of orders that they can draw is limited:

- Area fill is done by sending vector definitions of each of the lines that make up the shading pattern.
- Arcs are drawn as several very short vectors. Each vector is drawn by moving the pen by no less than half a pen width.
- Symbol sets are expanded into vectors in the host.

Because anything drawn on a plotter stays drawn, overlaid segments need to be processed by GDDM before the drawing orders are sent to the plotter so that hidden parts of objects are not drawn. This process is referred to in GDDM as *reverse clipping*, and applies only to GDDM support of these plotters.

Image devices

This section describes devices which support the GDDM ADMIMG image format known as IM image.

The IBM 3193 display station and the 3117 and 3118 scanners were designed specifically for this purpose, but most graphics devices can display these images without any additional action on the part of the user because of GDDM's emulation facilities.

An image is an array of dots called pixels. GDDM only supports bilevel images so each pixel can only be on or off. When these images are uncompressed, each pixel is represented by one bit. When they are compressed, the image may be reduced by as much as a factor of 20.

Devices such as the 3193 directly support the compressed image format, so the data stream is usually much shorter, leading to a reduced time to display the image. If the device only supports this format by emulation, the image must be built first and then rastered completely in the host, which is always a lengthy process.

Character devices

Some older devices (such as IBM 3278, 3279, and 3290 displays, and 3268 and 3287 printers) use dots to make up the picture, but split the screen into character-size cells. Each of these cells must be defined by GDDM unless the character to be displayed corresponds to one of the alphanumeric characters defined in the device hardware. When all of the cells have been defined, GDDM places them at the appropriate cell positions on the screen to build the picture.

All the work to produce the dot patterns to make up the cell definitions is done by GDDM in the host so these devices tend to be slower than their more modern counterparts.

Page printers

These are image type devices used to print PSEGs and LIST38xx files, such as the 3800 family and the 3812, 3816, 3112, 3116, 3912, 3916, 4028, and 4224 printers.

These AFPDS printers have two important characteristics:

- GDDM does not usually communicate directly with these devices. It creates a print file in disk storage that is later processed by another program, such as Print Services Facility (PSF), to produce the printed output.
- They have a high resolution, so millions of pixels may be required in order to display a picture. For example, at 240 pixels to an inch, a U.S. standard 8.5 x 11.0 inches piece of paper, or a European standard 210 x 297 millimeters A4 piece of paper (8.3 x 11.7 inches), both require more than 50 million pixels for a full size image. Looking after that number of pixels is costly in resources.

GDDM's support of these devices is very similar to that for character devices, but with special considerations because of the high resolution. GDDM rasters the vectors onto an APA bit map, which is broken up into logical cells of 32 × 32 pixels. Cells that are empty can be omitted from the file, which can speed up displaying the picture when there are blank areas.

Later devices, such as the 3112, 3116, 3912, 3916, 4028, and some models of other IBM printers, support the GOCA and IOCA formats. This means they are capable of a higher degree of processing in the device and therefore can work with a significantly reduced data stream. To take advantage of this performance improvement, it is essential that you choose the correct GDDM device token for your device when setting up your GDDM nickname user-default specifications. The device tokens are defined in Appendix C, “Device tokens supplied by GDDM” on page 367. Refer to Chapter 18, “Nickname user-default specifications” on page 205 for more information about setting up GDDM nickname user-defaults.

Chapter 22. Tuning and customizing by subsystem

This chapter describes what you can do to tune GDDM. Chapter 21 described why you should do it. Scanning Chapter 21 should help you make sensible decisions about the suggestions that follow. To use this chapter for tuning, you should look at the hints and tips for the particular subsystem or subsystems that you are interested in.

Hints and tips on efficient usage for all subsystems

Details of application-programming techniques are documented in the *GDDM Base Application Programming Guide*. The following is a list of hints and tips that you might want to think about:

- Customize your 3270-PC/G and /GX workstations with sufficient segment storage for GDDM to run in retained mode. Even for output-only type applications, this may give significant performance advantages. Retained mode usually means shorter data streams and, therefore, less use of the terminal access methods like VTAM.
- Very complex pictures may take quite some time to draw in the workstation. This can cause problems. While drawing, the work station does not communicate with the host operating system, and to the host it may seem that the terminal has gone out of service, and so it disconnects it.

  You can avoid this happening by resetting “timeout” values in your operating system. The “Things You Can Do” section for each subsystem later in this chapter tells you how to go about it.
- There are various GDDM default values that can affect performance. Here is a summary:

Default	***Meaning***
AM3270	Indicates how your 3270 terminals are attached. You can give GDDM a (small) helping hand by letting it know that you only have local SNA devices for example.
ICUFMDF	Determines the ICU defaults.
ICUFMSS	Determines the ICU symbol sets.
IMSUMAX	(IMS only) Specifies the maximum number of concurrent conversations to be supported.
IOBFSZ	Determines the GDDM transmission buffer size.
IOCOMPR	Indicates whether GDDM is to create compressed PS loads for character devices.
IOSYNCH	Indicates whether GDDM is to perform synchronized terminal I/O.
MAPGSTG	Defines the mapgroup storage threshold for GDDM run-time alphanumeric mapping.
SAVBFSZ	Defines the FSSAVE transmission buffer size for saved pictures.

- Processing-option values can also be specified using GDDM defaults or in a DSOPEN parameter list. Those that can influence performance include:

Procopt	***Meaning***
FASTUPD	Indicates the picture update mode to be used for 3270-PC/G and /GX, 3179-G, and 5550 family displays. You can shorten data streams by specifying (FASTUPD,YES).
HRISPILL	Defines whether a spill file will be used to hold expanded GDF for composed-page print file generation. You can trade off host-processor time against storage by specifying (HRISPILL,YES).
HRISWATH	Defines the number of slices (swathes) that the picture is divided into for composed-page print file generation. You can again trade off host-processor time against virtual storage. The larger the number of swathes, the less virtual storage is needed. For 3800 or 3820 print file generation, the largest useful swathe count is the picture depth in inches multiplied by 8. For 4250 print file generation, the largest useful swathe count is the picture depth in inches multiplied by 19.
SEGSTORE	Defines whether GDDM operates in retained or nonretained mode for the 3270-PC/G and /GX. Local interactions are only possible if (SEGSTORE,YES) is specified or implied by default.
CTLMODE	Defines whether user control is available to the user. The default is (CTLMODE,YES). If user control is not to be used, specifying (CTLMODE,NO) causes a small decrease in dynamic storage requirement.
LCLMODE	Defines whether GDDM is to use a higher precision in the vectors that it sends to the 3270-PC/G and /GX workstations. Specify only if panning/zooming is to be used in user control.

  Suggested settings for some of these are given in the “Things you can do” section for each subsystem later in this chapter.

  A more detailed explanation of each default is provided in Appendix A, “External defaults” on page 307.

CICS tuning—background information

CICS tuning is discussed here from the point of view of loading and packaging.

Loading

CICS operates in a fundamentally different way from that of most of the other subsystems that GDDM runs under. All users run their programs in the same address space/partition. If these programs are reentrant, like GDDM, the same copy can be used by other transactions on MVS. Putting GDDM in the link-pack area (LPA) gives no advantage, unless you have more than one CICS region running GDDM transactions, or are sharing the code with TSO or IMS users.

Program load under CICS can occur at two different times:

- At system initialization, if RES=YES in the PPT entry or RESIDENT(YES) in the CSD file
- At program run-time otherwise

Resident programs stay permanently in the CICS address space/partition. Most programs that are nonresident are not deleted from the CICS address space/partition until a short-on-storage condition arises. Frequently used programs are therefore likely to be in main storage if they have been used previously, even if they are nonresident.

If a nonresident program has been deleted from main storage, it has to be reloaded from DASD before it can be run. This may increase transaction-response time. Declaring GDDM programs as resident avoids this but causes short-on-storage conditions to occur more often.

Packaging

In some CICS environments, there are restrictions governing packaging GDDM modules together to form large composite CICS programs:

- Some levels of CICS have restrictions on the size of application program modules. Some of the composite GDDM modules that you can create may exceed this.
- At load time, sufficient real storage to contain the program has to be acquired. This may give paging problems where large programs are loaded in storage-constrained environments.

CICS keeps programs that are frequently used in main storage, even if they are nonresident, so packaging is unlikely to reduce loading delays.

Controlling the data stream

The I/O synchronization option (the IOSYNCH external default) determines whether the WAIT option is put on the EXEC CICS SEND calls that GDDM issues to send data streams to the device.

This is useful for controlling teleprocessor-attached terminals in a non-SNA environment. It prevents a GDDM application from issuing a SEND until the response from a previous SEND has been returned from the access method to CICS. This technique helps to avoid increasing the response times for non-GDDM users on the same teleprocessing line in the non-SNA environment.

In a SNA environment, PACING and VPACING should be used to avoid increasing the response times for non-GDDM users on the same control unit.

The size of the GDDM SEND transmission is determined by specifying the IOBFSZ external default.

The size of each transmission from CICS to the terminal access method is governed by the BUFFER parameter in the CICS TCT. For SNA, the maximum size is 1536 bytes, and so if GDDM SENDs a message that is larger than this, segmentation occurs. For non-SNA, the BUFFER parameter is automatically set to the size of the GDDM SEND, so messages are not segmented.

id=contgdm.Controlling GDDM in the processor

GDDM periodically gives back control to CICS to allow any transaction with a higher priority to be dispatched. For GDDM transactions, you should specify low TRNPRTY values in the DFHPCT entries or PRIORITY values in the CSD file entries; for non-GDDM transactions, specify high ones. This prevents GDDM from increasing the response times of non-GDDM transactions. There might well be an adverse effect on the GDDM response times, however.

CICS tuning—things you can do

Here are some suggestions and comments about some of the things you can do to tune CICS.

- Declare GDDM programs as nonresident in the PPT or CSD file.
- There is little advantage to be gained in packaging GDDM in most CICS environments.
- There is no advantage in putting GDDM in the VSE shared virtual area or in a link-pack area (LPA) unless you have more than one CICS region running GDDM transactions, or, on MVS, you are sharing GDDM between subsystems.
- If you do package GDDM, remember that composite GDDM programs must not exceed the maximum size for CICS application programs.
- There is no advantage in putting GDDM in the linklist on MVS.
- Configure the 3274 to support data-stream decompression if it is TP attached and the line speed is 19 200 bits per second or less.
- The size of the GDDM transmission buffer should be increased to 4KB for all local and SNA remote users by specifying the IOBFSZ external default.
- For remote non-SNA GDDM users, consider specifying the GDDM transmission buffer size (the IOBFSZ external default). Do not make it larger than the normal value, which is 1536. For remote SNA users, changing the BUFFER parameter in the CICS TCT, or the SENDSIZE if you are using RDO, has the same effect.

  A large size means:

  – The response times of other remotely attached users becomes degraded.
  – The response times for remotely attached GDDM users is better.
  – Fewer calls to the terminal access method. Therefore, there is lower processor utilization.

  A small size means:

  – Less impact on response times of other remotely attached users.
  – Worse response times for remotely attached GDDM users.
  – More calls to the terminal access method. Therefore, there is higher processor utilization.
- Control data streams in remote networks by using:
  – EXEC CICS SEND WAIT for non-SNA, specified by the IOSYNCH=YES external default.
  – PACING/VPACING and PASLIM for SNA.

- Prevent GDDM from impacting non-GDDM transactions in the processor by use of low TRNPRTY values in DFHPCT or PRIORITY in the CSD file.
- If your graphics terminals are attached to a 3274-1D controller:
  - For CICS/MVS: the Prepare To Read feature should be specified in the System Generation I/O device macro for each terminal.
  - For CICS/VSE: the mode parameter should be set to “05” in each terminal ADD statement in the IPL procedure.

IMS tuning—background information

The way that GDDM behaves is slightly different in an IMS environment from the way it behaves under other subsystems. The principal differences are:

- GDDM accesses a user data base, the SYSDEF data base, when it needs to determine the characteristics of devices used by an application. This is done because GDDM is not in direct communication with the display device and cannot, unlike other subsystem environments, issue a “query device.” The information on the SYSDEF data base is referenced by using the logical terminal (LTERM) name of the device.
- GDDM sends the data stream for the picture it has created by doing inserts (ISRTs) to the IMS message queue. Under other subsystems, each of the GDDM output requests is treated as a separate message and is sent to the device as soon as possible by the subsystem. Under IMS, they are treated as segments of a single message which is not sent to the device until GDDM has passed the complete data stream to IMS and issued a PURGE. This difference has important performance implications.
- Under IMS, application programs cannot use the GDDM Device Input call ASREAD to read input from the display device. This means that GDDM or GDDM Presentation Graphics Facility (PGF) application programs cannot be interactive.

  Under IMS, execution of a message processing program (MPP) is not bound to a specific LTERM. When an MPP associated with a specific transaction is running, it must (if it is conversational) accept input from any terminal that can input the transaction. IMS transactions can be written to be conversational by using a scratch pad area (SPA) to preserve the status of a conversational transaction for one LTERM. However, the nature of the architecture of GDDM means that the status of a conversation is reflected in the contents of many tables and control blocks throughout the Message Processing Region (MPR), and this cannot be simply put into the SPA. Therefore, an application can use GDDM only in nonconversational mode. The interactive utilities are implemented under IMS in a way that avoids this restriction. The mechanism is described later.

Output of GDDM data streams

The way that GDDM and IMS interact during the transmission of GDDM-produced data streams has important performance implications. There are three separate operations involved:

- GDDM sends the data stream to the message queue

  GDDM ISRTs the data stream as segments of a multisegment message to the output message queue. These points should be noted:

  – The size of the ISRT is controlled by the GDDM transmission buffer size (specified by the IOBFSZ external default).

  – With remote non-SNA transmission, the 3274 Model C has a restriction that no more than 3KB of uncompressed character definitions may be received in one transmission. Only GDDM knows where the data stream may be segmented to avoid this restriction, so it puts no more than 3KB of character definitions into a transmission buffer before an ISRT is made. For remote non-SNA, IMS sends each ISRT as a separate transmission.

    It should be noted that only BTAM may be used for remote non-SNA.

  – It is advantageous to increase the GDDM transmission buffer size, to 4KB by specifying the IOBFSZ external default. From the preceding information it can be seen that this change is not used to full advantage in the remote non-SNA environment.

  – Some tuning of the IOBFSZ external default to the long message queue length should be considered. This is because a message-queue element cannot be shared between segments of different messages. It is possible for a GDDM ISRT to span more than one message-queue element. To avoid wasting space in the message queue, make the GDDM transmission buffer size (the IOBFSZ external default) equal to, or a multiple of, the long message-queue logical-record length.

    The arithmetic for this must allow the space for the 4-byte “LLZZ” added by GDDM, and the space for appropriate headers in the message queue. See the description of the RECLNG parameter in the MSGQUEUE macro in the *IMS Installation Guide*.

- IMS moves the message to buffer(s) in Communications I/O Pool

  The GDDM output data stream is held in the message queues until it is completed, when GDDM issues a PURGE call. It is then considered sendable, and a device-dependent module is scheduled to convert the data stream into a device-specific form in the Communications I/O Pool (CIOP) OUTBUF buffers. For GDDM, Message Format Services (MFS) is bypassed. The transfer to the CIOP occurs when the device is available. These points should be noted:

  – Except with BTAM remote attachment, the blocking of the output data stream sent by IMS is governed by the OUTBUF buffer size. The choice of the OUTBUF size is severely constrained by the protocol of attachment of the output device.

  – For local non-SNA there can be only one OUTBUF buffer. This must be large enough to hold an entire output message, but must not be more than 32KB long. This means that there are GDDM graphic pictures that cannot be shown in the local non-SNA environment.

Unless there is some strong reason against, it is recommended that the maximum value (32KB) is used in the non-SNA environment. Violation of the maximum value causes these messages to be displayed on the IMS master console:

```
DFS0089I  OUTPUT EXCEEDS BUFFER SIZE

          LTERM xxxxxxxx NODE yyyyyyyy

DFS9989I  VTAM NODE yyyyyyyy IS INOPERABLE
```

and causes the IMS user terminal to be made inoperable. (Descriptions for the above messages are in the *IMS Messages and Codes Reference Manual*.)

If this occurs, and the device token being used specifies COMPRES=NO, a way of reducing the length of the data stream is to use a different device token, which allows data stream compression (assuming the controller is configured for PS compression), that is, COMPRES=YES. See Chapter 10, “Creating your own device tokens” on page 101 for a description of the COMPRES parameter.

– For BTAM remote attachment of GDDM devices, the value of OUTBUF is not used. In this case, IMS uses the ISRT sizes as block sizes to avoid the 3KB restriction for the 3274 Model C described earlier.

- IMS uses the access method to send the message to the device.

  This is a regular function of IMS. All that is unusual about it with GDDM is the occurrence of large data streams. You should consider the following:

  – With local non-SNA support, the entire GDDM data stream is put into one IMS OUTBUF buffer in the CIOP. When this is sent, it is done with one call to the access method (for example, one VTAM SEND). This can cause delays for other users of the channel if large data streams are sent. The channel is held busy while the control unit processes the entire GDDM data stream.

  – With BTAM remote support, IMS does not communicate with any other devices on a BTAM line group while a complete GDDM picture is being sent. In scheduling the device-dependent module, IMS uses the concept of a communications ITASK, in controlling output to different devices. IMS considers a complete BTAM line group as a resource for the communication task scheduling. This means that when a GDDM picture is being sent to one device of a line group, the other devices are locked out during the transmission of the entire picture.

    As an example, for an average 6KB (compressed) GDDM graphic data stream, sent on a 4 800 bits per second line, the lockout would last approximately 10 seconds. For a complex 17KB (compressed) data stream it would last approximately 30 seconds. If data-stream compression is not used, these times double.

    Note: This seriously affects the performance of all terminals on the same BTAM linegroup.

Cross-domain considerations

The IMS parameter OUTBUF defines the largest request unit (RU) that can flow from IMS into the network. For IMS-GDDM local non-SNA terminals, a value of 32KB is recommended for OUTBUF.

In a cross-domain environment, the RU is sent through a 3705. The MAXDATA parameter on the network control program (NCP) must be big enough to handle the largest RU that can enter the network; in this case it is 32KB.

The product of two other NCP parameters (MAXBFRU * UNITSZ), should be greater than MAXDATA. When MAXDATA = 32 000, for example on local non-SNA terminals, this leads to increased storage requirements in the host computer.

Use of nonrecoverable transactions

If GDDM transactions are inquiry-only, it is an advantage to make them nonrecoverable. This is done by specifying

```
INQ=(YES,NORECOVER)
```

as a parameter to the TRANSACT macro. If this is done, performance is improved, because:

- The output data stream is not logged.
- In SNA transmission, the message-queue elements and CIOP buffers do not have to be held until the picture display at the device is completed. This can be important, particularly for remote attachment.

Message Processing Region (MPR) priority

Care must be taken with the class scheduling and resource management to ensure that GDDM applications are run in Message Processing Regions (MPRs) of a suitable priority. For example:

- In a processor-constrained system, it would be advisable to run GDDM applications in a low-priority MPR to avoid impacting the response times of other transactions.
- In a storage-constrained system, it may be better to run GDDM applications in medium- or high-priority MPRs to minimize the time for which storage is required.

Message queue and Communications I/O Pool sizes

GDDM data streams are large. This increases the message queue and the CIOP space requirements. These points should be considered:

- The message queues become the bottleneck if there are delays in outputting messages to terminals, because the CIOP buffers are allocated only when the communication path to the device is available.
- It may be prudent to use the terminals in RESPONSE mode to help regulate the flow of output.
- With local non-SNA terminals, the use of large 32KB OUTBUFs for GDDM output affects the CIOP utilization. However, non-GDDM transactions that use MFS do not cause the large OUTBUFs to be allocated.

- Message-queue and CIOP utilizations should be carefully monitored during the introduction of GDDM to an installation, and periodically thereafter.

The Interactive Chart Utility and symbol editors under IMS

GDDM applications cannot be run as conversational transactions, because the architecture of GDDM makes it impossible to preserve the status of one conversation between inputs for that conversation. However, the Interactive Chart Utility and the symbol editors must work in a conversational mode. To solve the problem of preserving status, the following technique is used:

- A utility scheduler program, ADMUTIL, is provided that runs in "wait for input" mode. A user who wants to use the Interactive Chart Utility, or one of the symbol editors, must enter the transaction code for the utility scheduler, with the name of the specific utility as a parameter.
- ADMUTIL then schedules a new "instance" of the utility which will be a subtask of the scheduler. Unique GETMAINed working storage is obtained for the subtask, and the appropriate GDDM code is loaded, or located in the LPA.
- Only one copy of the GDDM code is used by all users.
- DL/I calls cannot be made from within a subtask, so the scheduler issues DL/I calls for the subtask.
- The code of the subtask is run until a response from the terminal operator is required. It passes control back to the scheduler, where a GU is reissued. The next message is read from the message queue, and the LTERM is checked to see if it is either a request for a new instance of a utility, or a response from a terminal operator for an existing instance of a utility.

  If the input message is a response for an existing instance of a utility, the scheduler passes control to the correct instance of the utility. Otherwise, a new instance of a utility is scheduled.

These points affecting performance should be noted:

- The GETMAINed storage associated with an "instance" of a utility cannot be shared. This means that only a finite number of parallel conversations can be supported, before the storage capacity of the MPR is exceeded.
- All transactions for the utilities are routed through one MPR. The IMS class scheduling cannot be used to balance the load on the MPRs, which may result in large variations in response time if there are many users of the interactive utilities. You can control the number of concurrent users with the IMSUMAX external default.
- In light use, the MPR is still occupied, because ADMUTIL runs in "wait for input" mode.
- It has been observed that the measurement of processor time by the IMS Monitor program is affected by the tasking structure of the utility scheduler. The processor time recorded appears to be only that associated with the utility scheduler. The processor time for the "instances" of the utilities, which are attached as subtasks, is not counted.

IMS tuning—things you can do

Here are some suggestions and comments about some of the things you can do to tune IMS.

- Put GDDM into a link-pack area if you have several potential, concurrent GDDM users or if you are not short of real storage. Remember that the search time for modules in the LPA can be increased if you have a large number of STEPLIB data sets.
- Package GDDM into a single load module to reduce the loading overhead, even if the code is in a link-pack area.
- Configure the 3274 to support data-stream decompression if it is teleprocessor-attached, and the line speed is 19 200 bits per second or less.
- Carefully consider the performance implications of managing large output data streams in non-SNA environments, before installing GDDM in such environments.
- Increase the size of the GDDM transmission buffer to 4KB, except for remote non-SNA environments, by specifying the IOBFSZ external default.
- Tune the long message record size to the GDDM transmission/save buffer sizes.
- Consider message-queue and CIOP space before GDDM is installed, and monitor after installation.
- Consider running terminals in response mode to regulate the data flow.
- Use resource-management and class-scheduling facilities to control processor usage by GDDM applications under IMS.
- Use nonrecoverable transactions when appropriate.
- Review the teleprocessor-access-method parameters that can affect performance:
  - Choice of buffer sizes and RU sizes
  - PACING/VPACING and PASLIM parameters in SNA.
- The use of a utility scheduler has enabled the GDDM interactive utilities to run under IMS. However:
  - A dedicated “wait for input” MPR is required.
  - It has been observed that the measurement of processor time by the IMS Monitor program is wrong for the MPR when the ADMUTIL scheduler is run.
- Review the number of concurrent users that may be expected for the interactive utilities, in light of the performance constraints of the implementation. The number of concurrent users is limited by:
  - The storage space within the MPR. This depends on which utilities are used. The GDDM code is shared between instances of the utilities, so only one copy is required. There is also a dynamic storage requirement for each user.
  - The response time for the MPR running ADMUTIL. This depends on many factors, such as the loading of the system, the relative priority of the MPR, and the hardware used.

- Vary the IMSUMAX external default to overcome variable response times for the GDDM utilities which run under the ADMUTIL scheduler. IMSUMAX defines the maximum number of concurrent conversations with the scheduler. Its default value is 5.
- If your graphics terminals are attached to a 3274-1D controller, the Prepare To Read feature should be specified in the OS System Generation I/O device macro.

TSO tuning—background information

This section contains several terms used in TSO tuning. If you need an explanation, further information is given in the *MVS Initialization and Tuning Guide*.

Controlling the data stream

The I/O synchronization option (the IOSYNCH external default) determines which of two types of TPUT is to be performed:

- TPUT NOHOLD (no synchronization of terminal I/O)
- TPUT HOLD (wait for a response from the terminal before sending any more data; that is, synchronize the I/O).

The TPUT HOLD option therefore regulates the data flow into the network from the GDDM application.

If the HOLD option is selected for a particular user, every TPUT issued causes the address space to incur an OUTPUT WAIT SWAP until a response has been received from the terminal, when the address space is swapped back in.

Another method of controlling the data stream is to use the TSO system options. The size of the TSO output buffer is set by the MAXRU parameter in the LOGMODE table for SNA (maximum 1536 bytes); for non-SNA it is the same size as the application TPUT, that is, no segmenting of data occurs.

The number of buffers available to manage TSO terminal output is defined on a global basis in TSOKEY00 for VTAM; this is a member in SYS1.PARMLIB. The philosophy of output-buffer handling for VTAM is to specify a maximum amount of buffer space that can be in use for output by an address space. This is defined by HIBFREXT, which is the amount of buffer space. When this level is reached, MVS puts the address space into an OUTPUT WAIT SWAP, until the amount of buffer space builds up to a minimum level (LOBFREXT), when a new TSO transaction is started in the first period. Buffer space is returned to the free buffer pool as the positive responses are received from the terminal on receipt of the output.

For example:

Assume:	TSO output buffer size	= 1024 bytes
	HIBFREXT	= 4KB
	LOBFREXT	= 2KB

The IOBFSZ external default buffer size = 1024 bytes.

If GDDM does four TPUTs in succession, 4KB of output buffers are used, and the high buffer extent is reached. The address space is swapped out until 2KB of buffers have been freed by the receipt of two positive responses.

This mechanism may therefore come into effect if large amounts of data are being given to the access method. It would probably not be seen with locally attached terminals, because responses would be returned rapidly, but it may come into play in the remote environment if the TPUT NOHOLD option is used. Therefore, you can use it to control the amount of output data entering the network from one user, and to produce less swapping than the TPUT HOLD option. However, great caution should be exercised if this type of control is imposed on a TSO system, because the size of output buffers and the high/low extents are global. Other terminal output is subject to the same constraints, so the high and low buffer extents should be chosen to allow other TSO users to run without incurring excess swapping.

The I/O synchronization option offered by GDDM can be applied to individual users by giving each an External Defaults File. It is likely that these users may incur more SWAP activity than would be the case using the HI/LOBFREXT technique, although total system swaps should be fewer.

For remote GDDM terminals not on dedicated lines, it is probably advisable to use the TPUT HOLD option if you want to prevent response times increasing unduly for non-GDDM users.

Swapping

SWAP activity may become significant to the total system utilization, particularly if the GDDM output buffer size (the IOBFSZ external default) is small and the graphics output is complex. This causes swap instruction processing greater than would usually take place in TSO, as follows:

SWAP OUT/IN PAIR = APPROX. 83K INSTRUCTIONS

(This assumes a swap page rate of 12, and 3 pages per START I/O.)

For a sample benchmark data stream:

Total compressed data-stream length = 6718 bytes
GDDM/TSO output buffer sizes = 1024 bytes
Number of transmissions = 7

TOTAL ADDITIONAL SWAP INSTRUCTIONS/PICTURE = 581K

The increased swapping produces contention both for processor cycles and for the DASD subsystem which, depending on the existing workload, may be critical.

This swapping can theoretically be reduced by use of logical swapping; however, in a heavily loaded system, most swaps are physical.

If an address space incurs an “output wait swap” because of buffer utilization, a new transaction is initiated when processing resumes. The statistics on the number of transactions therefore become distorted, and there is greater contention for the MPL in the first period.

TSO performance groups

The addition of GDDM transactions to an existing TSO system should lead you to reevaluate the IPS, particularly if more than one or two graphics terminals are going to be running concurrently with other TSO users. GDDM transactions are not insignificant, and increase contention in the system, particularly for the active-address-space limits specified in the MPL. Swapping increases, particularly if TPUT HOLD is specified or if the maximum amount of buffer space is used. This affects the total processor utilization and the response time. Paging activity may also increase; the contention for real storage may increase the swap and page data-set response.

Two alternatives are available to minimize the impact of graphics on the system:

- Increase the MPL
- Run GDDM in a different PGN and DOMAIN.

The second alternative has several advantages. It allows increases in the number of graphics terminals to be absorbed by the system with minimal impact on other users; the IPS for the GDDM PGN can be tuned separately; and the typical graphics transaction for an installation can be monitored. Both alternatives may have real-storage implications; increasing the MPL to allow more resident TSO transactions may cause the paging rate to increase to unacceptable levels.

TSO tuning—things you can do

Here are some suggestions and comments about some of the things you can do to tune TSO.

- Put GDDM into a link-pack area if you have several potential, concurrent GDDM users or if you are not short of real storage. If you are using the TSO CALL command to access the GDDM utilities, such as the ICU, from the SADMMOD data set, you must move the utility load modules to another data set and amend your CALL command to point to the new data set. Otherwise, you will still be loading the GDDM code from the SADMMOD data set because the CALL command always searches the data set on the command before looking in the LPA. Remember that the search time for modules in the LPA can be increased if you have a large number of STEPLIB data sets.

  Note: ADMECOMT and ADMOPUT are non-reentrant and must not be executed from the LPA.

- Package GDDM into a single load module to reduce the loading overhead, even if the code is in a link-pack area.
- Configure the IBM 3274 to support data stream decompression if it is teleprocessor-attached, and the line speed is 19 200 bits per second or less.
- Increase the size of the GDDM transmission buffer, the IOBFSZ external default to 4KB for all local and SNA remote users.
- For remote non-SNA GDDM users, consider changing the GDDM transmission buffer size, the IOBFSZ external default. Do not make it larger than the normal value which is 1 536. For remote SNA users, changing the MAXRU parameter in the LOGMODE table has the same effect.

  A large size means:

  – The response times of other remotely attached users becomes degraded.
  – The response times for remotely attached GDDM users is better.

  - Fewer calls to the terminal-access method, so lower processor utilization.

  A small size means:

  - Less impact on the response times of other remotely attached users
  - Worse response times for remotely attached GDDM users
  - More calls to the terminal-access-method, so higher processor utilization.

- Control the data streams in remote networks by using either:
  - TPUT HOLD, specified by the ADMMDFT IOSYNCH=YES default, which can apply to individual users, or
  - HI/LOBFREXT mechanism, which is system-wide.
- Control GDDM in the processor by either:
  - Increasing the MPL, or
  - Running GDDM in a different PGN and DOMAIN.
- If your graphics terminals are attached to a 3274-1D controller, specify the Prepare To Read feature in the OS System Generation I/O device macro.

VM tuning—background information

Shared segment

The largest potential performance improvement in a VM environment is obtained by putting GDDM into a saved segment. It has been observed that response times for the initial GDDM display in an application can be reduced by several seconds when a saved segment is used. This is discussed earlier in this chapter.

There are several reasons why GDDM code in a saved segment might not be used, even if you intend it to be:

- The name of the segment may be wrong. The names suggested as test names in the GDDM 3.2 ADMDCSS CONFIG file are:

GDDM-Base	ADMBA320
GDDM-PGF	ADMPG213
GDDM-IVU	ADMIV113
GDDM-GKS	ADMGK113
GDDM-IMD	ADMIM213

  The test names can be changed.

  The default names are:

GDDM-Base	ADMXSS00
GDDM-PGF	ADMPG000
GDDM-IVU	ADMIV000
GDDM-GKS	ADMGK000
GDDM-IMD	ADMIM000
GDDM-REXX	ERXRX000

- The starting address of the segment may overlap with addresses in the user's virtual machine. This could well occur if virtual machine storage size gets increased for any reason.

- Explicitly including the GDDM packaging stub ADMUX00V when loading the application overrides the use of most of the GDDM code in the saved segment.
- This applies to 43xx Processors only.

  When a saved segment is protected (the default) the VM Command Processor (CP) must scan through all pages of the segment each time a virtual machine using it stops being dispatched. This is done to check if the change bit is on in a page. You can avoid this by coding PROTECT=OFF on the NAMESYS.

  You should use this technique only if your applications have all been thoroughly tested. If not, you may find that the shared segment gets corrupted by someone inadvertently storing data into shared virtual storage.

On 308x and 3090 machines, CP uses the segment-protect hardware rather than the scanning approach.

Time-outs for 3179-G, 3270-PC/G and /GX, 4224, and 5550 devices

While 3179-G color display stations, 3270-PC/G and /GX workstations, or 4224 printers are drawing they have no contact with the host. Although picture draw time is usually very short, for very complex pictures it can take several tens of seconds. Here, VM may act as if the terminal has gone out of service and disconnect it. You can vary the length of time that VM will wait before disconnecting the terminal with the

```
CP SET MITIME GRAF time
```

command. This specifies the number of seconds that VM will wait after a “missing interrupt” before disconnecting the terminal.

The default setting is 30 seconds. If you are drawing very complex pictures, particularly on the 3179-G or 3270-PC/G, which take longer to draw pictures than the 3270-PC/GX, then you should increase this. A figure of 90 seconds is suggested as a good starting point.

Chapter 23. Repackaging for performance

Use this chapter to:

- Repackage the GDDM executable code to reduce dynamic loading.
- Repackage a GDDM utility or GDDM application program so that dynamic loading is eliminated or reduced.
- Run multiple versions of GDDM with different defaults modules, or override shared defaults modules such as those in a VM saved segment.

Background to repackaging

GDDM executable code modules are, by default, loaded dynamically as needed. You can repackage them so that some or all of them are loaded during program initialization. You can also repackage them with a GDDM utility or application program so that everything is loaded together, eliminating dynamic loading entirely.

To understand this, and the similar techniques used to run with different defaults modules, you must understand how dynamic loading takes place.

If you do not do any repackaging, the following series of loads takes place when a GDDM application program or GDDM utility is run:

1. The GDDM utility or application program is loaded.
2. The GDDM initially loaded modules are loaded. These are:
   a. The application interface controller
   b. The external defaults module
   c. The subsystem initializer.
3. Subsequently, other GDDM modules are loaded as needed by the GDDM subsystem initializer, unless they are in shared system storage (saved segment on VM, link pack area (LPA) on MVS, or shared virtual area on VSE).

This process is shown in Figure 46 on page 252.

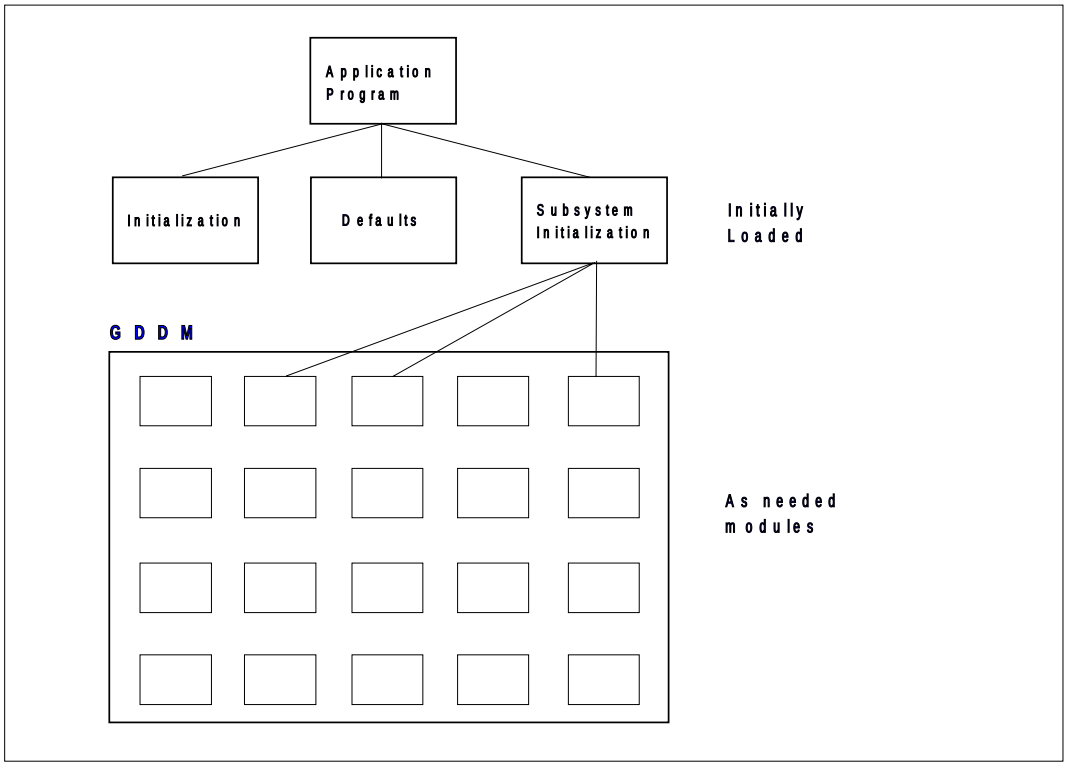


Figure 46. The default order of loading during GDDM utilities and programs

You can reduce the number of loads by packaging the items together. You can do this in several ways, the most important of which are:

1. Repackage the GDDM executable code so that “as needed” modules are loaded with the subsystem initializer
2. Repackage a utility or application program with all or part of the GDDM executable code
3. Repackage a utility or application program with a version of the GDDM external defaults module.

These options are shown in Figure 47 on page 253. Note that any module that you do not repackage is loaded dynamically as needed. GDDM still works regardless of what you put in or leave out of the repackaged modules.

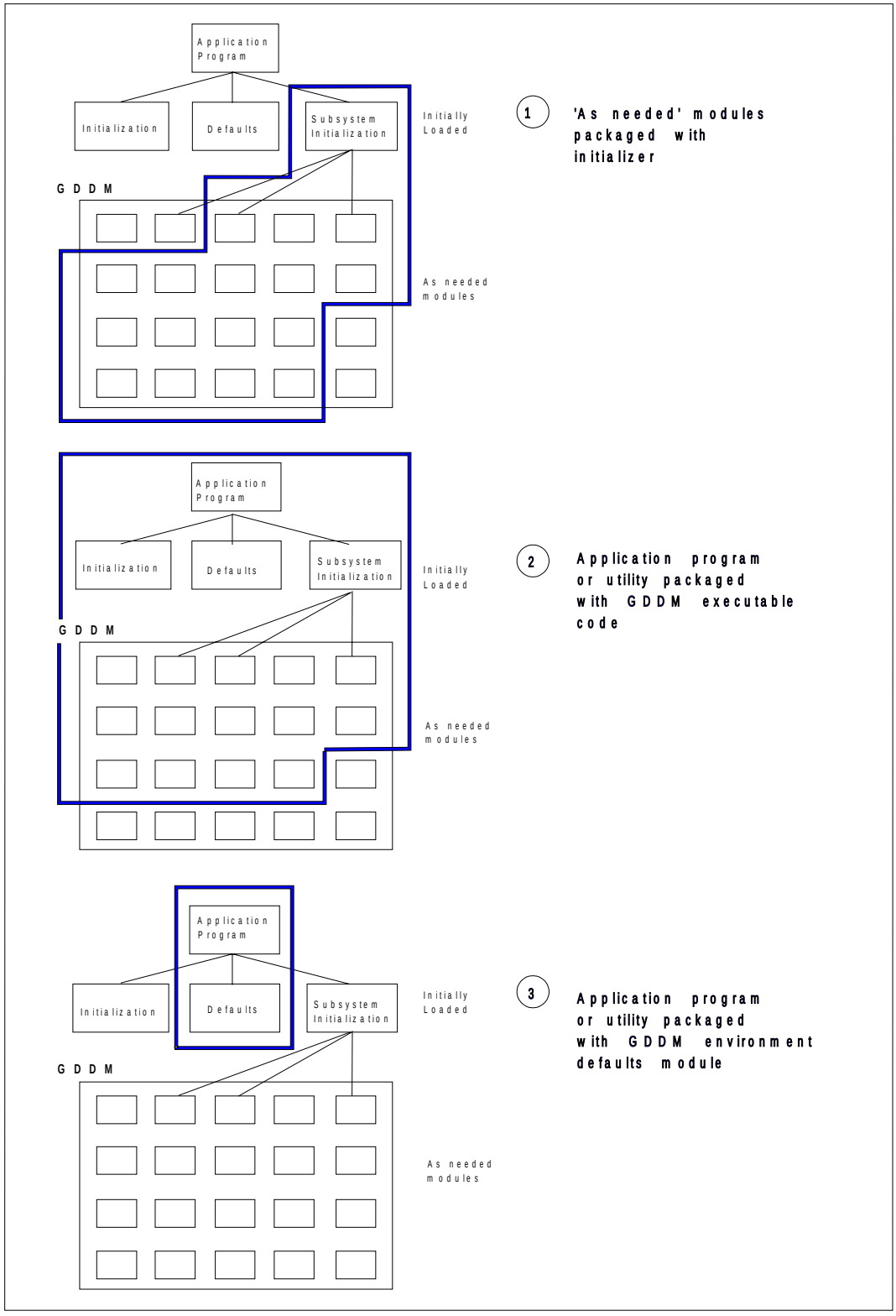


Figure 47. Possible loading combinations

How to repackage

You repackage GDDM using the linkage editor (or equivalent loading program). However, you have to do the linking of the GDDM modules that are “loaded as needed” in a particular way.

GDDM supplies modules known as *packaging stubs* that contain references to groups of associated modules. You must link these packaging stubs with the subsystem initialization module to “drag in” the associated modules. This is shown in Figure 48 on page 255.

Provisos about packaging

Before you consider packaging, you should be aware of these consequences.

Re-link-editing and possible future releases

GDDM releases from Version 1 Release 3 have been designed with the intention that application programs do not have to be re-link-edited as a matter of course for any subsequent releases of GDDM. However, if packaging options that involve link-editing the application are used, the application has to be re-link-edited for any subsequent release of GDDM.

Retaining original libraries for service

GDDM service procedures assume that the GDDM code has not been repackaged. Repackaged GDDM routines should, therefore, be kept separate from the original GDDM code. The original code should be retained to allow service to be applied. Packaged application programs or repackaged GDDM routines should be regenerated to include the serviced modules.

For more information about service procedures, see the *GDDM Program Directory* for your computer system.

Repackaging the GDDM executable code on its own

GDDM executable code is repackaged using the subsystem initialization module and packaging stubs. The subsystem initialization module is always loaded at the start of a program. Table 35 shows that it is called ADME000x, where “x” varies according to the subsystem you are using.

To repackage the executable code, link-edit the appropriate ADME000x module with one or more packaging stubs. The packaging stubs link other GDDM routines to form a load module that contains all the modules referenced in the packaging stubs. The composite module is given the same name as the ADME000x module. Note that this can only be done with the ADME000x modules, and that these modules can tell if they have been linked with executable code only if this has been done through packaging stubs.

The effect of repackaging the GDDM executable code is shown in Figure 48 on page 255.

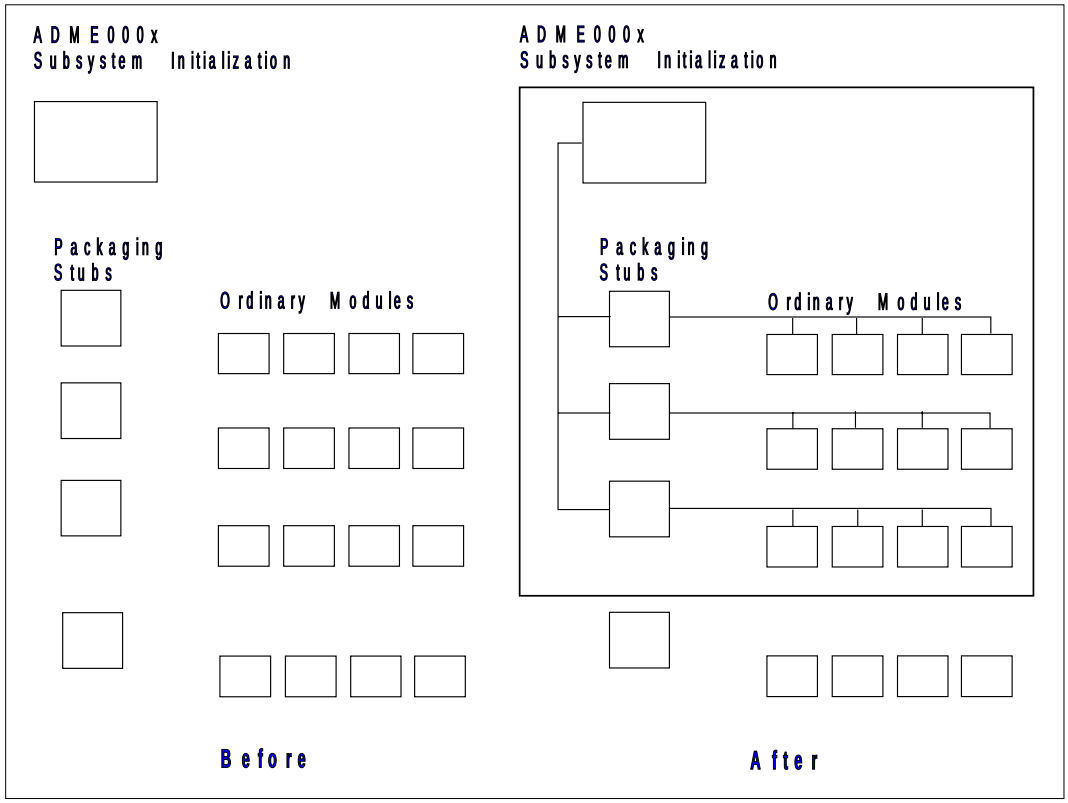


Figure 48. How packaging stubs are used to repackage the executable code

Table 35 shows the name of the subsystem initialization module for each subsystem.

Table 35. Names of initially loaded modules

Subsystem	Application interface initialization module	External defaults modules	Subsystem initialization modules
CICS	ADMACIN	ADMADFC	ADME000C
IMS	ADMACIN	ADMADFI	ADME000I
TSO	ADMACIN	ADMADFT	ADME000O
TSO Print Utility (VTAM)	ADMACIN	ADMADFT	ADME000O
VM	ADMACIN	ADMADFV	ADME000V

Levels of packaging stubs

Two levels of packaging stub are provided to give you more control of the modules you repackage. The full screen manager and the ICU can both be subdivided by use of second-level packaging stubs. (The full screen manager does most of the graphic and alphanumeric processing within GDDM.)

The contents of packaging stubs

Table 36 shows the contents of the first-level packaging stubs. Table 37 on page 258 and Table 38 on page 259 show the contents of the second-level packaging stubs for the full screen manager and the ICU respectively. Note that the use of a first-level stub automatically includes all the associated second-level stubs. Code sizes quoted are approximate and for guidance only.

Table 36 (Page 1 of 3). GDDM first-level packaging stubs

Name	Contents	Size
	GDDM Base subsystem-adapter routines	
ADMUX000	– common (used on all subsystems)	101KB
ADMUX00C	– subsystem-dependent – CICS	81KB
ADMUX00I	– subsystem-dependent – IMS	73KB
ADMUX00O	– subsystem-dependent – VTAM (used by TSO print utility on VTAM)	94KB
ADMUX00T	– subsystem-dependent – TSO	95KB
ADMUX00V	– subsystem-dependent – VM	89KB
ADMUXA00	Application-adapter routines – common	223KB
ADMUXA0x	Messages	
. . . . . A	– U.S. English	48KB
. . . . . B	– Brazilian Portuguese	53KB
. C	– Simplified Chinese (PRC)	37KB
. . . . . D	– Danish	48KB
. . . . . F	– French	49KB
. . . . . G	– German	50KB
. . . . . H	– Korean (Hangeul)	49KB
. I	– Italian	54KB
. . . . . K	– Japanese (Kanji)	52KB
. . . . . N	– Norwegian	48KB
. . . . . Q	– French (Canadian)	49KB
. . . . . S	– Spanish	54KB
. . . . . T	– Traditional Chinese (Taiwan)	39KB
. V	– Swedish	49KB
ADMUXD00	Full-screen manager routines (see Table 37 on page 258)	1848KB
ADMUXI00	Image Symbol Editor subroutines	127KB
ADMUXO00	Print utility subroutines	12KB
ADMUX300	Image routines	282KB
ADMUX400	Composite document routines	70KB

Table 36 (Page 2 of 3). GDDM first-level packaging stubs

Name	Contents	Size
ADMUXB00	GDDM-PGF Presentation Graphics routines	208KB
ADMUXB0x	Day/date routines	
. . . . . A	– U.S. English	1KB
. . . . . B	– Brazilian	1KB
. C	– Simplified Chinese (PRC)	1KB
. . . . . D	– Danish	1KB
. . . . . F	– French	1KB
. . . . . G	– German	1KB
. . . . . H	– Korean (Hangeul)	1KB
. I	– Italian	1KB
. . . . . K	– Japanese (Kanji)	1KB
. . . . . N	– Norwegian	1KB
. . . . . Q	– French (Canadian)	1KB
. . . . . S	– Spanish	1KB
. . . . . T	– Traditional Chinese (Taiwan)	1KB
. V	– Swedish	1KB
ADMUXPAx	GDDM-PGF and ICU messages	
. . . . . A	– U.S. English	20KB
. . . . . B	– Brazilian	21KB
. C	– Simplified Chinese (PRC)	20KB
. . . . . D	– Danish	21KB
. . . . . F	– French	20KB
. . . . . G	– German	20KB
. . . . . H	– Korean (Hangeul)	19KB
. I	– Italian	22KB
. . . . . K	– Japanese (Kanji)	20KB
. . . . . N	– Norwegian	19KB
. . . . . Q	– French (Canadian)	20KB
. . . . . S	– Spanish	21KB
. . . . . T	– Traditional Chinese (Taiwan)	20KB
. V	– Swedish	19KB
ADMUXP00	ICU routines (see Table 38 on page 259)	312KB
ADMUXP0x	Menus and HELP	
. . . . . A	– U.S. English	398KB
. . . . . B	– Brazilian	418KB
. C	– Simplified Chinese (PRC)	398KB
. . . . . D	– Danish	414KB
. . . . . F	– French	406KB
. . . . . G	– German	395KB
. . . . . H	– Korean (Hangeul)	385KB
. I	– Italian	428KB
. . . . . K	– Japanese (Kanji)	391KB
. . . . . N	– Norwegian	393KB
. . . . . Q	– French (Canadian)	406KB
. . . . . S	– Spanish	408KB
. . . . . T	– Traditional Chinese (Taiwan)	398KB
. V	– Swedish	350KB
ADMUXV00	Vector Symbol Editor subroutines	200KB
ADMUX100	GDDM-IMD subroutines	335KB
ADMUX10A	GDDM-IMD messages	20KB

Table 36 (Page 3 of 3). GDDM first-level packaging stubs

Name	Contents	Size
ADMUXJ00	GDDM-GKS routines	325KB
ADMUXJAx	GDDM-GKS messages	
. . . . . A	– U.S. English	7KB
. . . . . B	– Brazilian	7KB
. C	– Simplified Chinese (PRC)	7KB
. . . . . D	– Danish	7KB
. . . . . F	– French	7KB
. . . . . G	– German	7KB
. . . . . H	– Korean (Hangeul)	7KB
. I	– Italian	7KB
. . . . . K	– Japanese (Kanji)	7KB
. . . . . N	– Norwegian	7KB
. . . . . Q	– French (Canadian)	7KB
. . . . . S	– Spanish	7KB
. . . . . T	– Traditional Chinese (Taiwan)	7KB
. V	– Swedish	7KB
ADMUX500	GDDM-IVU routines	200KB
ADMUX5Ax	GDDM-IVU messages	
. . . . . A	– U.S. English	5KB
. . . . . B	– Brazilian	5KB
. C	– Simplified Chinese (PRC)	5KB
. . . . . D	– Danish	5KB
. . . . . F	– French	5KB
. . . . . G	– German	5KB
. . . . . H	– Korean (Hangeul)	5KB
. I	– Italian	5KB
. . . . . K	– Japanese (Kanji)	5KB
. . . . . N	– Norwegian	5KB
. . . . . Q	– French (Canadian)	5KB
. . . . . S	– Spanish	5KB
. . . . . T	– Traditional Chinese (Taiwan)	5KB
. V	– Swedish	5KB

Table 37 (Page 1 of 2). Full screen manager second-level packaging stubs

Name	Contents	Size
Alphanumeric stubs	For alphanumeric-only support, use all the following:	
ADMUXDA0	Alphanumerics	62KB
ADMUXDB0	Partitions	7KB
ADMUXDC0	Common device processor	39KB
ADMUXDE0	Partition sets	7KB
ADMUXDN0	Window control processor	11KB
ADMUXDP0	Direct printer support	44KB
ADMUXDS0	Supervisor	156KB
ADMUXDW0	Data stream generator	285KB
	Total	611KB
Graphic stubs	For basic graphics support, use all the alphanumeric stubs plus:	
ADMUXDG0	Graphics device processor	760KB
ADMUXDH0	Graphics generator (for PS graphics)	42KB

Table 37 (Page 2 of 2). Full screen manager second-level packaging stubs

Name	Contents	Size
Image stub	For image support, use all the alphanumeric stubs plus:	
ADMUXD30	Image processor	32KB
Mapping stub	For runtime mapping support, use all the alphanumeric stubs plus:	
ADMUXDM0	Mapping processor	60KB
HPA	For high-performance alphanumerics support, use all the alphanumeric stubs plus:	
ADMUXDQ0	HPA call processor	16KB
ADMUXDR0	HPA data stream generator	34KB
Miscellaneous stubs	Miscellaneous individual stubs to be added as needed:	
ADMUXDI0	Queued printer support (input)	22KB
ADMUXDJ0	Page printers	66KB
ADMUXDK0	IPDS	74KB
ADMUXDL0	System printers	6KB
ADMUXDO0	Queued printer support (output)	19KB
ADMUXDT0	Plotters	84KB
ADMUXD40	Text processor	42KB
	Total	313KB

Table 38 (Page 1 of 2). ICU second-level packaging stubs

Name	Contents	Size
ADMUXPMx	Menus, messages: choose from list	
. . . . . A	– U.S. English	61KB
. . . . . B	– Brazilian	65KB
. C	– Simplified Chinese (PRC)	61KB
. . . . . D	– Danish	63KB
. . . . . F	– French	64KB
. . . . . G	– German	60KB
. . . . . H	– Korean (Hangeul)	67KB
. I	– Italian	64KB
. . . . . K	– Japanese (Kanji)	68KB
. . . . . N	– Norwegian	62KB
. . . . . Q	– French (Canadian)	64KB
. . . . . S	– Spanish	63KB
. . . . . T	– Traditional Chinese (Taiwan)	61KB
. V	– Swedish	59KB

Table 38 (Page 2 of 2). ICU second-level packaging stubs		
Name	**Contents**	**Size**
ADMUXPHx	Help menus: choose from list	
. . . . . A	– U.S. English	338KB
. . . . . B	– Brazilian	355KB
. C	– Simplified Chinese (PRC)	338KB
. . . . . D	– Danish	354KB
. . . . . F	– French	344KB
. . . . . G	– German	337KB
. . . . . H	– Korean (Hangeul)	320KB
. I	– Italian	368KB
. . . . . K	– Japanese (Kanji)	326KB
. . . . . N	– Norwegian	333KB
. . . . . Q	– French (Canadian)	344KB
. . . . . S	– Spanish	348KB
. . . . . T	– Traditional Chinese (Taiwan)	338KB
. V	– Swedish	293KB

Which stubs to use

You should include only those packaging stubs that are of interest to you. This means that you exclude any message routines that are in languages that you do not use at your enterprise (for instance, exclude French, English, and so on if you use German).

When you have excluded these obvious stubs, you should consider the processing code.

The main processing for all graphics in both GDDM Base and GDDM-PGF is done by the full screen manager and major savings can be made by including this in any repackaging.

For the full screen manager and the ICU, second levels of packaging stubs are supplied, to enable their functions to be subdivided.

You may consider excluding the utility subroutines from the package and repackaging them separately. If you are doing this, the Image Symbol Editor and Vector Symbol Editor stubs are candidates for exclusion, as is the print utility stub. Note that for GDDM-IMD, the frame definitions are held as GDDM objects, and cannot therefore be packaged with GDDM-IMD.

Before you exclude the ICU stub, you should be aware that the ICU can be called by an application program. Also, you should be aware that the ICU makes use of the GDDM-PGF processing routines, which in turn make use of the full screen manager.

For application image handling, the image routines, ADMUX300, can be used without the full screen manager

For device-image handling, ADMUX300 is required as well as ADMUXD30.

Rather than using ADMUXD00, which gives the whole of the full screen manager, individual functions can be packaged. Table 37 on page 258 shows the stubs required to do this. You could use these stubs if, for example, you are using GDDM only for its alphanumeric support.

Rather than using ADMUXP0x, which gives all the ICU frames, you can separate the HELP panels from the menu panels. Table 38 on page 259 shows the stubs required to do this. You could use these stubs, for example, to repackage only the working code, and have the HELP panels loaded as needed.

Repackaging executable code with a utility or application program

When you repackage GDDM executable code with an application program or the invoking routine for a GDDM utility, you use the linkage editor to link the program and the subsystem-initialization module. This, in turn, is linked with the executable code. In practice this means packaging:

- The application program or the invoking routine for a GDDM utility
- The subsystem-initialization module
- Any packaging stubs that are relevant.

Special requirements for utilities

When repackaging the invoking routine for a GDDM utility, the internal link-edit name of the utility-invoking routine must be used. These names are shown in Table 39. If utilities are not listed in this figure, they cannot be repackaged.

Table 39. Internal link-edit names of GDDM utilities

Subsystem	Image Symbol Editor	Vector Symbol Editor	Chart Utility	Print Utility	GDDM-IMD
CICS	ADMISSEC	ADMVSSEC	ADMPSTBC	ADMOPUC	ADM1IMDC
IMS	ADMKSCHI	ADMKSCHI	ADMKSCHI	ADMOPUI	—
TSO	ADMISSET	ADMVSSET	ADMPSTBT	ADMOPUT and ADMOPST	ADM1IMDT
VM	ADMISSEV	ADMVSSEV	ADMPSTBV	ADMOPUV	ADM1IMDV

Notes:

Any link-editing of the CICS utility-invoking routines must also include DFHELII at offset 0 and DFHEAI0. (They are the CICS EXEC Interface stubs.)

Any link-editing of the IMS utility-invoking routines must also include ASMTDLI, the IMS Application Interface stub.

On MVS/XA, the above routines have these RMODE attributes:

For IMS, VTAM, TSO: RMODE(24)
For CICS: RMODE(ANY)

Special requirements for application programs

As described in the *GDDM Base Application Programming Guide*, applications have to be linked with a GDDM interface module (for example, ADMASLC on CICS). This can be done either automatically through using a special form of FSINIT within the program, or explicitly through linkage editor statements.

Exactly the same rules apply when including packaged GDDM executable code.

Eliminating dynamic loading completely

If you want to eliminate dynamic loading completely, you must include in the repackaged module the initially loaded modules and all the modules that would otherwise be loaded as needed.

All the initially loaded modules can be included by using packaging stubs. The common adapter stub (ADMUX000) automatically includes the application interface controller. The subsystem adapter stub (ADMUX00x) automatically includes the corresponding external defaults module and subsystem-initialization module.

There is no need to eliminate dynamic loading completely. Any modules that are needed, but are not included in the package, are loaded dynamically when required.

Repackaging an application or utility with a special defaults module

It is possible to override a shared defaults module by packaging an application program or GDDM utility-invoking routine with a special external defaults module. This is done by linking the application or utility-invoking routine with the external defaults module. Normal linkage editor (or equivalent) methods are used to link the application or utility-invoking routine and the required version of the defaults module.

The same method can be used if you want to use two sets of defaults in your enterprise, for example, if you wanted to use the ICU in both French and English. Assuming English was specified in the normal defaults module, you would link a copy of the ICU invoking routine with a defaults module that specified French, and make the resulting load module available with a different name. French users would use this version.

The name of the defaults module varies according to the subsystem as shown in Table 35 on page 255. You must assemble a suitable version of the defaults module before doing the packaging.

If you are repackaging with an application program you must follow the normal linking methods as described in “Special requirements for application programs” on page 261.

If you are repackaging with a utility-invoking routine, you must use the internal linkage-editor name for the utility-invoking routine as shown in Table 39 on page 261.

This use of a defaults module is in addition to other methods of specifying user defaults, which are described in Chapter 17, “GDDM user-default specifications” on page 197.

Repackaging for multiple subsystems

If you use GDDM in more than one MVS-based subsystem, you can repackage the executable code and place it in shareable storage available to all the subsystems.

The packaging stubs required for all of the subsystems that you intend to use should be included in such a composite routine. A linkage-editor NAME and ENTRY statement should be specified for one of the subsystem-initialization routines, and linkage-editor ALIAS statements should be specified for the others.

Application programs and utility-invoking routines can be included in the composite module, but only for one subsystem. This limitation arises from the link-editing mechanisms used by GDDM.

Special requirements for various systems and subsystems

The subsystems discussed here are CICS, IMS, and VM.

Special requirements for CICS

On some levels of CICS, there is a limit on the size of any one load module. Therefore, any repackaging of GDDM and GDDM-PGF modules for use in the non-MVS/XA CICS environment should be such that this limit is not exceeded.

Composite modules including utility-invoking routines or the subsystem initialization module for use under CICS must include DFHELII and DFHEAI0. DFHEAI must be at offset 0 in the composite load module.

Special requirements for IMS

Under IMS, the utility-invoking routines must be linked with the ASMTDLI, the IMS application interface stub.

Special requirements for VM

Under VM, the packaging facilities are used in the construction and operation of the GDDM saved segments. This is more fully described in the *GDDM/VM Program Directory*.

If the packaging stub ADMUX00V is loaded with an application program, it suppresses use of the saved segment.

Instructions for repackaging

The list below is a suggested order for doing repackaging:

1. If you are using CICS, IMS, or VM, see “Special requirements for various systems and subsystems.” If you are repackaging the defaults module, go to Step 4.
2. Look at “The contents of packaging stubs” on page 256 and decide which packaging stubs to include.
3. Find the name of your subsystem initialization module in Table 35 on page 255.
4. If you are changing the defaults module:
   a. Find the name from Table 35 on page 255.
   b. Change the defaults module as required. For more information, see Chapter 17, “GDDM user-default specifications” on page 197.
   c. Reassemble the defaults module.
5. If you are linking with a GDDM utility-invoking routine, find its internal link-edit name from Table 39 on page 261.
6. Use the linkage editor or equivalent to create a module containing the required elements and having the correct name.

The examples that follow can be used as models. Further information on link-editing with GDDM can be found in the *GDDM Base Application Programming Guide*.

Examples of repackaging

Examples are shown here for IMS, TSO, VSE, MVS, and VM.

Repackaging executable code under IMS

The following example shows the statements required to repackage GDDM executable code. The newly packaged load module contains all the GDDM and GDDM-PGF routines required for running GDDM-PGF Presentation Graphics Routines (excluding the ICU) using U.S. English.

The packaging shown in this example does not eliminate the dynamic loading of initially-loaded routines, `ADMADFI` and `ADMACIN`.

```
//LINK      EXEC PGM=IEWL,PARM='RENT,REFR,REUS,SIZE=(256K,64K)',
//               REGION=512K
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSLIB    DD   DSN=GDDM.SADMMOD,DISP=SHR
//SYSLMOD   DD   DSN=new-packaged-gddm-load-library,DISP=OLD
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *
 INCLUDE SYSLIB(ADME000I)      IMS subsystem initializer
 INCLUDE SYSLIB(ADMUX000)      Common subsystem adapter
 INCLUDE SYSLIB(ADMUX00I)      IMS subsystem adapter
 INCLUDE SYSLIB(ADMUXA00)      Common Application adapter routines
 INCLUDE SYSLIB(ADMUXA0A)      U.S. English messages
 INCLUDE SYSLIB(ADMUXD00)      Full screen manager
 INCLUDE SYSLIB(ADMUXB00)      GDDM-PGF common processing routines
 INCLUDE SYSLIB(ADMUXB0A)      GDDM-PGF U.S. English date text routine
 INCLUDE SYSLIB(ADMUXPAA)      GDDM-PGF U.S. English messages
 ORDER ADME000I
 ENTRY ADME000I
 NAME  ADME000I(R)
/*
```

Execution of the example generates a packaged version of the routine ADME000I, and stores this version in a new GDDM load module library.

Repackaging ICU with executable code under TSO

The following example shows the statements required to generate a version of the ICU to be used under TSO. It contains all the required routines for running using U.S. English. Such repackaging would make the ICU run faster because no dynamic loading would be required while it was running.

The ICU is a special GDDM application program using the GDDM System Programmer Interface (SPI). Similar statements are required to package a user application program.

```
//LINK      EXEC PGM=IEWL,PARM='RENT,REFR,REUS,SIZE=(256K,64K)',
//               REGION=512K
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSLIB    DD   DSN=GDDM.SADMMOD,DISP=SHR
//SYSLMOD   DD   DSN=newly-packaged-gddm-load-library,DISP=OLD
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *
 INCLUDE SYSLIB(ADMPSTBT)      ICU invoking routine link-edit name
 INCLUDE SYSLIB(ADMASPT)       GDDM/TSO interface stub
 INCLUDE SYSLIB(ADMUX000)      Common subsystem adapter
 INCLUDE SYSLIB(ADMUX00T)      TSO subsystem adapter
 INCLUDE SYSLIB(ADMUXA00)      Common Application adapter routines
 INCLUDE SYSLIB(ADMUXA0A)      U.S. English messages
 INCLUDE SYSLIB(ADMUXD00)      Full screen manager
 INCLUDE SYSLIB(ADMUXB00)      PGR common processing routines
 INCLUDE SYSLIB(ADMUXB0A)      PGR U.S. English date-text routine
 INCLUDE SYSLIB(ADMUXPAA)      PGR U.S. English messages
 INCLUDE SYSLIB(ADMUXP00)      ICU common processing routines
 INCLUDE SYSLIB(ADMUXP0A)      ICU U.S. English messages
 ORDER ADMPSTBT
 ENTRY ADMPSTBT
 ALIAS ADMCHART                Alias name of the ICU
 NAME  ADMPSTBT(R)
/*
```

Execution of the example generates a packaged version of the routine ADMPSTBT, and stores this version in a new GDDM load module library.

ADMCHART is an alias name for ADMPSTBT, through which the utility can be invoked.

Repackaging GDDM/VSE with the initially loaded modules

The following example shows the statements required to generate a version of a VSE PL/I application program using the reentrant interface to GDDM. The program is packaged with the initially loaded modules.

```
// OPTION CATAL
   PHASE phase-name, *
   INCLUDE DFHPL1I                   CICS PL/I EXEC interface stub
   INCLUDE PL/I-relocatable-module   The application program
   INCLUDE ADMASRB                   GDDM entry points (reentrant)
   INCLUDE ADMASLC                   GDDM CICS interface
   INCLUDE ADMUX000                  Common subsystem adapter
   INCLUDE ADMUX00C                  CICS subsystem adapter
// EXEC LNKEDT
/&
```

Repackaging executable code for subsystems under MVS

The following example shows the statements required to repackage all GDDM executable code (including GDDM-PGF and GDDM-IMD subroutines) for CICS, IMS, and TSO on MVS. The newly packaged load modules contain all the required GDDM/MVS, GDDM-PGF, and GDDM-IMD routines for execution using U.S. English, but do not include the GDDM utility-invoking routines listed in Table 39 on page 261. The packaging shown in this example does not eliminate the dynamic loading of initially-loaded routine, ADMADFx.

```
//LINK      EXEC PGM=IEWL,PARM='RENT,REFR,REUS,SIZE=(256K,64K)',
//          REGION=512K
//SYSUT1    DD   UNIT=SYSDA,SPACE=(CYL,(5,1))
//SYSLIB    DD   DSN=GDDM.SADMMOD,DISP=SHR
//SYSLMOD   DD   DSN=new-packaged-gddm-load-library,DISP=OLD
//SYSPRINT  DD   SYSOUT=*
//SYSLIN    DD   *

  INCLUDE SYSLIB(ADME000C)     CICS subsystem initializer
  INCLUDE SYSLIB(ADMUX00C)     CICS subsystem adapter
  ORDER DFHELII
  ORDER ADME000C
  ENTRY ADME000C
  NAME  ADME000C(R)

  INCLUDE SYSLIB(ADME000I)     IMS subsystem initializer
  INCLUDE SYSLIB(ADMUX00I)     IMS subsystem adapter
  ORDER ADME000I
  ENTRY ADME000I
  NAME  ADME000I(R)

  INCLUDE SYSLIB(ADME000O)     TSO subsystem initializer
  INCLUDE SYSLIB(ADMUX00O)     VTAM subsystem adapter
  INCLUDE SYSLIB(ADMUX00T)     TSO subsystem adapter
  ORDER ADME000O
  ENTRY ADME000O
  NAME  ADME000O(R)

  INCLUDE SYSLIB(ADMUX000)     Common subsystem adapter
  INCLUDE SYSLIB(ADMUXA00)     Common Application adapter routines
  INCLUDE SYSLIB(ADMUXA0A)     U.S. English messages
  INCLUDE SYSLIB(ADMUXD00)     Full screen manager
  INCLUDE SYSLIB(ADMUXB00)     GDDM-PGF common processing routines
  INCLUDE SYSLIB(ADMUXB0A)     GDDM-PGF U.S. English date text routine
  INCLUDE SYSLIB(ADMUXPAA)     GDDM-PGF U.S. English messages
  INCLUDE SYSLIB(ADMUXP00)     ICU common processing subroutines
  INCLUDE SYSLIB(ADMUXP0A)     ICU U.S. English panels
  INCLUDE SYSLIB(ADMUXI00)     Image Symbol Editor subroutines
  INCLUDE SYSLIB(ADMUXO00)     Print Utility subroutines
  INCLUDE SYSLIB(ADMUXV00)     Vector Symbol Editor subroutines
  INCLUDE SYSLIB(ADMUX100)     GDDM-IMD subroutines
  INCLUDE SYSLIB(ADMUX10A)     GDDM-IMD messages
  ORDER ADMACIN
  NAME  ADMACIN(R)
/*
```

Execution of the example generates packaged versions of the routines ADME000C (for CICS), ADME000I (for IMS), ADME000O (for TSO and TSO Print Utility), and ADMACIN (common to all subsystems), and stores these versions in a new GDDM load module library.

Repackaging application program with executable code under VM

The following example shows the VM commands entered to create a composite load module for an application program and to execute the composite module. All GDDM routines other than those required for the production of language-dependent error messages are included in the composite module. The language-dependent routines would be dynamically loaded should the application program require the generation of an error message.

The commands shown suppress the use of any copy of GDDM and PGF installed into a shared segment:

```
GLOBAL TXTLIB ADMPLIB ADMRLIB ADMGLIB PLILIB
LOAD appln-name ADMUX000 ADMUX00V ADMUXA00 ADMUXD00 (START
```

Loaded items are:

- The application
- The common subsystem adapter routine (ADMUX000)
- The VM subsystem adapter routine (ADMUX00V)
- The common application adapter routine (ADMUXA00)
- The full screen manager (ADMUXD00)

Part 5. GDDM workstation support

Chapter 24. Working with GDDM-OS/2 Link

This chapter describes how to install and set up GDDM-OS/2 Link for your users.

Making GDDM-OS/2 Link available

Note: Workstations running OS/2 and Communications Manager/2 V1.1.1 or later do not need GDDM-OS/2 Link because the host graphics emulator code is included in Communications Manager/2.

You can also run host graphics in the OS/2 environment without GDDM-OS/2 Link by running the IBM Personal Communications/3270 emulator Version 3 or later. This emulator supports host graphics under Windows and, if you are running IBM Personal Communications/3270 Version 4.1.1 or later, it supports host graphics under Windows 95.

This section describes how to make GDDM-OS/2 Link available under the MVS subsystems of TSO and CICS (GDDM-OS/2 Link is not available under the IMS subsystem), VM, and VSE subsystem CICS.

GDDM-OS/2 Link is installed on the personal computer system under OS/2.

You need:

- GDDM Base and GDDM NLS, which includes GDDM-OS/2 Link, installed successfully on your host computer.
- A personal computer system running IBM Operating System/2 Extended Edition Version 1.2 or later.

GDDM Base Version 3.2 must be used to download the version of GDDM-OS/2 Link that is provided with it. After the initial download of GDDM-OS/2 Link, the personal computer system can access any other host that has GDDM Base Version 2.3, or later, installed.

Installing GDDM-OS/2 Link

Migrating from earlier releases

For existing GDDM-OS/2 Link users, the new version of GDDM-OS/2 Link is automatically downloaded to the personal computer system the first time a GDDM Version 3 Release 2 application is run. This assumes that if the host application runs under TSO, the ADMPC ddname has been correctly set up to point to the new SADMPCF data set, or if the host application runs under CICS, the new GDDM-OS/2 Link objects have been loaded to the ADMF data set. No changes need to be made to the personal computer system to run the new level of GDDM-OS/2 Link. You can now move on to "Testing GDDM-OS/2 Link" on page 275.

When a new level of GDDM-OS/2 Link has been downloaded to the workstation, the function of GDDM-OS/2 Link at the new level becomes available only after you stop and restart Communications Manager/2 twice, as described in the on-screen instructions. (We recommend that you do not stop the upgrade process after one restart.) When you first access the new level of GDDM-OS/2 Link, GDDM downloads the relevant control files. The executable files are loaded the next time you access GDDM-OS/2 Link. Any existing version of GDDM-OS/2 Link on your personal computer system is overwritten.

National language support

The NLS used by GDDM-OS/2 Link is determined by the language that the OS/2 Communications Manager has been installed with.

Installing GDDM-OS/2 Link for the first time

For new users, instructions for installing GDDM-OS/2 Link should be provided by your system support personnel. GDDM-OS/2 Link can easily be installed from a CMS host session. However, further work is needed before GDDM-OS/2 Link can be installed on TSO or CICS.

This is what you do:

Under CICS on MVS or VSE, you must already have:

- Defined the CICS transaction GQFI. See the information on updating CICS resource definitions in the appropriate *GDDM Program Directory*.
- Load the GDDM-OS/2 Link objects into the ADMF data set. See the information on managing the CICS VSAM data sets required by GDDM in the appropriate *GDDM Program Directory*. The ADMF data set must be defined in your CICS start-up JCL, and have an entry in your FCT or CSD.

On TSO:

- Customize the CLIST GQFINSTH in the SADMSAM data set and concatenate your SADMSAM data set to the SYSPROC DD statement, or copy GQFINSTH to an existing SYSPROC data set.
- Allocate the DD statement ADMPC to point to the SADMPCF data set.

On all host systems:

1. On the workstation, start an IBM 3270 terminal emulation session (see the documentation for the level of OS/2 that you are using) and log onto the host

system on which GDDM Base and GDDM Base NLS (which includes GDDM-OS/2 Link) is installed.

- Ensure that you have access to these installed licensed programs and that your host system is in the "ready" state.
- On VM, exit from any subsystem such as PROFS or CMS SUBSET mode. Set all messages off and suppress the blip characters while you are downloading GDDM-OS/2 Link code. (Issue the CMS command `SET BLIP OFF`.)

2. Select the OS/2 command prompt and type the following command to install host graphics:

   ```
   HGINST
   ```

3. Press the ENTER key.

   This starts the OS/2 HGINST installation program.

   You first see the main installation panel for loading GDDM-OS/2 Link into your workstation:

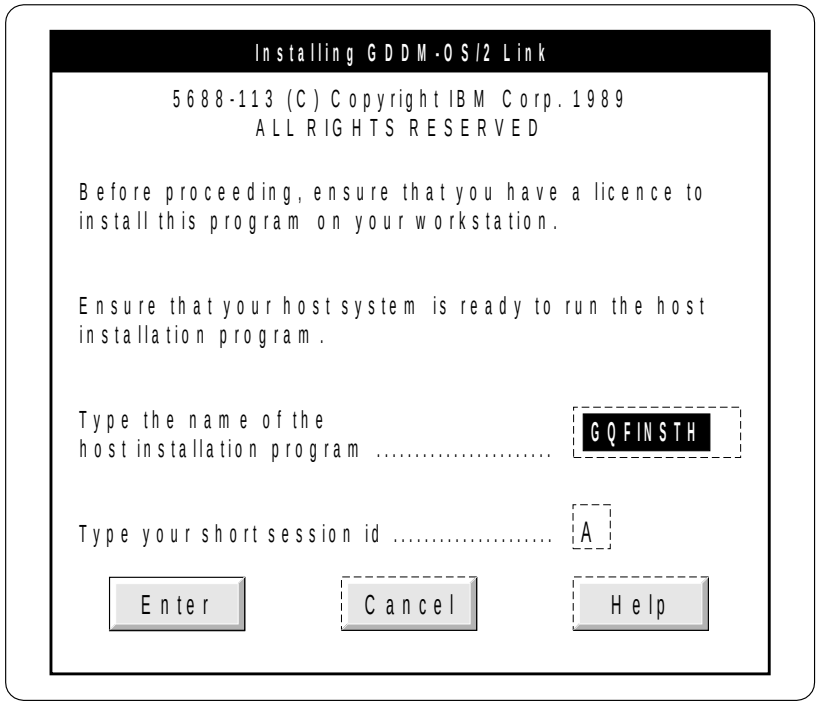


Figure 49. GDDM-OS/2 Link installation panel

4. On CICS, change the name of the host installation program to match the CICS transaction name (normally GQFI) to be used for this download.
5. Check the short session id, and change it as required. (The short session id is the one-letter name with which the session was configured; note that on-line help information is available.)
6. Press the ENTER key (or select Enter with the mouse).

   The transfer of files from your host computer starts and you see the following panel:

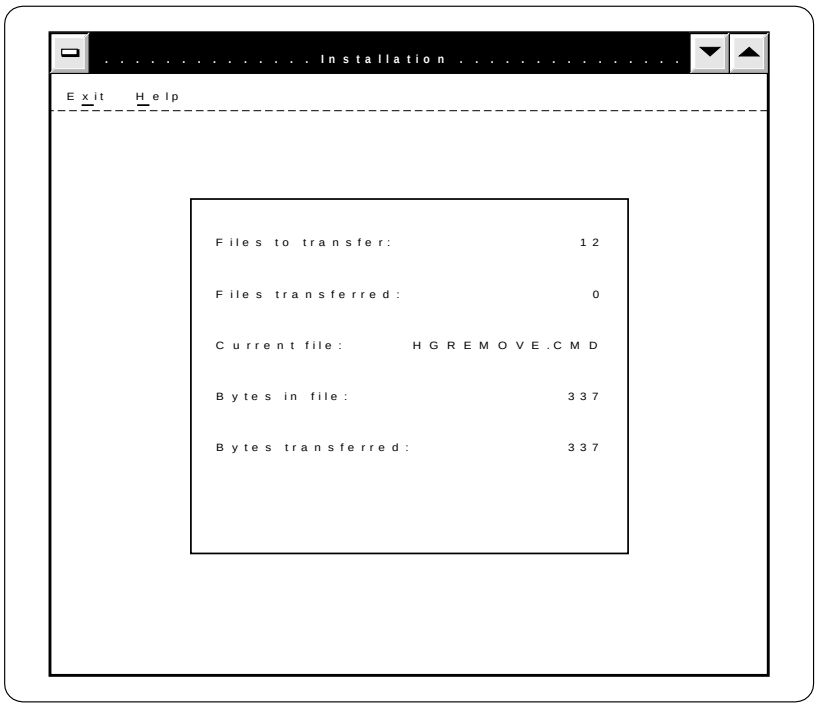


Figure 50. GDDM-OS/2 Link installation file-transfer panel

The values on the right of this panel change as the files are transferred.

On completion, you should see a message indicating that the installation has completed successfully. However, if any errors have occurred, see "Recovering from installation errors" which lists some of the more common errors. Other errors are described in the documentation for the level of OS/2 that you are using. You may also find it useful to check the OS/2 Communications Manager Error log for more information.

7. Although the GDDM-OS/2 Link files have been transferred, GDDM-OS/2 Link cannot immediately be used for host graphics applications on the workstation. To make GDDM available for use, you must first stop and then restart any current 3270 terminal emulation sessions on your workstation.
8. When you have done this, ensure that you have installed GDDM-OS/2 Link successfully by running a GDDM application; see "Testing GDDM-OS/2 Link" on page 275.

Recovering from installation errors

While running the HGINST program, you may receive the message `Installation failed, error code: nnnn`.

The values of `nnnn` and their meanings are:

30xx Problems with data transmission from the host computer; for example, line problems (data lost, or sequence checks). This probably indicates a hardware problem.

50xx and **800x** Problems associated with insufficient memory. Make more memory available; for example, by ensuring that swapping is enabled.

5014 This is a general communication failure that can happen while downloading GDDM-OS/2 Link if the controller fails or if an IBM 3270 session error occurs for one of the following reasons:

- GDDM is not installed or available on the host
- The IOBFSZ external default setting exceeds the limit defined by the network.
- On MVS, the TSO CLIST, GQFINSTH, has not been customized

9001 The short session name you supplied is not available. The session may not be started or you gave an invalid name.

9003 The connection to the Communications Manager cannot be made. The Communications Manager may not be started, or an incorrect system level is in use.

9011 An invalid host presentation space short session id was specified. For example, the session requested for the Install was configured as a logical printer session, rather than a Logical Terminal session.

9019 A system error has occurred.

9021 This resource is unavailable. The host presentation space is already being used by another system function. For example, a file transfer is in progress using the same session as that requested for the Install program.

190xx These are errors generated by the Communications Manager; **xx** is the EHLLAPI function return code. For more information, see the *IBM OS/2 EHLLAPI Programming Reference* for the level of OS/2 that you are using. (In particular, **19009** is an EHLLAPI system error, usually caused by the Communications Manager not being active.)

20xxx These are OS/2 errors; **xxx** is the OS/2 error code. To find out more about one of these messages, or if you receive a message that is not one of the above, type HELP followed by the error code number in an OS/2 session, and press ENTER.

Try to correct the cause of the error and then run the installation process again. If it continues to fail, you may have to report the problem to the IBM Support Center. For more information, see the *GDDM Diagnosis* book.

Testing GDDM-OS/2 Link

This is how to test that GDDM-OS/2 Link has been installed successfully on the personal computer system.

To test GDDM-OS/2 Link you need:

- GDDM Base and GDDM NLS, which includes GDDM-OS/2 Link, installed successfully on your host computer.
- A personal computer system running IBM Operating System/2 Extended Edition Version 1.2 or later.
- GDDM-OS/2 Link successfully downloaded to the personal computer system.

There are no specific tests for GDDM-OS/2 Link. All you do is run a GDDM host application, such as the Image Symbol Editor or the installation verification program. See the information on testing GDDM in the appropriate *GDDM Program Directory*.

If GDDM-OS/2 Link has been successfully installed, it is now ready to be used with GDDM.

Running GDDM-OS/2 Link

When you have ensured that GDDM-OS/2 Link is successfully installed on the personal computer system, it is now ready for use. Information about using GDDM-OS/2 Link can be found in the online *GDDM-OS/2 Link User's Guide*, which is installed as part of GDDM-OS/2 Link.

You can access the online information in one of two ways:

- Via the VIEW.EXE program from the GDDM-OS/2 Link group.
- Via *hypertext* links from the online help system.

While using GDDM-OS/2 Link, you may receive error messages. GDDM-OS/2 Link messages start with GQF, and are described in the *GDDM Messages* book. Online help information is also available for these messages. To use the online help, type HELP followed by the error code number in an OS/2 session, and press ENTER.

More information about diagnosing problems with GDDM-OS/2 Link can be found in the *GDDM Diagnosis* book.

Controling host-graphics support for each emulator session

If you are using Communications Manager/2 V1.1 (with APAR PJ07845) or later, you can enable or disable host graphics for each 3270 terminal emulator session you use. The default setting is for all emulation sessions to have host-graphics support but when you don't need it, you can switch this support off and save your system's resources.

Depending on the level of Communications Manager/2 you are running, you can either select **Host Graphics..** directly from the system menu or via the **Emulator operations...** selection. In the **Host Graphics..** dialogue box, you can enable or disable host graphics for the current emulator session using the check box provided. You can also use the radio buttons provided to specify how Redraw should be performed.

Choosing an alternative locator cursor for interactive graphics applications

If you have GDDM-OS/2 Link installed, you may notice a greater variety of cursors and locator echoes available to you in your interactive graphics applications.

If you have a keyboard with an alternative cursor (ALT-CUR) key, you can toggle through the following types of system cursor:

- Black-and-white target
- Black-and-white cross
- XOR target
- XOR cross
- XOR crosshair

Note: The functions described above are available only if the host machine is running GDDM 3.1.1 or later, together with Communications Manager/2 V2.1 or later. Even if you are using the latest release of GDDM-OS/2 Link

these functions are not available unless GDDM is running on the host to which your workstation is connected.

Removing GDDM-OS/2 Link

If for any reason you need to remove the GDDM-OS/2 Link licensed program from a workstation, use the command file HGREMOVE.CMD to erase the GDDM-OS/2 Link files. This command can be used to recover from installation errors, or to delete GDDM-OS/2 Link from a workstation when transferring the right to use GDDM-OS/2 Link from one device to another under the license agreement. Be aware that before you run HGREMOVE, you must stop the communication link with the host computer.

Note: This information is also described in the online *GDDM-OS/2 Link User's Guide*, which is available to you when you have successfully installed GDDM-OS/2 Link.

Chapter 25. Working with GDDM-PCLK

Note: Workstations running DOS and the IBM Personal Communications/3270 Emulator Program Version 4 or later do not need GDDM-PCLK because the host graphics emulator code is included in Personal Communications/3270.

This emulator supports host graphics under Windows and, if you are running IBM Personal Communications/3270 Version 4.1.1 or later, it supports host graphics under Windows 95.

This chapter describes how to install and set up GDDM-PCLK for your users.

Setting up terminal emulators for GDDM-PCLK

Before GDDM-PCLK can be run to display graphics produced by a host computer graphics program, there must be a *terminal emulator* present on the personal computer system.

This section describes options that you can select when installing the supported emulators, which are:

- IBM PC3270 Emulation Program, Entry Level, Version 1.1, 1.2, or 2.0
- IBM Personal Communications/3270 (see below)
- IBM 3270 Workstation Program Version 1.1 (see below)
- IBM 3270 Workstation Program Version 1.2 (for PS/2 Model 70 and higher)
- IBM PC 3270 Emulation Program (Version 3.05)

For installation instructions, see the appropriate emulator documentation.

Alphanumerics and graphics can be merged when using GDDM-PCLK with all emulators, but the IBM Personal Communications/3270 and the IBM 3270 Workstation Program emulators provide the best performance.

IBM PC3270 Emulation Program, Entry Level, Versions 1.1, 1.2, and 2.0

If you are using Versions 1.1 or 1.2, ensure that APAR IR82709 is installed.

For more information, refer to the appropriate book:

- *IBM PC3270 Emulation Program, Entry Level, Version 1.1.*
- *IBM PC3270 Emulation Program, Entry Level, Version 1.2.*
- *IBM PC3270 Emulation Program, Entry Level, Version 2.0.*

IBM Personal Communications/3270

For best performance, if the Corrective Service Diskette (CSD) has been installed on PC3270, load the interface program PCSPCLK after you have started the emulator. If the CSD has not been installed, ensure that the Low Level API (LLAPI) option is selected when configuring the emulator.

IBM 3270 Workstation Program Version 1.1

Ensure that you have the necessary service-level updates. You may find there is not enough space in the "NAME OF SYSTEM EXTENSION" field on Panel 1.1 for the input value C:\PCLK11\PX.EXE. If this is so, you must arrange to get the necessary update for the Workstation Program code.

You must provide the following input values for the fields specified for the terminal emulator panels–the program default values are valid for the remaining fields.

Home panel

Check that you have enabled the COPY option.

This results in Panel 1.1 being shown.

Panel 1.1

Field	Input value	Comment
NAME OF SYSTEM EXTENSION	C:\PCLK11\PX.EXE	The name of the system extension. The drive specified here must be the same drive that contains the GDDM-PCLK program.
STORAGE	16	The amount of storage required for the system extension, in KB.
DOS	No	Indicates whether the system extension is to use DOS functions.
DEFAULT DRIVE	C	The default disk drive for this system extension. It must be the same drive that contains the GDDM-PCLK program.

Panel 3

In the "STORAGE" field, define a personal computer system session with enough storage to run GDDM-PCLK. For typical storage requirements, see "Workstation storage requirements for GDDM-PCLK" on page 222.

For more information about the installation process, see the *IBM 3270 Workstation Program User's Guide and Reference* manual.

IBM 3270 Workstation Program Version 1.2

The setting up instructions are the same as for the IBM 3270 Workstation Program Version 1.1. See page 280.

IBM 3270 Emulation Program (Version 3.05)

Check that you loaded PSCAPI before starting the emulation program, and check that you have enabled the Application Programming Interface (API) option on the Emulation Configuration panels.

For more information, see *IBM 3270 Emulation Program Version 3.05: User's Guide*.

Making GDDM-PCLK available

This section describes how to make GDDM-PCLK available under the MVS subsystems of TSO and CICS (GDDM-PCLK is not available under the IMS subsystem), VM, and VSE subsystem CICS.

GDDM-PCLK is installed on the personal computer system under DOS.

You need:

- GDDM Base and GDDM NLS, which includes GDDM-PCLK, installed successfully on your host computer.
- A personal computer system running DOS Version 2.1 or later, with 500KB of free disk space on either:
  - A suitable terminal emulator; see “Setting up terminal emulators for GDDM-PCLK” on page 279.
  - A fixed disk, or
  - One empty formatted diskette, or two diskettes if they are of 360KB capacity.

    If you are installing on a 360KB double-sided disk drive, you need to do the following:

    1. Copy the display driver (either GQDXX10A.EXE or GQDXX06A.EXE onto another diskette and delete it from the PCLK11 subdirectory.
    2. Before running GDDM-PCLK, preload the display driver by inserting the diskette (see above) on which the display driver was stored and type `GQDXX10A` or `GQDXX06A`, as appropriate.

    GDDM-PCLK can now be installed. The display driver remains resident until the workstation is next rebooted. When GDDM-PCLK finds a display driver preloaded, it does not check that the driver exists in the PCLK11 subdirectory.

Setting up a nickname file: If you intend to use a host computer graphics program that has not been written specifically for GDDM-PCLK, you must supply a *nickname* user-default specification on your host computer system, in either the external defaults module, or the external defaults file.

Add the following line, which must always be preceded by a blank:

```
ADMMNICK FAM=1,PROCOPT=((PCLK,YES))
```

The nickname statement provides a *processing option* that tailors the host graphics program to GDDM-PCLK, without you having to change the application itself.

For more information about nicknames and supplying user-default specifications, see Chapter 18, “Nickname user-default specifications” on page 205 and Chapter 17, “GDDM user-default specifications” on page 197.

GDDM Base Version 3.2 must be used to download the version of GDDM-PCLK that is provided with it. After the initial download of GDDM-PCLK, the personal computer system can access any other host computer that has GDDM Base Version 2.2, or later, installed.

Installing GDDM-PCLK

Migrating from earlier releases

For existing GDDM-PCLK users, the new version of GDDM-PCLK is automatically downloaded to the personal computer system the first time a GDDM Version 3 Release 2 application is run. This assumes that if the host application runs under TSO, the ADMPC ddname has been correctly set up to point to the new SADMPCF data set, or if the host application runs under CICS (on MVS or VSE), the new GDDM-PCLK objects have been loaded to the ADMF data set. No changes need to be made to the personal computer system to run the new level of GDDM-PCLK. You can ignore the rest of the instructions in this section and move on to "Testing GDDM-PCLK" on page 287.

Any existing version of GDDM-PCLK on your personal computer system is overwritten.

Installing GDDM-PCLK for the first time

For new users, instructions for installing GDDM-PCLK should be provided by your system support personnel. GDDM-PCLK can easily be installed from a CMS host session. However, further work is needed before GDDM-PCLK can be installed on TSO or CICS.

This is what you do:

Under CICS on MVS or VSE, you must already have:

- Defined the CICS transaction GQFI. See the information on updating CICS resource definitions in the appropriate *GDDM Program Directory*.
- Load the GDDM-PCLK objects into the ADMF data set. See the information on managing the CICS VSAM data sets required by GDDM in the appropriate *GDDM Program Directory*. The ADMF data set must be defined in your CICS start-up JCL, and have an entry in your FCT or CSD.

On TSO:

- Customize the CLIST GQFINSTH in the SADMSAM data set and concatenate your SADMSAM data set to the SYSPROC DD statement, or copy GQFINSTH to an existing SYSPROC data set.
- Allocate the DD statement ADMPC to point to the SADMPCF data set.

GDDM-PCLK installation for new users is a three-stage process:

1. The GDDM-PCLK installation file is received from a host computer session on the personal computer system.
2. The GDDM-PCLK installation file is run on the personal computer system.
3. A GDDM application is run on the host computer session that enables GDDM to download the remaining GDDM-PCLK files from the host computer system. This can be done using a host computer session running TSO or CICS.

Distributing GDDM-PCLK to your users

You should decide at this stage how you are going to distribute GDDM-PCLK to your users. Choose one of the following methods:

- Copy the GDDM-PCLK installation file to a diskette and tell your users how to run it. This is the recommended method. The GDDM-PCLK installation file fits onto a 360KB diskette.
- Run the GDDM-PCLK installation file, then copy all of the GDDM-PCLK files from the \PCLK11 directory to a diskette. Your users do not need the GDDM-PCLK installation file in this instance. If the personal computer system onto which you copy the files uses different hardware to that used for the download, some additional files may be automatically downloaded when you first run GDDM-PCLK. You must ensure that the GDDM-PCLK files are copied to the directory \PCLK11, otherwise GDDM-PCLK will not work.
- Using the information supplied in this section, tell your users how to receive and run the GDDM-PCLK installation file themselves.

National language support

You must also decide which national language you are going to install on your personal computer system. From GDDM Base Version 3 Release 2, you cannot change the language of GDDM-PCLK on your personal computer system other than by running the appropriate GDDM-PCLK installation file. Therefore, if you want to run GDDM-PCLK with more than one national language at your enterprise, you must download the GDDM-PCLK installation file for each language you require.

To download and use a particular language on your personal computer system, you do not have to have that language installed on your host computer system. Table 40 shows the national language support for GDDM-PCLK.

Table 40. NLS for GDDM-PCLK

Language	**NLS identifier**
U.S. English	US
Brazilian	BR
German	GR
Italian	IT
Spanish	SP

Stage 1: Receiving the GDDM-PCLK installation file

This is what you do:

On MVS or VM:

1. Start the terminal emulator and log on to a TSO or CMS host computer session that has access to the SADMPCF data set (on MVS) or the VM disk containing the GDDM code. Make sure that the host computer session is in the “ready” state.
2. Jump to the DOS session.
3. If you are not installing GDDM-PCLK onto a fixed disk, insert a blank formatted diskette.

4. Select the appropriate file transfer command for the terminal emulator you are using:
   - If you are using the IBM 3270 Emulation Program, Entry Level emulator, the command is:

     On MVS:

     ```
     RECEIVE d:PCLKINST.EXE 'GDDM.SADMPCF(GQDINSxx)'
     ```

     On VM:

     ```
     RECEIVE d:PCLKINST.EXE GQDINSxx ADMPC *
     ```

   - If you are using the IBM Personal Communications/3270 emulator, the command is:

     On MVS:

     ```
     RECEIVE d:PCLKINST.EXE h:GDDM.SADMPCF(GQDINSxx)
     ```

     On VM:

     ```
     RECEIVE d:PCLKINST.EXE h:GQDINSxx ADMPC *
     ```

   where, in both cases, `d` is the letter for the diskette drive on which you are installing GDDM-PCLK, `h` is the host computer session short name (for example A or B), and `xx` is the national language identifier for the language you want to install, as shown in Table 40 on page 283.

On VSE:

1. Receive the GDDM-PCLK file onto your personal computer system using your normal download procedure. You must specify the options BINARY NOCRLF when you receive the file. For more information, see *Using IBM Workstations*.

   The GDDM-PCLK installation files are shipped as Z-type members in the GDDM sublibrary with the name GQDINSxx, where xx is the NLS identifier as shown in Table 40 on page 283. Receive the file on the personal computer system with the name PCLKINST.EXE.

2. If you want to run multiple national language versions of GDDM-PCLK at your enterprise, repeat Steps 3 and 4.

   On VSE, repeat the download for each national language you require. Either read the installation file for each language onto a separate diskette, or if you are using a fixed disk, change the name for PCLKINST.EXE for each language (for example, PCLKINUS.EXE) as you download it, otherwise you will overwrite the file. If you do change the name, you must substitute the name you have chosen when you come to run PCLKINST.

Stage 2: Running the GDDM-PCLK installation file

This is what you do:

1. Jump to the DOS session.
2. If you are installing onto 360KB diskettes, you need to make a second blank formatted diskette available.
3. Type `PCLINKST d:`, where `d` is the letter of the diskette drive onto which you want to install GDDM-PCLK. If you are installing onto 360KB diskettes, `d` must not be the current drive.

If you are installing onto 360KB diskettes, and you have only one drive, when you are prompted to load a new diskette, remove the diskette containing the GDDM-PCLK installation file and insert the blank diskette.

4. Press ENTER

   The IBM logo panel is displayed:

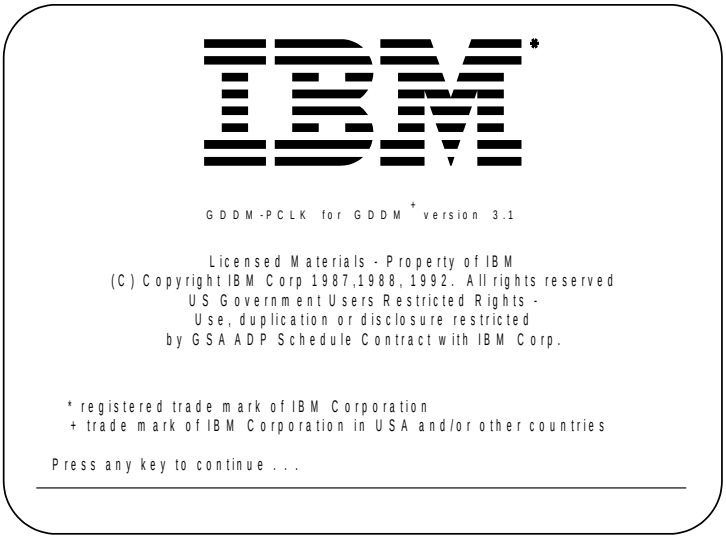


Figure 51. IBM logo panel

   Note: The appearance of this picture depends on the characteristics of your display terminal.

5. Press any key.

   The installation process creates a subdirectory called \PCLK11 on the disk or diskette in the specified drive, and installs the GDDM-PCLK program files into it. There are 8 GDDM-PCLK files in total. You receive the message:

   ```
   GQD1010  Installation in progress. Number of files left to create = n
   ```

   where `n` is a count of the number of files that GDDM-PCLK has yet to create.

   When installation is complete, the message:

   ```
   GQD1020 Installation complete.
   ```

   is displayed.

6. Press any key to clear the message and return to DOS.

You should now have successfully installed GDDM-PCLK on your personal computer system.

Stage 3: Running GDDM-PCLK with a GDDM application to complete the installation

There are some things you have to do before you can run GDDM-PCLK.

Setting up a mouse: If you intend to use a mouse, ensure that it is connected, and that you are familiar with its button arrangements. Ensure also that any separate mouse driver program has been loaded.

1. Start the terminal emulator and log onto TSO or CICS (on MVS), CMS (on VM). or CICS (on VSE).

2. Ensure that you have access to the GDDM-PCLK files on the host computer session.

   For TSO you must have the ddname ADMPC allocated to the SADMPCF data set. For CICS (under MVS or VSE), you must have copied the GDDM-PCLK files to the ADMF data set. See the appropriate *GDDM Program Directory*. The ADMF data set must be allocated in the startup JCL, and have an entry in your FCT or CSD.

   If you do not have access to the host-computer files, you receive the message

   ```
   ADM0307 E FILE 'a' NOT FOUND
   ```

   when you run the application program.

3. Start an application program that uses GDDM; for example, the GDDM-PGF Interactive Chart Utility (ICU).

4. When you receive the prompt:

   ```
   ADM0873 I IF AVAILABLE, PLEASE SELECT PCLK.  OTHERWISE, PRESS ENTER.
   ```

   hot key to the DOS session.

5. Ensure that the drive you specified with the PCLKINST command is current. Make the GDDM-PCLK directory (\PCLK11) current or specify it in your directory path.

6. Type the command PCLK, and press ENTER

   The IBM logo panel is displayed (see Figure 51 on page 285).

7. Press any key to get the GDDM-PCLK Main Panel:

Figure 52. GDDM-PCLK main panel

   Note: The appearance of this picture depends on the characteristics of your display terminal.

8. Select option 1.

   The remaining GDDM-PCLK files are automatically transferred from the host computer. The message

   ```
   GQD0030 File transfer in progress. Please wait. Number of bytes
           transferred so far = n
   ```

is displayed as each file is received by the personal computer system.

When all the files have been transferred, the GDDM-PCLK installation is complete, and you can stop your application program.

Testing GDDM-PCLK

This is how to test that GDDM-PCLK has been installed successfully on the personal computer system.

To test GDDM-PCLK you need:

- GDDM Base and GDDM NLS, which includes GDDM-PCLK, installed successfully on your host computer.
- A personal computer system of at least the minimum configuration attached to your host computer system. See "Workstation storage requirements for GDDM-PCLK" on page 222.
- A suitable terminal emulator.
- GDDM-PCLK successfully downloaded to the personal computer system.

There are no specific tests for GDDM-PCLK. All you do is run a GDDM host application, such as the Image Symbol Editor or the installation verification program (see the appropriate *GDDM Program Directory*.), from the personal computer system on which you have set up GDDM-PCLK, and check that the results you get are what you expected.

If GDDM-PCLK has been successfully installed, it is now ready to be used with GDDM.

Starting GDDM-PCLK

To run GDDM-PCLK:

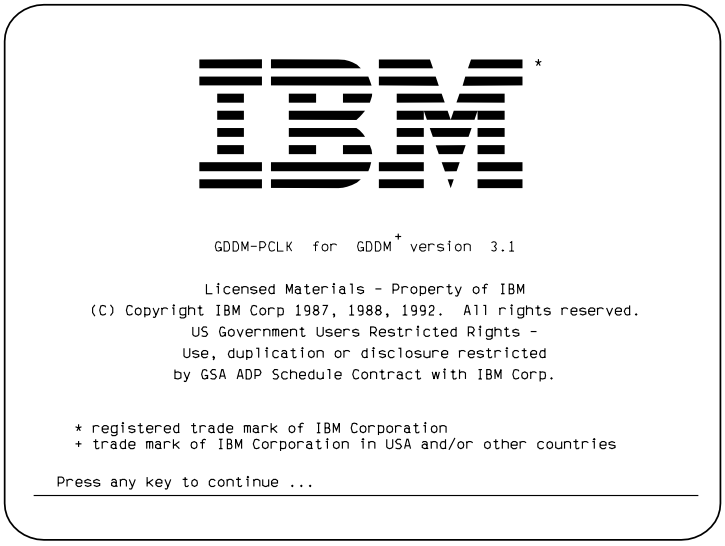
1. Log on to a host session.
2. Start a program that uses GDDM to write to your personal computer system; for example, the GDDM-PGF Interactive Chart Utility (ICU).
3. When you receive the prompt:

```
ADM0873 I IF AVAILABLE, PLEASE SELECT PCLK.  OTHERWISE, PRESS 'ENTER'
```

   hot-key to the DOS session.

4. Ensure that the drive you specified with the `pclkinst` command is current. Make the GDDM-PCLK subdirectory (\PCLK11) current or specify it in your directory path.
5. Type the command `pclk`
6. Press **ENTER**.

   The IBM logo panel is displayed in the national language specified when the operating system was installed on your host computer:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 53. IBM logo panel

7. Press any key to get to the GDDM-PCLK main panel:

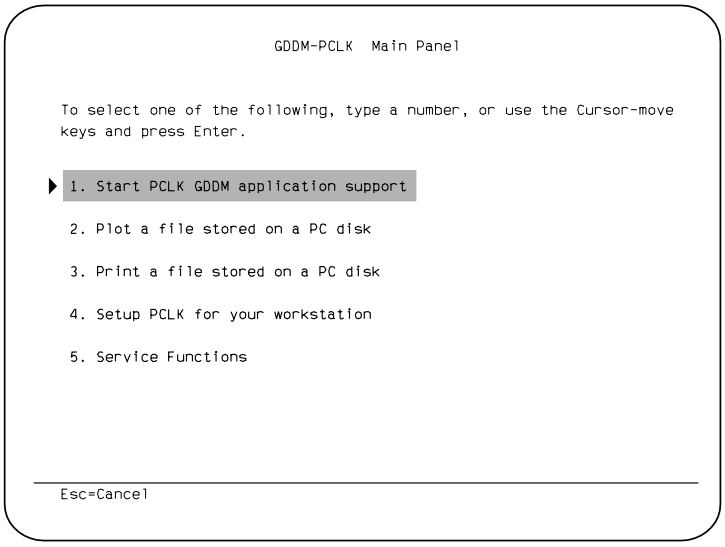


Figure 54. GDDM-PCLK main panel

The main panel gives you the options shown in Figure 54. A pointer symbol (►) indicates the option that is chosen when you press **ENTER**. On color screens, the option with the pointer also has a highlighted background. On monochrome screens, the option with the pointer is underscored.

You can use the Up Arrow (▲) and Down Arrow (▼) keys to move the pointer to a different option, or you can type the number of the option you want, without pressing **ENTER**.

8. If this is the first time you have used GDDM-PCLK, or you need to change your personal computer system hardware setup, select option 4.

9. If this is not the first time you have used GDDM-PCLK, continue at “Using GDDM-PCLK” on page 293.

Restarting GDDM-PCLK

If you have already run GDDM-PCLK in this session, you can bypass the normal starting procedure just defined:

1. Type the command `pclk /q`
2. Press **ENTER**.

   GDDM-PCLK resumes with option 1 selected.

Setting up GDDM-PCLK for your PC (option 4)

Option 4 on the main panel provides a series of panels that enable you to define to GDDM-PCLK which plotter, printer, terminal emulator, and performance and usability options you intend using.

There are three parts to the Setup Panel, and two versions of part 1 of the panel, depending on the terminal emulator you are using.

Setting up: part 1

If your terminal emulator gives you more than one host session, part 1 of the panel enables you to select which host session you want to use, and it looks like this:

```
Setup                  GDDM-PCLK  Setup

To select an item, position the cursor against it and press the
Space Bar. Press Enter to save selections on ALL THREE parts
                                                      Part 1 of 3

Plotter:                              Terminal emulator:

Attached?. . . . . ▶ No       Host session . . . ▶ a Host1
                     Yes                           b Host2
                                                   c Host3
Port . . . . . ▶ COM1     COM3                     d Host4
                 COM2     COM4                     e Host5
                                                   f Host6
Baud . . . . . ▶ 4800                             g Host7
                 9600                              h Host8


Model number . . . . . ▶ PCLK to query plotter
                         Use IBM model number  . . . . . .  [    ]

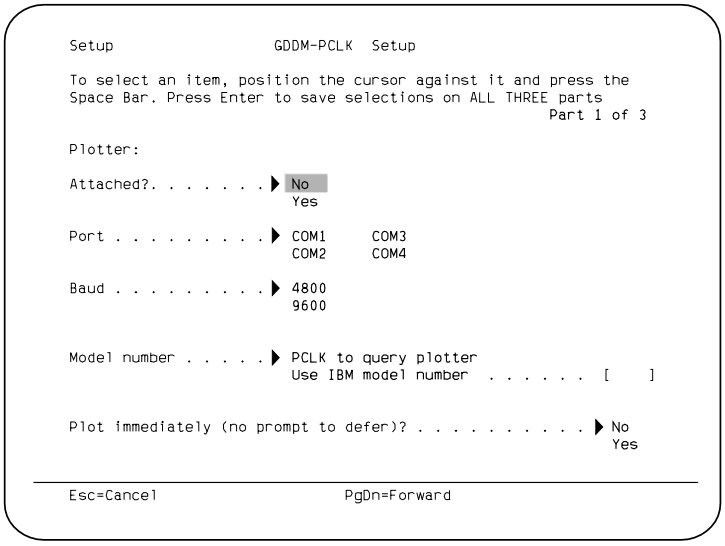

Plot immediately (no prompt to defer)? . . . . . ▶ No
                                                    Yes

Esc=Cancel                    PgDn=Forward
```

Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 55. Setup panel: part 1 (with host-session selection)

If you are using a terminal emulator that lets you access only one host session, part 1 of the panel looks like this:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 56. Setup panel: part 1 (without host-session selection)

The ► symbol indicates the current options. Do not press **ENTER** until you are sure that all three panels show the options that you want. Pressing **ENTER** saves the current options on all three panels.

Each question on the Setup panel is now dealt with in turn:

Attached?
Select `Yes` if you have a plotter attached. If you specify that a plotter is not attached, you cannot subsequently select a port for it.

Host session
When this option is available, select one of the host-session names displayed.

Port
Select either `COM1`, `COM2`, `COM3`, or `COM4` as appropriate for your plotter. For an explanation of these, see the section on DOS device names in the *DOS Reference* book that applies to the level of DOS on your personal computer system.

Baud
Select communication with your plotter at either 4800 or 9600 baud. This setting should match the setting on your plotter configuration switches; for more information, see the operating instructions for your plotter.

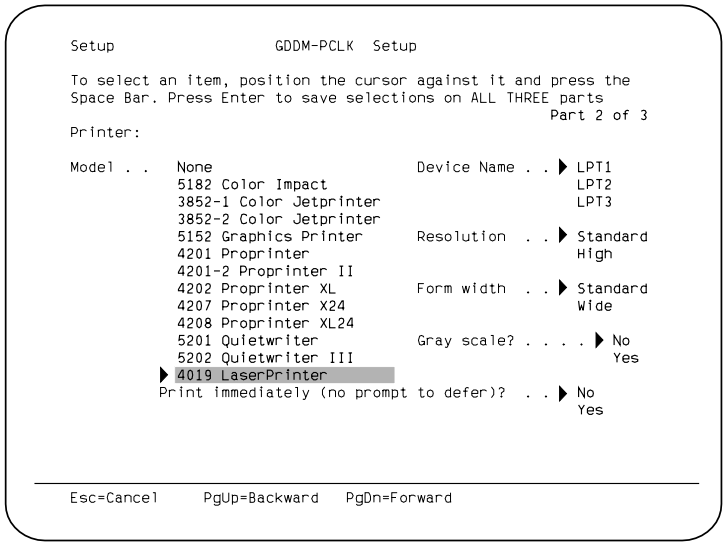
Note: The 9600 baud option is only available on PS/2 system units.

Model number
If you intend to use different plotters with your personal computer system, select `PCLK to query plotter` and do not type an IBM model number; otherwise, type the IBM model number of the plotter (for example, 7372). Note that you can type the model number for a plotter that you want to use for deferred plotting, even if you have specified that a plotter is not attached. This results in plot files being saved in a subdirectory with that model number as its name.

Plot immediately (no prompt to defer)?
If you intend always to plot immediately and do not want to be reminded about deferred plotting, select `Yes`. But remember that when using graphics and alphanumerics in merged mode, it is more efficient to defer your plotting.

Setting up: part 2

When you have answered these questions, either press **ENTER** to finish setting up, or press the **PgDn** key to display part 2 of the panel. Part 2 enables you to set up GDDM-PCLK for the type of printer you are using with your personal computer system:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 57. Setup panel: part 2

You can change the options in the same way as in part 1 of the panel. Each question on this panel is now dealt with in turn:

Model
Select `None` or a number from the list displayed.

If you do select a printer model of `None`, you cannot subsequently select a device name, resolution, form width, gray scale, or defer option for it.

Device name
Select one of `LPT1`, `LPT2`, or `LPT3` as appropriate for your printer. For an explanation of these, see the section about DOS device names in the *DOS Reference* book that applies to the level of DOS on your personal computer system.

Resolution
Select `High` to get a print with a maximum amount of detail. Note that printing takes longer in high-resolution mode.

Form width
Select `Standard` if the paper size in your printer is A4 (ISO) or quarto (ANSI size A).

Select `Wide` if the paper size in your printer is wider than A4 or quarto.

Gray scale?
If you select Yes, the different shades of gray are used to represent the different colors of your picture when it is printed. Seven shades of gray scale are available.

Note: This option is only available for IBM 4019, 5051, and 5052 printers.

Print immediately (no prompt to defer)?
This has the same effect as "plot immediately"; see page 291.

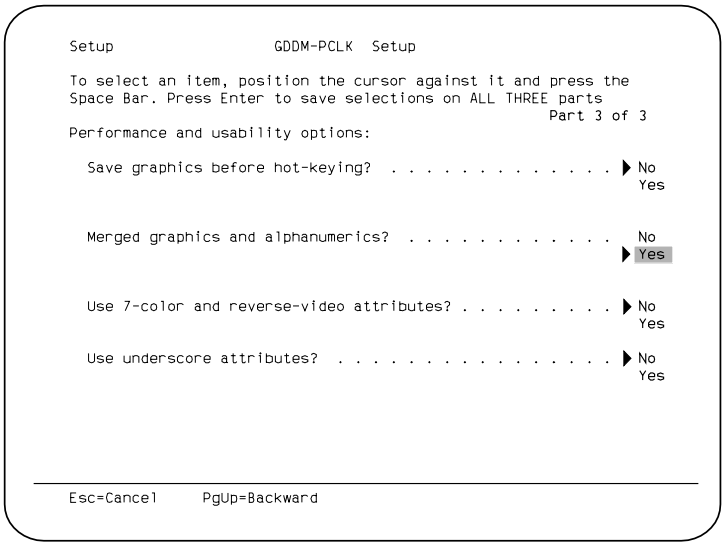
If you intend always to print immediately and do not want to be reminded about deferred printing, select Yes for this question. But remember that when using merged graphics and alphanumerics mode, it is more efficient to defer your printing.

You can press the **PgUp** key to return to part 1 of the setup panel.

Setting up: part 3

When you have answered these questions, either press **ENTER** to finish setting up, or press the **PgDn** key to display part 3 of the panel. Part 3 enables you to define performance and usability options for your plotted or printed output. You can define whether you want to:

- Save the graphics you have created before you hot-key
- Merge graphics and alphanumerics
- Use the default display attributes



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 58. Setup panel: part 3

Each question on this panel is now dealt with in turn:

Save graphics before hot-keying?
If you select Yes and there is enough memory available, GDDM-PCLK saves the graphics when you press **Ctrl**+**F9** before hot keying to the host session, and restores the graphics when you press **Ctrl**+**F9** again after hot keying back from the host computer session.

Merged graphics and alphanumerics?
By default, GDDM-PCLK runs in merged graphics and alphanumerics mode, which means that you don't have to hot key to see your graphics. However, for

performance reasons, in this mode you are recommended to defer any plotting or printing. We strongly advise that you do **not** run in non-merged mode.

Use 7-color and reverse-video attributes?
If you select `Yes`, you can use the following field and character attributes:

- Color (seven available)
- Reverse-video.

If you select `Yes` and your current emulator does not support the IBM 3270 Extended Data Stream feature, your request is ignored.

If you do not select this option, you can improve performance, and GDDM-PCLK needs less memory.

Use underscore attributes?
If you select `Yes`, you can use the underscoring character attribute.

This question is displayed for all terminal emulators, but is only enabled when you have selected the `Yes` option to `Use 7-color and reverse-video attributes?`. The default selection for this question is `Yes`.

If GDDM-PCLK is working with a control unit terminal (CUT) mode emulator, the `Yes` option is disabled, and the `No` option is preselected.

This enables the user to control independently whether 7-color and reverse-video attributes are supported at a performance cost, or underscore attributes at a memory and performance cost.

Finishing setup

When you have completed part 1, part 2, and part 3 of the GDDM-PCLK Setup panel, press **ENTER** to save the changes and return to the main panel.

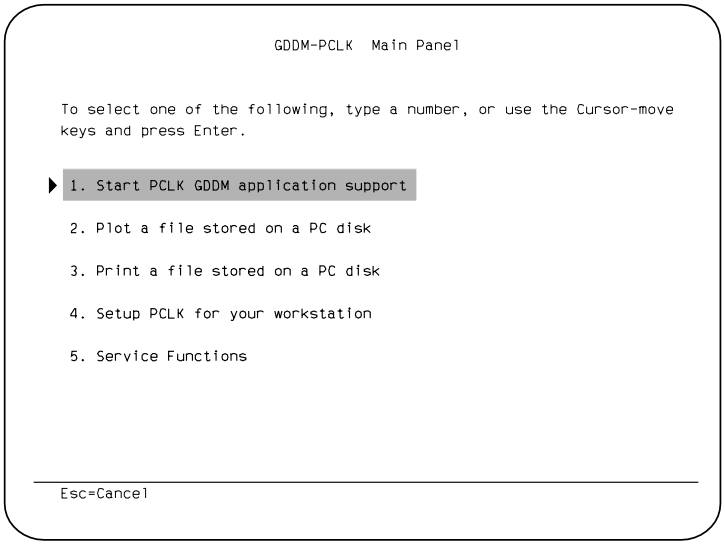
If you press **Esc**, you quit the panel. Any changes you made to part 1, part 2, or part 3 of the panel are ignored, and you return to the main panel.

You should now have completed setting up GDDM-PCLK for your preferred methods of processing, and your particular arrangement of personal computer system-attached plotters and printers. Now that GDDM-PCLK is set up, the first thing you should do is ensure that you have installed it successfully. For more information, see “Testing GDDM-PCLK” on page 287.

Using GDDM-PCLK

When you have ensured that GDDM-PCLK is successfully installed on the personal computer system, it is now ready for use.

1. Return to the GDDM-PCLK main panel, unless it is already displayed:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 59. GDDM-PCLK main panel

2. Select option 1.

While using GDDM-PCLK, you may receive error messages. GDDM-PCLK messages start with GQD, and are described in the *GDDM Messages* book.

More information about diagnosing problems with GDDM-PCLK can be found in the *GDDM Diagnosis* book.

GDDM-PCLK in merged mode

GDDM-PCLK operates in *merged* mode; that is, the graphics and alphanumerics are displayed together, giving a DOS session the appearance of a host session.

Running GDDM-PCLK in merged mode

To start and run GDDM-PCLK:

1. Hot-key to the terminal emulator host session, and log on to a host session.
2. Start a program that uses GDDM to display graphics (for example, the GDDM-PGF Interactive Chart Utility). The ADM0873 or ADM0874 message is displayed.
3. Hot-key to the DOS session. The drive that GDDM-PCLK is installed on must be the current drive. Ensure that the GDDM-PCLK subdirectory is current (it is called PCLK11).
4. Type

   ```
   pclk
   ```

   and press ENTER. The IBM logo panel is displayed.
5. Press any key to display the GDDM-PCLK main panel.

6. To select the first option, ensure that the pointer symbol (►) is pointing to option 1, and press ENTER, as shown in Figure 59. (To move the pointer symbol, use the cursor up (▲) and cursor down (▼) keys.)

GDDM-PCLK-supplied control keys in merged mode

In merged mode the GDDM-PCLK control keys are assigned as follows:

Ctrl+F1	t	Alpha cursor style [1]
Ctrl+F2	t	Alpha cursor/graphics cursor move [1] [2] [3]
Ctrl+F3		Exit GDDM application support
Ctrl+F4		Erase graphics and redraw alpha [1] [4]
Ctrl+F5		PA3 to host to redraw screen
Ctrl+F6		Not defined
Ctrl+F7	t	Graphic cursor style [3]
Ctrl+F8	t	Graphic cursor speed [2] [3]
Ctrl+F9		Suspend PCLK for hot-key
Ctrl+F10		Not defined

[1] Effective only in merged graphics mode.
[2] Not effective when a mouse is installed.
[3] Effective only when the graphic cursor is displayed.
[4] With some emulators, GDDM-PCLK cannot control when graphics are erased, and so they may not be erased when required. Therefore, the **F4** control-key function is provided to let the user decide when to erase graphics.
t "Toggle-action" key.

Single-screen workstation

1. Press **Ctrl**+**F9** to suspend GDDM-PCLK. This prompt is displayed:

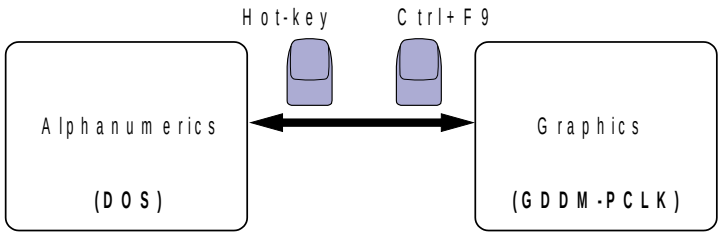
   ```
   GQD0020 Either "hot-key" to host session,
           or press Ctrl+F9 to resume GDDM-PCLK.
   ```

2. Hot-key to the host session and start your GDDM application.

   When your host program displays graphics, the following prompt is displayed:

   ```
   ADM0874 I  PLEASE SELECT GDDM-PCLK
   ```

3. Hot-key to the DOS session.
4. Press **Ctrl**+**F9** to resume GDDM-PCLK as shown in Figure 60. The graphics are displayed.



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 60. Ctrl+F9 action

5. When you have finished viewing the graphics, press **Ctrl**+**F9** to deselect GDDM-PCLK. This prompt is displayed again:

```
GQD0020 Either "hot-key" to host session,
        or press Ctrl+F9 to resume GDDM-PCLK.
```

6. Hot-key back to your host program.

Remember that the hot key gets you in and out of DOS, and **Ctrl**+**F9** gets you in and out of GDDM-PCLK. The sequence of actions is shown in Figure 60 on page 295.

Dual-screen workstation

If you have installed the IBM Personal Communications/3270 Emulation Program Entry Level, you can resume the DOS session without having to hot-key back to it.

Press the **Alt**+**R** keys in the host session. The resumed DOS session is active *concurrently* with the host session. Host-generated graphics are automatically displayed on the color display. You need to hot-key only to start or end GDDM-PCLK, or to switch access between the host session and the DOS session (for example, to perform interactive graphics). Pressing the **Alt**+**S** keys in the host session suspends the DOS session. If you have installed a different type of terminal emulator, this prompt is displayed when your host program displays graphics:

```
ADM0874 I  PLEASE SELECT GDDM-PCLK
```

You should then:

1. Hot-key to GDDM-PCLK in the DOS session to view the graphics on the DOS screen.
2. When you have finished viewing the graphics, hot-key back to your host program. The graphics continue to be displayed on the DOS screen.

GDDM-PCLK-supplied control keys in merged mode

In merged mode, the GDDM-PCLK control keys are assigned as follows:

Ctrl+F1		Not applicable
Ctrl+F2		Not applicable
Ctrl+F3		Exit GDDM application support
Ctrl+F4		Not applicable
Ctrl+F5		Not applicable
Ctrl+F6		Not defined
Ctrl+F7	t	Graphic cursor style
Ctrl+F8	t	Graphic cursor speed[1]
Ctrl+F9		Suspend GDDM-PCLK for hot-key
Ctrl+F10		Not defined.

[1] Not applicable when a mouse is installed.
t "Toggle-action" key.

Restarting GDDM-PCLK

If you have already run GDDM-PCLK in this session, you can bypass the normal startup procedure. To start GDDM-PCLK (with option 1 selected), type:

```
pclk /q
```

and press **ENTER**.

Exiting GDDM-PCLK

There are two ways to exit from GDDM-PCLK:

Normal exit

To exit GDDM-PCLK:

1. Press the **Ctrl**+**F3** keys. This ends the GDDM-PCLK application session.
2. At the GDDM-PCLK Main Menu, press the **Esc** key to return to DOS.

Quick exit

The quick exit option can be used to exit GDDM-PCLK to DOS without displaying the selection panels. To use quick exit, you need to have started GDDM-PCLK with the quick start option (see “Restarting GDDM-PCLK” on page 296). When you want to exit GDDM-PCLK application support, press **Ctrl**+**F3**.

Code pages on GDDM-PCLK

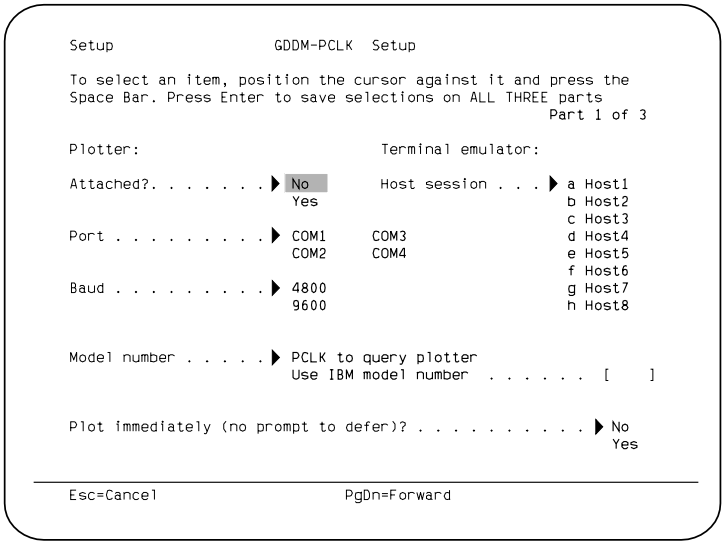
The code pages define the assignment of graphic characters and control function meanings to all code points (a code point being a 1-byte code representing one of 256 potential characters).

If the code page is changed between starting the emulator and starting GDDM-PCLK, incorrect characters may be displayed in alphanumeric fields.

Setting up GDDM-PCLK for your workstation (option 4)

Option 4 provides a series of panels that enable you to define to GDDM-PCLK which plotter, printer, terminal emulator, and performance and usability options you intend to use. If you are not sure about any of this information, ask your system-support personnel.

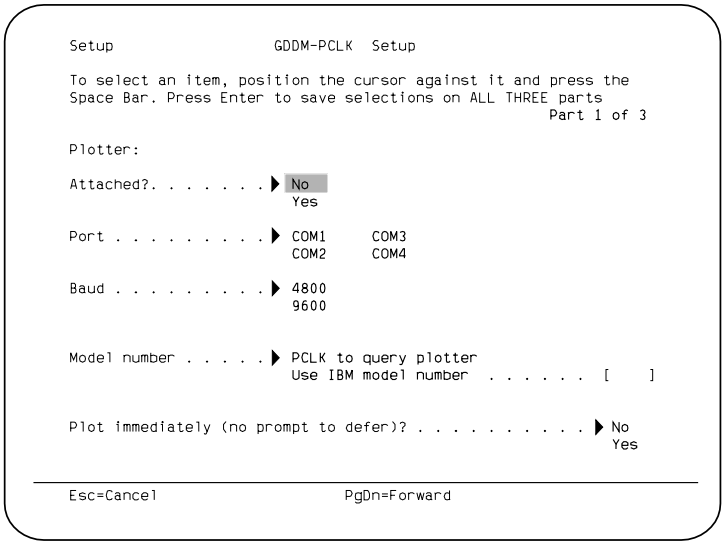
There are three parts to the Setup Panel, and two versions of part 1 of the panel, depending on the terminal emulator you are using. If your terminal emulator gives you more than one host session, part 1 of the panel lets you select which host session you want to use, and it looks like this:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 61. Setup panel: part 1 (with host-session selection)

If you are using a terminal emulator that lets you access only one host session, part 1 of the panel looks like this:



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 62. Setup panel: part 1 (without host-session selection)

The ► symbol indicates the current options. Do not press **ENTER** until you are sure that all three panels show the options that you want. Pressing **ENTER** saves the current options on the three panels.

On color screens, the cursor position is shown by a highlighted background. On monochrome screens, the option with the cursor is underscored. Use the cursor-move or tab keys to move to another option. The current options are indicated by a ► symbol. When you have moved to the option you want, press the

spacebar to make it current. For example, to tell GDDM-PCLK that you have a plotter attached to COM2, move the cursor to Yes and press the **spacebar**. A ► appears against COM1. Move the cursor to COM2, so that the background of COM2 is highlighted or underscored. Press the **spacebar**, and the ► moves to COM2 to show that it is now the current option.

Each question on this panel is now dealt with in turn:

Attached?
If you specify that a plotter is not attached, you cannot select a port for it.

Port
Select either COM1 or COM2. For an explanation of these, refer to the *DOS Reference* book, or to the *DOS User's Guide and Reference*.

Model number
If you intend using different plotters with your PC, select PCLK to query plotter and do not enter an IBM model number; otherwise, enter the IBM model number of the plotter (for example, 7372). Note that you can enter the model number for a plotter that you want to use for deferred plotting, even if you have specified that a plotter is not attached. This results in plot files being saved in a subdirectory with that model number as the name.

Plot immediately (no prompt to defer)?
If you intend always to plot immediately and do not want to be reminded about deferred plotting, select Yes for this question. But remember that when using merged mode, it is more efficient to defer your plotting.

If you press the **PgDn** key, part 2 of the panel lets you set up GDDM-PCLK for the type of printer you are using with your PC:

```
Setup                  GDDM-PCLK  Setup

To select an item, position the cursor against it and press the
Space Bar. Press Enter to save selections on ALL THREE parts
                                                        Part 2 of 3
Printer:

Model . .   None                        Device Name . . ▶ LPT1
            5182 Color Impact                             LPT2
            3852-1 Color Jetprinter                       LPT3
            3852-2 Color Jetprinter
            5152 Graphics Printer       Resolution  . . ▶ Standard
            4201 Proprinter                               High
            4201-2 Proprinter II
            4202 Proprinter XL          Form width  . . ▶ Standard
            4207 Proprinter X24                           Wide
            4208 Proprinter XL24
            5201 Quietwriter            Gray scale? . . . . ▶ No
            5202 Quietwriter III                             Yes
          ▶ 4019 LaserPrinter
          Print immediately (no prompt to defer)?  . . ▶ No
                                                          Yes


Esc=Cancel     PgUp=Backward   PgDn=Forward
```

Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 63. Setup panel: part 2

You can change the options in the same way as in part 1 of the panel. Each question on this panel is now dealt with in turn:

Model
Select None or a number from the list displayed.

If you do select a printer model of None, you cannot select a device name, resolution, form width, or defer option for it.

Device name
Select one of LPT1, LPT2, or LPT3. For an explanation of these, refer to the *DOS Reference* book, or to the *DOS User's Guide and Reference*.

Resolution
Select High to get a print with a maximum amount of detail. Note that printing takes longer in high-resolution mode.

Form width
Select Standard if the paper size in your printer is A4 (ISO) or quarto (ANSI size A).

Select Wide if the paper size in your printer is wider than A4 or quarto.

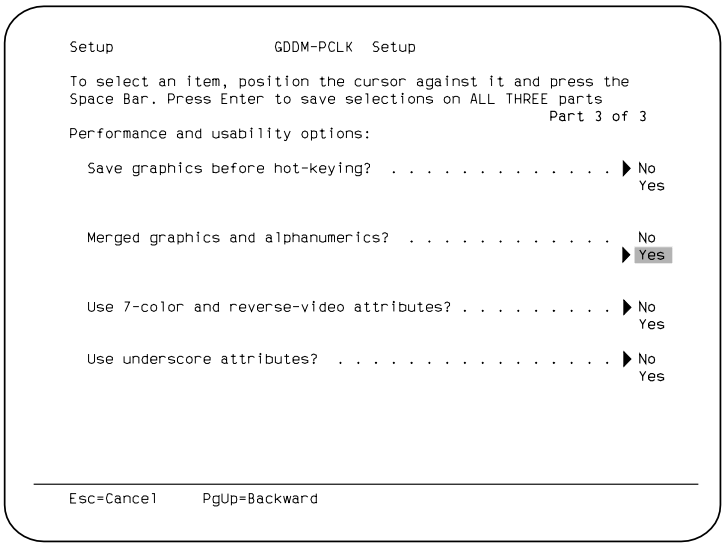
Print immediately (no prompt to defer)?
This has the same effect as "plot immediately"; see page 291.

If you intend always to print immediately and do not want to be reminded about deferred printing, select Yes for this question. But remember that when using merged mode, it is more efficient to defer your printing.

If you need to, you can press the **PgUp** key to return to part 1 of the setup panel.

If you press the **PgDn** key, part 3 of the panel lets you define performance and usability options for your plotted or printed output. You can define whether you want to:

- Save the graphics you have created before you hot-key.
- Merge alphanumerics and graphics.
- Use the default display attributes.



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 64. Setup panel: part 3

Each question on this panel is now dealt with in turn:

Save graphics before hot-keying?
If you select `Yes` and there is enough memory available, GDDM-PCLK saves the graphics when you press **Ctrl**+**F9** before hot-keying to the host session, and restores the graphics when you press **Ctrl**+**F9** again after hot-keying back from the host session.

Merged graphics and alphanumerics?
By default, GDDM-PCLK runs in merged graphics and alphanumerics mode, which means that you don't have to hot-key to see your graphics. However, in this mode you are recommended to defer any plotting or printing. We strongly advise that you do **not** run in non-merged mode.

Use 7-color and reverse-video attributes?
If you select `yes`, you can use these field and character attributes:

- Color (seven)
- Reverse-video
- Underlining.

If you select `yes` and your current emulator does not support the 3270 Extended Data Stream feature, your request is ignored.

If you do not select `7-color and reverse-video attributes` you can improve performance, and GDDM-PCLK needs less memory.

When you have completed part 1, part 2, and part 3 of the Set-up panel, press **ENTER** to save the changes and return to the main panel.

If you press **Esc**, you quit the panel. Any changes you made to part 1, part 2, or part 3 of the panel are ignored, and you return to the main panel.

You should now have completed setting up GDDM-PCLK for your preferred methods of processing, and your particular arrangement of PC-attached plotters and printers. You are now ready to use GDDM-PCLK for GDDM application support.

Creating PIF files

You can create picture interchange format (PIF) files through the User Control output panel.

Note: GDDM-PCLK cannot read, display, or print PIF files.

To create PIF files while GDDM User Control is active:

1. Select option 4 (Output) from the User Control Graphics panel displayed on your host screen.
2. When the Output panel is displayed, leave all the input fields unchanged and press **PF5** (save PIF).
3. When the file transfer panel appears (see Figure 65 on page 302), type in the path name, the file name, and select the overwrite option as required. Press **ENTER**.
4. You then see this message displayed:

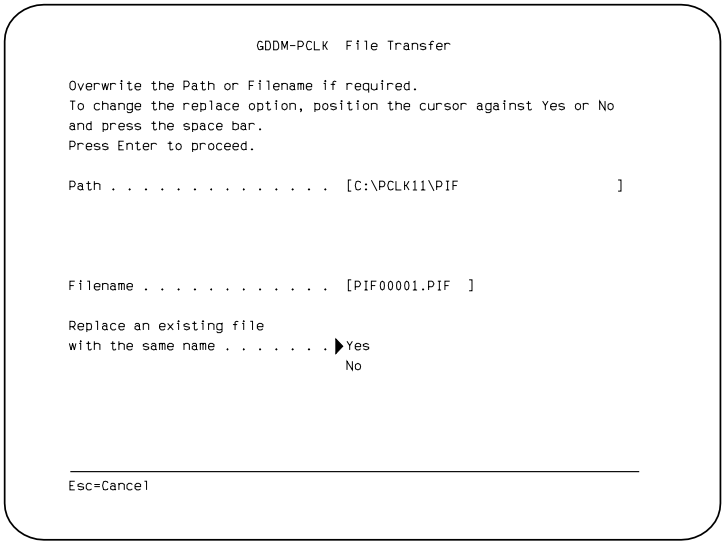
   ```
   GQD0030  File transfer in progress. Please wait.
            Number of bytes transferred so far = n
             (C:\PCLK11\PIF\PIF00001.PIF)
   ```

   where `n` is a running count of the number of bytes transferred.

5. Press **PF3** to return to the User Control graphics panel.

If you see messages that say the selection is invalid, check that you have this nickname statement in your nickname file (remember to precede it with a blank character):

```
ADMMNICK FAM=1,PROCOPT=((CTLSAVE,YES))
```



Note: The appearance of this picture depends on the characteristics of your display terminal.

Figure 65. PIF file transfer

Assigning keys in GDDM-PCLK

Note: With some terminal emulators, the key assignments in the GDDM-PCLK session do not match those in the emulator's host session.

A key-assignment utility (PCLKKEYS.EXE) is provided with GDDM-PCLK to enable you to reassign keys from the default layouts. To run the utility:

1. Make the \PCLK11 subdirectory current, and type:

   ```
   PCLKKEYS
   ```

   The standard GDDM-PCLK logo panel is displayed.

2. Press any key to continue.

3. The Keyboard Remap Utility main panel is displayed. This has a table showing the current assignments for your keyboard, from which you can select the assignment you want to change. You can reassign keys only if they do not currently have a function assigned to them. If you want to reassign a key that is already assigned, you must first deassign that key.

   To deassign a key:

   a. Use the cursor-up (▲) and cursor-down (▼) keys to select the assignment.
   b. Press **F2**.

4. Use the cursor-up and cursor-down keys to select the function that you want to reassign.

5. Press the **ENTER** key. The list of keys to which it is possible to assign that function is then displayed.
6. Use the cursor-up and cursor-down keys to select the key to which you want to assign that function. If you do not want that function to be assigned to any key, select `unassigned`.
7. Press the **ENTER** key to confirm your selection, or press **Esc** to leave the current assignment unchanged. The utility then returns to the main panel.
8. When you have done all your reassignments, press **F3** to save your changes and exit, or press **Esc** to quit the key-assignment utility without making any changes.

To restore the standard GDDM-PCLK assignments, press **F5** when the Keyboard Remap Utility main panel is displayed (step 3 on page 302 in the procedure).

Running GDDM-PCLK on a network

GDDM-PCLK can be used with the Token Ring adapter and the IBM Local Area Network (LAN) Program.

You should note the following points:

- GDDM-PCLK is not a multi-user program; that is, there can be only one user for each installed version of GDDM-PCLK.
- Other network users who want to use GDDM-PCLK must install a version of GDDM-PCLK in its own subdirectory.
- In LAN gateway mode, it is recommended that all users have their own copy of the emulator program. It can be stored on a shared disk; it need not be on a separate disk.

For more detailed information, see the *IBM PC Local Area Network Program: User's Guide*.

Common problems with GDDM-PCLK

There are three common problems that you may encounter when using GDDM-PCLK:

- The graphics cannot be displayed.
- The graphics look wrong.
- The disk becomes full during data transfer.

Each of these is discussed separately in this section.

The graphics cannot be displayed

If you see this message:

```
ADM0275 W  GRAPHICS {(IMAGE)} CANNOT BE SHOWN. REASON CODE n
```

it means that the construction of the graphics or image has been suppressed, and the area of the display where the graphics or image should appear remains blank.

This problem may have been caused by one of the following:

- You have not specified the GDDM-PCLK procopt to GDDM.
- You have specified the GDDM-PCLK procopt to GDDM, but you have also specified one of the CICS procopts (BMSCOORD or PSCNVCTL). GDDM-PCLK is not compatible with these CICS procopts.
- You pressed **ENTER** instead of hot-keying to the personal computer system session and starting the GDDM-PCLK program, when the personal computer system was opened by the host application program. (You can find out more about this problem in the description of message `ADM0873` in the *GDDM Messages* book.

The reason code (`n`) depends on the terminal emulator that you are using.

How to recover

1. Terminate your application.
2. Ensure that the GDDM-PCLK procopts have been set up correctly.
3. Restart your application.
4. Do not press **ENTER** to get to the personal computer system session. Instead, use the hot-key function.

Note: The message shown above may be displayed for several other reasons. See the *GDDM Messages* book for more details.

The graphics look wrong

There are two kinds of problems described in this section.

Black rectangle on the screen

If your application deletes text that has been drawn on top of graphics, a black rectangle may be left where the text was originally drawn.

For some display adapters, GDDM-PCLK holds graphics and text in the same part of memory. If some text is deleted, the graphics from the same area of the PC screen are also lost, which leaves a black rectangle on the screen.

How to recover

You must direct the application to redraw the graphics. To do this, press the **Clear** key, or **Ctrl**+**F5**.

Graphics corrupted or colors incorrect

Problems that can occur are:

- The graphics screen is corrupted by random pixels or blocks of color.
- The graphics are white and shown only as regularly spaced vertical slices.
- The colors are incorrect.

When using a terminal emulator, you have hot-keyed to the host session without first pressing **Ctrl**+**F9**. The PC screen must be clear of graphics before you hot-key to the host session.

How to recover

There are two ways of recovering from this problem. You can:

1. Hot-key to the PC session (if you are not already there).
2. Press **Ctrl**+**F9** to clear the graphics.
3. Hot-key to the host session.
4. Redisplay the graphics from the host session. To do this, you may simply need to press the **Clear** key.

Alternatively, you can:

1. Hot-key to the PC session (if you are not already there).
2. Press **Ctrl**+**F9** to clear the graphics.
3. Press **Ctrl**+**F9** to return to the graphics.
4. Stay in the PC session, and redisplay the graphics from the host program. Use the GDDM-PCLK-supplied host key equivalents as required by the host program. To redisplay the graphics, you may simply need to press **Alt**+**F2** (Clear).

The disk becomes full during automatic data transfer

While GDDM-PCLK is being downloaded automatically from the host to the PC, you may see the following message:

```
GQD0410  Disk full
```

This happens because the automatic download process does not delete the old version of GDDM-PCLK until the new version has been downloaded. This does not affect a high-density diskette, but it can cause a low-density diskette to become full before the process is complete.

How to recover

You must release as much space as possible (for example, by deleting unwanted plot or print files). Delete space from the GDDM-PCLK panels, or by entering DOS commands in the DOS command area. After releasing the space, you must start GDDM-PCLK again.

If you find that, even after you have deleted space, the diskette is still becoming full, try the following:

1. Copy the file `pclk.exe` from the \PCLK11 subdirectory to another drive.
2. Delete `pclk.exe` from the \PCLK11 subdirectory.
3. With the drive that contains the \PCLK11 subdirectory as the default drive, run `pclk.exe` in the other drive.
4. Choose option 1 from the GDDM-PCLK main menu. A new version of `pclk.exe` is automatically downloaded from the host to the \PCLK11 subdirectory.
5. Delete the copy of `pclk.exe` that you made in step 1.

Printing and plotting with GDDM-PCLK

GDDM-PCLK enables you to print output generated by a GDDM program running on the host directly to a print device attached to your workstation. You also have the choice of deferring printing to another time, which means that the output file is stored for later printing. You cannot plot alphanumerics using GDDM-PCLK.

Immediate and deferred plotting

To plot immediately, select “Yes” on part 1 of the setup menu in reply to the prompt:

```
Plot immediately (no prompt to defer)?
```

For immediate plotting, the complexity of a plot can almost be unlimited. For deferred plotting, the complexity is limited by the amount of disk space available at the time of plotting.

Immediate and deferred printing

To print immediately, select “Yes” on part 2 of the setup menu in reply to the prompt:

```
Print immediately (no prompt to defer)?
```

This avoids your being reminded about deferred printing. If you are in *merged mode*, you are recommended to select “No” in reply to this prompt.

Printing or plotting files stored on a disk

To print or plot a file that has been deferred (and therefore is stored on a disk), select option 3 from the GDDM-PCLK main panel. This panel shows only those deferred files that were created for the plotter or printer specified on the Setup panel. You can also delete print or plot files from this panel.

While each file is being printed, you receive a message to this effect. To cancel a print request, press the **Esc** key or its equivalent for your keyboard.

Appendix A. External defaults

General-use programming interface

This appendix begins with a summary list of external defaults. For each external default, the following information is provided:

- The source syntax of the ADMMDFT statement
- A brief description of the external default
- The GDDM default setting (if any)
- The encoded version of the ADMMDFT statement (as specified on the ESEUDS and SPINIT calls)
- Whether the external default can be specified in the external defaults module ("M"), in the external defaults file ("F"), on the SPINIT call ("S") and on the ESSUDS and ESEUDS calls ("C")
- The subsystems under which the external-default is valid

A more detailed description of each external default is provided, in alphabetical order of external default, on page 314. More information about the external defaults mechanism is provided in Chapter 17, "GDDM user-default specifications" on page 197.

External defaults: summary list

The external defaults are listed here in alphabetic order of keyword.

Table 41 (Page 1 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
—	No operation	—	{0|1}	Y N Y Y	All
ABNDRET={NO|YES}	Abend-return processing	NO	3,3,{0|1}	N N Y N	VM
AM3270=({LOCREM|REMOTE|LOCAL},{SNANOSNA|NONSNA|SNA})	Device attachment	LOCREM SNANOSNA	4,12,{0|1|2|},{0|1|2}	Y Y Y Y	All
APPCPG=n	Application code page	00351	3,125,n	Y Y Y Y	All
AUNLOCK={NO|YES}	Always-unlock-keyboard	NO	3,10,{0|1}	Y Y Y Y	All
CALLINF=(len,addr)	Call information feedback block: length, address	0,0 (none)	4,1101,L(CIB),A(CIB)	N N Y N	All
CECPINP={YES|NO}	CECP keyboard input	YES	3,126,{1|0}	Y Y N N	All
CICAUD=(stg-addr,pgm-addr)	Audit trail anchor block addresses: storage, program	0,0 (none)	4,1201,A(STGANCH),A(PGMANCH)	N N Y N	CICS
CICDECK=aaaa	Deck output transient data name	ADMD	3,202,aaaa	Y Y Y N	CICS
CICDFPX=aaaa	Defaults file temporary storage prefix	ADMD	3,210,aaaa	Y N Y N	CICS
CICGIMP=aaaaaaaa	GDDM-IMD ADMGIMP file-control name	ADMGIMP	4,203,aaaa,aaaa	Y Y N N	CICS
CICIADS=aaaa	GDDM-IMD ADS output transient data name	ADMG	3,207,aaaa	Y Y N N	CICS
CICIFMT=aaaaaaaa	GDDM-IMD staged data file-type	ADMIFMT	4,208,aaaa,aaaa	Y Y N N	CICS
CICPRNT=aaaa	Print Utility transaction name	ADMP	3,205,aaaa	Y Y Y N	CICS
CICSTGF=aaaaaaaa	GDDM-IMD staging file file-control name	ADMX	4,209,aaaa,aaaa	Y Y N N	CICS
CICSYSP=aaaa	System printer output transient data name	ADMS	3,206,aaaa	Y Y Y N	CICS
CICTIF={NO|YES|EXT}	Transaction independence	NO	3,14,{0|1|2}	N N Y N	CICS
CICTQRY=aaaa	CICS device query temporary storage prefix	ADMQ	3,211,aaaa	Y Y Y Y	CICS
CICTRCE=aaaa	Trace output transient data name	ADMT	3,201,aaaa	Y Y Y N	CICS
CICTSPX=aaaa	Print Utility temporary storage prefix	ADMT	3,204,aaaa	Y Y Y N	CICS
CMSAPLF={DATAANAL|APLTEXT}	APL default specification	APLTEXT	3,15,{0|1}	Y Y Y Y	VM
CMSCOLM=aaaaaaaa	Page-printer output filetype for color masters	ADMCOL+	4,510,aaaa,aaaa	Y Y Y N	VM
CMSCPT=aaaaaaaa	CGM conversion profile filetype	ADMCGM	4,512,aaaa,aaaa	Y Y Y Y	VM
CMSDECK=aaaaaaaa	Deck output filetype	ADMDECK	4,503,aaaa,aaaa	Y Y Y N	VM

Table 41 (Page 2 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
CMSDFTS=(aaaaaaaa, bbbbbbbb)	Defaults file: filename, filetype	PROFILE ADMDEFS	6,511,aaaa,aaaa, bbbb,bbbb	Y N Y N	VM
CMSIADS=aaaaaaaa	GDDM-IMD ADS output filetype	COPY	4,506,aaaa,aaaa	Y Y N N	VM
CMSIFMT=aaaaaaaa	GDDM-IMD Export data filetype	ADMIFMT	4,507,aaaa,aaaa	Y Y N N	VM
CMSMONO=aaaaaaaa	AFPDS and HRIG: monochrome filetype	ADMIMAGE	4,509,aaaa,aaaa	Y Y Y N	VM
CMSMSLT=aaaaaaaa	GDDM-IMD MSL filetype	ADMMSL	4,508,aaaa,aaaa	Y Y N N	VM
CMSPRNT=aaaaaaaa	Queued printer output filetype	ADMPRINT	4,504,aaaa,aaaa	Y Y Y N	VM
CMSSYSP=aaaaaaaa	System printer output filetype	ADMLIST	4,505,aaaa,aaaa	Y Y Y N	VM
CMSTEMP=aaaaaaaa	Work-file filetype	ADMUT1	4,501,aaaa,aaaa	Y Y Y N	VM
CMSTRCE=(aaaaaaaa, bbbbbbbb)	Trace output: filename, filetype	ADM00001 ADMTRACE	6,502,aaaa,aaaa, bbbb,bbbb	Y Y Y N	VM
COMMENT=(cccccccc, ccccccccc,.....)	Comments for module identification	N/A	1-8000,0,cccc, cccc,....	Y Y Y Y	All
CPN4250=aaaaaaaa	4250 code-page name	AFTC0395	4,109,aaaa,aaaa	Y Y Y Y	TSO, VM
CTLSAVE={YES|NO}	User Control SAVE function control	YES (NO on CICS)	3,119,{0|1}	Y Y Y Y	CICS, VM, TSO
DATEFRM={1|2|3|4}	Date convention	4	3,5,{1|2|3|4}	Y Y Y Y	All
DATRN=addr	Alphanumeric defaults module control	ADMDATRN	3,118,addr	Y N Y Y	All
DBCSDFT={GDDM|NO|YES}	DBCS default selection	GDDM	3,18,{0|1|2}	Y Y Y Y	All
DBCSDNM=(aaaaaa, bbbbbb)	DBCS default symbol-set name	ADMIK ADMVK	6,128,aaaa,aabb, bbbb,bbbb	Y Y Y Y	All
DBCSLIM=n	DBCS symbol set component in-core threshold	4	3,113,n	Y Y Y Y	All
DBCSLNG=c	DBCS symbol set language	K	3,111,X'xx000000'	Y Y Y Y	All
DFTXTNA=aaaaaaaa	Label on first ADMMDFTX macro	—	4,123,aaaa,aaaa	Y Y Y Y	VSE
ERRFDBK=(GDDMDFLT)	Error exit: use GDDM-supplied feedback block	GDDMDFLT	3,1102,0	Y N Y Y	All
ERRFDBK=(USERAREA, addr,len)	Error exit: use user-supplied feedback block	—	5,1102,2,addr,len	Y N Y Y	All
ERRTHRS=n	Error threshold value	4	3,101,n	Y Y Y Y	All
FF3270P={NO|AFTER|BEFORE|BOTH}	Form feed	AFTER	3,11,{0|1|2|3}	Y Y Y Y	All
FRCEVAL={NO|YES}	Force validation of HPA	NO	3,127,{0|1}	N Y N N	All
ICUFMDF={0|1|2}	ICU format	0	3,121,{0|1|2}	Y Y Y Y	All

Table 41 (Page 3 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
ICUFMSS={0|1|2}	ICU symbol sets	0	3,122,{0|1|2}	Y Y Y Y	All
ICUISOL={0|1|2}	ICU isolate value	0	3,112,{0|1|2}	Y Y Y Y	All
ICUPANC={TURQUOISE|BLUE}	ICU panel color	TURQ	3,120,{5|1}	Y Y Y Y	All
IMSDECK=aaaaaaaa	Deck output LTERM name	ADMDECK	4,302,aaaa,aaaa	Y N Y N	IMS
IMSEXIT=aaaaaaaa	Interactive Utility exit character string	EXIT	4,311,aaaa,aaaa	Y N N N	IMS
IMSICU=aaaaaaaa	Transaction name for Interactive Chart Utility	CHART	4,306,aaaa,aaaa	Y N N N	IMS
IMSISE=aaaaaaaa	Transaction name for Image Symbol Editor	ISSE	4,304,aaaa,aaaa	Y N N N	IMS
IMSMAST=aaaaaaaa	Interactive Utility shutdown LTERM name	MASTER	4,313,aaaa,aaaa	Y N N N	IMS
IMSMODN=aaaaaaaa	GDDM message output descriptor (MOD) name	DFS.EDT	4,317,aaaa,aaaa	Y N Y Y	IMS
IMSPRNT=aaaaaaaa	Print Utility transaction name	ADMPRINT	4,303,aaaa,aaaa	Y N Y N	IMS
IMSSDBD=aaaaaaaa	GDDM system definition database DBD name	ADMSYSDF	4,307,aaaa,aaaa	Y N Y N	IMS
IMSSEGS=(aaaaaaa, bbbbbbbb,....)	Segment/Key field names: Object database root segment Object database dependent segment Object database root key field Object database dependent key field System definition database segment System definition database key field	 ADMOBROO ADMOBDEP ADMOBRKY ADMOBDKY ADMSDSGM ADMSDKEY	14,308, aaaa,aaaa, bbbb,bbbb, cccc,cccc, dddd,dddd, eeee,eeee, ffff,ffff	Y N Y N	IMS
IMSSHUT=aaaaaaaa	Interactive Utility shutdown string	SHUTDOWN	4,312,aaaa,aaaa	Y N N N	IMS
IMSSYSP=aaaaaaaa	System printer output destination name	ADMLIST	4,314,aaaa,aaaa	Y N Y N	IMS
IMSTRCE=aaaaaaaa	Trace output ddname	ADMTRACE	4,301,aaaa,aaaa	Y N Y N	IMS
IMSUISZ=n	Input area size	3000	3,310,n	Y N N N	IMS
IMSUMAX=n	Maximum number of users	5	3,309,n	Y N N N	IMS
IMSVSE=aaaaaaaa	Transaction name for Vector Symbol Editor	VSSE	4,305,aaaa,aaaa	Y N N N	IMS
IMSWTOD=(n,n,n,....)	Write-to-operator descriptor codes	(7)	3,316,X'xxxx0000'	Y N Y N	IMS
IMSWTOR=(n,n,n,....)	Write-to-operator routing codes	(2)	3,315,X'xxxx0000'	Y N Y N	IMS
INSCPG=n	Installation code page	00037	3,124,n	Y N N N	All
IOBFSZ=n	Transmission buffer size	1536	3,104,n	Y Y Y Y	All
IOCOMPR={NO|YES}	Compressed PS loads	YES	3,9,{0|1}	Y Y Y Y	All
IOSYNCH={NO|YES}	Synchronized I/O	NO	3,8,{0|1}	Y Y Y Y	CICS, TSO

Table 41 (Page 4 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
MAPGSTG=n	Mapgroup storage threshold	8192	3,106,n	Y Y Y Y	All
MIXSOSI={NO|YES}	DBCS strings with shift-out/shift-in	NO	3,17,{0|1}	Y Y Y Y	All
NATLANG=c	National language	A	3,4,X'xx000000'	Y Y Y N	All
NUMBFRM={1|2|3}	Number convention	1	3,7,{1|2|3}	Y Y Y Y	All
OBJFILE=(aaaaaaaa, bbbbbbbb,....)	VSAM data-set names for:		4-24,107,	Y Y Y Y	CICS
	Symbol sets	ADMF	aaaa,aaaa,		
	Generated mapgroups	ADMF	bbbb,bbbb,		
	Saved pictures	ADMF	cccc,cccc,		
	Chart formats	ADMF	dddd,dddd,		
	Chart data	ADMF	eeee,eeee,		
	GDDM-IMD tutorial pages	ADMGIMP	ffff,ffff,		
	GDF files	ADMF	gggg,gggg,		
	(reserved)	–	hhhh,hhhh,		
	(reserved)	–	iiii,iiii,		
	Projection definitions	ADMF	jjjj,jjjj,		
	Image data	ADMF	kkkk,kkkk,		
	Reserved for GDDM-PCLK and GDDM-OS/2 Link files	ADMF	llll,llll		
OBJFILE=(aaaaaaaa, bbbbbbbb,....)	Database DBD names for:		4-24,107,	Y N Y N	IMS
	Symbol sets	ADMOBJ1	aaaa,aaaa,		
	Generated mapgroups	ADMOBJ1	bbbb,bbbb,		
	Saved pictures	ADMOBJ1	cccc,cccc,		
	Chart formats	ADMOBJ1	dddd,dddd,		
	Chart data	ADMOBJ1	eeee,eeee,		
	(reserved)	—	ffff,ffff,		
	GDF files	ADMOBJ1	gggg,gggg,		
	(reserved)	—	hhhh,hhhh,		
	(reserved)	—	iiii,iiii,		
	Projection definition	ADMOBJ1	jjjj,jjjj,		
	Image data	ADMOBJ1	kkkk,kkkk,		
	(reserved)	—	llll,llll		

Table 41 (Page 5 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
OBJFILE=(aaaaaaaa, bbbbbbbb,....)	ddnames for:		4-24,107,	Y Y Y Y	TSO
	Symbol sets	ADMSYMBL	aaaa,aaaa,		
	Generated mapgroups	ADMGGMAP	bbbb,bbbb,		
	Saved pictures	ADMSAVE	cccc,cccc,		
	Chart formats	ADMCFORM	dddd,dddd,		
	Chart data	ADMCDATA	eeee,eeee,		
	GDDM-IMD tutorial pages	ADMGIMP	ffff,ffff,		
	GDF files	ADMGDF	gggg,gggg,		
	Reserved for GDDM-GKS metafiles	—	hhhh,hhhh,		
	Chart data definition	ADMCDEF	iiii,iiii,		
	Projection definition	ADMPROJ	jjjj,jjjj,		
	Image data	ADMIMG	kkkk,kkkk,		
	Reserved for GDDM-PCLK and GDDM-OS/2 Link files	ADMPC	llll,llll		
OBJFILE=(aaaaaaaa, bbbbbbbb,....)	VSAM data-set names for:		4-24,107,	Y Y Y Y	VSE
	Symbol sets	ADMF	aaaa,aaaa,		
	Generated mapgroups	ADMF	bbbb,bbbb,		
	Saved pictures	ADMF	cccc,cccc,		
	Chart formats	ADMF	dddd,dddd,		
	Chart data	ADMF	eeee,eeee,		
	GDDM-IMD tutorial pages	ADMGIMP	ffff,ffff,		
	GDF files	ADMF	gggg,gggg,		
	(reserved)	—	hhhh,hhhh,		
	(reserved)	—	iiii,iiii,		
	Projection definitions	ADMF	jjjj,jjjj,		
	Image data	ADMF	kkkk,kkkk,		
	Reserved for GDDM-PCLK and GDDM-OS/2 Link files	ADMF	llll,llll		
OBJFILE=(aaaaaaaa, bbbbbbbb,....)	Filetypes for:		4-24,107,	Y Y Y Y	VM
	Symbol sets	ADMSYMBL	aaaa,aaaa,		
	Generated mapgroups	ADMGGMAP	bbbb,bbbb,		
	Saved pictures	ADMSAVE	cccc,cccc,		
	Chart formats	ADMCFORM	dddd,dddd,		
	Chart data	ADMCDATA	eeee,eeee,		
	GDDM-IMD tutorial pages	ADMTUTPG	ffff,ffff,		
	GDF files	ADMGDF	gggg,gggg,		
	(reserved)	—	hhhh,hhhh,		
	Chart data definition	ADMCDEF	iiii,iiii,		
	Projection definition	ADMPROJ	jjjj,jjjj,		
	Image data	ADMIMG	kkkk,kkkk,		
	Reserved for GDDM-PCLK and GDDM-OS/2 Link files	ADMPC	llll,llll		
PARMVER={NO|YES}	Parameter verification (SPI)	NO	3,1,{0|1}	N N Y N	All
SAVBFSZ=n	FSSAVE buffer size	1024	3,105,n	Y Y Y Y	All
SOSIEMC=c	DBCS SO/SI emulation character	”	3,110,X'xx000000'	Y Y Y Y	All
STGRET={NO|YES}	Short-on-storage processing	NO	3,2,{0|1}	N N Y N	All
TIMEFRM={1|2|3|4}	Time convention	1	3,6,{1|2|3|4}	Y Y Y Y	All
TRACE={0|n}	Trace word value	0	3,102,n	Y Y Y Y	All
TRCESHR={NO|YES}	Trace share	NO	3,117,{0|1}	Y Y Y Y	TSO, VM

Table 41 (Page 6 of 6). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
TRCESTR= 'xxxxxx...	Trace control	(none)	3-8000,114,xxxx, xx....	Y Y Y Y	All
TRCEWID= {SINGLE|DOUBLE}	Trace output width	SINGLE	3,115,{0|1}	Y Y Y Y	All
TRTABLE=n	Trace table size, in-core	100	3,103,n	Y Y Y N	All
TSOAPLF= {DATAANAL| APLTEXT}	APL default specification	DATAANAL	3,16,{0|1}	Y Y Y Y	TSO
TSOCOLM= aaaaaaaa	High-resolution image generation: color ddname or high-level qualifier	ADMCOL+	4,409,aaaa,aaaa	Y Y Y N	TSO
TSOCPT=aaaaaaaa	CGM conversion profile ddname	ADMCGM	4,414,aaaa,aaaa	Y Y Y Y	TSO
TSODECK= aaaaaaaa	Deck output ddname	ADMDECK	4,402,aaaa,aaaa	Y Y Y N	TSO
TSODFTS= aaaaaaaa	Defaults file ddname	ADMDEFS	4,411,aaaa,aaaa	Y N Y N	TSO
TSOEMUL= {NO|YES}	TSO Emulation	NO	2,413	Y Y Y Y	TSO
TSOGIMP=aaaaaaaa	GDDM-IMD ADMGIMP ddname	ADMGIMP	4,403,aaaa,aaaa	Y Y N N	TSO
TSOIADS=aaaaaaaa	GDDM-IMD ADS output ddname	ADMGNADS	4,406,aaaa,aaaa	Y Y N N	TSO
TSOIFMT=aaaaaaaa	GDDM-IMD export data ddname	ADMIFMT	4,407,aaaa,aaaa	Y Y N N	TSO
TSOMONO= aaaaaaaa	Page-printer output ddname or data set name low-level qualifier	ADMIMAGE	4,408,aaaa,aaaa	Y Y Y N	TSO
TSOPRNT=aaaaaaaa	Print data-set qualifier	ADMPRINT	4,404,aaaa,aaaa	Y Y Y N	TSO
TSORESV={NO|YES}	Reserve master print queue DASD	NO	3,415,{0|1}	Y N N N	TSO
TSOSYSP= aaaaaaaa	System printer output ddname	ADMLIST	4,405,aaaa,aaaa	Y Y Y N	TSO
TSOS99S=n	Dynamic allocation size	742710	3,410,n	Y Y Y Y	TSO
TSOS99U= aaaaaaaa	Dynamic allocation unit specification	SYSDA	4,412,aaaa,aaaa	Y Y Y Y	TSO
TSOTRCE= aaaaaaaa	Trace output ddname	ADMTRACE	4,401,aaaa,aaaa	Y Y Y N	TSO
VSECOLM= aaaaaaaa	Page printer files for image generation: color file name	ADMCOL+	4,603,aaaa,aaaa	Y Y Y Y	VSE
VSEDFTS= aaaaaaaa	Defaults file name	SYSIPT	4,604,aaaa,aaaa	Y N Y N	VSE
VSEMONO= aaaaaaaa	Page printer files for image generation: monochrome file name	ADMIMAGE	4,602,aaaa,aaaa	Y Y Y Y	VSE
VSETRCE= aaaaaaaa	Trace file name	ADMTRCE	4,601,aaaa,aaaa	Y Y Y N	VSE

External defaults: full descriptions

This part of the appendix lists the GDDM external defaults and their values in alphabetic order of the external-default description parameter. For example, for a description of the “always-unlock-keyboard” external default, look for “AUNLOCK” in this list.

When GDDM is first installed, some of these external defaults have initial settings. Such settings are underlined in the following list.

Note: Unless otherwise stated, where an operand is defined as a 4- or 8-character string, it can be specified as a shorter value, in which case the string is left-justified and padded with blanks to 4 or 8 characters.

ABNDRET={NO|YES}
Subsystem: VM only**.**

Defines whether, in case of a controlled abnormal-end (abend) condition, GDDM should return control to the application program immediately with a corresponding error code and message. The message includes the abend code that GDDM would otherwise have issued.

This external default can cause GDDM to return control to the application only in controlled abend situations. It does not enable return from uncontrolled abends, such as program checks and abends issued by underlying subsystem services. Also, as an abend can indicate a major internal error, successful return to the application cannot be guaranteed.

GDDM does not try to correct the abend situation or to release resources before returning to the application. Successful continuation of the GDDM session after return cannot be ensured.

AM3270={{LOCAL|REMOTE|LOCREM},{SNA|NONSNA|SNANOSNA})
Subsystem: All**.**

Defines the attachment mode of 3270 devices. Such devices can be defined as locally attached (LOCAL), remotely attached (REMOTE), or a mixture of both (LOCREM). Similarly, all 3270 devices can be SNA devices, non-SNA devices, or a mixture of both.

This external default specifies device characteristics that GDDM may not otherwise be able to deduce, and allows GDDM to optimize its device processing.

If GDDM can deduce that all devices are locally attached, it does not usually generate “compressed PS load” data streams, even if the device shows that it supports compression and even if the IOCOMPR=YES external default has been specified.

If GDDM can deduce that all devices are either locally-attached or SNA, it does not constrain “PS load” data streams to conform to the 3KB transmission limit required for remote non-SNA devices.

APPCPG=n
Subsystem: All**.**

Defines the code-page to be used by GDDM applications. It applies to names (for example, window names, symbol set names, map names) and character strings on FSLOG and FSLOGC calls. It also applies to GDDM objects. For example, when a page is saved using GSSAVE, graphics strings in the resulting GDF file are coded according to the application code page.

This table lists the code pages supported by GDDM:

00037	U.S.A., Canada, Portugal[1], Netherlands, Brazil[2]
00273	Austria, Germany
00277	Denmark, Norway
00278	Finland, Sweden
00280	Italy
00281	Japan (Latin characters)
00284	Spain, Latin America
00285	United Kingdom, Ireland
00290	GDDM Katakana
00297	France
00351	GDDM default EBCDIC
00500	Multilingual page (MLP), Switzerland, Belgium[3]
00871	Iceland
01027	Japan (Latin) Extended
00870	Latin 2
00875	Greece
00880 and 1025	Cyrillic
00905 and 1026	Turkey
01112	Baltic multilingual
01122	Estonia

1 00037 has superseded 00282 for Portugal
2 00037 has superseded 00275 for Brazil
3 00500 has superseded 00274 for Belgium

AUNLOCK={NO|YES}
Subsystem: All.
Specifies whether GDDM is to operate in "always-unlock-keyboard mode." For more information, see the description of the AUNLOCK processing option in Appendix B, "Processing options" on page 333.

CALLINF=(length,address)
Subsystem: All.
Specifies two 4-byte fields containing the length and address of a call-information feedback block provided by the application program.

The area passed by the application must be at least eight bytes long. The first four bytes receive the address of the call formats descriptor module. The second four bytes receive the address of the APL request code module.

If either call-information module cannot be located, the 8-byte call information feedback block is set to binary zeros.

CECPINP={YES|NO}
Subsystem: All.
Specifies whether the full range of CECP code points is to be allowed in alphanumeric input data from the keyboard of a family-1 device.

CICAUD=(stg-addr,pgm-addr)
Subsystem: CICS.
Specifies two 4-byte fields, each containing the address of a 4-byte anchor by which GDDM locates a record of currently acquired storage resources and currently acquired program resources, respectively. For a full explanation of this processing, refer to the *GDDM Base Application Programming Guide*.

CICDECK=aaaa
Subsystem: CICS
A 4-character string that is the transient-data destination used by GDDM for object module output from the Image Symbol Editor or the GDDM-PGF Vector Symbol Editor.

CICDFPX=aaaa
Subsystem: CICS
A 4-character string containing the 4-byte prefix used by GDDM to determine the CICS Temporary Storage names used for external defaults files. This option is intended for use in problem determination only. For information on how to use it in that context, see the *GDDM Diagnosis* book.

CICGIMP=aaaaaaaa
Subsystem: CICS
An 8-character string that is the CICS File Control data-set name used by GDDM for retrieving the generated mapgroups required for the operation of GDDM-IMD.

CICIADS=aaaa
Subsystem: CICS
A 4-character string that is the external default Transient Data destination used by GDDM for the output of ADSs (application data structures) resulting from the use of GDDM-IMD.

CICIFMT=aaaaaaaa
Subsystem: CICS
An 8-character string that is a external default “file-type” assigned to data exported to a VSAM “staging” data set, as a result of using GDDM-IMD’s Export Utility.

CICPRNT=aaaa
Subsystem: CICS
A 4-character string that is the transaction name assigned to the GDDM CICS Print Utility; refer to the *GDDM Base Application Programming Guide*.

CICSTGF=aaaaaaaa
Subsystem: CICS**.**
An 8-character string that is the default CICS File Control data-set name of the VSAM “staging” data set to be used with GDDM-IMD.

CICSYSP=aaaa
Subsystem: CICS**.**
A 4-character string that is the default transient data destination used by GDDM for output resulting from system printers. Such devices are defined as described in the *GDDM Base Application Programming Guide*.

CICTIF={NO|YES|EXT}
Subsystem: CICS**.**
Shows whether GDDM is to use transaction-independent services. For a full description of this processing, see the *GDDM Base Application Programming Guide*.

The values are:

NO Transaction-independent services are not to be used.
YES GDDM storage is retained between transactions.

EXT All GDDM storage is allocated above 16MB, and is retained between transactions. All ASREAD calls are read-only operations.

Note: The EXT option requires CICS support for both SHARED and FLENGTH options in the EXEC CICS GETMAIN command.

CICTQRY=aaaa
Subsystem: CICS**.**
A 4-character string that is the prefix for the CICS temporary storage queue names used for saving device query information.

CICTRCE=aaaa
Subsystem: CICS**.**
A 4-character string that is the transient data destination used by GDDM for diagnostic trace output.

CICTSPX=aaaa
Subsystem: CICS**.**
A 4-character string that is the 4-byte prefix used by GDDM to construct CICS Temporary Storage names for passing data to the GDDM CICS Print Utility; refer to the *GDDM Base Application Programming Guide*.

CMSAPLF={DATAANAL|APLTEXT}
Subsystem: VM**.**
Identifies the APL feature installed on **nonqueriable** IBM 3270 printers.

DATAANAL GDDM is to assume that any APL feature installed on any IBM 3270 printer is the Data Analysis-APL feature, unless specific application program device-definition information shows otherwise. The Data Analysis-APL feature applies to such printers as the IBM 3284, 3286, and 3288.

APLTEXT GDDM is to assume that any APL feature installed on any IBM 3270 printer is the APL/Text feature, unless specific application program device-definition information shows otherwise. The APL/Text feature applies to such printers as the IBM 3287 and 3289.

CMSCOLM=aaaaaaaa
Subsystem: VM**.**
An 8-character string defining the default filetypes used by GDDM under VM for multicolored output resulting from high-resolution image devices.

The character string must contain a “+” substitution character.

CMSCPT=aaaaaaaa
Subsystem: VM**.**
An 8-character string defining the default filetype used by GDDM under VM for files containing Computer Graphics Metafile (CGM) conversion profiles.

CMSDECK=aaaaaaaa
Subsystem: VM**.**
An 8-character string that is the filetype used by GDDM under VM for object module output resulting from requests through the Image Symbol Editor or the GDDM-PGF Vector Symbol Editor.

CMSDFTS=(aaaaaaaa,bbbbbbbb)
Subsystem: VM.
Two 8-character strings that are the filename and filetype of the External Defaults File under VM.

CMSIADS=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for the output of ADSs (application data structures) resulting from the use of GDDM-IMD.

CMSIFMT=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for exporting data as a result of using GDDM-IMD's Export Utility.

CMSMONO=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for monochrome page-printer output.

CMSMSLT=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for GDDM-IMD map specification libraries (MSLs).

CMSPRNT=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for generating files to be printed by the GDDM VM Print Utility, ADMOPUV; refer to the *GDDM Base Application Programming Guide*.

CMSSYSP=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for disk file output resulting from system printers.

CMSTEMP=aaaaaaaa
Subsystem: VM.
An 8-character string that is the default filetype used by GDDM under VM for intermediate file operations.

CMSTRCE=(aaaaaaaa,bbbbbbbb)
Subsystem: VM.
Two 8-character strings that are the default filename and filetype used by GDDM under VM for trace output.

COMMENT=(cccccccc,cccccccc,.....)
Subsystem: All.
Specifies a comment as a list of strings of 8 or less nonblank characters, which are ignored by GDDM external default processing. The list must not contain more than 8000 such strings. This external default can be used to imbed a comment into an encoded UDSL for documentation purposes.

CPN4250=aaaaaaaa
Subsystem: All except IMS**.**

An 8-character string that is the system default code-page name used for an IBM 4250 printer. For a list of possible values, see the description of the GSCPG call in the *GDDM Base Application Programming Reference* book.

CTLSAVE={YES|NO}
Subsystem: Not IMS**.**

Shows whether GDDM is, by default, to allow the application to control the picture-saving facilities offered in the User Control environment.

The external default value varies according to the subsystem:

Under CICS it is NO
Under VM and TSO it is YES
Under IMS it is not available.

DATEFRM={1|2|3|4}
Subsystem: All**.**

The date convention to be used by GDDM and GDDM-PGF:

1 MM/DD/YYYY (US convention)
2 DD.MM.YYYY (European convention)
3 YYYY-MM-DD (ISO and Japanese convention)
4 DD MMM YYYY (MMM are the first 3 characters of the month name).

Note that GDDM-IMD always displays the date in an abbreviated form; that is, the first two digits of the year (YYYY) are omitted.

DATRN=addr
Subsystem: All**.**

Provides a means by which a program can pass to GDDM the address of an alphanumeric defaults module to be used instead of ADMDATRN.

DBCSDFT={GDDM|NO|YES}
Subsystem: All**.**

This external default, which has meaning only when the NATLANG external default specifies a double-byte-character-set (DBCS) language, introduces the concept of the default error message destination, and enables the user to control DBCS support for it. DBCSDFT allows the user to specify, or to ask GDDM to specify, whether the default error message destination can support DBCS languages. The default is that GDDM should determine this.

The values are:

GDDM GDDM must determine whether the device can support DBCS
NO The device cannot support DBCS
YES The device can support DBCS.

Some examples of default error message destinations are:

- The user screen (for TSO)
- Transaction-initiating terminals (for CICS and IMS)
- FSQERR destination
- FSEXIT destination.

DBCSDNM=(aaaaaa,bbbbbb)
Subsystem: All.

Specifies the prefixes of the GDDM-supplied DBCS default symbol sets to be loaded when GSCS(8) is called, or when the external default MIXSOSI=YES is specified. The first name (**aaaaaa**) is used for mode-2 (image symbol GSCM(2)) text. The second name (**bbbbbb**) is used for mode-3 (vector symbol GSCM(3)) text. Both mode-2 and mode-3 names must be supplied. The names must be valid DBCS symbol-set names as defined for the GSLSS call.

These symbol sets are loaded as required by GDDM when processing GSCHAP, GSCHAR, or GSQTB calls. The external default DBCSLIM=n specifies the limit on the number of wards that can be loaded concurrently.

The prefixes of the GDDM-supplied DBCS default symbol sets are as follows:

ADMIK Standard Kanji mode-2
ADMVK Standard Kanji mode-3
ADMVQ High-quality Kanji mode-3
ADMVC Standard Simplified Chinese mode-3

The following are examples of source format specifications:

```
ADMMDFT DBCSDNM=(ADMIK,ADMVK)
```

This gives standard Kanji mode-2 and mode-3 text. (These are the default values.)

```
ADMMDFT DBCSDNM=(ADMIK,ADMVC)
```

This gives standard Simplified Chinese mode-3 text. (Mode-2 text will use standard Kanji.)

When creating encoded GDDM user default specifications, each DBCS default symbol set name prefix must be padded with trailing blanks to be exactly eight characters long.

Note: If both the DBCSLNG and DBCSDNM external defaults are used, the one specified last takes precedence.

DBCSLIM=n
Subsystem: All.

An integer, in the range 1 through 16, that is the DBCS symbol-set component (ward) in-core threshold. This limit applies to each DBCS set loaded by a GSLSS call and to the default DBCS sets if these are loaded. GDDM usually optimizes DBCS symbol set functions by retaining loaded DBCS symbol set components (wards) in main storage up to the specified number of components for each DBCS symbol set.

DBCSLNG=c
Subsystem: All.

The DBCS symbol set used. The GDDM-supplied default is K for Kanji. If another language is used, the character chosen must be used in the symbol-set names as a replacement for K.

Note: This external default is obsolete and has been superseded by DBCSDNM. The use of this external default is not recommended. If both the DBCSLNG and DBCSDNM external defaults are used, the one specified last takes precedence.

DFTXTNA=aaaaaaaa
Subsystem: VSE**.**
The label on the first ADMMDFTX macro that defines the Job Control Language (JCL) to be used for batch printing.

ERRFDBK=(aaaaaaaa)
Subsystem: All**.**
Shows that an error feed-back block is used. The meaning of “aaaaaaaa” can be:

GDDMDFLT
Shows that the GDDM-supplied default error feed-back block is used. This external default can only be specified in encoded format and cannot, therefore, be specified in an ESSUDS call or in an External Defaults File.

USERAREA,addr,len
Shows that a user or application program-supplied error feed-back block is used. The arguments are the address and length of an error feed-back block provided by the application program. This external default can be specified only in encoded format and cannot, therefore, be specified in an ESSUDS call or in an External Defaults File.

If an application program error feedback block is located in this manner, GDDM’s default error exits do not send error messages to the user’s terminal device. Rather, these default error exits return error details in the application program error feedback block. The format of the information returned in the feedback block is defined in the *GDDM Base Application Programming Guide*. GDDM never clears this error feedback block; it is set only as a result of a GDDM default error exit being invoked.

Note that the ERRFDBK option establishes the **default** error action. The FSEXIT(0,n) call shows that the **default** error action is to be taken. FSEXIT(addr,n) shows that the FSEXIT-defined user error exit is to be used. A subsequent FSEXIT(0,n) restores the **default** error action.

ERRTHRS=n
Subsystem: All**.**
A nonnegative integer that is the default error-threshold value. This value has the same meaning as the error-severity value specified in the FSEXIT call. However, the specified threshold can have effect from the start of initialization.

The error threshold value can also be changed in the FSEXIT call.

FF3270P={NO|AFTER|BEFORE|BOTH}
Subsystem: All**.**
Shows whether GDDM, including the GDDM Print Utility, by external default, performs a form feed (page eject) at the start, end, or start **and** end of processing on an IBM 3270-family printer.

| It does not apply to cut-sheet devices (including IPDS cut-sheet printers).

FRCEVAL={NO|YES}
Subsystem: All**.**
Allows the user to control the validation of high-performance alphanumerics data.

For example, when a tested application (for example, a shipped licensed program that does not use validation), is suspected of a bug, validation can be

turned back on to determine whether the application or GDDM is at fault by specifying:

```
ADMMDFT FRCEVAL=YES
```

in the external defaults file. This external default cannot be specified in the external defaults module, on SPINIT calls, or by API call.

ICUFMDF={0|1|2}
Subsystem: All.
Allows the user to control the use of chart format defaults in the Interactive Chart Utility of GDDM-PGF. All applications on the system (new, old, or stand-alone ICU) have their chart format defaults controlled by this one parameter. The values that can be specified are:

0 Release-dependent ICU choice.

Allows the ICU to choose the chart format defaults – the actual defaults may change from one release of GDDM to the next. This value is usually the same as choosing "2" except when the ICU is invoked by CHART with FORMNAME=* and DISPLAY≠1 or ≠2; in this case ICUFMDF is set as if "1" had been chosen.

1 Use the chart format defaults as specified in GDDM Version 1 Release 4.

2 Use the chart format defaults as specified in GDDM Version 2 Release 1.

ICUFMSS={0|1|2}
Subsystem: All.
Specifies the external default use of symbol sets in formats value in the Interactive Chart Utility of GDDM-PGF.

The values that can be specified are:

0 Release-dependent ICU choice (same as 2).
1 Use an asterisk (*) for all symbol sets named in format defaults.
2 Use Vector Symbol Sets as named in the format defaults.

ICUISOL={0|1|2}
Subsystem: All.
Allows you to control users' access to the Save, Restore, and Directory panels of the GDDM-PGF Interactive Chart Utility (ICU). This value is inspected only if the chart-control parameter of the GDDM-PGF CHART call has the isolate value set to zero.

The values that can be specified are:

0 The Save, Restore, and Directory panels of the ICU are made available to the operator.

1 The Save, Restore, and Directory panels are not made available to the operator.

2 The Save and Restore panels are made available to the operator, but the Directory panel is not.

ICUPANC={TURQUOISE|BLUE}
Subsystem: All.
Specifies the default use of the basic panel color for the Interactive Chart Utility of GDDM-PGF.

The values that can be specified are:

TURQUOISE The default.
BLUE

IMSDECK=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the logical terminal name (LTERM) used by GDDM for object module output resulting from requests through the Image Symbol Editor or the GDDM-PGF Vector Symbol Editor.

IMSEXIT=aaaaaaaa
Subsystem: IMS**.**
An 8-character string used as a parameter to the GDDM interactive utility transaction to cause exit processing for all conversations from a particular LTERM.

IMSICU=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the transaction name for requesting the Interactive Chart Utility of GDDM-PGF.

IMSISE=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the transaction name for requesting the Image Symbol Editor.

IMSMAST=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the LTERM name of the only LTERM allowed to issue the shutdown request to the GDDM interactive utility transaction.

IMSMODN=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the message output descriptor (MOD) name used by GDDM for sending nonconversational messages to IBM 3270-family displays.

IMSPRNT=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the transaction name assigned to the GDDM IMS Print Utility.

IMSSDBD=aaaaaaaa
Subsystem: IMS**.**
An 8-character string that is the DBD name by which the GDDM system definition database is accessed.

IMSSEGS=(aaaaaaaa,bbbbbbbb,cccccccc,dddddddd,eeeeeeee,ffffffff)
Subsystem: IMS**.**
Six 8-character strings, which are the names of the IMS segments and key fields:

`aaaaaaaa`	object database root-segment name
`bbbbbbbb`	object database dependent segment name
`cccccccc`	object database root-segment key field name
`dddddddd`	object database dependent segment key field name
`eeeeeeee`	system definition database segment name
`ffffffff`	name of the key field in the above segment.

IMSSHUT=aaaaaaaa
Subsystem: IMS.
An 8-character string used as a parameter to the GDDM interactive utility transaction to cause immediate termination of the transaction.

IMSSYSP=aaaaaaaa
Subsystem: IMS.
An 8-character string that is the default destination for output from a system printer.

IMSTRCE=aaaaaaaa
Subsystem: IMS.
An 8-character string that is the ddname used by GDDM for trace output.

IMSUISZ=n
Subsystem: IMS.
An integer, in the range 1 through 32000, which is the size of the data area reserved to contain the MFS Bypass input to the GDDM interactive utility transaction.

IMSUMAX=n
Subsystem: IMS.
An integer, in the range 1 through 32765, which is the maximum number of concurrent conversations to be supported by the GDDM interactive utility transaction.

IMSVSE=aaaaaaaa
Subsystem: IMS.
An 8-character string that is the transaction name for requesting the GDDM-PGF Vector Symbol Editor.

IMSWTOD=(n,n,n,n,....)
Subsystem: IMS.
The descriptor codes for a write-to-operator (WTO) macro. This is used by GDDM to issue error messages if all other methods fail. For a description of valid descriptor codes, see the *OS/VS2 MVS Supervisor Services and Macro Instructions* manual.

In the encoded-UDS format, the external default value should be coded as X'xxxx 0000', in which bit n=1 (n=0 through 31) corresponds to descriptor code “n+1” being requested.

IMSWTOR=(n,n,n,n,...)
Subsystem: IMS.
The routing codes for a write-to-operator (WTO) macro. This is used by GDDM to issue error messages if all other methods fail. For a description of valid routing codes, see the *OS/VS2 MVS Supervisor Services and Macro Instructions* manual.

In the encoded-UDS format, the default value should be coded as X'xxxx 0000', in which bit n=1 (n=0 through 31) corresponds to routing code “n+1” being requested.

INSCPG=n
Subsystem: All.
The code-page to be used by GDDM as the default for the installation. It applies to all character data that is not explicitly tagged; for example, object names in auxiliary storage.

This table shows the code pages supported by GDDM:

00037	U.S.A., Canada, Portugal[1], Netherlands, Brazil[2]
00273	Austria, Germany
00277	Denmark, Norway
00278	Finland, Sweden
00280	Italy
00281	Japan (Latin characters)
00284	Spain, Latin America
00285	United Kingdom, Ireland
00290	GDDM Katakana
00297	France
00351	GDDM default EBCDIC
00500	Multilingual page (MLP), Switzerland, Belgium[3]
00871	Iceland
01027	Japan (Latin) Extended
00870	Latin 2
00875	Greece
00880 and 1025	Cyrillic
00905 and 1026	Turkey
01112	Baltic multilingual
01122	Estonia

1 00037 has superseded 00282 for Portugal
2 00037 has superseded 00275 for Brazil
3 00500 has superseded 00274 for Belgium

IOBFSZ=n
Subsystem: All**.**

An integer, in the range 1024 through 32000, which is the size of the transmission buffer used by GDDM for IBM 3270-family devices. GDDM splits **outbound** terminal transmissions to fit within this buffer size. Under IMS, this is the size of segments, excluding the LLZZ prefix, that are inserted into the Message Queue.

On a non-SNA connection, for an IBM 3179-G, an IBM 3192-G, an IBM 3472-G color display station, a 3270-PC/G, /GX, or /AT workstation, or a device supported by GDDM-PCLK or GDDM-OS/2 Link, the outbound transmission size is restricted to approximately 3500 bytes to avoid possible controller timeouts.

Inbound transmission sizes are regulated according to the system you are using:

Under CICS
Maximum **inbound** transmission size is regulated by CICS system generation (specifically, the Terminal I/O Area lengths defined in the Terminal Control Table (TCT)), and is not affected by the value of IOBFSZ.

Under IMS
User transactions cannot receive **input**; therefore, this field does not apply to input processing. The size of the input area allocated in the GDDM interactive utility transaction is defined in the IMSUISZ external default.

Under TSO
The maximum **inbound** transmission size is regulated by TSO and VTAM system and network definition. Within this bound, IOBFSZ determines the size

of an individual work buffer but does not otherwise affect or limit inbound transmission processing.

Under VM
IOBFSZ determines the **default inbound** transmission buffer size used by GDDM. GDDM acquires temporary buffers of 32000 bytes for larger inbound terminal data streams (resulting from 3270 READ MODIFIED commands). IOBFSZ should not be greater than the RSCS buffer size if RSCS is used with 3287 or 4224 printers.

IOCOMPR={NO|YES}
Subsystem: All.
Shows whether GDDM is to create compressed PS load data streams. See also the description of the AM3270 external default on page 314.

Some IBM 3270-series terminals optionally support compression of programmed symbol (PS) data streams. If such compression is to be inhibited, it is generally recommended that this be done on a specific basis through device-configuration parameters. However, the IOCOMPR external default can be used to inhibit compression, on a global basis, of all PS load data streams generated by GDDM.

IOSYNCH={NO|YES}
Subsystem: CICS, TSO.
Shows whether GDDM is to perform synchronized terminal I/O. Usually, the use of synchronized terminal I/O implies longer transmission times and increased processing overhead. It may be useful to prevent jamming a network with large data streams used for graphics. In this context, this control might be used with a smaller value of IOBFSZ and SAVBFSZ.

The meaning of synchronized terminal I/O differs according to the subsystem in use:

Under CICS
Each GDDM outbound terminal transmission which expects input to be received, specifies “definite,” requiring that the terminal returns a definite response, where applicable, before GDDM continues with the next transmission. Each GDDM outbound terminal transmission which does not expect input to be received, specifies “wait”, requiring that the application program waits until the transmission has been completed.
Under TSO
Each GDDM outbound terminal transmission (using TPUT) specifies “hold,” requiring that the transmission physically arrives at the terminal, where applicable, before GDDM continues with the next transmission.

MAPGSTG=n
Subsystem: All.
An integer defining the mapgroup storage threshold. GDDM usually optimizes mapping functions by retaining loaded mapgroups in main storage up to the specified threshold value.

MIXSOSI={NO|YES}
Subsystem: All.
Defines whether alphanumeric and graphic character strings can contain shift-out (SO) (X'0E') and shift-in (SI) (X'0F') characters to mix one-byte (SBCS) characters with two-byte (DBCS) characters.

MIXSOSI must be set to “YES” if you are using utilities such as the GDDM-PGF ICU on a Kanji device. If an ICU chart is created when MIXSOSI=YES, it can be displayed only when MIXSOSI=YES, even if it does not contain double-byte characters.

The default double-byte character set is always used in mixed strings, whether the default set was selected by DBCSNM, DBCSLNG, or simply defaulted.

NATLANG=c
Subsystem: All.

The language used by GDDM, the GDDM-PGF Interactive Chart Utility, and Presentation Graphics routines in generating messages, user control panels, Menu Panels, Help Panels, and generated charts. The meanings of “c” are defined as:

A U.S.-English
B Brazilian Portuguese
C Simplified Chinese (People’s Republic of China)
D Danish
F French
G German
H Korean (Hangeul)
I Italian
K Japanese (Kanji)
N Norwegian
Q Canadian French
S Spanish
T Traditional Chinese (Taiwan)
V Swedish.

The corresponding National Language Support special feature must be installed.

In the encoded-UDS format, the default value must be coded as X'xx000000', where “xx” is the hexadecimal equivalent of the character “c”.

NUMBFRM={1|2|3}
Subsystem: All.

The number representation convention to be used by GDDM and GDDM-PGF is:

1 N,NNN,NNN.MMM (Period decimal convention)
2 N.NNN.NNN,MMM (Comma decimal convention)
3 N NNN NNN,MMM (French decimal convention).

OBJFILE=([aaaaaaaa],[bbbbbbbb],...)
Subsystem: All.

Up to twelve 8-character strings that show the default file-types (VM), default ddnames (TSO), default File Control data-set names (CICS), or default DBD names (IMS):

`aaaaaaaa` symbol sets
`bbbbbbbb` generated mapgroups
`cccccccc` saved pictures
`dddddddd` chart formats
`eeeeeeee` chart data
`ffffffff` GDDM-IMD tutorial pages
`gggggggg` GDF files

hhhhhhhh	GDDM-GKS metafiles
iiiiiiii	Chart data definition (under TSO and VM)
	Reserved (under CICS and IMS)
jjjjjjjj	Projection definition
kkkkkkkk	Image data
llllllll	GDDM-PCLK objects and GDDM-OS/2 Link files

PARMVER={NO|YES}
Subsystem: All.
Shows whether all calls through the system programmer interface should be verified for correctness of function code and number of parameters. Requesting this function incurs additional processing overheads.

SAVBFSZ=n
Subsystem: All.
An integer, in the range 1024 through 32000, which is the FSSAVE transmission buffer size used by GDDM. The FSSAVE function stores preformatted data streams ready for subsequent recall and display by FSSHOW. SAVBFSZ determines the transmission buffer size used by such a saved data stream. The value of SAVBFSZ at the time of the FSSAVE call must not exceed the value of IOBFSZ at the time of the FSSHOW call.

For maximum efficiency, you should choose the FSSAVE buffer size so that the value `4088/(n + 5)` is greater than 2 and close to an integer (whole number).

Be careful about choosing an unnecessarily high value as this may cause problems with BSC line protocol.

For 3179-G, 3192-G, or 3472-G color display stations, 3270-PC/G, /GX, or /AT workstations, and devices supported by GDDM-PCLK, the size saved is restricted to approximately 3500 bytes to avoid possible controller timeouts when subsequently showing the saved file.

SOSIEMC=c
Subsystem: All.
Shows the character that is used as the shift-out/shift-in emulation character in mixed character strings. The character can be any keyable character that is consistent with the syntax of GDDM defaults; however, the character specified **must not then be used for any other purpose** (for example, as its own keyable value) in a mixed-string field.

The emulation character is ignored unless the MIXSOSI=YES external default is specified and the device is a family-1 display other than an IBM 5550.

In the encoded-UDS format, the default value must be coded as `X'xx000000'`, where “xx” is the hexadecimal equivalent of the character “c”.

STGRET={NO|YES}
Subsystem: All.
Shows whether not-enough-storage or short-on-storage conditions should be processed by GDDM, and whether control should be returned immediately to the application program with a corresponding error code. Otherwise, storage requests are unconditional, with subsequent action determined by the subsystem.

Note: Requesting this function causes GDDM to issue conditional storage requests only where these are available in the subsystem. Some subsystem requests are implicitly unconditional; in these cases, subsequent action is determined by the subsystem.

TIMEFRM={1|2|3|4}
Subsystem: All.
The time convention to be used by GDDM and GDDM-PGF is:

1	HH:MM xM	(U.S. convention; XM=AM or PM)
2	HH.MM	(International convention)
3	-HH.MM.SS	(ISO convention)
4	,HH,MM,SS	(Japanese convention).

Note that GDDM-IMD always displays the time using the International convention (format 2).

TRACE={0|n}
Subsystem: All.
An integer that is the default value of the trace control word at initialization. You can specify the value either as a decimal integer or as an assembler-language hexadecimal constant. The use of trace is described in the *GDDM Diagnosis* book.

TRCESHR={NO|YES}
Subsystem: TSO and VM.
Shows whether the trace output file is to be shared between more than one instance of GDDM. The use of trace is described in the *GDDM Diagnosis* book.

TRCESTR='aaaaaaaaaaaaa'
Subsystem: All.
Shows the default value of the trace control word at initialization, which is no trace. The alphanumeric string **aaaaaaaaaaaaa**, which can be from 1 through 256 characters long, indicates the type of trace; the use of trace is described in the *GDDM Diagnosis* book.

TRCEWID={SINGLE|DOUBLE}
Subsystem: All.
Shows the default value of the trace output width control word at initialization.

SINGLE	GDDM is to produce the trace output as 4-word hexadecimal.
DOUBLE	GDDM is to produce the trace output as 8-word hexadecimal, thus saving space.

The use of trace is described in the *GDDM Diagnosis* book.

TRTABLE=n
Subsystem: All.
An integer, in the range 5 through 1000, defining the number of trace entries to be held in the cyclic in-storage trace table.

TSOAPLF={DATAANAL|APLTEXT}
Subsystem: TSO.
Shows the APL feature that is installed on **nonqueriable** IBM 3278, and IBM 3279 Model 2 displays.

DATAANAL
GDDM is to assume that any APL feature installed on any display of the above type is the Data Analysis-APL feature, unless specific application program device-definition information shows otherwise. The Data Analysis-APL feature applies to such terminals as the IBM 3279.

APLTEXT
GDDM is to assume that any APL feature installed on any display of the above type is the APL/Text feature, unless specific application program device-definition information shows otherwise. The APL/Text feature applies to such terminals as the IBM 3278 and IBM 3279.

TSOCOLM=aaaaaaaa
Subsystem: TSO**.**
An 8-character string defining the default ddnames or high-level qualifiers used by GDDM for multicolored output resulting from high-resolution image devices.

The character string must contain a "+" substitution character.

TSOCPT=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that defines the default ddname used by GDDM for data sets containing Computer Graphics Metafile (CGM) conversion profiles.

TSODECK=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for object module output resulting from requests through the Image Symbol Editor or the GDDM-PGF Vector Symbol Editor.

TSODFTS=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM to access an External Defaults File.

TSOEMUL={NO|YES}
Subsystem: TSO**.**
This specifies whether, when operating in the MVS batch environment, TSO terminal I/O supervisor calls are emulated through the MVS screening facility. The emulation routines are compatible with the current version of TSO. For information on MVS SVC screening, see the *OS/VS2 System Programming Library: Supervisor Manual*, and for information on TSO see the OS/VS2 *TSO Guide to Writing a Terminal Monitor Program or a Command Processor*.

TSOGIMP=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for retrieving the generated mapgroups required for the operation of GDDM-IMD.

TSOIADS=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for the output of ADSs (application data structures) resulting from the use of GDDM-IMD.

TSOIFMT=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for exporting data as a result of using GDDM-IMD's Export Utility.

TSOMONO=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname or data set name low-level qualifier used by GDDM for monochrome page-printer output.

TSOPRNT=aaaaaaaa
Subsystem: TSO**.**
An 8-character string used to generate a name of the form "aaaaaaaa.REQUEST.QUEUE" to identify the Print Utility Master Print Queue data set, where this has not otherwise been identified by DD statement. This string is also used to generate names of the form

```
[dsn-prefix.][userid.]aaaaaaaa.REQUEST.#nnnnn,
```

which are assigned to intermediate data sets required for queued printer support.

TSORESV={NO|YES}
Subsystem: TSO**.**
If TSORESV=YES, when operating in an MVS/TSO or MVS/BATCH environment, the DASD device upon which the master print queue resides is protected by a hardware RESERVE macro instruction when it is being updated.

TSOSYSP=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for output resulting from system printers.

TSOS99S=n
Subsystem: TSO**.**
An integer defining the size (in bytes) of the intermediate data sets that are dynamically allocated for queued printer support. The IBM-supplied default of 742710 is approximately equivalent to three 3330 cylinders.

TSOS99U=aaaaaaaa
Subsystem: TSO**.**
An 8-character string defining the UNIT specification used for intermediate data sets that are dynamically allocated by GDDM in TSO Batch or MVS Batch. In foreground TSO or if the option is set to blanks (by specifying it as TSOS99U=()), GDDM allows the UNIT specification to be defaulted from the TSO user attribute data set (UADS), where available.

TSOTRCE=aaaaaaaa
Subsystem: TSO**.**
An 8-character string that is the default ddname used by GDDM for trace output.

VSECOLM=aaaaaaaa
Subsystem: VSE**.**
An 8-character string defining the default file name used by GDDM for multicolored output resulting from files containing graphics or images suitable for use by page printers.

The character string must contain a "+" substitution character.

VSEDFTS=aaaaaaaa
Subsystem: VSE**.**
An 8-character string, which is the file name of the VSE external defaults file.

VSEMONO=aaaaaaaa
Subsystem: VSE**.**
An 8-character string defining the default file name used by GDDM for monochrome output resulting from files containing graphics or images suitable for use by page printers.

VSETRCE=aaaaaaaa
Subsystem: VSE**.**
An 8-character string, which is the file name used by GDDM for trace output.

End of General-use programming interface

Appendix B. Processing options

General-use programming interface

This appendix begins with a summary list of all GDDM processing options (procopts). Full descriptions of the processing options are provided beginning on page 336. For information about the way in which procopts are specified, see Chapter 18, “Nickname user-default specifications” on page 205.

Processing options: summary list

The procopts are listed here in alphabetical order. Default settings, shown in either the Arguments or Examples columns, are printed bold and underlined. If no default setting is identified for a particular procopt, either there is no default setting or the default setting is determined by the device being used.

For a list of the procopts in order of processing-option group (as required for input to DSOPEN), see the *GDDM Base Application Programming Reference* book.

Table 42 (Page 1 of 3). Summary list of processing options (procopts)

Nickname keyword	Arguments	Procopt group code	Examples
AUNLOCK	{NO|YES}	3	(AUNLOCK,NO)
BMSCOORD	{**NO**|YES}	1	(BMSCOORD,NO)
CDPFTYPE	{**PRIM**|SEC|OVLY} Alternative to OFDSTYPE. Retained for compatibility with earlier releases of GDDM.	5	(CDPFTYPE,SEC)
CMSATTN	{**BASIC**|EXTENDED},n,addr	1001	(CMSATTN,BASIC,0,0)
CMSINTRP	{**PA1PA2**|PA2|PA1|NONE}	1000	(CMSINTRP,PA1PA2)
COLORMAS	n	3000	(COLORMAS,**0**)
CPSPOOL	xxxxxxxx,xxxxxxxx,...	1002	(CPSPOOL,TO,RSCS)
CPTAG	xxxxxxxx,xxxxxxxx,...	1003	(CPTAG,OUR3287,PRT,=,GRAPH)
CTLFAST	{**NO**|YES}	27	(CTLFAST,YES)
CTLKEY	type,value	29	(CTLKEY,**4,3**)
CTLMODE	{*|YES|NO}	28	(CTLMODE,NO)
CTLPRINT	{**YES**|NO}	30	(CTLPRINT,NO)
CTLSAVE	{YES|NO}	31	(CTLSAVE,YES)
DEVCPG	n	34	(DEVCPG,00273)
DEVCSET	n	51	(DEVCSET,1172)
FASTUPD	n	26	(FASTUPD,**0**)
FRCETYPE	{**FSFRCE**|DSFRCE}	2003	(FRCETYPE,DSFRCE)
GINKEY	type,value	45	(GINKEY,**0,0**)
GRAYLINE	{**NO**|YES}	48	(GRAYLINE,YES)
HRIDOCNM	xxxxxxxx	22	(HRIDOCNM,FIGURE9)

Table 42 (Page 2 of 3). Summary list of processing options (procopts)

Nickname keyword	Arguments	Procopt group code	Examples
HRIFORMT	{BITMAP|**CDPF**|GRIMAGE|GRCIMAGE} Alternative to OFFORMAT. Retained for compatibility with earlier releases of GDDM.	9	(HRIFORMT,CDPF)
HRIPSIZE	w,d,{TENTHS|MILLS}	8	(HRIPSIZE,50,30,TENTHS)
HRISPILL	{**YES**|NO}	6	(HRISPILL,YES)
HRISWATH	n	7	(HRISWATH,**1**)
IMGINIT	{**BLACK**|WHITE|BACKGND}	42	(IMGINIT,BLACK)
INRESRCE	{YES|**NO**}	32	(INRESRCE,YES)
INVKOPUV	{**NO**|YES}	1004	(INVKOPUV,YES)
IPDSBIN	m,n	40	(IPDSBIN,1,2)
IPDSCPI	{**100**|120|133|150|167|170|180|200}	43	(IPDSCPI,100)
IPDSIMSW	{**YES**|NO}	41	(IPDSIMSW,NO)
IPDSLPI	{6|**8**}	39	(IPDSLPI,8)
IPDSQUAL	{*|DP|DPQ|DPT|DPTQ|NLQ|LLL|MLL|HLH|LLH}	35	(IPDSQUAL,NLQ)
IPDSROT	{**0**|90|180|270}	37	(IPDSROT,180)
IPDSTRUN	{YES|NO}	38	(IPDSTRUN,NO)
LCLMODE	{**NO**|YES}	21	(LCLMODE,NO)
LOADDSYM	{**NO**|YES}	19	(LOADDSYM,YES)
OFDSTYPE	{**PS**|EPS}	5	(OFDSTYPE,EPS)
OFDSTYPE	{**DOC**|PSEG|OVLY}	5	(OFDSTYPE,PSEG)
OFFORMAT	{BITMAP|**IMAGE**|GRIMAGE|GRCIMAGE}	9	(OFFORMAT,IMAGE)
ORIGINID	{**NO**|YES}	20	(ORIGINID,YES)
OUTONLY	{**NO**|YES}	2	(OUTONLY,NO)
PATTRAN	m,n	44	(PATTRAN,5,1)
PCLK	{YES|**NO**}	33	(PCLK,YES)
PCLKEVIS	{YES|**NO**}	36	(PCLKEVIS,YES)
PLTAREA	xmin,xmax,ymin,ymax	14	(PLTAREA,0,70,0,70)
PLTDELAY	n	47	(PLTDELAY,**0**)
PLTFORMF	{NO|YES}	10	(PLTFORMF,NO)
PLTPAPSZ	{*|A4|A3|...|A|B|...}	15	(PLTPAPSZ,*)
PLTPENP	n	13	(PLTPENP,10)
PLTPENV	n	11	(PLTPENV,30)
PLTPENW	n	12	(PLTPENW,**3**)
PLTROTAT	{**NO**|YES}	16	(PLTROTAT,NO)
POSTPROC	xxxxxxxx	50	(POSTPROC,COLORPRI)
PRINTCTL	n,n,n,n,.....	4	(PRINTCTL,**1,1,66,0,0,0,80,0**)
PRINTDST	{class| *}, [destname|ddname| *|=], [writer],[forms])	2002	(PRINTDST,G,=,MYPRT)

Table 42 (Page 3 of 3). Summary list of processing options (procopts)

Nickname keyword	Arguments	Procopt group code	Examples
PRTPSIZE	w,d,{TENTHS|MILLS}	8	(PRTPSIZE,80,110,TENTHS)
PRTROT	{**0**|90|180|270}	37	(PRTROT,90)
PSCHAR	{**7**|8}	49	(PSCHAR,8)
PSCNVCTL	{**NO**|START|CONTINUE}	25	(PSCNVCTL,START)
SEGSTORE	{**YES**|NO}	17	(SEGSTORE,NO)
SPECDEV	{IBM5080|*}, {ddname|*}	23	(SPECDEV,IBM5080)
STAGE2ID	xxxxxxxx,xxxxxxxx,...	18	(STAGE2ID,*,AUX2)
TOFILE	{**NO**|YES},{**REP**|NOREP}	46	(TOFILE,NO,REP)
TSOINTRP	{**PA1**|NONE}	2000	(TSOINTRP,NONE)
TSORESHW	n	2001	(TSORESHW,**0**)
WINDOW	{**NO**|YES}	24	(WINDOW,YES)

Processing options: full descriptions

The procopts are described here in alphabetical order. For each procopt, the following information is supplied:

- The nickname syntax
- A brief description of the function of the procopt
- Applicable subsystems
- Applicable output families

In this list, default values are underlined thus: **NO**.

AUNLOCK (always-unlock-keyboard mode)

Nickname syntax: (AUNLOCK,{NO|YES})

Always-unlock-keyboard mode means that functions such as FSFRCE, which normally cause output without unlocking the keyboard, should instead unlock the keyboard, while still returning immediately to the application. This could be useful in the IMS environment, to avoid the need for the operator to press RESET before being able to enter the next transaction.

It is also useful in CICS pseudoconversational applications to cause keyboards to be unlocked on FSFRCE instead of DSCLS, which improves performance.

The default value is defined in the AUNLOCK parameter in GDDM's external defaults (see Appendix A, "External defaults" on page 307), and is subsystem-dependent.

This procopt is set to the value current at DSOPEN time. It is valid from the issue of DSOPEN to the issue of DSCLS. The value cannot be altered dynamically; if a change is required, the device must be reinitialized.

Note: For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.

Subsystems: All
Devices: Family-1

NO Normal mode (default for CICS, TSO, VM)
YES Always-unlock-keyboard mode (default for IMS).

BMSCOORD (coordination mode)

Nickname syntax: (BMSCOORD,{**NO**|YES})

Coordination mode allows a GDDM CICS application program to use Basic Mapping Support (BMS) for the alphanumeric portion of the screen, and lets GDDM build and display the graphics portion. The GDDM output functions are modified so that they alter only that part of the screen covered by the graphics field and do not corrupt any data established by BMS.

Subsystems: CICS
Devices: Family-1

NO Not in coordination mode (default)
YES In coordination mode.

CDPFTYPE

See “OFDSTYPE (output file data-stream type)” on page 350.

CMSATTN (CMS attention handling)

Nickname syntax: (CMSATTN,{**BASIC**|EXTENDED}, n,addr)

Determines how asynchronous interrupts (attentions) are handled in a GDDM application.

For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.

Subsystems: VM
Devices: Family-1 device from which the program is being run, or auxiliary device attached to that device

BASIC
Basic attention handling (the default); only an unsolicited ENTER causes an attention to be raised.

GDDM passes the attention to the next higher layer in the stack of attention handlers, and takes no action on its own behalf. All other interrupts received by GDDM are ignored.

EXTENDED
Extended attention handling; all unsolicited interrupts received by GDDM cause an attention to be raised.

GDDM partially decodes the inbound data stream causing the attention, and builds an **attention feedback block**. This contains the identifier of the attention in a similar format to that returned on ASREAD. After this information is filled in, control is passed to the next higher attention handler in the stack. The feedback block is not owned by GDDM, but is supplied by the user via this procopt. However, if either the length or the address of the block is zero, the feedback block is not filled in.

n
The length of the attention feedback block. See the description of extended attention handling (above).

addr
The address of the attention feedback block. See the description of extended attention handling (above).

CMSINTRP (CMS PA1/PA2 protocol)

Nickname syntax: (CMSINTRP,{**PA1PA2**|PA2|PA1|NONE})

Under VM, a user can usually interrupt a running program to contact the underlying supervisors. A GDDM application can choose, by this option, whether it requires this capability. The default is to retain the capability.

Notes:

1. PA2 can cause entry to CMS subset mode only when GDDM has a read outstanding at the terminal, but not if a real partition other than partition zero is active.
2. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.

Subsystems: VM
Devices: Family-1 device from which the program is being run, or auxiliary device attached to that device

PA1PA2 PA1 causes entry to CP mode; PA2 causes entry to CMS subset mode (default).

PA2 PA1 is returned to the application; PA2 causes entry to CMS subset mode.

PA1 PA1 causes entry to CP mode; PA2 is returned to the application.

NONE PA1 and PA2 are returned to the application.

COLORMAS (color-master table identifier)

Nickname syntax: (COLORMAS,n)

Identifies the color-master table to be used.

A color-master table defines how each input color is to be analyzed into one or more color masters. If this option is not specified, a single monochrome master is generated. The output file format must be IMAGE or BITMAP for color masters to be generated correctly. Use the OFFORMAT processing option to set it to IMAGE or BITMAP if the device specified by the token has function characteristics higher than IMAGE.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family-4

n The identifier of the color master table: a number that is placed after the letters “ADM” to create a color table name. For example, the number 1 results in color table ADM00001 being used. Specifying 0 (the default) means that a monochrome master is generated.

For more information about color-master tables, see Chapter 14, “Color-separation master tables” on page 145.

CPSPOOL (CMS CP SPOOL parameters)

Nickname syntax: (CPSPOOL,xxxxxxxx,xxxxxxxx,....)

Causes a CP SPOOL command to be issued for punch files that result from opening a family-1 device with a name-list of “PUNCH”, or a family-4 device with a name-list of “PRINTER”.

If specified, this option causes a CP SPOOL command of this form:

```
CP SPOOL PUNCH xxxxxxxx xxxxxxxx ........ ........
```

or this form:

```
CP SPOOL PRINTER xxxxxxxx xxxxxxxx ........ ........
```

to be issued at the time of the DSOPEN call.

A specification of the form (CPSPOOL,TO,RSCS) can be used to direct punch files to a product capable of processing them (such as RSCS Networking Version 2).

A specification of the form (CPSPOOL,TO,psfid, DEST,destname,.....) can be used to direct family-4 print files to a product capable of processing them (such as PSF/VM 2.1 or CDPF). The DEST value determines the printer to which PSF directs the output.

GDDM does not restore any previous spooling control options when the device is closed.

Subsystems: VM
Devices: Family-1 device 'PUNCH'
Family-4 device 'PRINTER'

CPTAG (CMS CP TAG parameters)

Nickname syntax: (CPTAG,xxxxxxxx,xxxxxxxx,....)

Causes a CP TAG command to be issued for punch files that result from opening a family-1 device with a name-list of “PUNCH”, or a family-4 device with a name-list of “PRINTER” and a name-count of “1” under VM.

If specified, this option causes a CP TAG command of this form:

```
CP TAG DEV PUNCH xxxxxxxx xxxxxxxx ........ ........
```

or

```
CP TAG DEV PRINTER xxxxxxxx xxxxxxxx ........ ........
```

to be issued at the time of the DSOPEN call.

GDDM inserts one blank character after each specified token, removes any extra blank characters, and removes any blank characters surrounding the character “=”. Thus, a specification of the form:

```
(CPTAG,PRINTER1,PRT,=,GRAPH)
```

causes the following CP TAG command to be issued:

```
CP TAG DEV PUNCH PRINTER1 PRT=GRAPH
```

A specification like the one above can be used to notify products capable of processing punch files (such as RSCS Networking Version 2) that the punch file contains graphics.

GDDM does not restore any previous tag information when the device is closed.

Subsystems: VM
Devices: Family-1 device 'PUNCH'
Family-4 device 'PRINTER'

CTLFAST (User Control fast-path mode)

Nickname syntax: (CTLFAST,{**NO**|YES})

Selects fast-path mode for User Control functions that require pointings. When (CTLFAST,YES) is specified and a User Control function that requires pointing (MOVE, SIZE, POINT, CENTER, ZOOM-IN, ZOOM-OUT) is selected by a PF key, it is assumed that the user has already positioned the cursor at the first pointing.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays

NO Fast path mode is not selected (the default)
YES Fast path mode is selected

CTLKEY (User Control key)

Nickname syntax: (CTLKEY,type,value)

Selects a User Control key. The default is (CTLKEY,4,3), which is PA3.

Note: For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays

type The type of key selected for entering User Control:

- **0** None. User Control cannot be entered by key action.
- **1** A PF key (see value below) is used to enter User Control.
- **4** A PA key (see value below) is used to enter User Control.

value The number of the PA or PF key used:

- **0** None. User Control cannot be entered by key action.
- **n** The number of the PA or PF key defined for User Control.

CTLMODE (User Control)

Nickname syntax: (CTLMODE,{ *|YES|NO})

Selects User Control

Notes:

1. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.
2. When running under CICS with external default (CICTIF=EXT), CTLMODE must be set to NO.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays

* User Control is available for devices not capable of supporting real partitions (the default).
YES User Control is always available, forcing emulated partitions.
NO User Control is not allowed.

CTLPRINT (User Control print)

Nickname syntax: (CTLPRINT,{**YES**|NO})

Controls the print or plot facilities offered in User Control.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays

YES Printing is allowed in User Control (the default).
NO Printing is not allowed in User Control.

CTLSAVE (User Control save)

Nickname syntax: (CTLSAVE,{NO|YES})

Controls the picture-saving facilities offered in the User Control environment.

The default value is defined in the CTLSAVE parameter in GDDM's external defaults (see Appendix A, "External defaults" on page 307), and is subsystem-dependent.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays

NO Saving is not allowed from User Control.
YES Saving is allowed from User Control.

DEVCPG (device code-page)

Nickname syntax: (DEVCPG,n)

Specifies the code page that GDDM is to use for a device. This code-page overrides that returned by a CECP device when GDDM opens it.

Subsystems: All
Devices: All

n The global code-page identifier

DEVCSET (set the device character set)

Nickname syntax: (DEVCSET,n)

This option specifies the character set that GDDM is to use for a device. This character set overrides the one returned by the device when GDDM opens it. A character set specified in the DEVCSET procopt takes precedence over one specified in a DEVCPG procopt.

Subsystems: All
Devices: All

The identifier of the character set to be used for the device.

Note: If this processing option is not specified, GDDM selects the correct table from ADMDATRN using the value specified on the DEVCPG procopt or the code-page identifier returned by the device query.

FASTUPD (fast update mode)

Nickname syntax: (FASTUPD,n)

Selects the level of picture degradation that is acceptable to enable a fast update of the graphic data on the device. The option selected can subsequently be queried by the FSQUPD call and changed by the application using the FSUPDM call; see the description of the FSUPDM call in the *GDDM Base Application Programming Reference* book.

The main use of this processing option is to control fast-update mode by means of a nickname.

It has an effect only on IBM 3270-PC/G, /GX, and /AT workstations, 3179-G, 3472-G, and 3192-G color display stations, 5550-family workstations, and devices supported by GDDM-PCLK or GDDM-OS/2 Link. On these devices, the color mixing can be degraded to use exclusive-OR mode to enable segments to be changed or deleted without causing a redraw of the picture.

Subsystems: TSO, CMS, CICS
Devices: Family-1 3270-PC/G, GX, and /AT workstations, 3179-G, 3472-G, and 3192-G displays, 5550-family workstations, and devices supported by GDDM-PCLK or GDDM-OS/2 Link.

Possible values are:

0 No degradation of picture fidelity (the default)
1 Picture degradation acceptable using GDDM's chosen method for the picture.

FRCETYPE (Output type definition)

Nickname syntax:
(FRCETYPE,{**FSFRCE**|DSFRCE})

This option indicates whether the device defined in the namelist, which is a partitioned data set, is to be opened so that page segments or overlays are saved as members of this data set. It also indicates the permitted type of output call (FSFRCE or DSFRCE).

Notes:

1. The namelist can be a DDNAME that refers to an already allocated partitioned data set.
2. If a member name is included in the namelist, it is ignored because the member name specified in the DSFRCE call takes precedence.

Subsystems: MVS TSO and MVS/Batch
Devices: Family-4

1 The option group code: 2003.

2 Specifies whether the device is to be opened as a partitioned data set for saving multiple page segments or overlays.

- **0** Use FSFRCE to output a page to the device defined in the namelist.
- **1** Use DSFRCE to output a page to be a member of the partitioned data set defined in the namelist.

GINKEY (graphics input key)

Nickname syntax: (GINKEY,type,value)

Selects a key for switching from graphics input to alphanumeric input on ASCII graphics displays. On ASREAD and GSREAD calls, if a graphics input device is enabled, the graphics cursor appears. Having positioned the graphics cursor, the user can switch to alphanumeric input using the specified key. The default is the ENTER key.

Subsystems Not IMS
Devices ASCII graphics displays

type

- **0** The ENTER key (the default)
- **1** PF key (see value below)
- **4** PA key (see value below)

value the number of the PF key or PA key selected.

Note: The keyboards for some ASCII graphics displays require multiple key strokes for some PF keys and PA keys. You should select a key that is available as a single key action on the keyboard. For example, you should not select a key that requires the user to press ESC followed by another key.

GRAYLINE (PostScript grayline attribute)

Nickname syntax: (GRAYLINE,{**NO**|YES})

This option specifies how colored text and lines are to be drawn when family-4 output is generated using a device token for a monochrome PostScript printer. If you use a device token for a color printer, the setting of GRAYLINE is ignored.

Subsystems: All
Devices: Family-4 (PostScript)

NO All characters and lines are drawn in black. (Default)
YES Characters and lines are drawn in color, which a monochrome PostScript printer interprets as shades of gray.

The printed results depend on the device to which the output is sent. If it is a color printer, GRAYLINE, YES has no effect and full color output is produced. The GRAYLINE procopt has no effect on the way monochrome printers deal with colored **areas**.

HRIDOCNM (document name)

Nickname syntax: (HRIDOCNM,xxxxxxxx)

Provides a name for the document or primary data stream that is passed to CDPF. This name is printed in the picture separator line, above each picture. This can be used to identify the owner of the printed output.

Subsystems: TSO, VM, MVS, and VSE
Devices: Family-4, 4250 printers only

HRIFORMT

See “OFFORMAT (output file format)” on page 351.

HRIPSIZE (output paper size)

Nickname syntax: (HRIPSIZE,w,d,{TENTHS|MILLS})
Alternative syntax: (PRTPSIZE,w,d,{TENTHS|MILLS})

Specifies the size of the paper, as width by depth. The default size of the paper is given by the device characteristics, which are defined by the device token being used.

Notes:

1. Although the term “paper size” is used, the output medium need not be paper.
2. The picture output may be slightly smaller than that specified in the width and depth parameters.
3. Most printers are unable to print right to the edges of the loaded paper but have a small margin or “unprintable area” around the edges. The size of this margin varies with the model of printer used and the size of paper loaded.

   On PostScript printers, the setting of the OFDSTYPE procopt determines how much space GDDM allows for the unprintable area.
4. If the IPDSROT (or PRTROT) processing option has been specified with a rotation of 90 degrees or 270 degrees, the width and depth are interchanged.
5. This option does not apply to AFPDS output generated using cell-based device tokens.
6. The alternative name for this option, PRTPSIZE, can also be used in nickname statements.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family-4

w The paper width in the specified units
d The paper depth in the specified units
units Can be one of:
- **TENTHS** Units are tenths of an inch
- **MILLS** Units are millimeters

HRISPILL (spill file usage)

Nickname syntax: (HRISPILL,{**YES**|NO})

Determines whether a spill file in external storage is to be used while processing a high-resolution image file for a 4250 printer.

The use of a spill file reduces the main storage requirements at the cost of processing time. If a spill file is not used and segments are used, primitives outside segments (temporary data) do not form part of the final image, except where they occur between the last GSSCLS and ASREAD or FSFRCE calls.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family-4

Spill file usage:
- **YES** Store internal picture description on disk in a spill file (the default)

NO Store internal picture description in main storage.

HRISWATH (number of swathes)

Nickname syntax: (HRISWATH,n)

Determines whether a high-resolution image is to be processed as one horizontal "swathe" or many. ("Swathes" are also called slices.)

The use of swathing reduces storage requirements but at the cost of processing time.

Subsystems: TSO, VM, MVS, and VSE
Devices: Family-4

The value "n" is the number of swathes to be used. The default is 1, which causes the output image to be generated with just one pass through the internal picture description.

IMGINIT (image field initialization)

Nickname syntax: (IMGINIT,{**BLACK**|WHITE|BACKGND})

Specifies the initial value to be used for current device bilevel images.

Subsystems: All
Devices: Family-1, -2, and -4

BLACK Black (the default)
WHITE White
BACKGND Black on displays and white on printers.

Note: When specifying this procopt by means of a nickname, ensure that the procopt is not applied to programs that already invert the initial image field in order to get a white background.

INRESRCE (inline resources)

Nickname syntax: (INRESRCE,{**NO**|YES})

Indicates whether the CDPU is to transfer inline resources from the CDPDS input file to the AFPDS output file.

Subsystems: All
Devices: Family-4 AFPDS printers

NO Inline resources are not supported (the default).
YES Inline resources are supported.

INVKOPUV (automatic invocation of VM print utility)

Nickname syntax: (INVKOPUV,{**NO**|YES})

Indicates whether GDDM is to invoke the GDDM print utility automatically after a print file has been created.

If this function is requested, a temporary print file is created, and the print utility is requested to print this file on the device specified by the **name-list** parameter. After printing, the temporary file is erased.

You can use this procopt as an alternative to entering the ADMOPUV command.

Subsystems: VM
Devices: Family-2

NO Do not invoke print utility (the default)
YES Invoke print utility automatically.

IPDSBIN (IPDS paper feed bin)

Nickname syntax: (IPDSBIN,m,n)

Specifies the paper-feed bins to be used for the main document and header page.

Subsystems: Not IMS
Devices: 3812 Model 2, 3816, 3112, 3116, 3912, 3916, 4028, and 4224 printers that have auto sheet feed (ASF) installed, family-1 and -2

Values are:

m Paper feed bin for main document. Can be in the range 0 through 100
n Paper feed bin for header page. Can be in the range 0 through 100.

For the 4224 with ASF, possible source values are:

0 Default bin as specified on the 4224 operator's panel
1–3 Automatic input bins
100 Manual feed

and the default is (IPDSBIN,0,0).

For the 3812 Model 2 and 3816, possible source values are:

0 Cassette 1
1–2 Cassettes

and the default is (IPDSBIN,1,2).

For the 3112, 3116, 3912, 3916, and 4028, the allowable source values are:

0 Default bin as specified on the 4028 operator's panel
1 Primary bin
2 Secondary bin
100 Manual feed

and the default is (IPDSBIN,0,0).

Note: When you print composite documents, this procopt overrides any bin selection made by CDPDS MMC structured fields, and all forms of the document are printed from the bin selected by the procopt.

GDDM determines the paper size for all bins from the device token or by querying the device. If the token or query reply does not give the paper size for a particular bin, the values for bin 1 are used.

IPDSCPI (IPDS characters per inch)

Nickname syntax: (IPDSCPI,**100**|120|133|150|167|170|180|200)

Specifies the font pitch in characters per 10 inches. If the selected pitch is not supported by the printer, the nearest pitch that ensures the picture fits on the physical page is selected.

This procopt overrides the printer setting. The printer itself should be set to the default 10 characters per inch.

Subsystems: Not IMS
Devices: Family-1 and -2 IPDS printers

100 10.0 characters per inch (the default)
120 12.0 characters per inch
133 13.3 characters per inch
150 15.0 characters per inch
167 16.7 characters per inch
170 17.0 characters per inch
180 18.0 characters per inch
200 20.0 characters per inch

Note: Use of any pitch other than the default may cause problems printing programmed symbols, APL, Katakana, and mode-1 graphics text.

IPDSIMSW (IPDS Image swathing)

Nickname syntax: (IPDSIMSW,{**YES**|NO})

Enables data to be sent as a series of compressed subimages (swathed).

Subsystems: Not IMS
Devices: 3812 Model 2, 3816, and 4028, family-1 and -2

YES Data is sent to the printer as a series of compressed subimages (swathed). This is the default.

NO Data is sent to the printer as a single compressed image (unswathed).

Note: The unswathed option normally gives better performance, especially when used with scanned bilevel documents. For some images, such as scanned photographs, the swathed option gives better performance and uses less storage. Frequent users of such gray-scale images may choose not to use this procopt.

IPDSLPI (IPDS lines per inch)

Nickname syntax: (IPDSLPI,{6|**8**})

Selects printing at 6 or 8 lines per inch.

Do not use this procopt when printing composite documents. This procopt overrides the printer setting. The printer itself should be set to the default 8 lines per inch.

Subsystems: Not IMS
Devices: Family-1 and -2 IPDS printers

8 Print at 8 lines to the inch (the default)
6 Print at 6 lines to the inch

IPDSQUAL (IPDS printer quality)

Nickname syntax:
(IPDSQUAL,{*|DP|DPQ|DPT|DPTQ|NLQ|LLL|MLL|HLH|LLH})

Selects the print quality on IPDS printers.

Subsystems: Not IMS
Devices: IPDS printers

*	Printer hardware setting (the default)
DP or DPQ	Data-processing quality text, high-density graphics, and high-contrast bar codes
DPT or DPTQ	Data-processing text quality text, high-density graphics, and high-contrast bar codes
NLQ	Near-letter-quality text, high-density graphics, and high-contrast bar codes
LLL	Data-processing-quality text, low-density graphics, and low-contrast bar codes
MLL	Data-processing-quality text, low-density graphics, and low-contrast bar codes
HLH	Near-letter-quality text, low-density graphics, and high-contrast bar codes
LLH	Data-processing-quality text, low-density graphics, and high-contrast bar codes

IPDSROT (IPDS Page rotation)

Nickname syntax: (IPDSROT,{**0**|90|180|270})
Alternative syntax: (PRTROT,{**0**|90|180|270})

The whole page to be printed is rotated by the specified amount.

Subsystems: Not IMS
Devices: Family-1 and -2 IPDS 3812 Model 2, 3816, 4028; family-4 PostScript, AFPDS, and CDPF output

0	Portrait (top of page is drawn at upper edge of paper (the default))
90	Landscape (top of page is drawn at right edge of paper)
180	Portrait (top of page is drawn at lower edge of paper)
270	Landscape (top of page is drawn at left edge of paper)

Notes:

1. The alternative name for this option, PRTROT, can also be used in nickname statements.
2. When rotating AFPDS output, some printers may not support the printing of presentation text to the degree of rotation specified. For more information, see *Advanced Function Printing: Printer Information*, G544-3290.

IPDSTRUN (IPDS data-stream truncation)

Nickname syntax:(IPDSTRUN,{NO|YES})

Specifies whether the data stream sent to a 4234 printer is to be truncated when the storage capacity of the device is exceeded, or whether the printer should rewind the paper to cope with excess data.

Subsystems Not IMS
Devices 4234 printers, family-1 and -2

NO Do not truncate the data stream (the default for 4234)
YES Truncate the data stream (the default for other IPDS printers).

LCLMODE (local interactive graphics mode)

Nickname syntax: (LCLMODE,{**NO**|YES})

Indicates whether panning, zooming, and scaling of graphics on 3270-PC/G, /GX, or /AT workstations is to be performed using local data streams or by rebuilding the picture in the host.

Full information on how to use local interactive graphics mode is given in the *GDDM User's Guide*.

Subsystems: All
Devices: Family-1 3270-PC/G, /GX, and /AT workstations

NO Local interactive graphics mode not allowed (the default)
YES Local interactive graphics mode allowed.

LOADDSYM (load default symbol sets)

Nickname syntax: (LOADDSYM,{**NO**|YES})

Indicates whether the workstation is to use the device's default symbol sets or the GDDM default symbol sets. If the application program requires any alternative characters in the symbol set (for example, national use characters), GDDM's default symbol sets must be used. For information on changing GDDM's default symbol sets, see Chapter 12, "The GDDM default symbol sets" on page 131.

Note: Using GDDM's symbol sets reduces the amount of storage in the workstation that is available for segment storage and for symbol sets loaded by the application program.

Subsystems: All
Devices: Family-1 3270-PC/G, /GX, and /AT workstations, 3179-G, 3192-G, and 3472-G color display stations, and devices supported by GDDM-OS/2 Link

NO Use the workstation's default mode-2 and mode-3 symbol sets (the default)

YES Load GDDM's mode-2 and mode-3 symbol sets, replacing the device's default symbol sets.

OFDSTYPE (output file data-stream type)

Nickname syntax: (OFDSTYPE,{**PS**|EPS})
Alternative syntax: (OFDSTYPE,{**DOC**|PSEG|OVLY})
Alternative syntax: (CDPFTYPE,{**PRIM**|SEC|OVLY})

Determines whether the formatted output file is to be constructed as primary data stream, or as secondary data stream, or as an overlay.

Primary data stream is a complete PostScript, AFPDS or CDPF document that can be printed. Secondary data stream can be an encapsulated PostScript (EPS) file or a page segment (PSEG) that should be imbedded in a document to be printed.

A complete document (primary data stream) can be printed. Except for encapsulated PostScript files, secondary data stream must be imbedded in a document before it can be printed. Primary data streams can be processed by:

- PostScript printers
- IBM Print Services Facility (PSF) for printing on advanced function printers
- IBM Composed Document Print Facility (CDPF) for printing on the 4250 printer

Notes:

1. If a 4250 output file is to contain text that refers to the 4250-printer fonts in addition to graphics picture data, it is recommended that the file be formatted as a page segment and included as part of another document.
2. Family-4 output created by the CDPU is always a document (primary data stream), regardless of the setting of the OFDSTYPE procopt.
3. When you create a PostScript document, (without using the CDPU), GDDM scales and positions the output so that it all appears in the printable area. The size of the printable area depends on the PostScript printer used. No adjustment is made for output from the CDPU.
4. When you create encapsulated PostScript (EPS) output, the full size of the page specified is assumed to be available.
5. If the creation of EPS produces more than one page of output, only the first page is placed in the family-4 PostScript print file.
6. For compatibility with previous releases of GDDM, the option name CDPFTYPE may also be used. The old forms PRIM and SEC are allowed as synonyms for DOC and PSEG. If the OVLY parameter is specified when generating output with a PostScript device token, it is ignored and the default setting, PS, is used.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family 4

PS or **DOC** or **PRIM** Produce primary data stream (a document) (the default). The device token for the device with which this procopt is associated determines which kind of document is produced.

EPS or **PSEG** or **SEC** Produce secondary data stream (encapsuleated PostScript or a page segment). Not supported for the CDPU.

OVLY Produce an overlay segment (secondary data stream). Not supported for the CDPU or PostScript.

OFFORMAT (output file format)

Nickname syntax: (OFFORMAT,{BITMAP|IMAGE|GRIMAGE|GRCIMAGE})
Alternative syntax: (HRIFORMT,{BITMAP|CDPF|GRIMAGE|GRCIMAGE})

Determines the format of the output file. Unformatted output is a representation of the picture as one bit for each pixel. Formatted output is in a form suitable for processing either by the Print Services Facility (PSF) for AFP data streams, and other printers, or by the Composed Document Printing Facility (CDPF) for the 4250. By default, GDDM produces output to suit the highest functional characteristics of the device token used.

For compatibility with previous releases of GDDM, the option name HRIFORMT may also be used. The old form CDPF can be used as a synonym for IMAGE.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family 4

BITMAP Produce unformatted output.

IMAGE or **CDPF** Produce formatted output for IBM 4250 or AFPDS printers depending on the device token used. For AFPDS printers, this format of output contains only IM (uncompressed) image. All graphical constructs are converted to image (the default where no user device token is specified).

GRIMAGE Output of any graphics contained within segments in the GDDM graphics field is as GOCA graphics orders. (GOCA produces a shorter data stream than graphics converted to image.) Output of any image in the GDDM image field is in IM uncompressed form. The use of this value is supported by PSF/VM Version 2.1 and PSF/MVS Version 2.1 or later.

GRCIMAGE Output of any graphics contained within segments in the GDDM graphics field is as GOCA graphics orders. Output of any image in the GDDM image field is in IO compressed form. (This reduces the size of the file, compared with GRIMAGE output.) The compression algorithm used is MMR 8815. The use of this value is supported by PSF/VM Version 2.1 and PSF/MVS Version 2.1 or later.

Note: The number of pixels per line width specified for each device token applies only to rastered graphics (when procopt OFFORMAT or HRIFORMT is set to BITMAP or IMAGE/CDPF). The standard line width for the current device is used when procopt OFFORMAT or HRIFORMAT is set to GRIMAGE or GRCIMAGE.

ORIGINID (origin identification)

Nickname syntax: (ORIGINID,{**NO**|YES})

Indicates whether GDDM is to draw an origin identification string (consisting of a user ID, the date, and the time) in the bottom left-hand corner of the graphics field.

For plotters, the identification appears inside a background-shaded box, so that no part of the picture can obscure it. However, if the plotting area is small, the origin identification string might be clipped and the right-hand side might be lost.

For family-1 printers, the identification is similar to an alphanumeric field. The identification is truncated, if necessary, by the page width.

When specified for a family-2 device, the processing option is passed (in the print file) to the print utility, which specifies the processing option when opening the output device.

Note: This option is regarded as being "mergeable". That is, if ORIGINID is not specified, its current value remains in effect.

Subsystems: All
Devices: All, but used by family-1 plotters and printers and family-2 printers only

NO No origin identification (the default)
YES Origin identification required.

OUTONLY (output-only mode)

Nickname syntax: (OUTONLY,{**NO**|YES})

Output-only mode means that functions such as ASREAD and FSSHOW, which normally imply a wait for the operator to enter data, should instead return immediately to the application without unlocking the keyboard (unless this has been imposed by the always-unlock-keyboard mode, see option group 3). One use of this option is to allow a device to be opened so that it can display a continuous series of pictures using FSSHOW, without any operator intervention.

Subsystems: All
Devices: Family-1

NO Not output-only mode (default)
YES Output-only mode.

PATTRAN (translating user-defined shading patterns)

Nickname syntax: (PATTRAN,m,n)

Specifies which tables defined within the ADMDGTRN source module are to be used to translate user-defined shading patterns in the range 65 through 254 to the 16 GDDM-defined patterns. GDDM provides three tables, and others may be added by the user.

Subsystems: All
Devices: Family-1 or -2 IPDS printers, family-4 printers when procopt OFFORMAT is set to GRIMAGE or GRCIMAGE, 3179-G, 3192-G, 3472-G, 3270-PC/G, 3270-PC/GX, ASCII graphics displays, and devices running GDDM-PCLK or GDDM-OS/2 Link.

m The table number to be used for monochrome (geometric) shading patterns. This number is padded on the left with zeros to make a 5-digit field and a prefix of ADM is added to form the label of a table in the ADMDGTRN module.

n The table number to be used for triplane (color) shading patterns. The format of this table number is as above.

For more information, see Chapter 13, "Translation of user-defined shading patterns" on page 141.

PCLK (GDDM-PCLK)

Nickname syntax: (PCLK,{**NO**|YES})

Indicates whether GDDM-PCLK is to be made available. If it is set to YES, users of GDDM applications on nongraphics displays, such as 3278s, are prompted to indicate whether they want to use GDDM-PCLK.

The PCLK procopt is ignored if coordination mode is also selected (BMSCOORD procopt is set to YES).

Subsystems: Not IMS
Devices: Devices running GDDM-PCLK

NO GDDM-PCLK not available (the default)
YES GDDM-PCLK available.

PCLKEVIS (encoded data fields on personal computers)

Nickname syntax: (PCLKEVIS,{**NO**|YES})

Indicates whether the fields are to be displayed or are to be made nondisplayable.

Subsystems: Not IMS
Devices: Devices running GDDM-PCLK

NO Encoded data fields to be nondisplayable (the default)
YES Encoded data fields to be displayed.

(PCLKEVIS,YES) must be used with GDDM-PCLK if your terminal emulator normally discards nondisplayable characters.

PLTAREA (plotting area)

Nickname syntax: (PLTAREA,xmin,xmax,ymin,ymax)

Specifies the area of the paper into which GDDM is to draw the picture on a plotter. If all values are specified as zero, the user defines the plotting area (before the DSOPEN call is issued) by pressing the appropriate buttons (P1, P2, and ROTATE) on the plotter, when these buttons are supported; otherwise, the maximum plotting area is used. For long plots, PLTAREA should be allowed to default or be set to its default value (0,100,0,100).

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: All family-1 plotters

xmin The minimum x value as a percentage of the maximum paper width. The default is 0.

xmax The maximum x value as a percentage of the maximum paper width. The default is 100.

ymin The minimum y value as a percentage of the maximum paper height. The default is 0.

ymax The maximum y value as a percentage of the maximum paper height. The default is 100.

PLTDELAY (plot roll medium delay)

Nickname syntax: (PLTDELAY,n)

Specifies the delay between successive frames when long plots are drawn.

This processing option is ignored for short (single-sheet) plots, and if the plotter does not support roll-feed media.

Subsystems: CICS, TSO, VM
Devices: All family-1 plotters

n Any value in the range 0 (default) through 1200. The delay, in seconds, between successive frames when drawing long plots.

Note: Inner layers of non-plastic roll media do not stabilize, with respect to changes in relative humidity, until fully exposed to the air. The medium can expand or contract during plotting, resulting in inaccurate plots.

PLTFORMF (plotter page feed)

Nickname syntax: (PLTFORMF,{**YES**|NO})

Specifies whether a page feed is required after each GDDM page sent to the plotter by an output call such as FSFRCE. GDDM issues a warning message (ADM0094) when the device is opened if it does not support page feed. The GDDM default action is to cause a page feed for those devices that support it.

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: Family-1 6182, 6186 plotters

NO No page feed
YES Page feed (the default).

PLTPAPSZ (plotter paper size)

Nickname syntax: (PLTPAPSZ,{*|A4|A3|...|A|B|...})

Specifies the size of the paper that is loaded in a plotter. Plotters that have paper-size switches must have them set correctly to indicate the size of the paper loaded; otherwise, the aspect ratio might be distorted, the picture might not be placed centrally, or only part of the picture might be drawn.

If this option is not specified, GDDM uses whatever paper size is already loaded in the plotter.

When IBM-GL files are being created, the PLTPAPSZ procopt should be used to specify the paper size loaded on the eventual output device.

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: All family-1 plotters

Possible values are:

*, meaning the default (whatever paper size is loaded)

A or A4 size
B or A3 size
C or A2 size
D or A1 size
E or A0 size

When plotting on roll-feed plotters, use these paper-size codes:

For 36-in wide paper, specify E, or A0, or smaller
For 24-in wide paper, specify A1, or D, or smaller
For 11-in wide paper, specify A3 or B

This value is specified in ISO dimensions (A0, A1, A2, A3, A4, A5) or ANSI dimensions (A, B, C, D, E).

PLTPENP (plotter pen pressure)

Nickname syntax: (PLTPENP,n)

Specifies how hard the plotter pen is to be pressed onto the plot bed.

The recommended values are:

- On paper:

  10 grams: Fiber-tipped pens
  18 grams: Roller
  50 grams: Drafting

- On transparencies:

  18 grams: Fiber-tipped pens

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: Family-1 7374 and 7375 plotters

Possible values are:

0 The pressure set by the user on the plotter control buttons (see below)
1 – 255 The pressure, in grams, related to the actual pressure that can be set on the plotter with the control buttons.

If a value greater than the maximum for the plotter is specified, the maximum pressure is set.

If a value less than the minimum for the plotter is specified, the minimum pressure is set.

The range of values that can be set on the 7374 and 7375 plotters using the plotter control buttons is:

Button	***Pressure***
1	10 grams
2	18 grams
3	26 grams
4	34 grams
5	42 grams
6	50 grams

7 58 grams
8 66 grams.

Note: Refer to the information on the pressure-select instruction in the appropriate color plotter programming manual.

PLTPENV (plotter pen velocity)

Nickname syntax: (PLTPENV,n)

Specifies the pen velocity to be used by a plotter. The value applies to **all** the pens in the plotter. The default (0) uses the velocity set up on the plotter. It may be necessary to specify a lower value for pens used on material such as transparencies.

The recommended values are:

- On paper:

  50 centimeters/second: Fiber-tipped pens
  60 centimeters/second: Roller
  15 centimeters/second: Drafting

- On transparencies:

  10 centimeters/second: Fiber-tipped pens

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: Family-1 7371, 7372, 7374, and 7375 plotters

0 The velocity set up by the plotter operator (the default)
1 – 255 The velocity in centimeters per second, related to the actual velocity values available for each plotter.

If a value greater than the maximum for the plotter is specified, the maximum velocity is set. This is:

38 centimeters/second: For a 7371 and 7372
60 centimeters/second: For a 7374 and 7375.

Note: Refer to the information about the velocity-select (VS) instruction in the appropriate color plotter programming manual.

PLTPENW (plotter pen width)

Nickname syntax: (PLTPENW,n)

Specifies the width of the pens to be used in a plotter. Applies to **all** the pens in the plotter.

GDDM uses the pen width to determine how far apart to space lines when the plotter fills areas. If the plotter uses pens of different widths in the same picture, the pen-width value must be set to the size of the pens used for filling areas.

The pen width is used for:

- Image pixel size
- Shading line separation

- Double-width line separation
- Background line width where clipped from underlying areas

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices All family-1 plotters

The plotter pen width in tenths of a millimeter:

0 Pen width of 0.3 millimeters (the default)
1 – 10 Pen width of 0.1 through 1.0 millimeters

PLTROTAT (plotter picture orientation)

Nickname syntax: (PLTROTAT,{**NO**|YES})

By default, GDDM draws the plotted picture with the x (horizontal) axis along the longest side of the paper ("landscape" format). This option allows the picture to be rotated by 90 degrees, so that the x axis is along the shorter side of the paper ("portrait" format). This does not affect the way in which the paper is placed in the plotter; instead, it specifies the orientation of the picture relative to the paper on the plotter bed.

GDDM ignores option group 16 when the drawing area is set by pressing buttons on the plotter (see option group 14) because this action controls the orientation of the picture.

Note: This processing option does not apply to plotting via GDDM-OS/2 Link. If it is specified on the GDDM host session, it is ignored.

Subsystems: CICS, TSO, VM
Devices: All family-1 plotters

Possible values are:

NO No rotation (the default value)
YES Rotate the picture by 90 degrees.

POSTPROC (Postprocessing of family-4 and GL output)

Nickname syntax: (POSTPROC,xxxxxxxx)

This option specifies the name of a procedure or program to be invoked by GDDM when family-4, or family-1 GL plot file output (created with TOFILE), has been generated. The procedure or program is then passed the name of the family-4 or GL plot file as a parameter.

Subsystems: TSO and VM
Devices: Family-4, and Family-1 plotters with TOFILE.

xxxxxxxx The name of a procedure or program that is to perform post processing on the family-4 output.

If the program is an EXEC running on the CMS subsystem, it must be able to operate in CMS SUBSET mode. If the program is running on the TSO subsystem, the TSO Service facility (IKJEFTSR) is used to invoke a TSO command, CLIST or REXX exec.

PRINTCTL (print control options)

Nickname syntax: (PRINTCTL,n,n,n,n,....)

(where n,n,n,n,... represents the values defined below). This option controls printing and copy functions. The group has this format:

1	Heading indicator
2	Number of copies
3	Page depth
4	Top margin
5	Left margin
6	Bottom margin
7	Max FSLOG characters/line
8	Alphanumeric device type

Notes:

1. This option is of variable length and is regarded as being “mergeable” (that is, if some of the values are omitted, their current values are not changed).
2. Of the parameters listed below, only number of copies, depth of top margin, and width of left margin apply to family-2 print files spooled to plotters.
3. If, for plotters, the PLTAREA processing option is specified in addition, the PRINTCTL margins take effect first. The resulting user page is positioned with respect to the plotting area as defined by the PLTAREA processing option.
4. For family-4 devices, this procopt applies only to those devices defined by cell-based device token.
5. Print margins specified on the PRINTCTL procopt are in addition to margins specified via the GDDM-PGF ICU interface.

Subsystems: All
Devices: All

1 The heading indicator (family-2 only):
- **0** Do not print a heading page
- **1** Print a heading page (the default).

2 The number of copies (applicable to family-2 only). The default is 1. If 0 is specified, 1 is assumed.

3 The page depth in rows (FSLOG and FSLOGC only). The default is the maximum page depth for the device.

The page depth specifies the vertical size of a page of paper, fold-to-fold, in rows. If zero is specified for this parameter, a value of 66 (or the device maximum) is assumed.

4 The depth of the top margin, in rows. The default is 0.

The top and left margins (fullwords 6 and 7) specify the top left-hand corner, within each page of the paper, of the printed data. Also, for FSLOG

and FSLOGC purposes, a bottom margin may be specified. The total number of printed lines for each page for FSLOG and FSLOGC data is:

```
(page depth)–(top margin)–(bottom margin)
```

Note: The maximum page size for the device is taken from the device definition, as defined by the device-token parameter.

5 The width of the left margin, in columns. The default is 0.

See the description for the top margin.

6 The depth of the bottom margin, in rows (FSLOG and FSLOGC only). The default is 0.

7 Maximum number of characters per line (FSLOG and FSLOGC only). The default is the maximum page width for the device less the width of the left margin.

Left margin + maximum number of characters per line must not exceed the maximum page width for the device.

8 Alphanumeric device type for translation. The default is 0.

For information about the values that can be specified, see the description of ASTYPE in the *GDDM Base Application Programming Reference* book.

PRINTDST (TSO family-2 and family-4 print-file destination)

Nickname syntax:
(PRINTDST,{class| *},[destname|ddname| *|=],[writer],[forms])

This option controls the destination of the family-2 print output and can also place parameters onto the JES spool queue.

The default destination is the ADMPRINT queue.

Under TSO, this procopt is a convenient way of sending family-4 output to a print server, such as PSF/MVS.

Subsystems: TSO (including TSO/Batch and MVS/Batch)
Devices: Family-2 and family-4

An 8-character token containing one of:

class Appropriate output class for the JES spool system.

***** Output is to go to ADMPRINT queue or a ddname. Not valid for Family 4.

An 8-character token containing one of:

destname The JES Remote workstation name, associated through JES/328X, with the required target printer.

***** Output is to go to the ADMPRINT queue. Not valid for Family 4.

ddname The ddname of a DD statement describing the output data set to be used.

= Use the *namelist* as the destination name.

7 and 8 An 8-character token consisting of:

writer The name of the external writer program that is to process the SYSOUT data set.

9 and 10 An 8-character token consisting of:

forms Identifies the forms on which the SYSOUT data set is to be printed or punched.

Note: GDDM processes only the first four characters.

Notes:

1. The parameters are positional. At least *class* must be present.
2. This processing option is "mergeable," that is, if a parameter is omitted, the current value is not changed.
3. Parameters that are not required must be entered as blanks in the encoded form.
4. The default values for *destname*, *writer*, and *forms* are system defined.
5. Enterprises with many output devices can use the generic form of the print destination, '=' to reduce the number of nickname statements in the user defaults module.

   "The format of the nickname UDS" on page 205 for more details.

PRTPSIZE

See "HRIPSIZE (output paper size)" on page 344.

PRTROT

See "IPDSROT (IPDS Page rotation)" on page 348.

PSCHAR (PostScript character storage)

Nickname syntax: (PSCHAR,{**7**|8})

This option indicates how character data is to be stored in the ASCII files generated for PostScript printers.

Subsystems: All
Devices: Family-4 (PostScript)

7 Characters are stored using PostScript hexadecimal representation so that the generated file can be printed on a PostScript printer with either a 7 or 8-bit connection. (Default)

8 Characters are stored using 8-bit ASCII. The generated file can be printed only on a PostScript printer with an 8-bit connection.

Note: To print PostScript files generated by GDDM on your workstation-attached printer, you should download them to your workstation in **binary** format.

PSCNVCTL (CICS pseudoconversational control)

Nickname syntax: (PSCNVCTL,{**NO**|START|CONTINUE})

Specifies whether GDDM is to run in transaction-dependent pseudoconversational mode.

Notes:

1. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.
2. If transaction-independent pseudoconversation has been specified using the external default (CICTIF=EXT), PSCNVCTL must be set to NO.

Subsystems: CICS (both MVS and VSE)
Devices: Default family-1 display device only

NO Do not use pseudoconversational mode (the default).
START Start use of pseudoconversational mode.
CONTINUE Continue use of pseudoconversational mode.

SEGSTORE (retained or unretained mode)

Nickname syntax: (SEGSTORE,{**YES**|NO})

Indicates whether a 3270-PC/G, GX, or /AT workstation is to operate in retained or unretained mode.

Retained mode means that graphics segments are held in the display's segment buffers and are not re-sent from the host when a picture is redisplayed.

Unretained mode means that graphics segments are not held in the display's segment buffers. Segments have to be re-sent from the host to the display whenever a picture is updated.

Even if retained mode is specified, the device may be run in unretained mode if it is customized as being in output-only mode, or if there is not enough storage available in the device, or multiple graphics fields are being displayed.

Retained mode should be the preferred mode of operation because retained segments are required to perform functions locally.

However, if an application needs more segment storage than is available in the device, this can lead to continual switching between retained and unretained modes (with undesirable performance overhead). In such cases, it may be preferable to request unretained mode, and avoid the switching between modes.

Subsystems: All
Devices: Family-1 3270-PC/G, /GX, and /AT workstations

YES Retained mode (the default)
NO Unretained mode

SPECDEV (special device)

Nickname syntax: (SPECDEV,{IBM5080|*},{ddname|*})

Provides a token defining the type of special device and a namelist providing information specific to a type of special device.

Subsystems: TSO, VM
Devices: Family-1

An 8-byte special device name:

'IBM5080 ' To use the 5080 or 6090 Graphics System for graphics
'* ' To turn off the use of the IBM 5080 or 6090 Graphics System

Notes:

1. The device name, ddname, is the same as the name in the command that you issue before you use the 5080 or 6090 Graphics System.
   - Under CMS, the command is:

     ```
     FILEDEF ddname GRAF cuu
     ```

   - Under TSO, the command is:

     ```
     ALLOC FILE (ddname) UNIT (cuu)
     ```

   where in both cases, `cuu` is the address of the control unit for the display unit.
2. The use of a blank indicates DUM5080; that is, no actual 5080 or 6090 need be attached.

STAGE2ID (deferred device name-list for print utility)

Nickname syntax: (STAGE2ID,xxxxxxxx,xxxxxxxx,...)

Specifies the name-list for the device on which the print utility is to produce the output from a print file. The list of 8-byte name-parts defined in this group is passed (in the print file) to the print utility for use as its DSOPEN name-list parameter value.

For example, if a name-list of (*,aux-id) is specified, the print utility uses this in its DSOPEN call to access the auxiliary device attached to the session device.

If this processing option is not specified, the file is printed on the device specified in the original DSOPEN **name-list** parameter.

Under VM, this list is ignored if the ON parameter in the ADMOPUV command is specified (ON overrides the values specified in the list).

Subsystems: CICS, TSO, VM
Devices: Family-2

TOFILE (plot file output)

Nickname syntax: (TOFILE,{**NO**|YES},{**REP**|NOREP})

This option causes graphics output to be stored in GL (graphics language) format in auxiliary storage in a file or data set. GL files can be accepted by HP and HP-compatible plotters. The name of the file or data set is determined by the name-list parameter of the DSOPEN call (or as resolved by nickname processing). The user can specify whether an existing file or data set of the same name can be overwritten.

Subsystems VM, TSO, and MVS/Batch
Devices All family-1 plotters

Specifies whether or not plotter output is to be stored in a file:

NO Plotter output is not to be stored in file but drawn on an attached plotter device (default).

YES Plotter output is to be stored in a file identified by the DSOPEN namelist parameter (or nickname). It is not to be drawn directly on a plotter.

Specifies whether the new file can overwrite an existing file with the same name:

REP Overwrite existing file (default).
NOREP Do not overwrite existing file.

Notes:

1. A device token for a plotter device must be specified in the parameter list of the DSOPEN call.
2. If plot file output is specified, the **name-list** parameter of the final DSOPEN call defines the name parts that constitute the name by which the plot file created is to be known by the underlying subsystem. The data set characteristics of these plot files are described in the *GDDM Base Application Programming Guide*.

   The format of the name-parts is subsystem-dependent:

   Under VM, there are three elements in the array defined as follows:

   1 The CMS filename of the file. Can be any valid CMS filename.

   2 The CMS filetype of the file. Can be any valid CMS filetype.

   3 The CMS filemode of the file. Can be any valid CMS filemode. If not specified, a filemode of "A" is assumed. "*" is not allowed.

   Under TSO or MVS/Batch, the name-list parameter specifies either:

   - An allocated ddname. Plot file output is stored in the sequential data set or PDS member allocated to the specified ddname.

   or

   - A data set name (DSNAME). Plot file output is stored in the data set identified. The DSNAME is interpreted according to TSO naming conventions. If contained in quotes, it is taken as a fully qualified (complete) data set name. If it is not contained in quotes (that is, if it is partially qualified), the complete name is formed by prefixing the TSO qualifier in the normal way.

     If the output is to be stored as a member of a partitioned data set, the PDS and member name can be specified. For example:

     ```
     'USERID.MYPLOTS.GL(PIC1)'
     ```

     or

     ```
     MYPLOTS.GL(PIC2)
     ```

     where PIC1 and PIC2 are the names of the members to be created in the partitioned data set USERID.MYPLOTS.GL.

     If the name exceeds eight characters, it must be placed in consecutive elements of the namelist parameter. Each element contains eight characters of the name, including any quotes and brackets, with no embedded blanks.

   For information about file formats, refer to the *GDDM Base Application Programming Guide*.

3. The POSTPROC procopt can be used to perform postprocessing of the plot file output.

TSOINTRP (TSO CLEAR/PA1 protocol)

Nickname syntax: (TSOINTRP,{**PA1**|NONE})

Under TSO, an end user can usually interrupt a running program to contact the underlying supervisor. A GDDM application can choose, by this option, whether it requires this capability.

Notes:

1. This processing option is valid only for the TSO console. If it is specified from another device, this processing option causes an error. Nickname statements that specify this option should include the parameters `FAM=1` and `NAME=*`.
2. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.

Subsystems: TSO
Devices: Family-1

PA1 PA1 causes attention, CLEAR is ignored (TSO default action)
NONE PA1 and CLEAR are returned to the GDDM application (PA1 does not cause an attention).

TSORESHW (TSO reshow protocol)

Nickname syntax: (TSORESHW,n)

Specifies the Attention Identifier (AID) that signals that the display was corrupted (typically, by line-by-line output). It can be set to be either the default PA key or a PF key. Changing it to a PF key releases the default PA key for other use.

Any key functions specified in this option are not available to the application program. When pressed by the terminal user, the specified keys cause the current picture to be rebuilt and reshown.

Notes:

1. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.
2. This processing option is valid only for the TSO console. If it is specified from another device, this processing option causes an error. Nickname statements that specify this option should include the parameters `FAM=1` and `NAME=*`.

Subsystems: TSO
Devices: Family-1

Possible values of the keys treated as “reshow” AIDs are:

0 PA2 is treated as the “reshow” AID (the default)
1 – 24 The number of the PF key to be treated as the “reshow” AID.

WINDOW (window mode)

Nickname syntax: (WINDOW,{**NO**|YES})

Indicates whether the device is to be used for windowing. It allows the use of the WSCRT call to define a window on the device. Subsequent calls of DSOPEN for the same device (same device name-list) open virtual devices, which appear in the window.

Specifying that a device is to be windowed creates a default operator window, with an identifier of 0, and associates the device with the window. You do not have to use this window. Instead, you may prefer to use only windows that you explicitly create.

The use of the WINDOW processing option inhibits the use of real partitions.

Notes:

1. For a GDDM program running under the control of a task manager, if this processing option is specified for a virtual device, it is ignored, and the processing option for the real device is used instead.
2. When running under CICS with external default CICTIF=EXT specified, WINDOW mode must be set to NO.

Subsystems: CICS, TSO, VM
Devices: Family-1 displays, except the IBM 5080 Graphics System

The type of window mode:

NO Not in window mode (the default)
YES In window mode.

End of General-use programming interface

Appendix C. Device tokens supplied by GDDM

General-use programming interface

This appendix lists the device tokens that are supplied by GDDM. It is also possible to create your own. For information about this, see Chapter 10, “Creating your own device tokens” on page 101.

The GDDM-supplied device tokens are grouped as follows:

Device tokens for queriable terminals, plotters, and printers (family 1 or 2)

This set of token definitions is part of ADMLSYS1. The buffer code corresponds to the code in the **dev** parameter of the ADMM3270 macro. A device token of * is sufficient for printers if they are directly attached.

Locally attached 3179 Models G1 and G2, 3192-G, and 3472-G displays

Device token	Model	Screen size Rows by columns	Mouse	Tablet
L3179G	3179-G	32 by 80	No	No
L3179GM	3179-G	32 by 80	Yes	No
L3472G	3472-G	32 by 80	No	No
L3472GM	3472-G	32 by 80	Yes	No
L3472GT	3472-G	32 by 80	No	Yes

Locally attached 3270-PC displays

Device token	Model
L3270PC	3270-PC displays

Locally attached 3270-PC/G workstations

Device token	Model	Screen size Rows by columns	Mouse	Tablet
L5279A1	3270-PC/G	32 by 80	No	No
L5279A1M	3270-PC/G	32 by 80	Yes	No
L5279A1T	3270-PC/G	32 by 80	No	Yes
L5279A2	3270-PC/G	49 by 80	No	No
L5279A2M	3270-PC/G	49 by 80	Yes	No
L5279A2T	3270-PC/G	49 by 80	No	Yes
ADMKPCA1	3270-PC/G	32 by 80	No	No

(**See note 1 on page 371.**)

Locally attached 3270-PC/GX workstations

Device token	Model	Screen size Rows by cols	Mouse	Tablet	Color	Dual screen
L5379CS	3270-PC/GX	32 by 80	No	No	Yes	No
L5379CSM	3270-PC/GX	32 by 80	Yes	No	Yes	No
L5379CST	3270-PC/GX	32 by 80	No	Yes	Yes	No
L5379MS	3270-PC/GX	32 by 80	No	No	No	No
L5379MSM	3270-PC/GX	32 by 80	Yes	No	No	No
L5379MST	3270-PC/GX	32 by 80	No	Yes	No	No
L5379CD	3270-PC/GX	32 by 80	No	No	Yes	Yes
L5379CDM	3270-PC/GX	32 by 80	Yes	No	Yes	Yes
L5379CDT	3270-PC/GX	32 by 80	No	Yes	Yes	Yes
L5379MD	3270-PC/GX	32 by 80	No	No	No	Yes
L5379MDM	3270-PC/GX	32 by 80	Yes	No	No	Yes
L5379MDT	3270-PC/GX	32 by 80	No	Yes	No	Yes

Locally attached 3279 displays

Device token	Model
L79A2	3279–2
L79A3	3279–3

Remotely attached 3279 displays: (See note 2 on page 371.)

Device token	Model
R79A2	3279–2
R79A3	3279–3

Plotters attached to 3179 Models G1 and G2, 3192-G, and 3472-G displays

Device token	Model
L3179G80	6180
L3179G82	6182
L3179G84	6184
L3179G85	6185
L3179G86	6186
L3179G87	6187
L3G862	6186-2
L3G872	6187-2
L3179G71	7371
L3179G72	7372

Special plotter tokens for non-IBM plotters: (Similar to the IBM plotters listed above) attached to IBM 3179 Models G1 and G2, 3192-G, and 3472-G displays (**See note 3 on page 371.**)

| Device token
| L3179A71
| L3179A72
| L3179A80
| L3179A82
| L3179A84
| L3179A85
| L3179A86

Plotters attached to 3270-PC/G and /GX workstations

Device token	Plotter
L6180	6180
L6182	6182
L6184	6184
L6185	6185
L6186	6186
L6187	6187
L61862	6186-2
L61872	6187-2
L7371	7371
L7372	7372
L7374	7374
L7375	7375

Plotters attached to 5550-family workstations

Device token	Plotter
Device token	Plotter
L5550G71	7371
L5550G72	7372

Locally attached 3262, 3268, and 3287 printers

Device token	Model	Protocols	Page size Rows by columns	Comments
L68	3268	LU-3		
L68S	3268	LU-3	68 by 132	
L68Q	3268	LU-3	88 by 85	
L87	3287	LU-3		4-color only
L87S	3287	LU-1 (SCS)		4-color only
L3262	3262 belt printer			

Remotely attached 3287 printers: (See note 2 on page 371.)

Device token	Model	Protocols	Comments
R87	3287	LU-3	4-color only
R87S	3287	LU-1 (SCS)	4-color only

3812 Model 2 IPDS printers with 3270 attachment feature

Device token	Protocols	Page size Rows by columns	Paper
X3812A4	LU-0	93 by 82	A4
X3812Q	LU-0	88 by 85	Quarto (U.S. letter)
X3812L	LU-0	112 by 85	Legal
S3812A4	LU-1 (SCS)	93 by 82	A4
S3812Q	LU-1 (SCS)	88 by 85	Quarto (U.S. letter)

Device token	Protocols	Page size Rows by columns	Paper
S3812L	LU-1 (SCS)	112 by 85	Legal

3816 IPDS printer with 3270 attachment feature

Device token	Protocols	Page size Rows by columns	Paper
X3816A4	LU-0	93 by 82	A4
X3816Q	LU-0	88 by 85	Quarto (U.S. letter)
X3816L	LU-0	112 by 85	Legal
S3816A4	LU-1 (SCS)	93 by 82	A4
S3816Q	LU-1 (SCS)	88 by 85	Quarto (U.S. letter)
S3816L	LU-1 (SCS)	112 by 85	Legal

Image display

Device token	Display station
L3193	3193

Image display with attached scanner

Device token	Scanner
L319317	3117 flat-bed
L319318	3118 sheet-feed

3112, 3116, 3912, and 3916 IPDS printers

Device token	Protocols	Page size Rows by columns	Paper
X3912A4	LU-0	90 by 80	A4
X3912Q	LU-0	84 by 82	Quarto (U.S. letter)
X3912L	LU-0	108 by 82	Legal
S3912A4	LU-1 (SCS)	90 by 80	A4
S3912Q	LU-1 (SCS)	84 by 82	Quarto (U.S. letter)
S3912L	LU-1 (SCS)	108 by 82	Legal

4028 IPDS printers

Device token	Protocols	Page size Rows by columns	Paper
X4028A4	LU-0	90 by 80	A4
X4028Q	LU-0	84 by 82	Quarto (U.S. letter)
X4028L	LU-0	108 by 82	Legal
S4028A4	LU-1 (SCS)	90 by 80	A4
S4028Q	LU-1 (SCS)	84 by 82	Quarto (U.S. letter)
S4028L	LU-1 (SCS)	108 by 82	Legal

4224 printers

Device token	Protocols	RAM	Paper size Rows by columns	Loadable alphanumeric symbol sets
X4224SS	LU-0	64KB	68 by 132	No
X4224SE	LU-0	512KB	68 by 132	Up to 6
X4224QS	LU-0	64KB	88 by 85	No
X4224QE	LU-0	512KB	88 by 85	Up to 6
X4224A4S	LU-0	28KB	93 by 82	No
X4224A4E	LU-0	445KB	93 by 82	No

Device token	Protocols	RAM	Paper size Rows by columns	Loadable alphanumeric symbol sets
S4224SS	LU-1 (SCS)	64KB	68 by 132	No
S4224SE	LU-1 (SCS)	512KB	68 by 132	Up to 6
S4224QS	LU-1 (SCS)	64KB	88 by 85	No
S4224QE	LU-1 (SCS)	512KB	88 by 85	Up to 6
S4224A4S	LU-1 (SCS)	28KB	93 by 82	No
S4224A4E	LU-1 (SCS)	445KB	93 by 82	No

4230 printers: (See note 4.)

Device token	Protocols	RAM	Paper size Rows by columns	Loadable alphanumeric symbol sets
X4230S	LU-0	128KB	68 by 132	No
X4230Q	LU-0	128KB	88 by 85	No
X4230A4	LU-0	128KB	93 by 82	No
S4230S	LU-1 (SCS)	128KB	68 by 132	No
S4230Q	LU-1 (SCS)	128KB	88 by 85	No
S4230A4	LU-1 (SCS)	128KB	93 by 82	No

4234 printers: (See note 4.)

Device token	Protocols	RAM	Paper size Rows by columns	Loadable alphanumeric symbol sets
X4234S	LU-0	512KB	68 by 132	Up to 4
X4234Q	LU-0	512KB	88 by 85	Up to 4
X4234A4	LU-0	512KB	93 by 82	Up to 4
S4234S	LU-1 (SCS)	512KB	68 by 132	Up to 4
S4234Q	LU-1 (SCS)	512KB	88 by 85	Up to 4
S4234A4	LU-1 (SCS)	512KB	93 by 82	Up to 4

Notes:

1. Generated for use by the ADMUPC utility for dummy devices.
2. PS compression is specified, for these controlling attached devices.
3. Instead of PLOTTER, these device tokens use PLOTNAO, which specifies that the plotter is *not* an auxiliary-only plotter. These tokens are intended for use only in circumstances where incompatibilities between the plotter and the display terminal prevent GDDM from querying the plotter device. The normal PLOTTER device tokens cannot be used because their the “auxiliary-only” attribute prevents the attachment of the primary device.
4. The device tokens used with LU-1 (SCS) protocols require the printer to be set to 10 characters per inch and 8 lines per inch.

Device tokens for Kanji devices and 3290 displays (family 1)

This set of token definitions is part of ADMLSYS1.

Japanese displays and printers

Device token	Model	Protocols
K78A2	3278-2 display	
K83S	3283 printer	LU-1 (SCS)
K83	3283-2 printer	LU-1 (SCS)

Japanese 5550-family workstations (nongraphics)

Device token	Model	Protocols
L5550A	5550 display	
L5553A	5553 printer	LU-3
L5553AI	5553 printer	LU-1 (SCS)
L5550G4	3270-PC Version 4.0	
L5550H4	3270-PC Version 4.0 color	
L5553B34	5553 printer	LU-3
L5553BI4	5553 printer	LU-1 (SCS)

Japanese 5550-family workstations (graphics)

Device token	Model	Font
L5550GC2	3270-PC/G version 2 display	16 × 24
L5550GH2	3270-PC/G version 2 display	24 × 24
L5550GC3	3270-PC/G version 3 display	16 × 24
L5550GH3	3270-PC/G version 3 display	24 × 24
L5550GC5	3270-PC/G version 5 display	16 × 16
L5550GH5	3270-PC/G version 5 display	24 × 24

3290 displays with APL, 16 partitions, whole screen and variable cell sizes

Device token	Model	Screen size Rows by columns	Cellsize
ADMK9020	model 2	24 × 80	9 × 16
ADMK9030	model 3	32 × 80	9 × 16
ADMK9040	model 4	43 × 80	9 × 16
ADMK9050	model 5	27 × 132	7 × 16
ADMK9060	model 6	62 × 160	6 × 12

Device tokens for nonqueriable terminals and printers (family 1 or 2)

This set of token definitions is part of ADMLSYS1.

3277 display terminals

Device token	Model
ADMK7710	3277 Model 1
ADMK771A	3277 Model 1 with APL
ADMK7720	3277 Model 2
ADMK772A	3277 Model 2 with APL

3278 display terminals

Device token	Model
ADMK7810	3278 Model 1
ADMK781A	3278 Model 1 with APL
ADMK7820	3278 Model 2
ADMK782A	3278 Model 2 with APL
ADMK7830	3278 Model 3
ADMK783A	3278 Model 3 with APL
ADMK7840	3278 Model 4
ADMK784A	3278 Model 4 with APL
ADMK7850	3278 Model 5
ADMK785A	3278 Model 5 with APL

Nonqueriable printers

Device token	Model
ADMKQUEP	default token for family-2 (queued) printers
ADMK8710	3287 Model 1
ADMK871A	3287 Model 1 with APL

Device tokens for ASCII devices (family 1)

These definitions are for ASCII graphics devices. This set of token definitions is part of ADMLSYSA.

Device token	Model	Screen size Rows by columns	Screen size Pixels	Mouse	Graphics colors
DEC240	DEC VT240	24 by 80	240 by 800	No	None
DEC241	DEC VT241	24 by 80	240 by 800	No	4
DEC330	DEC VT330	24 by 80	480 by 800	No	None
DEC330M	DEC VT330	24 by 80	480 by 800	Yes	None
DEC340	DEC VT340	24 by 80	480 by 800	No	16
DEC340M	DEC VT340	24 by 80	480 by 800	Yes	16
TEK4105	Tektronix 4105	30 by 80	360 by 480	No	8
TEK4205	Tektronix 4205	30 by 80	360 by 480	No	16
TEK4205M	Tektronix 4205	30 by 80	360 by 480	Yes	16
TEK4207	Tektronix 4207	32 by 80	480 by 640	No	16
TEK4207M	Tektronix 4207	32 by 80	480 by 640	Yes	16
TEK4208	Tektronix 4208	32 by 80	480 by 640	No	16
TEK4208M	Tektronix 4208	32 by 80	480 by 640	Yes	16
TEK4209	Tektronix 4209	32 by 80	480 by 640	No	16
TEK4209M	Tektronix 4209	32 by 80	480 by 640	Yes	16

Device tokens for GDDM-PCLK displays, printers, and plotters

This set of token definitions is part of ADMLSYS1.

GDDM-PCLK workstation configurations

Device token	Graphics card	Rows by columns	Pixels	Graphics colors
LPCM	CGA	24 by 80	640 by 200	None
LPCC1	EGA	24 by 80	640 by 200	16
LPCC2	EGA	24 by 80	640 by 350	16
LPCC3	PS/2 display adapter	24 by 80	640 by 480	16
LPCC4	PS/2 display adapter 8514/A	24 by 80	1024 by 768	16
LPCC5	PS/2 display adapter MCGA	24 by 80	640 by 480	2

GDDM-PCLK workstation configurations with a locally attached printer

Device token	Printer
LPC3852	3852 color ink-jet
LPC4201	4201 Proprinter
LPC42012	4201 Model 2 Proprinter
LPC4202	4202 Proprinter XL
LPC4207	4207 Proprinter X24
LPC4208	4208 Proprinter XL24
LPC5152	5152 monochrome graphics
LPC5182	5182 color impact
LPC5201	5201 Quietwriter
LPC5202	5202 Quietwriter III

GDDM-PCLK workstation configurations with a locally attached plotter

Device token	Plotter	Pens
LPC7371	7371	2
LPC7372	7372	6
LPC7374	7374	8
LPC7375	7375	8
LPC6180	6180	8
LPC6182	6182	8
LPC6184	6184	8
LPC6185	6185	8
LPC6186	6186	8
LPC6187	6186	8
LPC61862	6186-2	8
LPC61872	6187-2	8

Device tokens for system printers (family 3)

This set of token definitions is part of ADMLSYS3.

System printers

Device token	Comments
ADMKSYSP	Default for non-3800 printers

1403 printers

Device token	Model	Rows by columns	Lines per inch
S1403N6	1403	66 by 85	6
S1403N8	1403	88 by 85	8
S1403W6	1403	66 by 132	6
S1403W8	1403	88 by 132	8

3800 printers

Note: Device tokens for 3800 printers can also be used for 3812 and 3820 printers.

Device token	Model	Rows by columns	Lines per inch
S3800N6	3800	60 by 85	6
S3800N8	3800	80 by 85	8
S3800N12	3800	120 by 85	12
S3800W6	3800	60 by 136	6
S3800W8	3800	80 by 136	8
S3800W12	3800	117 by 136	12
S3800N6S	3800	45 by 110	6
S3800N8S	3800	60 by 110	8
S3800W6S	3800	45 by 136	6
S3800W8S	3800	60 by 136	8

Device tokens for cell-based AFPDS page printers (family 4)

This set of token definitions is part of ADMLSYS4. These tokens support alphanumeric and alternate device functions and are designed to facilitate production of family-4 AFPDS output from applications that normally produce family-1 or family-2 print output. The tokens define a font and code page to be used for alphanumerics that can be modified to suit your installation. For more information, see Chapter 10, “Creating your own device tokens” on page 101.

3800 printer at 240 ppi: IM: (See note 1 on page 378.)

Device token	Rows by columns	Pixels	Lines per inch
A3800S	60 by 110	1800 by 2640	8
A3800Q	85 by 82	2550 by 1968	8
A3800A4	90 by 80	2700 by 1920	8
B3800S	45 by 110	1800 by 2640	6
B3800Q	63 by 82	2520 by 1968	6
B3800A4	67 by 80	2680 by 1920	6

3812 and 3816 printers at 240 ppi: GOCA, IOCA

Device token	Rows by columns	Pixels	Lines per inch
A3816Q	85 by 82	2550 by 1968	8
A3816A4	90 by 80	2700 by 1920	8
B3816Q	63 by 82	2520 by 1968	6
B3816A4	67 by 80	2680 by 1920	6

3812 and 3816 printers supported by VM3812

Device token	Rows by columns	Pixels	Lines per inch
AVM38Q	85 by 82	2550 by 1968	8
AVM38A4	90 by 80	2700 by 1920	8
BVM38Q	63 by 82	2520 by 1968	6
BVM38A4	67 by 80	2680 by 1920	6

3820, 3827, and 3828 printers at 240 ppi: IM

Device token	Rows by columns	Pixels	Lines per inch	Comments
A3820Q	85 by 82	2550 by 1968	8	
A3820A4	90 by 80	2700 by 1920	8	
B3820Q	63 by 82	2520 by 1968	6	
B3820A4	67 by 80	2680 by 1920	6	
J3820Q	63 by 82	2520 by 1968	6	DBCS Kanji support
J3820A4	67 by 80	2680 by 1920	6	DBCS Kanji support

3825 printer at 240 ppi: GOCA, IOCA: (See note 2 on page 378.)

Device token	Rows by columns	Pixels	Lines per inch
A3825Q	85 by 82	2550 by 1968	8
A3825A4	90 by 80	2700 by 1920	8
B3825Q	63 by 82	2520 by 1968	6
B3825A4	67 by 80	2680 by 1920	6

3835 printer at 240 ppi: GOCA, IOCA: (See note 2 on page 378.)

Device token	Rows by columns	Pixels	Lines per inch
A3835S	60 by 110	1800 by 2640	8

Device token	Rows by columns	Pixels	Lines per inch
B3835S	45 by 110	1800 by 2640	6

3112, 3116, 3912, 3916, 4028 printer at 300 ppi: GOCA, IOCA

Device token	Rows by columns	Pixels	Lines per inch
A4028Q	84 by 82	3150 by 2460	8
A4028A4	90 by 80	3375 by 2400	8
B4028Q	63 by 82	3150 by 2460	6
B4028A4	67 by 80	3350 by 2400	6

4224, 4230 and 4234 printers at 144 ppi: GOCA, IM

Device token	Rows by columns	Pixels	Lines per inch
A4224S	85 by 132	1530 by 1901	8
A4224Q	84 by 82	1512 by 1181	8
A4224A4	90 by 80	1620 by 1152	8
B4224S	63 by 132	1512 by 1901	6
B4224Q	63 by 82	1512 by 1181	6
B4224A4	67 by 80	1608 by 1152	6

Notes:

1. IBM 3800 device tokens apply to IBM 3800 Model 3, Model 6, Model 8, and 3900 printers.
2. These device tokens are for printers that have the Advanced Function Image and Graphics Feature. For 3825 and 3835 printers without this feature, use the 3800 or 3820 Device Tokens.

Device tokens for PostScript printers (family 4)

This set of token definitions is part of ADMLSYS4.

Notes:

1. All tokens in this table produce PostScript output files.
2. The following device tokens produce output with a line width of 2 by default. A copy of these tokens is also supplied with a suffix of "3" so that PostScript output with slightly bolder lines can be produced. For example, using device token P4079B3 gives the same characteristics as device token P4079B, but produces output for a 4079 printer (paper size B) with a line width of 3. See the LINEW parameter under "Creating your own PostScript device tokens with the ADMMPSCR macro" on page 120.

Generic level-1 monochrome printer

Device token	Paper size–inches width by depth	Resolution in ppi	Type	Paper
PPS1MA4	8.0 × 11.7	300	PS1M	A4
PPS1MQ	8.5 × 11.0	300	PS1M	Quarto

Generic level-2 monochrome printer

Device token	Paper size–inches width by depth	Resolution ppi	Type	Paper
PPS2MA4	8.0 × 11.7	300	PS2M	A4
PPS2MQ	8.5 × 11.0	300	PS2M	Quarto

Generic level-1 color printer

Device token	Paper size–inches width by depth	Resolution ppi	Type	Paper
PPS1CA4	8.0 × 11.7	300	PS1C	A4
PPS1CQ	8.5 × 11.0	300	PS1C	Quarto

Generic level-2 color printer

Device token	Paper size–inches width by depth	Resolution ppi	Type	Paper
PPS2CA4	8.0 × 11.7	300	PS2C	A4
PPS2CQ	8.5 × 11.0	300	PS2C	Quarto
PPS2CA3	8.0 × 16.0	360	PS2C	A3
PPS2CB	11.0 × 17.0	360	PS2C	B

4029 monochrome printer

Device token	Paper size–inches width by depth	Resolution ppi	Type	Paper
P4029A4	8.0 × 11.7	600	PS1M	A4
P4029Q	8.5 × 11.0	600	PS1M	Quarto

4079 color printer

Device token	Paper size–inches width by depth	Resolution ppi	Type	Paper
P4079A3	8.0 × 16.0	360	PS1C	A3
P4079B	11.0 × 17.0	360	PS1C	B

Device tokens for page printers (family 4)

This set of token definitions is part of ADMLSYS4.

Note: All tokens in this table produce AFPDS-type output files for processing by PSF, unless stated otherwise.

3800, 3812, 3816, or 3820 printer

Device token	Paper size–inches width by depth	Resolution ppi	Pixels per unit line width	Paper
IMG120	8.5 × 11.0	120	3	Quarto
IMG1201	8.5 × 11.0	120	1	Quarto
FINE120	13.9 × 12.5	120	1	
EXECUTIV	7.5 × 10.5	240	3	
A4	8.3 × 11.7	240	3	A4
P38PPN1	8.5 × 10.0	240	1	
P38PPN3	8.5 × 10.0	240	3	
IMG240	8.5 × 11.0	240	3	Quarto
IMG2401	8.5 × 11.0	240	1	Quarto
LETTER	8.5 × 11.0	240	3	Quarto
LEGAL	8.5 × 14.0	240	3	Legal
FINE240	13.9 × 12.5	240	1	
IMG240X	13.9 × 12.5	240	3	

3112, 3116, 3912, 3916, 4028 printers

Device token	Paper size–inches width by depth	Resolution ppi	Pixels per unit line width	Paper
IMG144A4	8.3 × 11.7	144	1	A4
IMG144Q	8.5 × 11.0	144	1	Quarto
IMG144S	13.2 × 8.5	144	1	

4250 printer: (See notes 1 on page 381 and 2 on page 381.)

Device token	Paper size–inches width by depth	Resolution ppi	Pixels per unit line width	Paper
ADMKHRIG	8.5 × 11.0	600	6	
IMG85	8.5 × 11.0	600	6	
IMG117	11.7 × 10.0	600	6	
IMG600X	11.7 × 14.0	600	6	
FINE600	11.7 × 14.0	600	1	
IMGA3X	11.7 × 16.5	600	6	A3
IMG600Y	17.0 by 11.0	600	6	

Canonical (unformatted) bit image output: (See notes 3 on page 381 and 4 on page 381.)

Device token	Pixels
CAN512	512 × 512
CAN1024	1024 × 1024

Notes:

1. ADMKHRIG is the default device token for family 4 (page) printers.
2. The IMGxxxx device tokens produce CDPF-type output files for processing by CDPF.
3. The CAN512 and CAN1024 device tokens produce unformatted (bit-map) output.
4. The values given in DSOPEN's processing option groups 5, 8, and 9 are overridden when the CAN512 and CAN1024 device tokens are used.

End of General-use programming interface

Appendix D. Name-lists

General-use programming interface

This appendix describes name-lists. Name-lists identify the physical device that is to be opened for use by a GDDM application program.

Name-lists can be specified on DSOPEN calls and on nicknames, which are described in Chapter 18, “Nickname user-default specifications” on page 205.

Reserved names “*” and blanks

In all environments, for all families, there is a convention for two reserved values of the name-list(1) field.

- When this field is specified but is “*”, the terminal used is as described under the options below for a name-count of 0, where this is valid. In other words, this is an explicit way to specify the default device name.
- When the field contains blanks, the device is a **dummy** one, that is, no real device is associated with this GDDM device. GDDM generates the data streams required but does not send them to any real device, nor does it try to receive data from a device.

  This option can be used to check a GDDM application when a real device with the necessary features is unavailable, or it can be used with the FSSAVE mechanism (see the *GDDM Base Application Programming Reference* book) to generate SAVE files for a device that is unavailable when the application is to be run.

  When this option is selected, the application program must provide a device token parameter to supply the device characteristics that are to be used by GDDM.

Family-1 name-list

Under all subsystems, the device name can specify the user console:

- By omitting the name list
- By setting all name-parts to “*”.

Also, (under CICS, TSO, or VMS), the name-list can identify an auxiliary device, such as a plotter that is attached to a 3270-PC/G, /GX, or /AT workstation, or a printer or plotter that is attached to a workstation using GDDM-PCLK or GDDM-OS/2 Link. In such a case, **name-list(1)** identifies the 3270-PC/G, /GX, or /AT, or GDDM-PCLK workstation, and **name-list(2)** (other than “*”) identifies the auxiliary device (the plotter or printer). GDDM uses this name to identify the appropriate port on the attaching workstation.

Notes:

1. The name given in ***name-list(2)*** must be the same as the name given in the IEEE customization panel when the 3270-PC/G, /GX, or /AT workstation was set up. (This is not the same as the device type which must be of the form "IBMnnnn".)
2. A ***name-list(2)*** value of "ADMPLOT" has a special meaning. In this case, GDDM uses the first plotter defined in the IEEE customization panel when the 3270-PC/G, /GX, or /AT workstation was set up, regardless of the configured name.
3. In the case of GDDM-PCLK 1.1, only one plotter can be configured, so ADMPLOT should always be used. The special value ADMPCPRT should be used to open a PCLK-attached printer.
4. Use ADMPMOP for the GDDM-OS/2 default.
5. Printers attached to 3192-G and 3472-G devices are supported as separate devices with their own name. They are not considered auxiliary devices, which are addressed via their attaching workstation.

CICS name-list

Family-1 – 3270 terminals
The name-count value must be 0, 1, or 2:

0 The device used is that identified by the terminal control table (TCT) for the transaction.
1 Name-list(1) must contain either "*" or blanks. If it contains "*", the terminal is used as described for a name-count of 0.
2 Name-list(1) must contain either "*" or blanks.

If name-list(2) contains "*", the terminal is used as described for a name-count of 1. Otherwise, the name-list(2) value is the name of an auxiliary device (a plotter or printer).

Family-2 – queued printer
The name-count value must be 1.

The name-list(1) value is the terminal identifier of the printer in the TCT.

Family-3 – system printer
The name-count value must be either 0 or 1:

0 A name is taken from the GDDM defaults. The supplied default is ADMS.
1 A name is taken from name-list(1).

The name is assumed to be the name of a transient data destination that can route the output to a subsystem printer. The transient data destination should be one defined in the CICS destination control table.

When name-list(1) contains "*", the printer is used as described for a name-count of 0.

Family-4 – page-printer files
Not applicable under CICS.

VSE/Batch name-list

Only applicable to family-4 files.

Family-4 – page-printer files
The name-count value must be 1.

The name-list(1) value must contain the DLBL file-name of 7 characters.

IMS name-list

Family-1 – 3270 terminals
The name-count value must be either 0 or 1:

0 An LTERM name is taken from the LTERM field of the I/O PCB.
1 An LTERM name is taken from name-list(1).

There must be at least one TP PCB whose destination is set to the LTERM name.

If name-list(1) contains "*", the terminal is used as described for a name-count of 0.

Family-2 – queued printers
The name-count value must be 1.

The name-list(1) value is an LTERM name. This LTERM must be for a 3270-family printer. There must be at least one TP PCB whose destination is set to the name of the GDDM-supplied print utility transaction.

Family-3 – system printers
The name-count value must be either 0 or 1:

0 An LTERM name is taken from the GDDM defaults. The supplied default is ADMLIST.
1 An LTERM name is taken from name-list(1).

There must be at least one TP PCB whose destination is set to the LTERM name. This LTERM must be for a SPOOL printer.

If name-list contains "*", the printer is used as described for a name-count of 0.

Family-4 – page-printer files
Not applicable under IMS.

TSO name-list

Family-1 – 3270 terminals
The name-count value must be 0, 1, or 2:

0 The device is the TSO console from which the application is being run.

1 Name-list(1) must contain either "*" or blanks. If it contains "*", the terminal is used as described for a name-count of 0.

2 Name-list(1) must contain either "*" or blanks. If name-list(2) contains "*", the terminal is used as described for a name-count of 1. Otherwise, the name-list(2) value is the name of an auxiliary device (a plotter or printer).

Note: For detail of the name-list value required when plotter output is to be saved in IBM-GL format, see the description of the TOFILE processing option in Appendix B, "Processing options" on page 333.

Family-2 – queued printers
The name-count value must be 1.

The name-list(1) value is the device identifier of the printer. This device identifier must be one of the names in the Master Print Queue data set of the GDDM print utility. Under VTAM, the device identifier must be included in SYS1.VTAMLIST.

Family-3 – system printers
The name-count value must be either 0 or 1:

0 A ddname for a SYSOUT file is taken from the GDDM defaults. The supplied default is ADMLIST.
1 A ddname for a SYSOUT file is taken from name-list(1).

If name-list(1) contains "*", the printer is used as described for a name-count of 0.

Family-4 – page-printer files
The name-count value must be in the range 1 through 7.

The name-list value defines:

- The DDNAME for a sequential file or member of a partitioned dataset. The DDNAME must be already allocated to the sequential file or partitioned dataset and member.
- The DSNAME for a sequential file.
- The DSNAME of a partitioned dataset and the member name, in the form pdsname(membername).

More than one data set is generated if a color master table is being used (as specified by processing option group 3000).

By allocating the DDNAME to a suitable SYSOUT class, family-4 output can be sent directly to a print server, such as PSF/MVS.

Monochrome master

The name-list value must be of one of these:

* A name is taken from the GDDM external default TSOMONO. The supplied default is ADMIMAGE.

  The inferred name is searched for as a DDNAME. If it cannot be found as a DDNAME, it is formed into a DSNAME of the form "qualifier(s).name" (where qualifier(s) is the active dsn-prefix, or user ID, or both of these).

name1.name2[.name3..] , 'name1[.name2. name3..]' or name1.name2[(membername)]

The specified name is taken as a DSNAME, according to TSO naming conventions. Unless contained in quotes, the specified name must contain one (and only one) component of ".*". Whether contained in quotes or not, if any one component of the name is "*", that component is replaced with a value taken from the GDDM defaults. The supplied default is ADMIMAGE.

If contained in quotes, the name is taken as a complete DSNAME. If not contained in quotes, it is formed into a complete DSNAME of the form:

```
'qualifier(s).name1.name2...'
```

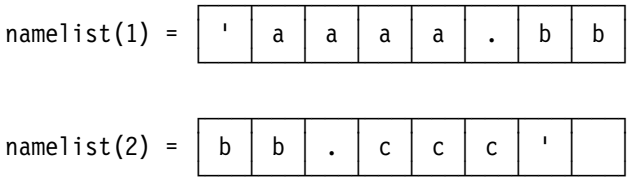
where qualifier(s) is the active dsn-prefix, or user ID, or both of these.

If the specified name is longer than 8 characters, it must be placed in consecutive members of the array, and, if necessary, padded with blanks.

For example, if the DSNAME is contained in quotes and is:

```
aaaa.bbbb.ccc
```

then it would look like this:



In PL/I, a string can be overlaid on the array to simplify this (but the name-count must still specify the number of 8-byte tokens).

Color masters

The name-list value must be of one of these:

* A value is taken from the GDDM defaults. The supplied default is ADMCOL+. The "+" is replaced by 1, 2, 3, and so on (up to a maximum of 9) for each color master data set.

  The first derived name (for example, ADMCOL1), is searched for as a ddname. If it is found as a ddname, all the other derived names must also exist as ddnames. If it cannot be found as a ddname, all the derived names are formed into DSNAMEs of the form:

  ```
  'qualifier(s).name'
  ```

  where qualifier(s) is the active dsn-prefix, or user ID, or both of these.

name1.name2[.name3..] or 'name1[.name2.name3..]'

The specified name is taken to identify DSNAMEs, according to TSO naming conventions. The specified name must contain one (and only one) component of "*". That component is replaced with a value taken from the GDDM defaults. The supplied default is ADMCOL+. The "+" is replaced by 1, 2, 3, and so on, (up to a maximum of 9) for each color master data set.

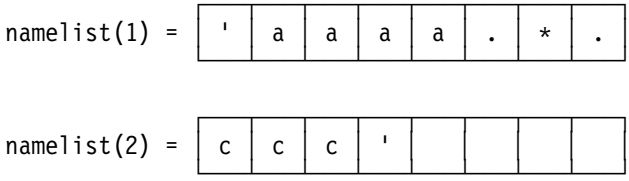
If contained in quotes, the derived names are taken as complete DSNAMEs. If not contained in quotes, they are formed into complete DSNAMEs of the form "qualifier(s).name1.name2..." (where qualifier(s) is the active dsn-prefix, or user ID, or both of these).

If the specified name is longer than 8 characters, it must be placed in consecutive members of the array, and, if necessary, padded with blanks.

For example, if the DSNAME is contained in quotes and is

```
aaaa.*.ccc
```

where "*" is replaced by ADMCOL1, ADMCOL2, and so on, then it would look like this:



In PL/I, a string can be overlaid on the array to simplify this (but the name-count must still specify the number of 8-byte tokens).

In this example, the derived DSNAMEs when using a color table specifying four-color masters would be:

```
'aaaa.ADMCOL1.ccc'
'aaaa.ADMCOL2.ccc'
'aaaa.ADMCOL3.ccc'
'aaaa.ADMCOL4.ccc'
```

MVS/Batch name-list

Family-1 – 3270 terminals
The name-count value must be 1.

Name-list(1) must either contain blanks or the TSOEMUL external default must be set to “YES”.

Note: For detail of the name-list value required when plotter output is to be saved in IBM-GL format, see the description of the TOFILE processing option in Appendix B, “Processing options” on page 333.

Family-2 – queued printers
The name-count value must be 1.

The name-list(1) value is the device identifier of the printer. This device identifier must be one of the names in the Master Print Queue data set of the GDDM print utility. Under VTAM, the device identifier must be included in SYS1.VTAMLIST.

Family-3 – system printers
The name-count value must be either 0 or 1:

0 A ddname for a SYSOUT file is taken from the GDDM defaults. The supplied default is ADMLIST.
1 A ddname for a SYSOUT file is taken from name-list(1).

If name-list(1) contains “*”, the printer is used as described for a name-count of 0.

Family-4 – page-printer files
The name-count value must be 1 through 6.

The name-list value defines the DDNAME(s) or DSNAME(s) of the data set(s) that will be generated. More than one data set is generated if a color master table is being used (as specified by processing option group 3000).

SYSOUT(n) is a valid setting.

Monochrome master

The name-list value must be of one of these:

* A name is taken from the GDDM defaults. The supplied default is ADMIMAGE.

  The inferred name is searched for as a DDNAME.

  For further details about MVS/BATCH name-list, refer to the TSO name-list section.

VM name-list

Family-1 – IBM 3270 terminals
The name-count value must be 0, 1, or 2:

0 The device is the CMS console from which the application is being run.
1 Name-list(1) must contain one of these:
- "*"
- Blanks
- "PUNCH"
- A character form of device address (for example "061").

If name-list(1) contains "*", the terminal is used as described for a name-count of 0.

If name-list(1) = "PUNCH", GDDM writes the 3270 device output to the CMS virtual punch, in the form described in *GDDM Base Application Programming Guide*. In this case, the application must provide a device token parameter to supply the device characteristics that are to be used by GDDM.

2 Name-list(1) must contain one of these:
- "*"
- Blanks
- A character form of device address (for example "061").

If name-list(2) contains "*", the terminal is used as described for a name-count of 1. Otherwise, the name-list(2) value is the name of an auxiliary device (a plotter or printer).

Note: For detail of the name-list value required when plotter output is to be saved in IBM-GL format, see the description of the TOFILE processing option in Appendix B, "Processing options" on page 333.

Family-2 – queued printers
The name-count value must be in the range 1 through 3.

Unless processing option group 1004 (INVKOPUV) is specified, the name-list(1), name-list(2), and name-list(3) values define the file name, file type, and file mode (respectively) of the print file that is to be generated. The supplied default for filetype is ADMPRINT. Filemode defaults to A1.

If automatic invocation of the VM Print Utility is requested (as specified by INVKOPUV), name-count and name-list identify a family-1 device, and must therefore be as defined for family-1 (above).

Family-3 – system printers
The name-count value must be 0, 1, 2, or 3:

0 The device is the currently defined printer; that is, device 00E.
1 through 3 Name-list(1), name-list(2), and name-list(3) define the file name, file type, and file mode (respectively) of the print file that is to be generated. The supplied default for filetype is ADMLIST. Filemode defaults to A1.

When name-list(1) contains "*", the printer is used as described for a name-count of 0.

Family-4 – page-printer files
The name-count value must be in the range 1 through 3.

The name-list(1), name-list(2), name-list(3) values define the file name, file type, and file mode, respectively, of the CMS file(s) that is generated. More than one file

is generated if a color master table is being used (as specified by processing option group 3000).

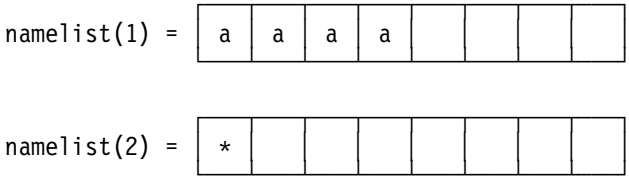
For both monochrome and multicolor masters, “A1” is assumed if the filemode is omitted.

A family-4 namelist value of “PRINTER” causes subsequent output to be directed to the virtual printer, instead of to a file. It is intended for use only with AFPDS files, and takes effect only if the name-count value is “1.” If a namelist value of “PRINTER” is used with non AFPDS family-4 files, errors are likely to occur.

Monochrome master
When the filetype is omitted or is specified as “*”, the filetype is taken from the GDDM external default CMSMONO. The supplied default is ADMIMAGE.

Color masters
If only one color master is specified in the MASTERS option of the ADMMCOLT macro, the filetype can be explicitly given. Otherwise, when the filetype is specified, it must be “*”. The filetype is taken from the GDDM defaults. The supplied default is ADMCOL+. The “+” is replaced by 1, 2, 3, and so on (up to a maximum of 9) for each color master file.

For example, if:



the derived file identifiers when using a color table specifying four-color masters would be:

```
aaaa ADMCOL1 A1
aaaa ADMCOL2 A1
aaaa ADMCOL3 A1
aaaa ADMCOL4 A1
```

End of General-use programming interface

Appendix E. APL and GDDM/MVS or GDDM/VM

General-use programming interface

APL uses GDDM for its character handling and its graphics. The APL feature installed on display devices or printers can be one of two types:

- APL-Data-Analysis feature number 1066

  This is available on devices such as the 3277-2, 3284-2, 3286-2, and 3287 attached to 3271 or 3272 controllers.

- APL/Text feature number 1120

  This is available on devices such as the 3179-G, 3276, 3278, 3279, 3287, 5279, and 5379 attached to 3274 controllers.

  APL/Text is available with the more recent devices that support programmed symbols, color, extended highlighting, and alternate sizes. Only devices that support “Read Partition Query” can use APL/Text.

GDDM needs to be able to distinguish between these two features. GDDM can discover which APL feature is installed on most devices by querying the device or by checking subsystem tables. However, GDDM cannot do this for:

- Nonqueriable, non-LU1 printers under TSO

  For these printers, GDDM selects the appropriate APL feature by checking the bind image (specified on the PSERVIC operand of the MODEENT macro for VTAM). If the default and alternate screen sizes are both defined (regardless of their values), GDDM selects APL/Text. If they are not both defined, GDDM selects APL-Data Analysis.

- Nonqueriable display terminals under TSO

  For these terminals, GDDM indirectly checks the bind image (PSERVIC operand of the MODEENT macro for VTAM). If the default and alternate screen sizes are different, the APL/Text feature is selected. If they are the same, or if the alternate screen size is omitted, GDDM uses the value specified on the TSOAPLF external default. (TSO inserts a dummy alternate value of the same size as the default value if the alternate size is omitted.)

- Nonqueriable printers under VM/CMS

  For these printers, GDDM uses the value specified on the CMSAPLF external default.

Notes:

1. APL2 users should use code page 00351. You can set this as the installation default (as described in Chapter 16, “GDDM’s code-page support” on page 159.) Alternatively, (if you want to use a CECP code page as the installation default, for example) notify APL2 users that they must use code page 00351.
2. Some terminal types might not work satisfactorily with APL2. For example, the additional APL characters available with APL2 are not supported by the APL-Data-Analysis feature, and the 3278 and 3279 terminals do not support keyboard input of the APL2 characters.

3. If you have a mixture of terminal types that have different types of APL feature, and the APL feature for those devices is identified to GDDM on the TSOAPLF or CMSAPLF external default, multiple specifications of the default might be required. That is, particular users or groups of users might require their own external defaults modules or files that identify the APL feature for the devices they are using.

   Possible combinations that might cause this situation are: a 3277-2 display station and a nonqueriable 3278-2 display station under MVS; or 3286 and 3287 printers attached to a 3271 or 3272, combined with a nonqueriable 3287 attached to a 3274 under VM.

4. Under MVS, you do not need to link-edit APL again if you are upgrading from GDDM Version 1 Release 3 or Release 4, or from any release of GDDM Version 2.

5. Under VM, the APL saved segment might need to be saved again, unless GDDM 3.1 is installed using the same shared segment names that APL was using previously.

End of General-use programming interface

Appendix F. The GDDM Object Import/Export utility (IMS)

The purpose of the GDDM Object Import/Export Utility is to enable objects to be transferred between (1) GDDM applications running on one IMS system, and (2) those running on either another IMS system, or in a totally different environment (for example, a TSO development system).

Note: ADMSAVE objects produced under IMS can be used only under IMS. Other GDDM objects are, however, transferable between IMS and other systems.

The utility runs either as an IMS type-3 Batch Processing Program (BPP), or as a type-2 Batch Message Processing Program (BMP). It uses two instances of GDDM: one, reached by means of the stub that is link-edited with it, operates in an IMS environment; the other, which is dynamically loaded, provides the environment-dependent code normally used in an MVS environment, which is used to access the partitioned data sets (PDSs).

The program specification block (PSB) for the utility must specify CMPAT=YES, so that there will always be an I/O program communication block (PCB), even when the utility is being executed as a BPP. The same PSB can be used for BPP and BMP execution — the database PCB will always be the second one.

The PSBs required for this transaction are described in the *GDDM/MVS Program Directory*.

When run as a BPP, the utility can be used, with a suitable PSB, to perform the initial database load, although the input must be in alphabetical order of object name and type. If this is inconvenient, the initial load run can just refer to a single object (possibly a dummy), and the remainder can be specified on a subsequent update run.

Four PDSs can be used by the utility, one for each of the four possible object types. If both input and output is performed for a given object type, the same PDS will be used for it. The PDSs are located by DDNAME. The values assumed for these names are taken from the TSO GDDM external defaults module, and you should modify the sample JCL for those defaults.

The utility reads a control file containing a list of operations required. For an import/export operation, it retrieves the object from the appropriate GDDM instance, and passes it to the other. For a delete operation, it issues an explicit delete object call to the IMS GDDM instance.

Each record is either:

- A comment, in which case it has an asterisk in column 1.
- Or takes the form:

  ```
  B  function B objectname B objecttype (B (comment))
  ```

  where:

`B`	represents one or more blanks
`function`	may be IMPORT, EXPORT, or DELETE.
`brackets ()`	are not written, but indicate optional data

`objectname` is the name of the object to be processed
`objecttype` is any one of the currently defined GDDM objects.

For example:

```
IMPORT ADMCOLSD ADMSYMBL
IMPORT ADMDHIIA ADMSYMBL
IMPORT ADMDHIIC ADMSYMBL
```

The object name may be specified in generic form using an asterisk (*). Thus, "ADMI*" specifies all those objects that have a name starting with ADMI. Similarly, "*ITAL*" specifies all those objects that have the character string ITAL in their name, such as ADMITALA.

Because there is a large number of GDDM required symbol sets, a lot of input statements for the initial symbol set load are required. With the generic form of specification, it is possible to enter:

```
IMPORT * ADMSYMBL
```

Also, for subsequent activity on the database, statements of the form

```
EXPORT ADMDH* ADMSYMBL
```

or

```
DELETE *MY* ADMSAVE
```

are allowed. In general, the `objectname` specification must be alphanumeric and not more than eight characters. It can start, or end, or both, with an asterisk, but cannot contain embedded asterisks. The `objecttype` must always be specified in full.

Notes:

1. Only columns 1 through 72 are inspected.

   The utility writes a SYSPRINT report, containing the results of processing each object. Sample JCL to run the utility is provided on the installation tape as job stream ADMUJI07. The input data stream corresponds to that required to install the main symbol sets distributed with GDDM.
2. Two PSBs are defined, ADMFOUL to be used when loading a database for the first time, and ADMFOU to be used when updating an existing database.
3. If the IMS subsystem has been defined to use different segment and key-field names for this database, this change should be reflected in the external defaults module.
4. The utility issues symbolic checkpoint calls when being used to update an existing database. This limits the impact on the system resources enqueue space and dynamic log space, and also reduces the possibility of enqueue lockout on the database.

Appendix G. Conversion table for GDDM and PostScript typefaces

The default table for conversions from IBM presentation-text fonts to PostScript fonts is shown in Table 43.

The font identifiers listed here all begin C0....., which denotes fonts for the IBM 382x family of printers. GDDM also converts the identifiers of some presentation-text fonts for 38pp output to PostScript typefaces, by converting the first two characters of their identifiers (C1,C2,C3,C4) to C0 and then using this table. To convert any other 38pp fonts or rotations, you must specify the mapping in the table.

Table 43 (Page 1 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0A055A0	Helvetica	10
C0A055B0	Helvetica	11
C0A055B1	Helvetica	48
C0A055D0	Helvetica	12
C0A055F0	Helvetica	14
C0A055H0	Helvetica	16
C0A055J0	Helvetica	18
C0A055N0	Helvetica	20
C0A055N1	Helvetica	60
C0A055T0	Helvetica	24
C0A055Z0	Helvetica	30
C0A055Z1	Helvetica	72
C0A05500	Helvetica	9
C0A05560	Helvetica	5
C0A05570	Helvetica	6
C0A05580	Helvetica	7
C0A05590	Helvetica	8
C0A075A0	Helvetica-Bold	10
C0A075B0	Helvetica-Bold	11
C0A075B1	Helvetica-Bold	48
C0A075D0	Helvetica-Bold	12
C0A075F0	Helvetica-Bold	14
C0A075H0	Helvetica-Bold	16
C0A075J0	Helvetica-Bold	18
C0A075N0	Helvetica-Bold	20
C0A075N1	Helvetica-Bold	60
C0A075T0	Helvetica-Bold	24
C0A075Z0	Helvetica-Bold	30
C0A075Z1	Helvetica-Bold	72

Table 43 (Page 2 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0A07500	Helvetica-Bold	9
C0A07560	Helvetica-Bold	5
C0A07570	Helvetica-Bold	6
C0A07580	Helvetica-Bold	7
C0A07590	Helvetica-Bold	8
C0A155A0	Helvetica-Oblique	10
C0A155B0	Helvetica-Oblique	11
C0A155B1	Helvetica-Oblique	48
C0A155D0	Helvetica-Oblique	12
C0A155F0	Helvetica-Oblique	14
C0A155H0	Helvetica-Oblique	16
C0A155J0	Helvetica-Oblique	18
C0A155N0	Helvetica-Oblique	20
C0A155N1	Helvetica-Oblique	60
C0A155T0	Helvetica-Oblique	24
C0A155Z0	Helvetica-Oblique	30
C0A155Z1	Helvetica-Oblique	72
C0A15500	Helvetica-Oblique	9
C0A15560	Helvetica-Oblique	5
C0A15570	Helvetica-Oblique	6
C0A15580	Helvetica-Oblique	7
C0A15590	Helvetica-Oblique	8
C0A175A0	Helvetica-BoldOblique	10
C0A175B0	Helvetica-BoldOblique	11
C0A175B1	Helvetica-BoldOblique	48
C0A175D0	Helvetica-BoldOblique	12
C0A175F0	Helvetica-BoldOblique	14
C0A175H0	Helvetica-BoldOblique	16
C0A175J0	Helvetica-BoldOblique	18
C0A175N0	Helvetica-BoldOblique	20
C0A175N1	Helvetica-BoldOblique	60
C0A175T0	Helvetica-BoldOblique	24
C0A175Z0	Helvetica-BoldOblique	30
C0A175Z1	Helvetica-BoldOblique	72
C0A17500	Helvetica-BoldOblique	9
C0A17560	Helvetica-BoldOblique	5
C0A17570	Helvetica-BoldOblique	6
C0A17580	Helvetica-BoldOblique	7
C0A17590	Helvetica-BoldOblique	8
C0C055A0	NewCenturySchlbk-Roman	11
C0C055B0	NewCenturySchlbk-Roman	12

Table 43 (Page 3 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0C055D0	NewCenturySchlbk-Roman	14
C0C055F0	NewCenturySchlbk-Roman	16
C0C055H0	NewCenturySchlbk-Roman	18
C0C055J0	NewCenturySchlbk-Roman	20
C0C055N0	NewCenturySchlbk-Roman	24
C0C055T0	NewCenturySchlbk-Roman	30
C0C055Z0	NewCenturySchlbk-Roman	36
C0C05500	NewCenturySchlbk-Roman	10
C0C05560	NewCenturySchlbk-Roman	06
C0C05570	NewCenturySchlbk-Roman	07
C0C05580	NewCenturySchlbk-Roman	08
C0C05590	NewCenturySchlbk-Roman	09
C0C075A0	NewCenturySchlbk-Bold	11
C0C075B0	NewCenturySchlbk-Bold	12
C0C075D0	NewCenturySchlbk-Bold	14
C0C075F0	NewCenturySchlbk-Bold	16
C0C075H0	NewCenturySchlbk-Bold	18
C0C075J0	NewCenturySchlbk-Bold	20
C0C075N0	NewCenturySchlbk-Bold	24
C0C075T0	NewCenturySchlbk-Bold	30
C0C075Z0	NewCenturySchlbk-Bold	36
C0C07500	NewCenturySchlbk-Bold	10
C0C07560	NewCenturySchlbk-Bold	06
C0C07570	NewCenturySchlbk-Bold	07
C0C07580	NewCenturySchlbk-Bold	08
C0C07590	NewCenturySchlbk-Bold	09
C0C155A0	NewCenturySchlbk-Italic	11
C0C155B0	NewCenturySchlbk-Italic	12
C0C155D0	NewCenturySchlbk-Italic	14
C0C155F0	NewCenturySchlbk-Italic	16
C0C155H0	NewCenturySchlbk-Italic	18
C0C155J0	NewCenturySchlbk-Italic	20
C0C155N0	NewCenturySchlbk-Italic	24
C0C155T0	NewCenturySchlbk-Italic	30
C0C155Z0	NewCenturySchlbk-Italic	36
C0C15500	NewCenturySchlbk-Italic	10
C0C15560	NewCenturySchlbk-Italic	06
C0C15570	NewCenturySchlbk-Italic	07
C0C15580	NewCenturySchlbk-Italic	08
C0C15590	NewCenturySchlbk-Italic	09
C0C175A0	NewCenturySchlbk-BoldItalic	11

Table 43 (Page 4 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0C175B0	NewCenturySchlbk-BoldItalic	12
C0C175D0	NewCenturySchlbk-BoldItalic	14
C0C175F0	NewCenturySchlbk-BoldItalic	16
C0C175H0	NewCenturySchlbk-BoldItalic	18
C0C175J0	NewCenturySchlbk-BoldItalic	20
C0C175N0	NewCenturySchlbk-BoldItalic	24
C0C175T0	NewCenturySchlbk-BoldItalic	30
C0C175Z0	NewCenturySchlbk-BoldItalic	36
C0C17500	NewCenturySchlbk-BoldItalic	10
C0C17560	NewCenturySchlbk-BoldItalic	06
C0C17570	NewCenturySchlbk-BoldItalic	07
C0C17580	NewCenturySchlbk-BoldItalic	08
C0C17590	NewCenturySchlbk-BoldItalic	09
C0D0GB10	Courier-Bold	10
C0D0GB12	Courier-Bold	9
C0D0GC15	Courier	7
C0D0GI12	Courier-Oblique	9
C0D0GP12	Courier	10
C0D0GR10	Courier	10
C0D0GT10	Courier	10
C0D0GT12	Courier	9
C0D0GT15	Courier	8
C0D0GT20	Courier	5
C0D0GT24	Courier	4
C0D0RT10	Courier	10
C0D0SB12	Courier-Bold	9
C0D0SI10	Courier-Oblique	9
C0D0SI12	Courier-Oblique	9
C0D0SO12	Courier	9
C0D0ST10	Courier	10
C0D0ST12	Courier	9
C0D0ST15	Courier	8
C0J055J0	Times-Roman	20
C0J055Z0	Times-Roman	36
C0LITTLE	Courier-Bold	7
C0L0DUMP	Courier	10
C0L0FM10	Courier	10
C0L0FM12	Courier	9
C0L0FM15	Courier	8
C0L0GU10	Courier	10
C0L0GU12	Courier	9

Table 43 (Page 5 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0L0GU15	Courier	8
C0L0KATA	Courier	10
C0L0TU10	Courier	10
C0L00AOA	Courier	10
C0L00AON	Courier	10
C0L00BOA	Courier	10
C0L00BON	Courier	10
C0L00GSC	Courier	10
C0L00GUC	Courier	10
C0L00OAB	Courier	10
C0L00T11	Courier	10
C0S0AE10	Courier	10
C0S0AE20	Courier	5
C0S0BITR	Courier	10
C0S0BRTR	Courier-Bold	10
C0S0CE10	Courier	08
C0S0CE12	Courier	9
C0S0CH10	Courier	10
C0S0CI10	Courier-Oblique	10
C0S0CO10	Courier	10
C0S0CR10	Courier	10
C0S0DOTR	Courier	10
C0S0EBTR	Courier-Bold	10
C0S0EITR	Courier-Oblique	10
C0S0ELTR	Courier	10
C0S0EOTR	Courier	10
C0S0ESTR	Courier	10
C0S0LB12	Courier-Bold	9
C0S0LR12	Courier-Bold	9
C0S0OB10	Courier-Bold	10
C0S0OR10	Courier	10
C0S0PB12	Courier-Bold	9
C0S0PI12	Courier-Oblique	9
C0S0PR10	Courier	10
C0S0PR12	Courier	9
C0S0SR12	Courier	9
C0S0S192	Courier	9
C0S0S193	Courier	9
C0S0S198	Courier	10
C0S055A0	Bookman-Light	11
C0S055B0	Bookman-Light	12

Table 43 (Page 6 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0S055D0	Bookman-Light	14
C0S055F0	Bookman-Light	16
C0S055H0	Bookman-Light	18
C0S055J0	Bookman-Light	20
C0S055N0	Bookman-Light	24
C0S055T0	Bookman-Light	30
C0S055Z0	Bookman-Light	36
C0S05500	Bookman-Light	10
C0S05560	Bookman-Light	06
C0S05570	Bookman-Light	07
C0S05580	Bookman-Light	08
C0S05590	Bookman-Light	09
C0S075A0	Bookman-Demi	11
C0S075B0	Bookman-Demi	12
C0S075D0	Bookman-Demi	14
C0S075F0	Bookman-Demi	16
C0S075H0	Bookman-Demi	18
C0S075J0	Bookman-Demi	20
C0S075N0	Bookman-Demi	24
C0S075T0	Bookman-Demi	30
C0S075Z0	Bookman-Demi	36
C0S07500	Bookman-Demi	10
C0S07560	Bookman-Demi	06
C0S07570	Bookman-Demi	07
C0S07580	Bookman-Demi	08
C0S07590	Bookman-Demi	09
C0S155A0	Bookman-LightItalic	11
C0S155B0	Bookman-LightItalic	12
C0S155D0	Bookman-LightItalic	14
C0S155F0	Bookman-LightItalic	16
C0S155H0	Bookman-LightItalic	18
C0S155J0	Bookman-LightItalic	20
C0S155N0	Bookman-LightItalic	24
C0S155T0	Bookman-LightItalic	30
C0S155Z0	Bookman-LightItalic	36
C0S15500	Bookman-LightItalic	10
C0S15560	Bookman-LightItalic	06
C0S15570	Bookman-LightItalic	07
C0S15580	Bookman-LightItalic	08
C0S15590	Bookman-LightItalic	09
C0S175A0	Bookman-DemiItalic	11

Table 43 (Page 7 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0S175B0	Bookman-DemiItalic	12
C0S175D0	Bookman-DemiItalic	14
C0S175F0	Bookman-DemiItalic	16
C0S175H0	Bookman-DemiItalic	18
C0S175J0	Bookman-DemiItalic	20
C0S175N0	Bookman-DemiItalic	24
C0S175T0	Bookman-DemiItalic	30
C0S175Z0	Bookman-DemiItalic	36
C0S17500	Bookman-DemiItalic	10
C0S17560	Bookman-DemiItalic	06
C0S17570	Bookman-DemiItalic	07
C0S17580	Bookman-DemiItalic	08
C0S17590	Bookman-DemiItalic	09
C0T055A0	Times-Roman	11
C0T055B0	Times-Roman	12
C0T055B1	Times	48
C0T055D0	Times-Roman	14
C0T055F0	Times-Roman	16
C0T055H0	Times-Roman	18
C0T055J0	Times-Roman	20
C0T055N0	Times-Roman	24
C0T055N1	Times	60
C0T055T0	Times-Roman	30
C0T055Z0	Times-Roman	36
C0T055Z1	Times	72
C0T05500	Times-Roman	10
C0T05560	Times-Roman	6
C0T05570	Times-Roman	7
C0T05580	Times-Roman	8
C0T05590	Times-Roman	9
C0T075A0	Times-Bold	11
C0T075B0	Times-Bold	12
C0T075B1	Times-Bold	48
C0T075D0	Times-Bold	14
C0T075F0	Times-Bold	16
C0T075H0	Times-Bold	18
C0T075J0	Times-Bold	20
C0T075N0	Times-Bold	24
C0T075N1	Times-Bold	60
C0T075T0	Times-Bold	30
C0T075Z0	Times-Bold	36

Table 43 (Page 8 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0T075Z1	Times-Bold	72
C0T07500	Times-Bold	10
C0T07560	Times-Bold	6
C0T07570	Times-Bold	7
C0T07580	Times-Bold	8
C0T07590	Times-Bold	9
C0T155A0	Times-Italic	11
C0T155B0	Times-Italic	12
C0T155B1	Times-Italic	48
C0T155D0	Times-Italic	14
C0T155F0	Times-Italic	16
C0T155H0	Times-Italic	18
C0T155J0	Times-Italic	20
C0T155N0	Times-Italic	24
C0T155N1	Times-Italic	60
C0T155T0	Times-Italic	30
C0T155Z0	Times-Italic	36
C0T155Z1	Times-Italic	72
C0T15500	Times-Italic	10
C0T15560	Times-Italic	6
C0T15570	Times-Italic	7
C0T15580	Times-Italic	8
C0T15590	Times-Italic	9
C0T175A0	Times-BoldItalic	11
C0T175B0	Times-BoldItalic	12
C0T175B1	Times-BoldItalic	48
C0T175D0	Times-BoldItalic	14
C0T175F0	Times-BoldItalic	16
C0T175H0	Times-BoldItalic	18
C0T175J0	Times-BoldItalic	20
C0T175N0	Times-BoldItalic	24
C0T175N1	Times-BoldItalic	60
C0T175T0	Times-BoldItalic	30
C0T175Z0	Times-BoldItalic	36
C0T175Z1	Times-BoldItalic	72
C0T17500	Times-BoldItalic	10
C0T17560	Times-BoldItalic	6
C0T17570	Times-BoldItalic	7
C0T17580	Times-BoldItalic	8
C0T17590	Times-BoldItalic	9
C0V055A0	AvantGarde-Book	11

Table 43 (Page 9 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0V055B0	AvantGarde-Book	12
C0V055D0	AvantGarde-Book	14
C0V055F0	AvantGarde-Book	16
C0V055H0	AvantGarde-Book	18
C0V055J0	AvantGarde-Book	20
C0V055N0	AvantGarde-Book	24
C0V055T0	AvantGarde-Book	30
C0V055Z0	AvantGarde-Book	36
C0V05500	AvantGarde-Book	10
C0V05560	AvantGarde-Book	06
C0V05570	AvantGarde-Book	07
C0V05580	AvantGarde-Book	08
C0V05590	AvantGarde-Book	09
C0V075A0	AvantGarde-Demi	11
C0V075B0	AvantGarde-Demi	12
C0V075D0	AvantGarde-Demi	14
C0V075F0	AvantGarde-Demi	16
C0V075H0	AvantGarde-Demi	18
C0V075J0	AvantGarde-Demi	20
C0V075N0	AvantGarde-Demi	24
C0V075T0	AvantGarde-Demi	30
C0V075Z0	AvantGarde-Demi	36
C0V07500	AvantGarde-Demi	10
C0V07560	AvantGarde-Demi	06
C0V07570	AvantGarde-Demi	07
C0V07580	AvantGarde-Demi	08
C0V07590	AvantGarde-Demi	09
C0V155A0	AvantGarde-BookOblique	11
C0V155B0	AvantGarde-BookOblique	12
C0V155D0	AvantGarde-BookOblique	14
C0V155F0	AvantGarde-BookOblique	16
C0V155H0	AvantGarde-BookOblique	18
C0V155J0	AvantGarde-BookOblique	20
C0V155N0	AvantGarde-BookOblique	24
C0V155T0	AvantGarde-BookOblique	30
C0V155Z0	AvantGarde-BookOblique	36
C0V15500	AvantGarde-BookOblique	10
C0V15560	AvantGarde-BookOblique	06
C0V15570	AvantGarde-BookOblique	07
C0V15580	AvantGarde-BookOblique	08
C0V15590	AvantGarde-BookOblique	09

Table 43 (Page 10 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C0V175A0	AvantGarde-DemiOblique	11
C0V175B0	AvantGarde-DemiOblique	12
C0V175D0	AvantGarde-DemiOblique	14
C0V175F0	AvantGarde-DemiOblique	16
C0V175H0	AvantGarde-DemiOblique	18
C0V175J0	AvantGarde-DemiOblique	20
C0V175N0	AvantGarde-DemiOblique	24
C0V175T0	AvantGarde-DemiOblique	30
C0V175Z0	AvantGarde-DemiOblique	36
C0V17500	AvantGarde-DemiOblique	10
C0V17560	AvantGarde-DemiOblique	06
C0V17570	AvantGarde-DemiOblique	07
C0V17580	AvantGarde-DemiOblique	08
C0V17590	AvantGarde-DemiOblique	09
C0Z05640	Helvetica	4
C00005XA	ISILGothic	05
C00005ZA	ISILGothic	05
C00007XA	ISILGothic	07
C00007XB	ISILGothic-Bold	07
C00007XC	ISILGothic-Oblique	07
C00007XD	ISILGothic-BoldOblique	07
C00007XE	ISILGothic	07
C00007ZA	ISILGothic	07
C00007ZB	ISILGothic-Bold	07
C00007ZC	ISILGothic-Oblique	07
C00007ZD	ISILGothic-BoldOblique	07
C00007ZE	ISILGothic	07
C00008XA	ISILGothic	08
C00008XB	ISILGothic-Bold	08
C00008XC	ISILGothic-Oblique	08
C00008XD	ISILGothic-BoldOblique	08
C00008XE	ISILGothic	08
C00008ZA	ISILGothic	08
C00008ZB	ISILGothic-Bold	08
C00008ZC	ISILGothic-Oblique	08
C00008ZD	ISILGothic-BoldOblique	08
C00008ZE	ISILGothic	08
C00009XA	ISILGothic	09
C00009XB	ISILGothic-Bold	09
C00009XC	ISILGothic-Oblique	09
C00009XD	ISILGothic-BoldOblique	09

Table 43 (Page 11 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C00009XE	ISILGothic	09
C00009ZA	ISILGothic	09
C00009ZB	ISILGothic-Bold	09
C00009ZC	ISILGothic-Oblique	09
C00009ZD	ISILGothic-BoldOblique	09
C00009ZE	ISILGothic	09
C00010XA	ISILGothic	10
C00010XB	ISILGothic-Bold	10
C00010XC	ISILGothic-Oblique	10
C00010XD	ISILGothic-BoldOblique	10
C00010XE	ISILGothic	10
C00010ZA	ISILGothic	10
C00010ZB	ISILGothic-Bold	10
C00010ZC	ISILGothic-Oblique	10
C00010ZD	ISILGothic-BoldOblique	10
C00010ZE	ISILGothic	10
C00011XA	ISILGothic	11
C00011XB	ISILGothic-Bold	11
C00011XC	ISILGothic-Oblique	11
C00011XD	ISILGothic-BoldOblique	11
C00011XE	ISILGothic	11
C00011ZA	ISILGothic	11
C00011ZB	ISILGothic-Bold	11
C00011ZC	ISILGothic-Oblique	11
C00011ZD	ISILGothic-BoldOblique	11
C00011ZE	ISILGothic	11
C00012XA	ISILGothic	12
C00012XB	ISILGothic-Bold	12
C00012XC	ISILGothic-Oblique	12
C00012XD	ISILGothic-BoldOblique	12
C00012XE	ISILGothic	12
C00012ZA	ISILGothic	12
C00012ZB	ISILGothic-Bold	12
C00012ZC	ISILGothic-Oblique	12
C00012ZD	ISILGothic-BoldOblique	12
C00012ZE	ISILGothic	12
C00014XA	ISILGothic	14
C00014XB	ISILGothic-Bold	14
C00014XC	ISILGothic-Oblique	14
C00014XD	ISILGothic-BoldOblique	14
C00014XE	ISILGothic	14

Table 43 (Page 12 of 12). The PostScript typefaces that most closely match IBM presentation-text fonts

IBM presentation-text font	Closest matching PostScript typeface	Size in points
C00014ZA	ISILGothic	14
C00014ZB	ISILGothic-Bold	14
C00014ZC	ISILGothic-Oblique	14
C00014ZD	ISILGothic-BoldOblique	14
C00014ZE	ISILGothic	14

Appendix H. Macros that are general-use programming interfaces

The macros identified in this appendix are provided as programming interfaces for customers by GDDM.

Warning: Do not use as programming interfaces any GDDM macros other than those identified in this appendix.

Macros:

ADMMAFP	Device tokens for AFPDS printers
ADMMCFNT	Font identifiers and PSF member names
ADMMCOLT	Color-separation masters
ADMMDFT	External-default specifications
ADMMDFTX	VSE batch printing
ADMMEXIT	User exits
ADMMFONT	Font table definition
ADMMIMAG	Device tokens for high-resolution image printers
ADMMKFNT	AFPDS font table definition
ADMMNICK	Nickname specifications
ADMMSYSP	Device tokens for system printers
ADMMTRAN	Translation of user-defined shading patterns
ADMM3270	Device tokens for 3270-family devices and queued printers

Mapping constants declarations:

ADMUAIMC	Assembler
ADMUBIMC	C/370
ADMUCIMC	COBOL
ADMUPIMC	PL/I

Request control parameter tables:

ADMURCPB	GDDM Base
ADMURCPO	GDDM-PGF

There are no macros intended for use as part of a general-use programming interface in GDDM-GKS, GDDM-IVU, or GDDM-IMD.

Glossary

This glossary defines technical terms used in GDDM documentation. If you do not find the term you are looking for, refer to the index of the appropriate GDDM manual or view the *IBM Dictionary of Computing*, located on the Internet at:

`http:\\www.networking.ibm.com/nsg/nsgmain.htm`

A

AAB. Application anchor block.

ACB. Application control block.

active operator window. In GDDM, the operator window with the highest priority in the viewing order.

active partition. The partition containing the cursor. Contrast with *current partition*.

advanced function printing. The ability of licensed programs to use the all-points-addressable concept to print text and illustrations.

adjunct. In mapped alphanumerics, one of a set of optional subfields in an application data structure that specifies some attribute of a data field; for example, that it is highlighted. An adjunct enables the attribute to be varied at run time.

ADMGDF. See *graphics data format (GDF)*.

ADS. Application data structure.

AFPDS. Advanced-function presentation data stream.

AIC. Application interface component.

alphanumeric character attributes. In GDDM, the highlighting, color, and symbol set to be used for individual characters.

alphanumeric cursor. A physical indicator on a display. It can be moved from one hardware cell to another.

alphanumeric field. A field (area of a screen or printer page) that can contain alphabetic, numeric, or special characters. In GDDM, contrast with *graphics field*.

alphanumeric field attributes. In GDDM, the intensity, highlighting, color, and symbol set to be used for field type, field end, output conversion, input conversion, translate table assignment, transparency, field outlining, and mixed-string fields.

alphanumerics. Pertaining to alphanumeric fields. In GDDM there are three types of alphanumerics:

- Procedural alphanumerics
- Mapped alphanumerics
- High performance alphanumerics (HPA)

alternate device. In GDDM, a device to which copies of the primary device's output are sent. Usually the alternate device is a printer or plotter. See also *primary device*.

annotation. An added descriptive comment or explanatory note.

APA. All points addressable.

aperture. See *pick aperture*.

API. Application programming interface.

APL. One of the programming languages supported by GDDM.

application data structure (ADS). A structure created by GDDM-IMD that contains an entry for each variable field within a *map*. The data to be displayed in a mapped field is placed into the application data structure by the user's program.

application image. In GDDM, an image contained in GDDM main storage, and independent of any device or GDDM page. Contrast with *device image*.

application programming interface (API). The formally defined interface used by an application programmer to pass commands to, and get responses from, an IBM system control program or licensed program.

area. In GDDM, 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.

attention identifier. A number indicating which button the operator pressed to satisfy a read operation. For example, 0 (returned from GDDM to the application program) means that the operator pressed the Enter key.

attribute byte. The screen position that precedes an alphanumeric field on a 3270-family device and holds the attribute information. See also *trailing attribute byte*.

attributes. Characteristics or properties that can be controlled, usually to obtain a required appearance; for example, the color of a line. See also *alphanumeric character attributes*, *alphanumeric field attributes*, and *graphics attributes*.

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*.

bar code. A code representing characters by sets of vertical parallel bars of varying thickness and separation that are read optically by transverse scanning.

BASIC. One of the programming languages supported by GDDM.

BDAM. Basic Direct Access Method.

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*.

BMS. Basic Mapping Support (CICS).

BPAM. Basic Partitioned Access Method.

business graphics. The methods and techniques for presenting commercial and administrative information in chart form; for example, the creation and display of a sales bar chart. Contrast with *general graphics*.

C

CALS. Continuous Acquisition and Life-Cycle Support.

CDPDS. Composite Document Presentation Data Stream.

CDPF. Composed Document Print Facility.

CDPU. Composite Document Print Utility.

CECP. Country-extended code page.

cell. See *character cell*.

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.

channel-attached. Pertaining to devices that are attached directly to a computer by means of data (I/O) channels. Synonymous with *local*. Contrast with *link-attached*.

character. A letter, digit, or other symbol.

character attributes. See *alphanumeric character attributes*. See also *graphics text attributes*.

character box. In GDDM, the rectangle or (for sheared characters) the parallelogram boundaries that govern the size, orientation, spacing, and italicizing of individual symbols or characters to be shown on a display screen or printer page.

The box width, height, and, if required, shear are specified in world coordinates and can be program-controlled. See also *character mode*. Contrast with *character cell*.

character cell. The physical, rectangular space in which any single character or symbol is displayed on a screen or printer device. The size and position of a character cell are fixed. Size is usually specified in pixels on a given device; for example, 9 by 12 on an IBM 3279 Model 3 display. Position is addressed by row and column coordinates. Synonymous with *hardware cell* and *symbol cell*. Contrast with *character box*.

character code. The means of addressing a symbol in a symbol set, sometimes called *code point*.

The particular form and range of codes depends on the GDDM context. For example:

- For the Image Symbol Editor, a hexadecimal constant in the range X'41' 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 field*. 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 mode-2 characters and the size of mode-3 characters. It also governs the size of the chart margins and thus the plotting area.

character matrix. Synonym for *dot matrix*.

character mode. In GDDM, the type of characters to be used. There are three modes:

- Mode-1 characters are loadable into PS and are of device-dependent fixed size, spacing, and orientation, as are hardware characters.
- Mode-2 characters are image (ISS) characters. Size and orientation are fixed. Spacing is variable by program.
- Mode-3 characters are vector (VSS) characters. Box size, position, spacing, orientation, and shear of individual characters are variable by program.

chart. In GDDM, usually means business chart; for example, a *bar chart*.

choice device. A logical input device that enables the application program to identify keys pressed by the terminal operator.

CICS. Customer Information Control System. A subsystem of MVS or VSE under which GDDM can be used.

clipping. In computer graphics, removing parts of a display image that lie outside a viewport. Synonymous with *scissoring*.

CMS. Conversational Monitor System. A time-sharing subsystem that runs under VM/SP.

COBOL. One of the programming languages supported by GDDM.

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 IBM 4250 page printer.

code point. Synonym for *character code*.

Composite Document Presentation Data Stream (CDPDS). A data stream containing graphics, image, and text that is the input to the GDDM Composite Document Print Utility (CDPU).

Composed Document Print Facility (CDPF). An IBM licensed program for processing documents destined for the IBM 4250 page printer.

composed-page image file. An intermediate form, residing on disk, of a picture destined for a page printer.

composed-page printer. See *page printer*.

composed-page printer format. A general term describing the format of print data destined for output by using either *CDPF* or *PSF*.

composite document. A document that contains both formatted text, such as that produced by the DCF program, and graphic or image data, such as that produced by GDDM. It is a combination of text and pictures on a page or set of pages. The pictures can be computer graphics or images created by scanning paper originals.

Composite Document Print Utility (CDPU). A utility that can print or display composite documents

compressed data stream. A data stream that has been made more compact by use of a data-compression algorithm.

constant data. In GDDM, data that is defined in a map and need not be known to the application program.

correlation. The translation (by GDDM) of a screen position into a part of the user's picture. This follows a *pick* operation.

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.

CSD. (1) Under MVS or VSE, CICS system definition. (2) In personal computer systems, Corrective Service Diskette; the means by which service is applied to the personal computer system.

current partition. The partition selected for processing by the application program. Contrast with *active partition*.

current position. In GDDM, the end of the previously drawn primitive. Unless a "move" is performed, this position is also the start of the next primitive.

cursor. A physical indicator that can be moved around a display screen. See *alphanumeric cursor* and *graphics cursor*.

CUT. Control unit terminal.

D

DASD. Direct access storage device.

data stream compatibility (DSC). In IBM 8100 systems, the facility that provides access to System/370 applications that communicate with IBM 3270 Information Display System terminals.

data stream compression. The shortening of an I/O data stream for the purpose of more efficient transmission between link-attached units.

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.

DCF. Document Composition Facility.

DCSS. Discontiguous saved segment (VM/SP).

DCT. Destination control table (CICS).

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. See also *drawing default* and *standard default*.

denibblized data. The decoded data stream used between the GDDM DOS Support feature in the host and GDDM-PCLK on the workstation.

designator character. The first byte of a light-pen-detectable field that indicates whether or not the field has been selected.

device echo. A visual identification of the position of the graphics cursor. The form of the device echo is defined by the application program.

device family. In GDDM, a device classification that governs the general way in which I/O will be processed. See also *processing option*. For example:

- Family 1: 3270 display or printer
- Family 2: queued printer
- Family 3: system printer (alphanumerics only)
- Family 4: page printer

device image. In GDDM, an image contained in a device or GDDM page. Contrast with *application image*.

device suffix. In GDDM-IMD, a suffix to a mapgroup name that indicates the device class.

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).

DIF. In GDDM terms, data interchange format.

digital image. A two-dimensional array of picture elements (pixels) representing a picture. A digital image can be stored and processed by a computer, using bits to represent pixels. In GDDM, pixels have the value black or white. Often called simply *image*.

direct transmission. In GDDM image processing, the transfer of image data direct from a source outside GDDM to an image device, including manipulation by a projection in the device, and without GDDM maintaining a copy or buffer of the data.

display device. Any output unit that gives a visual representation of data; for example, a screen or printer. More commonly, the term is used to mean a screen and not a printer.

display point. Synonym for *pixel*.

display-point matrix. Synonym for *dot matrix*.

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, an IBM 3179 display.

DL/1. Data language 1. A language for database processing operations.

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. See *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*. Contrast with *single-byte character set*.

DPCX. Distributed Processing Control Executive. An IBM 8100 system control program.

DPPX. Distributed Processing Programming Executive. An IBM 8100 system control program.

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.

DSC. Data stream compatibility.

dual characters. See *double-byte characters*.

dummy device. An output destination for which GDDM does all the normal processing but for which no actual output is generated. Used, for example, to test programming for an unavailable output device.

E

EBCDIC. Extended binary coded decimal interchange code. A coded character set consisting of 8-bit coded characters.

echo. In interactive graphics, the visible form of the locator or other logical input device.

ECSA. Extended character set adapter.

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.

extended data stream. For IBM 3179, 3192, 3278, 3279, and 3287 devices, input/output data formatted and encoded in support of color, programmed symbols, and extended highlighting. These features extend the IBM 3270 data stream architecture.

extended highlighting. The emphasizing of a displayed character's appearance by blinking, underscore, or reverse video.

external defaults. GDDM-supplied values that users can change to suit their own needs.

extracted image. In GDDM, an image on which transform element calls operate. It may imply the whole source image or just a part of it, depending on whether a define sub-image transform element has been applied in its derivation.

F

FCT. File control table (CICS).

field. An area on the screen or the printed or plotted page. See *alphanumeric field*, *graphics field*, and *mapped field*.

field attributes. See *alphanumeric field attributes*.

field list. The high performance alphanumerics data structure used to define alphanumeric fields.

fillet. A curve that is tangential to the end points of two adjoining lines.

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.

floating area. The part of a page reserved for *floating maps*.

floating map. A map whose absolute position on the GDDM page is not fixed. During execution, a floating map takes the next available space that satisfies its specification.

floating-point feature. A processing unit feature that provides four 64-bit floating-point registers to perform floating-point arithmetic calculations.

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.

FORTRAN. One of the programming languages supported by GDDM.

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*.

frame. In GDDM-IMD, a synonym for *panel*.

full-screen alphanumeric operation. Full-screen processing operations on alphanumeric fields.

full-screen mode. A form of screen presentation in which the contents of an entire terminal screen can be displayed at once. Full-screen mode is often used for fill-the-blanks prompting, and is an alternative to line-by-line I/O.

full-screen processor. A host software component that, together with display terminal functions, supports display terminal input/output in full-screen mode.

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-GKS. GDDM Graphical Kernel System. A member of the GDDM family that runs under TSO and CMS and provides an alternative graphics programming interface to that of the GDDM base product. It is an implementation of the Graphical Kernel Standard, ISO 7942, of the International Organization for Standardization.

GDDM/graPHIGS. A member of the GDDM family used for creating hierarchical three-dimensional structures on the IBM 5080 Graphics System. It is based on the proposed ANSI standard for the Programmer's Hierarchical Interactive Graphics System (PHIGS).

GDDM Interactive Map Definition. GDDM-IMD. A member of the GDDM family of licensed programs. It enables users to create alphanumeric layouts at the terminal. The user defines the position of each field within the layout and may assign attributes, default data, and associated variable names to each field. The resultant map can be tested from within the utility.

GDDM-IVU. GDDM Interactive View Utility. A member of the GDDM family of licensed programs. It enables users to view, create, modify, store, and print images.

GDDM-OS/2. A licensed program that enables IBM PS/2 and other personal-computer systems with OS/2 installed to run GDDM application programs in the host computer.

GDDM-PCLK. A licensed program that enables IBM PS/2 and other personal computers with graphics-display adapters, and IBM 3270 terminal emulators 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.

GDDM storage. The portion of host computer main storage used by GDDM.

GDF. Graphics data format.

general graphics. The methods and techniques for converting data to or from graphics display in mathematical, scientific, or engineering applications; that is, in any application other than business graphics. See also *business graphics*.

generated mapgroup. The output produced when a source GDDM-IMD mapgroup is generated. It contains the information needed by GDDM at execution to position the mapped fields on the GDDM page.

GIF. Graphics Interchange Format.

GKS. Graphical Kernel System. See *GDDM-GKS*.

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. In GDDM, color selection, color mix, line type, line width, graphics text attributes, marker symbol, and shading pattern definition.

graphics cursor. A physical indicator that can be moved (often with a joystick, mouse, or stylus) to any position on the screen.

graphics data format (GDF). A picture definition in an encoded order format used internally by GDDM and, optionally, providing the user with a lower-level programming interface than the GDDM API.

graphics data stream. The data stream that produces graphics on the screen, printer, or plotter.

graphics field. A rectangular area of a screen or printer page, used for graphics. Contrast with *alphanumeric field*.

graphics input queue. A queue associated with the graphics field onto which elements arrive from logical input devices. The program can remove elements from the queue by issuing a graphics read.

graphics primitive. A single item of drawn graphics, such as a line, arc, or graphics text string. See also *graphics segment*.

graphics read. A form of read that solicits graphics input or removes existing elements from the graphics input queue.

graphics segment. A group of graphics primitives (lines, arcs, and text) that have a common window and a common viewport and associated attributes. Graphics

segments allow a group of primitives to be subject to various operations. See also *graphics primitive*.

graphics text attributes. In GDDM, the symbol (character) set to be used, character box size, character angle, character mode, character shear angle, and character direction.

graPHIGS. See *GDDM/graPHIGS*.

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*.

green lightning. The name given to the flashing streaks on an IBM 3270 screen while a programmable symbol set is being loaded.

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 cell. Synonym for *character cell*.

hardware characters. Synonym for *hardware symbols*.

hardware symbols. The characters that are supplied with the device. The term is loosely used also for GDDM mode-1 symbols that are loaded into a PS store for subsequent display. Synonymous with *hardware characters*.

hexadecimal. Pertaining to a numbering system with base sixteen.

host. See *host computer*.

high performance alphanumerics. The creation of alphanumeric displays using field list data structures. Contrast with *procedural* and *mapped* alphanumerics.

host computer. The primary or controlling computer in a multiple-computer installation.

I

ICU. Interactive Chart Utility.

identity projection. In GDDM image processing, a projection that is transferred from source image to target image without any processing being performed on it.

image. Synonym for *digital image*.

image data stream. The internal form of the GDDM data in an image environment.

image field. A rectangular area of a screen or printer page, used for image. Contrast with *alphanumeric field* and *graphics field*.

Image Object Content Architecture (IOCA). An architected collection of constructs used to interchange and present images.

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)*.

IMD. See *GDDM Interactive Map definition*.

IMS/VS. Information Management System/Virtual Storage. A subsystem of MVS under which GDDM can be used.

include member. A collection of source statements stored as a library member for later inclusion in a compilation.

input queue. See *graphics input queue*.

integer. A whole number (for example, –2, 3, 457).

Intelligent Printer Data Stream (IPDS). A structured-field data stream for managing and controlling printer processes, allowing both data and controls to be sent to the printer. GDDM uses IPDS to communicate with the IBM 4224 printer.

Interactive Chart Utility (ICU). A GDDM-PGF menu-driven program that allows business charts to be created interactively by nonprogrammers.

interactive graphics. In GDDM, those graphics that can be moved or manipulated by a user at a terminal.

Interactive Map definition. A member of the GDDM family of licensed programs. It enables users to create alphanumeric layouts at the terminal. The operator defines the position of each field within the layout and may assign attributes, default data, and associated variable names to each field. The resultant map can be tested from within the utility.

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.

interactive subsystem. (1) One or more terminals, printers, and any associated local controllers capable of operation in interactive mode. (2) One or more system programs or program products that enable user applications to operate in interactive mode; for example, CICS.

intercept. In a chart, a method of describing the position of one axis relative to another. For example, the x axis can be specified so that it intercepts (crosses) the y axis at the bottom, middle, or top of the plotting area of a chart.

inter-device copy. The ability to copy a page or the graphics field from the current primary device to another device. The target device is known as the alternate device.

IOCA. See *Image Object content Architecture*.

IPDS. See *Intelligent Printer Data Stream*.

ISE. Image Symbol Editor.

ISO. International Organization for Standardization.

ISPF. Interactive System Productivity Facility.

ISS. Image symbol set.

IVU. Image View Utility. See *GDDM-IVU*.

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

Latin. Of or pertaining to the Western alphabet. In GDDM, a synonym for *single-byte character set*.

legend. A set of symbolic keys used to identify the data groups in a business chart.

line attributes. In GDDM, color, line type, and line width.

link pack area. An MVS term that describes an area of shared storage.

link-attached. Pertaining to devices that are connected to a controlling unit by a data link. Synonymous with *remote*. Contrast with *channel-attached*.

local. Synonym for *channel-attached*.

local character set identifier. A hexadecimal value stored with a GDDM symbol set, which can be used by symbol-set-loading means other than GDDM in the context of local copy on a printer.

locator. A logical input device used to indicate a position on the screen. Its physical form may be the alphanumeric cursor or a graphics cursor moved by a joystick.

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.

LPA. Link pack area.

LTERM. In IMS/VS, logical terminal.

M

map. A predefined format of alphanumeric fields on a screen. Usually constructed outside of the application program.

map specification library (MSL). The data set in which maps are held in their source form.

mapgroup. A data item that contains a number of maps and information about the device on which those maps are to be used. All maps on a GDDM page must come from the same mapgroup.

mapped alphanumerics. The creation of alphanumeric displays using predefined maps. Contrast with *procedural alphanumerics* and *high performance alphanumerics*.

mapped field. An area of a page whose layout is defined by a map.

mapped graphics. Graphics placed in a graphics area within a mapped field.

mapped page. A GDDM page whose content is defined by maps in a mapgroup.

mapping. The use of a map to produce a panel from an output record, or an input record from a panel.

marker. In GDDM, a symbol centered on a point. Line graphs and polar charts can use markers to indicate the plotted points.

MDT. Modified data tag.

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.

MFS. Message format service.

MICR. Magnetic ink character recognition.

mixed character string. A string containing a mixture of *Latin* (one-byte) and *Kanji* or *Hangeul* (two-byte) characters.

Mixed Object Document Content Architecture (MO:DCA). An architected, device-independent data stream for interchanging documents.

mode-1/-2/-3 characters. See *character mode*.

mountain shading. A method of shading surface charts where each component is shaded separately from the base line, instead of being shaded from the data line of the previous component.

mouse. A device that a user moves on a flat surface to position a pointer on a screen.

MSHP. Maintain System History Program. A software process for installing licensed programs on VSE systems.

MSL. Map specification library.

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

name-list. A means of identifying which physical device is to be opened by a GDDM program. It can be used as a parameter of the DSOPEN call, or in a *nickname*.

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 US English.

negate. In bi-level image data, setting zero bits to one and one bits to zero.

neutral color. White on a display, black on a printer. Contrast with *background color*.

nibblized data. The encoded data stream used between the GDDM DOS Support feature in the host and GDDM-PCLK on the workstation.

nickname. In GDDM, a means of referring to a device, the characteristics and identity of which have been already defined.

NLS. National Language Support.

nonqueriable printer. A printer about which GDDM cannot obtain any information.

NSS. Named saved system (VM/XA and VM/ESA).

null character. An empty character represented by X'00' in the EBCDIC code. Such a character does not occupy a screen position.

O

operator reply mode. In GDDM, the mode of interaction available to the operator (display terminal user) with respect to the modification (or not) of alphanumeric character attributes for an input field.

operator window. Part of the display screen's surface on which the GDDM output of an application program can be shown. An operator window is controlled by the end user; contrast with *partition*. A *task manager* may create a window for each application program it is running.

outbound structured field. An element in IBM 3270 data streams from host to terminal with formatting that allows variable-length and multiple-field data to be sequentially translated by the receiver into its component fields without the receiver having to examine every byte.

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 IBM 3820 or IBM 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. A picture file in a form that can be printed. It can only be printed if it is embedded in a primary document. Also known as a *PSEG*o file.

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*.

partition. Part of the display screen's surface on which a page, or part of a page, of GDDM output can be shown. Two or more partitions can be created, each displaying a page, or part of a page, of output. A partition is controlled by the GDDM application; contrast with *operator window*.

partition set. A grouping of partitions that are intended for simultaneous display on a screen.

partitioned data set (PDS). A data set in direct access storage that is divided into partitions, called members, each of which can contain a program, part of a program, or data.

PCB. In GDDM, program communication block (IMS/VS).

PCLK. See *GDDM-PCLK*.

PDS. Partitioned data set (MVS).

pel. Picture element. See *pixel*.

PGF. Presentation Graphics Facility. A member of the GDDM family of licensed programs. It is concerned with business graphics, rather than general graphics.

PHIGS. Programmer's Hierarchical Interactive Graphics System.

pick. The action of the operator in selecting part of a graphics display by placing the graphics cursor over it.

pick aperture. A rectangular or square box that is moved across the screen by the graphics cursor. An item must lie at least partially within the pick aperture before it can be picked.

pick device. A logical input device that allows the application to determine which part of the picture was selected (or picked) by the operator.

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.

picture space. In GDDM, an area of specified aspect ratio that lies within the graphics field. It is centered on the graphics field and defines the part of the graphics field in which graphics will be drawn.

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*.

polar chart. A form of business chart where the x axis is circular and the y axis is radial.

polyfillet. In GDDM, a curve based on a sequence of lines. It is tangential to the end points of the first and last lines, and tangential also to the midpoints of all other lines.

polyline. A sequence of adjoining lines.

popping. A method of ordering data whereby each item in a list or sequence takes the value of the previous item in the list or sequence, and is then removed from the list; when this happens, the list or sequence of data is said to be "popped."

ppi. Pixels per inch.

PQE. Printer queue element.

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. See also *alternate device*.

primitive. See *graphics primitive*.

primitive attribute. A specifiable characteristic of a graphics primitive. See *graphics attributes* and *graphics text attributes*.

Print Services Facility (PSF). An IBM licensed program for processing documents destined for the IBM 3800 Model 3 page printer.

print utility. A subsystem-dependent utility that sends print files from various origins to a queued printer.

procedural alphanumerics. The creation of alphanumeric displays using the GDDM alphanumeric API. Contrast with *mapped alphanumerics* and *high performance alphanumerics*.

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.

procopt. Processing option.

profile. In GDDM, 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.

programmed symbols (PS). Dot patterns loaded by GDDM into the PS stores of an output device.

projection. In GDDM image processing, an application-defined function that specifies operations to be performed on data extracted from a source image. Consists of one or more *transforms*. See also *transform element*.

PS. Programmed symbols.

PS overflow. A condition where the graphics cannot be displayed in its entirety because the picture is too complex to be contained in the device's PS stores.

PSB. Program specification block (IMS).

PSEG. See *page segment*.

PSF. Print Services Facility.

PSP bucket. A database containing descriptions of faults found in programs. Used by Service personnel.

PS/2. Personal System/2.

puck. Synonym for *four-button cursor*.

PUT. Program update tape.

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*.

queued printer. A printer belonging to the subsystem under which GDDM runs, to which output is sent indirectly by means of the GDDM Print Utility program. In some subsystems, this may allow the printer to be shared between multiple users. Contrast with *system printer*.

R

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).

real device. A GDDM device that is not being windowed by means of operator window functions. Contrast with *virtual device*.

reentrant. The attribute of a program or routine that allows the same copy of the program or routine to be used concurrently by two or more tasks.

remote. Synonym for *link-attached*.

reply mode. See *operator reply mode*.

resolution. In graphics and image processing, the number of pixels per unit of measure (inch or meter).

reverse clipping. Where one graphics primitive overlaps another, removing any parts of the underlying primitive that are overpainted by the overlying primitive.

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.

scissoring. Synonym for *clipping*.

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.

SCS. SNA character string.

segment. See *graphics segment*.

segment attributes. Attributes that apply to the segment as an entity, rather than to the individual primitives within the segment. For example, the visibility, transformability, or detectability of a segment.

segment library. The portion of auxiliary storage where segment definitions are held. These definitions are GDDM objects in graphics data format (GDF) and are managed by GDDM API calls. GDDM handles the file accesses to and from auxiliary storage.

segment priority. The order in which segments are drawn; also the order in which they are detected.

segment transform. The means to rotate, scale, and reposition segments without re-creating them.

selector adjunct. A subfield of an application data structure that qualifies a data field.

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.*

SMP/E. System Modification Program/Extended. A software process for installing licensed programs on MVS systems.

SNA. System Network Architecture.

source image. An image that is the data input to image processing or transfer.

spill file. A means of reducing storage requirements at the cost of processing time, when creating high-resolution output files for page printers, for example.

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 overridden by the user.

string device. A logical input device that enables an application program to process character data entered by the terminal operator.

stroke device. A logical input device that enables an application program to process a sequence of x,y coordinate data entered by the terminal operator.

stylus. A pen-like pointer used on the surface of a tablet to indicate position on a screen.

surface chart. A chart similar to a line graph, except that no markers appear and the areas between successive lines are shaded.

swathe. A horizontal slice of printer output, forming part of a complete picture. Page printer images are often constructed in swathes to reduce the amount of storage required.

symbol. Synonymous with *character*. For example, the following terms all have the same meaning: vector symbols, vector characters, vector text.

symbol cell. Synonym for *character cell*.

symbol matrix. Synonym for *dot matrix*.

symbol set. A collection of symbols, usually but not necessarily forming a font. GDDM applications may use the hardware device's own symbol set. Alternatively, they can use image or vector symbol sets 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. Contrast with *queued printer*.

T

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 image. An image that is the destination of processed or transferred data.

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.

task manager. A program that supervises the concurrent running of other programs.

temporary graphics. Graphics created outside a segment.

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*.

terminal emulator. A program that enables a device such as a personal computer system to enter and receive data from a host computer system as if it were a particular type of attached 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*.

tilted pie chart. A pie chart drawn in three dimensions, which has been tilted away from full face to reveal its three-dimensional properties.

trailing attribute byte. The screen position following an alphanumeric field. This attribute byte can specify, for example, that the cursor should auto-skip to the next field when the current field is filled.

transfer operation. In GDDM image processing, an operation in which a projection is applied to a source image, and the result placed in a target image. The source and target images can be device or application images in any combination, or one or other of them (but not both) can be image data within the application program.

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.

See also *projection* and *transform element*.

transform element. In GDDM image processing, a specific function in a transform, which can be one of the following: define sub-image, scale, orient, reflect, negate, define place in target image.

A given transform element can be used only once in a *transform*.

transformable. A segment must be defined as transformable if it will subsequently be moved, scaled, or rotated.

transparency. (1) A document on transparent material suitable for overhead projection. (2) An alphanumeric attribute that allows underlying graphics or image to show.

TSO. Time Sharing Option. A subsystem of MVS under which GDDM can be used.

TWA. Transaction work area.

U

UDS. User default specification.

UDSL. A list of user default specifications (UDSs).

unformatted data. In GDDM image processing, compressed or uncompressed binary image data that has no headers, trailers, or embedded control fields other than any defined by the compression algorithm, if applicable. The data is in row major order, beginning with the top left of the picture.

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.

user default specification (UDS). The means of changing a GDDM external default value. The external default values that a UDS can change are those of the GDDM or subsystem environment, GDDM user exits, and device definitions.

user exit. A point in GDDM execution where a user routine will gain control if such has been requested.

V

variable cell size. In most devices, the hardware cell size is fixed, but the IBM 3290 Information Panel has a cell size that can be varied. This, in turn, causes the number of rows or columns on the device to alter.

vector. (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.

Venetian blind effect. The name given to the appearance of bars across shaded patterns on an IBM 3270-PC when GDDM tries to match the image symbol sets.

Venn diagram. A form of business chart in which, in GDDM, two or more populations and their intersection are represented by overlapping circles.

viewport. A subdivision of the picture space, most often used when two separate pictures are to be displayed together.

virtual device. A GDDM device that is being windowed by use of operator window functions. Contrast with *real device*.

virtual screen. The presentation space viewed through an *operator window*.

VM/ESA. IBM Virtual Machine Enterprise Systems Architecture.

VM/SP CMS. IBM Virtual Machine/System Product Conversational Monitor System; a system under which GDDM can be used.

VMXA. IBM Virtual Machine Extended Architecture; a system under which GDDM can be used.

VSE. Virtual storage extended; an operating system consisting of VSE/Advanced Functions and other IBM programs.

Note: In GDDM, the abbreviation VSE has sometimes been used to refer to the Vector Symbol Editor, but to avoid confusion, this usage is deprecated.

VSS. Vector symbol set.

W

Ward. One of the 190 matrices used to contain the symbols of a double-byte character set. The value in the first byte of each double-byte character code refers to the ward in which the character is contained. The value in the second byte denotes the character's position in the matrix.

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 should be displayed of a page that is deeper or wider than its partition.

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*.

wrap-around field. An alphanumeric field that extends to the right-hand edge of the page and continues at the start of the next row.

WTP. Write-to-programmer.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

Z

Sending your comments to IBM

GDDM

System Customization and Administration

SC33-0871-02

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 IBMMAIL
  - IBMLink: WINVMD(IDRCF)
  - Internet: idrcf@winvmd.vnet.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

System Customization and Administration

SC33-0871-02

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

GDDM System Customization and Administration SC33-0871-02

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/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

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.

SC33-0871-02