GDDM

Base Application Programming Reference

Version 3 Release 2

SC33-0868-02

GDDM

Base Application Programming Reference

Version 3 Release 2

SC33-0868-02

Note!

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

| **Third Edition (December 2001)**

This edition applies to the following IBM GDDM series of 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

| and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. It also applies to GDDM/MVS as an | element of OS/390 (program number 5645-001). 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.

Table of contents

Notices . . . xiii
Programming interface information . . . xiii
Trademarks and service marks . . . xiii

Preface . . . xv
What this book is about . . . xv
Who this book is for . . . xv
What you need to know . . . xv
How to use this book . . . xv
Terminology used . . . xv

Summary of changes . . . xvii
Summary of changes for GDDM 3.1 . . . xvii

More GDDM information . . . xix
GDDM publications . . . xix
Books from related libraries . . . xx

Chapter 1. GDDM programming interface . . . 1
The nonreentrant interface . . . 1
The reentrant interface . . . 1
The system programmer interface . . . 2
Application programming language considerations . . . 2
APL . . . 2
Assembler language . . . 3
BASIC (IBM) . . . 3
C . . . 3
COBOL . . . 5
FORTRAN . . . 5
PL/I . . . 6
REXX . . . 6

Chapter 2. A summary of the calls by function . . . 13
Types of functions . . . 13
Control functions . . . 14
Copy functions . . . 15
Device functions . . . 15
Graphics functions . . . 15
High-performance alphanumeric functions . . . 17
Image functions . . . 17
Mapped alphanumeric functions . . . 18
Operator window functions . . . 18
Page functions . . . 19
Partition functions . . . 19
Procedural alphanumeric functions . . . 19
Symbol set functions . . . 20
Utility call functions . . . 20

Chapter 3. The GDDM calls . . . 21
Format of the GDDM call descriptions . . . 21
Syntax of GDDM calls . . . 21
Error messages in GDDM calls . . . 21
Alphabetic list of GDDM calls . . . 21
APDEF . . . 21
APDEL . . . 22
APMOD . . . 22
APQIDS . . . 23
APQNUM . . . 24
APQRY . . . 24
APQSIZ . . . 25
APQUID . . . 25
ASCCOL . . . 25
ASCGET . . . 26
ASCHLT . . . 26
ASCPUT . . . 27
ASCSS . . . 27
ASDFLD . . . 28
ASDFLT . . . 29
ASDFMT . . . 29
ASDTRN . . . 30
ASFBDY . . . 31
ASFCLR . . . 31
ASFCOL . . . 32
ASFCUR . . . 32
ASFEND . . . 33
ASFHLT . . . 33
ASFIN . . . 33
ASFINT . . . 34
ASFMOD . . . 34
ASFOUT . . . 35
ASFPSS . . . 35
ASFSEN . . . 36
ASFTRA . . . 37
ASFTRN . . . 37
ASFTYP . . . 38
ASGGET . . . 38
ASGPUT . . . 39
ASMODE . . . 39
ASQCOL . . . 39
ASQCUR . . . 40
ASQFLD . . . 41
ASQHLT . . . 42
ASQLEN . . . 42
ASQMAX . . . 43
ASQMOD . . . 43
ASQNMF . . . 44
ASQSS . . . 44
ASRATT . . . 45
ASREAD . . . 45
ASRFMT . . . 47
ASTYPE . . . 48
CDPU . . . 50
CGLOAD . . . 51
CGSAVE . . . 54
DSCLS . . . 55
DSCMF . . . 56
DSCOPY . . . 57
DSDROP . . . 58
DSFRCE . . . 58
DSOPEN . . . 59
DSQCMF . . . 62
DSQDEV . . . 62

DSQUID 63
DSQUSE 63
DSRNIT 63
DSUSE 64
ESACRT 64
ESADEL 65
ESAQRY 65
ESASEL 65
ESEUDS 66
ESLIB 67
ESPCB 68
ESQCPG 68
ESQEUD 68
ESQOBJ 69
ESQUNL 69
ESQUNS 69
ESSCPG 70
ESSUDS 70
FSALRM 71
FSCHEK 71
FSCLS 72
FSCOPY 72
FSENAB 73
FSEXIT 74
FSFRCE 74
| FSGET 75
| FSGETE 75
| FSGETS 76
FSINIT 76
FSLOG 77
FSLOGC 77
FSOPEN 78
FSPCLR 79
FSPCRT 79
FSPDEL 80
FSPQRY 80
FSPSEL 81
FSPWIN 81
FSQCPG 82
FSQDEV 82
FSQERR 83
FSQSYS 84
FSQUPD 84
FSQUPG 84
FSQURY 85
FSQWIN 91
FSREST 91
FSRNIT 91
FSSAVE 92
FSSHOR 92
FSSHOW 93
FSTERM 94
FSTRAN 94
FSTRCE 95
FSUPDM 95
GSAM 96
GSARC 96
GSARCC 97
GSAREA 97
GSBMIX 98
GSBND 99
GSCA 99
GSCALL 100
GSCB 102
GSCBS 103
GSCD 103
GSCH 105
GSCHAP 105
GSCHAR 106
GSCLP 107
GSCLR 108
GSCM 108
GSCOL 109
GSCOPY 109
GSCORR 110
GSCORS 111
GSCP 112
GSCPG 113
GSCS 113
GSDEFE 114
GSDEFS 114
GSDSS 117
GSELPS 117
GSENAB 118
GSENDA 119
GSFLD 120
GSFLSH 121
GSFLW 121
GSGET 121
GSGETE 122
GSGETS 122
GSIDVF 123
GSIDVI 124
GSILOC 125
GSIMG 126
GSIMGS 127
GSIPIK 128
GSISTK 128
GSISTR 129
GSLINE 130
GSLOAD 130
GSLSS 133
GSLT 134
GSLW 134
GSMARK 135
GSMB 135
GSMIX 136
GSMOVE 137
GSMRKS 138
GSMS 138
GSMSC 138
GSPAT 139
GSPFLT 140
GSPLNE 141
GSPOP 142
GSPS 142
GSPUT 143
GSQAGA 143
GSQAM 144
GSQATI 144

GSQATS . . . 145
GSQBMX . . . 145
GSQBND . . . 146
GSQCA . . . 146
GSQCB . . . 146
GSQCBS . . . 147
GSQCD . . . 147
GSQCEL . . . 147
GSQCH . . . 147
GSQCHO . . . 148
GSQCLP . . . 148
GSQCM . . . 148
GSQCOL . . . 148
GSQCP . . . 149
GSQCPG . . . 149
GSQCS . . . 149
GSQCUR . . . 150
GSQFLD . . . 150
GSQFLW . . . 150
GSQLID . . . 151
GSQLOC . . . 151
GSQLT . . . 151
GSQLW . . . 152
GSQMAX . . . 152
GSQMB . . . 152
GSQMIX . . . 153
GSQMS . . . 153
GSQMSC . . . 153
GSQNSS . . . 153
GSQORG . . . 154
GSQPAT . . . 154
GSQPIK . . . 154
GSQPKS . . . 155
GSQPOS . . . 155
GSQPRI . . . 155
GSQPS . . . 156
GSQSEN . . . 156
GSQSIM . . . 157
GSQSS . . . 157
GSQSSD . . . 157
GSQSTK . . . 158
GSQSTR . . . 159
GSQSVL . . . 159
GSQTA . . . 159
GSQTAG . . . 160
GSQTB . . . 160
GSQTFM . . . 161
GSQVIE . . . 161
GSQWIN . . . 161
GSREAD . . . 162
GSRSS . . . 163
GSSAGA . . . 164
GSSATI . . . 165
GSSATS . . . 166
GSSAVE . . . 166
GSSCLS . . . 167
GSSCPY . . . 168
GSSCT . . . 168
GSSDEL . . . 169
GSSEG . . . 170
GSSEN . . . 170
GSSINC . . . 171
GSSORG . . . 171
GSSPOS . . . 172
GSSPRI . . . 172
GSSTFM . . . 173
GSSVL . . . 174
GSTA . . . 175
GSTAG . . . 176
GSUWIN . . . 176
GSVECM . . . 177
GSVIEW . . . 177
GSWIN . . . 178
IMACLR . . . 179
IMACRT . . . 179
IMADEL . . . 180
IMAGID . . . 180
IMAGT . . . 181
IMAGTE . . . 182
IMAGTS . . . 182
IMAPT . . . 183
IMAPTE . . . 184
IMAPTS . . . 184
IMAQRY . . . 185
IMARES . . . 186
IMARF . . . 186
IMARST . . . 187
IMASAV . . . 188
IMATRM . . . 188
IMPCRT . . . 189
IMPDEL . . . 189
IMPGID . . . 189
IMPRST . . . 190
IMPSAV . . . 190
IMRBRI . . . 191
IMRCON . . . 191
IMRCVB . . . 192
IMREX . . . 193
IMREXR . . . 193
IMRNEG . . . 194
IMRORN . . . 194
IMRPL . . . 195
IMRPLR . . . 195
IMRRAL . . . 196
IMRREF . . . 197
IMRSCL . . . 197
IMXFER . . . 198
ISCTL . . . 199
ISENAB . . . 200
ISESCA . . . 201
ISFLD . . . 201
ISIBOX . . . 202
ISILOC . . . 203
ISLDE . . . 204
ISQBOX . . . 204
ISQCOM . . . 205
ISQFLD . . . 205
ISQFOR . . . 206
ISQLOC . . . 206
ISQRES . . . 207

ISQSCA . . . 207
ISSE . . . 208
ISXCTL . . . 208
MSCPOS . . . 209
MSDFLD . . . 210
MSGET . . . 211
MSPCRT . . . 211
MSPQRY . . . 212
MSPUT . . . 212
MSQADS . . . 213
MSQFIT . . . 216
MSQFLD . . . 216
MSQGRP . . . 217
MSQMAP . . . 217
MSQMOD . . . 218
MSQPOS . . . 218
MSREAD . . . 219
PSDSS . . . 219
PSLSS . . . 220
PSLSSC . . . 221
PSQSS . . . 222
PSRSS . . . 222
PSRSV . . . 223
PTNCRT . . . 223
PTNDEL . . . 224
PTNMOD . . . 225
PTNQRY . . . 225
PTNQUN . . . 226
PTNSEL . . . 226
PTSCRT . . . 227
PTSDEL . . . 228
PTSQPI . . . 228
PTSQPN . . . 228
PTSQPP . . . 229
PTSQRY . . . 229
PTSQUN . . . 230
PTSSEL . . . 230
PTSSPP . . . 230
SPINIT . . . 231
SPMXMP . . . 231
SSQF . . . 232
SSREAD . . . 232
SSWRT . . . 233
WSCRT . . . 233
WSDEL . . . 234
WSIO . . . 235
WSMOD . . . 235
WSQRY . . . 236
WSQUN . . . 237
WSQWI . . . 237
WSQWN . . . 237
WSQWP . . . 238
WSSEL . . . 238
WSSWP . . . 239

Chapter 4. Device variations . . . 241
Operator windows, partitions, primary, alternate, and dual screens . . . 241
  Operator windows . . . 241
  Partitions . . . 241
  Primary and alternate screen sizes . . . 241
  Dual screen devices . . . 241
  Dual screen size . . . 241
Device-specific saved pictures . . . 242
  GDF saved as 2-byte integers . . . 242
  ADMSAVE files . . . 242
Screen redraw . . . 242
  Graphics primitives outside segments . . . 242
Programmed symbol sets (PS) and graphics text . . . 242
  Character mode 1 (hardware character sets) . . . 242
  Character mode 2 (image symbol sets) . . . 243
  PS stores and device cell-size dimensions . . . 243
  FSCHEK call . . . 243
Alphanumerics . . . 245
  Alphanumeric field attributes . . . 245
  Double-byte character sets (DBCS) . . . 245
  3278–52 . . . 245
  Alphanumeric colors . . . 245
Graphics colors . . . 245
Color mixing . . . 246
  Foreground color mix mode . . . 246
  Combinations of foreground and background mix modes . . . 246
Graphics line types and widths . . . 247
  Line types (GSLT) . . . 247
  Line widths (GSFLW and GSLW) . . . 247
Graphics area shading . . . 247
Graphics image . . . 247
Graphics logical input devices . . . 247
  Choice devices . . . 247
  Locator devices (GSILOC) . . . 247
  Pick devices (GSIPIK) . . . 248
  Stroke devices (GSISTK) . . . 248
  String devices (GSISTR) . . . 249
Image . . . 249

Chapter 5. APL request codes module . . . 251
The address table . . . 251
The request code table . . . 251
GDDM Base APL codes, in numeric order . . . 252

Chapter 6. GDDM-REXX programming interface . . . 255
GDDM-REXX commands, subcommands, and utilities . . . 255
  Summary . . . 255
  Syntax conventions . . . 255
GDDMREXX command . . . 255
  GDDMREXX INIT . . . 255
  GDDMREXX TERM . . . 256
  GDDMREXX VERSION . . . 256
GXGET subcommand . . . 256
  GXGET AAB . . . 256
  GXGET CDT . . . 256
  GXGET LASTMSG . . . 256
  GXGET MSG . . . 257
  GXGET NAMES . . . 257
  GXGET TRACE . . . 257
GXSET subcommand . . . 257
  GXSET AAB . . . 257
  GXSET MSADS . . . 257
  GXSET MSG . . . 257

GXSET MSVARS . . . 258
GXSET TRACE . . . 258
GDDM calls . . . 258
ERXMSVAR EXEC . . . 258

Chapter 7. Symbol sets . . . 261
How GDDM handles symbol sets . . . 261
Loading programmed symbol stores . . . 261
PS store numbers . . . 261
Symbol-set identification . . . 261
Using preloaded PS sets . . . 262
Selecting symbol sets by device type . . . 262
Using PS with graphics . . . 262
Loading graphics symbol sets . . . 262
PS overflow caused by picture complexity . . . 263
Using symbol sets in printing . . . 263
Using DBCS symbol sets . . . 264
Naming conventions for sample image symbol sets . . . 264
Sample image symbol sets . . . 265
Sample vector symbol sets . . . 265
Illustrations of vector typefaces . . . 267

Chapter 8. Symbol set formats . . . 275
Image symbol set component format . . . 276
Vector symbol set component format . . . 277

Chapter 9. GDDM object file formats . . . 279
Record structure . . . 279
The record identification field . . . 279
The header record information field . . . 279
Data records . . . 279

Chapter 10. GDF order descriptions . . . 281
Compatibility . . . 281
Saving GDF orders . . . 281
Format of GDF objects . . . 282
Coordinates and aspect ratio . . . 282
GDF orders: summary . . . 282
General structure . . . 283
Order formats . . . 283
Padding . . . 283
Primitives . . . 283
Attributes . . . 285
GDF orders: full descriptions . . . 285
Arc . . . 286
Arc parameters . . . 286
Area . . . 286
Background color mix . . . 286
Call segment . . . 287
Character angle . . . 287
Character box . . . 287
Character box spacing . . . 287
Character direction . . . 288
Character precision . . . 288
Character set . . . 288
Character shear . . . 288
Character string . . . 289
Color . . . 289
Comment . . . 290
Current position . . . 290
End area . . . 291
Fillet . . . 291
Foreground color mix . . . 291
Fractional line width . . . 292
Full arc . . . 292
Image – begin . . . 292
Image – data . . . 293
Image – end . . . 293
Line . . . 293
Line type . . . 294
Line width . . . 294
Marker . . . 294
Marker box . . . 294
Marker scale . . . 295
Marker type . . . 295
Model transform . . . 295
Pattern . . . 296
Pick (tag) identifier . . . 296
Pop . . . 296
Process specific control . . . 296
Relative line . . . 297
Segment attribute . . . 297
Segment attribute modify . . . 297
Segment characteristics . . . 298
Segment end . . . 298
Segment end prolog . . . 298
Segment position . . . 298
Segment start . . . 299
Segment viewing window . . . 300
Text alignment . . . 300
Process specific control orders (PSC) . . . 300
Symbol-set PSC orders . . . 301
Begin symbol-set mapping . . . 301
Map symbol-set identifier . . . 301
End symbol-set mapping . . . 302
Picture prolog . . . 302
Begin picture prolog . . . 302
End picture prolog . . . 302
Set default arc parameters . . . 302
Set default background mix . . . 302
Set default character angle . . . 302
Set default character box . . . 303
Set default character-box spacing . . . 303
Set default character direction . . . 303
Set default character precision . . . 304
Set default character set . . . 304
Set default character shear . . . 304
Set default coordinate type . . . 304
Set default extended color . . . 305
Set default foreground mix . . . 305
Set default fractional line width . . . 305
Set default line type . . . 306
Set default marker box . . . 306
Set default marker type . . . 306
Set default pattern . . . 306
Set default pick identifier . . . 307
Set default picture scale . . . 307
Set default text alignment . . . 307
Set default viewing window . . . 308
Set picture boundary . . . 308

Set picture origin 309

Chapter 11. Image file formats 311
Formats and compression types 311
3193 data stream and page printer formats 311
Unformatted data 311

Chapter 12. Picture interchange format files 315
Processing PIF files under TSO 315
The conversion operation 315
The transfer operation 315
Commands to use under TSO 316
The format of a PIF file 317
Processing PIF files under VM/CMS 317
The conversion operation 317
The transfer operation 317
Commands to use under VM/CMS 318
The format of a PIF file 319
Creating PIF data under GDDM 319
Creating PIF data at a workstation 319
How PIF data relates to GDF data 319
Base PIF 320
Restrictions and considerations 320
The structure of a PIF file 320

Chapter 13. Computer Graphics Metafiles 323
Application program calls 323
CGLOAD 323
CGSAVE 324
Utility programs 324
ADMUCG 324
ADMUGC 325
SEND and RECEIVE 327
External defaults 327
CGM file format 327
National language code pages 328
Conversion profiles 329
Format of a conversion profile 329
Picture mapping information 329
GDF order processing (CGSAVE call) 339
CGM order processing (CGLOAD call) 340

Chapter 14. Graphics Interchange Format (GIF) files 343
GIF file structure 343
ADMUGIF 343
Keyword parameters 343
Invoking ADMUGIF under VM/CMS 344
Invoking ADMUGIF under MVS/TSO 344
For both VM/CMS and MVS/TSO 345

Chapter 15. Format of a Composite Document Presentation Data Stream 347
Document structure 347
Structured fields 348
Summary of structured fields 348
Structured field formats 349
AFPDS structured fields supported by the CDPU . . . 356
Summary of AFPDS structured fields supported by the CDPU 356

Chapter 16. Application data structure for mapping 357
Adjunct fields 357
COBOL example 358
Assembler language example 358
PL/I example 358
Adjunct field names 358
Adjunct values 358
Character attributes 363
Setting character attributes from the terminal 364
Designator characters for light-pen or cursor selection . 364
Map-defined input editing 365
AID translation 365
Folding 365
Justification and padding 365
Copying the application data structure into the program .366
Overlaying application data areas 366
Double-byte character string fields 366
Mixed double-byte and single-byte character fields in maps 367
GDDM-supplied mapping constants 367

Chapter 17. GDDM high-performance alphanumerics 369
HPA data structure 369
The field list 369
The data buffer 372
The bundle list 373
How to use high-performance alphanumerics 375

Chapter 18. External defaults 379
GDDM's default values 379
Changing GDDM's default values 379
External defaults: format 379
Alphabetical list of GDDM external defaults 384

Chapter 19. Processing options 395
Processing options 395
Processing options: format 395
Processing options: full descriptions 396

Chapter 20. Name-lists 415
Reserved names "*" and blanks 415
Family-1 name-list 415
CICS name-list 415
VSE/Batch name-list 416
IMS name-list 416
TSO name-list 416
MVS/Batch name-list 417
VM name-list 418

Chapter 21. Device characteristics tokens 421
Creating your own device tokens 421
Device tokens for ASCII graphics displays 421
GDDM-supplied device tokens 421

Chapter 22. Special-purpose programming in GDDM 431
Using the system programmer interface 431
Initialization 431
The system programmer interface block 432

Format of the system programmer interface block . . 432
Specifying user exits 432
Exit values 433
GDDM user-exit conventions 433
The task switch exit 434
The call intercept exit 435
The coordination exit 436
Storage exit routines – interface specifications . . . 437
Call format descriptor module 438
The address table 438
The call descriptor table 438
The parameter descriptor table 440

Glossary 443

Index 457

Figures

1. GDDM default EBCDIC character codes (code page 00351) 49
2. Katakana character codes (Tables 32772, 32792, and 32793) (code page 00290) 49
3. Japan (Latin) extended character codes (code page 01027) 50
4. Character angle and mode-2 text positioning (GSCA) 100
5. Character direction (GSCD) 104
6. Character shear (GSCH) 105
7. How an ellipse is drawn (GSELPS) 118
8. Sample geometric shading patterns (GSPAT) . . 140
9. GDDM-defined shading patterns (GSPAT) . . . 141
10. Curved fillets (GSPFLT) 141
11. Text box enclosing rotated characters (GSQTB) 160
12. Example of how a viewport is defined (GSVIEW) 178
13. Structure returned from MSQADS 214
14. GDF file conversion – format 1 to format 2 . . . 281
15. Accepted data streams (3193DSF and PPF) . . 312
16. IMAGT data streams from GDDM 313
17. GDF file conversion procedure under TSO . . . 316
18. GDF file conversion procedure under VM/CMS . 318
19. The structure of a PIF file 321
20. Example of a conversion profile. 328
| 21. CGM color_mapping keyword 334
| 22. Character-height and character-width factors for conversion of fonts between ADMGDF and CGM formats. 338
23. Field list array 370
24. The bundle list array 374

Tables

1. Example color “mix” mode table (GSMIX) 136
2. Results of exclusive-OR mode (GSMIX) 137
3. Device classes for GDDM Interactive Map Definition (MSPCRT) 212
4. Device cell-size dimensions 243
5. Triggering keys for locator and pick devices . . 248
6. APL request codes module – address table . . . 251
7. APL request codes module – request code table 251
8. GDDM Base APL codes, in numeric order . . . 252
9. PS store number and PS key relationship 261
10. Cell sizes for sample image symbol sets 264
11. Sample image symbol sets 265
12. Sample vector symbol sets 265
13. Symbol-set definition format 275
14. Image symbol set component format 276
15. Vector symbol set component format 277
16. GDDM object types 279
17. GDDM stored object file format 279
18. GDDM stored object — record identification field format 279
19. GDDM stored object — header record information field format 279
20. Summary of GDF orders in order of code values 282
21. Alphabetic summary of GDF order codes and usage 284
22. Numeric list of Symbol-set process specific control orders. 301
23. Numeric list of Picture prolog process specific control orders. 301
24. Valid combinations of format and compression . 311
| 25. GDDM-supplied conversion profiles for conversion
| of data between ADMGDF and CGM formats. . 323
26. Picture Mapping Keywords 329
27. Conversion default values 330
28. Color Table created by CGSAVE/ADMUGC . . . 330
29. Keyword Defaults 331
| 30. CGM color_mapping parameter 332
| 31. GDDM colors used for CGM interpretation . . . 332
32. GDF patterns and CGM mapping 335
33. GDDM-supplied conversion profiles 338
34. CGM test patterns for conversion profile creation 338
35. GDF order processing 339
36. CGM order processing 340
37. Structured fields in code order 348
38. Structured fields in alphabetic order 348
39. Format of bar code data 349
40. AFPDS structured fields supported by the CDPU 356
41. Adjunct field naming conventions 358
42. Values used in adjunct fields 359
43. Character attribute types and values 364
| 44. GDDM mapping constants tables 367
45. GDDM external defaults 379
46. Summary of processing options and nickname keywords 395
47. GDDM-supplied device tokens 421
48. GDDM-supplied device tokens for Kanji devices, and 3290 displays (family 1) 425
49. GDDM-supplied device tokens for support of ASCII devices (family 1) 426
| 50. GDDM-supplied device tokens for nonqueriable
| displays and printers (family 1) 426
51. GDDM-supplied device tokens for GDDM-PCLK displays, printers, and plotters (family 1) 427
52. GDDM-supplied device tokens for system printers (family 3) 428
53. GDDM-supplied device tokens for cell-based AFPDS page printers (family 4) 428
| 54. GDDM-supplied device tokens for PostScript
| printers (family 4) 429
55. GDDM-supplied device tokens for page printers (family-4) 430
56. SPIB format 432
57. GDDM exits — options 433
58. Call format descriptor module – address table . 438
59. Call format descriptor module – call descriptor table 439
60. Call format descriptor module – parameter descriptor table 441

Notices

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

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you. References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM's 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 user's responsibility.

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

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

Programming interface information

This book is intended to help you program the IBM Graphical Data Display Manager (GDDM). This book primarily documents General-use Programming Interface and Associated Guidance Information provided by GDDM.

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

However, this book also documents Product-sensitive Programming Interface and Associated Guidance Information provided by GDDM.

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 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 and/or other countries:

APL2
CICS
GDDM
graPHIGS
IBM
OS/390
Personal System/2
PS/2
System/370

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

Corel Draw	Software Publishing Corporation
DEC	Digital Equipment Corporation
Freelance Plus	Lotus Corporation
Harvard Graphics	Corel Systems Corporation
HELVETICA	Allied Corporation
Micrografx Designer	Micrografx Inc.
Tektronix	Tektronix Inc.
TIMES	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

What this book is about

The *GDDM Base Application Programming Reference* book provides the reference information that is needed to program with the IBM GDDM Version 3 Release 2 series of products, and with GDDM/MVS as an element of OS/390.

Who this book is for

This book is for application programmers who use GDDM.

What you need to know

You must have some knowledge of programming and systems terminology, and some familiarity with this release of GDDM, to understand what task your application program is trying to perform.

How to use this book

This book contains introductory and conceptual information that can be read sequentially. It also contains descriptions of the calls in alphabetical order. If you know the function of a call but not its name, refer to Chapter 2, “A summary of the calls by function” on page 13.

Terminology used

Throughout this book, these conventions are used:

GDDM
The unqualified term “GDDM” means the GDDM Base product, unless otherwise stated.

Personal computer (or PC), personal computer system, or workstation.
These terms refer to any member of the IBM Personal Computer family, including the Personal System/2 (PS/2) family, unless otherwise stated.

3270-PC/G and 3270-PC/GX
References to these devices also refer to the 3270-PC AT/G and the 3270-PC AT/GX, unless otherwise stated.

Terminal
This term means any computer terminal or workstation, unless otherwise stated.

Summary of changes

This softcopy-only edition corrects references to Hong Kong and Taiwan.

The following major changes have been made for GDDM 3.2:

- The DSFRCE, FSGET, FSGETE, and FSGETS calls have been added. See Chapter 3, “The GDDM calls” on page 21.
- New symbol sets are supported. See Chapter 7, “Symbol sets” on page 261.
- The information in Chapter 13, “Computer Graphics Metafiles” on page 323 has been enhanced.
- Information on graphics interchange format (GIF) files has been added. See Chapter 14, “Graphics Interchange Format (GIF) files” on page 343.
- The FRCETYPE processing option has been added. See Chapter 19, “Processing options” on page 395.
- New device tokens are supported. See Chapter 21, “Device characteristics tokens” on page 421.

Summary of changes for GDDM 3.1

The *GDDM Base Application Programming Reference* book replaces the former *GDDM Base Programming Reference* (in two volumes). The order and contents of chapters have been rearranged to provide logical groupings of information. Descriptive material has been moved to other books in the library, while reference material has been concentrated in this book. Changes in individual chapters include:

- The “GDDM programming interface” chapter now includes information on the interface to REXX, which was formerly contained in the *GDDM-REXX Guide*, SC33-0478.
- A new chapter on “Device variations” has been added. This chapter summarizes, for each broad group of GDDM functions, the devices or groups of devices that do not conform to the general description of the function.
- The “APL request codes module” chapter is confined to a list in numerical order of the GDDM Base APL codes.
- A new chapter on “GDDM-REXX programming interface” has been added from material formerly contained in the *GDDM-REXX Guide*, SC33-0478.
- Illustrations of vector typefaces have been taken from the former *GDDM Typefaces and Shading Patterns*, SC33-0554 and added to the “Symbol-set formats and files” chapter.
- The tables of external defaults for each sub-system in the “External defaults” chapter have been consolidated into a single table.
- Information on the Call format descriptor module has been merged with Chapter 22, Special-purpose programming in GDDM.
- Presentation of devices tokens has been improved.
- Descriptions of the C/370 interface and of two new programming calls, DSCOPY and ESQUNS, have been added.

changes

More 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

Chapter 1. GDDM programming interface

This chapter explains:

- The external programming interfaces to GDDM that are available, and how to use them
- The syntax conventions for coding GDDM calls
- The types of data that are required for parameters in the calls
- Programming-language-dependent considerations.

Access to the GDDM functions by the application program is by GDDM interface modules that are link-edited or loaded with the program. The interface modules convert call statements in the program to a standard internal interface to invoke the GDDM functions. This makes GDDM itself independent of the subsystem being used, and allows the use of three different application interfaces:

Nonreentrant interface
This is the standard interface for most application programs that use GDDM and do not require any special processing. Quasi-reentrancy (as defined by CICS) can be achieved using this interface.

Reentrant interface
This allows the programs using GDDM to be made reentrant with the advantages that reentrancy provides; that is, the ability of the program to be used by more than one user at the same time.

System programmer interface
This is provided for programmers who intend to use GDDM as the basis for a graphics system of their own. It enables GDDM functions to be written in a coded form, it gives greater control over the subsystem environment, and it allows more programming flexibility.

Notes:

1. An application program using the nonreentrant interface cannot use either of the other interfaces.
2. An application program can use the reentrant and system programmer interfaces interchangeably.

The nonreentrant interface

The nonreentrant interface applies to application programs that need not be reentrant; for example, a program written in FORTRAN.

Each CALL statement takes the form:

```
CALL ffffff (parameter,...)
```

where `ffffff` and the parameters are the appropriate GDDM call name and parameters as described in Chapter 3, "The GDDM calls" on page 21.

On return to the program, all registers except Register 15 are restored to their entry values. The top (high-order) half of Register 15 is set to one of these error severity codes:

0 Normal
4 Warning
8 Error
12 Severe error
16 Irrecoverable error.

The bottom (low-order) half of Register 15 contains the error code identifying the particular response.

Additional error feedback information can be obtained by using the GDDM FSQERR call or by using an error exit specified in the FSEXIT call. These calls are described in Chapter 3, "The GDDM calls" on page 21 and in the *GDDM Base Application Programming Guide*.

Under CICS, the nonreentrant form of interface is usable only if specific extra actions are taken to make the program quasi-reentrant (as defined by CICS). These actions are described in the *GDDM Base Application Programming Guide*. Subject to these considerations, this form of interface actually has quasi-reentrant characteristics.

The reentrant interface

Application programs requiring reentrancy can use another form of the CALL statement:

```
CALL ffffff (aab, parameter,...)
```

where `ffffff` and the parameters are as described for the nonreentrant interface, and `aab` (application anchor block) is an application-provided, word-aligned control block of this format:

Offset	Length	Name	Description
0	8	AAB	Application Anchor Block
0	4	AABFC	GDDM feedback code
0	2	AABSC	GDDM severity code
2	2	AABEC	GDDM error code
4	4	AABAP	GDDM anchor pointer

When using this interface, the application program must provide the storage for the application anchor block (at least 8 bytes for the use of GDDM). The program is free to extend the application anchor block for other uses (typically, to provide for passing information to an error exit routine).

The GDDM anchor pointer (AABAP) is set by the GDDM Application Interface Component at initialization, and identifies the GDDM instance being addressed. It is reset to zero on termination. This pointer helps GDDM's reentrancy, and is used by GDDM to retain storage across activations.

The severity code (AABSC) is set to the error severity code (see above) on return to the application, and the error code

(AABEC) identifies the particular response. As for the nonreentrant interface, more error information can be obtained by using the FSQERR and FSEXIT calls.

Reentrancy of the GDDM invocation is determined by the reentrant properties of the application anchor block. If the application anchor block is in reentrant storage, GDDM is reentrant. If the application anchor block is in quasi-reentrant storage, GDDM is quasi-reentrant.

If the application program is modular, and reentrant use of GDDM from several modules is required, each such module must have access to the application anchor block. For example, the program can pass the application anchor block as a parameter across its module calls. Alternatively, under a subsystem such as CICS, quasi-reentrancy can be achieved by locating the application anchor block in the application program's transaction work area (TWA).

The system programmer interface

The system programmer interface is a special interface available to "system programming" types of applications. It is available only in reentrant form, and shares many features with the reentrant interface.

Each call takes the form:

```
CALL ADMASP (aab, rcp, parameters,...)
```

where `ADMASP` is the defined system programmer interface entry point, `aab` is as defined for the reentrant interface, `rcp` is the request control parameter (defined below), and `parameters` are the parameters for the function specified in the request control parameter.

The request control parameter (RCP) is a 4-byte, fullword-aligned function code defining the GDDM function to be called. For a definition of the format of the RCP address table, call description table, and parameter description table for GDDM calls, see Chapter 22, "Special-purpose programming in GDDM" on page 431. The GDDM request control parameter code is given, in both hexadecimal and decimal format, for each GDDM call listed and described in Chapter 3, "The GDDM calls" on page 21.

ADMASP is a single entry point resolved by the GDDM interface modules that are link-edited or loaded with the application. The use of the application anchor block is as described for the reentrant interface. Calls to the system programmer and reentrant interfaces may be mixed, if the same application anchor block is passed on each call.

In the simplest case, the system programmer interface merely provides a means of accessing a GDDM function by a function code (the RCP) rather than by selecting an entry point. Assembler-language macros defining mnemonics for these function codes are provided.

This interface provides an alternative initialization function (SPINIT) that allows control of environmental aspects. For more information on the system programmer interface, see Chapter 22, "Special-purpose programming in GDDM" on page 431. A summary list of RCP codes for GDDM (and GDDM-PGF) is given in the *GDDM Diagnosis* book.

Application programming language considerations

This section provides information about the application programming languages supported by GDDM, which are as follows:

- APL
- Assembler
- BASIC
- C
- COBOL
- FORTRAN
- PL/I
- REXX

APL

GDDM supports APL2. The *APL2 Programming: System Services Reference* manual, describes AP 126 (the GDDM Auxiliary Processor).

In Chapter 3, "The GDDM calls" on page 21, the APL code that corresponds to the name of a GDDM call is given within the description of the syntax for that call. Also, the APL codes with the call names are listed in numerical order in Chapter 5, "APL request codes module" on page 251.

The GDDM Auxiliary Processor, AP 126, manages requests from an APL program. To use it, follow this procedure:

1. Offer to share a pair of variables with AP 126 to be used as control and data variables. The control variable name must begin with CTL, and the data variable name must begin with DAT. If names longer than three characters are used for these variables, the names must be identical after the third character and must be no longer than 11 characters.
2. Check that the degree of coupling returned is two for each variable.
3. If the call requires any character data, use an APL specification statement to assign the data to the data variable.
4. Assign to the control variable the request code for the GDDM call to be issued, along with any required numeric parameters.
5. Ensure that the request has completed successfully by referencing the control variable; 0 is returned as the first element of the control variable if the request was successful. The second and third elements are the severity level and error codes for the GDDM call. These will also be 0 if the request was successful. The fourth and fifth elements are the lengths of the numeric and character

data returned, respectively. Any numeric data returned by the call is given after the fifth element. Any character data returned is given in the data variable.

For example, to share a pair of variables called CTL126 and DAT126 with AP 126, then issue the GSCOL call to select color 4:

```
      126 ⎕SVO ¨ 'CTL126' 'DAT126'
2 2               ⍝ Degree of coupling should be 2
      CTL126←514 4 ⍝ Issue GSCOL call, color 4
      CTL          ⍝ Display return code
0 0 0 0 0          ⍝ Zero means call was successful
```

If you are using APL2, Version 1 Release 3 or later, you can use the GDMX function. This function allows the name of the GDDM call to be used directly, rather than the numeric code that AP 126 requires. It performs all the necessary return code checking. Here is the same example using GDMX:

```
'GSCOL' GDMX 4 ⍝ Issue GSCOL call, color 4
```

Note: GDDM is automatically initialized and terminated by APL2. No explicit FSINIT or FSTERM calls are needed to start and stop usage of GDDM by AP 126.

Assembler language

In Assembler language, linkage is performed according to the usual operating system (OS) conventions:

1. Register 1 points to the address list containing the parameter addresses. The high-order bit must be set in the last word. If no parameters are to be passed, Register 1 must contain the value 0 or must point to a fullword of value 0 with the high-order bit set.
2. Register 13 points to a register save area of at least 72 bytes (18 fullwords).
3. Register 14 points to the return point in the application program.
4. Immediately before a call to GDDM, Register 15 points to the entry point with the name “callname.” On return from a call to GDDM, the top half of register 15 contains the error severity code; the bottom half contains the error number.
5. A branch to “callname” is performed.

Parameters must be declared as:

Fullword integer	F-constant
Halfword integer	H-constant
Floating-point	E-constant
Character	C-constant
Array	Contiguous fullwords or halfwords
Structures	Use the appropriate storage mapping.

In Assembler language, linkage can be performed using the OS CALL macro, coded with the VL option. For example:

```
         CALL ASFCOL,(ID,COLOR),VL
         .........
ID       DC   F'1'
COLOR    DC   F'3'
```

Note: Calls with no parameters must be coded in this form to ensure that register 1 is set correctly:

```
CALL FSINIT,(0),VL
```

BASIC (IBM)

A call interface to GDDM is provided by IBM BASIC. The first argument to the CALL GDDM command is the request control parameter code; for example:

```
100  ASDFLD=201852672
110  CALL GDDM (ASDFLD,2,30,44,1,1,0)
```

More information on using IBM BASIC with GDDM is given in the publications for IBM BASIC.

C

In C/370 you must declare each GDDM function as using OS linkage. To do this, use the pragma linkage preprocessor directive. For example:

```
#pragma linkage(fsinit,OS)
```

Declarations for the functions can also be provided which detail parameter types. For example, when using the nonreentrant interface:

```
extern int gsline(float x,float y);
```

When using the reentrant interface:

```
extern int gsline(char aab[], float x, float y);
```

Header files containing linkage directives and function declarations for GDDM functions are provided on the GDDM installation tape. Files with names of the form ADMUCINx are for use with the nonreentrant interface. Sets with names of the form ADMUCIRx are for use with the reentrant interface.

The functions are grouped in the header files by name, as follows:

Nonreentrant use	Reentrant use	Calls starting ...
ADMUCINA	ADMUCIRA	A
ADMUCIND	ADMUCIRD	D
ADMUCINE	ADMUCIRE	E
ADMUCINF	ADMUCIRF	F
ADMUCING	ADMUCIRG	G
ADMUCINI	ADMUCIRI	I
ADMUCINK	ADMUCIRK	CD and CG
ADMUCINM	ADMUCIRM	M
ADMUCINP	ADMUCIRP	P
ADMUCINS	ADMUCIRS	S
ADMUCINW	ADMUCIRW	W

To include these header files in a C/370 program, use include statements. For example:

```
#include <admucina.h>
#include <admucinf.h>
#include <admucing.h>
```

All calls from the system programmer's interface are made using the admasp function. This can be made available in C/370 programs by using the linkage convention. For example:

```
#pragma linkage(admasp,OS)
```

The header files contain sample declarations for the types used in GDDM functions. Specifying the types allows for a check for mismatched types at compilation. If possible, the types specified are converted to the type required by the compiler.

Declaring variables: This section describes how to declare variables to match the function declarations used in the header files.

Parameters as described for each function should be declared as follows:

Fullword integer Int

Halfword integer Short

Floating point Float

Character Array of type char

Array of n-byte character tokens
Two-dimensional array of type char

Arrays Use a one-dimensional of the type specified.

Structures Use appropriate storage mapping.

For example:

Fullword integers, halfword integers, and floating point values

Specified by user:

```
int n=2;
gscol(n);
```

Returned by GDDM:

```
int n;
gscol(&n);
```

or

```
int *n;
gscol(n);
```

Note: To receive values returned by GDDM, the parameter should be an address. To ensure this, either use the address(&) operator, or declare the variable to be used as a pointer(*).

Array of fullword integers:

```
int array[10];
apqids(1,10,array);
```

Note: Because *array* is a pointer to the first element in the array of fullword integers, it can be used in the same way for parameters specified by the user and returned by GDDM.

Character: Parameters described as being of type 'character' should be declared as *char xxx*[*n*], where *xxx* is the name of the variable, and *n* is the number of characters it is to contain. For example:

```
char string[7];
strncpy(string,"EXAMPLE",7);
gschar(10.0,10.0,7,string);
```

The strncpy function is used to avoid adding the NULL character to the variable *string*. C uses the NULL character to delimit strings.

Parameters described as 'an array of *n* byte character tokens' should be declared as char *xxx*[*a*][*n*], where *xxx* is the variable name, *n* is the number of bytes in each character token, and *a* is the number of such tokens required. For example:

```
char nlist[2][8];
strncpy(nlist[0],"*       ",8);
strncpy(nlist[1],"ADMPLOT ",8);
```

This example initializes a variable *nlist*. This could be used for the name list parameter on a dsopen call, to open a directly attached plotter. The source strings are padded to 8 characters with blanks. This is to avoid the NULL character being placed anywhere in the target string. Because all the character variables are arrays, they are passed in the same way, whether they are values specified by the user or returned by GDDM.

Structures: Some parameters to GDDM calls can use structures to split information into separate fields. The appli-

cation anchor block parameter, which is used on all reentrant calls, is an example of this. It can be declared as a structure of the following form:

```
struct {
        int feedback;
        int *anchor;
       } aab;
```

The header file ADMTSTRC.H is provided with sample type definitions for structures used in GDDM calls. These types are as follows:

Admaab (Application anchor block)
The first parameter passed to all REENTRANT application calls.

Admccs (Chart control structure)
Used for the chart_control parameter in the CHART call.

Admers (Error record structure)
The structure returned by GDDM from the FSQERR call. It is also passed to any user-specified error exit defined by the FSEXIT call.

Admspib (Spib block)
The structure used in the SPINIT call, to initialize the System Programmer's Application Call Interface.

You can declare a variable to be any of these types. The variable can then be passed directly to the functions using it, by first 'casting' the type to be that expected by the call. Alternatively, a union operator can be used to map the structure to the same area of memory as a variable of the type expected by the function. The following examples show both ways of using the structures.

In the following example, the variable *error* is cast to be of the type expected by the function. The address operator (&) is used because the parameter (error) is being returned by GDDM.

```
Admers error;

fsqerr(160,(char *)&error);
```

In the following example, the function is passed a variable of the type it expects. Use of the union operator has ensured that the structure begins at the same place in memory as the variable. Because of this, the elements in the structure will be set on return from the fsqerr call.

```
union {
       Admers error;
       char str[160];
      } u;

fsqerr(160,u.str);
```

The address of the error record structure is also passed by GDDM to an error exit specified by means of the fsexit function. The sample Admers type can also be used to declare the parameters to the error exit function. The following example shows a possible declaration for such a function:

```
void err_exit(Admers *ers);
```

Casting must be used to specify this function in the fsexit function. For example:

```
fsexit((int)&err_exit,8);
```

COBOL

The call format for COBOL is as follows:

```
CALL 'callname' USING parameter-1,parameter-2,
     ...parameter-n.
```

For example,

```
MOVE 1 TO ID.
MOVE 3 TO COLOR.
CALL 'ASFCOL' USING ID,COLOR.
```

Parameters must be declared as:

Fullword integer	PICTURE S9(8) COMPUTATIONAL or equivalent
Halfword integer	PICTURE S9(4) COMPUTATIONAL or equivalent
Floating-point	COMPUTATIONAL-1
Character	PICTURE X (n)
Arrays	The OCCURS clause.
Structure	`01 structure-name.` `COPY mapname.`

FORTRAN

Versions of the VS FORTRAN compiler earlier than Release 3.0 cause extra parameters to be generated for character items in CALL statements, when the **LANGLVL(77)** compiler option is in effect; GDDM does not accept these extra parameters. Therefore, to prevent these extra parameters being generated, the **SC** compiler option must be specified.

The current release of VS FORTRAN does not generate these extra parameters, and so the **SC** compiler option is not required.

Parameters must be declared as:

Fullword integer	INTEGER*4
Halfword integer	INTEGER*2
Floating-point	REAL*4
Character	String literals or numeric data array initialized with string literals
Arrays	A one-dimensional array. (You can use a multidimensional array if it causes the correct storage mapping. FORTRAN arrays are stored in column-major order.)
Structures	No FORTRAN application data structure is generated by GDDM-IMD.

For example:

```
CALL FSINIT
CALL GSSEG (0)
```

PL/I

In PL/I, it is necessary to declare each GDDM function used as an external entry; for example:

when using the nonreentrant interface:

```
DECLARE FSREST ENTRY (FIXED BINARY (31))
               EXTERNAL OPTIONS (ASM,INTER);
```

when using the reentrant interface:

```
DECLARE FSREST ENTRY (*,FIXED BINARY (31))
               EXTERNAL OPTIONS (ASM,INTER);
```

when using the system programmer interface:

```
DECLARE ADMASP EXTERNAL ENTRY OPTIONS (ASM,INTER);
```

Some data sets of PL/I DECLARE statements for GDDM functions are provided on the GDDM installation tape. Data sets with names of the form ADMUPINx are for use with the nonreentrant interface, and data sets with names of the form ADMUPIRx are for use with the reentrant interface. These data sets are provided:

Nonreentrant use	Reentrant use	Calls starting...
ADMUPINA	ADMUPIRA	A
ADMUPIND	ADMUPIRD	D
ADMUPINE	ADMUPIRE	E
ADMUPINF	ADMUPIRF	F
ADMUPING	ADMUPIRG	G
ADMUPINI	ADMUPIRI	I
ADMUPINK	ADMUPIRK	CD and CG
ADMUPINM	ADMUPIRM	M
ADMUPINP	ADMUPIRP	P
ADMUPINS	ADMUPIRS	S
ADMUPINW	ADMUPIRW	W

All these data sets can be incorporated into a PL/I program by %INCLUDE statements, for example:

```
%INCLUDE SYSLIB(ADMUPINA);
%INCLUDE SYSLIB(ADMUPING);
%INCLUDE SYSLIB(ADMUPINF);
```

For compatibility with programs written for GDDM Version 1 Release 1 or Release 2, two other data sets of PL/I declarations are provided. Each of these data sets contains PL/I declarations for **all** the GDDM functions that are available in Release 2. These data sets do not include the new functions that have been available since Version 1 Release 2. The data sets are:

ADMUPLNB All Version 1 Release 2 functions (for nonreentrant use)

ADMUPLRB All Version 1 Release 2 functions (for reentrant use).

Note: These data sets contain **sample** declarations for the GDDM functions. These declarations have parameter descriptions that match exactly the parameter specifications in the descriptions of the call statements. The declarations allow a check at compilation that parameters have been correctly declared, and, if necessary, automatic conversion by dummy parameters.

You can modify the parameter descriptors in these data sets to allow more flexibility in the declarations, but you must be aware that checking or conversion may not occur if this is done.

Where a parameter of a GDDM function is listed as an array, the sample declarations assume that the array is one-dimensional. You can change the declaration of the array parameter while still maintaining the same storage mapping. For example, the sample declaration of the ASDFMT call for nonreentrant use is:

```
DECLARE ASDFMT ENTRY (BIN FIXED(31),
                      BIN FIXED(31),
                      (*) BIN FIXED(31) CONN)
               EXTERNAL OPTIONS (ASM,INTER);
```

The declaration and call for this definition would be:

```
DECLARE A(15) BIN FIXED(31);
CALL ASDFMT(3,5,A);
```

If you want, you could redefine the declaration of ASDFMT as:

```
DECLARE ASDFMT ENTRY (BIN FIXED(31),
                      BIN FIXED(31),
                      (*,*) BIN FIXED(31) CONN)
               EXTERNAL OPTIONS (ASM,INTER);
```

The declaration and call for this definition would be:

```
DECLARE A(5,3) BIN FIXED(31);
CALL ASDFMT(3,5,A);
```

Parameters must be declared as:

Fullword integer FIXED BINARY(31)
Halfword integer FIXED BINARY(15)
Floating point FLOAT DECIMAL (6)
Character CHARACTER(n)
Note: Must not be VARYING.
Arrays A one-dimensional array. (You can use a multidimensional array if it causes the correct storage mapping. PL/I arrays are usually stored in row-major order.)
Structures

```
DECLARE  1 structure-name
%INCLUDE  mapname;
```

REXX

GDDM Base is available to programs written in REXX. You can use any of the GDDM calls, except as noted in “Restrictions” on page 10 and “Differences” on page 10.

GDDM call syntax: In GDDM-REXX, a GDDM call takes the form of the call name followed by a list of parameters. Parameters are separated by blanks both from the call name and from other parameters. Parts of the call can be enclosed in either single or double quotes to prevent processing by REXX.

Methods of passing parameters: Parameters may be REXX variable names or literal values. When parameters are variables, they must be preceded by a period (.).

```
/* variables                          */
'GSLINE .x .y'
/* constants                          */
'GSLINE 10 20'
/* mixture of variables and constants  */
'GSLINE .x 20'
/* variable character string  */
'ASCPUT .id .length .words'
/* constant character string */
'ASCPUT   1     5   "Hello"'
```

The parameters passed can be any type of REXX variable; the correct conversion will be made to the type required by GDDM-REXX.

Note that you cannot pass a variable that contains a string of parameters. For example, you cannot use

```
parmstring='10 20'; 'GSLINE .parmstring'
```

Values that can be passed: The values passed can either be integer, floating point, or string. Strings may be of variable length or of fixed length. Fixed-length strings are sometimes called tokens. Values allowed are shown below.

Integer

Range from –2**31 to 2**31–1. Floating point and a decimal point are allowed provided any fractional digits are zero. Take care with negative values. Include quotes `'GSCOL -1'` or parentheses if outside quotes `'GSCOL' (-1)`. Note that the REXX NUMERIC DIGITS instruction has no effect on the range.

Examples: `1  21  1234  -1`

Float

Floating point or decimal notation. Maximum and minimum restricted to System/370 short floating point form. However, GDDM graphics calls limit values to a smaller range (absolute values of nonzero parameters in the range 1.0E–18 to 1.0E18).

Examples: `1.3  1e+3  1E-3  1.0e3`

Strings (fixed length)

Most are names of GDDM objects and should be coded in uppercase. GDDM does not recognize the lowercase versions.

Example: `ADMUUKSF`

Strings (variable length)

Enclosed in double quotes or single quotes. If GDDM calls are in single quotes (as advised), double quotes should be used. If you want a quote displayed or printed, use it twice if it is already used as a string delimiter.

Examples: `"String"  'string'  'don''t'`
`"He said, ""Don't"""`

Types of parameters: Parameters must be either scalars (a single value) or arrays (a list of values with defined dimensions).

Strings: GDDM-REXX lets you pass a character string that contains DBCS (double-byte character set) characters. They must be between Shift-out and Shift-in (SO/SI) brackets. (SO = X'0E' and SI = X'0F'.) It is an error if an unmatched SO occurs in a string.

Under certain circumstances, strings or token parameters may be entered without quotes. However, you are recommended to enclose strings and token parameters in quotes in all cases. Parsing can be interrupted if you omit required quotes.

The following example shows strings containing special characters enclosed in matching quotes.

```
/* string containing blanks       */
s = 'a b c'
'GSCHAR 30 50 5' s
/* "s" is evaluated - the command */
/* passed to GDDM after evaluation*/
/* is 'GSCHAR 30 50 5 a b c'      */
/* which has too many parameters  */
```

Character strings that contain blanks or DBCS characters can be passed in variables without the need for any special GDDM-REXX string delimiters. However, the system interpreter requires an `OPTIONS ETMODE` statement before the SO/SI characters in literal strings or comment statements.

Passing array parameters: Array parameters can be passed in the following ways:

1. Using a REXX stemmed variable. The array will be taken from the member .1 or .1.1 (and so on).

   ```
   xarray.1=10; xarray.2=20 /* and so on */
   yarray.1=10              /* and so on */
   /* Note dot after stemmed name    */
   'GSPLNE 3 .xarray.  .yarray.'
   ```

2. Using a prefix. REXX variables of the form **prefix1**, **prefix2**, for one dimension and **prefix1.1**, **prefix1.2** for two dimensions will be looked for and their values used. The variables with these new names are formed into a list and passed as an array to GDDM. For example:

   ```
   vx1=10; vx2=8; vx3=5     /* and so on */
   'GSPLNE 3 .vx .vy'
   ```

   is passed to GDDM as

   ```
   'GSPLNE 3 (.vx1 .vx2 .vx3) (.vy1 .vy2 .vy3)'
   ```

3. Enumerated between parentheses. For example:

   ```
   bot=5; mid=25
   /* each member */
   /* enumerated in parentheses      */
   'GSPLNE 3 (10 20 .bot)(20 .mid 30)'
   ```

   or, for a two-dimensional array:

   ```
   'CHBAR 3 2 ((10 40) (20 50) (30 60))'
   ```

4. By stem or prefix names in parentheses; these represent columns in a two-dimensional array. For example:

```
b.1=10; c.1=40
b.2=20; c.2=50
b.3=30; c.3=60
/* dimensioned values placed in   */
/* parentheses when required      */
/* array needs more dimensions    */
'CHBAR 3 2 (.b. .c.)'
```

   Note that this works only for columns, not for rows. The listed values to achieve the same results would be either of these forms:

```
'CHBAR 3 2 ((.b.1 .c.1) (.b.2 .c.2) (.b.3 .c .3))'

'CHBAR 3 2 ((10 40) (20 50) (30 60))'
```

The method of handling parameters that do not exactly match the specifications varies according to the type of parameters. All mismatches not described below are treated as errors.

Parameters that are too short: The method of handling parameters that are too short varies according to the type of parameters. All mismatches not described below are treated as errors.

Strings
Padded with blanks to the required length. This may be the length specified in the call or, for tokens, the length specified by GDDM.
Too few array elements
Extra members are generated. If the array is a stemmed variable, the REXX variables are searched for, and it is an error if they do not exist. If the array is an enumerated list, extra values are added as necessary. They are zero for floating point and integer parameters, and blanks for strings and tokens.
Too few array dimensions
The item is rescanned to produce the correct number of dimensions. If the array is a stemmed variable, the necessary suffixed names are generated, and it is an error if they do not exist. If the array is an enumerated list, additional values are generated using the following rules:

1. If the size of the missing dimension is explicitly given, the list will be scanned this number of times. If an individual list entry is a value, that value will be reused. For example:

```
/* vector used twice             */
'ASDFMT 2 . (1 2 3 4 5 6)'
```

   becomes

```
'ASDFMT 2 6 ((1 2 3 4 5 6) (1 2 3 4 5 6))'
```

   If it is a variable, it is suffixed:

```
/* extra dimension with suffixes */
'ASDFMT 1 6 (.f .r 3 4 5 6)'
```

   is processed as

```
'ASDFMT 1 6 ((.f1 .r1 3 4 5 6))'
```

2. If the size of the missing dimension is not explicitly given, that dimension is defaulted to 1, and the result is a **1-by-n** array. For example:

```
/* 1-by-6 array generated        */
'ASDFMT . . (1 2 3 4 5 6)'
```

   becomes

```
'ASDFMT 1 6 ((1 2 3 4 5 6))'
```

3. If the list is of two dimensions less than required, the list is scanned the required number of times to produce, for each of the required number of rows, the values in the columns. If the entry is a literal value it is reused. For example:

```
/* rescanning gives 2-by-6 array */
'ASDFMT 2 6 1'
```

   becomes

```
'ASDFMT 2 6 ((1 1 1 1 1 1) (1 1 1 1 1 1))'
```

   Omission of either size parameter defaults to the value 1.

Parameters that are too long: The method of handling parameters that are too long varies according to the type of parameters. All mismatches not described below are treated as errors.

Strings
Truncated to the required length, with an error message.
Too many array elements
Extra members are ignored.
Too many array dimensions
This is treated as an error and a message is given.

Omitting parameters: Parameters can be replaced by dots if they are:

1. Returned values not required by the program.

```
/* omit returned values          */
'ASREAD . . .'
```

2. Lengths that GDDM-REXX can discover from your input.

```
/* omit length you are passing   */
'ASCPUT 1 . "Hello"'
```

3. Array dimensions that GDDM-REXX can discover from your input.

```
/* omit array defining counts    */
'CHBAR  . . ((1 2) (3 4) (5 6))'
```

If two or more parameters depend on an omitted length or count value, the first one from which the value can be determined is used, and subsequent parameters are processed with this “discovered” value.

Finding syntax from reference sources: The parameter syntax of GDDM calls can be found by use of the sample program ERXPROTO or in the GDDM programming reference manuals or summaries.

Interdependent parameters, array dimensions, string lengths: Many GDDM calls have interdependent parameters where earlier lengths and counts describe the lengths of strings and count of elements in arrays. These calls are described in Chapter 3, “The GDDM calls” on page 21

`GSCHAR(x,y,length,string)`

where `length` is the length of the `string`; and

`GSPLNE(count,xarray,yarray)`

where `count` is the number of elements in each of the two arrays.

Where there is one such dependency, the length is the length of the string, or the count is the number of elements in the array.

Where there are two dependencies, strings are given in a one-dimensional array of strings of the given length; numbers are given in a two-dimensional array, with the first count specifying the number of groups and the second specifying the number of elements in each group.

`CHXLAB(count,length,text)`

`text` is an array of `count` strings each of the number of characters in `length`.

`CHBAR(components,count,y-values)`

`y-values` is a two-dimensional array with `components` rows and `count` columns.

This is shown explicitly for GDDM Base, for GDDM-PGF, and for GDDM-GKS in the relevant reference manuals. The same information is available from the ERXPROTO EXEC.

```
/* ERXPROTO forms for the two     */
/* calls described above          */
'CHXLAB cnt1 len2 char.cnt1.len2'
'CHBAR cnt1 cnt2 float.cnt1.cnt2'
```

Parameter syntax in ERXPROTO: ERXPROTO produces output as a string consisting of the callname and the types of parameters in this form:

```
'callname type type .....'
```

Examples of the form that the **type** parameter may take are:

cnt*x*
Counts used as array dimensions. The number *x* is the position of the parameter within the string. For example, **cnt2** is the second parameter in the string.

len*x*
Lengths of character strings. The number *x* is the position of the parameter within the string. For example, **char3** is the third parameter in the string.

float
Floating point parameter.

.float
The dot that precedes it means that this floating point parameter is returned by GDDM.

intg
Integer parameter.

intg.cnt1.cnt2
An integer array of dimensions given by the **cnt1** and **cnt2** parameters – see below for rules to deduce dimensions and sizes.

float.*x*
A floating point array of constant dimension *x*.

char.len*x*
Character string parameter of length given by the **len***x* parameter.

char.*x*
Character string parameter of constant length *x*.

For some examples of output of the ERXPROTO program, see below.

Here is a set of rules that let you produce valid calls from the ERXPROTO syntax.

1. Look for **cnt** values that are array dimensions, and calculate the values you will need for them.

   ```
   /* two sets so two dimensions     */
   /* in example cnt1=2,cnt2=3       */
   'CHBAR cnt1 cnt2 float.cnt1.cnt2'
   /* none .len3 is a length         */
   'GSCHAR float float len3 char.len3'
   'ASREAD .intg .intg .intg'  /* none */
   ```

   For a by-name array you would enter values `array.1.1=10` to `array.2.3=60` (using the values you needed) and use the parameter **.array.** (with a closing dot).

2. Look for any array parameters and work out the correct dimensions. Array parameters are followed by **.cnt*x***, for example **float.cnt1**. There is one dimension for each following **.cnt*x***. That means one set of brackets for each following dot if you are listing the array elements in the call.

   ```
   /* cnt1 and cnt2, two dimensions  */
   /* three (cnt2) elements in each  */
   /* inner parenthesis, two (cnt1)  */
   /* sets of inner parentheses. Two */
   /* levels of nested parentheses,  */
   /* one for each count             */
   'CHBAR  2  3  ((n n n) (n n n))'
   /* no arrays no action            */
   'GSCHAR float float len3 char.len3'
   'ASREAD .intg  .intg  .intg'  /* none */
   ```

3. Look for the character strings and fill in the length values.

   ```
   /* Characters are ABCD so length is four */
   'GSCHAR float float 4  "ABCD"'
   ```

4. Fill in the float or integer values or variable names. These are integers where the parameter says **intg**, any form of number where it says **float**, and character strings where it says **char**.

```
/* put in array values            */
'CHBAR 2  3 ((3 4 5) (5 6 7)) '
/* fill in values                 */
'GSCHAR 10.5  50    4  "ABCD"'
/* fill in all values             */
'ASREAD .type  .val   .count'
```

Parameter syntax in the reference manuals: The reference manuals present GDDM calls using a syntax that has parentheses around the parameters and commas between them, for example:

GSLOAD	**(name,count1,opt_array,seg_count,count2, descriptor)**
APL code	593
GDDM RCP code	X'0C0C1201' (202117633)

The parameters, **name**, **count1**, **opt_array**, and **count2** are all defined as *specified by the user*. Create REXX assignment statements for all these variables and place them before the GDDM call; for example (using abbreviated variable names):

```
/* name of the ADMGDF file to be loaded   */
name='MYGDF'
/* number of elements in opt_array.       */
cnt1=2
/* starting segment number to be assigned */
opta.1=22
/* accommodate to current window size     */
opta.2=2
/* return up to 110 bytes of descriptor   */
cnt2=110
```

The call to a GDDM function can be made in any one of four ways:

1. Omit the parentheses around the parameters and replace the commas with blanks:

   ```
   GSLOAD name cnt1 opta segc cnt2 des
   ```

2. Use a valid REXX variable name for each parameter, optionally using stemmed variable names for array parameters:

   ```
   GSLOAD name cnt1 opta. segc cnt2 des
   ```

3. Place a period '.' before each parameter name. This indicates that the parameter is to be passed "by name":

   ```
   GSLOAD .name .cnt1 .opta. .segc .cnt2 .des
   ```

4. Surround the entire statement with single or double quotes (this is to ensure that the interpreter passes the complete call to GDDM-REXX without attempting substitution):

   ```
   'GSLOAD .name .cnt1 .opta. .segc .cnt2 .des'
   ```

After the GSLOAD call is performed, the REXX variables **segc** and **des** will contain the *returned by GDDM* values.

Dependency between parameters: Rules for deducing the dependency between string and array parameters are as follows:

1. Look for any dependencies between parameters – the parameter names **length** and **count** always show dependencies but they are not the only ones. You must read the parameter descriptions to be sure.
2. When you have found a dependency, check whether the item it describes is numerical or a character string.
3. If there is one dependency, it is the number of elements in a one-dimensional array for numbers, or the length for a character string.
4. If there is more than one dependency:

   For numerical parameters: The number of dependencies specifies the number of dimensions; the first item in the list becomes the number of elements in the first dimension, the second item becomes the number of elements in the second dimension, and so on.
   For string parameters: The number of dependencies is one more than the number of dimensions of the array. The last dependency is the length of each of the strings in the array. Prior dependencies specify the number of elements in each dimension.

Restrictions: The following restrictions apply to the use of GDDM calls in GDDM-REXX:

CHART: There is no CHART call. Use the CS... calls that give an improved programming interface to the Interactive Chart Utility (ICU). (See also the sample ERXCHART EXEC.)

SPINIT: There is no SPINIT call. GDDM-REXX does not support programs that explicitly use the GDDM system programmer interface.

Differences: The following paragraphs describe how GDDM calls from GDDM-REXX are implemented differently from calls from other programming languages.

Array parameters: Array parameters are treated more strictly in GDDM-REXX than they are in other high-level languages – in particular, arrays must be multi-dimensional when they describe lists of lists of values. You should be careful with calls where GDDM-REXX requires an array of strings, for example, GSPLNE. Other languages may accept all values concatenated in one string.

FSINIT: This call is not usually required in GDDM-REXX programs. If you do use it, it initializes a new instance of GDDM within the instance of GDDM-REXX, thereby making the program reentrant.

FSTERM: In GDDM-REXX programs, FSTERM is required only if the program is reentrant. It is used to terminate an instance of GDDM within the instance of GDDM-REXX.

If you use the GXGET AAB to open multiple instances of GDDM under the reentrant interface, FSTERM ends an instance.

Reentrant support: Reentrant support is provided by means of subcommands that extract and set the anchor block, rather than by a separate set of reentrant calls. These subcommands are GXGET AAB and GXSET AAB. They are described in Chapter 6, "GDDM-REXX programming interface" on page 255.

Mapping: The ERXMSVAR EXEC, and two subcommands GXSET MSVARS and GXSET MSADS, are provided so that maps created with GDDM-IMD can be used in REXX EXECs through GDDM mapping calls.

Mapping with GDDM-REXX is further described in "ERXMSVAR EXEC" on page 258.

PA2 as escape to subset: By default in GDDM/VM, PA2 acts as an escape into subset mode. If CMS subset is entered in this way from a GDDM-REXX EXEC, it is not possible to use further EXECs that use GDDM-REXX.

PA2 is set by means of the CMSINTRP procopt. Processing options are discussed in Chapter 19, "Processing options" on page 395.

Note: Under MVS, PA2 causes a reshow.

GDDM-REXX EXECs in subset mode or invoked from other programs: If you plan to use GDDM-REXX EXECs from CMS subset mode, or if your GDDM-REXX EXECs are likely to be invoked from other programs,

the GDDM-REXX command module should be loaded in the nucleus with the following command:

```
NUCXLOAD GDDMREXX
```

When the GDDM-REXX exec has run, you can use the `NUCXDROP` command to free nucleus storage.

Termination on MVS: On MVS, GDDM-REXX is not automatically terminated when a EXEC ends. The programmer must ensure that a GDDMREXX TERM command is always executed before exiting from an EXEC. This includes exits cause by errors and by attention interrupts by the terminal user.

programming interface

Chapter 2. A summary of the calls by function

This chapter provides a summary of the GDDM calls by function.

For an understanding of the background to programming with GDDM, and examples of the use of the GDDM calls, see the *GDDM Base Application Programming Guide*.

For detailed descriptions of the GDDM Base calls and their parameters, see Chapter 3, “The GDDM calls” on page 21.

Types of functions

The GDDM functions are described under the following headings in this chapter:

- **Control functions**: used for starting (initializing) and stopping (terminating) GDDM processing, for controlling device input/output, and for providing some general services.
- **Copy functions**: used for copying (or sending) the contents of the current page, a graphics field, or an image field to a device other than the primary device.
- **Device functions**: used for describing a device to GDDM, and for querying the device characteristics.
- **Graphics functions**: used for creating, displaying and modifying pictures. The pictures are built up from graphics primitives such as lines, arcs, and symbols from previously defined symbol sets. In addition to general graphics functions, two distinct sub-groups of functions can be identified:
  - **Graphics segment functions**: used for defining segments and handling their attributes.
  - **Interactive graphics functions**: used for obtaining input from the operator by means of a logical input device.
- **High-performance alphanumeric functions**: used for defining, modifying, and deleting alphanumeric fields by means of field-lists. They are intended for use by applications that require minimal instruction path length within GDDM.
- **Image functions**: used for controlling the capture, transformation, and display of image data. These include two sub-groups of functions:
  - **Image management functions**
  - **Image presentation functions**
- **Mapped alphanumeric functions**: used for controlling the input and output of data, using maps that have been created with GDDM Interactive Map Definition.
- **Operator window functions**: used for controlling operator windows, which are rectangular subdivisions of a display device screen that have a different **virtual device** appearing in each. Operator windows allow more than one GDDM application to be run on the screen while under the control of a task manager, or one application to run a number of virtual devices sharing the same screen.
- **Page functions**: used for creating and deleting pages. A page is a rectangular area displayed on a device. The page functions determine how much of the page is displayed.
- **Partition functions**: used for creating and controlling real and emulated partitions. Several alternative logical screens, called **partition sets**, can be created on a device, but only one partition set may be shown at one time. Belonging to each partition set are one or more **partitions** that are rectangular subdivisions of the display area.

  Partitions are logical subdivisions of the screen. They can be used by applications for ‘pop up’ windows and similar constructs.
- **Procedural alphanumeric functions**: used for defining, modifying, and deleting alphanumeric fields and their attributes.
- **Symbol set functions**: used for passing symbol sets to and from files, the application program, GDDM storage, and the display device.
- **Utility functions**: used for calling GDDM utility programs from within application programs.

In this chapter, each functional group includes a list of the calls, in alphabetic order, that are available for each group. Each list comprises the call name, and a brief description of that call. The calls used to *specify* or *define* parameter values are listed first, followed by the calls used to *query* parameter values.

The syntax of the calls is described in Chapter 3, “The GDDM calls” on page 21.

Notes:

1. Groups of related calls may usually be identified by the first two or three characters in the call name.
2. Obsolete calls are omitted from the summary table and detailed description. A few calls appear more than once under different headings.

Control functions

Initialization and termination

FSINIT (Or one of its aliases) initializes GDDM processing.

FSRNIT Terminates and reinitializes GDDM processing.

FSTERM Terminates GDDM processing and releases resources (such as storage).

SPINIT Initializes GDDM using the system programmer interface. The system programmer interface is described in detail in Chapter 22, "Special-purpose programming in GDDM" on page 431.

Specifying device input and output

ASREAD Makes all outstanding changes to the device and waits for new keyboard input.

| DSFRCE Outputs the current page as a page segment or overlay as a member of a partitioned data set.

FSCHEK Checks the complexity of a picture to find out whether the complexity will cause PS overflow at the next FSFRCE, ASREAD, GSREAD, or MSREAD call.

FSENAB Enables or disables input to be entered into the primary device.

FSFRCE Updates the device to include all changes that have occurred since the last transmission to the device.

| FSGETS Initiates retrieval of family-4 output by the application program.

| FSGET Retrieves family-4 output for the application program

| FSGETE Ends the retrieval of family-4 output by the application program.

FSREST Causes retransmission of all device data on the next transmission to the device.

FSSAVE Saves a device-dependent form of the contents of the current page on auxiliary storage for later retrieval and display by FSSHOR or FSSHOW.

FSSHOR Gets a specified picture from storage, displays it and returns the identity of the key used to terminate the display.

FSSHOW Gets a specified picture from storage and displays it.

FSUPDM Controls the way in which graphics data is updated on a particular device.

GSREAD Returns the next graphics event. If necessary, it performs all output outstanding, and waits for input from a graphics logical input device.

MSREAD Displays a map definition created by GDDM-IMD, waits for an interrupt from the terminal, and returns data from the map.

Querying device input and output

FSQUPD Queries the way in which graphics data is updated on a particular device.

Error handling

FSEXIT Specifies an application program routine to receive control at the end of a call whose error return code equals or exceeds a prescribed value.

FSQERR Returns information about the last GDDM call whose error severity return code was nonzero.

FSTRCE Controls internal trace functions. It is intended for diagnosing possible internal errors in GDDM.

Environment control functions

ESACRT Creates an application group.

ESADEL Deletes an application group.

ESAQRY Queries an application group.

ESASEL Selects an application group.

ESEUDS Specifies encoded or source-format user default specifications that can be used to change GDDM-provided defaults information for the application program.

ESLIB Identifies the subsystem libraries to be used to access or store various types of GDDM data, such as symbol sets.

ESPCB Identifies the program communication blocks that can be used by GDDM when the application program is running under IMS/VS.

ESQCPG Queries the code page of a GDDM object.

ESQEUD Queries the encoded user default specification.

ESQOBJ Queries the existence of a GDDM object on auxiliary storage.

ESQUNL Queries the length of nickname information for the specified family.

ESQUNS Queries the nickname information for the specified family.

ESSCPG Sets the code page of a GDDM object.

ESSUDS Specifies a single encoded or source-format user default specification that can be used to change GDDM-provided defaults information for the application program.

Querying the GDDM environment

FSQSYS Returns information on the release of GDDM that has been link-edited with the application program, the release of GDDM that has been loaded dynamically, the current GDDM subsystem environment (indicating CICS, IMS, MVS, TSO, or CMS), and (in some cases) the subsystem qualifier.

FSQURY Returns information about the primary device, including its partition characteristics.

Other control functions

FSALRM Sounds the terminal alarm on the next transmission to the device for the page selected when FSALRM was called.

FSTRAN Performs code-page conversions on strings in the user's storage.

Copy functions

DSCOPY Sends a transformed picture to the alternate device.
FSCOPY Sends the page to the alternate device.
FSLOG Sends a character string to the alternate device.
FSLOGC Sends a character string with carriage-control character to the alternate device.
GSARCC Controls whether the aspect ratio is to be preserved when a picture is copied to another device.
GSCOPY Sends graphics to the alternate device.

Device functions

Specifying device characteristics

DSCLS Closes a device so that it can no longer be used.
DSCMF Sets the User Control function.
DSDROP Stops a device being either primary or alternate.
DSOPEN Opens a device and makes its details known to GDDM.
DSRNIT Closes and reinitializes a device.
DSUSE Specifies a device as either primary or alternate.

Querying device characteristics

DSQCMF Queries the User Control function.
DSQDEV Returns the characteristics of the device. The returned data reflects that specified on DSOPEN together with additional information concerning the device properties.
DSQUID Returns an unused identifier that may be used on DSOPEN.
DSQUSE Returns the identifier of the primary or alternate device.
FSQURY Returns information on the device, partition, graphics, image, image cursor, and scanner characteristics of the primary device.

Graphics functions

Defining the elements in the hierarchy of graphics objects

GSBND Defines a data boundary.
GSCLP Enables or disables clipping.
GSCLR Deletes all graphics from the graphics field.
GSFLD Defines the graphics field. The default is the entire page.
GSPS Defines the picture space. The default is the entire graphics field.
GSUWIN Defines a uniform graphics window such that the aspect ratio of the current viewport is preserved.
GSVIEW Defines the current viewport. The default is the entire picture space.
GSWIN Defines the current graphics window. The definition is used for all later graphics operations until it is redefined.

Querying the elements in the hierarchy of graphics objects

GSQBND Returns the current data boundary definition.
GSQCEL Returns the default graphics cell size in current window units.
GSQCLP Returns the current clipping state.
GSQCUR Returns the cursor position within the current window.
GSQFLD Returns the position and size of the graphics field.
GSQPS Returns the current picture space definition.
GSQVIE Returns the current viewport definition.
GSQWIN Returns the current window definition.

Specifying attributes of graphics primitives

GSAM Sets the attribute mode. If attributes are preserved, they may be restored by a call to GSPOP.
GSBMIX Controls the way that the background color of a primitive is combined with the color of any primitive that it overlaps (background color-mix mode).
GSCA Specifies a direction angle for the baseline of character strings. Ignored for mode-1 characters.
GSCB Specifies the character-box size. Ignored for mode 1; controls spacing for mode 2; controls character size and spacing for mode 3.
GSCBS Specifies the character-box spacing.
GSCD Specifies the direction at which characters in a string are to be drawn, relative to the baseline direction specified by GSCA.
GSCH Specifies the current character shear angle. Ignored for mode 1. Only affects positioning for mode 2.
GSCM Specifies the character mode (mode-1, -2, or -3) to be used for graphics text.
GSCOL Specifies the current color.
GSCS Specifies the identifier of the symbol set to be used for graphics text.
GSDEFE Ends the definition of drawing defaults.
GSDEFS Starts the definition of drawing defaults.
GSFLW Sets the current line width as a fractional value.
GSLT Sets the current line type.
GSLW Sets the current line width.
GSMB Sets the size of the marker box.
GSMIX Controls the way that the color of a primitive is combined with any underlying color (foreground color-mix mode).
GSMS Specifies the current marker symbol to be used.
GSPAT Specifies the current shading pattern.
GSPOP Restores attributes to their previous values if preserved by a prior call to GSAM.
GSSCT Specifies the current transform for scaling, shear, rotation, and displacement of graphics primitives.
GSSEN Sets the mixed string attribute to enable double-byte (DBCS) and single-byte (SBCS) character sets to be mixed in the same string in graphics text.
GSSVL Specifies the segment viewing limits.

GSTA Specifies the graphics text alignment. This determines the positioning of the character box relative to the character baseline and start point.

Querying graphics primitives attributes and characteristics

GSQAM Returns the current attribute mode.
GSQBMX Returns the current background color-mixing mode.
GSQCA Returns the direction angle for the baseline of character strings.
GSQCB Returns the character-box size.
GSQCBS Returns the character-box spacing.
GSQCD Returns the direction at which characters in a string are drawn, relative to the baseline.
GSQCH Returns the current character shear angle.
GSQCM Returns the current character mode.
GSQCOL Returns the current color.
GSQCP Returns the current position.
GSQCS Returns the identifier of the symbol set used for graphics text.
GSQFLW Returns the current fractional line width.
GSQLT Returns the current line type.
GSQLW Returns the current line width.
GSQMB Returns the size of the marker box.
GSQMIX Returns the foreground color-mix mode.
GSQMS Returns the current marker symbol.
GSQPAT Returns the current shading pattern.
GSQSEN Returns the graphics text mixed string attribute.
GSQSVL Returns the segment viewing limits.
GSQTA Returns the current graphics text alignment.
GSQTB Queries details of the graphics text box occupied by a specified character string.

Drawing graphics primitives

GSARC Draws a circular arc about a specified point, starting at the current position and subtending a specified angle.
GSAREA Starts the construction of a shaded area defined by subsequent line-drawing calls.
GSCHAP Draws a specified character string starting at the current position.
GSCHAR Draws a specified character string starting at a specified point. It is equivalent to a call to GSMOVE followed by a call to GSCHAP.
GSCP Sets the current position to the specified point.
GSELPS Draws an elliptical arc from the current position to a specified point.
GSENDA Ends the construction of a shaded area.
GSLINE Draws a straight line from the current position to a specified point.
GSMARK Draws a single marker symbol at a specified point.
GSMRKS Draws a series of marker symbols at specified points.
GSMOVE Moves the current position to a specified point, without drawing.
GSPFLT Draws a curved fillet defined by lines joining several points.
GSPLNE Draws a sequence of lines, starting at the current position and passing through a specified set of points.
GSVECM Combines moves and lines in any order, for a specified set of points. It is equivalent to a series of GSMOVE and GSLINE calls.

Drawing images

GSIMG Draws an image at the current position. An image is a dot pattern stored as ones and zeros.
GSIMGS Draws a scaled image.

Note: Do not confuse these calls with the image calls (see page 17), whose names begin with the letter "I."

Saving and loading the picture

CGSAVE Saves segments or all the graphics data in the current GDDM page, into a Computer Graphics Metafile (CGM) on auxiliary storage.
CGLOAD Retrieves a copy of a Computer Graphics Metafile (CGM) from auxiliary storage and loads it into the graphics field on the current GDDM page.
GSGET Gets one record of graphics data from the current page.
GSGETE Ends the retrieval of graphics data.
GSGETS Starts the retrieval of graphics data.
GSLOAD Retrieves a copy of a graphics data format (GDF) object from the segment library on auxiliary storage and loads it into the current GDDM page.
GSPUT Puts graphics data into the current graphics viewport.
GSSAVE Saves segments or all the graphics data from the current GDDM page and puts them into the segment library, as a GDF object, on auxiliary storage.

Graphics segment functions

Defining graphics segments and handling attributes

GSCALL Calls a segment from within another segment.
GSCLR Deletes all graphics from the graphics field.
GSSAGA Sets or modifies the transform for scaling, shear, rotation, and displacement of the specified segment.
GSSATI Sets the initial segment attributes that are used when subsequent segments are created.
GSSATS Modifies the attributes of the specified segment.
GSSCLS Closes the current segment. No more primitives may be added to it.
GSSCPY Copies the specified segment into the current stream of primitives.
GSSCT Specifies the current transform for scaling, shear, rotation, and displacement of graphics primitives.
GSSDEL Deletes a specified segment.
GSSEG Creates a segment.
GSSINC Copies the primitives of the specified segment into the current stream of primitives.
GSSORG Sets the position of the origin of the specified segment.

GSSPOS Sets the position of the specified segment.
GSSPRI Changes the order of priority by which segments are drawn and detected.
GSSTFM Sets or modifies the transformation matrix of a specified segment.
GSSVL Specifies the segment viewing limits.

Querying graphics segment information

GSQAGA Returns the transform (geometric attributes) of the specified segment.
GSQATI Returns the initial attributes that are assigned to segments when they are created.
GSQATS Returns the value of the specified attribute in the specified segment.
GSQMAX Returns the number of currently defined segments and the highest segment identification number.
GSQORG Returns the position of the segment origin of the specified segment.
GSQPOS Returns the position of the specified segment.
GSQPRI Returns the identity of the segment next in priority to the specified segment.
GSQTFM Queries the transformation matrix of the specified segment.

Interactive graphics functions

Picking tagged primitives

GSTAG Sets a tag value for following primitives.

Initializing logical devices

GSENAB Enables and disables a logical input device.
GSILOC Initializes a locator input device and defines the echo that is seen on the screen.
GSIPIK Initializes a pick input device and defines the echo that is seen on the screen.
GSISTK Initializes a stroke input device and defines the echo that is seen on the screen.
GSISTR Initializes a string input device and defines the echo that is seen on the screen.
GSIDVF Sets an initial floating-point data value. For a locator device, this value is a coordinate for the initial positioning of the locator echo. For a pick device, this value is the pick aperture.
GSIDVI Sets an initial integer data value. For a locator device, this value is the segment identifier to used for the locator echo. For a string device, this value is the initial cursor position.

Input and output functions

GSREAD Updates the graphics screen, waits for, and returns graphics input.
GSFLSH Clears all items from the graphics input queue.

Interactive graphics query functions

GSCORS Queries (correlates) segment and tag information for the structure of each tagged primitive in the current graphics field that intersects a defined aperture.
GSQCHO Returns choice data from the current input record presented by the GSREAD call.
GSQLID Returns information about a specified logical input device.
GSQLOC Returns locator data from the current input record presented by the GSREAD call.
GSQPIK Returns pick data from the current input record presented by the GSREAD call.
GSQPKS Returns the pick structure data from the current input record presented by the GSREAD call.
GSQSIM Returns whether the next record to be returned by the GSREAD call from the graphics input queue was produced by the same terminal operator action as the current record.
GSQSTK Returns stroke data from the current input record.
GSQSTR Returns string data from the current input record.
GSQTAG Returns the current tag value as set by the most recent GSTAG call.

High-performance alphanumeric functions

APDEF Defines a field list.
APDEL Deletes a field list.
APMOD Modifies a field list.
APQIDS Returns the identifiers of field lists.
APQNUM Returns the number of field lists.
APQRY Returns a field list.
APQSIZ Returns the size of a field list.
APQUID Returns a unique field-list identifier.

Image functions

Image management

Specifying image definition

IMACLR Clears a rectangle within an image.
IMACRT Creates an image.
IMADEL Deletes a specified image.
IMAGID Requests and reserves a unique image identifier.
IMARES Sets the resolution attributes of an image.
IMARF Sets the resolution of an image to defined or undefined.
IMATRM Trims an image down to the size of a specified rectangle.

Querying image definition

IMAQRY Returns the attributes of an image.

Transfer operation

IMAGT Retrieves data from an image.
IMAGTE Ends the retrieval of data from an image.
IMAGTS Starts the retrieval of data from an image.
IMAPT Enters data into an image.
IMAPTE Ends data entry into an image.
IMAPTS Starts data entry into an image.
IMARST Restores a completed image from auxiliary storage.

IMASAV Saves an image on auxiliary storage.
IMXFER Transfers data between two images, applying a projection.

Projection management

IMPCRT Creates an empty projection.
IMPDEL Deletes a projection.
IMPGID Requests a projection identifier.
IMPRST Restores a projection from auxiliary storage.
IMPSAV Saves a projection on auxiliary storage.

Transform element

IMREX Defines a rectangular sub-image from a source image – pixel coordinates.
IMREXR Defines a rectangular sub-image from a source image – real coordinates.
IMRNEG Negates the pixels of an extracted image.
IMRORN Rotates an extracted image.
IMRPL Places a transformed image into a target image – pixel coordinates.
IMRPLR Places a transformed image into a target image – real coordinates.
IMRRAL Sets the current resolution/scaling algorithm.
IMRREF Reflects an extracted image.
IMRSCL Scales an extracted image.

Scanner-related transform element

IMRBRI Defines the brightness of a gray-scale image.
IMRCON Defines the contrast of a gray-scale image.
IMRCVB Converts a gray-scale image to binary.

Image presentation

Specifying image field and control

ISCTL Sets the image control parameters.
ISFLD Defines the image field.
ISXCTL Sets the image control parameters (extended).

Querying image field

ISQFLD Queries the image field.

Device input

ISENAB Enables or disables an image cursor.
ISESCA Controls the echoing of a scanner image.
ISIBOX Initializes an image box cursor.
ISILOC Initializes an image locator cursor.
ISLDE Loads paper into a scanner.

Query device characteristics

ISQBOX Queries the image box cursor.
ISQCOM Returns the image compressions supported by the device.
ISQFOR Returns the image formats supported by the device.
ISQLOC Returns the image locator cursor.
ISQRES Returns the image resolutions supported by the device.
ISQSCA Queries the image scanner device.

Mapped alphanumeric functions

High-level function

MSREAD Displays a map definition created by GDDM-IMD, waits for an interrupt from the terminal, and returns data from the map.

Create a page for mapping

MSPCRT Creates a page with specified identification, size, and associated mapgroup.

Creating and manipulating the data associated with a mapped field

MSDFLD Creates a mapped field with specified identification, position on the page, and associated map.
MSPUT Updates the data in the specified map by passing the ADS for the map.
MSCPOS Positions the cursor in a field contained within the map at the next MSPUT call.
MSGET Retrieves data from the specified map by returning the ADS for the map.

Querying mapping functions

MSPQRY Returns the size of the specified page and the mapgroup associated with it.
MSQADS Returns a descriptor of the fields that make up the ADS for a specified map.
MSQFIT Returns the number of times that the specified map fits into the floating area of the mapgroup associated with the page.
MSQFLD Returns the size and position of the specified mapped field.
MSQGRP Returns the size of the page that would be created by the specified mapgroup and the size and position of the floating area.
MSQMAP Returns the size and position of the specified map and the size of the associated ADS.
MSQMOD Returns the identities and ADS lengths of mapped fields that have been changed after an ASREAD or GSREAD call.
MSQPOS Returns the position of the cursor in a map.

Operator window functions

Window processing

WSCRT Creates an operator window.
WSDEL Deletes an operator window.
WSIO Performs windowed device input/output.
WSMOD Modifies the attributes of the current operator window.
WSSEL Selects an operator window as the current operator window.
WSSWP Sets or resets the operator window viewing priorities.

Querying window characteristics

WSQRY Returns the identity and attributes of the current operator window.
WSQUN Returns an unused operator window identifier.
WSQWI Returns the operator window identifiers.
WSQWN Returns the number of operator windows.
WSQWP Returns the operator window viewing priorities.

Page functions

Page manipulation

FSPCLR Clears **all** objects and fields from the current page.
FSPCRT Creates a new page, with specified identifier and size. The page is empty.
FSPDEL Deletes a specified page.
FSPSEL Selects a specified page, which becomes the current one, for display or updating.
FSPWIN Sets the origin and size of a page window, or alters the origin of a page window.

Querying page attributes

FSPQRY Returns the size of a specified page.
FSQCPG Returns the identifier of the current page.
FSQUPG Returns an unused page identifier.
FSQWIN Returns origin and size of the current page window.

Partition functions

Partition sets

PTSCRT Creates a partition set with specified grid and selection of real or emulated partitions.
PTSDEL Deletes the specified partition set and any partitions within it.
PTSSEL Selects a partition set as the current partition set.

Partitions

PTNCRT Creates a partition with specified size and position.
PTNDEL Deletes the specified partition.
PTNMOD Modifies the attributes of the specified partition.
PTNSEL Selects the specified partition as the current partition.
PTSSPP Sets the partition viewing priorities.

Querying partition and partition sets

PTNQRY Returns the attributes of the current partition.
PTNQUN Returns an unused partition identifier.
PTSQPI Returns the partition identifiers.
PTSQPN Returns the number of partitions.
PTSQPP Returns the partition viewing priorities.
PTSQRY Returns the attributes of the current partition set.
PTSQUN Returns an unused partition-set identifier.

Procedural alphanumeric functions

Alphanumeric fields

ASDFLD Defines a field with specified identifier, position, size, and type. Any existing field with the same identifier is deleted.
ASDFMT Defines a group of alphanumeric fields for the current page; all existing fields are deleted.
ASFCLR Clears all unprotected fields, all protected fields, or both, and resets their character attributes to defaults.
ASRFMT Defines a group of alphanumeric fields for the current page; existing fields that are not redefined are not deleted.

Specifying or changing field attributes

ASDFLT Sets the default attributes to be used for new field definitions on the current page.
ASFBDY Defines the outlining (or boundary) for a field.
ASFCOL Sets the field color.
ASFEND Specifies the action to be taken after terminal input to of each row of a field.
ASFHLT Sets the field highlighting.
ASFIN Specifies the type of null-to-blank conversion to occur on input to a field.
ASFINT Sets the field intensity.
ASFMOD Defines an alphanumeric field (or fields) as modified or unmodified.
ASFPSS Sets the symbol set to be used as the primary symbol set for a field.
ASFOUT Specifies the type of blank-to-null conversion to occur on output from the field to the device.
ASFSEN Sets the field mixed-string attribute.
ASFTRA Sets the field transparency attribute.
ASFTRN Specifies the set of symbol code translation tables to be used for a field.
ASFTYP Sets the field type.
ASRATT Sets field attributes for designated existing fields.

Querying field attributes

ASQFLD Returns the attributes of one or more fields.
ASQMAX Returns the number of alphanumeric fields on the current page, and the maximum field identifier currently defined.
ASQMOD Returns the identifiers and lengths of modified fields for the current page.
ASQNMF Returns the number of modified fields for the current page.

Specifying field contents and character attributes

ASCCOL Sets the colors to be used for individual character positions in a field.
ASCHLT Sets the highlighting for each character position in a field.
ASCPUT Fills a specified field with a character string.
ASCSS Sets the symbol-set identifiers to be used for individual characters in a field.

ASMODE Determines whether the terminal operator may modify character attributes.

Querying field contents and character attributes

ASCGET Returns the character contents of a specified field.
ASQCOL Returns the current character color attributes of a field.
ASQHLT Returns the current highlighting for characters in a field.
ASQLEN Returns the real and effective lengths of the contents of a field.
ASQSS Queries the current symbol-set attributes for characters in a field.

Cursor operations

ASFCUR Positions the cursor within the current page or a specified field.
ASQCUR Queries the current cursor position. This is returned either in page coordinates or as a position within a field.

Defining input/output translation tables

ASDTRN Defines or redefines a set of I/O translation tables for the current page.

Symbol set functions

Loading symbol sets

GSCPG Specifies a 4250 code page as the current one.
GSDSS Loads a set of symbol set definitions from data passed by the application program.
GSLSS Loads a symbol set from auxiliary storage.
PSDSS Loads a symbol set into a device programmed symbol (PS) store from data passed by the application program.
PSLSS Loads a symbol set into a device PS store from auxiliary storage.
PSLSSC Loads a symbol set into a device PS store from auxiliary storage, the PS store does not already contain a set with the specified identifier.

Releasing symbol sets

GSRSS Releases a symbol set from GDDM storage.
PSRSS Releases a symbol set from a device PS store.

Reserving or releasing a PS store

PSRSV Reserves or releases a PS store for explicit control and use by an application program.

Reading or writing a symbol set from or to auxiliary storage

SSREAD Reads a symbol set from auxiliary storage and returns it to the application program.
SSWRT Writes a symbol set from application-program storage to auxiliary storage.

Querying symbol sets

GSQCPG Returns either the name of the current code page or the name of the code page associated with the specified symbol set identifier.
GSQCS Returns the current symbol set.
GSQNSS Returns the number of graphics symbol sets currently loaded.
GSQSS Returns information about all currently loaded graphics symbol sets.
GSQSSD Returns information about a specific symbol set that has been loaded by GSDSS or GSLSS.
PSQSS Returns information about the status of device PS stores.
SSQF Returns information about a specified symbol set on auxiliary storage.

Utility call functions

CDPU Calls the Composite Document Print Utility from a user application program.
ISSE Calls the Image Symbol Editor from a user application program.

Chapter 3. The GDDM calls

This chapter contains descriptions of all GDDM Base calls and their parameters, in alphabetic order of call name.

Format of the GDDM call descriptions

- The call mnemonic, of up to six characters, shown as a heading.
- A brief description of the call function.
- The call syntax, with parameters if there are any. For more information, see the text in "Syntax of GDDM calls."
- The APL code, defined in decimal.
- The RCP code, defined in hexadecimal and decimal.
- A description of each parameter. Each parameter is defined as being specified by the user or returned by GDDM. Parameters that must be assigned a value by the user when a call is made are indicated by the phrase *(specified by user)*. Parameters that receive information from GDDM are indicated by the phrase *(returned by GDDM)*.

  The data type of each parameter is defined.

  Extra information is provided where necessary.
- A more detailed description of the call with references, for example, to such items as programming techniques, related GDDM calls, and other documentation.
- A list of the principal errors associated with the call. For more information, see "Error messages in GDDM calls."

Syntax of GDDM calls

The syntax shown for the call statements is for the nonreentrant interface using PL/I or FORTRAN, omitting the CALL verb, and in the case of PL/I, the line-terminating semicolon (;).

The conventions used in presenting the syntax for calls are:

- The call name, one through six characters, is shown in uppercase.
- The parameter list, if present, is delimited by parentheses, and the parameter separator, if there is more than one parameter, is a comma(,).
- Uppercase words, parentheses, and commas must be coded exactly as shown.
- Lowercase words should be replaced by arguments appropriate to the programming language being used.
- The parameters of the call statements are shown separated by spaces; these are for purposes of clarity, and are not required during coding.

Error messages in GDDM calls

All functions produce an error message if the parameter count is wrong. Also, all functions other than FSINIT and SPINIT generate an error message if GDDM has not been initialized. To avoid repetition, these error messages are listed below, and are not included in the call statement descriptions.

```
ADM0001 U  GDDM STORAGE ANCHOR IS INVALID OR HAS NOT
           BEEN INITIALIZED
ADM0003 E  INCORRECT NUMBER OF ARGUMENTS (=0) ON
           REENTRANT GDDM CALL
ADM0004 E  INCORRECT NUMBER OF ARGUMENTS (=0) ON SPI
           GDDM CALL
ADM0005 E  INCORRECT NUMBER OF ARGUMENTS (=1) ON SPI
           GDDM CALL
ADM0006 E  INCORRECT NUMBER OF ARGUMENTS (=2) ON SPINIT
           GDDM CALL
```

Functions that involve device input/output may produce a range of error messages, some of them subsystem-dependent. These error messages are also not included in the call statement descriptions.

All error messages are explained in detail in the *GDDM Messages* manual.

The remainder of this chapter lists and describes the GDDM Base calls.

Alphabetic list of GDDM calls

APDEF

Function

To define a field list.

APDEF	**(field-list-id, depth-1, width-1, field-list, length, data-buffer, depth-2, width-2, bundle-list, mode)**
APL code	280
GDDM RCP code	X'0C380000' (204996608)

Parameters

field-list-id *(specified by user) (fullword integer)*
The identifier of the new field list. It must be greater than 0, and unique for the current page.

depth-1 *(specified by user) (fullword integer)*
The number of rows in the new field list array (stored in row major order). It must be greater than or equal to 1.

width-1 *(specified by user) (fullword integer)*
The number of columns in the new field list array. It must be greater than or equal to 6.
field-list *(specified by user) (an array of halfword integers)*
The new field list.
length *(specified by user) (fullword integer)*
The length of the data buffer. It must be greater than or equal to 0.
data-buffer *(specified by user) (character)*
The data buffer to be associated with the field list.
depth-2 *(specified by user) (fullword integer)*
The number of rows in the new bundle list array (stored in row major order). It must be greater than or equal to 0.
width-2 *(specified by user) (fullword integer)*
The number of columns in the new bundle list array. It must be greater than or equal to 4.
bundle-list *(specified by user) (an array of halfword integers)*
The bundle list to be associated with the field list.
mode *(specified by user) (fullword integer)*
The mode of operation. This consists of a set of indicators, which have these values:

1 Validate
There is to be validation of the parameters, field list, data buffer, and bundle list.

2 Locate
The field list, data buffer, and bundle list are not copied by GDDM. The storage they occupy must not be released until the field list has been deleted, and it must not be altered except according to the rules described in Chapter 17, “GDDM high-performance alphanumerics” on page 369. If this indicator is not set, move mode is implied.

Note: Locate mode must not be used with interpreted programming languages such as REXX.

4 Cursor
The cursor row and cursor column fields in the field list header are to contain the alphanumeric cursor position. Only one field list per page may be used for this purpose at any one time.

If more than one indicator is required, the mode should be set to the sum of the numbers corresponding to the indicators required.

Description

Defines a new field list to GDDM. The field list describes a set of alphanumeric fields on the current page. The new fields must not overlap each other, or any existing fields in other field lists on the same page.

Field lists may not be defined on mapped pages or pages containing procedural alphanumerics. Similarly, procedural or mapped alphanumeric fields may not be defined on pages containing field lists.

The format of field list, data buffer, and bundle list are described in Chapter 17, “GDDM high-performance alphanumerics” on page 369.

Note: When used under non-XA subsystems of CICS, the total storage occupied by the field list, data buffer, and bundle list must not be greater than 64KB.

Principal errors

```
ADM0222 E  MODE n IS INVALID
ADM3000 E  CURRENT PAGE IS MAPPED OR HAS PROCEDURAL
           ALPHANUMERICS
ADM3001 E  FIELD LIST n ALREADY EXISTS
ADM3002 E  FIELD LIST IDENTIFIER n IS INVALID
ADM3003 E  LENGTH (n) IS INVALID
ADM3006 E  CURSOR POSITION ALREADY DEFINED IN ANOTHER
           FIELD LIST
ADM3009 E  DEPTH (n1) OR WIDTH (n2) IS INVALID
ADM3013 E  FIELD LIST n TOTAL STORAGE EXCEEDS SUBSYSTEM
           MAXIMUM
```

APDEL

Function

To delete a field list.

APDEL	**(field-list-id)**
APL code	281
GDDM RCP code	X'0C380100' (204996864)

Parameters

field-list-id *(specified by user) (fullword integer)*
The identifier of the field list to be deleted.

Description

Deletes a field list.

Principal errors

```
ADM3002 E  FIELD LIST IDENTIFIER n IS INVALID
ADM3008 E  FIELD LIST n DOES NOT EXIST
```

APMOD

Function

To modify a field list.

APMOD	(field-list-id, depth-1, width-1, field-list, length, data-buffer, depth-2, width-2, bundle-list, mode)
APL code	282
GDDM RCP code	X'0C380200' (204997120)

Parameters

field-list-id *(specified by user) (fullword integer)*
The identifier of the field list to be modified.
depth-1 *(specified by user) (fullword integer)*
The number of rows in the modified field list array (stored in row major order). If zero is specified, the field list is not modified.
width-1 *(specified by user) (fullword integer)*
The number of columns in the modified field list array. It must be greater than or equal to 6.
field-list *(specified by user) (an array of halfword integers)*
The modified field list.
length *(specified by user) (fullword integer)*
The length of the modified data buffer. If zero is specified, the data buffer is not modified.
data-buffer *(specified by user) (character)*
The modified data buffer.
depth-2 *(specified by user) (fullword integer)*
The number of rows in the modified bundle list array (stored in row major order). If zero is specified, the bundle list is not modified.
width-2 *(specified by user) (fullword integer)*
The number of columns in the modified bundle list array. It must be greater than or equal to 4.
bundle-list *(specified by user) (an array of halfword integers)*
The modified bundle list.
mode *(specified by user) (fullword integer)*
The mode of operation as specified by the APDEF call. If –1 is specified, the mode is unchanged.

Description

Modifies the specified field list. With this call it is possible to change any of:

- The size and contents of the field list, and also its location if operating in locate mode.
- The size and contents of the data buffer, and also its location if operating in locate mode.
- The size and contents of the bundle list, and also its location if operating in locate mode.
- The mode of operation.

Modifications to field list, data buffer, and bundle list must conform to the rules defined in the *GDDM Base Application Programming Guide*.

Note: When used under non-XA subsystems of CICS, the total storage occupied by the field list, data buffer, and bundle list must not be greater than 64KB.

Principal errors

```
ADM0222 E  MODE n IS INVALID
ADM3002 E  FIELD LIST IDENTIFIER n IS INVALID
ADM3003 E  LENGTH (n) IS INVALID
ADM3006 E  CURSOR POSITION ALREADY DEFINED IN ANOTHER
           FIELD LIST
ADM3007 E  CANNOT SWITCH BETWEEN LOCATE MODE AND MOVE
           MODE
ADM3008 E  FIELD LIST n DOES NOT EXIST
ADM3009 E  DEPTH (n1) OR WIDTH (n2) IS INVALID
ADM3013 E  FIELD LIST n TOTAL STORAGE EXCEEDS SUBSYSTEM
           MAXIMUM
```

APQIDS

Function

To query field list identifiers.

APQIDS	(type, no-of-elements, array)
APL code	283
GDDM RCP code	X'0C380300' (204997376)

Parameters

type *(specified by user) (fullword integer)*
The category of field list. The categories are:
1 All field lists.
The identifiers of all the field lists on the current page are returned in order of creation.
2 All modified field lists.
The identifiers of all the field lists on the current page with input flags set are returned in order of creation.
no-of-elements *(specified by user) (fullword integer)*
The number of field list identifiers to be queried. This is the number of elements in **array**.
array *(returned by GDDM) (an array of fullword integers)*
An array of field list identifiers. If there are more elements in **array** than field lists in the specified category, the remaining elements are set to –1.

Description

Returns the identifiers of the field lists that fall into the category defined by the **type** parameter.

Principal errors

```
ADM3117 E  TYPE (n) IS INVALID
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
```

APQNUM

Function

To query field list numbers.

APQNUM	(element-number, number-of-elements, array)
APL code	284
GDDM RCP code	X'0C380400' (204997632)

Parameters

element-number *(specified by user) (fullword integer)*
The number of the first element in **array**. It must be in the range 1 through 2.

number-of-elements *(specified by user) (fullword integer)*
The number of numbers to be returned. It is also the number of elements in **array**. It must be in the range 0 through 2.

array *(returned by GDDM) (an array of fullword integers)*
An array of numbers of field lists, by category. The array elements are:

1. The number of field lists on the current page.
2. The number of field lists on the current page with input flags set.

Description

Returns the number of field lists as an array, by category, on the current page.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
```

APQRY

Function

To query a field list.

APQRY	(field-list-id, depth-1, width-1, field-list, length, data-buffer, depth-2, width-2, bundle-list, mode)
APL code	285
GDDM RCP code	X'0C380500' (204997888)

Parameters

field-list-id *(specified by user) (fullword integer)*
The identifier of the field list to be queried.

depth-1 *(specified by user) (fullword integer)*
The number of rows in the field list array (stored in row major order). It must be either zero or the same as the current depth of the field list, which is the depth specified on the most recent call to APDEF or APMOD that affected the field list. If zero is specified, the field list is not returned.

width-1 *(specified by user) (fullword integer)*
The number of columns in the field list array. It must be either zero or the same as the current width of the field list, which is the width specified on the most recent call to APDEF or APMOD that affected the field list. If zero is specified, the field list is not returned.

field-list *(returned by GDDM) (an array of halfword integers)*
The queried field list.

length *(specified by user) (fullword integer)*
The length of the data buffer. It must be either zero or the same as the current length of the data buffer, which is the length specified on the most recent call to APDEF or APMOD that affected the data buffer. If zero is specified, the data buffer is not returned.

data-buffer *(returned by GDDM) (character)*
The data buffer associated with the field list.

depth-2 *(specified by user) (fullword integer)*
The number of rows in the bundle list array (stored in row major order). It must be either zero or the same as the current depth of the bundle list, which is the depth specified on the most recent call to APDEF or APMOD that affected the bundle list. If zero is specified, the bundle list is not returned.

width-2 *(specified by user) (fullword integer)*
The number of columns in the bundle list array. It must be either zero or the same as the current width of the bundle list, which is the width specified on the most recent call to APDEF or APMOD which affected the bundle list. If zero is specified, the bundle list is not returned.

bundle-list *(returned by GDDM) (an array of halfword integers)*
The queried bundle list.

mode *(returned by GDDM) (fullword integer)*
The mode of operation as specified by the APDEF call.

Description

Returns the contents of the specified field list, data buffer, bundle list, and the mode of operation.

Principal errors

```
ADM3002 E  FIELD LIST IDENTIFIER n IS INVALID
ADM3003 E  LENGTH (n) IS INVALID
ADM3008 E  FIELD LIST n DOES NOT EXIST
ADM3009 E  DEPTH (n1) OR WIDTH (n2) IS INVALID
```

APQSIZ

Function

To query the size of a field list

APQSIZ	(field-list-id, depth-1, width-1, length, depth-2, width-2)
APL code	286
GDDM RCP code	X'0C380600' (204998144)

Parameters

field-list-id *(specified by user) (fullword integer)*
The identifier of the field list to be queried.
depth-1 *(returned by GDDM) (fullword integer)*
The number of rows in the field list array (stored in row major order).
width-1 *(returned by GDDM) (fullword integer)*
The number of columns in the field list array.
length *(returned by GDDM) (fullword integer)*
The length of the associated data buffer.
depth-2 *(returned by GDDM) (fullword integer)*
The number of rows in the bundle list array (stored in row major order).
width-2 *(returned by GDDM) (fullword integer)*
The number of columns in the bundle list array.

Description

Returns the size of the specified field list, data buffer, and bundle list.

Principal errors

```
ADM3002 E  FIELD LIST IDENTIFIER n IS INVALID
ADM3008 E  FIELD LIST n DOES NOT EXIST
```

APQUID

Function

To query unique field list identifier.

APQUID	(field-list-id)
APL code	287
GDDM RCP code	X'0C380700' (204998400)

Parameters

field-list-id *(returned by GDDM) (fullword integer)*
A value that is not currently in use for a field list identifier.

Description

Returns a value that is not currently in use for a field list identifier.

Principal errors

None.

ASCCOL

Function

To specify character colors within a field.

ASCCOL	(field-id, length, color-string)
APL code	421
GDDM RCP code	X'0C080601' (201852417)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.
length *(specified by user) (fullword integer)*
The length of **color-string**. This is padded with default values to the length (width by depth) of the field.
color-string *(specified by user) (character)*
The new colors for each character position within the field (starting at the top left-hand corner and working from left to right for each row of the field). Each character must be one of the following (characters are first given in the EBCDIC form, then in hexadecimal):

Blank	X'40'	Inherit the color set by ASFCOL (the default).
1	X'F1'	Blue.
2	X'F2'	Red.
3	X'F3'	Pink (magenta).
4	X'F4'	Green.
5	X'F5'	Cyan (turquoise).
6	X'F6'	Yellow.
7	X'F7'	Neutral (white on displays, black on printers).

If the field is defined as “mixed without position” (see ASFSEN), the attributes for SO/SI positions in the string (that can be obtained by a call to ASCGET) have no effect and are ignored.

Description

Defines the color attributes to be used for individual characters in the specified field.

Note: The default color for each character is governed by the current value set by ASFCOL (or defaulted) for the whole field.

Because ASCPUT sets character attributes to their **default** values, ASCCOL is effective only when it **follows** a call to ASCPUT.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0215 W  FIELD n1 CHARACTER n2 OF ATTRIBUTE STRING IS
           INVALID
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASCGET

Function

To get field contents.

ASCGET	**(field-id, length, string)**
APL code	422
GDDM RCP code	X'0C080903' (201853187)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field containing the required characters.

length *(specified by user) (fullword integer)*
The number of bytes to be returned. This number of bytes is returned, regardless of the size of the field.

string *(returned by GDDM) (character)*
The character codes, having been through any input translation; see ASFTRN.

Description

Returns the contents of the specified field. The field can contain EBCDIC, DBCS (used for Kanji and Hangeul), or mixed (EBCDIC and DBCS) characters.

EBCDIC character codes are translated if any input table has been assigned to the field; see ASFTRN. The EBCDIC parts of mixed fields are also translated; however, the DBCS parts of fields are not translated.

If the returned length is greater than the field length, the string is padded with the pad character. The pad character (null by default), depends on the setting of input processing for the field as determined by ASFIN.

For values 0 and 1, the pad character is a null (X'00').

For value 2, the pad character is a blank (X'40').

If the field is defined as “mixed without position” (see ASFSEN), SO (shift-out) and SI (shift-in) control codes are inserted to delimit the DBCS portion of the returned string. Therefore, the length of the returned string is greater (by the number of inserted control codes) than the length of the string as displayed. The length necessary to retrieve the field contents is obtained by means of the **i/p length** parameter of the ASQLEN call or the ASQMOD calls.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASCHLT

Function

To specify character highlights within a field.

ASCHLT	**(field-id, length, highlight-string)**
APL code	423
GDDM RCP code	X'0C080600' (201852416)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

length *(specified by user) (fullword integer)*
The length of **highlight-string**. This is padded with default values to the length (width by depth) of the field.

highlight-string *(specified by user) (character)*
The new highlighting for each character position within the field (starting at the top left-hand corner and working from left to right for each row of the field). Each byte must be one of the following:

EBCDIC	**Hex.**	
Blank	X'40'	Inherit the highlighting set by ASFHLT
1	X'F1'	Blink
2	X'F2'	Reverse video
4	X'F4'	Underscore

If the field is defined as “mixed without position” (see ASFSEN), the attributes for SO/SI control code positions in the string (that can be obtained by a call to ASCGET) have no effect and are ignored.

Description

Defines the highlight attributes to be used for individual characters in the specified field.

Note: The default attribute for each character is governed by the current value set by ASFHLT (or defaulted) for the whole field.

Because ASCPUT sets character attributes to their **default** values, ASCHLT is effective only when it **follows** a call to ASCPUT.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0215 W  FIELD n1 CHARACTER n2 OF ATTRIBUTE STRING IS
           INVALID
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASCPUT

Function

To specify field contents.

ASCPUT	**(field-id, length, string)**
APL code	424
GDDM RCP code	X'0C080603' (201852419)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be filled.

length *(specified by user) (fullword integer)*
The exact length of the **string** in bytes. If the number of bytes in **string** is less than the total number of characters that can be accommodated in the field, the field is padded with nulls after the specified characters. If the number of bytes specified exceeds the number that can be accommodated, characters on the right of the string are lost.

string *(specified by user) (character)*
The new character codes for each position within the field (starting at the top left-hand corner and working left to right for each row of the field). Each EBCDIC character is encoded as one byte. Each EBCDIC character may pass through an output translation; see ASFTRN. The string may contain DBCS characters; see ASFPSS and ASFSEN.

If the depth of the specified field is greater than 1, and its width is less than either the page width or the screen-width of the device, a mixed string of 1-byte (SBCS) and 2-byte characters (DBCS) cannot be specified.

Each DBCS character is encoded as two bytes. These DBCS characters are valid:

X'0000' (null)
X'4040' (blank)
X'xxyy' Where **xx** is in the range X'41' through X'FE', and **yy** is in the range X'41' through X'FE'.

Description

Fills the specified field with the given character string. All character attributes for the field are set to their **default** values. The string may pass through output translation; see ASFTRN.

If the field is defined as “mixed without position” (see ASFSEN), the length of the string as displayed is less than the value of the **length** parameter by the number of SO/SI (shift-out/shift-in) control codes.

Validity checking is performed on any DBCS characters (used for Kanji and Hangeul) contained in the string.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
ADM0223 W  FIELD n DBCS CHARACTER X'xxxx' IS INVALID
           AND REPLACED BY BLANK
ADM0228 W  FIELD n CHARACTER X'xx' REPLACED BY SHIFT-IN
           CHARACTER
ADM0229 W  FIELD n LAST CHARACTER WAS SHIFT-OUT AND HAS
           BEEN IGNORED
```

ASCSS

Function

To specify character symbol sets within a field.

ASCSS	**(field-id, length, symbol-set-id-string)**
APL code	425
GDDM RCP code	X'0C080602' (201852418)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

length *(specified by user) (fullword integer)*
The length of **symbol-set-id-string**. This is padded with default values to the length (width by depth) of the field.
symbol-set-id-string *(specified by user) (character)*
The new symbol-set identifiers for each character position within the field (starting at the top left-hand corner and working left to right for each row of the field). Each character must be one of:

X'00' or X'40'	Inherit the symbol-set identifier set by ASFPSS (the default)
X'01' through X'03'	Loadable symbol sets (3800-system printer)
X'41' through X'DF'	Loadable symbol sets (3270-family devices)
X'F1'	Alternative nonloadable symbol set (3270-family devices).

If the field is defined as "mixed without position" (see ASFSEN), the attributes for SO/SI control code positions in the string that can be obtained by ASCGET have no effect and are ignored.

Description

Defines the symbol-set identifiers to be used for individual characters in the specified field.

Note: The default symbol-set identifier for each character is governed by that specified by ASFPSS (or defaulted) for the whole field.

Because ASCPUT sets character attributes to their **default** values, ASCSS is effective only when it **follows** a call to ASCPUT.

For information about which devices support this attribute, see "Alphanumeric field attributes" in Chapter 4, "Device variations" on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0215 W  FIELD n1 CHARACTER n2 OF ATTRIBUTE STRING IS
           INVALID
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASDFLD

Function

To define or delete a single field.

ASDFLD	**(field-id, row, column, depth, width, type)**
APL code	401
GDDM RCP code	X'0C080700' (201852672)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field. This must be a positive integer.
row *(specified by user) (fullword integer)*
The row for the top left-hand corner of the field. Rows are numbered from top to bottom down the page, starting with 1. Zero indicates that the field is to be deleted.
column *(specified by user) (fullword integer)*
The column for the top left-hand corner of the field. Columns are numbered from left to right across the page, starting with 1. Zero indicates that the field is to be deleted.
depth *(specified by user) (fullword integer)*
The number of rows that the field occupies. This must be such that the field does not extend beyond the bottom of the page. Zero indicates that the field is to be deleted. For a mixed field, the depth of the field cannot be greater than 1 unless the width of the field is equal to the width of the device; see ASFSEN.
width *(specified by user) (fullword integer)*
The number of columns that the field occupies. For a field the depth of which is greater than 1, the width must not extend beyond the right-hand side of the page. If the field is intended for use with a dual-character primary symbol set, the width must be even; see ASFPSS. Zero indicates that the field is to be deleted.
type *(specified by user) (fullword integer)*
These field types are valid:
–**1** Default (Same as 0, unless othewise defined by a prior call to ASDFLT)
0 Unprotected alphanumeric.
1 Alphanumeric output, numeric input.
2 Protected alphanumeric.
3 Protected alphanumeric, immediate pen-selectable.
4 Protected alphanumeric, deferred pen-selectable.
5 Protected alphanumeric, pen-enterable.
6 Protected alphanumeric, general light-pen.

Description

Defines or deletes a single field. Any existing field of the same field-identifier is deleted. If the cursor was positioned within the deleted field, and there are no mapped fields, the cursor position is reset to the top left-hand corner of the page window – no error is returned if the field does not exist. If there are mapped fields, the cursor is placed at the static cursor position; see MSDFLD.

Attributes of the field assume their default values (which may have been modified by a call to ASDFLT) except for **type**, which is specified by the **type** parameter above.

If any of the position or size parameters are zero, the field is deleted. If there are no mapped fields, the cursor position is reset to the top left-hand corner of the page window. If there are mapped fields, the cursor is placed at the static cursor position; see MSDFLD.

For the best performance, alphanumeric fields should be defined in numeric identifier order from left to right, top to bottom.

A field is usually contained within the rectangle described by the row, column, depth, and width parameters. The rectangle must be within the top, bottom, and left-hand boundaries of the current page, and except when the depth is 1, the right-hand boundary as well.

When the depth is 1, the rectangle can extend beyond the right-hand boundary of the page. The field contents are wrapped around to the start of the next row and subsequent rows until the total desired field width is achieved, provided that the field does not overlap any existing field. This type of field is treated in all other respects as a field with a single row.

A page that is to be used for mapping (that is, one created using an MSPCRT call) can have alphanumeric fields defined in it by using this call but they must not overlap any mapped fields created by using the MSDFLD call. Furthermore, care should be taken when placing alphanumeric and mapped fields next to each other, because of the attribute bytes that enclose the alphanumeric fields. Undesirable results can occur if attributes intrude into a mapped field.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0203 W  FIELD n LIGHT-PEN ATTRIBUTE MAY BE
           INEFFECTIVE
ADM0204 W  FIELD n IS TOO SMALL TO BE A LIGHT PEN FIELD
ADM0205 E  FIELD n POSITION IS INVALID
ADM0206 E  ALPHANUMERIC FIELD a1 OVERLAPS ALPHANUMERIC
           FIELD a2
ADM0211 W  {FIELD n1} ATTRIBUTE {n2} VALUE n3 IS
           INVALID
ADM0224 W  FIELD n1 WIDTH n2 MUST BE EVEN FOR A DBCS
           FIELD
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
ADM0227 W  FIELD n CANNOT BE SET TO MIXED STATUS
ADM0968 E  ALPHANUMERIC FIELD a1 AND MAPPED FIELD a2
           OVERLAP
ADM3012 E  CURRENT PAGE HAS HIGH PERFORMANCE
           ALPHANUMERICS
```

ASDFLT

Function

To set default field attributes.

ASDFLT	**(count, array)**
APL code	406
GDDM RCP code	X'0C080200' (201851392)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in **array**. This must be in the range 1 through 12.

array *(specified by user) (an array of fullword integers)*
This contains **count** values, the following representing the maximum set:

1 Type	See ASFTYP.
2 Intensity	See ASFINT.
3 Color	See ASFCOL.
4 Primary symbol set	See ASFPSS.
5 Highlight	See ASFHLT.
6 End	See ASFEND.
7 Nulls	See ASFOUT.
8 Blanks	See ASFIN.
9 Table number	See ASFTRN.
10 Transparency	See ASFTRA.
11 Enable/disable shift-control codes	See ASFSEN.
12 Field outlining	See ASFBDY.

Note: Specifying a value of –1 leaves the default unchanged. Otherwise, any default values defined for the current page by a previous call to ASDFLT are replaced.

Description

Sets the default attributes to be assumed for new field definitions. These defaults, where specified, are used in preference to the standard defaults on the GDDM page. The standard defaults are listed under the appropriate call descriptions.

Principal errors

```
ADM0208 E  COUNT n IS INVALID
ADM0211 W  {FIELD n1} ATTRIBUTE {n2} VALUE n3 IS
           INVALID
```

ASDFMT

Function

To define alphanumeric fields, deleting all existing fields.

ASDFMT	**(n-fields, count, array)**
APL code	402
GDDM RCP code	X'0C080801' (201852929)

Parameters

n-fields *(specified by user) (fullword integer)*
The number of fields to be defined. This may be zero.

count *(specified by user) (fullword integer)*
The number of attributes provided for each field; it must be in the range 5 through 17.

array *(specified by user) (an array of fullword integers)*
An array of field definitions, each of which may contain the following elements. The number of elements actually present is limited by **count**; thus, the array contains **n-fields** definitions, each consisting of **count** elements. All of the elements for definition **i** precede those for definition **i+1**.

1 Field-id
The number of the field. Zero indicates no field definition.
2 Row
The row for the top left-hand corner of the field. Rows are numbered from top to bottom of the page, starting with 1. Zero indicates no field definition.
3 Column
The column for the top left-hand corner of the field. Columns are numbered from left to right across the page, starting with 1. Zero indicates no field definition.
4 Depth
The number of rows that the field occupies. This must be such that the field does not extend beyond the bottom of the page. Zero indicates no field definition. For a mixed field (see ASFSEN), the depth of the field cannot be greater than 1 unless the width of the field is equal to the width of the device.
5 Width
The number of columns that the field occupies. For a field the depth of which is greater than 1, the width must not extend beyond the right-hand side of the page. If the field is intended for use with a dual-character primary symbol set (see ASFPSS), the width must be even. Zero indicates no field definition.

6 Type	See ASFTYP.
7 Intensity	See ASFINT.
8 Color	See ASFCOL.
9 Primary symbol set	See ASFPSS.
10 Highlight	See ASFHLT.
11 End	See ASFEND.
12 Nulls	See ASFOUT.
13 Blanks	See ASFIN.
14 Table number	See ASFTRN.
15 Transparency	See ASFTRA.
16 Enable/disable shift-control codes	See ASFSEN.
17 Field outlining	See ASFBDY.

Description

Defines alphanumeric fields for the current page. All existing alphanumeric fields are deleted. To define multiple fields **without** deleting existing fields, use ASRFMT. The cursor position is reset to the top left-hand corner of the page window if it was previously in an alphanumeric field.

The number of attributes specified for each field is determined by **count**. Any attributes that are not specified are chosen according to their default values; see ASDFLT.

For the best performance, alphanumeric fields should be defined in numeric identifier order, from left to right, from top to bottom.

Principal errors

```
ADM0203 W  FIELD  n  LIGHT-PEN  ATTRIBUTE  MAY  BE
           INEFFECTIVE
ADM0204 W  FIELD n IS TOO SMALL TO BE A LIGHT PEN FIELD
ADM0205 E  FIELD n POSITION IS INVALID
ADM0206 E  ALPHANUMERIC FIELD a1 OVERLAPS ALPHANUMERIC
           FIELD a2
ADM0207 E  NUMBER OF FIELDS n IS NEGATIVE
ADM0208 E  COUNT n IS INVALID
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0224 W  FIELD n1 WIDTH n2 MUST BE EVEN FOR A DBCS
           FIELD
ADM0225 E  FIELD n ALREADY EXISTS
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
ADM0227 W  FIELD n CANNOT BE SET TO MIXED STATUS
ADM3012 E  CURRENT   PAGE   HAS   HIGH   PERFORMANCE
           ALPHANUMERICS
```

ASDTRN

Function

To define I/O translation tables.

ASDTRN	**(table-no, o/p-table, i/p-table)**
APL code	403
GDDM RCP code	X'0C080300' (201851648)

Parameters

table-no *(specified by user) (fullword integer)*
The number to be assigned for this translation table set. This must be in the range 2 through 7. Table set number 1 is reserved for uppercase input translation; see ASFTRN. Zero means that no translation is to be done.
o/p-table *(specified by user) (256-byte character string)*
The output translation table.

i/p-table *(specified by user) (256-byte character string)*
The input translation table.

Description

Defines or redefines to GDDM a translation-table set **for the current page** according to the definitions below. The tables are **not** copied from user storage, and so the user copy should not be released while in use. It is recommended that the tables, once defined, are not modified without subsequent redefinition (again using ASDTRN). These tables should be available to all alphanumeric fields on the appropriate pages while those pages exist.

Both output and input translation tables consist of 256 one-byte entries. The offset of each entry represents the character code before translation. The contents of the table at that offset represent the character code after translation.

A translation-table set that replaces an existing one does **not** affect any characters already written (see ASCPUT) using the previous definitions. It **does** immediately affect input using ASCGET.

The output translation is applied when characters are inserted into the field, and the input translation is applied when characters are extracted.

Principal errors

```
ADM0200 E  TABLE NUMBER n IS INVALID
```

ASFBDY

Function

To define a field outline.

ASFBDY (field-id, outlining)	
APL code	436
GDDM RCP code	X'0C08050B' (201852171)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.
outlining *(specified by user) (fullword integer)*
The new outlining attribute for the field. Possible values are:
–1 Leave the outlining as it is.
0 None (the default).
1 Underline.
2 Vertical line on the right.
4 Overline.
8 Vertical line on the left.

For an outlining attribute that is composed of more than one of these lines, specify (as an integer) the sum of the numbers corresponding to the lines required. For example, a value of 11 causes the field to be outlined on three sides (omitting the line at the top) and a value of 15 causes the field to be outlined on all four sides.

Description

Defines the outlining attribute for the specified field.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFCLR

Function

To clear fields.

ASFCLR (code)	
APL code	404
GDDM RCP code	X'0C080400' (201851904)

Parameters

code *(specified by user) (fullword integer)*
The operation required. Possible values are:
0 Erase all unprotected fields within the page (fill with null characters and default character attributes).
1 Erase all protected fields within the page (fill with null characters and default character attributes).
2 Both of the above.

Description

Fills alphanumeric fields with their default character codes and attributes.

Principal errors

```
ADM0218 E  CODE n IS INVALID
```

ASFCOL

Function

To define a field color.

ASFCOL (field-id, color)	
APL code	407
GDDM RCP code	X'0C080502' (201852162)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

color *(specified by user) (fullword integer)*
The new color for the field. Possible values are:

–1	Leave the color as it is.
0	Default.
1	Blue.
2	Red.
3	Pink (magenta).
4	Green.
5	Turquoise (cyan).
6	Yellow.
7	Neutral (white on color displays, black on printers).

Description

Defines the color of *all* character positions in the specified alphanumeric field.

For information about specifying the color of *individual* characters within the field, see ASCCOL.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFCUR

Function

To position the cursor.

ASFCUR (field-id, row, column)	
APL code	430
GDDM RCP code	X'0C080100' (201851136)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field in which the cursor is to be positioned. If this is zero, the cursor is to be positioned on the current page.

row *(specified by user) (fullword integer)*
The new position of the cursor within either the field or the current page, depending on the value of **field-id**.

The value –1 and a valid **field-id** applies the string coordinate to the **column** value.

column *(specified by user) (fullword integer)*
The new position of the cursor within the field, the field contents, or the current page, depending on the value of **field-id**.

If the string coordinate is specified by the row value of –1, the column represents the character position in the field contents, that is, the position in the string that can be retrieved by ASCGET.

For a cursor position within a dual-character field, the column represents the dual character on which the cursor is to be positioned (for example, for column=2, the cursor is positioned in the third or third and fourth physical columns within the field, depending upon the device).

For a cursor position within a mixed field, the column indicates the byte position. If this is at a DBCS character position on a device which supports DBCS characters, the cursor is positioned at the start of the DBCS character.

If the cursor position in the string coordinates coincides with the position of the SO/SI (shift-out/shift-in) control codes in the “mixed without position” field, the cursor is displayed in the next character position.

Description

Positions the cursor by either field, string, or page coordinates.

The default position for the cursor is the top left-hand corner of the page. This applies even when there are no alphanumeric fields on the page.

If the cursor is moved outside the page window, the window is altered to accommodate it; this function is called “scrolling.”

Note: ASFCUR allows both vertical and horizontal scrolling (Screen Up and Screen Down, and Screen Left and Screen Right).

If horizontal scrolling is required, horizontal movement of the window places the left-hand edge as close as possible to the cursor, in a similar manner to vertical scrolling.

If the page is mapped, either cursor adjuncts and the MSCPOS call or the ASFCUR call can be used to position the cursor.

Principal errors

```
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0221 E  CURSOR POSITION OF FIELD n1, ROW n2, COLUMN
           n3 IS INVALID
ADM3156 I  PAGE n1 WINDOW ROW ALTERED TO n2 AND COLUMN
           TO n3
```

ASFEND

Function

To define field end action.

ASFEND	**(field-id, end)**
APL code	408
GDDM RCP code	X'0C080505' (201852165)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

end *(specified by user) (fullword integer)*
The new setting of the field end attribute. Possible values are:

–**1** Leave the field end setting as it is.
0 Autoskip at end of input field (the default).
1 Do not autoskip at end of input field.

> **Note:** This value has a hardware-dependent effect. If there are fewer than three spaces before the next unprotected field, the cursor will move to the next field regardless of the value of this parameter.

Description

Defines the kind of action that is taken after terminal input to each row of the specified field. The setting controls the cursor action when completing input to the field. Use of autoskip causes the cursor to jump to the next unprotected field.

All fields have their width controlled; however, this setting only has relevance to unprotected fields.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD n1} ATTRIBUTE {n2} VALUE n3 IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFHLT

Function

To define field highlighting.

ASFHLT	**(field-id, highlight)**
APL code	409
GDDM RCP code	X'0C080504' (201852164)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

highlight *(specified by user) (fullword integer)*
The new highlighting for the field. Possible values are:

–**1** Leave the highlighting as it is.
0 Normal (the default).
1 Blink.
2 Reverse video.
4 Underscore.

Description

Defines the highlighting for all character positions in the specified field. For information on specifying highlighting of individual characters within the field, see ASCHLT.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD n1} ATTRIBUTE {n2} VALUE n3 IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFIN

Function

To define input null-to-blank conversion.

ASFIN (field-id, blanks)	
APL code	410
GDDM RCP code	X'0C080507' (201852167)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

blanks *(specified by user) (fullword integer)*
The new setting of the blanks attribute. Possible values are:

–**1** Leave the blanks setting as it is.
0 No input conversion (the default).
1 Convert nulls to blanks on input (ASREAD) for all nulls that precede the last nonnull character (viewing the field as one long string, from top left to bottom right).
2 Convert all nulls to blanks on input (ASREAD) for each row of the field.

Description

Defines the kind of nulls-to-blanks conversion to occur on device input for a specified field.

Note: Conversion is effective only for non-light-pen fields. Nulls imbedded between non-null characters are not converted to blanks. Blanks will only be inserted into the field where the presence of nulls can be assumed by GDDM (to complete a field, or a row within a field).

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFINT

Function

To define field intensity.

ASFINT (field-id, intensity)	
APL code	411
GDDM RCP code	X'0C080501' (201852161)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

intensity *(specified by user) (fullword integer)*
The new intensity for the field. Possible values are:

–**1** Leave the intensity as it is.
0 Invisible.
1 Normal (the default).
2 Bright.

Notes:

1. A bright field may be light-pen selectable, regardless of the setting of ASFTYP. See also the *IBM 3270 Information Display System: Component Description*.
2. An “invisible” field is displayed if it is light-pen detectable.

Description

Defines the intensity of the specified field. The contents of an invisible field are not sent to anything other than the currently-attached device, or a printer, unless it is light-pen detectable.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFMOD

Function

To change field status.

ASFMOD (field-id, mod)	
APL code	412
GDDM RCP code	X'0C081100' (201855232)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be affected. If this parameter is zero, all of the **unprotected** fields (type 0) on the current page are affected.

mod *(specified by user) (fullword integer)*
The new setting of the modify attribute. Possible values are:

0 Make the field (or fields) unmodified.
1 Make the field (or fields) modified.

Description

Designates an alphanumeric field (or fields) as modified or unmodified. A modified field is added to the list of modified fields. The field is then available to ASQMOD as a modified field. The other way in which a field can become modified is by operator changes. Modified fields become unmodified when they are returned by ASQMOD. An unmodified field (the initial status) is not returned by a call to ASQMOD.

Under CICS/VS and IMS/VS, GDDM causes the physical modified data tag (MDT) bits to be set for fields that are designated as modified so that the fields can be returned as input to a subsequent application program after GDDM terminates.

Under OS/TSO and VM/CMS, GDDM does not generally cause the physical MDT bits to be set for fields that are designated as modified.

Principal errors

```
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFOUT

Function

To define output blank-to-null conversion.

ASFOUT	**(field-id, nulls)**
APL code	413
GDDM RCP code	X'0C080506' (201852166)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

nulls *(specified by user) (fullword integer)*
The new setting of the nulls attribute. Possible values are:

–1 Leave the nulls setting as it is.

0 No output conversion (the default).

1 Convert trailing blanks to nulls on each row of the field on output.

Description

Defines the kind of blanks-to-nulls conversion to occur on field output to the device for the specified field.

Note: This does not affect what is returned to the application program with the ASCGET call; only device input to this field can achieve that.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFPSS

Function

To define primary symbol set for a field.

ASFPSS	**(field-id, symbol-set-id)**
APL code	414
GDDM RCP code	X'0C080503' (201852163)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

symbol-set-id *(specified by user) (fullword integer)*
The new primary symbol set alias for the field. Possible values are:

–1
Leave the primary symbol set as it is.

0
For a 3270-family device, the base nonloadable symbol set (the default). For a 3800-system printer, the first loadable symbol set (the default); use the CHARS parameter of the subsystem-dependent family-3 system print destination definition (for example, CP SPOOL or MVS DD statement) to specify the loaded symbol sets when printing.

1 through 3
For a 3800-system printer, the second, third, and fourth loadable symbol sets (respectively). Use the CHARS parameter described above to specify the loaded symbol sets when printing.

65 through 223
For a 3270-family device, loadable symbol sets corresponding to X'41' through X'DF'; see ASCSS. The alias must have been made known to GDDM with a call to PSDSS, PSLSS, or PSLSSC to load the symbol set.

248 (X'F8')
For DBCS devices, the DBCS (used for Kanji and Hangeul) nonloadable symbol set. This is a double-character symbol set. Each code point consists of two bytes instead of one. This selection is rejected if the field has been set to mixed status; see ASFSEN.

Note: For a 3270-family device, the alternative nonloadable symbol set (APL/TEXT) cannot be selected as the primary symbol set. If they are available, these characters are accessible as part of the base nonloadable symbol set but they may also be referenced using character attributes; see ASCSS.

Description

Defines the primary symbol set for all character positions in the specified field. For information about specifying the symbol set identifiers for individual characters within the field, see ASCSS – these may not be used when the symbol set chosen is a double-character symbol set.

If the primary symbol set for a field is changed from single character to double character, or the converse, the field contents are cleared.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
ADM0224 W  FIELD n1 WIDTH n2 MUST BE EVEN FOR A DBCS
           FIELD
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
```

ASFSEN

Function

To define field mixed-string attribute.

ASFSEN	**(field-id, mixed)**
APL code	437
GDDM RCP code	X'0C08050A' (201852170)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

mixed *(specified by user) (fullword integer)*
The mixed-string attribute for the field. Possible values are:

–1 Leave the mixed status of the field as it is.
0 Non-mixed (the default).
1 Mixed with position.
2 Mixed without position.

Description

Defines the mixed-string attribute for the specified field.

Usually, the strings presented to GDDM by ASCPUT are either all single-byte characters (where each character is represented by one byte in the string) or all double-byte characters (where each character is represented by two bytes in the string) according to the primary symbol set as defined by the ASFPSS call. This is called a normal, or nonmixed, string.

An application program can, using the ASCPUT call, present a mixed string of single-byte and double-byte (DBCS) characters. If it does so, GDDM searches such a string for the shift-out (X'0E') and shift-in (X'0F') control codes, which indicate the start and end of double-byte portions of the string.

Shift-out (SO) and shift-in (SI) control codes are treated in one of two ways, according to the mixed-string attribute defined by ASFSEN:

- If the attribute is 0 (nonmixed), or 1 (mixed-with-position), the SO/SI control codes each take one byte character position.
- If the attribute is 2 (mixed-without-position), the SO/SI control codes take no position.

On devices that support mixed alphanumeric fields (such as the IBM 5550-family Multistations, the application can present mixed strings to GDDM using ASCPUT in **any** alphanumeric field. However, if it is necessary to allow the input of mixed strings, the field must be defined explicitly as “mixed” using the ASFSEN call.

If the field is defined as “mixed with position,” mixed strings can be input by delimiting one-byte and double-byte characters with SO/SI control codes. If the field is defined as “mixed without position,” mixed strings can be input without using the SO/SI control codes. When the input string is returned by ASCGET, the SO/SI control codes are inserted by GDDM.

On display devices that do not support mixed alphanumeric fields, and if MIXSOSI=YES has been specified as an external default, GDDM emulates the SO/SI control codes using a substitution character defined by the SOSIEMC external default. However, for these devices, the application can present mixed strings to GDDM using ASCPUT, but only in alphanumeric fields that have been defined as “mixed” using the ASFSEN call. DBCS characters are presented and input in the hexadecimal code point. For these devices, the mixed-string attribute values 1 and 2 do not change the support of the SO/SI emulation.

If the mixed-string attribute is changed, the field contents are cleared. When emulating, changing from 1 to 2 and from 2 to 1 is not treated as a change and, therefore, the contents will not be cleared.

If a field has a PSS attribute of DBCS, it cannot be set to mixed status. Also, if a field has a depth of greater than 1, it cannot be set to mixed status unless its width is the same as the width of the device.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
ADM0227 W  FIELD n CANNOT BE SET TO MIXED STATUS
```

ASFTRA

Function

To define field transparency attribute.

ASFTRA	**(field-id, transparency)**
APL code	434
GDDM RCP code	X'0C080509' (201852169)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

transparency *(specified by user) (fullword integer)*
The transparency attribute for the field. Possible values are:

–**1** Leave the transparency as it is
0 Opaque (the default)
1 Transparent.

Description

Defines the transparency attribute for the specified field. A transparent field allows graphics and image data to be seen “through” an alphanumerics field where the two fields cross one another.

For information about which devices support this attribute, see “Alphanumeric field attributes” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFTRN

Function

To assign translation table set to a field.

ASFTRN	**(field-id, table-no)**
APL code	415
GDDM RCP code	X'0C080508' (201852168)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

table-no *(specified by user) (fullword integer)*
The new translation table number for the field. Possible values are:

–**1** Leave the translation table number as it is.
0 No translation (the default).
1 Uppercase translation. Lowercase characters are translated to uppercase on input only. This action is suppressed when the primary symbol set is the base nonloadable symbol set and the device does not support lowercase characters (for example, a Katakana terminal).
2 through 7 Translation-table set number; see ASDTRN.

Description

Assigns a translation-table set to be used for the specified field. The translation tables are defined using ASDTRN.

Translation has no effect for dual-character fields; see ASFPSS.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASFTYP

Function

To define field type.

ASFTYP (field-id, type)	
APL code	416
GDDM RCP code	X'0C080500' (201852160)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be modified.

type *(specified by user) (fullword integer)*
The new type for the field. Possible values are:

–1 Leave the field type as it is.
0 Unprotected alphanumeric (the default).
1 Alphanumeric output, numeric input.
2 Protected alphanumeric.
3 Protected alphanumeric, immediate pen-selectable.
4 Protected alphanumeric, deferred pen-selectable.
5 Protected alphanumeric, pen-enterable.
6 Protected alphanumeric, general light-pen.

Description

Defines the type of the specified field. This type groups together various physical field attributes.

The width of light-pen fields (types 3, 4, 5, and 6) must be at least 3. A designator character is always placed (by GDDM) in the first character position of each field row for types 3, 4, and 5. This designator character is a blank for an immediate pen-selectable field, an ampersand “&” for an enterable field, and either a greater-than sign “>” or a question mark “?” for a deferred pen-selectable field. The “?” is displayed when the field is unmodified (the default), and the “>” when the field has been modified by the operator.

For the general light-pen field (type 6), it is the programmer’s responsibility to provide designator characters at the start of each row of the field.

Because of the nature of the 3270 field attributes, it is possible to create a pen-selectable field on a display without having defined it as such in the ASFTYP call. This situation occurs with any high-intensity field that has a valid light-pen designator character as the first character after the attribute byte (that is, in the first character position in each row). If such a field is selected, GDDM does not recognize it as a modified field (in the same way as changes to non-light-pen fields are not recognized when a GDDM-defined light-pen field is selected).

For information about which devices support this attribute, see “Alphanumeric field attributes” on page 245.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0203 W  FIELD  n  LIGHT-PEN  ATTRIBUTE  MAY  BE
           INEFFECTIVE
ADM0204 W  FIELD n IS TOO SMALL TO BE A LIGHT PEN FIELD
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
```

ASGGET

Function

To get double-character field contents.

Note: This call is not recommended for new programs. It is obsolete and has been superseded by ASCGET.

ASGGET (field-id, length, string)	
APL code	433
GDDM RCP code	X'0C081603' (201856515)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field containing the required double characters.

length *(specified by user) (fullword integer)*
The number of double characters to be returned. Twice **length** bytes (two for each double character) are returned, regardless of the size of the field.

string *(returned by GDDM) (array of 2-byte character tokens)*
The double-character codes, without translation. Remember to allocate twice as many bytes as there are double characters.

Description

Returns the double-character contents of the specified double-byte character string (DBCS, used for Kanji or Hangeul). The double-character codes are not translated in any way.

If the returned length is greater than the field length, the string is padded with the double-character padding code. The double-character padding code (default X'0000') depends on the setting of input processing for the field as determined by ASFIN.

For values 0 and 1, the pad double-character is a null (X'0000').

For value 2, the pad double-character is a blank (X'4040').

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASGPUT

Function

To specify double-character field contents.

Note: This call is not recommended for new programs. It is obsolete and has been superseded by ASCPUT.

ASGPUT	**(field-id, length, string)**
APL code	432
GDDM RCP code	X'0C081503' (201856259)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field to be filled.

length *(specified by user) (fullword integer)*
The exact length of the **string** in double characters (the number of bytes divided by two). If the number of double characters in the **string** is less than the total number of double characters that can be accommodated in the field, the field is padded with nulls after the specified double characters. If the number of double characters specified exceeds the number that can be accommodated, double characters on the right of the string are lost.

string *(specified by user) (array of 2-byte character tokens)*
The new double-character codes for each position within the field (starting at the top left-hand corner and working left to right for each row of the field). Each double character is coded as two bytes. Each double-character code-point is checked for validity. The following DBCS characters are valid:

X'0000' (null)
X'4040' (blank)
X'xxyy' Where **xx** is in the range X'41' through X'FE', and **yy** is in the range X'41' through X'FE'.

Description

Fills the specified double-byte character string field (DBCS, used for Kanji and Hangeul) with the given double-character string. The string is not translated in any way.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
ADM0223 W  FIELD n DBCS CHARACTER X'xxxx' IS INVALID
           AND REPLACED BY BLANK
```

ASMODE

Function

To define the operator reply mode.

ASMODE	**(mode)**
APL code	426
GDDM RCP code	X'0C080D00' (201854208)

Parameters

mode *(specified by user) (fullword integer)*
The mode of interaction available to the operator. Possible values are:

1 Field mode (the default). The operator may not modify character attributes (color, highlighting, or symbol sets). Any character attributes changed implicitly by operator typing (that is, resetting of character attributes to their default values) are **not** reflected back to GDDM.

2 Character mode. The operator may modify character attributes (color, highlighting, or symbol sets). Any character attributes changed implicitly or explicitly by operator typing are reflected back to GDDM.

Description

Defines the reply mode to be used for the current page.

For information about which devices support this attribute, see “Alphanumeric field attributes” on page 245.

Principal errors

```
ADM0222 E  MODE n IS INVALID
```

ASQCOL

Function

To query character colors for a field.

ASQCOL	**(field-id, length, color-string)**
APL code	427
GDDM RCP code	X'0C080901' (201853185)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field containing the required character attributes.

length *(specified by user) (fullword integer)*
The number of character attributes to be returned (may be zero). This number of bytes is returned, regardless of the size of the field.

color-string *(returned by GDDM) (character)*
See ASCCOL.

Description

Returns the character color attributes for a given length of the specified field.

If the field is defined as "mixed-without-position," (see ASFSEN), default color attributes are inserted, corresponding to the positions where SO/SI (shift-out/shift-in) control codes are inserted in the string returned by ASCGET. Therefore, the length of the returned string is greater than that displayed on the device by the number of inserted control codes. The length necessary to retrieve the field contents is obtained by means of the **i/p length** parameter of the ASQLEN call or the ASQMOD call.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASQCUR

Function

To query cursor position.

ASQCUR (code, field-id, row, column)	
APL code	431
GDDM RCP code	X'0C080F00' (201854720)

Parameters

code *(specified by user) (fullword integer)*
Indicates whether the cursor position is to be returned in page or field coordinates. Possible values are:

0 Return page coordinates.

1 Return field coordinates.
If this is not possible, page coordinates are returned. Field coordinates may be received under the following conditions:

- The last cursor operation was a call to ASFCUR specifying field coordinates, and the field is still defined.
- The last cursor operation was a call to ASREAD, which sets the cursor position to field coordinates whenever possible.

2 Return string coordinates.
If this is not possible, page coordinates are returned. String coordinates may be received under the same condition as the field coordinates; see above.

The cursor may not be set in page coordinates and then read in field or string coordinates without an intervening call to ASREAD.

field-id *(returned by GDDM) (fullword integer)*
Returns information on the coordinates of the cursor position. Possible values are:

0 Page coordinates have been returned.

>**0** The number of the field containing the cursor. Field coordinates have been returned.

row *(returned by GDDM) (fullword integer)*
The position of the cursor within either the field or the current page, depending on the value of **field-id**.

The value –1 is returned if the string coordinate is specified by the **code** parameter.

column *(returned by GDDM) (fullword integer)*
The position of the cursor within either the field or the current page, depending on the value of **field-id**. For a cursor positioned within a dual-character field, the column represents the dual character on which the cursor is positioned. For example, if **column=2**, the cursor is positioned in the third or third and fourth physical columns within the field, depending upon the device.

For a cursor position within a mixed field, the column indicates the byte position. If this is at a DBCS character position on a DBCS device, the cursor is double width; the column indicates the byte position at which the cursor begins and coincides with the start of the DBCS character.

If the string coordinate is specified by **code**, the column value represents the character position in the field contents, that is, the position in the string that can be returned by ASCGET.

Description

Returns the position of the alphanumeric cursor; that is, the cursor position as set by ASFCUR, ASREAD, or default. If ASQCUR is used for devices on which the alphanumeric cursor is also used as the graphics cursor (for example, a 3279), it does not return the position of the cursor as set by a GSREAD call, but retains the last setting for the cursor as set by ASFCUR or ASREAD.

Principal errors

```
ADM0218 E  CODE n IS INVALID
```

ASQFLD

Function

To query field attributes.

ASQFLD (code, n, count, array)	
APL code	418
GDDM RCP code	X'0C080A00' (201853440)

Parameters

code *(specified by user) (fullword integer)*
Specifies how the fields to be queried are identified. Possible values are:

0 single field (number n).
A single field is to be queried; the field number is given by **n**.

1 n user-specified field numbers.
n fields are to be queried; their identification numbers must already be in the field identifier elements of the **array** parameter.

If a specified field is undefined, a warning message is issued and some of the attribute elements are undefined; see the **field identifier** element description in the **array** parameter, below.

2 n sequential field numbers (1 through n).
n fields are to be queried; the field numbers are **1 through n**, sequentially.

Definitions are returned in that order in **array**. (The maximum field number can be obtained from ASQMAX).

Field identifiers for existing fields are set; if a definition for field **i** does not exist, the field identifier for the **i**th definition in the array is set to zero.

3 n defined field numbers.
n fields are to be queried; the field numbers are those of the first **n defined** fields.

Definitions for the first **n** defined fields are returned consecutively in field number order in **array**. (The number of defined fields can be obtained from ASQMAX.)

If the number of defined fields is less than **n**, the field identifiers for the remaining definitions in **array** are set to zero.

n *(specified by user) (fullword integer)*
For **code** = 0, **n** specifies the field number for the single definition to be returned.

For other values of **code**, **n** specifies the number of field definitions to be returned. In this case, **n = 0** indicates that no field definitions are to be returned.

count *(specified by user) (fullword integer)*
The number of attributes listed for each field. This must be between 1 and the maximum element number defined in **array**.

array *(returned by GDDM) (an array of fullword integers)*
An array of field definitions, each of which may contain the following elements. The number of elements actually present is limited by **count**; thus, the array contains **n** definitions (or 1 if **code** = 0), each consisting of **count** elements. All of the elements for definition **i** precede those for definition **i+1**.

1 Field identifier
The number of the field. Zero indicates no field definition. If the field number is undefined or zero: the row, column, depth, and width elements, where present, are set to zero; all other elements are undefined.
2 Row
The row for the top left-hand corner of the field. Rows are numbered from top to bottom of the page, starting from 1. Zero is returned for a field that does not exist.
3 Column
The column for the top left-hand corner of the field. Columns are numbered from left to right across the page, starting from 1. Zero is returned for a field that does not exist.
4 Depth
The number of rows that the field occupies. Zero is returned for a field that does not exist.
5 Width
The number of columns that the field occupies. Zero is returned for a field that does not exist.

6 Type
See ASFTYP.

7 Intensity
See ASFINT.

8 Color
See ASFCOL.

9 Primary symbol set
See ASFPSS.

10 Highlight
See ASFHLT.

11 End
See ASFEND.

12 Nulls
See ASFOUT.

13 Blanks
See ASFIN.

14 Table number
See ASFTRN.

15 Transparency
See ASFTRA.

16 Enable/disable shift-control codes
See ASFSEN.

17 Field outlining
See ASFBDY.

Description

Returns the general attributes of one or more fields. The values of **code** and **n** together determine which and how many field definitions are returned. The **count** parameter specifies how many attributes are to be returned for each definition.

A field number of zero, whether specified by the user or determined by GDDM, indicates no field definition. In this case, some of the attribute elements are undefined; see the description of the **field-id** parameter.

Principal errors

```
ADM0207 E  NUMBER OF FIELDS n IS NEGATIVE
ADM0208 E  COUNT n IS INVALID
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0213 W  FIELD n DOES NOT EXIST
ADM0218 E  CODE n IS INVALID
```

ASQHLT

Function

To query character highlights for a field.

ASQHLT	**(field-id, length, highlight-string)**
APL code	428
GDDM RCP code	X'0C080900' (201853184)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field containing the required character attributes.

length *(specified by user) (fullword integer)*
The number of character attributes to be returned. This number of bytes is returned, regardless of the size of the field.

highlight-string *(returned by GDDM) (character)*
The new highlighting for each character position within the field (starting at the top left-hand corner and working from left to right for each row of the field). Each byte is one of the following (characters are first given in the EBCDIC form, then in hexadecimal):

Blank	X'40'	Inherit the highlighting set by ASFHLT (the default).
1	X'F1'	Blink.
2	X'F2'	Reverse video.
4	X'F4'	Underscore.

Description

Returns the highlighting attributes for a given length of the specified field.

If the field is defined as “mixed-without-position,” (see ASFSEN), default highlight attributes are inserted, corresponding to the positions where SO/SI (shift-out/shift-in) control codes are inserted in the string returned by ASCGET. Therefore, the length of the returned string is greater than that displayed on the device by the number of inserted control codes. The length necessary to retrieve the field contents is obtained by means of the **i/p length** parameter of the ASQLEN call or the ASQMOD call.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASQLEN

Function

To query length of field contents.

ASQLEN	**(field-id, length, i/p-length, scr-length)**
APL code	443
GDDM RCP code	X'0C081800' (201857024)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field whose lengths are queried.

length *(returned by GDDM) (fullword integer)*
The length (in bytes for mixed or not-mixed alphanumeric field or in double characters for DBCS field) of the corresponding field in **field-id**; the length value is zero for a **field-id** of zero.

i/p-length *(returned by GDDM) (fullword integer)*
The not-null length (in bytes for mixed or not-mixed alphanumeric field or in double characters for DBCS field) of the corresponding field in **field-id**; the length value is zero for a **field-id** of zero. The not-null length is the length up to and including the last not-null character or double character in the field.

If the field is defined as “mixed without position,” this returned length includes SO/SI (shift-out/shift-in) control codes, which are inserted to delimit the DBCS portions of the string returned by ASCGET. Therefore, it indicates the amount of storage that the application must provide to retrieve the field contents.

scr-length *(returned by GDDM) (fullword integer)*
Indicates the same length as **i/p-length** with the following exception:

If the field is defined as “mixed without position,” this returned length excludes the SO/SI (shift-out/shift-in) control codes, that are returned to delimit the DBCS portions of the string returned by ASCGET. Therefore, it indicates the not-null length displayed on the screen.

Description

Returns the real and effective lengths of the specified field.

The effective lengths (**i/p-length** and **scr-length**) include the field contents up to the last not-null character.

The **i/p length** is the length of the string returned by ASCGET. The **scr-length** is the length of the field contents displayed on the screen.

If the field is defined as “mixed without position,” the **i/p length** may be larger than the real field length and the **scr-length** because of the insertion of SO/SI (shift-out/shift-in) control codes in fields containing both EBCDIC and DBCS (used for Kanji and Hangeul) characters.

Note: This call should be issued before issuing ASCGET, ASQCOL, ASQHLT, and ASQSS calls to ensure that the length of the field contents is correct if the field is defined as “mixed without position.”

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
```

ASQMAX

Function

To query the number of fields.

ASQMAX (n-fields, max-field)	
APL code	419
GDDM RCP code	X'0C080E00' (201854464)

Parameters

n-fields *(returned by GDDM) (full word integer)*
The number of fields currently defined.

max-field *(returned by GDDM) (full word integer)*
The maximum field number currently defined. If there are no fields currently defined, this value is zero.

Description

Returns the number and range of field numbers on the current page.

Principal errors

None.

ASQMOD

Function

To query modified fields.

ASQMOD (count, field-ids, lengths, i/p-lengths)	
APL code	420
GDDM RCP code	X'0C080B00' (201853696)

Parameters

count *(specified by user) (fullword integer)*
The number of fields to be queried.

field-ids *(returned by GDDM) (an array of fullword integers)*
An array of field numbers. The numbers of modified fields are returned in this order:

1. Any modified light-pen fields
2. Any other modified fields
3. Zero for all other entries.

Within these categories, fields are returned in order of field identifier.

lengths *(returned by GDDM) (an array of fullword integers)*
An array of length values. Each value indicates the length (in characters or dual characters as appropriate) of the corresponding field in **field-ids**; the **lengths** value is zero for a field-id of zero.

i/p-lengths *(returned by GDDM) (an array of fullword integers)*
An array of length values. Each value indicates the not-null length (in single bytes for mixed or not-mixed alphanumeric fields or in double bytes for DBCS fields) of the corresponding field in **field-ids**; the length value is zero for a field-id of zero. The not-null length is the length up to and including the last not-null character or dual character in the

field (that is, nulls that precede not-nulls are treated as not-nulls for this operation).

If the field is defined as “mixed without position,” this returned length includes SO/SI (shift-out/shift-in) control codes that are inserted to delimit DBCS portions of the string returned by ASCGET. Therefore, it indicates the amount of storage that the application must provide to retrieve the field contents.

Description

Returns the real and effective lengths of modified fields for the current page. The effective length includes the last not-null character.

When modified fields are returned, they become unmodified. For example, if ASREAD indicates that there are ten modified fields, and subsequently two calls are made to ASQMOD, each requesting seven fields, the first returns seven, and the second returns the remaining three (and four zero entries), after which there are no modified fields on the current page.

Notes:

1. Calls to ASREAD, ASFMOD, ASQMOD, and GSREAD affect the list of modified fields.
2. Calls to ASQNMF do not affect the list of modified fields.

Principal errors

```
ADM0220 E  COUNT n IS NEGATIVE
```

ASQNMF

Function

To query the number of modified fields.

ASQNMF	**(n-fields)**
APL code	435
GDDM RCP code	X'0C080E01' (201854465)

Parameters

n-fields *(returned by GDDM) (fullword integer)*
The number of modified alphanumeric fields.

Description

Returns the number of modified alphanumeric fields on the current page.

This call does not affect the *number* of modified alphanumeric fields.

Principal errors

None.

ASQSS

Function

To query character symbol sets for a field.

ASQSS	**(field-id, length, symbol-set-id-string)**
APL code	429
GDDM RCP code	X'0C080902' (201853186)

Parameters

field-id *(specified by user) (fullword integer)*
The number of the field containing the required character identifiers.

length *(specified by user) (fullword integer)*
The number of character identifiers to be returned. This number of bytes is returned, regardless of the size of the field.

symbol-set-id-string *(returned by GDDM) (character)*
The new symbol-set identifiers for each character position within the field (starting at the top left-hand corner and working left to right for each row of the field). Each character must be one of these:

X'00' or X'40'	Inherit the symbol-set identifier set by ASFPSS (the default)
X'01' through X'03'	Loadable symbol sets (3800-system printer)
X'41' through X'DF'	Loadable symbol sets (3270-family devices)
X'F1'	Alternative nonloadable symbol set (3270-family devices).

Description

Returns the symbol-set identifiers for a given length of the specified field.

If the field is defined as “mixed-without-position” (see ASFSEN), default symbol-set attributes are inserted, corresponding to the positions where SO/SI (shift-out/shift-in) control codes are inserted in the string returned by ASCGET. Therefore, the length of the returned string is greater than that displayed on the device by the number of inserted control codes. The length necessary to retrieve the field contents is obtained by means of the **i/p length** parameter of the ASQLEN call or the ASQMOD call.

Principal errors

```
ADM0201 E  FIELD IDENTIFIER n IS ZERO OR NEGATIVE
ADM0212 E  FIELD n DOES NOT EXIST
ADM0214 E  FIELD n1 STRING LENGTH n2 IS NEGATIVE
ADM0217 E  FIELD          n          CONTAINS          A
           {CHARACTER|DUAL-CHARACTER} STRING
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
ADM0227 W  FIELD n CANNOT BE SET TO MIXED STATUS
```

ASRATT

Function

To define field attributes.

ASRATT	**(n-fields, count, array)**
APL code	417
GDDM RCP code	X'0C080802' (201852930)

Parameters

n-fields *(specified by user) (fullword integer)*
The number of fields to be redefined. This may be zero.

count *(specified by user) (fullword integer)*
The number of attributes provided for each field; it must be in the range 5 through 17.

array *(specified by user) (an array of fullword integers)*
An array of field definitions, each of which may contain the following elements. The number of elements actually present is limited by **count**; thus, the array contains **n-fields** definitions, each consisting of **count** elements. All of the elements for definition **i** precede those for definition **i+1**.

1 Field-id	The number of the field. Zero indicates no field definition.
2 Row	Ignored.
3 Column	Ignored.
4 Depth	Ignored.
5 Width	Ignored.
6 Type	See ASFTYP.
7 Intensity	See ASFINT.
8 Color	See ASFCOL.
9 Primary symbol set	See ASFPSS.
10 Highlight	See ASFHLT.
11 End	See ASFEND.
12 Nulls	See ASFOUT.
13 Blanks	See ASFIN.
14 Table number	See ASFTRN.
15 Transparency	See ASFTRA.
16 Enable/disable shift-control codes	See ASFSEN.
17 Field outlining	See ASFBDY.

Description

Defines field attributes for the current page. Existing fields that are not referred to by this call remain unchanged. Existing fields that are referred to by this call have their attributes redefined.

The number of attributes specified for each field is determined by **count**.

Principal errors

```
ADM0203 W  FIELD   n   LIGHT-PEN   ATTRIBUTE   MAY   BE
           INEFFECTIVE
ADM0204 W  FIELD n IS TOO SMALL TO BE A LIGHT PEN FIELD
ADM0207 E  NUMBER OF FIELDS n IS NEGATIVE
ADM0208 E  COUNT n IS INVALID
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0211 W  {FIELD  n1}  ATTRIBUTE  {n2}  VALUE  n3  IS
           INVALID
ADM0212 E  FIELD n DOES NOT EXIST
ADM0224 W  FIELD n1 WIDTH n2 MUST BE EVEN FOR A DBCS
           FIELD
ADM0225 E  FIELD n ALREADY EXISTS
```

ASREAD

Function

To perform device output and input.

ASREAD	**(attype, attval, count)**
APL code	101
GDDM RCP code	X'0C100000' (202375168)

Parameters

attype *(returned by GDDM) (fullword integer)*
The type of attention interrupt received. Possible values are:

0 ENTER key

1 PF key

2 Light pen
Changes to all other types of field are lost.

3 Magnetic stripe (badge) reader
The magnetic stripe record field (the one containing the cursor) has special characters from the magnetic stripe reader replaced by blanks if the operation was a success. This operation causes a refreshment of the display at the next interaction with the display.

4 PA key
Changes to all fields are lost.

5 CLEAR key
This operation causes a refreshment of the display at the next interaction with it. Changes to all fields are lost. If mapping, the mapgroup option may say "Refresh Screen" and repeat the ASREAD call automatically instead of returning **attype=5**.

6 Other
The interrupt received from the device does not belong to any of the defined categories; **attval** and **count** values are undefined.

7 Output-only device
The primary device only accepts output, and does not return input; **attval** and **count** values are undefined. A warning message is issued to emphasize this situation.

10 The buttons on a mouse, tablet or four-button cursor device
This is only returned if the mouse or tablet is enabled for use with interactive graphics functions. An FSENAB call to enable graphics input must have been issued for graphics input to be available with ASREAD.

attval *(returned by GDDM) (fullword integer)*
The value, if any, associated with **attype**, as follows:

1 PF key number
3 0 = success, 1 = failure
4 PA key number
10 Button number.

count *(returned by GDDM) (fullword integer)*
The number of modified fields after execution of the function.

If the GDDM page is a mapped page, **count** returns the number of modified mapped fields in that page, even if there are modified procedural alphanumeric fields in that page. If the GDDM page is not a mapped page, **count** returns the number of modified procedural alphanumeric fields in that page.

If the GDDM page contains high-performance alphanumerics, **count** is always 0.

If **attype**=2, **count** contains the number of modified light-pen fields.

If multiple partitions (real or emulated) are in use, the operator can move the cursor from one partition into another during the ASREAD call. If this happens, ASREAD returns the number of modified fields in the **new** current partition; that is, the partition that now contains the cursor. If real partitions are in use, only data from the active partition is input. The terminal user's changes in other partitions are not available.

Description

Performs all outstanding output, and for an interactive device, requests input from the device. The ASREAD call is completed when an attention interrupt from the device is received and the attention information is returned to the caller.

Restrictions exist on the use of some keys, depending on the operating environment. For further information, see the *GDDM Base Application Programming Guide*.

In a windowing environment, it is possible for an attention interrupt to be pending, in which case the call is completed immediately.

A pending attention interrupt is created as follows; the device for which the ASREAD call is to be performed must be associated with an operator window of a windowed device, and the operator window must have a zero coordination exit; for more information on coordination exits, refer to Chapter 22, "Special-purpose programming in GDDM" on page 431.

When the WSIO call is issued for the windowed device, and the end user interacts with the operator window, a pending attention interrupt is created. If an attention interrupt is already pending, it is replaced by the new pending attention interrupt. The pending attention interrupt is removed by the ASREAD call, or by any other input/output function.

If the primary device is a queued printer (family-2), a system printer (family-3), or a page printer (family-4), the action of ASREAD is as described under FSCOPY. In this case, the function is not allowed if graphics retrieval is in progress; see GSGET.

By default, only procedural, mapped, and high-performance alphanumeric input is available with an ASREAD call. However, an FSENAB call can be used to enable graphics or image input to be entered in response to the ASREAD call.

For a plotter, pressing the Clear key on the attached workstation while ASREAD is executing cancels the output.

When running in CICS transaction-independent pseudoconversational mode, the ASREAD call only receives input from the device. It never sends output to the device.

For CICS transaction-dependent pseudoconversational applications, when the pseudoconversational mode is in use, the first ASREAD call in a sequence only performs its input function; it does not perform output. Subsequent calls to ASREAD work in the usual way.

However, there are two exceptions to this rule:

1. The mapgroup requests automatic processing of the CLEAR key. In this case, the ASREAD call performs in the usual way, that is, output is bypassed and the input data is processed, after which Mapping signals a screen refresh.

The result of this will be as if a second ASREAD call had occurred, that is, the screen contents are output again, and the transaction waits for input.

2. A GDDM line output error message occurs before an ASREAD call. In this case, the screen contents have been destroyed and therefore, for GDDM to continue, the screen contents have to be recreated.

   Therefore, the ASREAD call performs in the usual way, that is, output is performed, a “wait for input” state is entered, and the transaction becomes Conversational for this invocation.

Principal errors

```
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0233 W  SYMBOL SET IS NOT LOADED
ADM0270 E  SCREEN FORMAT ERROR
ADM0273 W  PS OVERFLOW
ADM0275 W  GRAPHICS {(IMAGE) }CANNOT BE SHOWN. REASON
           CODE n
ADM0276 W  DEVICE IS OUTPUT ONLY
ADM0498 E  PRINT TERMINATED. RETURN CODE X'xxxxxx' FROM
           DEVICE
ADM0909 W  NO GRAPHICS FIELD
ADM0911 W  COMPOSED TEXT BLOCK OVERLAPS PAGE BOUNDARY.
           TEXT IGNORED
ADM0920 E  CLEAR KEY PRESSED. PLOTTING IS TERMINATED
ADM2850 W  DBCS CHARACTERS IN AN SBCS FIELD ARE
           CONVERTED TO BLANKS
ADM2864 W  PICTURE IS TOO LARGE FOR 5080 DISPLAY LIST
           BUFFER
ADM3004 E  FIELD LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3005 E  DATA BUFFER n1, ERROR n2 AT INDEX n3
ADM3010 E  BUNDLE LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3173 W  GRAPHICS CANNOT BE SHOWN.  CELL WIDTH OR
           DEPTH EXCEEDS LOADABLE LIMIT
ADM3178 W  PATTERNS CANNOT BE SENT TO DEVICE.  AREA
           SHADING MAY BE INCORRECT
ADM3179 W  IMAGE CANNOT BE SHOWN. REASON CODE n
ADM3281 W  GRAPHICS MAY BE VISIBLE WITHIN OPAQUE
           ALPHANUMERIC FIELDS
ADM3282 W  AMOUNT OF DATA EXCEEDS THE STORAGE CAPACITY
           OF THE DEVICE
```

ASRFMT

Function

To define multiple fields without deleting existing fields.

ASRFMT	**(n-fields, count, array)**
APL code	405
GDDM RCP code	X'0C080800' (201852928)

Parameters

n-fields *(specified by user) (fullword integer)*
The number of fields to be defined or redefined. This can be zero.

count *(specified by user) (fullword integer)*
The number of attributes provided for each field; it must be in the range 5 through 17.

array *(specified by user) (an array of fullword integers)*
An array of field definitions, each of which may contain the following elements. The number of elements actually present is limited by **count**; therefore, the array contains **n-fields** definitions, each consisting of **count** elements. All of the elements for definition **i** precede those for definition **i+1**.

1 Field-id
The number of the field. Zero indicates no field definition.
2 Row
The row for the top left-hand corner of the field. Rows are numbered from top to bottom of the page, starting with 1. Zero indicates that the field is to be deleted.
3 Column
The column for the top left-hand corner of the field. Columns are numbered from left-to-right across the page, starting with 1. Zero indicates that the field is to be deleted.
4 Depth
The number of rows that the field occupies. This must be such that the field does not extend beyond the bottom of the page. Zero indicates that the field is to be deleted. For a mixed field (see ASFSEN), the depth of the field cannot be greater than 1 unless the width of the field is equal to the width of the device.
5 Width
The number of columns that the field occupies. For a field, the depth of which is greater than 1, the width must not extend beyond the right side of the page. If a field is intended for use with a double-character primary symbol set (see ASFPSS), then the width must be even. Zero indicates that the field is to be deleted.

6 Type
See ASFTYP.

7 Intensity
See ASFINT.

8 Color
See ASFCOL.

9 Primary symbol set
See ASFPSS.

10 Highlight
See ASFHLT.

11 End
See ASFEND.

12 Nulls
See ASFOUT.

13 Blanks
See ASFIN.

14 Table number
See ASFTRN.

15 Transparency
See ASFTRA.

16 Enable/disable shift-control codes
See ASFSEN.

17 Field outlining
See ASFBDY.

Description

Defines alphanumeric fields for the current page. Existing fields that are not referred to by the ASRFMT call remain unchanged. To define a complete new set of fields for the page, deleting any that were previously defined, use ASDFMT. Existing fields that are referred to by the ASRFMT call are first deleted, and then created according to the new definitions. If the cursor was positioned within any of the deleted fields, the cursor position is reset to the top left-hand corner of the page window.

The number of attributes specified for each field is determined by **count**. Any attributes that are not specified are chosen according to their default values; see ASDFLT.

Principal errors

```
ADM0203 W  FIELD n LIGHT-PEN ATTRIBUTE MAY BE
           INEFFECTIVE
ADM0204 W  FIELD n IS TOO SMALL TO BE A LIGHT PEN FIELD
ADM0205 E  FIELD n POSITION IS INVALID
ADM0206 E  ALPHANUMERIC FIELD a1 OVERLAPS ALPHANUMERIC
           FIELD a2
ADM0207 E  NUMBER OF FIELDS n IS NEGATIVE
ADM0208 E  COUNT n IS INVALID
ADM0209 E  FIELD IDENTIFIER n IS NEGATIVE
ADM0211 W  {FIELD n1} ATTRIBUTE {n2} VALUE n3 IS
           INVALID
ADM0224 W  FIELD n1 WIDTH n2 MUST BE EVEN FOR A DBCS
           FIELD
ADM0225 E  FIELD n ALREADY EXISTS
ADM0226 W  FIELD n CANNOT BE BOTH DBCS AND MIXED
ADM0227 W  FIELD n CANNOT BE SET TO MIXED STATUS
ADM3012 E  CURRENT PAGE HAS HIGH PERFORMANCE
           ALPHANUMERICS
```

ASTYPE

Function

To override alphanumeric character-code assignments.

Note: Code page conversion functions are recommended in preference to the ASTYPE call. Support for the call is continued for compatibility reasons, and to support APL code page 293.

ASTYPE (type)	
APL code	111
GDDM RCP code	X'0C081300' (201855744)

Parameters

type (specified by user) (fullword integer)
The new type number. Possible values are:

0 Translation from application code page to device code page.

1 No translation, except the removal of invalid characters.

2 EBCDIC: GDDM assumes the terminal is an EBCDIC terminal and finds a matching type number from the EBCDIC group.

3 Kanji/Hangeul: GDDM assumes the terminal is a Kanji/Hangeul terminal and finds a matching type number from the Kanji/Hangeul group.

n Select type “n,” where “n” is from the range supplied for your installation, overriding the default.

Description

Overrides alphanumeric character-code assignments by defining a type that is associated with a set of translation tables to be used on alphanumeric fields.

GDDM uses code conversion tables to ensure that a hexadecimal character code appears as the same character on all devices, thus giving programs a degree of device-independence. The meanings assumed for the hexadecimal values are shown in Figure 1 on page 49, Figure 2 on page 49, and Figure 3 on page 50.

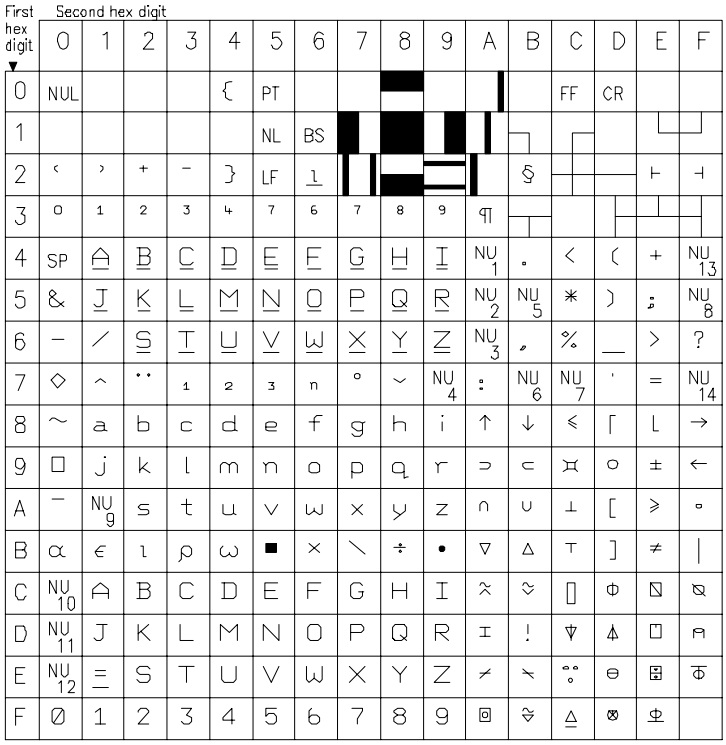


Figure 1. GDDM default EBCDIC character codes (code page 00351)

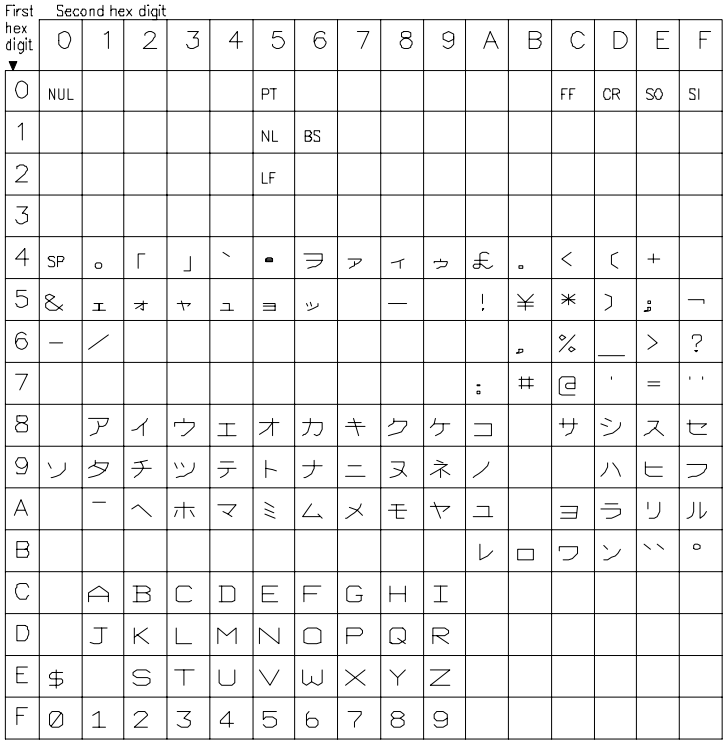
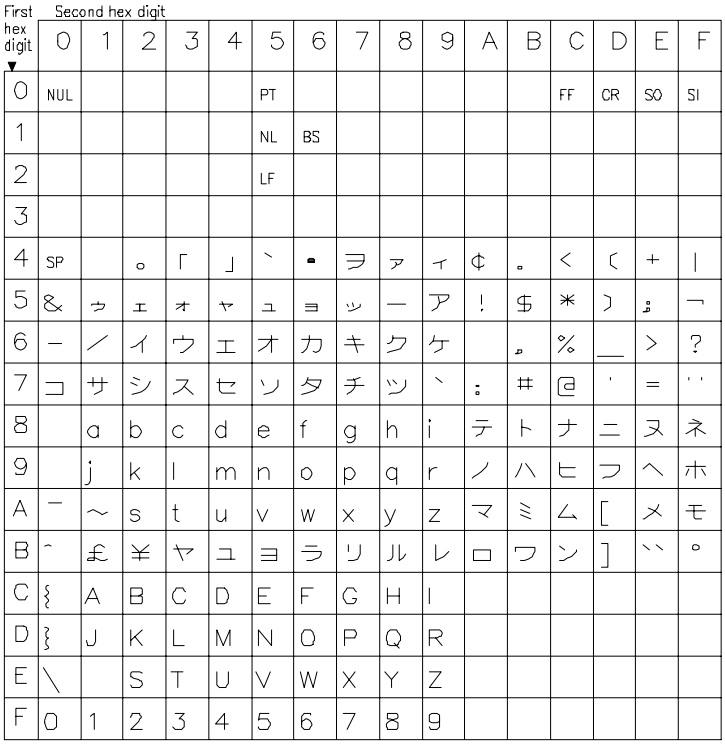


Figure 2. Katakana character codes (Tables 32772, 32792, and 32793) (code page 00290)



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 3. Japan (Latin) extended character codes (code page 01027)

The sets of translation tables are associated with type numbers that, in turn, are associated with device characteristics. It is the type number that is specified in the ASTYPE call.

At initialization, GDDM discovers the type of device in use and, by selecting a type number, specifies a set of translation tables. The ASTYPE call can be used either to specify a particular device type, or to specify the assumptions that GDDM uses to select a device type – for example, to assume that the device is a Kanji/Hangeul device.

The translation type you specify takes effect when data is transmitted to the device. To make it effective for data that was transmitted previously, call FSREST to cause retransmission of all the data to the device.

The ASTYPE call is a specialized call and should not be used in normal programming. A full description of the translation tables and associated type numbers is in the *GDDM System Customization and Administration* book.

ASTYPE is a tool for system programmers and special situations. Usually, ASTYPE should be left to the default. An example of the use of ASTYPE would be to allow a Kanji/Hangeul terminal to be used with APL. ASTYPE(2) would be specified before the use of APL and ASTYPE(3) before the use of Kanji/Hangeul.

The effect of a type value of 0, which is the default for the ASTYPE parameter, causes translation from the application code page to the device code page. However, the results, under Version 2 Release 2 of GDDM, will be the same as under earlier releases if ADMDATRN was not modified in the earlier releases and if no CECP application code page is explicitly specified.

Using an application code page of 00351 is equivalent to executing an ASTYPE call with a type value of 0 in an earlier release.

Using an application code page of 00290 or 1027 (GDDM Katakana) is equivalent to executing an ASTYPE call with a type value of 3 in an earlier release.

Executing an ASTYPE call with a type value of 293 will select an application code page of 293 (suitable for use with APL) on terminals that support it.

Principal errors

```
ADM0216 E  TYPE n IS INVALID
```

CDPU

Function

To control the printing of Composite Documents.

CDPU	**(CD-count, CD-name, opt-cnt, opt-list)**
APL code	1196
GDDM RCP code	X'40000000' (1073741824)

Parameters

CD-count *(specified by user) (fullword integer)*
The number of 8-byte name-parts in **CD-name**. It must be 1 for MVS, VSE, and CICS, and 3 for CMS.

CD-name *(specified by user) (array of 8-byte character tokens)*
This list gives the name-parts, that constitute the name, by which the composite document interface file is known to the underlying subsystem.

CICS Temporary queue name

VSE Batch DLBL file name (7 characters)

TSO DD name

CMS Filename Filetype Filemode
Either or both of the filetype and filemode can be omitted, in which case default values of LISTCDP and "*" respectively are used.

The number of name-parts that can be specified is subsystem-dependent. GDDM left-justifies each name part.

opt-cnt *(specified by user) (fullword integer)*
The number of fullwords in **opt-list**. Zero can be specified to indicate that **opt-list** is empty and is not to be inspected.

opt-list *(specified by user) (an array of fullword integers)*
Printing options, ignored in cases where the target printer does not support the corresponding facility. A default value is used for an element that is missing, or has a value of zero. Each element of the array can be one of:

1 Number of uncollated copies. The default is one copy. This option applies only to family-1 IPDS printers.

2 Duplex control:
- **1** Simplex – the default.
- **2** Normal duplex.
- **3** Tumble duplex.

3 View control.
This option applies on CICS, TSO, and CMS only, and then only when the document is viewed.
- **0** View the entire document.
- +**n** Draw page n with images included.
- –**n** Draw page n with images indicated by boxes.

If +**n** or –**n** is specified, the CDPU call creates a GDDM page containing the required output, but leaves the input or output for the application program to process.

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 been completed successfully (see note below).
- **4** The input file is deleted only if processing has not been completed successfully.

Note: Where "successfully" means there have been no messages, except informational ones generated while the document is being processed.

Description

Print or view a composite document. The existing contents of the page, if any, are deleted by a call to CDPU.

If an application calls the CDPU with the view control parameter set to a non-zero value, the application can control how the document is browsed. The CDPU creates a GDDM page containing the specified document page, but does no input or output. The application must issue its own ASREAD (or other input/output call) and interpret the returned values. In addition the application can:

- Define a graphics field for the document page to be shown in. The default is a field covering the whole screen.
- Display instructions to the user.
- Test for requests for document pages beyond the document end.

Note: If the IMGINIT,WHITE procopt is active, the whole screen will be white, not just that part of the screen which represents the page.

For more information on the printing of Composite Documents, refer to the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM2758 E  COMPOSITE DOCUMENTS ARE NOT SUPPORTED FOR
           THIS DEVICE
ADM2760 E  FIELD NOT CONVERTED
ADM2775 E  INVALID COMPOSITE DOCUMENT NAME COUNT, n
ADM2776 W  INVALID NUMBER OF OPTIONS, n
ADM2777 W  INVALID PARAMETER.  COPY COUNT n IS NEGATIVE
ADM2778 W  INVALID PARAMETER.  DUPLEX CONTROL VALUE n
           IS INVALID
ADM2779 W  COMPOSITE DOCUMENT CONTAINS ERRORS
ADM2780 W  INVALID ACTIVE ENVIRONMENT GROUP ON PAGE n
ADM2782 W  GRAPHIC POSITION IS OFF PAGE
ADM2795 W  DELETION CONTROL VALUE n IS INVALID - FILE
           NOT DELETED
```

CGLOAD

Function

| To import (load) a picture from a Computer Graphics Metafile
| (CGM).

CGLOAD	**(cgm-count,cgm-name,profile,opt-count,opt-array, seg-count,desc-len1,descriptor1,ret-len1,desc-len2, descriptor2,ret-len2)**
APL code	669
GDDM RCP code:	X'0C0C1F00' (202120960)

Parameters

cgm-count *(specified by user) (fullword integer)*
The number of 8-byte name parts in **cgm-name**. Valid range is 1 through 3 under VM, 1 through 7 otherwise.

cgm-name *(specified by user) (array of 8-byte character tokens)*
The name-parts that constitute the name by which the CGM file is known to the underlying subsystem. GDDM left-justifies each name part. The number of name parts that can be specified is subsystem-dependent. Under VM/CMS, there are three elements in the array, defined as follows:

1 The CMS filename of the file. Can be any valid CMS file name.
2 The CMS filetype of the file. Can be any valid CMS file type. If **cgm-count**<2, a filetype of "CGM" is assumed.
3 The CMS filemode of the file. Can be any valid CMS filemode, or "*," indicating the first occurrence of the file in the CMS search order. if **cgm-count**<3, a filemode of "*" is assumed.

Under TSO or MVS/Batch, this specifies either

- An allocated ddname, or
- A fully qualified data set name, such as 'CGM.PICTURE.NO43', or
- If the file is part of a partitioned data set, a name and member name such as 'CGM.PICTURE.EXTRACTS(NO43)', or
- If the quotation marks are omitted and the parameter is not a ddname, a name to which a data set qualifier is added to the front in the usual TSO way.

If the name exceeds eight characters in length, 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
```

it looks like this:

cgm-name(1) =	'	a	a	a	a	.	b	b
cgm-name(2) =	b	b	.	c	c	c	'	

profile *(specified by user) (8-byte character string)*
The name (left-justified) of the CGM profile. If this parameter is not specified, the default is ADM. However, on CMS, the default is the CGM filetype specified.

opt-count *(specified by user) (fullword integer)*
The number of elements specified in the **opt-array** parameter.

opt-array *(specified by user) (an array of fullword integers)*
Specifies how CGLOAD is to restore a picture from the CGM file. The parameter has six elements. If an element is not specified, the action taken is the same as if it had been specified as 0.

1 – picture-number
The sequence number within the CGM of the picture to be loaded. CGM source files may contain more than one picture and in this case the pictures are considered to be numbered sequentially starting with 1. **picture-number** is defined as follows:
–**1** Loads all pictures in the input file
0 the default, as specified in the profile. If the picture number is not specified in the conversion profile or is specified as zero, –1 is used.
≥**1** Specifies the number of the (single) picture to be loaded from the input file
If more than one picture is loaded, the individual pictures are not identifiable in terms of segment numbers.

2 – seg-base
Identifies which segment is used as the starting point for loading into the GDDM page. Segments are created using the next unused segments greater than or equal to **seg-base**. Allowed values are:
0 The default, as specified in the profile. If the seg-base is not specified in the conversion profile or is specified as zero, 1 is used.
≥**1** The CGM picture is loaded in unused segments with identifiers ≥**seg-base**.

3 – load-type
Specifies how the picture is to be restored from the CGM. The options are:
0 The default, as specified in the profile. If the load-type is not specified in the conversion profile or is specified as zero, 2 is used.
1 The picture is restored without transformation, using the page's current window and viewport coordinate system. To restore the saved data satisfactorily, a window coordinate system that corresponds to the coordinates used in the CGM must be defined using a GSWIN call or a GSUWIN call.
2 The picture space of the CGM picture is accommodated within the current viewport, preserving the aspect ratio that the picture had when it was

saved. Any primitives outside the picture space of the CGM picture may be lost.

3 Reserved.

4 At the time of the CGLOAD, the shape of the picture held in the file to be loaded is used to define the picture space, and the rest of the graphics hierarchy is defaulted. The load then proceeds as for **load-type = 2**.

If the picture space has been defined (or defaulted), this load-type is equivalent to **load-type = 2**.

4 – symbol-set

Specifies the action to be taken when loading a CGM picture that contains fonts. Note that GDDM loads symbol sets to emulate CGM fonts. The options are:

0 The default, as specified in the profile. If the symbol-set is not specified in the conversion profile or is specified as zero, 2 is used.

1 GDDM loads the symbol sets corresponding to the fonts in the CGM input file after mapping according to the profile. They are loaded with the identifiers specified in the profile. Any symbol sets that are already loaded are overwritten (unless the symbol set already loaded is the one that is actually required; that is, has the same name, type, and LCID, in which case it is not loaded again).

This option should be used to ensure that the values of the GDF_FONT_INDEXs specified in the profile are used as LCIDs without change.

2 Same action as in **1** except that no existing symbol sets are overwritten. Symbol sets which, if loaded, would clash with the LCID of one already loaded are moved to new LCIDs. The largest unused identifiers are used for this purpose. References to fonts in the CGM file are mapped to the new LCIDs if they have been moved, not to the LCID corresponding to the GDF_FONT_INDEX in the profile.

If there are insufficient unused identifiers for all the symbol sets to be loaded, the device default symbol set is used.

This option should be used where the application needs to ensure that no existing symbol sets or LCIDs are altered.

5 – seg-use

Specifies how many segments are to be created when loading a CGM picture. The options are:

0 The default, as specified in the profile. If the seg-use is not specified in the conversion profile or is specified as zero, 2 is used.

1 All CGM primitives are placed into separate segments. This option should be used where the resultant GDF file is intended to be edited.

2 The entire CGM picture (or pictures) loaded is placed in a single GDDM segment. This option should be used where the resultant GDF file is required to be small or where no subsequent editing of the file is required.

6 – code-page

Specifies the code page that was used by the CGM generating application. This option determines the way all text strings are translated. It is unlikely that the application writer using the CGLOAD call knows the required information; it is, therefore, recommended that this parameter is coded as zero, so that the information is obtained from the conversion profile. Allowed values are:

0 the default, as specified in the profile. If the code page is not specified in the conversion profile or is specified as zero, 850 is used.

437 Code page 437 (United States), as shown in the *GDDM System Customization and Administration* book.

819 Code page 819 (ISO/ANSI Multilingual), as shown in the *GDDM System Customization and Administration* book. This 8-bit ASCII code page contains **all** the characters in the GDDM CECP code pages.

850 Code page 850 (Multilingual), as shown in the *GDDM System Customization and Administration* book.

Other Code page specified in the user modifiable ADMDATRN module.

seg-count *(returned by GDDM) (fullword integer)*

Count of the number of segments created. The result depends on the setting of the **seg-use** parameter.

desc-len1 *(specified by user) (fullword integer)*

The maximum number of bytes to be returned in the **descriptor1** parameter.

descriptor1 *(returned by GDDM) (character string)*

A character string of at least **desc-len1** bytes to receive the text from the “Begin Metafile” element of the metafile followed by a X'15' byte followed by the text from the “Metafile Descriptor” element of the metafile. The number of bytes placed in **descriptor1** is returned in the **ret-len1** parameter.

ret-len1 *(returned by GDDM) (fullword integer)*

The number of bytes of information returned in **descriptor1**.

desc-len2 *(specified by user) (fullword integer)*

The maximum number of bytes to be returned in the **descriptor2** parameter.

descriptor2 *(returned by GDDM) (character string)*

A character string of at least **desc-len2** bytes to receive the picture name (from *begin picture* element) from the metafile. If more than one picture is being loaded (that is, **picture-number**=–1), the descriptors for these are concatenated together, with a x'15' byte placed in between each. The number of bytes placed in **descriptor2** is returned in the **ret-len2** parameter.

ret-len2 *(returned by GDDM) (fullword integer)*
The number of bytes of information returned in **descriptor2**.

Description

Retrieves one or all pictures from a Computer Graphics Metafile (CGM) on auxiliary storage and loads it into the current GDDM page. A segment must not be open when the CGLOAD call is made; CGLOAD does not leave any segment open. CGLOAD loads the CGM picture into the GDDM page, with the window, viewport, picture space and so on, under control of the **load-type** parameter. If GDDM detects an error, the picture reverts to its original state, without any changes applied.

CGLOAD loads primitives into their own unique segment, to assist with later editing of the picture, or can load the entire picture into one segment under control of the **seg-use** parameter.

The drawing defaults within the CGM file are incorporated into the segment data to be loaded. The current drawing defaults are not modified. Thus the loaded data reflects the drawing defaults in the CGM file, but CGLOAD does not affect any data currently displayed.

The way that CGM orders are converted to GDF orders and the limitations and restrictions of this process are described in Chapter 13, “Computer Graphics Metafiles” on page 323.

The corresponding CGM save function is provided by the CGSAVE call.

The permitted formats of the CGMs and conversion profiles handled by this call are defined in Chapter 13, “Computer Graphics Metafiles” on page 323.

CGLOAD is only available under the CMS, TSO and MVS/Batch environments. When CGLOAD is invoked in an unsupported environment, the error ADM3292 (listed below) is returned.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0128 W  SYMBOL SET n OPTION UNSUPPORTED
ADM0135 E  SYMBOL SET n TYPE UNSUPPORTED
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0173 E  STRING LENGTH n IS INVALID
ADM0182 W  INVALID CHARACTER CODE X'xx' IN STRING
ADM0232 E  CODE PAGE n IS NOT SUPPORTED
ADM0307 E  FILE 'a' NOT FOUND
ADM0313 E  FILE 'a' HAS INVALID RECORD CONTENT
ADM3157 E  SYMBOL SET IDENTIFIER n ALREADY IN USE
ADM3158 E  NO MATCH IN FONT FOR CODE PAGE INDEX ENTRY
ADM3270 W  MORE SYMBOL SETS THAN SYMBOL-SET IDENTIFIERS
ADM3290 E  INVALID CGM ORDER X'xx', LINE n1 OFFSET n2
ADM3291 E  INVALID VALUE n1 FOR ELEMENT n2 OF OPTION
           ARRAY
ADM3292 E  CGM FUNCTIONS ARE NOT SUPPORTED IN THIS
           ENVIRONMENT
ADM3293 E  ERROR AT ITEM n KEYWORD a1 IN FILE 'a2'
ADM3294 E  CGM ERROR CODE abbcc AT LINE n1 OFFSET n2
ADM3295 E  INVALID OR UNSUPPORTED CGM ORDER X'xx'
```

CGSAVE

Function

| Exports (saves) segments in a Computer Graphics Metafile
| (CGM).

CGSAVE	**(cgm-count, cgm-name,profile,seg-count, seg-array,opt-count,opt-array,desc-len1, descriptor1,desc-len2,descriptor2)**
APL code	670
GDDM RCP code	X'0C0C2000' (202121216)

Parameters

cgm-count *(specified by user) (fullword integer)*
The number of 8-byte name parts in the **cgm-name** parameter. Valid range is 1 through 3 under VM, 1 through 7 otherwise.

cgm-name *(specified by user) (array of 8-byte character tokens)*
The name-parts that constitute the name by which the CGM is to be known to the underlying subsystem. The format of the name-parts is subsystem-dependent and is defined in the same way as for the CGLOAD call, except that, under TSO, partitioned data set member names are not permitted. On CMS, the default filetype and filemode are “CGM” and “A1”.

profile *(specified by user) (8-byte character string)*
The name (left-justified) of the CGM profile file on auxiliary storage. The format of this parameter is identical to the parameter of the same name on CGLOAD. The profiles are intended for use in conversion in both directions.

If this parameter is not specified, the default is ADM. However, on CMS, the default is the CGM filetype specified.

seg-count *(specified by user) (fullword integer)*
The number of elements in the **seg-array** parameter.

seg-array *(specified by user) (an array of fullword integers)*
An array of segment identifiers. If the number of elements or the first segment identifier is zero, all the graphics data in the GDDM page is saved in the CGM file. If the number of elements is greater than zero, each identified segment is saved in the named file. The

segments are stored in the file in the order specified in **seg-array**. If this parameter has 0 after the first element, GDDM does not save any more elements beyond that point. Duplicate segment identifiers are not allowed.

opt-count *(specified by user) (fullword integer)*
The number of elements in the **opt-array** parameter.

opt-array *(specified by user) (an array of fullword integers)*
An array of control information. The parameter has two elements. If an element is not specified, the action taken is the same as if it had been specified as zero. The values that can be specified are:

1 – overwrite
Specifies whether the new CGM file can overwrite an existing CGM file of the same name on auxiliary storage. Possible values are:
0 Overwrite existing file. This is the default.
1 Do not overwrite existing file.

2 – code-page
Specifies the code page that the CGM receiving application is expecting. This option determines the way text strings are translated. It is unlikely that the application writer using the CGSAVE call knows the required information; it is, therefore, recommended that this parameter is coded as zero so that the information is obtained from the conversion profile. Allowed values are:

0 The default, as specified in the profile. If the code page is not specified in the conversion profile or is specified as zero, 850 is used.

437 Code page 437 (United States), as shown in the *GDDM System Customization and Administration* book.

819 Code page 819 (ISO/ANSI Multilingual), as shown in the *GDDM System Customization and Administration* book. This 8-bit ASCII code page contains **all** the characters in the GDDM CECP code pages.

850 Code page 850 (Multilingual), as shown in the *GDDM System Customization and Administration* book.

Other Code page specified in the user modifiable ADMDATRN module.

desc-len1 *(specified by user) (fullword integer)*
The number of characters in the **descriptor1** parameter.

descriptor1 *(specified by user) (array of character)*
A descriptive record, of up to 253 bytes, that is saved in the metafile as the text of the “Begin Metafile” element. The normal convention is that this descriptor is used for application-specific information, including an application identifier.

desc-len2 *(specified by user) (fullword integer)*
The number of characters in the **descriptor2** parameter.

descriptor2 *(specified by user) (array of character)*
A descriptive record, of up to 253 bytes, that is saved in the metafile as the picture name (in the *begin picture* element). The normal convention is that this descriptor is for use by the end user.

Description

Saves specific graphics segments, or all the graphics segments in the current GDDM page, into a CGM file on auxiliary storage. The segments or graphics data are saved in a file defined by the **cgm-name** parameter. No segment must be open when the CGSAVE call is issued.

The way that GDF orders are converted to CGM orders and the limitations and restrictions of this process are described in Chapter 13, “Computer Graphics Metafiles” on page 323.

The corresponding CGM load function is provided by the CGLOAD call.

The format of the conversion profiles used by, and the CGMs created by this call are defined in Chapter 13, “Computer Graphics Metafiles” on page 323.

The text of the “Metafile Descriptor” element of the metafile is generated automatically by CGSAVE and contains information about the date and time of the conversion and the name and version of the program performing the conversion.

CGSAVE is only available under the CMS, TSO, and MVS/Batch environments. When CGSAVE is invoked in an unsupported environment, the error ADM3292 (listed below) is returned.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0143 E  SEGMENT IDENTIFIER n IS DUPLICATE
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0161 E  GRAPHICS FIELD NOT DEFINED
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0232 E  CODE PAGE n IS NOT SUPPORTED
ADM0324 E  FILE 'a' ALREADY EXISTS
ADM3291 E  INVALID VALUE n1 FOR ELEMENT n2 OF OPTION
           ARRAY
ADM3292 E  CGM FUNCTIONS ARE NOT SUPPORTED IN THIS
           ENVIRONMENT
ADM3293 E  ERROR AT ITEM n KEYWORD a1 IN FILE 'a2'
ADM3294 E  CGM ERROR CODE abbcc AT LINE n1 OFFSET n2
```

DSCLS

Function

To close a device.

DSCLS (device-id, option)	
APL code	902
GDDM RCP code	X'0C000201' (201327105)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the device to be closed.

option *(specified by user) (fullword integer)*
Indicates an action to be performed at device closure. The meaning depends upon the device family:

Family-1 (3270-family) devices not in CICS pseudo-conversational mode

0 Erase the screen.
1 Do not erase the screen (see note 3).
2 Erase the screen and unlock the keyboard.
3 Do not erase the screen, but unlock the keyboard (see note 3).

Family-1 (3270-family) devices in CICS pseudo-conversational mode

0 Erase the screen. Also, unlock the keyboard and save any device data that has changed.
1 Do not erase the screen. Unlock the keyboard and save any device data that has changed.
2 Erase the screen, unlock the keyboard, and erase the saved device data.
3 Do not erase the screen, but unlock the keyboard and erase the saved device data.

Family-2 (queued printer files)

0 Cancel a print file (see note 4)
1 Enqueue a file for printing (see note 5).

Family-3 (system printer files)

0 Cancel a print file (see note 6).
1 Enqueue a file for printing (see note 7).

Family-4 (page printer files)

0 Cancel or delete the print file(s).
1 Enqueue or keep the print file(s).

Notes:

1. The **option** value is ignored for family-1 printers.
2. Under VM/CMS, if output has been directed to the virtual punch, the virtual punch is closed, and the punch file enqueued, whichever option is specified.
3. Under TSO, the screen is erased, whichever option value is specified.
4. Under IMS/VS, it is not possible to cancel a queued printer file, so it is enqueued, whichever option value is specified.
5. Under VM/CMS, if the DSOPEN procopt (INVKOPUV,YES) has been specified, the ***function*** of the GDDM Print Utility (ADMOPUV) is invoked, after which the print file is erased. Otherwise, a queued printer file is kept, but not enqueued, and an installation-defined ADMQPOST EXEC is invoked if present. For more information, see the *GDDM System Customization and Administration* book.
6. Under CICS/VS, IMS/VS, or TSO, it is not possible to cancel a system printer file, so it is enqueued, whichever option value is specified.
7. Under VM/CMS, when the system printer output has been directed to a disk file, the system printer file is kept, but not enqueued.
8. The keyboard is only unlocked if it is currently locked. It may have been previously unlocked by calling FSFRCE and using the AUNLOCK processing option.

Description

Terminates the use of a device by GDDM.

Any resources (such as symbol sets or page contents) that have been defined for the device are released. Any device that is subsequently opened with the same device identifier bears no relationship to the device now being closed.

Any usage currently in force for this device is discontinued.

If operator windows are being used by this instance of GDDM, a call to DSCLS may cause operator windows to be deleted. If an operator window that is associated with a virtual device in another instance of GDDM is deleted, the results of subsequent operations on that virtual device are undefined (see WSDEL).

Note: This function differs from most GDDM functions, in that the function tries to complete even if an error is returned.

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0082 E  DEVICE DOES NOT EXIST
ADM0089 W  INVALID OPTION
ADM0304 E  INVALID FILE NAME, 'a'
ADM0312 E  CONCURRENT USAGE OF FILE 'a' NOT ALLOWED
ADM0327 E  'a1' ERROR CODE X'xx', ON 'a2'
```

Note: Messages ADM0304, ADM0312, and ADM0327 are only issued in CICS transaction-dependent pseudoconversational mode.

DSCMF

Function

To enable automatic entry into User Control.

DSCMF	(control)
APL code	439
GDDM RCP code	X'0C080C01' (201853953)

Parameters

control *(specified by user) (fullword integer)*
Possible values are:
- **0** Disabled; that is, should not cause automatic entry into User Control. This is the default.
- **1** Enabled; subsequent “read” calls force automatic entry into User Control.

Description

This call causes subsequent “read” type calls (such as ASREAD) to force the screen automatically into User Control mode at the initial entry level. Initial entry level varies according to other factors related to the program environment, as follows:

Window entry level: If the DSOPEN processing option WINDOW is in effect.

Graphic entry level: If the device supports graphics and the DSOPEN processing option WINDOW is **not** in effect.

Output panel entry level: None of the above; that is, windows are not used, and graphics are not supported.

If the DSOPEN processing option CTLMODE is **not** active, the DSCMF call has no effect.

Principal errors

```
ADM0153 E  CONTROL VALUE n IS INVALID
ADM2997 E  USER CONTROL IS NOT ALLOWED
```

DSCOPY

Function

Send transformed picture to alternate device.

DSCOPY	(width, depth. hor-off, ver-off, count, option-array)
APL code	909
GDDM RCP code	X'0C180008' (202899464)

Parameters

width *(specified by user) (short floating point)*
The percentage width of the output area on the output medium.

depth *(specified by user) (short floating point)*
The percentage depth of the output area on the output medium.

hor-off *(specified by user) (short floating point)*
The percentage horizontal offset, from the left of the medium, of the output area on the output medium.

ver-off *(specified by user) (short floating point)*
The percentage vertical offset, from the top of the medium, of the output area on the output medium.

count *(specified by user) (fullword integer)*
The number of elements in the **option-array** parameter.

option-array *(specified by user) (array of fullword integers)*
An array of options for the DSCOPY call. There are three elements:

1 – source
Specifies which type of data to copy. Three possible values:
- **0** Graphics. Only the graphics field is copied to the output area (default).
- **1** Image. Only the image field is copied to the output area.
- **2** The whole page. All non-alphanumeric fields (graphics and image) are copied to the output area. .

2 – aspect ratio control
Specifies whether or not to maintain the aspect ratio of the required picture when copied to the output area. Two possible values:
- **0** Preserve aspect ratio of picture within output area (default).
- **1** Stretch picture to fit the output area.

3 – rotation
Specifies the clockwise rotation applied to the picture. Four possible values:
- **0** 0 degrees (default)
- **1** 90 degrees
- **2** 180 degrees
- **3** 270 degrees.

Description

Copies the contents of the chosen **source** to the current alternate device. The copied picture is sized and positioned (and rotated) according to the values specified in percentages of the target device's page (positioned to cell accuracy on cell-constrained devices). Thus, the size of the copy, in character cell units, can differ from that of the picture on the current page, and a larger picture may be obtained than would have been the case using FSCOPY.

If aspect-ratio-control value 1 is specified, the picture is “stretched” to fit the requested output area. Otherwise, the aspect ratio of the picture is preserved within the output area. If the picture has different proportions from those of the output area, this means that the picture will be smaller than the requested output area in one dimension, and centered within it.

Note: The picture displayed may be smaller than the field in which it is drawn, therefore, selecting 100% does not necessarily fill the page.

For more information about the handling of picture components during the copy process, refer to the FSCOPY call.

Notes:

1. A **width** or **depth** (but not both) of more than 100%, up to 1000%, may be specified, to allow the user to take advantage of devices which support drawings too long for a single sheet of paper (for example, roll-feed plotters). The dimension that can exceed 100% is that corresponding to the physical width after any rotation specified in the PLTROTAT processing option is applied. Thus if PLTROTAT specifies a rotation of 90 or 270 degrees, it is the depth value in this call that may exceed 100%.

   Specifying a value >100% for a copy to a single-sheet device has unpredictable results.
2. The setting of GSARCC has no bearing on this call.
3. Printing or plotting processing options (procopts) that affect the size, rotation or position of the output are honoured in addition to values specified by the DSCOPY call.
4. The limitations to rotated graphics described by the GSSAGA call also apply to graphics rotated by the DSCOPY call.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE |GSCOPY|MAPPING}'
           IS NOT SUPPORTED FOR THIS DEVICE
ADM0890 E  INVALID WIDTH n SPECIFIED
ADM0891 E  INVALID DEPTH n SPECIFIED
ADM0892 E  INVALID HORIZONTAL OFFSET n SPECIFIED
ADM0893 E  INVALID VERTICAL OFFSET n SPECIFIED
ADM0894 E  SUM OF WIDTH AND HORIZONTAL OFFSET n IS
           INVALID
ADM0895 E  SUM OF DEPTH AND VERTICAL OFFSET n IS
           INVALID
ADM0896 E  SOURCE TYPE n DOES NOT EXIST
ADM0897 E  SOURCE OF COPY n MUST BE IN THE RANGE 0
           THROUGH 2
ADM0898 E  ROTATION VALUE n MUST BE IN THE RANGE 0
           THROUGH 3
ADM0899 E  ASPECT RATIO CONTROL n MUST BE 0 OR 1
```

DSDROP

Function

To discontinue device usage.

DSDROP (usage, device-id)	
APL code	904
GDDM RCP code	X'0C000203' (201327107)

Parameters

usage *(specified by user) (fullword integer)*
The device usage code. Possible values are:
1 Current primary device
2 Current alternate device.

device-id *(specified by user) (fullword integer)*
The identifier of the device concerned.

Description

Indicates that a device is no longer to operate with the specified usage.

Unless it is also being used in another mode (for example, as both the primary and alternate device), the device now enters a state in which the only operations that can be performed against it are DSCLS, DSRNIT, DSQDEV, and DSUSE. Its resources are not, however, released, and its usage (together with all the contents, and so on, that existed at the time of the DSDROP) may subsequently be restored by means of a DSUSE call.

A device is automatically dropped when the application issues a DSUSE call for a different device.

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0076 E  INVALID DEVICE USAGE
ADM0082 E  DEVICE DOES NOT EXIST
ADM0084 E  DEVICE NOT IN USE
```

DSFRCE

Function

To output a member to a Partitioned Data Set

DSFRCE (member-name)	
APL Code	910
GDDM RCP code	X'0C10000C' (202375180)

Parameters

member-name *(specified by user) (8-byte character string)*
The name (left-justified) that is to be given to the current page, to be stored in page segment or overlay format as a member of a partitioned data set.

Description

Outputs the current page as a page segment or overlay, to a member of the partitioned data set defined by the DSOPEN for the current device.

The device must be opened with processing option FRCETYPE set to DSFRCE.

The characteristics of the member are those specified on the current DSOPEN.

Principal errors

```
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
```

DSOPEN

Function

To open a device.

DSOPEN	**(device-id, family, device-token, procopt-count, procopt-list, name-count, name-list)**
APL code	901
GDDM RCP code	X'0C000200' (201327104)

Parameters

device-id *(specified by user) (fullword integer)*
The device identifier. It must not be negative.

The values 0 and 1 should also be avoided, unless it is certain that they are not required for default devices.

Note that GDDM can automatically open devices, as follows:

- It opens the default primary device (**device-id** = 0), when no primary device has been specified (by a DSUSE), or
- It opens the default alternate device (**device-id** = 1), if an FSOPEN statement is issued.

family *(specified by user) (fullword integer)*
The device-family code, which can take these values:

1 3270-family devices
These include:

3270 displays and printers
Plotters
Scanners
5550-family displays and printers
5081 high function graphics display
Personal computer systems using GDDM-PCLK or GDDM-OS/2 Link, together with attached printers and plotters
ASCII displays.

For a full list of these devices, see the *GDDM General Information* manual for the latest release.

2 Queued printer files
Queued printer output for any of the family-1 devices.

3 System printer files
For a full list of system printers, see the *GDDM General Information* manual for the latest release.

4 Page printer files
For a full list of page printers, see the *GDDM General Information* manual for the latest release.

Note: A device-family code of zero, which can be returned by DSQDEV, FSQDEV, or FSQURY (see the descriptions of these functions), cannot be specified for the family parameter on DSOPEN.

device-token *(specified by user) (8-byte character string)*
Tells GDDM where to find the properties of the device. GDDM left-justifies the supplied parameter, and converts it to uppercase, if necessary. It can have these values:

- '⋆ '—GDDM is to determine device properties from subsystem tables, the device itself, or GDDM's own defaults, or both as appropriate to the device and subsystem.
- The name of a device token that is part of the ADMLSYS1, ADMLSYS3, ADMLSYS4, or ADMLSYSA table. A version of each of these tables is supplied with GDDM, and contains several device tokens. A table containing different definitions can be regenerated if required. For contents of the GDDM-supplied tables and other information, see Chapter 21, "Device characteristics tokens" on page 421.

Notes:

1. The device token determines the size of the default page. (See the maxpage parameter of the ADMM3270 macro in the *GDDM System Customization and Administration* book).
2. On a queued printer (family-2 device), the device token does not affect the device properties subsequently used by the GDDM print utility when it processes the print request; the device token is used only to determine the size of the default page. The device characteristics are established when the real output device is opened. If a heading page is specified, it is created at

output time using the default page size of the real output device.

procopt-count *(specified by user) (fullword integer)*
The number of fullwords in **procopt-list**. Zero can be specified to indicate that **procopt-list** is empty and is not to be inspected.

procopt-list *(specified by user) (an array of fullword integers)*
This list is used to pass miscellaneous processing options to GDDM. Some of these options are dependent on the device family, and some on a particular subsystem.

Each option is passed as an option group, consisting of a fullword option code, followed by one or more fullwords of option data. Option groups can be specified in any order. Only those option groups for which it is specifically required to override GDDM's default action need be specified. If an option group is not relevant in the execution circumstances (for example, CMS attention handling, while operating under TSO), that option group is ignored.

The parameter list takes the form:

First Option Group Code
Option 1
Option 2
(....)
Second Option Group Code
Option 1
(....)

The option groups are described in detail in Chapter 19, "Processing options" on page 395.

name-count *(specified by user) (fullword integer)*
The number of 8-byte name-parts in **name-list**. Zero can be specified to indicate that **name-list** is empty and is not to be inspected. Even if **name-list** is empty, it must be specified in the parameter list of the call.

name-list *(specified by user) (array of 8-byte character tokens)*
This list gives the name-parts that constitute the name by which the device is known to the underlying subsystem. The number of name-parts that can be supplied and their meanings are subsystem- and family-dependent. GDDM left-justifies each supplied name-part and converts it to uppercase, if necessary. Even if **name-list** is empty, it must be specified in the parameter list of the call.

The values for the **name-count** and **name-list** parameters for the various subsystems are shown in Chapter 20, "Namelists" on page 415.

Description

Opens (initializes) a device that GDDM is to access.

There is usually no need to issue a DSOPEN call when the output is to appear on the invoking terminal – known as the **user console**. By default, GDDM automatically opens the user console with device ID of 0 if it is required.

If a device other than the user console is to be made known to GDDM, it must have an explicit DSOPEN call unless a nickname is used (see below) that acts on GDDM's automatic opening of the user console.

After an explicit DSOPEN and before creating any output for a device, the device must be made current using a DSUSE call. According to the usage specified in the DSUSE call, statements apply to that device until a new device is made current. The scheme is the same as that for pages; that is, several devices are available but only one of them is current at any one time.

In a windowing environment, the DSOPEN call applies to a virtual device that is displayed in an operator window of a real device. However, unless the application needs to use the windowing calls, it does not have to distinguish between real and virtual devices. For more information, see the *GDDM Base Application Programming Guide*.

DSOPEN calls are required for auxiliary devices, such as plotters, which can be attached through 3179-G, 3192-G, or 3472-G display stations, 3270-PC/G or 3270-PC/GX workstations, 5550-family Multistations, or devices using GDDM-PCLK. To GDDM, a plotter is a family-1 device. It is identified by using DSOPEN's **name-count** and **name-list** parameters. For full information, see Chapter 20, "Namelists" on page 415.

Image scanners (3117 and 3118) are *always* attached to an image display (3193), and therefore DSOPEN calls cannot be made directly to them; DSOPEN calls must be made to the image display.

All loadable-character-set stores on the device are assumed to be available for use by GDDM, unless they are subsequently reserved by a call to PSRSV.

In CICS transaction-dependent pseudoconversational mode, on the initial invocation of a transaction and after processing the initial input, the PSCNVCTL processing option must be set to "start pseudoconversational mode." On subsequent invocations, it must be set to "continue pseudoconversational mode." For further information, see the *GDDM Base Application Programming Guide*.

Notes:

1. There are restrictions on the use of DSOPEN for some subsystems. For further information, see the *GDDM Base Application Programming Guide*.
2. To enable the invocation of the Interactive Chart Utility in circumstances that require special DSOPEN parameters, the PGF feature of GDDM provides the sample module ADMUCDSO. This module is supported under TSO and VM/CMS only. Information about how to use ADMUCDSO is given in the *GDDM-PGF Programming Reference* book.
3. Throughout the description of this call, references to TSO also apply to the MVS Batch environment.

Example of a DSOPEN call

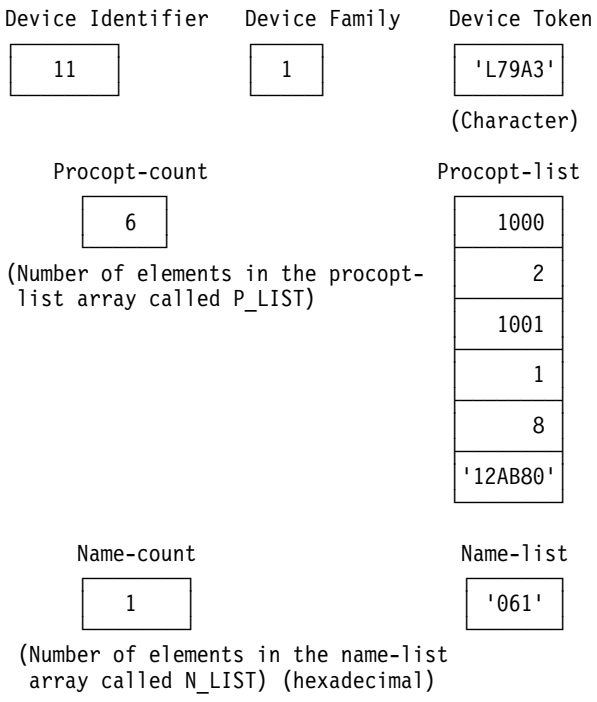
Here is an example of a DSOPEN call. The example assumes that the necessary declare statements for the programming language in use have been written (for a PL/I model, see the *GDDM Base Application Programming Guide*).

This example assigns a device identifier of 11 to a local 3279 Model 3 display (screen size 32 by 80), with processing option groups 1000 and 1001 (for CMS PA1/PA2 protocol and attention handling), and with a virtual address known to VM/CMS as '061'.

The DSOPEN call to define the device in this way is:

```
CALL DSOPEN(11,1,'L79A3',6,P_LIST,1,N_LIST)
```

The following diagram illustrates the values in each of the DSOPEN parameters:



The procopt-list called P_LIST is constructed as follows:

1000	First procopt group code (CMS PA1/PA2 protocol)
2	Procopt value for this group (PA1 to CP, PA2 to user)
1001	Second procopt group code (CMS attention handling)
1	First procopt value (extended attention handling) for group
8	Second procopt value (length of feedback block) for group
X'12AB80'	Third procopt value (address of feedback block) for group

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0075 E  INVALID DEVICE FAMILY n
ADM0077 E  DEVICE ALREADY EXISTS
ADM0078 E  INVALID NAME COUNT n
ADM0079 E  INVALID PROCESSING OPTIONS COUNT n
ADM0080 E  DEVICE IS NOT HARDCOPY. DEVICE TOKEN WAS 'a'
ADM0085 E  UNSUPPORTED PROCESSING OPTION CODE n
ADM0086 E  INVALID PROCESSING OPTION VALUE n1 FOR CODE
           n2
ADM0087 E  QUERY ERROR: TOKEN 'a',HDR X'llllttqq' OFF
           X'xx' REASON n
ADM0088 E  QUERY VALUE ERROR: TOKEN 'a', HDR
           X'llllttqq' , OFF X'xx'
ADM0090 E  NO USABLE AREA IN QUERY REPLY
ADM0091 E  DEVICE NOT SUPPORTED ON THIS SUBSYSTEM.
           DEVICE TOKEN WAS 'a'
ADM0092 E  CONFLICT BETWEEN PROCESSING OPTION CODE n
           AND DEVICE TOKEN 'a'
ADM0093 W  PAGE SIZE REDUCED TO n1,n2 TO FIT MEDIUM
ADM0094 W  PROCESSING OPTION CODE n NOT SUPPORTED BY
           PLOTTER
ADM0096 E  AUXILIARY DEVICE TYPE a IS UNKNOWN TO GDDM
ADM0097 E  AUXILIARY DEVICE 'a' NOT FOUND
ADM0100 S  DEVICE NOT SUPPORTED FOR FAMILY n. DEVICE
           TOKEN WAS 'a'
ADM0101 E  USABLE AREA WIDTH OR DEPTH NOT LESS THAN
           16384 PIXELS
ADM0102 E  INVALID PROCOPT GROUP. CODE = a1{, NUMBER =
           n}, REASON = a2
ADM0103 E  PLOTTER CANNOT SUPPORT THE REQUESTED PAPER
           SIZE
ADM0104 E  PLOT AREA IS TOO BIG FOR THE CURRENT PLOTTER
           SETUP
ADM0106 E  DEVICE TOKEN 'a' IS FOR AUXILIARY DEVICES
           ONLY
ADM0107 E  PLOTTER CANNOT BE SUPPORTED ON AN
           OUTPUT-ONLY TERMINAL
```

```
ADM0108 E  CANDIDATE OPERATOR WINDOW ALREADY IN USE OR
           NOT ACCESSIBLE
ADM0109 E  ONLY ONE DEVICE AT A TIME MAY BE WINDOWED
ADM0400 E  DEFAULT 'MAXIMUM TRANSMISSION SIZE' n IS
           INVALID
ADM0401 E  INVALID TERMINAL TYPE. THIS IS NOT A DISPLAY
           TERMINAL
ADM0402 E  TERMINAL ERROR. RETURN CODE n (DECIMAL) FROM
           a
ADM0403 E  TERMINAL a DISCONNECTED OR DOES NOT EXIST
ADM0404 E  DATA STREAM ERROR
ADM0407 E  DEFAULT 'MAXIMUM FSSAVE/FSSHOW BUFFER SIZE',
           n, IS INVALID
ADM0409 E  INVALID COLOR MASTER SET NUMBER n
ADM0441 S  TERMINAL ERROR{ ON a}. DEVICE DISCONNECTED
           OR NOT DEFINED
ADM0444 E  INVALID CMS ATTN OPTION FOR DEVICE 'a'
ADM0445 E  INVALID CMS PA1/PA2 OPTION FOR DEVICE 'a'
ADM0461 E  I/O PCB ALREADY IN USE OR NOT AVAILABLE
ADM0462 E  NO TP PCB WITH DESTINATION a AVAILABLE FOR
           USE BY THIS DEVICE
ADM0481 E  NAME COUNT OF n1 FOR 'a' IS INVALID FOR
           FAMILY n2
ADM0482 E  DEVICE NAME LIST 'a' IS INVALID FOR FAMILY n
ADM0483 E  COLOR MASTER a NOT FOUND
ADM0484 E  NO DEVICE TOKEN PROVIDED FOR DUMMY DEVICE
ADM0485 E  SUBSYSTEM DEVICE 'a' ALREADY OPEN
ADM0872 W  SCREEN SIZE TOO SMALL FOR USER CONTROL
```

DSQCMF

Function

To query automatic entry to User Control function.

DSQCMF	**(function)**
APL code	440
GDDM RCP code	X'0C080C02' (201853954)

Parameters

function *(returned by GDDM) (fullword integer)*
Defines the status of automatic entry to User Control. Possible values are:

0 Automatic entry to User Control is disabled.
1 Automatic entry to User Control is enabled.

Description

Returns the current status of automatic entry to User Control as set by the most recent DSCMF call.

Principal errors

None.

DSQDEV

Function

To query device characteristics.

DSQDEV	**(device-id, device-token, procopt-count, procopt-list, name-count, name-list, char-count, char-list)**
APL code	907
GDDM RCP code	X'0C000206' (201327110)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the device whose characteristics are required.

device-token *(returned by GDDM) (8-byte character string)*
This parameter indicates the source of GDDM's information about the device properties. It can take the following values:

- '★ '— GDDM-determined device properties from subsystem tables, the device itself, from GDDM's own defaults, or from both as appropriate.
- The name of a device definition that is part of the ADMLSYSn table (where n is 1, 3, 4, or A).

procopt-count *(specified by user) (fullword integer)*
The number of fullwords in **procopt-list**. Zero can be specified to indicate that the **procopt-list** is empty and is not to be returned.

procopt-list *(returned by GDDM) (an array of fullword integers)*
In this list, GDDM returns the processing option groups that are applicable to the particular device's family, and to the environment in which GDDM is operating. (Plotter processing options are returned for plotters only.) Each option group consists of several contiguous fullwords, the first of which is a code identifying the particular option. The list is padded with zeros, if necessary. If the **procopt-count** value is too small, a processing option group may be truncated. For variable-length "mergeable" groups only (groups 4 and 20), the embedded count is modified appropriately.

For information on processing option groups, and the families and environments to which they apply, refer to Chapter 19, "Processing options" on page 395.

name-count *(specified by user) (fullword integer)*
The number of 8-byte tokens in **name-list**. Zero can be specified to indicate that **name-list** is empty and is not to be returned.

name-list *(returned by GDDM) (array of 8-byte character tokens)*
The name by which the device is known to the underlying

subsystem. Again, for the meaning of these tokens for various device families and in various environments, refer to Chapter 20, “Name-lists” on page 415.

char-count *(specified by user) (fullword integer)*
The number of fullwords in **char-list**. Zero can be specified to indicate that **char-list** is empty and is not to be returned.

char-list *(returned by GDDM) (an array of fullword integers)*
An array of fullword integers to receive the device characteristics information. For an explanation of the device characteristics information that can be returned, see FSQURY (**code=0**).

Description

Requests the characteristics of the device to be returned. A device token, processing options list, and name-list are returned in DSOPEN format, including the values of any options that were defaulted when the device was opened; see DSOPEN. Another list, containing miscellaneous device characteristics, is also returned.

Storage for these lists is provided by the invoker. The various count parameters indicate how many elements (fullwords, or 8-byte tokens, as appropriate) for which there is room in each list. If there is not enough space, GDDM returns only as many elements as the space allows; if there is extra space, it pads the space with zeros or blanks, as appropriate.

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0078 E  INVALID NAME COUNT n
ADM0079 E  INVALID PROCESSING OPTIONS COUNT n
ADM0081 E  INVALID DEVICE CHARACTERISTICS COUNT
ADM0082 E  DEVICE DOES NOT EXIST
ADM0129 E  ARRAY COUNT n IS INVALID
```

DSQUID

Function

To query unique device identifier.

DSQUID	**(device-id)**
APL code	905
GDDM RCP code	X'0C000204' (201327108)

Parameters

device-id *(returned by GDDM) (fullword integer)*
An identifier for which no device currently exists.

Description

Requests return of a unique unused device identifier. This call can be used by a modular application program to obtain an identifier for a new device, without conflicting with a device that has already been opened, or that will be opened later, by another part of the application. The device identifier returned is the highest available unused number.

Principal errors

None.

DSQUSE

Function

To query device usage.

DSQUSE	**(usage, device-id)**
APL code	906
GDDM RCP code	X'0C000205' (201327109)

Parameters

usage *(specified by user) (fullword integer)*
The code for the usage for which the current device is required. For valid usage codes, see DSUSE.

device-id *(returned by GDDM) (fullword integer)*
Receives the identifier of the device currently operating with the specified usage. If there is none, a value of –1 is returned.

Description

Returns the identifier of the device currently operating with the specified usage.

Principal errors

```
ADM0076 E  INVALID DEVICE USAGE
```

DSRNIT

Function

To reinitialize a device.

DSRNIT	**(device-id, option)**
APL code	908
GDDM RCP code	X'0C000207' (201327111)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the device to be reinitialized.
option *(specified by user) (fullword integer)*
The action to be carried out at device reinitialization. For the use of this parameter, see DSCLS.

Description

Restores the status of a device to that which existed just after it was first opened, so that all resources (symbol sets, for example) that have been defined for the device are released.

Any usage currently in force for this device is discontinued; after a DSRNIT call has processed, the device is not in use as either the primary or the alternate device.

If operator windows are being used by this instance of GDDM, a call to DSRNIT may cause operator windows to be deleted. If an operator window that is associated with a virtual device in another instance of GDDM is deleted, the results of subsequent operations on that virtual device are undefined (see WSDEL).

Note: This function differs from most GDDM functions in that the function tries to complete even if an error is returned.

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0082 E  DEVICE DOES NOT EXIST
ADM0089 W  INVALID OPTION
```

DSUSE

Function

To specify device usage.

DSUSE	**(usage, device-id)**
APL code	903
GDDM RCP code	X'0C000202' (201327106)

Parameters

usage *(specified by user) (fullword integer)*
The device usage code. Possible values are:
1 Current primary device
2 Current alternate device.
device-id *(specified by user) (fullword integer)*
The identifier of the device concerned.

Description

DSUSE activates a device, and specifies the usage that the application may make of the device. The device so activated becomes the one whose contents are manipulated by subsequent calls. For example, the current primary device is the one within which a page would be created if an FSPCRT call were issued.

These rules apply to DSUSE:

- The device must already be open.
- Any device currently operating with the required usage is dropped from that usage.
- A device of any family can be used as the primary or alternate device. However, for family-4 alternate devices, a cell-based device token must be used.

Note: If you issue a function requiring a primary device to be in use, when no primary device is in use, then GDDM makes the default primary device the current primary device. (The default primary device is always given the device ID 0.) GDDM also makes the alternate device that has been opened by an FSOPEN call the current alternate device.

Principal errors

```
ADM0074 E  INVALID DEVICE IDENTIFIER
ADM0076 E  INVALID DEVICE USAGE
ADM0082 E  DEVICE DOES NOT EXIST
```

ESACRT

Function

To create an application group.

ESACRT	**(application-group-id)**
APL code	127
GDDM RCP code	X'000A0000' (655360)

Parameters

application-group-id *(returned by GDDM) (fullword integer)*
The identifier of the new application group that was created and made current.

Description

Creates a new application group and returns its identifier. The new application group becomes the current application group. All instances of GDDM initialized from this time until another application group is made current are associated with this application group.

The purpose of application groups is to manage GDDM resource recovery within a single operating task or subtask.

A call to ESADEL releases all the resource obtained by the GDDM instances associated with a particular application group, by terminating all such instances (see FSTERM). This function can be used, typically, by a task manager to release any remaining GDDM resources when a windowed application terminates.

If the task manager uses operating system tasking services to manage resource recovery, this function should not be needed, but it can be used with care. An application group must not be current if there is a chance of an instance of GDDM being initialized in another task or subtask. The consequences of ignoring this rule are undefined.

Principal errors

None.

ESADEL

Function

To delete application group.

ESADEL	**(application-group-id)**
APL code	128
GDDM RCP code	X'000B0000' (720896)

Parameters

application-group-id *(specified by user) (fullword integer)*
The identifier of the application group to be deleted.

The current application group could be deleted, either explicitly, or implicitly by causing the instance of GDDM which created it to be terminated. In either case, the new current application group is the application group associated with this instance of GDDM, if there is one, or else is zero.

Description

Releases all resources obtained by GDDM instances associated with the specified application group, by causing an FSTERM to be issued for all such instances.

If operator windows are being used by this instance of GDDM, a call to ESADEL may cause them to be deleted. If an operator window that is associated with a virtual device in another instance of GDDM is deleted, the results of subsequent operations on that virtual device are undefined (see WSDEL).

Principal errors

```
ADM0040 E  APPL  GROUP  ID  n  INVALID  OR  BELONGS  TO
           ANOTHER GDDM INSTANCE
```

ESAQRY

Function

To query the current application group.

ESAQRY	**(application-group-id)**
APL code	129
GDDM RCP code	X'000C0000' (786432)

Parameters

application-group-id *(returned by GDDM) (fullword integer)*
The identifier of the current application group, or 0 if there is no application group current.

Description

Returns the identifier of the current application group.

Principal errors

None

ESASEL

Function

To select an application group.

ESASEL	**(application-group-id)**
APL code	130
GDDM RCP code	X'000D0000' (851968)

Parameters

application-group-id *(specified by user) (fullword integer)*
The identifier of the application group to be made current, or 0.

Description

Selects an application group to become the current application group. If an application group identifier of 0 is specified, none of the existing application groups are considered to be current.

Principal errors

```
ADM0041 E  APPLICATION GROUP IDENTIFIER n IS INVALID
```

ESEUDS

Function

To specify encoded user default specification.

ESEUDS	**(length, encoded-UDSL)**
APL code	124
GDDM RCP code	X'00080000' (524288)

Parameters

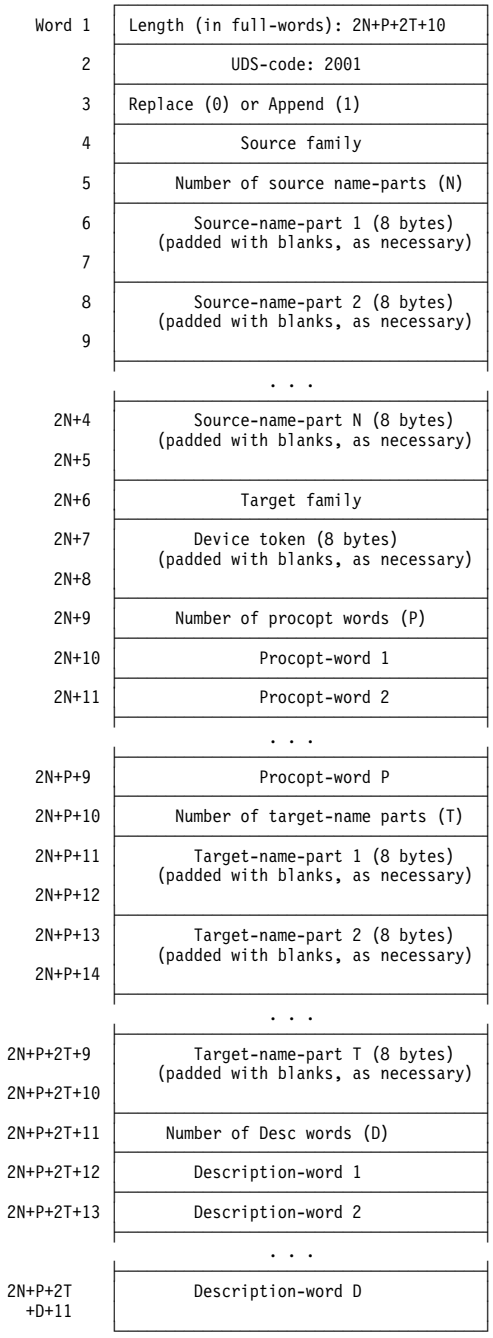
length *(specified by user) (fullword integer)*
The length in bytes of an encoded user default specification list (UDSL). The length must not exceed 32000.

encoded-UDSL *(specified by user) (character)*
The encoded user default specification list (UDSL). An encoded user-default specification list consists of encoded external defaults, see Chapter 18, “External defaults” on page 379, and encoded nicknames.

Description

Specifies one or more encoded user default specifications (UDSs) as an encoded user default specification list (UDSL).

Encoded format of a nickname UDS: The encoded format of a nickname UDS is shown here.



The operation of an encoded nickname UDS is identical to that of a source-format nickname UDS. However, the following points should be noted:

- A null source-name-list is expressed by specifying 0 as the number of source-name-parts (N) and by omitting the source-name-parts entirely.
- A null device token is expressed by specifying it as all blanks or all X'00'.
- A null procopt-list is expressed by specifying 0 as the number of procopt-words (P) and by omitting the procopt-words entirely.

- A null target-name-list is expressed by specifying 0 as the number of target-name-parts (T) and by omitting the target-name-parts entirely.

Principal errors

```
ADM0057 E  DEFAULTS ERROR. INVALID COUNT n IN UDS IN a
ADM0061 E  DEFAULTS ERROR. UDS {TYPE 'a1'| KEYWORD
           'a2'| CODE n} NOT ALLOWED IN a3
ADM0062 E  DEFAULTS ERROR. UDS {TYPE 'a'| CODE n}
           UNKNOWN
ADM0064 E  DEFAULTS ERROR. VALUE IN {'a1'} UDS {KEYWORD
           'a2'| CODE n} IS INVALID
ADM0065 E  DEFAULTS ERROR. UDS {KEYWORD 'a'| CODE n}
           NOT VALID ON THIS SUBSYSTEM
ADM0066 E  DEFAULTS ERROR. {'a1'} UDS {KEYWORD 'a2'|
           CODE n} HAS TOO MANY OPERANDS
```

ESLIB

Function

To perform library management.

Note: This function is not available under VM/CMS.

ESLIB	**(type, count, names)**
APL code	112
GDDM RCP code	X'08142000' (135536640)

Parameters

type *(specified by user) (fullword integer)*
The type of object. Possible values are:

0 All types of object
1 Image or vector symbol sets
2 Generated mapgroups
3 Pictures saved by an FSSAVE call
4 Chart format descriptions
5 Chart data files
6 Reserved for GDDM-IMD tutorial pages
7 GDF files saved by a GSSAVE call
8 Reserved for GDDM-GKS metafiles
9 Chart data definition tables
10 Projection definitions
11 Image data
12 Reserved for GDDM-PCLK and GDDM-OS/2 Link files

count *(specified by user) (fullword integer)*
The number of library names supplied. It must be in the range 1 through 256.

names *(specified by user) (array of 8-byte character tokens)*
The list of library names.

If any token is '★ ' the default for the given type of object, as defined in GDDM's external defaults, is substituted. Otherwise, the interpretation of the name depends on the subsystem in which the application program is running, as follows:

CICS The names must be those of VSAM data sets set up during the generation of the CICS system. The use of such data sets is more fully explained in the *GDDM System Customization and Administration* book.

IMS The names must be DBD names that match those appearing in the Program Specification Block for the application program. Their associated program communication blocks (PCBs) must have been made available for GDDM's use by the ESPCB call.

TSO The names must be ddnames or file names that have been allocated to partitioned data sets.

Description

The call names one or more libraries to be searched for GDDM objects. It can be used to simplify data set searches, and to improve data-set security by subsystem security methods. The meaning of **library** depends on the subsystem which GDDM is running under, thus:

CICS VSAM data set
IMS Database
TSO Partitioned data set.

A GDDM object is:

- A vector or image symbol set
- A GDDM-IMD-generated mapgroup
- A data stream saved by an FSSAVE call.
- A chart format descriptor
- A chart data descriptor
- A GDF file saved by a GSSAVE call.
- A chart data definition table
- A projection
- An image.

When a request is made to access an object (for example, to show a saved data stream by an FSSHOW call). GDDM searches the given libraries in the order specified in this statement. For a request to store an object, the first library in the list is used. That is, all libraries except the first are treated as "read only."

The library list may apply to all types of objects or to only a single type (see below). The default library list for each object is defined by external defaults; refer to Chapter 18, "External defaults" on page 379.

Principal errors

```
ADM0301 S  OBJECT TYPE OR FILENAME/DDNAME NOT DEFINED
           IN EXTERNAL DEFAULTS
ADM0311 S  FUNCTION NOT SUPPORTED
```

```
ADM0334 E  PCB NOT AVAILABLE FOR DBD/LTERM NAME 'a'
ADM0344 E  ERROR  IN  OBJECT  DATABASE  DEFINITION  IN
           EXTERNAL DEFAULTS
ADM0370 E  INVALID OBJECT TYPE - n
ADM0371 E  INVALID NAME COUNT - n
```

ESPCB

Function

To identify program communication block.

Note: This function is available only under IMS/VS.

ESPCB	**(type, pcb)**
APL code	113
GDDM RCP code	X'081C1000' (136056832)

Parameters

type *(specified by user) (fullword integer)*
Identifies the type of program communication block. Possible values are:
0 I/O PCB
1 TP PCB other than the I/O PCB
2 DB PCB.

pcb *(specified by user) (12-byte character string)*
Identifies the program communication block to be used (**not** a pointer to it).

Description

Identifies a program communication block (PCB) that can be used by GDDM. This call is only applicable to transactions running under the control of IMS/VS.

Principal errors

```
ADM0335 E  DUPLICATE PCB DEFINED
ADM0336 E  INVALID PCB TYPE
```

ESQCPG

Function

To query the code page of a GDDM object.

ESQCPG	**(object-name, type, cpgid)**
APL code	133
GDDM RCP code	X'00100000' (1048576)

Parameters

object-name *(specified by user) (8-byte character string)*
The name of the object. When a CECP Application code page is being used, this name is translated from the Application code page into the Installation code page.

type *(specified by user) (fullword integer)*
The object type. It must be one of:

1	ADMSYMBL	Symbol sets
2	ADMGGMAP	Generated mapgroups
3	ADMSAVE	FSSAVE files
4	ADMCFORM	Chart format files
5	ADMCDATA	Chart data files
6	Not used	
7	ADMGDF	GDF files
8	ADMGKSM	GKS metafiles
9	ADMCDEF	Chart definition files
10	ADMPROJ	Projection definition files
11	ADMIMG	Image data files

cpgid *(returned by GDDM) (fullword integer)*
The global code page identifier. If the object is not tagged this parameter is zero. No validation is performed that the code page global identifier is supported.

Description

Returns the code page global identifier of the named object. Objects that are not tagged with a code page global identifier return a value of zero.

Principal errors

```
ADM0307 E  FILE 'a' NOT FOUND
ADM0370 E  INVALID OBJECT TYPE - n
```

ESQEUD

Function

To query encoded user default specification.

ESQEUD	**(code, length, encoded-UDSL)**
APL code	135
GDDM RCP code	X'00120000' (1179648)

Parameters

code *(specified by user) (fullword integer)*
The identifier of the external default.

The only supported values are 4, the national language, and 125, the application code page.

length *(specified by user) (fullword integer)*
The length in bytes of the encoded user default specification list (UDSL). It must be set to 12.

encoded-UDSL *(returned by GDDM) (character)*
The encoded user default specification list (UDSL).

Description

Queries the current value of the specified external default. ESQEUD(125,12,eudslist), for example, can be used to query the current value of the application code page. For CECP 00500 this would return 3 fullwords (3, 125, and 500) in **eudslist**.

Principal errors

```
ADM0042 E  INVALID DEFAULT IDENTIFIER n
ADM0044 E  INVALID LENGTH n1 FOR QUERY CODE n2
```

ESQOBJ

Function

To query existence of GDDM object on auxiliary storage.

ESQOBJ	**(object-name, type, exists)**
APL code	140
GDDM RCP code	X'08142400' (135537664)

Parameters

object-name *(specified by user) (8-byte character string)*
The name of the object.

type *(specified by user) (fullword integer)*
The object type. It must be one of:

1	ADMSYMBL	Image or vector symbol sets
2	ADMGGMAP	Generated mapgroups
3	ADMSAVE	Pictures saved by an FSSAVE call
4	ADMCFORM	Chart format descriptions
5	ADMCDATA	Chart data files
6		Reserved
7	ADMGDF	GDF files saved by a GSSAVE call
8	ADMGKSM	GKS metafiles
9	ADMCDEF	Chart data definition tables
10	ADMPROJ	Projection definitions
11	ADMIMG	Image data

exists *(returned by GDDM) (fullword integer)*

0 The object does not exist.
1 The object exists.

Description

Queries the existence of a GDDM object on auxiliary storage. The existence of the object is indicated by the returned parameter.

Principal errors

```
ADM0304 E  INVALID FILE NAME, 'a'
ADM0370 E  INVALID OBJECT TYPE - n
```

ESQUNL

Function

To query length of device nickname information.

ESQUNL	**(family, buffer-length)**
APL code	136
GDDM RCP code	X'00130000' (1245184)

Parameters

family *(specified by user) (fullword integer)*
The family code. Determines the family or families for which the length of the nickname information is returned.
0 Families 0,1,2,3,4
1 Families 0 and 1 only
2 Families 0 and 2 only
3 Families 0 and 3 only
4 Families 0 and 4 only.

buffer-length *(returned by GDDM) (fullword integer)***.**
The length in bytes of the nickname information

Description

Queries the length, in bytes, of nickname information available for the specified family. This may be used to determine the length of the buffer required for ESQUNS.

Principal errors

```
ADM0075 E  INVALID DEVICE FAMILY n
```

ESQUNS

Function

To query devices defined by nickname.

ESQUNS	**(family, buffer-length, buffer, returned-length)**
APL code	137
GDDM RCP code	X'00140000' (1310720)

Parameters

family *(specified by user) (fullword integer)*
The family code. Determines the family or families for which information is returned.
0 Families 0,1,2,3,4
1 Families 0 and 1 only
2 Families 0 and 2 only
3 Families 0 and 3 only
4 Families 0 and 4 only.

buffer-length *(specified by user) (fullword integer)*
The length in bytes of the buffer. ESQUNL may be used to determine the length of the required buffer.

buffer *(returned by GDDM) (character)*
Buffer containing the device names and descriptions. The buffer is a group of variable length elements, one per device nickname. A typical element would be (lengths are all in fullwords)

```
   1    1     variable   1    variable
+-----+----+-----------+----+--------------------+
| FAM | NC |    NL     | DC | DESCRIPTION        |
+-----+----+-----------+----+--------------------+
```

where:
FAM The device-family code (see DSOPEN) of this entry. Nicknames of FAM=0 will always be returned.
NC (fullword integer) Number of fullwords in the following namelist.
NL The namelist information.
DC (fullword integer) Number of fullwords in the following description. Possible values range from 0 to 18.
DESCRIPTION The description information for this nickname. This is supplied on the DESC parameter of the nickname. For further information, refer to the *GDDM Base Application Programming Guide*.

The length of each entry is 3+NC+DC fullwords

Note: Only complete entries are returned in this buffer.

returned-length *(returned by GDDM) (fullword integer)*
The length, in bytes, of the data returned in the buffer. If zero, then NO matches were found using the specified **family**.

Description

Returns a list of devices which have been defined by nickname, together with their associated descriptions.

Principal errors

```
ADM0075 E  INVALID DEVICE FAMILY n
ADM3003 E  LENGTH (n) IS INVALID
```

ESSCPG

Function

To set the code page of a GDDM object.

ESSCPG	**(object-name, type, cpgid)**
APL code	134
GDDM RCP code	X'00110000' (1114112)

Parameters

object-name *(specified by user) (8-byte character string)*
The name of the object. When a CECP Application code page is being used, this name is translated from the Application code page into the Installation code page.

type *(specified by user) (fullword integer)*
The object type. It must be one of:

1	ADMSYMBL	Symbol sets
2	ADMGGMAP	Generated mapgroups
3	ADMSAVE	FSSAVE files
4	ADMCFORM	Chart format files
5	ADMCDATA	Chart data files
6	Not used	
7	ADMGDF	GDF files
8	ADMGKSM	GKS metafiles
9	ADMCDEF	Chart definition files
10	ADMPROJ	Projection definition files
11	ADMIMG	Image data files

cpgid *(specified by user) (fullword integer)*
The global code-page identifier. This must be in the range 0 through 65535. Other than for checking that the value is within range, no validation is performed that the code page global identifier is supported.

Description

Copies the named object, setting the code page global identifier. Normally the copy replaces the original object, but in VM for example, the object may be copied from another disk onto the A-disk.

Principal errors

```
ADM0307 E  FILE 'a' NOT FOUND
ADM0370 E  INVALID OBJECT TYPE - n
ADM0043 E  INVALID CODE PAGE IDENTIFIER n
```

ESSUDS

Function

To specify source-format user default specification.

ESSUDS (length, source-UDS)	
APL code	123
GDDM RCP code	X'00070000' (458752)

Parameters

length *(specified by user) (fullword integer)*
The length in bytes of a source-format UDS. The length must not exceed 32000.

source-UDS *(specified by user) (character)*
The source-format UDS. The first keyword in the UDS is taken to be the UDS-type. Therefore, the UDS must **not** start with a label.

Description

Specifies a single user default specification (UDS) in source format.

For a discussion and examples, refer to the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM0053 E  DEFAULTS ERROR. 'a1' UDS KEYWORD 'a2' IS IN
           CONFLICT
ADM0058 E  DEFAULTS ERROR. INVALID LENGTH n FOR 'a' UDS
ADM0061 E  DEFAULTS ERROR. UDS {TYPE 'a1'| KEYWORD
           'a2'| CODE n} NOT ALLOWED IN a3
ADM0062 E  DEFAULTS ERROR. UDS {TYPE 'a'| CODE n}
           UNKNOWN
ADM0063 E  DEFAULTS ERROR. 'a1' UDS KEYWORD 'a2'
           UNKNOWN
ADM0064 E  DEFAULTS ERROR. VALUE IN {'a1'} UDS {KEYWORD
           'a2'| CODE n} IS INVALID
ADM0065 E  DEFAULTS ERROR. UDS {KEYWORD 'a'| CODE n}
           NOT VALID ON THIS SUBSYSTEM
ADM0066 E  DEFAULTS ERROR. {'a1'} UDS {KEYWORD 'a2'|
           CODE n} HAS TOO MANY OPERANDS
ADM0069 E  DEFAULTS ERROR. 'a1' PROCESSING OPTION 'a2'
           UNKNOWN
```

FSALRM

Function

To sound the terminal alarm.

FSALRM	
APL Code	109
GDDM RCP code	X'0C080000' (201850880)

Parameters

None

Description

Sounds the terminal alarm on the next transmission to the device for the page selected when FSALRM was called.

Principal errors

None

FSCHEK

Function

To check picture complexity before output.

FSCHEK	
APL Code	106
GDDM RCP code	X'0C100002' (202375170)

Parameters

None

Description

Checks the complexity of a picture to discover whether it will cause programmed symbol (PS) overflow (or storage overflow for IPDS printers), at the next FSFRCE, ASREAD, GSREAD, or MSREAD call.

The FSCHEK call is ignored for devices that do not use programmed symbols to display pictures.

The FSCHEK call is not allowed if the primary device is a queued printer and if graphics retrieval is in progress; see GSGET.

Principal errors

```
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0273 W  PS OVERFLOW
ADM0275 W  GRAPHICS {(IMAGE) }CANNOT BE SHOWN. REASON
           CODE n
ADM3282 W  AMOUNT OF DATA EXCEEDS THE STORAGE CAPACITY
           OF THE DEVICE
```

FSCLS

Function

To close alternate device.

Note: This call is not recommended for new programs. It is obsolete and has been superseded by DSCLS.

FSCLS	**(option)**
APL code	601
GDDM RCP code	X'0C180004' (202899460)

Parameters

option *(specified by user) (fullword integer)*
Indicates an action to be carried out at device closure. The meaning of the action is as described under the DSCLS call.

Description

Terminates output to the currently open alternate device. For a queued printer, the intermediate file used to buffer printer output is closed.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0089 W  INVALID OPTION
```

FSCOPY

Function

To send page to alternate device.

FSCOPY	
APL code	602
GDDM RCP code	X'0C180001' (202899457)

Parameters

None

Description

Sends (copies) the current page contents to the currently open alternate device. The size of the copy, in character-cell units, is the same as that of the current page.

The copy is made as follows:

1. The file names specified for currently loaded symbol sets or user-defined pattern or marker sets are noted. These files are loaded before the page copy begins. If the name originally associated with the symbol set ended with the substitution character (see GSLSS), a new device-dependent suffix to the file name is used when the files are loaded for the alternate device. In this way the symbol sets, pattern sets, and marker sets can be tailored to the device in use.

   If a file name for a symbol set was supplied in a call to GSDSS or PSDSS, an attempt is made to load the set for printing. If this cannot be located, the default symbol set(s) are used.
2. Procedural alphanumeric, high-performance alphanumeric, mapped, graphics, and image fields are positioned relative to the top left-hand corner of the copy. Field and character attributes are retained (although some, such as blink, may not be supported by the output device).
3. The graphics picture is mapped onto the corresponding graphics field and the graphics are redrawn. The aspect ratio of the picture can be controlled by using the GSARCC call, but the default is to preserve the aspect ratio of the source, in which case (because of the different cell sizes) the relative positions of items using row/column coordinates and those using graphic coordinates may change.
4. Because graphics text written in mode 1 or mode 2 is dependent on the physical characteristics of the device, its appearance on the output device may differ from that displayed because of the different interpretation of some graphics attributes. Although the standard symbol sets, patterns, line types, and so on have been kept as close as possible on all devices, some differences (for example, the default pattern) have been forced by the characteristics of the hardware.

   If the current page is a **mapped** page, the **name** of the mapgroup is noted. The same mapgroup must be available to the print utility when the copy is printed. The mapgroup suffix is **not** reevaluated.
5. Because the format of graphics images is dependent on the physical characteristics of the device, the appearance on the output device may differ from that displayed because of the pixel aspect ratio.
6. Image data is effectively transferred using an identity projection. Because the target image has “defined” resolution, the resolution flag of the source image determines whether resolution modification or pixel-to-pixel mapping occurs. Overall, the attributes of the source image (such as field, size, or position) determine the

appearance of the target image if the resolution flag is set.

Notes:

1. The FSCOPY call causes a default primary device to be established if one is not already in use.
2. The FSCOPY call is ignored for the IBM 5080 Graphics System.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
ADM3004 E  FIELD LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3005 E  DATA BUFFER n1, ERROR n2 AT INDEX n3
ADM3010 E  BUNDLE LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
```

FSENAB

Function

To enable or disable device input.

FSENAB	**(input-type, control)**
APL code	313
GDDM RCP code	X'0C040E00' (201592320)

Parameters

input-type *(specified by user) (fullword integer)*
The type of input to be enabled or disabled. Possible values are:
1 Alphanumeric
2 Graphic
3 Image
4 CMS asynchronous interrupts (attentions)

control *(specified by user) (fullword integer)*
Specifies whether input is to be enabled or disabled. Possible values are:
0 Disable the specified input type.
1 Enable the specified input type.

Description

Enables or disables input to be entered into the primary device.

Alphanumeric, graphic, and image input can be individually enabled or disabled by this call. Mapping input is considered to be alphanumeric input.

When a device is opened, the initial enablement states are:

Alpha Enabled
Graphics Disabled
Image Disabled.
CMS asynchronous interrupts Enabled.

When an ASREAD or WSIO call is issued, any type of enabled data can be entered.

When a “delayed” GSREAD is issued (that is, with the first parameter = 1), graphics input is implicitly enabled. If necessary, the GSREAD call is performed, and then the previous graphics enablement is restored.

By explicitly enabling the types of input, the application can use the same read call throughout. ASREAD is the most appropriate because it does not affect the input enablement and it returns most information, including PF/PA key and modified alphanumeric field count. GSREAD(0,..,..) can then be used to process the graphics input queue.

When a WSIO call is issued by a window-managing application, the input from all owned virtual devices is solicited and in each virtual device only enabled input types may be entered.

If a window manager has its own windows and needs to obtain graphic input from any of them, it must explicitly enable them for graphics before issuing the WSIO call.

When alphanumerics is disabled, all fields are displayed protected, otherwise they are displayed normally.

When graphics is disabled, echoes are not displayed for any partition in the virtual device.

When image is disabled, the image cursor is not displayed.

Input-type 4 applies only to CMS systems where the current device is the user’s CMS console; in all other cases (MVS, VSE, or not the CMS console) the input-type is valid but the call is ignored. Disabling already disabled interrupts, or enabling already enabled ones has no effect. How enabled asynchronous interrupts are handled is determined by the CMSATTN processing option, specified when the CMS console is opened by GDDM.

An ASREAD call can be issued with no input type enabled. This makes only the attention interrupt key data available.

Principal errors

```
ADM3159 E  INPUT TYPE n IS INVALID
ADM3160 E  CONTROL VALUE n IS INVALID
```

FSEXIT

Function

To specify an error exit, or error threshold, or both.

FSEXIT	**(error-routine, severity)**
APL code	114
GDDM RCP code	X'00030000' (196608)

Parameters

error-routine *(specified by user) (fullword integer)*
The address of the routine to receive control.

Initially, a **default** error exit is set. This displays the error message on the device in use. This error exit is invoked for any error whose severity equals or exceeds:
8 Error (on IMS/VS)
4 Warning (on all other subsystems)

The default error exit can always be reestablished by specifying an error exit address consisting of a **fullword binary** zero.

The default error exit (but not severity) is always reestablished for the processing of the FSTERM call. A user error routine is not given control as the result of errors arising from FSTERM.

The ERRFDBK external default can be used to modify the action of the default error exit; refer to Chapter 18, “External defaults” on page 379.

severity *(specified by user) (fullword integer)*
Gives the minimum severity for which the error routine is to be entered. If a value of zero or less is specified, the error routine is invoked after every call to GDDM; if a severity level of 17 or above is specified, the error exit is never invoked. The severity levels are defined in FSQERR.

The default severity can also be modified by specifying a value for the ERRTHRS external default; refer to Chapter 18, “External defaults” on page 379.

Description

Specifies a user routine to receive control at the end of each call to GDDM if an error of at least the specified severity occurs.

The routine is called exactly as if it had been called from the point at which the GDDM function was invoked. A return from the routine, therefore, gives control to the statement after the one which invoked GDDM.

If an application program is using the nonreentrant interface, a single parameter is passed to the error routine, namely a 160-byte error record, the contents of which are described in FSQERR.

If the reentrant or system-programmer interface is used, two parameters are passed to the error routine. The first of these is the Application Anchor Block (AAB), previously passed by the application program to GDDM. The second is the error record referred to above.

FSEXIT can be called at any time. Subsequent FSEXIT calls override the error routine and severity specifications of the previous FSEXIT.

Note that the sample PL/I declarations described in “PL/I” on page 6 contain a first-parameter descriptor of “*” for FSEXIT, to allow this parameter to be of either type ENTRY or FIXED BINARY (31). The PL/I application programmer must ensure that a first parameter of zero is explicitly passed as a FIXED BINARY (31) value; for example:

```
CALL FSEXIT (BINARY (0,31,0), error-threshold)
```

In COBOL, FSEXIT cannot be used to specify a user error exit. However, it can still be used to specify an error threshold if the error-routine address is specified as 0.

A user error exit, whose address is passed on an FSEXIT call, is assumed to be executable in 31-bit mode if either:

1. The application call is in 31-bit mode, or
2. The top bit of the address passed on the FSEXIT call is set. (For example, the address uses the MVS/XA convention that the top bit of the address identifies its AMODE.)

The first condition enables a high-level language program to pass the address of an exit that is link-edited with itself. (It is difficult (or not possible) to set the top bit of an address in, for example, FORTRAN.)

If a 24-bit application uses a 31-bit user error exit (by setting the top bit of the address), it is the user exit’s responsibility to return control to the application in the correct AMODE, because control returns directly from the exit to the application.

Principal errors

None.

FSFRCE

Function

To perform device output.

FSFRCE	
APL Code	102
GDDM RCP code	X'0C100001' (202375169)

Parameters

None.

Description

Causes all changes that affect the current partition set and have occurred since the last call to ASREAD, GSREAD, MSREAD, or FSFRCE to be reflected on the device. This operation does not affect the status of any modified alphanumeric or mapped fields. The displayed partition set is replaced by any subsequent transmissions to the terminal (resulting from ASREAD, GSREAD, or MSREAD calls, other FSFRCE calls, or non-GDDM processes) as and when they occur. To hold the picture on the screen, ASREAD or GSREAD must be used instead of FSFRCE.

If the primary device is a queued printer (family-2), or a system printer (family-3), the action of FSFRCE is as described under FSCOPY. In this case, the function is not allowed if graphics retrieval is in progress; see GSGET.

For a plotter, pressing the Clear key on the attached workstation while FSFRCE is running cancels the output. However, when running under TSO, this only happens if the DSOPEN processing option 2000 is set to zero.

Principal errors

```
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0233 W  SYMBOL SET IS NOT LOADED
ADM0273 W  PS OVERFLOW
ADM0275 W  GRAPHICS {(IMAGE) }CANNOT BE SHOWN. REASON
           CODE n
ADM0498 E  PRINT TERMINATED. RETURN CODE X'xxxxxx' FROM
           DEVICE
ADM0909 W  NO GRAPHICS FIELD
ADM0911 W  COMPOSED TEXT BLOCK OVERLAPS PAGE BOUNDARY.
           TEXT IGNORED
ADM0920 E  CLEAR KEY PRESSED. PLOTTING IS TERMINATED
ADM2864 W  PICTURE IS TOO LARGE FOR 5080 DISPLAY LIST
           BUFFER
ADM3004 E  FIELD LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3005 E  DATA BUFFER n1, ERROR n2 AT INDEX n3
ADM3010 E  BUNDLE LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3173 W  GRAPHICS CANNOT BE SHOWN.  CELL WIDTH OR
           DEPTH EXCEEDS LOADABLE LIMIT
ADM3178 W  PATTERNS CANNOT BE SENT TO DEVICE.  AREA
           SHADING MAY BE INCORRECT
ADM3179 W  IMAGE CANNOT BE SHOWN. REASON CODE n
ADM3281 W  GRAPHICS MAY BE VISIBLE WITHIN OPAQUE
           ALPHANUMERIC FIELDS
ADM3282 W  AMOUNT OF DATA EXCEEDS THE STORAGE CAPACITY
           OF THE DEVICE
```

FSGET

Function

To retrieve Family-4 datastream.

FSGET	**(buffer, Data_length)**
APL code	621
GDDM RCP code	X'0C10000A' (202375178)

Parameters

buffer *(returned by GDDM) (character)*
A data area, of length at least 8202 bytes, to receive the Family-4 datastream record.

record-length *(returned by GDDM) (fullword integer)*
A variable that is set to the length of the output record. If it is zero then all of the records have been returned.

Description

The records are retrieved sequentially as FSGET calls are issued. The records are in a format corresponding to the device token and the PROCOPT specification for OFFORMAT(HRIFORMT). They are identical to those which would have been written to an output dataset using FSFRCE.

Calls to FSGET must have been preceded by a call to FSGETS.

The supplied buffer must be large enough to accept the longest possible record which is 8202 bytes. GDDM is unable to check the length of your buffer, and if it is insufficient, storage overwrite may occur.

Principal errors

```
ADM0286 E  DATASTREAM RETRIEVAL NOT INITIALIZED
```

FSGETE

Function

To terminate Family-4 buffered output.

FSGETE	
APL Code	622
GDDM RCP code	X'0C10000B' (202375179)

Parameters

None.

Description

Causes early termination of retrieval of Family-4 datastream, following an FSGETS call with or without intervening FSGET calls. The call is not required if retrieval is completed to end of data but will not fail if issued once at that time. The purpose of the call is to release storage resources if for any reason it is not required to complete retrieval.

Principal errors

ADM0286 E DATASTREAM RETRIEVAL NOT INITIALIZED

FSGETS

Function

To start Family-4 datastream retrieval.

FSGETS	
APL Code	620
GDDM RCP code	X'0C100009' (202375177)

Parameters

None.

Description

Causes the current page to be made available for family-4 datastream retrieval via the FSGET call. The call is only valid when the currently open primary device is Family-4. It is designed to be used with the specification of a *dummy* device on the **DSOPEN** call. That is by specifying *blanks* in the *name* parameter. Without a *dummy* device the call will also create an output dataset as if an **FSFRCE** call had been issued.

This function is not supported for **PostScript** output.

FSGETS causes the creation of an internal chain of buffers containing the datastream records in storage, ready for retrieval by the FSGET call. This may cause problems where there are storage limitations. To reduce the size of the datastream it is recommended that the procopt OFFORMAT specifies the GRIMAGE or GRCIMAGE option. This may be implied by the choice of device token.

If the OFDSTYPE(CDPFTYPE) procopt specifies DOC(PRIM) output then a call to FSGETS must be made after each page has been constructed by the application program. The records for each page must be retrieved using FSGET calls following every call to FSGETS. When formatted output has been requested with the OFFORMAT(HRIFORMT) procopt then GDDM inserts a BEGIN DOCUMENT (BDT) record in front of the first page. As it is not known at the time FSGETS is issued, which page is the last, an END DOCUMENT (EDT) record is never made available for FSGET.

This function is not intended for use with color separation masters.

Principal errors

ADM0287 E DATASTREAM RETRIEVAL ALREADY INITIALIZED

FSINIT

Function

To initialize GDDM processing.

FSINIT	
APL Code	117
GDDM RCP code	X'0C000001' (201326593)

Parameters

None

Description

Initializes GDDM processing.

This call (in the form above or one of the other forms below) must be the first GDDM statement to be executed, unless the system programmer interface is being used; see “The system programmer interface” on page 2.

This call is not usually required in GDDM-REXX programs. If you do use it, it initializes a new instance of GDDM within the instance of GDDM-REXX, thereby making the program reentrant.

The initialization process creates a control table holding various anchor pointers, and sets GDDM variables to their initial values.

Other forms for CICS, IMS, and TSO: To simplify link-editing, other forms of the FSINIT call exist, specific to the external interface being used and to some of the subsystem environments; for example, CICS, IMS, and TSO. The other forms simplify the use of linkage-editor automatic library call facilities to resolve the required interfaces.

For a discussion on link-editing a GDDM application program, refer to the *GDDM Base Application Programming Guide*.

The other forms of FSINIT are:

Nonreentrant interface:

```
CICS:              FSINNC
IMS:               FSINNI (non-PL/I),
                   FSINNPI (PL/I)
TSO:               FSINN
```

Reentrant Interface:

```
CICS:              FSINRC
IMS:               FSINRI (non-PL/I),
                   FSINRPI (PL/I)
TSO:               FSINR
```

Therefore, a user writing a reentrant application program for TSO can specify:

```
CALL FSINR(AAB);
```

A user writing a nonreentrant application program for CICS can specify:

```
CALL FSINNC;
```

Note: The sample PL/I declarations (such as ADMUPINF) do **not** include these other entry points. The PL/I application programmer must, therefore, supply an entry-point declaration for the particular form (FSINxx) to be used, as described in “PL/I” on page 6. For example:

```
DCL FSINNC EXTERNAL ENTRY OPTIONS(ASM INTER);
```

Principal errors

```
ADM0002 E  GDDM IS ALREADY INITIALIZED
```

FSLOG

Function

To send character string to alternate device.

FSLOG	**(length, string)**
APL code	603
GDDM RCP code	X'0C180003' (202899459)

Parameters

length *(specified by user) (fullword integer)*
The length of **string**. The maximum length is as defined (or defaulted) on the FSOPEN or DSOPEN statement, the default for which is 80. Specifying **length**=0 leaves a blank line in the output.

string *(specified by user) (character)*
The string of EBCDIC characters to be sent.

SO/SI (shift-out/shift-in) control-code characters can be included in the string and are sent to the device as “passthrough”; that is, they are not processed in any way. A mixed string of single and double-byte characters can, therefore, be presented, provided the alternate device supports SO/SI control codes as a delimiter of mixed strings, and the SO/SI control codes are correctly paired in the string. Neither of these conditions are checked by GDDM.

Description

Sends (copies) the specified string to the currently open family-1, family-2, family-3, or family-4 cell-based alternate device.

FSLOG calls can be mixed with FSLOGC, FSCOPY, and GSCOPY calls within a single open–close session. Consecutive FSLOG and FSLOGC data is batched together and sent separately from any FSCOPY and GSCOPY output.

Note: The FSLOG call does not require a primary device to be available, and so does not cause a default one to be set up.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0282 E  INVALID LOG DATA LENGTH
```

FSLOGC

Function

To send character string with carriage-control character to alternate device.

FSLOGC	**(length, string)**
APL code	606
GDDM RCP code	X'0C180005' (202899461)

Parameters

length *(specified by user) (fullword integer)*
The length of **string**. The maximum length is 1 greater than the length defined (or defaulted) in the FSOPEN or DSOPEN call, the default for which is 80.

string *(specified by user) (character)*
The string of EBCDIC characters to be sent. The first character is interpreted according to the values in the following table; an invalid carriage-control character is interpreted as X'09'.

	CTLASA code carriage-control characters	**CTL360 code bytes carriage-control characters**	
Action	**(Action before printing)**	**(Action after printing)**	**(Action without printing)**
No line	+	X'01'	X'03'

Action	CTLASA code carriage-control characters	CTL360 code bytes carriage-control characters	
	(Action before printing)	(Action after printing)	(Action without printing)
Space 1 line	blank	X'09'	X'0B'
Space 2 lines	0	X'11'	X'13'
Space 3 lines	-	X'19'	X'1B'
Space to new page	1	X'89'	X'8B'

SO/SI (shift-out/shift-in) control-code characters can be included in the string and are sent to the device as "passthrough"; that is, they are not processed in any way. A mixed string of single and double-byte characters can, therefore, be presented, provided the alternate device supports SO/SI control codes as a delimiter of mixed strings, and the SO/SI control codes are correctly paired in the string. Neither of these conditions are checked by GDDM.

Description

Sends (copies) the specified string to the currently open family-1, family-2, family-3, or family-4 cell-based alternate device. The first character of the string is interpreted as a carriage-control character.

Overstriking characters are processed in the following way. If several strings are to be printed on the same line:

- The end-result is underscored if an underscore occurs in any string for a specific print position.
- The end-result is blank if a blank (or a null character such as end-of-string) occurs in **all** the strings for a specific print position.
- Otherwise, the end-result for a specific print position is the **first** non-blank character (except underscore) in the strings for that position. For example, if the following two lines were to be printed in the **same** positions:

```
A   B
 DC X
```

  The line that would be printed would be:

```
ADC B
```

  FSLOGC calls can be mixed with FSLOG, FSCOPY, and GSCOPY calls within a single open–close session. Consecutive FSLOGC and FSLOG data is batched together and sent separately from any FSCOPY and GSCOPY output.

Note: The FSLOGC call does not require a primary device to be available, and so does not cause a default one to be set up.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0282 E  INVALID LOG DATA LENGTH
```

FSOPEN

Function

Note: This call is not recommended for new programs. It is obsolete and has been superseded by DSOPEN.

To open alternate device.

FSOPEN (destination, count, array)	
APL code	604
GDDM RCP code	X'0C180000' (202899456)

Parameters

destination *(specified by user) (8-byte character string)*
The alternate device destination as an eight-character string (left-justified). The interpretation of the string depends on the subsystem, and is as described under DSOPEN for the first element of **name-list**.

count *(specified by user) (fullword integer)*
The number of fullwords in **array**. It may be zero if all of the print options are to be defaulted.

array *(specified by user) (an array of fullword integers)*
An array with a maximum significant length of nine fullwords. If the first fullword is present, it is the parameter content type, and must be zero. The remaining fullwords, if present, comprise up to eight print-control processing options (option-group 4), from the heading indicator to the alphanumeric device type for translation. These are described under DSOPEN.

If fewer than the maximum number of words are supplied, the remainder are assigned default values.

Description

Initializes output to the specified family-2 alternate device. GDDM automatically opens a queued printer (family 2) device, with a device identifier of 1, a default device token '★ ', the specified destination as the only element in the name list, and processing options containing the specified array in the print control option group (4). It also automatically makes this the alternate device.

FSOPEN is retained for compatibility. It is a Version 1 Release 1 call whose functions have been duplicated and extended by DSOPEN.

Principal errors

```
ADM0071 E  INVALID PARAMETER COUNT
ADM0072 E  INVALID PARAMETER CONTENT TYPE
ADM0073 E  ALTERNATE DEVICE ALREADY OPEN
ADM0077 E  DEVICE ALREADY EXISTS
ADM0281 E  INVALID PARAMETER ARRAY.  REASON CODE n
```

FSPCLR

Function

To clear the current page.

FSPCLR	
APL Code	301
GDDM RCP code	X'0C040003' (201588739)

Parameters

None

Description

Deletes all objects from the current page.
This includes:

- Graphics fields.
- Image fields.
- Procedural alphanumerics fields.
- Maps (including the floating area).
- High-performance alphanumerics fields (field lists).

Any panning or zooming that has been performed on the picture (using User Control) is reset.

Principal errors

None

FSPCRT

Function

To create a page.

FSPCRT	**(page-id, depth, width, type)**
APL code	302
GDDM RCP code	X'0C040000' (201588736)

Parameters

page-id *(specified by user) (fullword integer)*
The identification of the new page. It must be greater than zero and unique within the current partition. Zero is reserved for the identification of the default page, which is always available.

depth *(specified by user) (fullword integer)*
The depth of the page, in rows, starting at 1 for the top-most row. If this is zero, the appropriate depth (according to the area occupied by the current partition) is used.

width *(specified by user) (fullword integer)*
The width of the page, in columns, starting at 1 for the left-most column. If this is zero, the appropriate width (according to the area occupied by the current partition) is used. For all devices, except roll-feed plotters and non-cell-based family-4 devices, the maximum page width is 255 columns.

type *(specified by user) (fullword integer)*
Values 0 through 3 are retained for compatibility with previous releases. These values are ignored; all other values are invalid.

For plotters, the number of characters across and down the page depends on the paper size and is independent of the size of the plotting area, as follows:

Paper size	Depth	Width
A4 or A	32	80
A3 or B	45	113
A2 or C	64	160
A1 or D	90	226
A0 or E	128	320

These values can be queried by using the FSQURY call.

For plotters, except roll-feed plotters, the width and depth must not be greater than the size of the plot area, and the corresponding number of plotter units must be less than 65536. For roll-feed plotters, either the width or, if plot rotation is in force, the depth may exceed the default page width or depth.

For IPDS printers, the width and depth must not exceed the current values as set through the operator panel, or the values contained in the device token that is being used.

For devices other than family-1 devices, the width must not be greater than the width of the current partition. The product of the width and depth must not be greater than 16000 for family-1 devices, or 32000 for family-3 devices.

Description

Creates a page of the specified size belonging to the device (or current partition, if partitions are in use). Usually the page size matches the partition (or the printer page), but when you create the page, you can request a size that is smaller than the partition (or printer page) or, on display devices, larger than the partition. The new page is empty.

The entire depth or width of the page need not be able to be displayed within the screen or current partition. Different sections of the page are visible, according to the setting of the page window; see FSPWIN.

If you do not specify the number of rows or columns, or both, in a page, GDDM defaults to device-dependent numbers. These are such that, if you do not specify a page window depth or width, the resulting cell size depth or width is the default for the device.

For family-4 devices, the effect of the page width and depth specification on the FSPCRT call depends on the device token in use:

- For AFPDS tokens which are cell-based, the depth and width values allow the page to be defined in alphanumeric rows and columns. Graphics, image, and alphanumeric fields are subsequently defined on the page in alphanumeric rows and columns, just the same as for family-1 and family-2 devices.
- For other family-4 device tokens (which do not specify cell sizes), the width and depth specified on the FSPCRT call divide the available paper area into a grid. The row spacing in this grid is given by the paper length divided by the depth specified on FSPCRT, and the column spacing by the paper width divided by the width specified on FSPCRT. Graphics and image fields are subsequently defined on the page in terms of this grid. If no FSPCRT call is issued, the row and column spacing defaults to pixels.

The new page becomes the one currently selected; see FSPSEL.

To create a page for mapping, use the MSPCRT call.

If you use a device with primary and alternate screen sizes (for example, the 3278 Model 5), you can select the required size in the FSPCRT call. Note that in order to obtain the primary screen size, you must suppress User Control, using the CTLMODE procopt, for example, with the nickname:

```
ADMMNICK PROCOPT=((CTLMODE,NO))
```

In order to obtain the primary screen size, use of the WINDOW procopt, partition calls and partition set calls must also be avoided.

Principal errors

```
ADM0130 E  PAGE n ALREADY EXISTS
ADM0131 E  PAGE TYPE n IS INVALID
ADM0134 E  PAGE IDENTIFIER n IS INVALID
ADM0137 E  PAGE SIZE n IS INVALID
ADM0138 E  PAGE DEPTH n1 OR WIDTH n2 IS TOO LARGE
ADM3155 E  PAGE n1 MAXIMUM NUMBER OF CHARACTERS (n2)
           EXCEEDED
```

FSPDEL

Function

To delete a page.

FSPDEL	**(page-id)**
APL code	303
GDDM RCP code	X'0C040002' (201588738)

Parameters

page-id *(specified by user) (fullword integer)*
Identifies the page to be deleted.

Description

Deletes a page. This causes all objects on the page to be deleted. See FSPCLR.

If this page was the current page, the default page becomes the new current page.

The default page (identifier zero) cannot be deleted.

Principal errors

```
ADM0132 E  PAGE n DOES NOT EXIST
ADM0133 E  ATTEMPT TO DELETE DEFAULT PAGE
```

FSPQRY

Function

To query specified page.

FSPQRY	**(page-id, depth, width, type)**
APL code	304
GDDM RCP code	X'0C040004' (201588740)

Parameters

page-id *(specified by user) (fullword integer)*
The page about which information is required.
depth *(returned by GDDM) (fullword integer)*
The depth of the page, in rows.

width *(returned by GDDM) (fullword integer)*
The width of the page, in columns.
type *(returned by GDDM) (fullword integer)*
The page type, as defined for FSPCRT.

Description

Returns information about the specified page. The **depth** and **width** parameters are returned exactly as they could have been specified in an FSPCRT call to create this page; the real values are returned, even if they were defaulted when the page was created.

Principal errors

```
ADM0132 E  PAGE n DOES NOT EXIST
```

FSPSEL

Function

To select a page.

FSPSEL	**(page-id)**
APL code	305
GDDM RCP code	X'0C040001' (201588737)

Parameters

page-id *(specified by user) (fullword integer)*
The page to be selected. A page identifier of zero causes the **default** page to be selected.

Description

Selects a page. This makes the named page the current one, and has two effects:

- When the device is updated, the page that is current at that time appears
- Any subsequent alphanumeric field creation or reference, mapped field creation or reference, or graphics field creation or reference, is associated with this page.

Only one page can be selected at any time, and the specified page remains selected until it is deleted, or another page is explicitly selected, or a new page is created.

If a page is deselected (for example, by creating another page) and then reselected, the contents of the page are unchanged. In particular, any previously open graphics segment will still be open.

Principal errors

```
ADM0132 E  PAGE n DOES NOT EXIST
ADM3155 E  PAGE n1 MAXIMUM NUMBER OF CHARACTERS (n2)
           EXCEEDED
```

FSPWIN

Function

To set page window.

FSPWIN	**(row, column, depth, width)**
APL code	309
GDDM RCP code	X'0C040C00' (201591808)

Parameters

row *(specified by user) (fullword integer)*
The new page window row number, or –1 (to leave the existing value unchanged). Must be 1 through page depth, or –1.
column *(specified by user) (fullword integer)*
The new page window column number, or –1 (to leave the existing value unchanged). Must be 1 through page width, or –1.
depth *(specified by user) (fullword integer)*
The new page window depth in rows, or –1 (leaving the existing value unchanged), or 0 (to indicate the default depth). Whenever the page window depth is changed, the page window row number is checked for correctness and reduced if necessary (see above).
width *(specified by user) (fullword integer)*
The new page window width in columns, or –1 (to leave the existing value unchanged), or 0 (to indicate the default width). A value other than –1 can only be specified immediately after FSPCRT, in which case this value must be not less than the page width. The value must be such that, using the minimum character-box width, this number of characters can be displayed side-by-side within the current partition.

Description

Sets the origin and size of a page window, or alters the origin of a page window.

The dimensions of the page window can be set explicitly only once, before any data has been placed on the page. The dimensions can be changed implicitly during a PTNMOD call.

The page window is a rectangular area covering some part of the page. It determines how much of the page is seen by the operator in the partition to which the page belongs.

The row value must be between 1 and the page depth. The row that is actually positioned at the top of the partition is given by the algorithm:

```
min(row,page.depth-window.depth+1)
```

This ensures that there is never any blank space following the last line of the page. The cursor is moved, if necessary, so that it is inside the new page window.

The column value must be between 1 and the page width. The column that is actually positioned at the top of the partition is given by the algorithm:

```
min(col,page.width-window.width+1)
```

On devices that support variable size cells (such as the 3290 display), the value of the size of the page window is used to select the cell size used when displaying the page. A large page window width and depth can be used to get a small cell size. A small page window depth and width can be used to get a large cell size. The largest cell size that you can get with a call to FSPCRT is the loadable size. To get a cell size between this size and the maximum, you need to create a page window with FSPWIN. For more information on variable cell sizes, refer to the *GDDM Base Application Programming Guide*.

The page window depth can be implicitly altered by PTNMOD, for devices on which partitions are supported. Altering the size of the partition does not change the cell size used for any of the pages that it contains. The row positioned at the top of the partition is determined by the algorithm given above.

The rules for setting column and width are similar to those for setting row and depth; that is, the column value and the width should not cause the window boundaries to go outside the page boundaries. If necessary, the cursor is moved so that it is inside the new page window at the top-left corner.

Notes:

1. GDDM is unaware of changes to the scroll position caused by hardware scrolling by the operator. The value used in the above algorithm is the value last known to GDDM.
2. When an image cursor is enabled at the time of the I/O call (ASREAD, FSFRCE), and hardware scrolling is being used, the device may alter the window position to ensure that the image cursor is visible.

Principal errors

```
ADM3150 E  PAGE n WINDOW SIZE CANNOT BE ALTERED
ADM3151 E  PAGE n1 WINDOW DEPTH (n2) IS INVALID
ADM3152 E  PAGE n1 WINDOW WIDTH (n2) IS INVALID
ADM3153 E  PAGE n1 WINDOW ROW (n2) IS INVALID
ADM3154 E  PAGE n1 WINDOW COLUMN (n2) IS INVALID
ADM3156 I  PAGE n1 WINDOW ROW ALTERED TO n2 AND COLUMN
           TO n3
```

FSQCPG

Function

To query current page identifier.

FSQCPG (page-id)	
APL code	306
GDDM RCP code	X'0C040005' (201588741)

Parameters

page-id *(returned by GDDM) (fullword integer)*
The identifier of the current page.

Description

Returns the identifier of the current page.

Principal errors

None.

FSQDEV

Function

To query device characteristics.

Note: This call is retained for compatibility with previous releases; it should not be used in new programs. It is recommended that the FSQURY call is used instead.

FSQDEV (count, array)	
APL code	110
GDDM RCP code	X'0C040500' (201590016)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in **array**. If fewer elements are supplied than are required, information about the remaining attributes is not returned. If more than the required number are supplied, those in excess are set to zero.

array *(returned by GDDM) (an array of fullword integers)*
Receives information about the device. This is described in the FSQURY call under the information for code=0.

Description

Returns the characteristics of the current primary device.

Principal errors

```
ADM0129 E  ARRAY COUNT n IS INVALID
```

FSQERR

Function

To query last error.

FSQERR (length, array)	
APL code	107
GDDM RCP code	X'00040000' (262144)

Parameters

length *(specified by user) (fullword integer)*
The length in bytes of the storage provided to receive the error data. The number of bytes of information provided is that specified in the length parameter, or 160, whichever is the smaller.

array *(returned by GDDM) (character)*
The data describing the error. The information is formatted as follows:

Error Record Structure The error record has a length of 160 bytes, and contains mixed integer and textual information. The format is as follows:

Severity (offset 0)
A fullword binary integer denoting the error severity. Possible values are:

0 Informative (or no error)
4 Warning
8 Error (function call ignored)
12 Severe error (resultant state unpredictable)
16 Irrecoverable (not passed to an error exit)

Error number (offset 4)
A fullword binary integer number identifying the error. The numbers correspond to the error-message numbers (listed in the *GDDM Messages* book) without the three-letter prefix. The number is zero if no error occurred.

The function name (offset 8)
Two fullwords (eight characters) containing the name of the function whose invocation caused the error. If the error exit threshold is zero or less, the function name is that of the GDDM function called. For FSQERR, the field contains blanks if no error occurred since the last call to FSQERR, or if there have been no errors since initialization.

Message length (offset 16)
A fullword binary integer containing the length (in characters) of the message, excluding trailing blanks. The maximum length is 80; it is zero if no error occurred.

Message text (offset 20)
The text of the error message associated with the error number, padded with trailing blanks if necessary to fill in the 80-character length.

The entry-point function code (offset 100)
A fullword binary integer (the request control parameter or RCP code) representing the GDDM function invoked. (For a list of RCP codes, refer to the *GDDM Diagnosis* book). If the error-exit threshold is zero or less, the RCP code is that of the last GDDM function called. For FSQERR, the number is zero if no error occurred since the last call to FSQERR, or if no error occurred since initialization.

Parameter list pointer (offset 104)
A fullword containing the contents of Register 1 at the point of call. This pointer makes it possible to obtain the parameters for the call that generated the error. If the error-exit threshold is zero or less, the information in this field is that for the last GDDM function called. Note that this value is not necessarily meaningful if the parameter-list area was reused by the application program; this can happen if FSQERR is invoked.

Return address (offset 108)
A fullword containing the contents of Register 14 at the point of call. This makes it possible to obtain the storage address of the call that generated the error. If the error-exit threshold is zero or less, the information refers to the last GDDM function called.

Arithmetic insert 1 (offset 112)
A fullword binary integer or short floating point number whose content depends on the error. Inserts are indicated in the appropriate error messages, which are listed and described in the *GDDM Messages* book. This word is zero if no error occurred.

Arithmetic insert 2 (offset 116)
See the description of “arithmetic insert 1” above.

Character insert 1 (offset 120)
Five fullwords (20 characters); the contents depend on the error, as noted for the arithmetic inserts. The field is blank if no error occurred.

Character insert 2 (offset 140)
See the description of “character insert 1” above.

Description

Returns information about the last error, that is, the last GDDM call whose returned severity code was nonzero. If no error other than an informative severity error occurred since initialization or since the last call to FSQERR, the value zero is returned.

Principal errors

None.

FSQSYS

Function

To query systems environment.

FSQSYS (count, array)	
APL code	122
GDDM RCP code	X'00060000' (393216)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in **array**. If fewer elements are supplied than are required, information about the remaining attributes is not returned. If more than the required number of elements are supplied, the excess elements are undefined.

array *(returned by GDDM) (array of 8-byte character tokens)*
System environment information, arranged as follows:

1. The identifier of the GDDM **resident** version/release (that is, the release of GDDM that has been link-edited with the application program). These values are returned (x, y, and z are all integers):

   `'VxRy.z '` if the release number is less than 10.
   `'VxRyy.z '` if the release number is equal to or greater than 10.

2. The identifier of the GDDM **transient** version/release (that is, the release of GDDM that is being dynamically loaded). These values are returned:

   `'VxRy.z '` if the release number is less than 10.
   `'VxRyy.z '` if the release number is equal to or greater than 10.

3. The subsystem environment. One of these values can be returned:

   `'CICS '`
   `'IMS '`
   `'TSO '`
   `'CMS '`
   `'MVS '`
   `'VSE '`

4. The subsystem qualifier. These values can be returned:

   `'MVS '` or
   `'VSE '` (for CICS/VS)
   `'BATCH '` (for TSO Batch)
   `' '` (for TSO Interactive)
   `'BATCH '` (for MVS Batch)
   `'BATCH '` (for VSE Batch)

   The value returned for IMS/VS or VM/CMS is undefined.

Description

Returns information about the current GDDM and subsystem environment.

Principal errors

`ADM0068 E INVALID ARRAY COUNT OF n SPECIFIED`

FSQUPD

Function

To query update mode.

FSQUPD (control)	
APL code	663
GDDM RCP code	X'0C0C1A01' (202119681)

Parameters

control *(returned by GDDM) (fullword integer)*
The current setting of Update Mode. Possible values are:

–1 Optimized update mode has no effect on the current device.
0 Not optimized.
1 Optimized by a GDDM-chosen method.

Description

Returns the current setting of the update mode.

The update mode is set by the FSUPDM call, or by a processing option when the device is opened, or by User Control.

This call can be used by an application to find out the default mode of processing so that it can restore this mode when temporarily entering an update mode.

Principal errors

None.

FSQUPG

Function

To query unique page identifier.

FSQUPG (page-id)	
APL code	307
GDDM RCP code	X'0C040900' (201591040)

Parameters

page-id *(returned by GDDM) (fullword integer)*
An identifier for which no page currently exists.

Description

Requests that a unique, unused page identifier is returned. This call may be used by a modular application program to obtain an identifier for a new page, without conflicting with a page already created by another part of the application program. The page identifier returned is the highest available unused number.

Principal errors

None

FSQURY

Function

To query device characteristics.

FSQURY (code, element-no, count, array)	
APL code	121
GDDM RCP code	X'0C040501' (201590017)

Parameters

code *(specified by user) (fullword integer)*
Specifies the type of information to be returned in **array**. Possible values are:

- **0** The characteristics of the device
- **1** The partition characteristics of the device
- **2** The graphics characteristics of the device
- **3** The characteristics of attached plotters
- **4** The characteristics of the image display
- **5** The characteristics of the image scanner
- **6** The characteristics of the image cursors

element-no *(specified by user) (fullword integer)*
Gives the index number of the first item of information to be returned in the **array** parameter.

count *(specified by user) (fullword integer)*
Gives the number of items of information to be returned in the **array** parameter.

array *(returned by GDDM) (an array of fullword integers)*
Returns the device characteristics information. The first item returned is set into the first element of the array, the second into the next, and so on. In the tables that follow, AE stands for “Array element.”

When **code=0**, the device characteristics information is returned:

AE ***Device information***

1 The device family code, which can take these values:

- **0** 3277
- **1** 3270-family devices
- **2** Queued printer files
- **3** System printer files
- **4** Page-printer and PostScript files

Note: The device family returned for a 5080 Graphics System depends on the 3270 display associated with the 5080 at the time the device is opened.

2 Device input/output capability:

- **1** Dummy device
- **2** Device supports input and output
- **3** Device supports output only

3 Default (nonscrollable) page depth in rows (screen depth for displays).

Note: For family-4 devices that are defined by non-cell-based device tokens, page depth and width are defined in pixels.

4 Default page width in columns (screen width for displays).

5 Default character-box depth (in display points for a display or a printer, or plotter units for a plotter). The value is defined in pixels for family-4 devices. For family-4 devices that are defined by non-cell-based device tokens, the default is 12 points (one sixth of an inch).

Note: A “plotter unit” is the smallest possible displacement of a pen on a plotter, and can be smaller than a “pen width.”

6 Default character-box width (in display points for a display or a printer, or plotter units for a plotter). The value is defined in pixels for family-4 devices. For family-4 devices that are defined by non-cell-based device tokens, the default is half the default character-box depth.

7 Vertical resolution in display points per meter for displays, plotter units per meter for plotters, or pixels per meter for family-4 devices.

8 Horizontal resolution in display points per meter for displays, plotter units per meter for plotters, or pixels per meter for family-4 devices.

Some alphanumeric-only devices that do not allow access to the screen on a pixel-by-pixel basis (such as the 3180), do not return the horizontal and vertical resolution of the screen to GDDM. In these cases,

GDDM may return nominal horizontal and vertical pixel resolutions that will not necessarily match the observed characteristics of the device. Applications can find out whether a device is of this type, through analysis of the "output class supported" field returned by FSQURY code=2.

9 Whether character attributes can be entered from the keyboard:

- **0** Can not be entered
- **1** Can be entered, if the keyboard feature is installed

Note: GDDM cannot detect whether the appropriate keyboard feature is actually installed or not.

10 Number of PS stores available.

11 Whether an APL feature (APL/TEXT or Data Analysis – APL, or APL2) is available:

- **0** No APL feature is available
- **1** APL feature is available
- **2** APL2 feature is available

12 Number of alphanumeric colors supported:

- **1** Monochrome devices
- **7** Seven-color display, 4224 printer

Note: The value returned for an ASCII display or four-color 3279 or 3287 is 1. If a color-master table is being used by a family-4 page printer, the number of colors is equal to the number of colors defined in the selected color-master table. For plotters, the number of colors equals the number of pens.

13 Availability of highlight feature:

- **0** No highlight feature
- **1** Highlight feature available

14 Alphanumeric device type for translation purposes; see ASTYPE.

15 Exceptional device:

- **0** Normal (not one of the following)
- **2** Device is an IBM 8775
- **3** Auxiliary device; a plotter
- **4** Auxiliary device; a printer attached to a device using GDDM-PCLK
- **5** Auxiliary device; a printer, plotter, or other hard-copy device attached to a device using GDDM-OS/2 Link

16 Background transparency support

- **0** No transparency support
- **1** Transparency support

17 Mixed (EBCDIC/DBCS) alphanumeric field support:

- **0** No support for mixed strings in alphanumeric fields.
- **1** Explicit support (by device) for mixed strings in any alphanumeric field. This means that you can use the DBCS nonloadable symbol set.
- **x** Emulated support (by GDDM) for mixed strings in alphanumeric fields defined as "mixed" (see ASFSEN), where **x** is the code-point corresponding to the emulation character.

18 Field outlining support:

- **0** No outline support
- **1** Outline support

19 Identifier of the device's code page for single-byte characters.

20 Identifier of the device's single-byte character set.

21 Identifier of the device's code page for double-byte characters, if applicable. Otherwise 0.

22 Identifier of the device's double-byte character set, if applicable. Otherwise 0.

When **code=1**, the partition characteristics information is returned:

AE ***Device information***

1 The number of real partitions supported by the device.

2 The number of emulated partitions that can be created on the device.

If 0 is returned in both array elements 1 and 2, no partition operations are possible.

The hardware partition identifiers that can be set with the PTNCRT call must be in the range 0 through the value returned in array element 1 minus 1.

3 The vertical scrolling characteristic of the device. Possible values are:

- **0** Scrolling is not allowed
- **1** Scrolling is allowed and is software controlled
- **2** Scrolling is allowed and is hardware controlled

4 The horizontal scrolling characteristic of the device. Possible values are:

- **0** Scrolling is not allowed
- **1** Scrolling is allowed and is software controlled
- **2** Scrolling is allowed and is hardware controlled

5 The total amount of storage in the scroll buffer of the device that is available to hold pages that are to be scrolled. This value is in bytes and is called M, below.

6 The amount of scroll buffer storage required as overhead for each row of a page. This value is in bytes and is called R, below.

7 The amount of scroll buffer storage required as overhead for each column of a page. This value is in bytes and is called C, below.

8 The amount of scroll buffer storage required as overhead for each position on the page. This value is in bytes and is called P, below.

GDDM rejects any attempt to create a page such that the sum of

```
no-of-rows*R + no-of-cols*C +
    no-of-rows*no-of-cols*P
```

over all the current pages in the partitions of the current partition set is greater than M.

9 The depth, in pixels, of that area of the screen available to GDDM. For a device with real partitions and a partition set with the partition control value set by the PTSCRT call to 0, this corresponds to the maximum depth allowed for the partition set grid. For any other device, or for other partition sets, the value returned is the same as for array element 27 of this group.

10 The width, in pixels, of that area of the screen available to GDDM. For a device with real partitions and a partition set with the partition control value set by the PTSCRT call to 0, this corresponds to the maximum width allowed for the partition set grid. For any other device, or for other partition sets, the value returned is the same as for array element 28 of this group.

11 The depth of the default grid.

12 The width of the default grid. The default grid is the grid that results in the device default character size being used.

13 The depth of the minimum grid.

14 The width of the minimum grid. The minimum grid is the grid that results in the largest characters supported by the device being used.

15 The depth of the maximum grid.

16 The width of the maximum grid. The maximum grid is the grid that results in the smallest characters supported by the device being used.

17 The depth of the minimum grid that can be used for graphics.

18 The width of the minimum grid that can be used for graphics.

19 The depth, in pixels, of the default cell size for the device.

20 The width, in pixels, of the default cell size for the device.

21 The depth, in pixels, of the largest cell size supported by the device.

22 The width, in pixels, of the largest cell size supported by the device.

23 The depth, in pixels, of the smallest cell size supported by the device.

24 The width, in pixels, of the smallest cell size supported by the device.

25 The depth, in pixels, of the maximum cell size that can be used for graphics on the device.

26 The width, in pixels, of the maximum cell size that can be used for graphics on the device.

27 The depth, in pixels, of that area of the screen available to GDDM when using emulated partitions. For a device with no real partitions, or a partition set with the partition control value set by the PTSCRT call to 1 or 2, this corresponds to the maximum depth allowed for the partition set grid.

28 The width, in pixels, of that area of the screen available to GDDM when using emulated partitions. For a device with no real partitions, or a partition set with the partition control value set by the PTSCRT call to 1 or 2, this corresponds to the maximum width allowed for the partition set grid.

When **code=2**, the graphics characteristics information is returned:

AE ***Device information***

1 Output class or classes supported. Possible values are:

- **1** Alphanumerics only
- **2** Graphics only
- **3** Alphanumerics and graphics overlaid
- **4** Alphanumerics and graphics separated

2 Graphics class. Possible values are:

- –**1** Undefined (if the output class value is 1)
- **0** Output only (for example, printers and plotters)
- **1** Interactive

3 Maximum horizontal size of the graphics field (in pixels).

4 Maximum vertical size of the graphics field (in pixels).

5 Number of pixels per meter horizontally (rounded to the nearest integer).

6 Number of pixels per meter vertically (rounded to the nearest integer).

7 Default cell-size width for graphics (in pixels).

8 Default cell-size height for graphics (in pixels).

9 Number of available colors or gray-scales (including background). For page printers with a color master table specified in procopt group 3000 of the DSOPEN call, this value identifies the number of entries in the color master table.

For plotters, the returned value is the number of pens plus 1 (for the background).

10 Not used.

11 Maximum choice device identifier supported. A returned value of –1 indicates that no choice device is available. To query the availability of a specific choice device, use the GSQLID call.

12 Maximum locator device identifier supported. A returned value of –1 indicates that no locator device is available. To query the availability of a specific locator device, use the GSQLID call.

13 Maximum pick device identifier supported. A returned value of –1 indicates that no pick device is available. To query the availability of a specific pick device, use the GSQLID call.

14 Maximum string device identifier supported. A returned value of –1 indicates that no string device is available. To query the availability of a specific string device, use the GSQLID call.

15 Maximum stroke device identifier supported. A returned value of –1 indicates that no stroke device is available. To query the availability of a specific stroke device, use the GSQLID call.

16 Not used.

17 The number of mouse or tablet buttons that are available to the application program. A returned value of 0 indicates that there are no mouse or tablet buttons available.

18 Graphics cursor support. Possible values are:

–**1** No graphics cursor supported.
0 Graphics cursor is emulated using the alphanumerics cursor.
1 The device provides a graphics cursor.

19 Foreground mix information.

1 OR
2 Overpaint (opaque)
4 Underpaint
8 Exclusive-OR
16 Leave alone (transparent)

A value is returned that indicates the modes of the foreground mix supported on the device. Each mode is represented by a number, and the value returned is the sum of these numbers. For example, if the device supported “OR” (1), “overpaint” (2), and “underpaint” (4), the value returned by the FSQURY call is 7.

Note that these numbers correspond to the decimal representation of a bit string that is five bits long, with each bit set to 1 if the appropriate mode is supported.

20 Background mix information

1 OR
2 Overpaint (opaque)
4 Underpaint
8 Exclusive-OR
16 Leave alone (transparent).

A value is returned that indicates the modes of background mix supported on the device. Each mode is represented by a number, and the value returned is the sum of these numbers. For example, if the device supported “overpaint” (2), and “leave-alone” (16), the value returned by the FSQURY call is 18.

Note that these numbers correspond to the decimal representation of a bit string that is five bits long, with each bit set to 1 if the appropriate mode is supported.

21 Foreground/Background Mix combination. A value is returned that indicates the modes of foreground mix with which background mix is supported on the device. The possible values are:

0 Background mix is supported with all modes of foreground mix available on this device
1 Background mix is supported only if the foreground mix mode is **overpaint**.

22 Alphanumerics and graphics interaction

1 When alphanumerics are removed from the screen, GDDM does not restore any underlying graphics. To restore the underlying graphics, the application may call FSREST.
2 When alphanumerics are removed from the screen, GDDM always restores any underlying graphics.

When **code=3**, the information on attached plotters is returned:

AE ***Device information***

1 Not used.

2 Pen velocity (see DSOPEN procopt group 11).

3 Pen pressure (see DSOPEN procopt group 13).

4 Paper-size value (1 through 5; see DSOPEN procopt group 15).

5 Paper-size code:

- **1** ISO
- **2** ANSI

6 Roll-feed capability

- **0** The plotter does not support roll feed media.
- **1** The plotter supports roll feed media.
- **2** GDDM cannot determine plotter roll feed capability.

  This value can be returned if no device token has been specified for the plotter, and the plotter is an attached IBM 6186 or IBM 6187 plotter, which does not have roll media loaded.

  This value can also be returned if no device token has been specified and the plotter is not directly attached.

7 Roll feed media

- **0** Roll feed media is not loaded.
- **1** Roll feed media is loaded.
- **2** The plotter is not directly attached and cannot be queried.

  This value is returned when output is to a family-2 file to be spooled to a plotter, or plot output is being redirected to an IBM-GL format file, or output to a dummy device.

8 Plotter page feed

- **1** No page feed.
- **2** Page feed.

9 Minimum **x** value as a percentage of the maximum paper width.

10 Maximum **x** value as a percentage of the maximum paper width.

11 Minimum **y** value as a percentage of the maximum paper height.

12 Maximum **y** value as a percentage of the maximum paper height.

If the array elements 9 through 12 are all 0, GDDM cannot determine the plotting area because the plotter is not directly attached.

13 Plotter picture orientation

- **0** GDDM cannot determine the orientation.

  This can occur when the plotter is not directly attached, or when the PLTAREA processing option is specified with all values given as zeros.
- **1** No rotation.
- **2** Picture rotated by 90 degrees.

14 Plot device

- **0** The device is not a plotter.
- **1** Plotter is a directly attached family-1 device, or a dummy device.
- **2** Plotter output is directed to a family-2 ADMPRINT file spooled to a plotter.
- **3** Plotter output is directed to an IBM-GL format file.

15 Plot delay. The delay, in seconds, between successive frames of a long roll medium plot.

16 Maximum page depth.

17 Maximum page width.

GDDM returns the maximum page size that should be specified in an FSPCRT call to the plotter. When GDDM uses the device characteristics to determine page depth and width, the following assumptions are made:

- The device does not support roll-feed media (if unable to determine roll-feed capability).
- The plot is not rotated (if unable to determine orientation).

If a plotter device token is specified for a family-2 device, the values returned are determined from the device token, and from the nickname matching the Stage 2 ID, provided that you have specified:

- A Stage 2 ID procopt, and
- A nickname matching that ID.

Otherwise, all values are returned as zeros.

If FSQURY is issued for a family-1 device with a plotter attached as an auxiliary device, the values returned are determined from the auxiliary device, provided that you have opened it. Otherwise, all values are returned as zeros.

When **code=4**, the image display characteristics information is returned:

AE ***Device information***

1 The width, in pixels, of the presentation area of the device that can be used for image.

2 The depth, in pixels, of the presentation area of the device that can be used for image.

3 The number of compression algorithms supported by the device. Transfer operations using an unsupported format invoke automatic conversion. For information on the compression algorithms, see ISQCOM.

4 The number of formats supported by the device. Transfer operations using an unsupported format invoke automatic conversion. For information on the formats, see ISQFOR.

5 The number of transforms supported by the device. Transfer operations using more than this number of transforms will require emulation of those in excess of the supported maximum or will be incomplete, depending on the ISCTL setting. 0 (zero) is returned if the device cannot perform transforms. –1 (minus 1) is returned if there is no limit to the number of transforms.

6 The maximum width, in pixels, of source image supported.

7 The maximum depth, in pixels, of source image supported.

When **code=5**, the image scanner characteristics information is returned:

AE ***Device information***

1 Scanner attached. Indicates whether or not a scanner is attached.

- **0** Scanner is not attached
- **1** Scanner is attached

2 The maximum width, in pixels, of the scanner area, at the maximum resolution indicated by the parameter returned in array element 5.

3 The maximum depth, in pixels, of the scanner area, at the maximum resolution indicated by the parameter returned in array element 6.

4 The units of resolution for the parameters below:

- **0** Inches
- **1** Meters

5 The maximum horizontal resolution, in the units given in array element 4.

6 The maximum vertical resolution, in the units given in array element 4.

7 The number of compression algorithms supported by the device. Transfer operations using an unsupported algorithm are emulated. For information on the compression algorithms, see ISQCOM.

8 The number of formats supported by the device. Transfer operations using an unsupported format are emulated. For information on the formats, see ISQFOR.

9 The number of transforms supported by the device. Transfer operations using more than this number of transforms are either emulated or are incomplete; see ISCTL. 0 (zero) is returned if the device cannot perform transforms. –1 (minus 1) is returned if there is no limit to the number of transforms.

10 Type of scanner

- **0** 3118 (feed-through scanner)
- **1** 3117 (flat-bed scanner)

11 Type of input available

- **0** Bi-level only. Possibly processed by in-scanner gray-scale transformations and conversions to bi-level, the following query responses indicate which of these are possible.

12 Automatic Document Feeder (ADF) feature.
Indicates whether the Automatic Document Feeder is installed.

- **0** ADF is not installed
- **1** ADF is installed

13 Brightness. Indicates whether or not the IMRBRI call is supported by the scanner; see IMRBRI.

- **0** Variable brightness not supported
- **1** Variable brightness supported

14 Contrast. Indicates whether or not the IMRCON call is supported on this device.

- **0** Variable contrast not supported
- **1** Variable contrast supported

15 Threshold. Indicates whether or not the IMRCVB call is supported on this device.

- **0** Variable threshold not supported
- **1** Variable threshold supported

16 Halftone. Indicates whether or not the halftone option of the IMRCVB call is supported on this device.

- **0** Halftone not supported
- **1** Halftone supported

When **code=6**, the image cursor characteristics information is returned:

AE ***Device information***

1 Image locator cursor. Indicates what type of image locator cursor is available. Image locator attributes are queried using the ISQLOC call.

- **0** Image locator cursor is not available
- **1** Image locator cursor is available in an emulated form
- **2** Image locator is available

2 Image box cursor. Indicates whether or not an image box cursor is available. Image box attributes are queried using the ISQBOX call.

- **0** Image box cursor is not available
- **2** Image box cursor is available

Description

The FSQURY call returns information on the characteristics of the primary device and of plotters that are attached as auxiliary devices.

When a ***5080 or 6090 Graphics System*** is queried, the graphics characteristics **(code=2)** refers to the 5080 or 6090

display, and the other information refers to the associated 3270 device.

The DSQDEV call can be used to obtain information about the DSOPEN parameters for the primary device.

Detailed descriptions of all the DSOPEN processing options, are provided in Chapter 19, “Processing options” on page 395.

Principal errors

```
ADM0129 E  ARRAY COUNT n IS INVALID
ADM0136 E  INVALID ELEMENT NUMBER (n) FOR QUERY CLASS a
ADM0139 E  QUERY CODE n IS INVALID
```

FSQWIN

Function

To query page window.

FSQWIN	**(row, column, depth, width)**
APL code	310
GDDM RCP code	X'0C040C01' (201591809)

Parameters

row *(returned by GDDM) (fullword integer)*
The page window top left row number.
column *(returned by GDDM) (fullword integer)*
The page window top column number.
depth *(returned by GDDM) (fullword integer)*
The page window depth in rows.
width *(returned by GDDM) (fullword integer)*
The page window width in columns.

Note: On some devices, the operator may change the window origin without GDDM being aware of the fact. The values returned are the values last set by GDDM.

Description

Returns the origin and size of the current page window.

Principal errors

None.

FSREST

Function

To restore entire screen contents on next output.

FSREST	**(code)**
APL code	103
GDDM RCP code	X'0C080C00' (201853952)

Parameters

code *(specified by user) (fullword integer)*
Specifies whether the contents of PS stores (or GS stores on PC 3270 PC/G terminals) are to be restored, together with the rest of the screen. Possible values are:
0 Data only. (Note that GDDM may still determine that some programmed symbols need to be transmitted.)
1 Data and symbols.

Description

Causes a complete retransmission of all data to the device upon the next call to FSFRCE, ASREAD, GSREAD, or MSREAD. Optionally, any programmed symbols that should be held by the device may also be transmitted.

If GDDM display device I/O is being interleaved with non-GDDM I/O to the same device, FSREST should be called before the first call to ASREAD, FSFRCE, GSREAD, or MSREAD that follows any non-GDDM I/O. This ensures that GDDM restores the screen contents correctly.

Principal errors

```
ADM0218 E  CODE n IS INVALID
```

FSRNIT

Function

To reinitialize GDDM.

FSRNIT	
APL code	118
GDDM RCP code	X'0C000002' (201326594)

Parameters

None.

Description

Reinitializes GDDM. This function is equivalent to the FSTERM call followed by FSINIT. GDDM retains only information about the primary device.

Do not use in association with GDDM-PGF; refer to the CHRNIT call description in the *GDDM-PGF Programming Reference* **book.**

FSRNIT can be used as a more efficient alternative to FSTERM and FSINIT, where it is required to restart GDDM with no memory of any previously defined pages, symbol sets, and so on, but where the same primary device is to be retained.

FSRNIT reinitializes the primary device (whether explicitly opened, or opened by default), and closes all other currently-open devices.

After the FSRNIT call has been processed, the primary device will be available as the subject of a DSUSE. It is not actually in use (although, as always, an implicit DSUSE call is issued against the default primary device if necessary).

If operator windows are being used by this instance of GDDM, a call to FSRNIT may cause them to be deleted. If an operator window that is associated with a virtual device in another instance of GDDM is deleted, the results of subsequent operations on that virtual device are undefined (see WSDEL).

Note: FSRNIT does not reinitialize the presentation graphics routines of GDDM-PGF. If GDDM-PGF is being used, issuing a call to FSRNIT produces unpredictable results.

Principal errors

None.

FSSAVE

Function

To save current device contents.

FSSAVE	**(picture-name)**
APL code	104
GDDM RCP code	X'0C100004' (202375172)

Parameters

picture-name *(specified by user) (8-byte character string)*
The name (left-justified) to be assigned to the picture when written to auxiliary storage.

Description

Saves a **device-dependent** form of the primary device contents on auxiliary storage for subsequent retrieval and display in output-only mode by FSSHOW.

Alphanumeric field attributes are preserved by the FSSAVE process. It is possible to enter data into input fields when the picture is subsequently displayed using FSSHOR or FSSHOW, but the data entered into such fields is not retrievable. This is a change from Version 1 Release 2 of GDDM where all alphanumeric fields were changed to protected when the picture was saved. Files saved before GDDM Version 1 Release 3 continue to have all alphanumeric fields protected when displayed using FSSHOR or FSSHOW.

The only limit on the complexity of the picture that can be saved is the limit that would be imposed were an attempt made to display the picture.

An FSSAVE call in an application running in an operator window saves the contents of the virtual screen (without borders), subject to the outer limits of the real screen. For example:

- If the virtual screen is smaller than the real screen, the virtual screen is saved.
- If the virtual screen is larger than the real screen, a real screen-sized virtual screen is saved.

For the FSSAVE-related GDDM external defaults, those effective for the real device are used.

For information about which devices support this call, see “Device specific saved pictures” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0233 W  SYMBOL SET IS NOT LOADED
ADM0273 W  PS OVERFLOW
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
ADM3004 E  FIELD LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3005 E  DATA BUFFER n1, ERROR n2 AT INDEX n3
ADM3010 E  BUNDLE LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3282 W  AMOUNT OF DATA EXCEEDS THE STORAGE CAPACITY
           OF THE DEVICE
```

FSSHOR

Function

To display a saved picture (extended FSSHOW).

FSSHOR (picture-name, key, qualifier)	
APL code	119
GDDM RCP code	X'0C100007' (202375175)

Parameters

picture-name *(specified by user) (8-byte character string)*
The name (left-justified) of the picture to be read from auxiliary storage.

key *(returned by GDDM) (fullword integer)*
The type of attention interrupt received. Possible values are:

0 ENTER key

1 PF key

2 Light pen

3 Badge reader.

The badge record field (the one containing the cursor) has special characters from the badge reader replaced by blanks when the operation succeeds. This operation causes a refresh of the display at the next call to FSFRCE or ASREAD.

4 PA key

5 CLEAR key

6 Other.

The interrupt received from the device does not belong to any of the defined categories; the **qualifier** value is undefined.

7 Output-only device.

The primary device only accepts output, and does not return input; the **qualifier** value is undefined. A warning message is returned to emphasize this situation.

qualifier *(returned by GDDM) (fullword integer)*
The value, if any, associated with **key**. Possible values are:

(for key=1)	PF key number
(for key=3)	**0** success
	1 failure
(for key=4)	PA key number.

Description

Obtains the specified picture from auxiliary storage, and displays it in the same way as the FSSHOW call. Also, it returns the identity of the key used to terminate the display.

Restrictions exist on the use of some keys, depending on the operating environment. For further information, refer to the *GDDM Base Application Programming Guide*.

Data can be entered into unprotected fields, but the data entered cannot be retrieved by the application.

For information about which devices support this call, see “Device specific saved pictures” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0272 E  SAVED DATA CANNOT BE SHOWN. REASON CODE  n
ADM0276 W  DEVICE IS OUTPUT ONLY
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
ADM0498 E  PRINT TERMINATED. RETURN CODE X'xxxxxx' FROM
           DEVICE
```

FSSHOW

Function

To display a saved picture.

FSSHOW (picture-name)	
APL code	105
GDDM RCP code	X'0C100005' (202375173)

Parameters

picture-name *(specified by user) (8-byte character string)*
The name (left-justified) of the picture to be read from auxiliary storage.

Description

Obtains the specified picture from auxiliary storage, and displays it.

This causes immediate output of the stored picture. **The picture retrieved must have been created on a device compatible with the one in use when FSSHOW is processed**.

On receipt of any keyboard interrupt, the screen is erased, the previous screen contents are restored, and processing may continue.

Even if the windowing processing option is specified, the saved picture is displayed using the whole real screen without the window borders, and without the other windows. The GDDM external defaults that are effective for the real device are used.

For information about which devices support this call, see “Device-specific saved pictures” on page 242.

Principal errors

```
ADM0272 E  SAVED DATA CANNOT BE SHOWN. REASON CODE  n
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
ADM0498 E  PRINT TERMINATED. RETURN CODE X'xxxxxx' FROM
           DEVICE
```

FSTERM

Function

To terminate GDDM processing.

FSTERM	
APL code	116
GDDM RCP code	X'0C000000' (201326592)

Parameters

None.

Description

Terminates GDDM processing for the application program, closes all the devices, and releases all the resources (such as storage) acquired.

In GDDM-REXX programs, this call is required only if the program is reentrant. It is used to terminate an instance of GDDM within the instance of GDDM-REXX.

The FSTERM call must be used as the last GDDM function executed in a program, to perform essential housekeeping.

Note to CICS/VS users: Under CICS/VS, it is the GDDM application programmer's responsibility to unlock the keyboard and leave the device in a state ready to process the next transaction.

This can be done by using the relevant option on the DSCLS call for the screen device.

If operator windows are being used by this instance of GDDM, a call to FSTERM may cause them to be deleted. If an operator window that is associated with a virtual device in another instance of GDDM is deleted, the results of subsequent operations on that virtual device are undefined (see WSDEL).

Principal errors

None.

FSTRAN

Function

To perform code conversion on a character string.

FSTRAN	**(type, from-page, to-page, length, in-string, out-string)**
APL code	132
GDDM RCP code	X'000F0000' (983040)

Parameters

type *(specified by user) (fullword integer)*
The type of code conversion. Possible values are:

- **0** EBCDIC code conversion. All characters are subject to code conversion.
- **1** Mixed code conversion. Characters following Shift-out (X'0E') up to the next Shift-in (X'0F') are copied without code conversion. The sequence of SO and SI characters is validated. There must be an even number of bytes between the SO and SI characters and they must appear as matched pairs with no nesting.
- **2** EBCDIC code conversion with folding. Same as 0 with folding of lower-case characters to uppercase.
- **3** Mixed code conversion with folding. Same as 1 with folding of lower-case single-byte characters to upper case.

from-page *(specified by user) (fullword integer)*
The code-page identifier for **in-string**. A value of 0 (the default) identifies the application code page. Other possible values are defined in the current ADMDATRN tables.

to-page *(specified by user) (fullword integer)*
The code-page identifier for **out-string**. A value of 0 (the default) identifies the application code page. Other possible values are defined in the current ADMDATRN tables.

length *(specified by user) (fullword integer)*
The length of **in-string** and **out-string**. The range of both is 0 through 16384.

in-string *(specified by user) (character)*
The character string to be converted.

out-string *(returned by GDDM) (character)*
The converted string.

Description

Copies a character string in user storage, converting it into a different code page.

Principal errors

```
ADM0230 E  TRANSLATION TYPE n IS INVALID
ADM0231 E  INVALID SO/SI SEQUENCE IN MIXED FIELD
```

```
ADM0232 E  CODE PAGE n IS NOT SUPPORTED
ADM0235 W  STRING CONTAINS UNTRANSLATABLE CHARACTERS
ADM0236 E  INVALID LENGTH n
```

Product-sensitive programming information

FSTRCE

Function

To control internal trace.

Note: This call is intended for diagnostic purposes only, and is not meant for use in production programs.

FSTRCE	**(control)**
APL code	108
GDDM RCP code	X'00020000' (131072)

Parameters

control *(specified by user) (fullword integer)*
Determines the level of the trace output to be generated.

Description

Controls the internal trace functions. FSTRCE is intended for internal error diagnosis. The control details and the output format are defined in the *GDDM Diagnosis* book.

By default, trace is deactivated.

Principal errors

None.

End of Product-sensitive programming information

FSUPDM

Function

To set update mode.

FSUPDM	**(control)**
APL code	662
GDDM RCP code	X'0C0C1A00' (202119680)

Parameters

control *(specified by user) (fullword integer)*
The update mode to be used. Possible values are:
0 No optimization (the default).
1 Optimization by a GDDM-chosen mode.

Description

Allows control over the way updates to graphics data take place on a particular device.

When data is updated (for example, when a segment is deleted), the graphics is usually redrawn to produce the correct representation on the screen. The FSUPDM call can be used to indicate that updates should be optimized so that only *changed* segments are updated.

The effect of selecting update mode is that the picture is put into a representation that allows optimized updating. A faithful representation is not ensured, but GDDM produces a “working model” that allows the operator to identify parts of the picture and to request the updates from the application.

The application can request that GDDM choose a particular update mode appropriate for the device or the application can request that no optimization is performed.

The only devices that support the FSUPDM call are the 3270-PC/G, the 3270-PC/GX, the 3179-G, 3192-G, and 3472-G color display stations, the 5550-family Multistation, and devices using GDDM-PCLK, which optimize picture updates by drawing the whole screen in exclusive-OR mix mode. This mix mode only reflects changes correctly if data does not overlap. If it does overlap, the final color of a particular pixel is obtained by exclusive-ORing the colors of the different data involved.

Also, when overlapping partitions are deleted or modified, there may in some cases be inaccuracies that reappear in the data.

The update mode can also be set by a DSOPEN processing option either by a DSOPEN call, or by a nickname override.

Also, User Control can be used to select a particular update mode in devices that support it.

The initial value of the update mode is that set by the processing option.

Note: Fast update mode is not supported for ASCII graphics devices.

Principal errors

```
ADM3271 E  UPDATE MODE n IS INVALID
```

GSAM

Function

To set attribute mode.

GSAM (n)	
APL code	647
GDDM RCP code	X'0C0C1311' (202117905)

Parameters

n *(specified by user) (fullword integer)*
The attribute mode. Possible values are:
0 Preserve attributes (the default)
1 Do not preserve attributes.

Description

Specifies the current attribute mode.

The attribute mode is used to specify whether primitive attributes should be preserved when set to a new value, so that they may be restored as required; see GSPOP. Any attributes that have been preserved in a called segment are automatically restored on return to the caller; see GSCALL.

The following primitive attributes are affected:

- Background Mix Mode (GSBMIX)
- Character Angle (GSCA)
- Character Box Size (GSCB)
- Character Direction (GSCD)
- Character Mode (GSCM)
- Character Set (GSCS)
- Character Shear (GSCH)
- Color (GSCOL)
- Current Position (GSCP)
- Current Transform (GSSCT)
- Line Width (GSFLW and GSLW)
- Line Type (GSLT)
- Marker Size (GSMB and GSMSC)
- Marker Symbol (GSMS)
- Mixing Mode (GSMIX)
- Pattern (GSPAT)
- Primitive Tag (GSTAG)
- Segment Viewing Limits (GSSVL)
- Text Alignment (GSTA)

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSARC

Function

To draw a circular arc.

GSARC (xc, yc, angle)	
APL code	521
GDDM RCP code	X'0C0C0600' (202114560)

Parameters

xc *(specified by user) (short floating point)*
The x coordinate of the center as an absolute point in world coordinates.
yc *(specified by user) (short floating point)*
The y coordinate of the center as an absolute point in world coordinates.
angle *(specified by user) (short floating point)*
The angle subtended by the arc in degrees.

If the angle is positive (nonzero), the arc is drawn counterclockwise; if the angle is negative, the arc is drawn clockwise.

There are no limits on the value of the angle specified; for example, if a value of 400 degrees is specified, a full circle is drawn, and then an arc of 40 degrees (both performed in a counterclockwise direction).

Description

Draws a circular arc about a specified point, starting at the current position. The arc subtends the specified angle at the center. If the angle is positive, the arc is drawn counterclockwise from the starting point; if negative, the arc is drawn clockwise.

Note that the direction of the arc is determined in world-coordinate space, assuming the x axis runs from left to right and the y axis runs from bottom to top. The direction may appear reversed on the display surface if the lower window limit is larger than the upper limit on either axis.

The arc has the color, line width, and line type given by the current values of these attributes.

The current position is set to the end of the arc.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
```

GSARCC

Function

To specify aspect-ratio control (for copy).

GSARCC (control)	
APL code	598
GDDM RCP code	X'0C0C000B' (202113035)

Parameters

control *(specified by user) (fullword integer)*
Specifies how the aspect ratio is to be determined when a copied picture is processed for the target device of a copy function. Possible values are:

0 Preserve the picture space aspect ratio (the default).

When the picture is copied to another device, the picture space is mapped onto a picture space with the same aspect ratio so that, for example, a displayed circle is presented as a circle. This happens even if the original picture space is determined by default.

1 Allow the picture space to be defaulted when copied.

This option maps the graphics field onto the corresponding graphics field of the target device (which occupies the same alphanumeric rows and columns), so that the graphics aspect ratio changes in the same way as that of the alphanumeric fields – as determined by hardware cell sizes. This happens even if the original picture space is set explicitly.

This control allows an area within an alphanumeric field layout that is assigned to graphics to be preserved when the page is copied. For example, if a GDDM page is the same size as a display screen (as happens with the default page), this control makes the FSCOPY call produce a true copy of the screen, but with a different aspect ratio.

Description

Specifies whether the aspect ratio is to be preserved when a picture is copied to another device (for example, by FSCOPY or GSCOPY).

The GSARCC call specifies whether the graphics aspect ratio itself, or the relationship between the graphics and alphanumeric fields, is to be preserved.

The control applies to the current page. It relates to any current or future graphics field on that page, but it does not cause a graphics field to be defined if none exists.

Principal errors

```
ADM0153 E  CONTROL VALUE n IS INVALID
```

GSAREA

Function

To start a shaded area.

GSAREA (control)	
APL code	522
GDDM RCP code	X'0C0C0408' (202114056)

Parameters

control *(specified by user) (fullword integer)*
Specifies whether boundary lines are to be drawn. Possible values are:
0 Do not draw boundary lines
1 Draw boundary lines.

Description

Begins the construction of a shaded area. The construction is terminated by the GSENDA call.

The following calls are the only ones that can be used between a GSAREA and a GSENDA call:

- GSARC GSCOL GSELPS GSFLW GSLINE GSLT GSLW GSMIX GSMOVE GSPFLT GSPLNE GSVECM
- GSCP and GSPOP are sometimes allowed.
- GSCALL is allowed if the segment being called contains only functions that are valid inside an area.
- All ASaaaa and FSaaaa calls.

The area shading is performed using the current pattern, as set by the GSPAT call. The color and color-mixing modes that are current at the time the GSAREA call is issued define the attributes to be applied to the pattern.

Note: If a multicolored pattern is required, the color must be set to neutral (by a GSCOL(7) call) before the GSAREA call is issued.

The area boundary consists of one or more **closed** figures, each constructed by calls to GSARC, GSLINE, GSPLNE, GSELPS, GSPFLT, or an element of GSVECM having a control value of 1. Calls to GSCOL and GSMIX can be used to control how the area boundary is to be colored. Calls to GSLT, GSLW, and GSFLW may be interspersed to control line attributes as desired. The starting point of each closed figure is the current position when GSAREA is called, or as specified by a subsequent call to GSMOVE. Figure con-

struction continues until a call to GSMOVE (or an element of GSVECM with a control value of zero), or GSENDA is met. The end point of the figure is the current position resulting from the last line or arc drawn.

Each figure should be closed; that is, the start and end points should be identical. If this is not so, the figure is arbitrarily closed by a straight line connecting the start and end points.

The figures formed in this way jointly define the area boundary. Any connected region with an **odd** number of line crossings from infinity is shaded with the current shading pattern and color. Regions with an **even** number of line crossings from infinity are **not** shaded.

If the control parameter of the GSAREA call is zero, the actual boundary lines are **not** drawn, but the shading ends at the boundaries. If the control parameter is 1, the boundary lines and any lines added to close figures are drawn, using the current line attributes, which can be changed the boundaries, as noted above.

The way that areas are shaded is device dependent, and this may lead to unexpected results. For example, with GDDM-OS/2 Link, areas are shaded to include the boundaries which may lead to hidden axes on a surface graph.

The current position is not changed by the GSAREA call, but can be changed by the moves and lines between GSAREA and GSENDA, including any used to close figures.

Area definitions may not be nested.

For information about restrictions on various devices, see “Graphics area shading” on page 247.

Principal errors

```
ADM0153 E  CONTROL VALUE n IS INVALID
ADM0159 E  ATTEMPT TO START SECOND AREA
```

GSBMIX

Function

To set current background color-mixing mode.

GSBMIX (n)	
APL code	664
GDDM RCP code	X'0C0C1317' (202117911)

Parameters

n *(specified by user) (fullword integer)*
Defines the color-mixing mode. Possible values are:

0 The drawing default; initially the same as 5 (or 2 if transparent mode is not supported on the device).
2 Opaque mode; the background of the primitive takes precedence over whatever is underneath.
5 Transparent mode; the background of the primitive has no effect on what is underneath.

Description

Controls the way that the background color of a primitive is combined with the color of any primitive which it overlaps.

The degree of support for various combinations of foreground and background mix modes is device-dependent. Variations are listed in the tables below. On IBM devices that support background mix orders any combination of foreground and background mix modes is allowed.

The background mix attribute remains in effect until it is changed by another GSBMIX call.

When a segment is created by the GSSEG call, the background mix attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the background mix attribute is reset to the value that was in effect when the segment was created.

The following primitives are affected by the background mix attribute:

Areas
The background of an area is defined to be every pixel within the area that is not set by the shading pattern.

Mode-1 and Mode-2 text
The background of a mode-1 or mode-2 character is every pixel within the character definition that is not set. However, the effect of background mix on mode-1 characters is device-dependent, and in some cases only one of the background mix modes is supported.

Mode-3 text
The background of a mode-3 character is the complete character box.

Note: For mixed-character strings, if the shift-out and shift-in control codes take a position in the displayed string (see GSSEN), the shift-out and shift-in characters do not have a background.

Images
For an image, the background is every pixel within the image that is not set.

Markers
For an image marker, the background is every pixel within the marker definition that is not set. The background of a vector marker is the complete marker box.

If a combination of foreground and background mix modes is requested that is not allowed, the call specifying this combination (GSMIX or GSBMIX) issues a warning message. In this case the requested mode is recorded, but while this

combination exists results are device dependent; normally all primitives are drawn with a background mix mode of transparent and with the requested foreground mix mode. If a subsequent change to the foreground mix mode results in the requested combination of modes becoming valid, the requested background mix mode is used.

Note: The fixed-point GDF that is generated for devices not supporting the background mix order contains backgrounds in the form of areas. If this GDF is stored by means of the GSGET or GSSAVE calls, and then redrawn at a different scale, the background of images and mode-2 text is not scaled by the same amount as the foreground. This does not occur with floating-point GDF.

For information about restrictions on various devices, see "Combinations of foreground and background mix modes" on page 246.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM3266 W  FOREGROUND MIX n1 BACKGROUND MIX n2
           COMBINATION INVALID
```

GSBND

Function

To define a data boundary.

GSBND	**(u1, u2, v1, v2)**
APL code	657
GDDM RCP code	X'0C0C000D' (202113037)

Parameters

u1 *(specified by user) (short floating point)*
u2 *(specified by user) (short floating point)*
The left- and right-hand extents of the data boundary in world coordinates.
v1 *(specified by user) (short floating point)*
v2 *(specified by user) (short floating point)*
The lower and upper extents of the data boundary in world coordinates.

Description

Explicitly defines the outer limits of the primitive data to be retained by GDDM when clipping is enabled. It may be set whenever there are no open graphics segments.

If the clipping mode is 0, the data boundary has no effect.

If the clipping mode is 1, parts of the picture that fall completely outside the boundary are removed and parts of the picture that lie across the boundary are clipped to the boundary, according to the normal clipping rules for the primitive.

If the clipping mode is 2, parts of the picture that fall completely outside the boundary are, in general, removed and parts of the picture that lie across the boundary are retained without being clipped.

For more information about the clipping mode, see GSCLP.

Setting the data boundary causes the establishment of a default graphics field, if the graphics field was not already set. If the boundary was not set when the graphics field is established, it defaults to be the same as the graphics window. The boundary can be reset to the default either by setting it to the same values as the current graphics window or by using zero for each parameter.

The parameters are checked to ensure that, with respect to the direction of the coordinate system of the graphics window, the right extent is greater than the left and that the upper extent is greater than the lower.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM3255 E  DATA BOUNDARY SPECIFICATION f IS INVALID
ADM3256 E  RIGHT DATA BOUNDARY f1 ¬> LEFT f2
ADM3257 E  UPPER DATA BOUNDARY f1 ¬> LOWER f2
```

GSCA

Function

To set current character angle.

GSCA	**(dx, dy)**
APL code	510
GDDM RCP code	X'0C0C0708' (202114824)

Parameters

dx *(specified by user) (short floating point)*
dy *(specified by user) (short floating point)*
Two short floating-point numbers giving the x and y separation of two points along the baseline.

If a baseline at an angle A is required, this angle can be obtained by setting dx=cos(A) and dy=sin(A) if one x-axis unit is physically equal to one y-axis unit.

If both dx and dy are zero, the attribute is set to the drawing default value.

Description

Specifies the angle of the baseline for the characters in a string, as a relative vector. The angle attribute remains in effect until it is changed by another GSCA call. When a segment is created by the GSSEG call, the angle attribute is set to the drawing default value. When a segment is closed by the GSSCLS call, the angle attribute is reset to the value that was in effect when the segment was created; see Figure 4 on page 100.

In character-mode 1, the call has no effect when characters are drawn.

In character-mode 2, the angle is used to determine the position of each character, but the orientation of characters within the character box is inherent in their definitions. The characters are positioned so that the lower left-hand corners of the character definitions are placed at the lower left-hand corners of the character boxes. This is illustrated in Figure 4.

In character-mode 3, the angle is observed accurately, and the character boxes are rotated to be normal to the character baseline. If the window coordinate system is such that one x-axis unit is not physically equal to one y-axis unit, a rotated character string appears to be sheared.

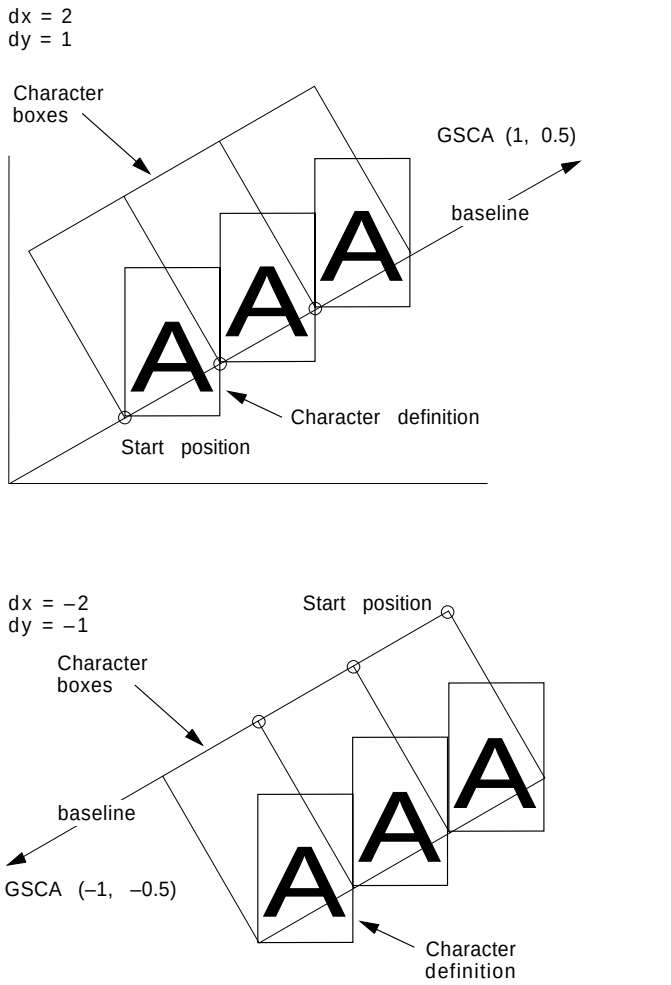


Figure 4. Character angle and mode-2 text positioning (GSCA)

Principal errors

```
ADM0157 E  ANGLE f1, f2 NOT DEFINED
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSCALL

Function

To call a segment.

GSCALL	**(segid, flag, sx, sy, hx, hy, rx, ry, dx, dy, type)**
APL code	653
GDDM RCP code	X'0C0C1402' (202118146)

Parameters

segid *(specified by user) (fullword integer)*
The identifier of the segment to be called.

flag *(specified by user) (fullword integer)*
The call flag. This must be cleared to 0.

sx *(specified by user) (short floating point)*
sy *(specified by user) (short floating point)*
A scale transformation in terms of an x-axis scaling (**sx**) and a y-axis scaling (**sy**). The segment origin is used as a reference point; the axes that are used to scale are parallel to the x and y axes but pass through the segment origin. A scale factor in the range 0 through 1 shrinks primitives; a scale factor of greater than 1 stretches primitives. A negative scale factor reflects primitives about the other axis.

Specifying scale factors **sx=1** and **sy=1** does not perform any scaling. This setting can be used to suppress scaling (to allow a simple rotation, for example).

hx *(specified by user) (short floating point)*
hy *(specified by user) (short floating point)*
A shear transformation in terms of the displacements that a point on the y axis makes after shearing. The axes used for shearing are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used in the GSCH call for character shear. Note that primitives below the x axis are sheared in the opposite direction to those above the x axis. The points on the x axis itself are not moved. If **hx=a** and **hy=b** are used, an identical effect is achieved with **hx=–a** and **hy=–b**.

Specifying shear factors **hx=0** and **hy=1** does not perform any shearing. This setting can be used to suppress shearing (to allow a simple rotation, for example).

Specifying **hy=0** is not valid (because it would produce an infinite shear).

rx *(specified by user) (short floating point)*
ry *(specified by user) (short floating point)*
A rotation transformation in terms of the displacements that a point on the x axis makes after rotating. The axes used for rotating are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used for the character angle in the GSCA call.

Specifying rotation components **rx=1** and **ry=0** does not perform any rotation. This setting can be used to suppress rotation (to allow a simple scaling, for example).

Because two zero values would be ambiguous, specifying **rx=0** and **ry=0** is taken as equivalent to **rx=1** and **ry=0** (no rotation).

The (rx,ry) values below produce these special cases:

(0, 0) No rotation; equivalent to (1,0)
(1, 0) No rotation
(0, 1) Rotation by 90 degrees counterclockwise
(1, 1) Rotation by 45 degrees counterclockwise
(0,–1) Rotation by 90 degrees clockwise
(–1,0) Rotation by 180 degrees clockwise (or counterclockwise).

and, in general:

(rx,ry) Rotation by theta degrees counterclockwise, where tan(theta)=ry/rx (assuming a uniform world-coordinate system).

Note: A rotation of (–1,0) is equivalent to a scale factor of **sx=–1** and **sy=–1**.

dx *(specified by user) (short floating point)*
dy *(specified by user) (short floating point)*
A displacement of **dx** parallel to the x axis and **dy** parallel to the y axis. This transformation does not use the segment origin.

Specifying displacements components of **dx=0** and **dy=0** does not perform any displacement. This setting can be used to suppress displacements (to allow a simple rotation, for example).

type *(specified by user) (fullword integer)*
How the existing current transformation is to be modified by the scaling, shear, rotation, and displacement components specified. Possible values are:

0 New/replace
Any current transform previously defined is discarded and is replaced by the combined effect of the specified components.

1 Additive
The combined effect of the specified components is added to the effect already present in the existing current transform. The new transform combines both effects in the order (i) old transform, and (ii) GSCALL parameter values. This option is the most useful for incremental updates to transforms.

2 Preemptive
The combined effect of the specified components is added to the effect of the existing current transform. The new transform combines both effects in the order (i) GSCALL parameter values, and (ii) the old transform. The effect is as if the GSCALL parameters modify the primitives of the segment (without transformation) and the existing transformation is applied again.

Description

Calls the segment with the specified identification number from the open segment.

This function is only valid from within a segment.

The transformation specified by the geometric attributes is set before calling the segment to allow its scale, shear, rotation, and position to be specified. This transform only applies to the called segment and is reset on return to the transform in operation before the call was made. If the **type** parameter specifies the additive or preemptive option, the transform is combined with the previous current transform (if any) specified on a GSSCT call.

If a GSCALL is issued while inside an area definition, the segment being called should only contain functions that are valid inside an area; see GSAREA. If an invalid function is used inside the called segment, that call is not processed.

If there is a segment transform on the called segment, it is always applied to its primitives before the current transform.

Consider the following call structure:

```
   Segment 1
   (Transform)   Always applied

   GSSCT(...)
   ....

GSCALL(2,.....type)
   ....

   GSSCLS

   Segment 2
   (Transform)   Always applied

   GSSCLS
```

Note: After a GSCALL, primitive attributes may not always match the values returned on the query attribute calls. If a called segment changes any attributes while the attribute mode (see GSAM) is set to 1 (do not preserve), these values will still be current on return from the called segment. However, as the called segment may not exist when the GSCALL order is given, or the called segment may be deleted and recreated before the picture is produced, any attribute query calls will only return the attribute values expected for the current segment.

If the called segment does not contain explicit set attribute calls, then the effect of set attribute calls in the calling segment is similar to the effects of changing the drawing default values by use of the GSDEFS call.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0149 E  NO CURRENT GRAPHICS SEGMENT
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM3262 W  CALL TO SEGMENT n PRODUCES RECURSIVE LOOP
ADM3265 W  CALLED SEGMENT n NOT FOUND
ADM3268 W  CALLED SEGMENT IS CURRENT
ADM3273 W  CALLED SEGMENT n CONTAINS FUNCTIONS INVALID
           INSIDE AN AREA
```

Note: Messages ***ADM3262*** and ***ADM3265*** are a result of GSCALL, but are not issued until the picture is displayed by means of a call to ASREAD, or FSREAD, for example.

GSCB

Function

To set character-box size.

GSCB	**(width, height)**
APL code	511
GDDM RCP code	X'0C0C0707' (202114823)

Parameters

width *(specified by user) (short floating point)*
The width of the character box in world-coordinate units.
height *(specified by user) (short floating point)*
The height of the character box in world-coordinate units.

The width determines the spacing of consecutive characters along the baseline. The height determines the spacing of consecutive lines.

Both the width and height can be positive, negative, or zero.

When either parameter is negative, the spacing occurs in the opposite direction to normal **and each character is drawn reflected** in character-mode 3. Thus, for example, a negative height in the standard direction in mode 3 means that the characters are drawn upside down and the string drawn below the baseline (assuming a standard window definition, with x increasing from left to right and y increasing from the bottom to the top).

A zero character width or height is also valid; here, the string of characters collapses into a line (or several lines). If both are zero, the string is drawn as a single point.

Description

Sets the character spacing, or size, or both for subsequent characters. The character-box size attribute remains in effect until it is changed by another GSCB call.

When a segment is created by the GSSEG call, the character-box size attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the character-box size attribute is reset to the value that was in effect when the segment was created.

In character-mode 1, the box size specified is ignored. Characters use the standard hardware size and spacing.

In character-mode 2, the spacing of consecutive characters is determined by the parameters given, but the size of characters is not affected, because this is inherent in the character definitions. Characters are placed as described under the GSCA call. The separation of characters along the

baseline is given by the **width** parameter, and the separation of consecutive lines (perpendicular to the baseline) is given by the **height** parameter. Note that the character-box width measures the separation along the baseline. Because in character-mode 2 the characters themselves cannot be rotated when the angle of the baseline is changed, it is usually also necessary to change the character-box size to achieve satisfactory spacing.

In character-mode 3, the parameters determine the size and aspect ratio of the characters. The size of each character is such as to fill the character box. Consecutive lines of characters are separated by the value specified in the **height** parameter. Characters are separated along the baseline (defined by GSCA) by the value in the **width** parameter.

Spacing works as follows. After GSCHAR or GSCHAP has drawn a nonproportionally spaced character, the current position is moved along by an amount equal to the width of the character box. After drawing a proportionally spaced character, the movement is a fraction of the character box width. This fraction is the ratio between the character's assigned width and the maximum, as recorded in the definition of the character.

The amount of space occupied by a proportionally spaced character string can be determined by the GSQTB call.

The default character-box size, where this call is not used, is the cell size of the current device. This means that the same text output on different devices may show differences in the spacing of character-mode 2 text.

Principal errors

```
ADM0155 E  CHARACTER SIZE f IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSCBS

Function

To set character-box spacing.

GSCBS	**(width-multiplier, height-multiplier)**
APL code	646
GDDM RCP code	X'0C0C130F' (202117903)

Parameters

width-multiplier *(specified by user) (short floating point)*
The amount by which the current character-box width is to be multiplied.

height-multiplier *(specified by user) (short floating point)*
The amount by which the current character-box height is to be multiplied.

Description

Sets the amount of space or overlap to be provided between successive character boxes in a character string.

When a segment is created by the GSSEG call, the character-box spacing attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the character-box spacing attribute is reset to the value that was in effect when the segment was created.

The character-box spacing text attribute has two components to specify the spacing in the horizontal (**width-multiplier**) and vertical (**height-multiplier**) directions. The way they are used depends on the current character direction (defined by the GSCD call.

When the character direction is horizontal, the width-multiplier controls the amount of space between successive character boxes, horizontally, and the height-multiplier controls the spacing allowed between rows of characters, vertically.

When the character direction is vertical, the height-multiplier controls the spacing between successive character boxes, vertically, and the width-multiplier controls the spacing between columns of characters, horizontally.

Both parameters are expressed as multipliers that are applied to the current character-box width and height to derive the amount of overlap or extra space that is to be provided. The parameters are short floating-point numbers which can be negative, zero, or positive:

- A negative value gives overlapping character boxes.
- A value of zero (the initial default) results in standard spacing.
- A positive value allows extra space between character boxes.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSCD

Function

To set current character direction.

GSCD	**(control)**
APL code	512
GDDM RCP code	X'0C0C0709' (202114825)

Parameters

control *(specified by user) (fullword integer)*
The character direction. Possible values are:

0 The drawing default; initially, the same as 1.

1 Character boxes (defined by GSCB) are arranged parallel to, and directed along, the baseline. New-line direction is 90 degrees clockwise from the baseline. This is the usual convention for Roman text.

2 Character boxes are arranged in columns directed 90 degrees **clockwise** from the baseline. New-column direction is the reverse of the baseline direction. This is the usual convention for Chinese characters. This option can be used for drawing Roman text vertically (a y-axis title on a graph, for example).

3 Character boxes are arranged parallel to, but in the reverse of, the baseline direction. New-line direction is 90 degrees clockwise from the baseline. This is the usual convention for Arabic text.

4 Character boxes are arranged in columns directed 90 degrees **counterclockwise** from the baseline. New-column direction is the reverse of the baseline direction.

Figure 5 illustrates these options. In each case, the current position at the start of the string is marked by the letters "**sp**", and the current position at the end of the string is marked by "**ep**".

Description

Specifies the direction in which the characters in a string are to be drawn, relative to the baseline specified in a GSCA call.

The character direction attribute remains in effect until it is changed by another GSCD call.

When a segment is created by the GSSEG call, the character direction attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the character direction attribute is reset to the value that was in effect when the segment was created.

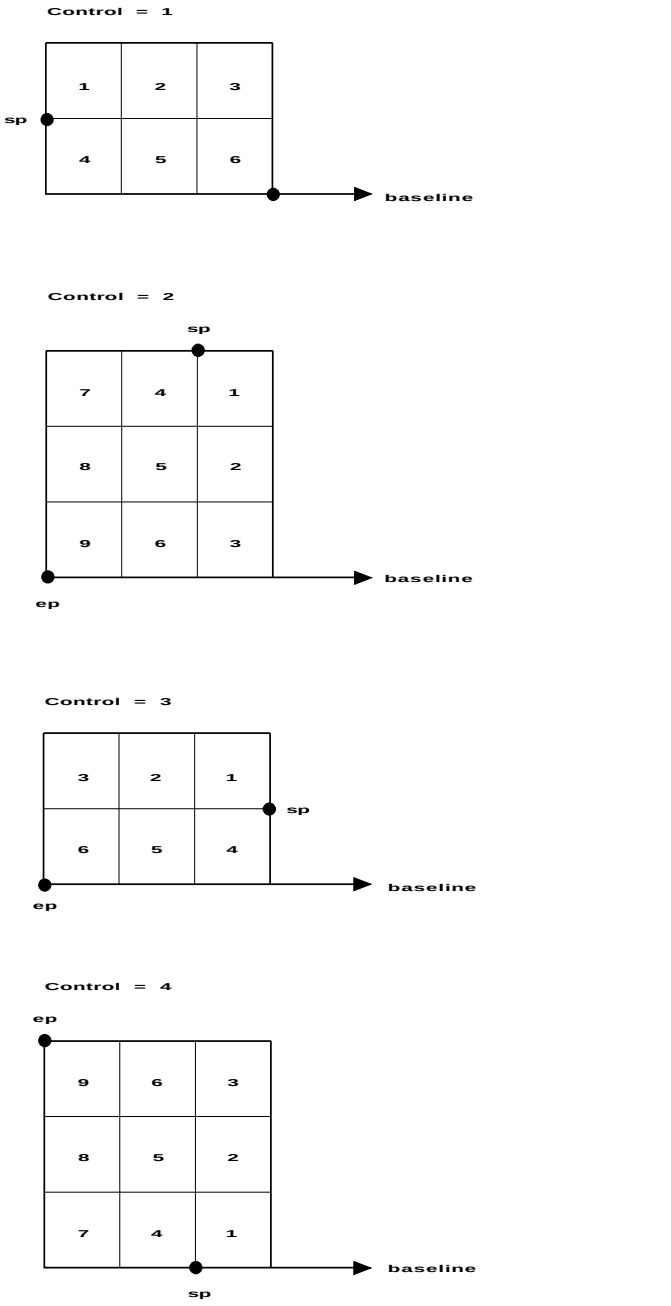


Figure 5. Character direction (GSCD)

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSCH

Function

To set current character shear.

GSCH (dx, dy)	
APL code	558
GDDM RCP code	X'0C0C070C' (202114828)

Parameters

dx *(specified by user) (short floating point)*
dy *(specified by user) (short floating point)*
The x and y separation of two points on an upright character stroke.

If **dx=0** and **dy=1** (initial default), "upright" characters result. If **dx** and **dy** are both positive or both negative, the characters slope from bottom left to top right. If **dx** and **dy** are of opposite signs, the characters slope from top left to bottom right. No character inversion ever takes place as a result of a shear alone. (Inversion can be performed with the GSCB call.)

Usually, it is an error to specify a zero value for **dy** because this would imply an "infinite" shear. However, if both `dx` and `dy` are zero, the attribute is set to the drawing default value.

Description

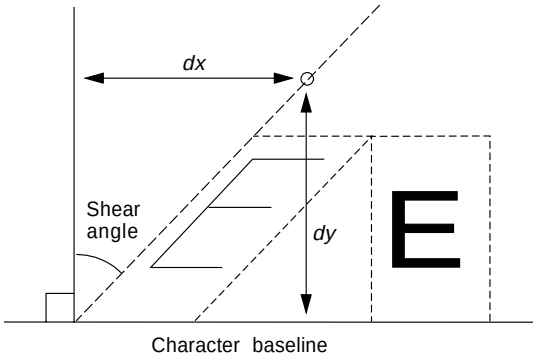


Figure 6. Character shear (GSCH)

Shears mode-3 characters to the angle defined by **dx,dy**. The character-shear attribute remains in effect until it is changed by another GSCH call.

When a segment is created by the GSSEG call, the character-shear attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the character-shear attribute is reset to the value that was in effect when the segment was created.

Characters in a character string may be subjected to a shear transformation. This causes upright lines to be inclined to their normal angle, rather as if the characters were *italic*. The top of the character box remains parallel to the character baseline.

The relative vector **dx,dy** defines the angle of the upright strokes relative to the baseline; see Figure 6.

The call has no effect on mode-1 (string precision) character strings.

For mode-2 (character precision) strings, the call may affect the position of individual characters but not their shape.

In character-mode 3, the transformation is observed accurately.

Principal errors

```
ADM0157 E  ANGLE f1, f2 NOT DEFINED
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSCHAP

Function

To draw a character string at current position.

GSCHAP (length, string)	
APL code	523
GDDM RCP code	X'0C0C0501' (202114305)

Parameters

length *(specified by user) (fullword integer)*
The number of characters in **string**.
string *(specified by user) (character)*
The string of characters to be drawn. See also **string** under GSCHAR.

Description

Draws a text string in the graphics field starting at the current position. Also allows a new text string to be appended to a previous one. For information on attribute settings, such as position, spacing, size, and clipping, that affect character drawing, see GSCHAR.

Principal errors

```
ADM0111 W  DBCS SYMBOL SET 'a' NOT AVAILABLE
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0169 E  CHARACTER STRING LENGTH n IS INVALID
ADM3252 W  CHARACTER   X'xx'   REPLACED   BY   SHIFT-IN
           CHARACTER
```

```
ADM3253 W  DBCS  CHARACTER  X'xxxx'  IS  INVALID  AND
           REPLACED BY A BLANK
ADM3264 W  DBCS CHARACTER STRING LENGTH n MUST BE EVEN
```

GSCHAR

Function

To draw a character string at a specified point.

GSCHAR (x, y, length, string)	
APL code	524
GDDM RCP code	X'0C0C0500' (202114304)

Parameters

x *(specified by user) (short floating point)*
Specifies the starting x position of the string in world coordinates.

y *(specified by user) (short floating point)*
Specifies the starting y position of the string in world coordinates.

length *(specified by user) (fullword integer)*
The length of the **string** parameter in bytes.

string *(specified by user) (character)*
Contains the characters to be drawn. Character codes are compatible with those in alphanumeric fields. (See the ASTYPE call.)

Character codes X'01', X'02', X'03', X'05', X'06', X'07', X'09', X'0B', X'0C', X'0D', X'11', X'12', X'13', X'14', X'2B', and X'FF' have reserved meanings and should not be used. X'0E' and X'0F' are shift-out and shift-in characters for double-byte character sets (DBCS), or are invalid if the MIXSOSI external default is set to NO. X'15' is interpreted as a new-line character. The *nth* new-line character in the string sets the current position to n character-box heights below that at the start of the string. "Below" is more strictly defined by a line whose direction is 90 degrees clockwise from the character baseline. For some more discussion, see GSCA and GSCD.

The symbols displayed depend on the current character mode. For a description of which symbol sets are used for each of the possible modes, see GSCM.

In character-modes 1 and 2, character code X'00' is device dependent. This will usually be a blank, but on some devices this character code will be undefined. In character-mode 3, code X'00' is displayed using the code-point definition from the vector symbol set selected to define the character; see GSCS and GSLSS.

In character-modes 2 and 3, if a DBCS symbol set is loaded, or if GSCS(8) is specified, the character string is treated as containing all double-byte characters. To obtain a mixed graphic character string of SBCS and DBCS, the external default MIXSOSI=YES must be specified, and the characters delimited by shift-out (SO) (X'0E') and shift-in (SI) (X'0F') control codes (see GSSEN). The DBCS symbol set selected will be the default DBCS symbol set, as named by the DBCSDNM external default. The shift-out and shift-in control codes do not have a background for purposes of background mix.

Mixed character strings and character strings with a length of more than 240 characters drawn using a default character angle, character box, character-box spacing, character shear, or character direction will have undefined results when those default attributes are changed by use of a drawing defaults section. The text alignment attribute for such character strings are fixed at the time the character string is created and if the default text alignment is used, changes to the default text alignment are not honored.

Description

Draws a character string starting at the specified point.

```
GSCHAR(x,y,length,string)
```

is equivalent to:

```
GSMOVE(x,y)
GSCHAP(length,string)
```

If the window is normal (that is, if x coordinates increase from left to right and y coordinates increase from the bottom to the top), the character angle set by GSCA is zero (GSCA(1,0)), and the width and height of the character box (set by GSCB) are both positive, each character in the string is positioned within a character box so that its lower left-hand corner is at the current position. The current position is advanced along the character baseline after each character is drawn to give the starting position for the next character.

The spacing and size of the characters is controlled by the current character-box size (GSCB), the current character-box spacing (GSCBS), and the current text alignment (GSTA). The style of the characters depends on the current character set (GSCS). The angle at which the string is drawn is determined by the current character angle (GSCA), the character shear is controlled by GSCH, and the character direction is controlled by GSCD.

Spacing works as follows. After GSCHAR or GSCHAP has drawn a nonproportionally spaced character, the current position is moved along by an amount equal to the width of the character box. After drawing a proportionally spaced character, the movement is a fraction of the character box width. This fraction is the ratio between the character's assigned width and the maximum, as recorded in the definition of the character.

Again assuming a character angle of zero degrees, if either:

The window is inverted so y increases from the top to the bottom, or
The character-box height is negative,

The top left-hand corner of the character box is at the current position.

Similarly, if either

The window is reversed so x increases from left to right, or
The character width is negative,

the bottom right-hand corner of the character box is at the current position. If the character angle is not zero, the position of the character box relative to the current position can be determined by deciding on the position if the angle were zero degrees (as described above) and then rotating the character box about the selected corner by the required number of degrees.

Vector symbols are always drawn using the standard default line type and line width. Shaded vector symbols are always filled with the standard default shading pattern.

The degree to which approximation of the position and size is allowed, as well as the area used during correlation of the character string, is controlled by the character-mode attribute (GSCM).

After the string has been drawn, the current position is the point at which the next character would have been drawn, had it existed. For characters drawn in character-mode 1 (using the following the call.

Note: If this current position is used (for example, as the starting point for a line or a subsequent string), the position is calculated for the primary device in use. This may appear wrong if the picture is transmitted to another device, such as a printer, with a different character size.

If clipping is enabled, and part of the character string lies outside the window, the results depend on the character mode, as follows:

- For character-mode 1 (string precision), if the start point is within the window, the complete character string is sent as output. (It may not all be visible if some of it lies outside the graphics field.) For plotters, each part of each character is treated separately; any part of any character that is outside the window is removed.
- For character-mode 2 (character precision), each character is treated separately. If the start point of a character lies within the window, it is sent as output; otherwise, it is discarded.
- For character-mode 3 (stroke precision), each part of each character is treated separately. Any part of any character that is outside the window is removed.

If clipping is not enabled, the result, if any part of the string is outside the window, is not defined.

Principal errors

```
ADM0111 W  DBCS SYMBOL SET 'a' NOT AVAILABLE
ADM0154 E  COORDINATE f IS INVALID
ADM0156 W  COORDINATE OUTSIDE PICTURE SPACE
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0169 E  CHARACTER STRING LENGTH n IS INVALID
ADM3252 W  CHARACTER   X'xx'   REPLACED   BY   SHIFT-IN
           CHARACTER
ADM3253 W  DBCS  CHARACTER  X'xxxx'  IS  INVALID  AND
           REPLACED BY A BLANK
ADM3264 W  DBCS CHARACTER STRING LENGTH n MUST BE EVEN
```

GSCLP

Function

To enable and disable clipping.

GSCLP	**(control)**
APL code	501
GDDM RCP code	X'0C0C0203' (202113539)

Parameters

control *(specified by user) (fullword integer)*
Controls the clipping mode. Possible values are:
0 Clipping disabled (the default)
1 Clipping enabled (precise clip)
2 Clipping enabled (rough clip)

Description

Enables and disables clipping to the current clipping rectangle. The setting applies only to the current page.

When clipping is enabled, parts of the picture that fall outside the clipping rectangle are, in general, removed. The data boundary, if it has been defined (with GSBND), is the clipping rectangle; otherwise, the current graphics window (determined by GSWIN or GSUWIN) is used.

Clipping can be performed to two degrees of precision, according to the clipping mode selected:

Precise Clipping (Mode 1)
If a primitive (for example a line) falls completely outside the clipping rectangle, it is discarded. If a primitive falls partly within and partly outside the clipping rectangle, the primitive is clipped to keep only the part within the rectangle. Lines and arcs are shortened to stop at the rectangle boundary. Markers appear if their center points are within the rectangle. For character strings, the result depends on the character mode; see GSCHAR.

Rough Clipping (Mode 2)
The general rule for all primitives is that if any part of the primitive lies within the clipping rectangle, the whole primitive is retained without change. Otherwise the primitive is discarded.

For primitives such as fillets (see GSPFLT) and polylines (see GSPLNE) which are constructed from a series of simpler primitives, the clipping rule is applied, individually, to each component of the primitive.

When the primitive is curved (for example, an arc or an ellipse), the clipping rule is applied to the major and minor axes of the curve.

Note: If clipping is ***not*** enabled, all parts of the drawing should lie within the window. If they do not, the results are not defined.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0153 E  CONTROL VALUE n IS INVALID
```

GSCLR

Function

To clear the graphics field.

GSCLR	
APL Code	506
GDDM RCP code	X'0C0C0303' (202113795)

Parameters

None.

Description

Clears the graphics field by deleting all of its graphics segments.

The effect is not apparent on the display until the next transmission to the device is required.

The picture space can be redefined following a GSCLR call if the GSPS call is issued before any other call is issued that would cause the picture space to default. If no explicit GSPS call is issued after GSCLR, the picture space that is retained is the one that was in effect before the GSCLR call.

Principal errors

None.

GSCM

Function

To set current character mode.

GSCM **(n)**	
APL code	513
GDDM RCP code	X'0C0C0705' (202114821)

Parameters

n *(specified by user) (fullword integer)*
Defines the character mode. Possible values are:

0 The drawing default; initially, the same as 1.

1 The hardware character generator is used to position individual characters in the string, with any limitations this may impose. The character set defined by the GSCS call must be a standard character set (either character set 0, or if featured on the device, the APL character set 1), or a previously loaded PS set or image symbol set.

If clipping is enabled (see GSCLP) and part of the character string lies outside the window, the complete character string is sent as output if the start point is within the window. (It may not all be visible if some of it lies outside the graphics field.) For plotters, each part of each character is treated separately; any part of any character that is outside the window is removed.

2 The values of the character box, character-box size, character angle, character direction, and text alignment specifications are used to position individual characters in the string. However, the size and orientation are determined when the symbol set is created. The character set must be the standard set 0 or an image symbol set loaded previously by GSDSS or GSLSS. If clipping is enabled (see GSCLP), each character is treated separately. If the start point of a character lies within the window, it is sent as output; otherwise, it is discarded.

Correlation for mode-2 text is based on the text box around the characters and does not necessarily include all of an image character.

If the user does not specify a suitable character set for mode 2 text, the default mode 1 character set is used.

3 For all devices, the character box, character-box size, character angle, character direction, and text alignment specifications are followed exactly. Characters are rotated so that they are at right angles to the character baseline. The character set must be the standard vector set 0 or a vector symbol set loaded previously by a GSDSS or GSLSS call.

If clipping is enabled, any portion of a character that is outside the window is truncated; see GSCLP.

Description

Controls the character mode to be used in drawing a character string in the graphics field. The character-mode attribute remains in effect until it is changed by another GSCM call.

When a segment is created by the GSSEG call, the character-mode attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the character-mode attribute is reset to the value that was in effect when the segment was created.

For limitations on specific devices, see “Graphics text” in Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSCOL

Function

To set current color.

GSCOL (n)	
APL code	514
GDDM RCP code	X'0C0C0701' (202114817)

Parameters

n *(specified by user) (fullword integer)*
A fullword integer in the range –2 through 32767.

- **–2** White; if a device does not explicitly support white, it is emulated using an appropriate supported color (for example, color 8 (background) for plotters and printers, and color 7 (neutral) for displays).
- **–1** Black; if a device does not explicitly support black, it is emulated using an appropriate supported color (for example, color 7 (neutral) for plotters and printers, and color 8 (background) for displays).
- **0** Drawing default; initially, these are the standard defaults for these devices, namely green on displays (orange on a 3290), black on printers, and the maximum pen stall number on plotters.
- **1** Blue
- **2** Red
- **3** Pink (magenta)
- **4** Green
- **5** Turquoise (cyan)
- **6** Yellow
- **7** Neutral; white on displays and black on printers.
- **8** Background; black on displays and (usually) white on printers and plotters.
- **9** Dark blue
- **10** Orange
- **11** Purple
- **12** Dark green
- **13** Dark turquoise (cyan)
- **14** Mustard
- **15** Gray
- **16** Brown

Description

Sets the current value of the color attribute. The color attribute remains in effect until it is changed by another GSCOL call.

Note: The color set by GSCOL applies to ***all*** subsequent graphics primitives until it is changed. To ensure their correct colors, the color should be set to 7 (neutral) before multicolored characters, markers, or shading patterns are used.

When a segment is created by the GSSEG call, the color attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the color attribute is reset to the value that was in effect when the segment was created.

For information about the limitations of and variations on specific devices, see “Graphics colors” on page 245.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSCOPY

Function

To send graphics to alternate device.

GSCOPY (depth, width)	
APL code	605
GDDM RCP code	X'0C180002' (202899458)

Parameters

depth *(specified by user) (fullword integer)*
The depth of the alternate device’s copy area, in rows.
width *(specified by user) (fullword integer)*
The width of the alternate device’s copy area, in columns.

Both **depth** and **width** must be positive, and they must not

exceed the maximum size of the page allowed on the alternate device; this is defined by the device token specified in a DSOPEN call or defaulted to '⋆ ' in FSOPEN.

Description

Sends (copies) the contents of the graphics field to the currently open family-1. family-2, or family-4 cell-based alternate device. The graphics are of the specified size on the alternate device's page. Thus, the size of the copy, in character-cell units, can be different from that of the graphics field on the current page, and a larger graphics picture may be obtained than might otherwise have been the case using FSCOPY.

By default, the aspect ratio of the picture space is preserved on the alternate device; that is, it has the same aspect ratio as that on the primary device. This applies even if the picture space size has been defaulted. The GSARCC call can be used to change the aspect ratio so that it matches that of the graphics field.

GDDM disables clipping when the graphics are sent to the alternate device; therefore, if different symbol sets are used on the original device and the alternate device, the results might not be exactly the same.

Comments in the description of FSCOPY concerning symbol sets, graphics text, and graphics images also apply to GSCOPY.

Notes:

1. The GSCOPY call causes a default primary device to be established, if one is not already in use.
2. The GSCOPY call is ignored for the IBM 5080 Graphics System.

Principal errors

```
ADM0070 E  NO ALTERNATE DEVICE
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0277 E  '{FSSAVE|FSSHOW|FSSHOR|FSCOPY|
           GSCOPY|DSCOPY|MAPPING|DSFRCE|FSFRCE}' IS NOT
           SUPPORTED FOR THIS DEVICE
ADM0283 E  INVALID OUTPUT SIZE
ADM0284 E  NO GRAPHICS FIELD DEFINED
```

GSCORR

Function

To correlate a tag to a primitive.

Note: This call is not recommended for new programs. It is obsolete and has been superseded by GSCORS.

GSCORR	**(ctype, x, y, atype, c1, spec-array, c2, seg-array, tag-array, num-hits)**
APL code	638
GDDM RCP code	X'0C0C1500' (202118400)

Parameters

ctype *(specified by user) (fullword integer)*
The type of segment on which correlation is to be performed. Possible values are:

- **0** Only visible and detectable segments with nonzero identifiers are correlated. This gives the same correlation criteria as used by a pick device; see GSIPIK.
- **1** All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The x and y coordinates of the position of the center of the aperture (in world coordinates).

atype *(specified by user) (fullword integer)*
The type of aperture to be used. Possible values are:

- **0** The default; same as 1.
- **1** Scaled pick aperture
  The **spec-array** parameter must contain a single element, which is a uniform scaling factor that is applied to the device's default pick aperture; see GSIPIK.
- **2** Rectangular aperture
  The **spec-array** parameter must contain two world-coordinate values, giving the width and height (respectively) of a rectangular aperture. The center of the rectangle is positioned at the point given by the values in the **x** and **y** parameters during correlation.

c1 *(specified by user) (fullword integer)*
The number of elements in the **spec-array** parameter.

spec-array *(specified by user) (array of short floating-point numbers)*
An array of floating-point numbers as defined by the **atype** parameter.

c2 *(specified by user) (fullword integer)*
The number of elements in the **seg-array** and **tag-array** parameters.

seg-array *(returned by GDDM) (an array of fullword integers)*
An array of segment identifiers. The elements are returned in reverse drawing order. See the description of the **num-hits** parameter.

tag-array *(returned by GDDM) (an array of fullword integers)*
An array of primitive tags. The elements are returned in reverse drawing order. See the description of the **num-hits** parameter.

num-hits *(returned by GDDM) (fullword integer)*
The number of hits returned in the **seg-array** and **tag-array** parameters.

A “hit” is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. GDDM counts this as a single hit; the hit is only recorded once in the **seg-array** and **tag-array** parameters returned by GDDM. The **num-hits** parameter, therefore, returns this number of distinct hits.

The vectors **seg-array** and **tag-array** are set to the hits that are found, up to the maximum defined in the **c2** parameter. Corresponding elements form the “hit” pair. The number returned in **num-hits**, therefore, contains the number of pairs set if the **c2** parameter is greater than the number of hits detected. The number of elements set in the **seg-array** and **tag-array** parameters is precisely the number returned in **num-hits**, and never exceeds the value specified in **c2**.

If the same value is returned in the **num-hits** parameter as is specified in the **c2** parameter, there may be more hits that cannot be returned in **seg-array** and **tag-array**. If all hits are important, specify arrays that are large enough to contain the maximum number of hits expected.

Description

Returns (correlates) a segment and tag pair for each tagged primitive in the current graphics field, that intersects the specified aperture. The aperture is compared with **all** eligible primitives, whether they belong to the current viewport or not.

Only primitives with a nonzero tag in segments with a nonzero identifier are correlated using this call.

GSCORR sets the default graphics field, picture space, viewport, and window if they have not already been defined.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0195 E  APERTURE TYPE n IS INVALID
ADM0196 E  APERTURE SPECIFICATION f IS INVALID
ADM3250 E  CORRELATION TYPE n IS INVALID
```

GSCORS

Function

To correlate segments and tags.

GSCORS	**(ctype, x, y, atype, c1, spec-array, c2, depth, seg-array, tag-array, num-hits)**
APL code	655
GDDM RCP code	X'0C0C1501' (202118401)

Parameters

ctype *(specified by user) (fullword integer)*
The type of segment on which correlation is to be performed. Possible values are:

0 Only visible and detectable segments with nonzero identifiers are correlated. This gives the same correlation criteria as used by a pick device; see GSIPIK.

1 All segments with nonzero identifiers are correlated, regardless of the detectability and visibility attributes of the segments.

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The x and y coordinates of the position of the center of the aperture (in world coordinates).

atype *(specified by user) (fullword integer)*
The type of aperture to be used. Possible values are:

0 The default; same as 1.

1 Scaled pick aperture
The **spec-array** parameter must contain a single element, which is a uniform scaling factor that is applied to the device’s default pick aperture; see GSIPIK.

2 Rectangular aperture
The **spec-array** parameter must contain two world-coordinate values, giving the width and height (respectively) of a rectangular aperture. The center of the rectangle is positioned at the point given by the values in the **x** and **y** parameters during correlation.

c1 *(specified by user) (fullword integer)*
The number of elements in the **spec-array** parameter.

spec-array *(specified by user) (array of short floating-point numbers)*
An array of floating-point numbers as defined by the **atype** parameter.

c2 *(specified by user) (fullword integer)*
The maximum number of hits that can be returned in the **seg-array** and **tag-array** parameters.

depth *(specified by user) (fullword integer)*
The number of segment and tag pairs to be returned for each hit.

seg-array *(returned by GDDM) (an array of fullword integers)*
An array of (**c2** times **depth**) elements containing segment identifiers. For each hit, a set of **depth** values are returned.

tag-array *(returned by GDDM) (an array of fullword integers)*
An array of (**c2** times **depth**) elements containing primitive tags. For each hit, a set of **depth** values are returned.

num-hits *(returned by GDDM) (fullword integer)*
The number of hits returned in the **seg-array** and **tag-array** parameters.

A "hit" is an instance of a segment identifier and tag pair for which the primitives lie completely or partially within the specified aperture. Two different primitives in the same segment might have the same tag, and would therefore produce the same hit. GDDM counts this as a single hit; the hit is only recorded once in the **seg-array** and **tag-array** parameters that GDDM returns. The **num-hits** parameter, therefore, returns this distinct number of hits.

The tables **seg-array** and **tag-array** are set to the hits that are found, up to the maximum defined in the **c2** parameter. Corresponding sets of elements form the "hit" pairs. The number returned in **num-hits** therefore contains the number of sets of **depth** pairs set if the **c2** parameter is greater than the number of hits detected. The number of elements set in the **seg-array** and **tag-array** parameters is the number returned in **num-hits** multiplied by the **depth**.

If the same value is returned in the **num-hits** parameter as is specified in the **c2** parameter, there may be yet more hits that cannot be returned in **seg-array** and **tag-array**. If all hits are important, specify arrays that are large enough to contain the maximum number of sets of hits that are expected.

Description

Returns (correlates) segment and tag pairs for each tagged primitive in the current graphics field, that intersects the specified aperture. The data returned for each "hit" (or correlation) consists of a set of segment and tag pairs. The first pair is the correlated pair, and successive segment and tag pairs represent successive levels of nesting, repeated until the root segment (non nested) is reached.

The aperture is compared with **all** eligible primitives, whether they belong to the current viewport or not.

Only primitives with a nonzero tag in segments with a nonzero identifier are correlated using this call.

GSCORS sets the default graphics field, picture space, viewport, and window if they have not already been defined.

The depth value specifies the number of sets of segment and tag pairs to be returned for each hit. If the root segment is reached before **depth** values, the remaining values are set to zero. If more than **depth** values are available, only that number are returned.

Example:

```
Start segment 1
Tag 10
Call 2
End segment 1

Start segment 2
Tag 20
Call 3
Tag 21
.....              Pick 1
.....
End segment 2

Start segment 3
Tag 30
.....              Pick 2
......
End segment 3
```

For pick 1 return, at depth=2:

Segment	Tag
2	21
1	10

For pick 2 return, at depth=5:

Segment	Tag
3	30
2	20
1	10
0	0
0	0

If depth was less than 3, only the first "depth" values would be returned.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0195 E  APERTURE TYPE n IS INVALID
ADM0196 E  APERTURE SPECIFICATION f IS INVALID
ADM3250 E  CORRELATION TYPE n IS INVALID
```

GSCP

Function

To set current position.

GSCP (x, y)

APL code	668
GDDM RCP code	X'0C0C1319' (202117913)

Parameters

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
Specify, in world coordinates, a point to which the current position is to be moved. The new value of the current position is (x,y).

Description

Sets the current position to the specified point.

This call is equivalent to the GSMOVE call, except that, if the current attribute mode is 0 (preserve attributes; see GSAM), the current position is saved before being set to the new value, so that it can be restored using a GSPOP call.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
```

GSCPG

Function

To set 4250 current code page.

GSCPG	**(type, code-page-name)**
APL code	215
GDDM RCP code	X'0C040D00' (201592064)

Parameters

type *(specified by user) (fullword integer)*
The type and usage of the code page. The only value that can be specified is:

5 A code page for a 4250 page printer.

code-page-name *(specified by user) (8-byte character string)*
The name (left-justified) of the code page to be used. The GDDM-supplied default code-page name is AFTC0395. The user can specify a code-page name of "*" to restore the default code page as the current one.

A selection of code-page names supplied by IBM with, and in support of, the 4250 page-printer fonts are:

AFTC0293 APL
AFTC0361 International
AFTC0363 Pi Font
AFTC0382 Austria, Germany, Switzerland (German)
AFTC0383 Belgium
AFTC0384 Brazil
AFTC0385 Canada (French)
AFTC0386 Denmark, Norway
AFTC0387 Sweden
AFTC0388 France, Luxembourg, Switzerland
AFTC0389 Italy, Switzerland (Italian)
AFTC0390 Japan
AFTC0392 Spain, Philippines
AFTC0393 Latin America (Spanish-speaking)
AFTC0394 United Kingdom, Ireland, Australia, Hong Kong S.A.R., New Zealand
AFTC0395 United States, Canada (English)
AFTC0829 Math Symbols.

Description

Specifies the name of the code page that is to become the current code page.

The current code page defines the set of characters and their associated code points that form the subset of a page printer font specified in a subsequent GSLSS call.

Note: The GSCPG call must be made before the associated GSLSS call.

A type-5 (4250) code-page name is supplied in the form AFTCnnnn. The code-page name specified overrides the default code-page name defined for the installation in GDDM's external defaults (the CPN4250 option). A code page is current until either another GSCPG call is issued or the device is closed.

Principal errors

```
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0307 E  FILE 'a' NOT FOUND
```

GSCS

Function

To set current symbol set.

GSCS	**(symbol-set-id)**
APL code	515
GDDM RCP code	X'0C0C0706' (202114822)

Parameters

symbol-set-id *(specified by user) (fullword integer)*
The symbol-set identifier.

For **character-mode 1**, it is:
0 The drawing default character set.
1 The APL set, if featured.
n Corresponding to the symbol-set identifier specified for a programmed symbol (PS) set that has previously been loaded into the device by a PSDSS, PSLSS, or PSLSSC call, or to an image symbol set that has been previously loaded by a GSLSS or GSDSS call.

n corresponding to the symbol-set identifier specified for a 4250 printer font that was the subject of a previous GSLSS call (**type**=5).

For **character-mode 2**, the symbol-set identifier is:

0 The drawing default character set.

n Corresponding to the symbol-set identifier specified for an image symbol set (ISS) defined to GDDM by GSDSS or GSLSS (**type**=1)

n Corresponding to the symbol-set identifier specified for a 4250 printer font that was the subject of a previous GSLSS call (**type**=5).

n Corresponding to the symbol-set identifier specified for a DBCS image symbol set (ISS) defined to GDDM by GSLSS (**type**=8).

For **character-mode 3**, the symbol-set identifier designates the particular vector symbol set (VSS) to be used. The symbol-set identifier can be:

0 Identifying the drawing default vector symbol set

n Corresponding to the symbol-set identifier specified for a vector symbol set defined by GSDSS or GSLSS (**type**=2).

n Corresponding to the symbol-set identifier specified for a DBCS vector symbol set (VSS) defined to GDDM by GSLSS (**type**=9).

Notes:

1. Character strings passed to GSCHAP or GSCHAR while a DBCS character set is selected are treated as DBCS characters.
2. Specifying **symbol-set-id**=8 or MIXSOSI=YES selects the default DBCS symbol sets, as named by the DBCSDNM external default.

Description

Sets the current value of the symbol-set attribute. The symbol-set attribute remains in effect until it is changed by another GSCS call.

When a segment is created by the GSSEG call, the attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the attribute is reset to the value that was in effect when the segment was created.

For limitations on specific devices, see Chapter 4, “Device variations” on page 241.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSDEFE

Function

To end drawing defaults definition.

GSDEFE	
APL code	661
GDDM RCP code	X'0C0C1901' (202119425)

Parameters

None.

Description

Ends the drawing defaults definition set by the GSDEFS call. Any following attribute calls are explicit settings of that attribute.

Principal errors

```
ADM3261 E  END DRAWING DEFAULTS DEFINITION IGNORED
```

GSDEFS

Function

To start the drawing defaults definition.

GSDEFS (count, array)	
APL code	660
GDDM RCP code	X'0C0C1900' (202119424)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in the **array**.

array *(specified by user) (an array of fullword integers)*
An array of options for the GSDEFS call. Currently, there is one element:

Type
Specifies the action GDDM is to take with the new defaults definition:

0 The default; same as 1.

1 The defaults definition specifies default values that should be merged into existing default values. Any attribute default specifically set overrides an existing default value; but any that are not referenced are not changed.

2 The default definition totally overrides any existing default definitions. Any attribute default not specified within this definition is set to the standard default value, which is device-dependent.

Description

Starts the drawing defaults definition.

Attribute calls following this call, until a GSDEFE call, set the drawing default to the value specified. If any attribute call is used more than once, the last occurrence takes precedence. The drawing defaults definition section can be used to completely replace the existing drawing default values, or to merge new values into them (see parameter definition below).

Attributes that can have their default values set are:

- Background mixing mode (GSBMIX)
- Character angle (GSCA)
- Character box (GSCB)
- Character box Spacing (GSCBS)
- Character direction (GSCD)
- Character mode (GSCM)
- Character shear (GSCH)
- Color (GSCOL)
- Line type (GSLT)
- Line width (GSLW or GSFLW)
- Marker box (GSMB)
- Marker symbol (GSMS)
- Mixing mode (GSMIX)
- Primitive tag (GSTAG)
- Shading pattern (GSPAT)
- Symbol set (GSCS)
- Text alignment (GSTA)

If drawing defaults have not been defined, the standard attributes, which are device-dependent, are used. To reset a specific drawing default value to its initial state, in most cases, zero values can be specified. However, for character box and marker box, the standard default for the device must be specified.

Any attribute query calls within this section return the current default value.

Once a default has been set, in most cases it can be referred to using a value of zero when explicitly setting the current attribute value, for example `GSCOL(0)`. Attributes that differ from this general case are:

Primitive tag
The default value is assumed at the start of a segment, and cannot be set explicitly. Therefore, for example, if the default tag is set to 1, all primitives in segments have a tag of 1, unless it is explicitly changed by a GSTAG call. A `GSTAG(0)` call does not set the current tag to the default value, but instead sets the current tag to zero.

Character angle
This attribute is set by specifying an x and a y component. If both are zero, the drawing default value is used.

Character box
This attribute is set by specifying an x and a y component. The default value is assumed at the start of a segment. If a GSCB call is then issued, it is not possible to restore the default value for the character box within the segment.

The drawing default for the character box is converted from world coordinates to device-dependent coordinates at the time the default is defined. Therefore, a subsequent change to the world-coordinate system using a GSWIN or GSUWIN call does not alter the actual size of the character box.

Consider the following example:

```
CALL GSUWIN(0,100,0,100);
CALL GSDEFS(1,1);
CALL GSCB(10,10);    /* Set default character
                                box to (10,10)  */
CALL GSDEFE;

CALL GSUWIN(0,200,0,200); /* Change the window */
CALL GSSEG(1);
CALL GSCM(3);        /* Character mode 3 -
                        current character box
                        assumes default value
                        when segment is opened */
CALL GSCHAR(....);
CALL GSCB(10,10);    /* Set current character
                                box to (10,10)  */
CALL GSCHAR(....);
CALL GSSCLS;
```

The first character string has the drawing default character box which was defined as (10,10) when a uniform window of (0,100,0,100) was in effect. The subsequent change to the window does not affect the default, so this first character string is drawn as if the change to the window had not occurred. The second character string has a character box defined as (10,10) in the new window (0,200,0,200), and is, therefore, half the size of the first string.

Character shear
This attribute is set by specifying an x and a y component. If both are zero, the drawing default value is assumed.

Marker box
This attribute is set by specifying an x and a y component. As for character box, once the marker box has been set within a segment, it cannot be set back to default.

The drawing default for the marker box is converted from world coordinates to device-dependent coordinates at the time the default is defined. Therefore, a subsequent change to the world coordinate system using a GSWIN or GSUWIN call does not alter the actual size of the default marker box. (See the description for character box above.)

Character-box spacing
The default value is assumed at the start of a segment, and cannot be set explicitly. Thus, for example, if the default character-box spacing is set to (1, 1), all character primitives will assume this value for the attribute, unless it has been

explicitly changed by a GSCBS call. A GSCBS(0,0) call does not set the current character-box spacing attribute to the default value, but instead sets it to be a spacing of (0, 0).

If a specific value is not set within the drawing default definition, the standard default value is used. The standard default values are device-dependent. For example, if the color is not set within the drawing defaults section, then the default color is green on multicolor displays, but black on printers.

The graphics field is defaulted if it has not already been set. Clearing the graphics field by use of the GSCLR call does not affect the current drawing defaults.

Drawing default values affect the entire picture. Any primitives with default attributes, whether they are specified before a drawing defaults definition or after, assume the attribute values specified by that definition.

Any primitives outside segments are discarded.

Thus, a call to GSDEFS can affect the appearance of existing primitives, and can cause a redraw of the picture. Subsequent drawing defaults definitions override the existing drawing default values for each attribute specified between the GSDEFS and GSDEFE calls. The **type** parameter indicates whether default values for attributes not specified in the definition are to be left unchanged or set to the standard default.

Under some conditions, changes to drawing defaults for some attributes are not honored.

- Character box, character angle, and character shear attributes are combined to form a geometric transformation. If one of these values is set, it causes the other two to be set as well. When a segment is opened, these attributes assume the default value. If none of them are explicitly set, subsequent changes to the defaults are reflected in the character primitives. However, as soon as one of them is set, all three are “fixed” and any subsequent changes to the default values for these attributes are not reflected in the following character primitives.
- If either the x or the y direction for the window is reversed, the character transform (that is, Character Box, Character Angle and Character Shear) is “fixed” when a segment is opened. Subsequent changes to default values for these attributes are not reflected for character primitives in segments that are created while the window is reversed.
- If a character string is very long (more than 240 characters) or is a mixed character string (that is, it contains shift-out (SO) or shift-in (SI) control codes, representing the change from single-byte to double-byte characters, or the converse) the text alignment attribute is “fixed” at the time the string is drawn. Subsequent changes to the default value for text alignment are not honored for such strings. Subsequent changes to the default values of character box, character angle, character-box spacing, and character direction will have undefined results on very long character strings or mixed character strings that have been drawn using the default values for the attributes listed.

Between the GSDEFS and the GSDEFE calls, the following calls cannot be issued:

FSCOPY	GSCALL	GSSDEFS	GSLOAD
GSMSC	GSPOP	GSSAGA	GSSATS
GSSAVE	GSSCPY	GSSCT	GSSDEL
GSSEG	GSSINC	GSSORG	GSSPOS
GSSPRI	GSSTFM	GSSVL	

and (if the device is a queued printer)

ASREAD FSCHEK FSFRCE.

Calls that draw lines, arcs, fillets, characters, markers, areas, and images are also not allowed between the GSDEFS and GSDEFE calls.

If the GSSAVE call is used to save a picture, any drawing default values that have been set are saved with the picture. If a picture containing drawing defaults is loaded using the GSLOAD call, the application can choose to ignore the defaults, append them to the current drawing defaults, use them to over-ride the current drawing defaults, use them to totally replace the current drawing defaults, or incorporate them into the segment data to be loaded; see GSLOAD.

If this last option is selected, any attribute that references a default is changed to set the drawing default value explicitly, and the default values currently in use by the application are not altered. Note that this option does not guarantee that the loaded picture completely reproduces the picture at the time it was saved. This is because any drawing default values not specified in the saved file, results in the current drawing default values being used for that attribute. If these have not been set, the standard default for the current device is used, and for some attributes this value is device-dependent (see above). Also, any segment that is both chained and called will not correctly inherit from its caller attributes for which a default value was specified. This could also result in a change in the picture.

Primitives that have been clipped may not fully reflect changes to drawing defaults. For example, if the default character set is changed, then a clipped string may not be re-displayed in the new character set. Similarly, a character string may change length if different proportionally-spaced fonts are used. If a string was totally clipped out because all characters were outside the clip area, then changing the font does not result in the characters appearing, even though the new font may have caused the length of the string to be such that it would have extended into the clip area. However, if no clipping is in effect, then all changes to the drawing defaults are fully reflected.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM3260 E  INVALID FUNCTION DURING DRAWING DEFAULTS
           DEFINITION
```

GSDSS

Function

To load a graphics symbol set from the application program.

GSDSS	**(type, symbol-set-name, symbol-set-id, length, data)**
APL code	201
GDDM RCP code	X'0C040301' (201589505)

Parameters

type *(specified by user) (fullword integer)*
The type and usage of this symbol set. Possible values are:

- **1** ISS to be retained by GDDM for dot-matrix graphics text.
- **2** VSS to be retained by GDDM for generation of vector graphics text.
- **3** Shading pattern set (must be ISS).
- **4** Marker symbol set (may be either ISS or VSS).

symbol-set-name *(specified by user) (8-byte character string)*
The name (left-justified) of the symbol set. For this call, the symbol-set name serves only to identify the set. No file operations are performed. The name is returned by GSQSS, and is also used to locate an equivalent character set if a copy is made to a printer. If neither of these operations is to be used, the name may be left blank.

symbol-set-id *(specified by user) (fullword integer)*
The identifier by which this symbol set is referred to in later statements. The GSDSS call checks that the identifier of a type-1 symbol set is not the same as an existing type-5 symbol-set identifier that has been loaded by a GSLSS call.

The allowable symbol-set identifiers are:

0	Pattern or marker symbol set
65 through 223	Other symbol sets.

Each loaded symbol set should have a unique identifier with respect to all other symbol sets loaded by GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC calls. This avoids any uncertainty that might arise from a device treating different types of symbol sets as equal candidates for displaying a character string. If, however, a symbol-set identifier is the same as one that has previously been issued for the same type, the new definitions replace the previous ones.

length *(specified by user) (fullword integer)*
The length of data storage provided for the value given in **data**.

data *(specified by user) (character)*
The symbol-set definitions to be loaded; see Chapter 8, “Symbol set formats” on page 275.

Description

Loads a set of symbol definitions from data passed by the application program. The symbol set may be an image symbol set (ISS) or a vector symbol set (VSS). For a plotter, vector symbol sets may be loaded; image symbol sets can also be loaded for characters and markers, but not for shading patterns.

The definitions are retained by GDDM for graphics use.

For information about restrictions on various devices, see “Graphics area shading” on page 247.

Principal errors

```
ADM0115 E  SYMBOL SET 'a' LENGTH n IS INVALID
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0123 E  SYMBOL SET n1 HAS INVALID FORMAT. REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0128 W  SYMBOL SET n OPTION UNSUPPORTED
ADM0135 E  SYMBOL SET n TYPE UNSUPPORTED
ADM3157 E  SYMBOL SET IDENTIFIER n ALREADY IN USE
```

GSELPS

Function

To draw an elliptical arc.

GSELPS	**(p, q, tilt-angle, xe, ye)**
APL code	551
GDDM RCP code	X'0C0C0601' (202114561)

Parameters

p *(specified by user) (short floating point)*
The major axis length; cannot be zero.
q *(specified by user) (short floating point)*
The minor axis length; cannot be zero.

tilt-angle *(specified by user) (short floating point)*
The inclination of the major axis to the x axis in degrees (the major axis is that with length P, regardless of which axis is the longer). Positive tilt angles result in counter-clockwise rotation of the ellipse, negative tilt angles result in clockwise rotation.
xe *(specified by user) (short floating point)*
The x coordinate of the end point of the arc.
ye *(specified by user) (short floating point)*
The y coordinate of the end point of the arc.

Description

Draws a curve that starts at the current position and follows an elliptic curve until it reaches the end point; see Figure 7. The ellipse has major and minor axis lengths given by the parameters **p** and **q**, and the major axis (p) is inclined to the x axis by the axis tilt angle supplied.

The arc is considered to be constructed as follows. An ellipse with the given major and minor axis lengths is constructed with the major axis lying along the x axis. The ellipse is then rotated by the axis tilt angle and moved so that current position and the end point both lie on the curve.

In general, there are two possible positions for the ellipse center. One is to the right and one is to the left of the line drawn from the start point to the end point viewed in that direction. If **p** and **q** have the same sign, the center point to the left of the line is chosen and the arc is drawn counter-clockwise about it. If **p** and **q** have opposite signs, the arc is drawn clockwise about the center to the right of the line.

The arc drawn is never longer than half an ellipse.

The arc has the color, line width, and line type given by the current values of these attributes.

The current position is set to the end of the arc.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
ADM0176 E  AXIS LENGTH f OUT OF RANGE
ADM0177 W  ARC RADIUS TOO SMALL
```

GSENAB

Function

To enable or disable a logical input device.

GSENAB	**(device-type, device-id, control)**
APL code	572
GDDM RCP code	X'0C0C0D00' (202116352)

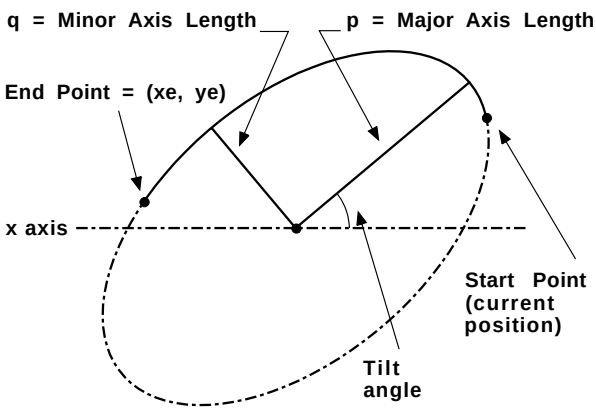


Figure 7. How an ellipse is drawn (GSELPS)

Parameters

device-type *(specified by user) (fullword integer)*
The type of the logical input device to be enabled. Possible values are:

1 Choice.
2 Locator.
3 Pick.
4 String
5 Stroke

device-id *(specified by user) (fullword integer)*
The identification of the logical input device to be enabled.

Possible values for the device identifications for the different device types are:

Choice

0 The ENTER key.
1 The PF keys.
2 Alphanumeric light-pen detect
4 The PA keys.
5 The CLEAR key.
8 The data keys.
10 The mouse or puck buttons.

Values correspond to the values returned by ASREAD.

Locator

1 The mouse or tablet, or cursor keys.

Pick

1 The mouse or tablet, or cursor keys.

String

1 The string device provides an area into which characters can be typed.

Stroke

1 The mouse or tablet (if configured).

control *(specified by user) (fullword integer)*
An integer that shows the new state of the given input devices:

0 Disabled. No operator input expected from the device. If the device cannot physically be disabled, any input is ignored by GDDM. This is the initial state.
1 Enabled. Operator input is expected from the device and is to be allowed when subsequent GSREAD calls are issued.

Note: If all three parameters are set to zero (that is, `GSENAB(0,0,0)`), all previously enabled input devices are disabled.

Description

Enables a logical input device. This must be done before input of a particular type can be received; initializing a device does not enable it.

If a locator device is enabled, the initial position, specified in the GSILOC call is used for the position of the graphics cursor. If the locator was not initialized before being enabled, the default cursor is used and its position set to the center of the graphics field.

The locator is triggered by the terminal operator causing an attention interrupt (for example, using the ENTER key). For details of the trigger key used on a particular device, see GSREAD. If it is required to know which key is pressed, choice logical input devices must also be enabled.

If a pick device is enabled, the segment and tag specified in the GSIPIK call is used to identify a particular primitive, and the initial position of the graphics cursor is set accordingly. If no primitive is found, the position of the graphics cursor is set to the center of the graphics field, and an informational message is issued. If the pick was not initialized before being enabled, the position of the graphics cursor is set to the center of the graphics field.

The pick is triggered by the terminal operator causing an attention interrupt (for example, using the ENTER key). For details of the trigger key used on a particular device, see GSREAD. If it is required to know which key is pressed, choice logical input devices must also be enabled.

The string data is triggered by pressing the ENTER key or the PF keys. It is also triggered by the mouse or puck buttons if they are used to support other enabled logical input devices.

If a stroke device is enabled, the initial position specified in the GSISTK call is used to position the initial marker (the X symbol). If the stroke device is not initialized before it is enabled, the symbol is positioned at the center of the graphics field. The specified or defaulted initial position of a stroke device overrides that of a locator device and that of a locator device overrides that of a pick device.

When a logical input device is enabled or disabled, the default graphics field, picture space, viewport, and window are set if they were not specified or defaulted. When the graphics field is deleted or redefined, all logical input devices are reset to their default state (disabled with default initial values).

A device can only be ***enabled*** if it is currently ***disabled***.

Enabled devices can be manipulated by the operator whenever the current page is displayed. Input from logical input devices is received by use of the GSREAD call.

It is permissible to enable different logical input device types (for instance, a locator and a pick) at the same time, even though the same hardware needs to be used for both. The position indicated is both returned to the application as locator input, and used to select a primitive to be returned to the application. The initial value of the pick is ignored in this circumstance, and the echo for the pick is the same as that for the locator. If the locator is disabled, the pick remains at the locator position.

Multiple choice devices can be enabled, but multiple locator devices, multiple pick devices, multiple string devices, or multiple stroke devices are not allowed.

When multiple partitions are used, each page can have its own set of logical input devices. The user can interact with all of the partitions that have logical input devices enabled.

Note: All logical input devices are associated with a graphics field. If you redefine the graphics field after an input device has been enabled, or select another page, the device is disabled.

For information about restrictions on various devices, see “Graphics logical input devices” on page 247.

Principal errors

```
ADM0153 E  CONTROL VALUE n IS INVALID
ADM3201 E  INPUT DEVICE TYPE n IS INVALID
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3205 I  DEFAULT USED IN GRAPHICS CURSOR POSITIONING
ADM3206 E  LOCATOR COORDINATE f OUTSIDE PICTURE SPACE
ADM3208 E  INPUT  DEVICE  TYPE  n1  IDENTIFIER  n2  IS
           ALREADY ENABLED
ADM3211 E  ECHO SEGMENT IDENTIFIER n IS INVALID
ADM3212 E  DEVICE IS OUTPUT ONLY
ADM3213 E  APERTURE f IS INVALID
```

GSENDA

Function

To end a shaded area.

GSENDA	
APL code	525
GDDM RCP code	X'0C0C0409' (202114057)

Parameters

None

Description

Ends the construction of a shaded area. The construction is started by the GSAREA call. If necessary, a final line is constructed to close the area.

The current position is not changed, unless a closure line must be drawn, in which case the current position is moved to the end point of the line.

Principal errors

```
ADM0160 E  END AREA IGNORED
```

GSFLD

Function

To define the graphics field.

GSFLD	**(row, column, depth, width)**
APL code	502
GDDM RCP code	X'0C0C0000' (202113024)

Parameters

row *(specified by user) (fullword integer)*
The row position on the page of the top left-hand corner of the graphics field.

column *(specified by user) (fullword integer)*
The column position on the page of the top left-hand corner of the graphics field.

Note: If a row or column of zero is specified, the graphics field is deleted.

depth *(specified by user) (fullword integer)*
The depth of the field.

width *(specified by user) (fullword integer)*
The width of the field.

Note: If either the depth or width is zero, the graphics field is deleted.

Description

Overrides the default graphics field on the current page. The graphics field is used to display graphics primitives. The region to be occupied by the graphics field is defined in row/column coordinates.

Note: Only one graphics field is allowed per page.

If the graphics field is redefined, the following occurs:

- The existing graphics contents are lost
- All segments in the original field are deleted
- Any logical input devices enabled for the page are reset to their default state
- Any User Control panning and zooming that has been performed on the picture is reset.

A new graphics field is created in the newly-defined area.

If the graphics field covers the screen, a field attribute byte (displayed as a blank) may (depending on the device) occupy the lower-right-hand corner of the screen. Because this position is not available for use, graphics field positioning and picture construction should be planned accordingly.

If no graphics field was created when required for a requested function, a graphics field is created automatically. This default graphics field covers the entire GDDM page.

For family-4 devices, the row and column units that apply to the GSFLD parameters depend on the device token in use:

- For cell-based AFPDS tokens, alphanumeric rows and columns are used, just as for family-1 and family-2 devices.
- For other family-4 device tokens (which do not specify cell sizes) the row and column units are determined by the FSPCRT call, which divide the available paper area into a grid. If no FSPCRT call is issued, the row and column units default to pixels.

On family-4 devices, if graphics are rastered into image (which depends on the device token used and the setting of the OFFORMAT procopt), the graphics field is rounded down to a multiple of 32 pixels in each direction.

For restrictions on various devices, see “Dual screen size” on page 241.

Principal errors

```
ADM0141 E  GRAPHICS FIELD POSITION n IS INVALID
ADM0144 E  GRAPHICS FIELD OVERLAPS IMAGE FIELD
ADM0164 E  GRAPHICS FIELD SIZE n IS INVALID
```

GSFLSH

Function

To clear the graphics input queue.

GSFLSH	(input-device-type, input-device-id)
APL code	573
GDDM RCP code	X'0C0C0E00' (202116608)

Parameters

input-device-type *(specified by user) (fullword integer)*
The type of the device whose input is to be flushed. If the type is 0, all input is flushed regardless of the device identifier specified.

input-device-id *(specified by user) (fullword integer)*
The identifier of the device whose input is to be flushed. If the identifier is –1, all input of the specified type is flushed.

Description

Flushes any items associated with a specific input device or device type from the graphics input queue. When the graphics input queue is flushed, the default graphics field, picture space, viewport, and window are set if they have not previously been specified or defaulted.

Principal errors

```
ADM3201 E  INPUT DEVICE TYPE n IS INVALID
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
```

GSFLW

Function

To set current fractional line width.

GSFLW	(linewidth-multiplier)
APL code	561
GDDM RCP code	X'0C0C070E' (202114830)

Parameters

linewidth-multiplier *(specified by user) (short floating point)*
Specifies the multiplier to be applied to the standard line width:
0.0 Drawing default
>**0.0** Linewidth multiplier (cannot exceed 100.0).

Description

Sets the current value of the line-width attribute as a floating-point value. The representation of the line width may thus be set to fractional values. The widest line is drawn that does not exceed the requested line width.

The standard line width in pixels for the current device (see table below) is multiplied by the line-width-multiplier, and the result rounded down to an integer value. This value defines, in pixels, the width of the lines subsequently drawn.

If the result is zero, the drawing default line width is used. The initial default line width is the standard line width. The default may be changed by a call to GSDEFS followed by a call to GSFLW or GSLW.

If the result is more than the maximum for the current device, the maximum is used.

If the result is less than the minimum for the current device, the minimum is used.

The attribute remains in effect until it is changed by another GSFLW call, or a GSLW call.

When a segment is created by the GSSEG call, the attribute is set to the default value.

When a segment is closed by the GSSCLS call, the attribute is reset to the value that was in effect when the segment was created.

Note: The line-width attribute does not affect mode-3 characters or vector markers that are drawn using GSCHAP, GSCHAR, GSMARK, or GSMRKS. These characters always use the standard default line width.

See also the GSLW call.

For restrictions on various devices, see “Graphics line types and widths” on page 247.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSGET

Function

To retrieve graphics data.

GSGET	(buffer-length, buffer, GDF-length)
APL code	555
GDDM RCP code	X'0C0C0B02' (202115842)

Parameters

buffer-length *(specified by user) (fullword integer)*
Gives the length of the data buffer supplied. The maximum possible length of a GDF order is 257 bytes (comprising 1 order-code byte, 1 length byte, and up to 255 data bytes), and therefore **buffer-length** must be specified to be at least 257 bytes long.

buffer *(returned by GDDM) (character)*
A data area, of stated length, to receive the GDF data.

GDF-length *(returned by GDDM) (fullword integer)*
A variable that is usually set to the length of GDF data placed in the buffer. If it is zero (and no error is reported), all the GDF data requested has already been returned.

Description

Retrieves graphics data from the current page into the supplied buffer.

The graphics data that is kept as a representation of the picture on the current page is converted to graphics data format (GDF) and placed in the buffer supplied.

For information about GDF, see Chapter 10, “GDF order descriptions” on page 281 and refer to the *GDDM Base Application Programming Guide*.

Retrieval of graphics data must start with a call to GSGETS. This shows the data that is needed. One or more calls to GSGET can then be used to fetch the data.

If the buffer is large enough to contain the GDF data requested, the data is returned and the last parameter is set to show its length. If the buffer is not large enough, as many complete GDF orders as will fit are placed into the buffer. More data can be obtained by another call to GSGET. All data has been extracted when a length of zero is returned (without error).

Each call to GSGET retrieves several *complete* GDF orders; *parts* of orders cannot be returned. (This means that buffers obtained from GSGET can be returned as input to GDDM through GSPUT. If there is not enough room in the buffer to take a complete order, no data is returned, and an error is raised.

The GSGETS call begins the retrieval of graphics data. This shows the data that is needed. One or more calls to GSGET can then be used to fetch the data. Graphics retrieval is ended by GSGETE.

If the amount of data returned does not completely fill the buffer, any unused space is padded with GDF no-operation orders; note that the X'FF' character is not a valid GDF order, and is only included to maintain compatibility with releases of GDDM earlier than Version 2 Release 1. This padding is not included in the value returned in the **GDF-length** parameter. After all the GDF data has been returned, the first padding character is set to X'FF'; see also GSPUT.

Principal errors

```
ADM0173 E  STRING LENGTH n IS INVALID
ADM0178 E  GRAPHICS RETRIEVAL NOT INITIALIZED
```

GSGETE

Function

To end retrieval of graphics data.

GSGETE	
APL Code	556
GDDM RCP code	X'0C0C0B01' (202115841)

Parameters

None.

Description

Ends the retrieval of graphics data (see also GSGET and GSGETS).

The function can be called whether or not all the graphics data has been retrieved.

Principal errors

```
ADM0178 E  GRAPHICS RETRIEVAL NOT INITIALIZED
```

GSGETS

Function

To start retrieval of graphics data.

GSGETS (count, array)	
APL code	554
GDDM RCP code	X'0C0C0B00' (202115840)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in the following array.

array *(specified by user) (an array of fullword integers)*
The data required. The array can have these elements:

1 The identifier of the segment required. If the identifier is zero or if the array has no elements, all segments are retrieved.

2 The format of the GDF information to be returned. Possible values are:

0 or 2 2-byte integer GDF. This is the default.

For restrictions on various devices, see "Device-specific saved pictures" on page 242.

All coordinates within GDF orders are defined by a device-dependent coordinate system (defined by the initial Comment order within the GDF).

4 4-byte short floating-point GDF. All coordinates within GDF orders are defined by the current window coordinate system (extended as necessary to cover the graphics field).

3 The type of GDF information to be returned. Possible values are:

0 or 1 For segments: Initial Comment order and segment GDF. This is the default.

2 For pictures: Initial Comment order, symbol set names, picture prolog, and segment GDF. For information on the format of GDF orders that are returned, see Chapter 10, "GDF order descriptions" on page 281.

Description

Starts the retrieval of graphics data format (GDF), symbol set, window, and coordinate type information from the current page. The GDF information can be returned in either fixed-point or floating-point format, depending on the value of the second element of the **array** parameter.

Retrieval of graphics data cannot be started when there is an open segment.

The graphics data for an individual named segment, or the data for all segments being kept for the page, may be retrieved,

GSGETS specifies the data required and begins the retrieval. Data is obtained by GSGET and the retrieval is ended by GSGETE.

The following calls cannot be issued after GSGETS until a GSGETE call has been issued:

FSCOPY	GSCALL	GSDEFE	GSDEFS
GSGETS	GSLOAD	GSSAGA	GSSATS
GSSAVE	GSSCPY	GSSDEL	GSSEG
GSSINC	GSSORG	GSSPOS	GSSPRI
GSSTFM			

and (if the primary device is a queued printer)

ASREAD FSCHEK FSFRCE.

For full information on GDF and its orders, see the Chapter 10, "GDF order descriptions" on page 281.

Principal errors

```
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0161 E  GRAPHICS FIELD NOT DEFINED
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0180 E  GRAPHICS RETRIEVAL NOT SUPPORTED BY DEVICE
```

GSIDVF

Function

To set initial floating-point data value for a graphics input device.

GSIDVF	**(device-type, device-id, element-number, float-value)**
APL code	571
GDDM RCP code	X'0C0C0C05' (202116101)

Parameters

device-type *(specified by user) (fullword integer)*
The type of device that is to be given an initial value. Possible values are:

2 Locator device
3 Pick device

device-id *(specified by user) (fullword integer)*
This must have a value of 1, which identifies the device. See GSILOC or GSIPIK for further details of the locator or pick device.

element-number *(specified by user) (fullword integer)*
The element of the locator or pick device that is to be given an initial value.

For a **locator** device (**device-type**=**2**), the element number can be:

0 Delete all float values for this device.
1 The initial value is an x coordinate.
2 The initial value is a y coordinate.

For a **pick** device (**device-type**=**3**), the element number can be:

0 Delete all float values for this device.
1 The initial value is the pick aperture.

float-value *(specified by user) (short floating point)*
The initial value to be set for the device element.

For a **locator** device, this provides data needed to define the locator echo:

For a rubber-band echo (echo-type 4), the x or y coordinate of the fixed end of the line.

For a rubber-box echo (echo-type 5), the x or y coordinate of the fixed corner of the box.

Notes:

1. If a coordinate has not been provided on GSIDVF before the device is enabled (see GSENAB), for echo types 4 and 5 the coordinate defaults to the corresponding coordinate of the initial locator position, as set by GSILOC.
2. If only the x coordinate is set, the initial rubber-band or rubber-box is a horizontal line. If only the y coordinate is set, the initial rubber-band or rubber-box is a vertical line.

For a segment echo (echo-type 6), the x or y offset from the locator position at which the segment origin is to be positioned. For the definition of a segment origin, see GSSPOS. If a coordinate has not been provided on GSIDVF before the device is enabled (see GSENAB), the default offset is 0,0.

For segment scaling (echo-type 8), the unit scale sizes on the x and y axes.

Note: If either the x or the y unit scale value is set to zero, no scaling is performed in this direction.

For a **pick** device, the initial value is the size of the pick aperture given as a ratio to the default aperture size. For 3270-family devices, it is the height of the hardware cells.

Description

Used with GSILOC to set initial values for rubber-banded, rubber-boxed, and segment echoes; and with GSIPIK to set the size of the aperture of the pick window. Its use is optional in all cases. GSIDVF can be used only when the specified locator or pick device is disabled.

When a data item is initialized, the default graphics field, picture space, viewport, and window are set if not previously specified or defaulted. When the graphics field is deleted or redefined, all data values for all input devices are deleted.

Principal errors

```
ADM3201 E  INPUT DEVICE TYPE n IS INVALID
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3208 E  INPUT  DEVICE  TYPE  n1  IDENTIFIER  n2  IS
           ALREADY ENABLED
ADM3209 E  ELEMENT NUMBER OR DATA VALUE n IS INVALID
```

GSIDVI

Function

To set initial integer data value for graphics input device.

GSIDVI	**(device-type, device-id, element-number, integer-value)**
APL code	570
GDDM RCP code	X'0C0C0C04' (202116100)

Parameters

device-type *(specified by user) (fullword integer)*
The type of device that is to be given an initial value. Possible values are:

2 Locator device
4 String device

device-id *(specified by user) (fullword integer)*
This must have a value of 1, which identifies the device. See GSILOC or GSISTR for further details of the locator or string device.

element-number *(specified by user) (fullword integer)*
The element of the **device-type** that is to be given an initial value.

For a **locator** device (**device-type=2**), the value can be:

0 Delete the segment identification associated with this locator.
1 The initial value is the segment identification of the segment that is to be used as the echo (echo-type 6).

For a **string** device (**device-type=4**), the value can be:

0 delete (reset to zero) any previous integer values
1 The integer value is the initial cursor position.

integer-value *(specified by user) (fullword integer)*
The initial value to be set for the device element.

For a **locator** device, the identification of the segment that is to be used as the echo for this locator. The segment must have the transformable attribute assigned with a GSSATI call.

For a **string** device, it is the field position under which the cursor must be placed. Note that a zero value is treated the same as a 1. The maximum value must not be greater than the maximum length of a string; see GSISTR.

Description

| Used with GSILOC, to set the segment identifier for locator
| echo-types 6,7,8, and 9 and with GSISTR, to set the field position under which the cursor must be placed. The GSIDVI call can be used only when the specified locator or string is disabled.

When a data element is initialized, the default graphics field, picture space, view-port, and window are set if not previously specified or defaulted. When the graphics field is deleted or redefined, all data values for all input devices are deleted.

Principal errors

```
ADM3201 E  INPUT DEVICE TYPE n IS INVALID
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3208 E  INPUT  DEVICE  TYPE  n1  IDENTIFIER  n2  IS
           ALREADY ENABLED
ADM3209 E  ELEMENT NUMBER OR DATA VALUE n IS INVALID
```

GSILOC

Function

To initialize locator.

GSILOC	**(device-id, echo-type, x-coord, y-coord)**
APL code	568
GDDM RCP code	X'0C0C0C00' (202116096)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the locator to be initialized. The only valid value is:

1 The mouse or tablet, or the cursor keys.

echo-type *(specified by user) (fullword integer)*
The echo is what the operator sees on the screen when using the locator. Possible values are:

| **0** The default echo or system cursor is used to show the locator position. The initial position is described by the
| **x-coord** and **y-coord** parameters. Users of
| GDDM-OS/2 Link can use the Alt-Cur key to toggle
| through the following range of system cursors:

| 1. Black and white target
| 2. Black and white cross
| 3. XOR target
| 4. XOR cross
| 5. XOR full-screen crosshair

1 A null echo.

2 The locator position is shown by a device-dependent echo (typically a cross-hair cursor).

The initial position is described by the **x-coord** and **y-coord** parameters.

3 The locator position is shown by a small tracking cross. The initial position is described by the **x-coord** and **y-coord** parameters.

4 The locator position is shown by a “rubber-band,” which is a line having one end fixed and the other end at the locator position. The initial position of the locator end of the line is given by the **x-coord** and **y-coord** parameters. This is also the initial position of the fixed end of the line. The GSIDVF call must be used to position the fixed end of the line if a different initial position is required.

5 The locator position is shown by a “rubber-box,” which is a rectangle with sides parallel to the x and y axes having one corner fixed and the opposite corner at the locator position. The initial position of the locator corner of the box is given by the **x-coord** and **y-coord** parameters. This is also the initial position of the fixed corner of the box. The GSIDVF call must be used to position the fixed corner of the box if a different initial position is required.

6 The locator position is shown by a transformable graphics segment that is moved round the screen as the workstation locator is moved. Note that a *copy* of the segment is “attached” to the locator. The original segment remains displayed at its current position. After a GSREAD using a locator with echo-type 6 has completed with an interrupt for a locator device, the segment is not repositioned to the locator position. If the application needs to move the segment, it must do so explicitly with a suitable call, for example, GSSPOS.

When a locator is enabled, the copy of the echo segment attached to the locator is always displayed as visible and nonhighlighted, regardless of the current attribute settings for the segment.

The segment attached to the locator may be clipped, according to where it was drawn and whether clipping was requested. To ensure that the completed segment is attached, it should be drawn at a position where the entire segment is visible.

By default, the local origin of the segment is displayed at the locator position as specified by the **x-coord** and **y-coord** parameters. The position of the segment in relation to the locator position can be altered by specifying the relative displacement of the segment origin to the locator position by use of the GSIDVF call, or by changing the segment origin by a GSSORG call.

The GSIDVI call must be used to identify the transformable segment that is to be the echo.

Note: Parts of the segment may be clipped out during User Control pan-and-zoom activities.

7 The locator position is shown by a transformable graphics segment that can be scaled along both axes as the workstation locator is moved. Note that a *copy* of the segment is "attached" to the locator. The original segment remains displayed at its current size. The scaling of the segment is based on the distance between a reference point set using the GSIDVF call. Scaling is performed independently along the x- and y axes.

If the application is to scale the actual segment, it must do so explicitly with a suitable call, such as GSSAGA.

The GSIDVI call must be used to identify the transformable segment that is to be the echo.

8 The locator position is shown by a transformable graphics segment that can be rotated through the angle that the workstation locator makes with a reference point set using the GSIDVF call. Note that a *copy* of the segment is "attached" to the locator. The original segment remains displayed at its current orientation. The rotation of the segment is based on the angle made by the moving locator and a static reference point set using the GSIDVF call.

If the application is to actually rotate the segment, it must do so explicitly with a suitable call, such as GSSAGA.

The GSIDVI call must be used to identify the transformable segment that is to be the echo.

9 The locator position is shown by a transformable graphics segment that can be sheared at the angle that the workstation locator makes with a reference point set using the GSIDVF call. Note that a *copy* of the segment is "attached" to the locator. The original segment remains displayed with its current shape. The shearing of the segment is based on the angle made by the moving locator and a static reference point selected by the end user.

The GSIDVI call must be used to identify the transformable segment that is to be the echo.

x-coord *(specified by user) (short floating point)*
y-coord *(specified by user) (short floating point)*
The x- and y-coordinates specify the initial position of the locator on the screen. The values are specified in world coordinates. The coordinates are mapped to a screen position using the window and viewport currently in effect. Any change to the window or viewport before a GSREAD call is issued does not affect the initial locator position on the screen.

Description

Provides an initial value and echo characteristics for a locator.

A locator cannot be initialized if it is enabled.

When a locator is initialized, the default graphics field, picture space, viewport, and window are set if they have not been specified. The coordinates supplied are converted to device coordinates before being stored. When the graphics field is deleted or redefined, all initial values for the locators are reset to the default.

FOR INFORMATION ABOUT RESTRICTIONS ON VARIOUS DEVICES, SEE "Graphics logical input devices" on page 247.

Principal errors

```
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3203 E  ECHO TYPE n IS UNSUPPORTED
ADM3206 E  LOCATOR COORDINATE f OUTSIDE PICTURE SPACE
ADM3208 E  INPUT  DEVICE  TYPE  n1  IDENTIFIER  n2  IS
           ALREADY ENABLED
```

GSIMG

Function

To draw a graphics image.

GSIMG	**(type, width, depth, length, image-data)**
APL code	552
GDDM RCP code	X'0C0C0A00' (202115584)

Parameters

type *(specified by user) (fullword integer)*
The type of the graphics image to be drawn. It must be cleared to 0.
width *(specified by user) (fullword integer)*
The width of the graphics image in pixels. It must be less than 2040.
depth *(specified by user) (fullword integer)*
The depth of the graphics image in pixels.
length *(specified by user) (fullword integer)*
The length in bytes of the graphics image data, including padding.
image-data *(specified by user) (character)*
The graphics image data to be displayed. The pixels must be given row by row, starting at the top and running from left to right within each row.

Description

Draws a graphics image at the current position. All graphics images handled are expected to be rectangular and to consist of an array of pixels (or display points), each pixel being represented by one bit.

The width and depth specified determine how many pixels there are in the horizontal and vertical directions. The data determines which of the pixels are visible. A bit set to 1 sets the associated pixel on; a bit cleared to 0 leaves the associated pixel unchanged, if the background mix is transparent, or sets it to the background color if the background mix is opaque.

The top left-hand corner of the graphics image is placed at the current position and the data supplied is drawn row by row starting at the top. Each row is drawn from left to right and must be padded out to an integral number of bytes if the image width specified is not a multiple of 8. If, for example, the graphics image width specified is 12, each row of data must be padded out to a length of 16 so that the data in the row occupies 2 bytes exactly. If this is not done, the graphics image is distorted.

The length of graphics image data specified must consider the padding of each row of data. The length must be given in bytes, and an error message is issued if it is wrong.

Because of the different sizes of pixels for different devices, the relationship of the graphics image with respect to other graphics primitives is device dependent.

The color of the graphics image is determined by the current value of the color attribute. The current position remains unchanged after the graphics image has been drawn.

If clipping is enabled (see GSCLP), the graphics image is output only if the current position is within the window. The complete graphics image is output even if some part of the graphics image lies outside the window (regardless of the current clipping state).

For information about restrictions on various devices, see “Graphics image” on page 247.

Principal errors

```
ADM0148 E  IMAGE TYPE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0170 E  IMAGE SIZE n IS INVALID
ADM0171 E  IMAGE DATA LENGTH n IS INVALID
```

GSIMGS

Function

To draw a scaled graphics image.

GSIMGS	**(type, width, depth, length, image-data, x-size, y-size)**
APL code	565
GDDM RCP code	X'0C0C0A04' (202115588)

Parameters

type *(specified by user) (fullword integer)*
The type of the graphics image to be drawn. It must be cleared to 0.

width *(specified by user) (fullword integer)*
The width of the graphics image in display points. It must be less than 2040.

depth *(specified by user) (fullword integer)*
The depth of the graphics image in display points.

length *(specified by user) (fullword integer)*
The length in bytes of the graphics image data, including padding.

image-data *(specified by user) (character)*
The graphics image data to be displayed. The display points must be given row by row, starting at the top and running from left to right within each row.

x-size *(specified by user) (short floating point)*
A number determining the size, in world coordinate units, of the graphics image window in the x direction.

y-size *(specified by user) (short floating point)*
A number determining the size, in world coordinate units, of the graphics image window in the y direction.

Description

Draws a graphics image in the same way as the GSIMG call, but scales the size of the graphics image as well.

The first five parameters are interpreted in the same way as the parameters of the GSIMG call. The **x-size** and **y-size** values are floating-point numbers determining the size of the “image window,” the graphics image being scaled independently in the x direction and y direction to fit within the window.

Only integral scaling-up is performed; each bit in the graphics image is mapped onto a small rectangular area, the width and depth of which is an integral number of pixels. If the window is smaller than the provided graphics image in either dimension, a scale factor of one is used in that dimension.

Principal errors

```
ADM0148 E  IMAGE TYPE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0170 E  IMAGE SIZE n IS INVALID
ADM0171 E  IMAGE DATA LENGTH n IS INVALID
ADM0183 E  IMAGE WINDOW SIZE f IS INVALID
```

GSIPIK

Function

To initialize pick device.

GSIPIK	(device-id, echo-type, segment-id, tag)
APL code	569
GDDM RCP code	X'0C0C0C01' (202116097)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the pick to be initialized:

1 This is the only value that can be specified.

echo-type *(specified by user) (fullword integer)*
The type of feedback the operator is to receive when operating the pick. This value must be zero.

segment-id *(specified by user) (fullword integer)*
tag *(specified by user) (fullword integer)*
The primitive at which the pick should initially be placed. The segment should be visible and detectable. The initial screen position is determined from the segment identifier and tag when the pick device is enabled.

If either the segment identifier or the tag are specified as 0, the initial position of the pick is at the center of the graphics field.

If, when GSENAB is issued, the segment/tag combination does not exist or the segment is invisible or nondetectable, the initial position of the pick is set at the center of the graphics field.

Description

Specifies the initial conditions for a pick device.

A pick cannot be initialized if it is currently enabled.

When a pick is initialized, the default graphics field, picture space, viewport, and window are defined if not previously specified by the application.

When the graphics field is deleted or redefined, all initial values for pick devices are reset to the default.

Note: If the pick is initialized and not changed by the terminal operator, it does not necessarily return the initial value (for example, if a detectable segment of higher priority also exists at the pick initial position).

For information about restrictions on various devices, see “Graphics logical input devices” on page 247.

Principal errors

```
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3203 E  ECHO TYPE n IS UNSUPPORTED
ADM3208 E  INPUT  DEVICE  TYPE  n1  IDENTIFIER  n2  IS
           ALREADY ENABLED
```

GSISTK

Function

To initialize stroke device.

GSISTK	(device-id, echo-type, sampling-method, x, y, count)
APL code	595
GDDM RCP code	X'0C0C0C07' (202116103)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the stroke device to be initialized:

1 This is the only value that can be specified; only one stroke device is available.

echo-type *(specified by user) (fullword integer)*
The type of feedback the operator is to receive when operating a stroke device. Possible values are:

0 Default (polyline).
1 Polyline.
The echo is a series of lines joining the **x,y** positions.
2 Polymarker.
The echo is a series of markers at the **x,y** positions. The default marker (a cross) is used.

sampling-method *(specified by user) (fullword integer)*
The method used to record **x,y** pairs. Possible values are:

0 Default (polylocator).
1 Polylocator.
When a button on the mouse or tablet is pressed, **x,y** values are generated.
2 Stream.
When a mouse or tablet button is pressed, the **x,y** positions are generated automatically to reflect the position of the locator device, and are echoed by lines joining the generated **x,y** positions. This traces an outline of the path taken by the locator device.

Note: Echo-type 2 is not supported for sampling-method 2.

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The starting position that is returned as the initial position when, for example, a mouse is defined as the stroke

device. The position is ignored when a tablet with a stylus is defined as the stroke device.

count *(specified by user) (fullword integer)*
The maximum number of pointings that are accepted. The maximum number that can be specified is 1024. The default is 64.

Description

Provides initial values for the stroke device.

A stroke device lets the operator enter a sequence of **x,y** positions, or "pointings," by moving a locator device (a mouse, tablet four-button cursor (puck), or tablet stylus) and using the buttons on this device.

The pairs of **x,y** values are generated either one at a time in response to the buttons (polylocator-sampling method), or as a continuous stream of values giving the trajectory of the moving locator device (stream-sampling method), started, restarted, or suspended by the use of the mouse or tablet buttons.

A stroke device cannot be initialized if it is currently enabled. When a stroke device is initialized, the default graphics field, picture space, view port, and window are set if they have not already been defined.

Note: CICS/VS does not allow a message to be read whose length exceeds the DFHTCT TYPE=LINE INAREAL parameter value by more than 4000 bytes (for BTAM-connected terminals), or whose length exceeds the DFHTCT TYPE=TERMINAL TIOAL ***value2*** (for VTAM-connected terminals). However, the stroke device contribution to the input in an inbound message is of length

`(6 * count)` bytes, using the stream-sampling method, and
`(9 * count)` bytes, using the polylocator-sampling method,

plus a fixed overhead of 22 bytes, and the stroke contribution may cause the CICS/VS restrictions to be exceeded. Therefore, it might be necessary to increase the DFHTCT values or restrict the **count** value to enable successful operation of the stroke device. Note also that the inbound message contains alphanumeric input and input from other logical-input devices.

For information about restrictions on various devices, see "Graphics logical input devices" on page 247.

Principal errors

```
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3203 E  ECHO TYPE n IS UNSUPPORTED
ADM3208 E  INPUT DEVICE TYPE n1 IDENTIFIER n2 IS
           ALREADY ENABLED
ADM3219 E  STROKE DEVICE INITIAL POSITION f OUTSIDE
           PICTURE SPACE
ADM3220 E  MAXIMUM NUMBER OF POINTINGS n IS INVALID
```

GSISTR

Function

To initialize string device.

GSISTR	**(device-id, echo-type, x, y, count, string)**
APL code	594
GDDM RCP code	X'0C0C0C06' (202116102)

Parameters

device-id *(specified by user) (fullword integer)*
The identifier of the string device to be initialized.
1 This is the only value that can be specified.
echo-type *(specified by user) (fullword integer)*
The type of string device. Possible values are:
1 Normal echo of characters (the same as mode-1 character strings). This is the default.
2 No echo (the character string is not displayed).
x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The world coordinates of the start of the string.
count *(specified by user) (fullword integer)*
The number of character positions in the string.
string *(specified by user) (character)*
The initial value of the character string. The length of the string must equal the value in **count**.

Description

Provides an initial value and echo characteristics for the string device.

The string device is defined as **device-id**=4 in the GSENAB call. It is a string of characters that can be initialized by the application program and typed into by the operator. A string device is echoed by displaying the characters in the string as the operator types them.

The string data is put into the input queue when the operator presses ENTER or a PF key. The string data is returned to the application program by issuing a GSREAD call, followed by a GSQSTR call.

A string device cannot be initialized if it is currently enabled (by a GSENAB call). When a string device is initialized, the default graphics field, picture space, viewport, and window are set if they have not already been defined.

When the string device is enabled, the operator can type character data starting at the first position in the string. When no more character positions remain, any further input is discarded. The string can be altered by using the backspace key or the move cursor left and right keys and typing

in new characters. A cursor (an underline) shows the current position (where the operator is entering or changing characters) in the string.

The string echo is treated as a mode-1 character string for positioning and clipping. Clipping does not alter the contents of the data in the string device.

The maximum length of a string is device dependent. The length of the default string device is 8 characters, initialized to blanks.

For information about restrictions on various devices, see "Graphics logical input devices" on page 247.

Principal errors

```
ADM3202 E  INPUT DEVICE IDENTIFIER n IS INVALID
ADM3203 E  ECHO TYPE n IS UNSUPPORTED
ADM3208 E  INPUT DEVICE TYPE n1 IDENTIFIER n2 IS
           ALREADY ENABLED
ADM3217 E  STRING DEVICE INITIAL POSITION f OUTSIDE
           PICTURE SPACE
ADM3218 E  INITIAL STRING LENGTH n IS INVALID
```

GSLINE

Function

To draw a straight line.

GSLINE	**(x, y)**
APL code	526
GDDM RCP code	X'0C0C0401' (202114049)

Parameters

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The end-point of the line in world coordinates.

Description

Draws a straight line from the current position to the specified end point.

The line has the color, line width, and line type given by the current values of these attributes. The current position is set to the end point of the line.

If the specified end point lies outside the window boundaries and clipping is not enabled, the results are undefined. If clipping is enabled, only the section of the line within the current window is visible.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
```

GSLOAD

Function

To load segments.

GSLOAD	**(name, count1, opt-array, seg-count, count2, descriptor)**
APL code	593
GDDM RCP code	X'0C0C1201' (202117633)

Parameters

name *(specified by user) (8-byte character string)*
The name (left-justified) of the GDF file from which segments or the GDF object is to be loaded. This must be a valid external object name for the subsystem being used.

count1 *(specified by user) (fullword integer)*
The number of elements specified in the **opt-array** parameter.

opt-array *(specified by user) (an array of fullword integers)*
Specifies how GSLOAD is to restore a picture when it is copied from the segment library. The parameter has seven elements; the options are:

1–seg-base
The starting segment identifier for loading segments into the GDDM page.

2–load-type
Specifies how the segment is to be restored from the segment library

3–draw-defaults
Specifies the action to be taken when loading a GDF object that contains drawing default specifications.

4–resolve
Specifies the action to be taken when loading a GDF object that contains call segment orders to segments that do not exist in the object

5–symbol-set
Specifies the action to be taken when loading a GDF object that references a symbol set.

6–seg-zero
Specifies how segments from the GDF object with an identifier of 0 are renumbered.

7–primitive tag value
Specifies a tag value to add to all untagged primitives to allow them to be picked with a locator device.

These elements are described in detail in the next section.

seg-count *(returned by GDDM) (fullword integer)*
When **seg-base** = **0**, **seg-count** is always 0. When **seg-base** > **0**, **seg-count** is the number of named segments created by the GSLOAD call. This may include previously unnamed segments and extra segments (created by GSLOAD to resolve call segment orders) depending upon the other parameters to GSLOAD.

count2 *(specified by user) (fullword integer)*
The length supplied for the **descriptor** parameter.

descriptor *(returned by GDDM) (character)*
The descriptive record, of up to 253 bytes, that is saved with the picture.

The elements of opt-array

- **seg-base** *(first element)*

Specifies whether segments being loaded are to be renumbered. The options are:

0 Segments loaded from auxiliary storage are not renumbered, and therefore retain their original segment identifiers (the default).

>**0** Segments loaded from auxiliary storage are renumbered with identifiers in the range **seg-base** through (**seg-base** + **seg-count** – 1). Any call segment order that references a segment within the GDF object has the segment identifier changed to the new identifier given to that segment.

- **load-type** *(second element)*

Specifies how the picture is to be restored from the segment library. The options are:

0 The default; same as 1.

1 The segment or picture primitives are restored without transformation, using the page's current window and viewport coordinate system. This is the default action. Note that if the picture being restored was saved using 2-byte integer coordinates (see GSSAVE), the picture data is defined in a device-dependent coordinate system. (There is no relationship between the application program's window coordinate system and the device-dependent system.) To restore the saved data satisfactorily, a window coordinate system that corresponds to the device-dependent system for the saved data must be defined using a GSWIN call or a GSUWIN call.

2 The picture space of the GDF object is accommodated within the current viewport, preserving the aspect ratio that the picture had when it was saved. Any primitives outside the picture space of the GDF object may be lost.

3 The bottom left-hand corner of the picture space of the GDF object is placed at the origin (0,0) of the world-coordinate system, preserving the size of the picture when it was created.

4 If, at the time of the GSLOAD, the picture space and the rest of the graphics hierarchy have not been defined, the shape of the picture held in the file to be loaded is used to define the picture space, and the rest of the graphics hierarchy is defaulted. The load then proceeds as for **load-type = 2**.

If the picture space has been defined (or defaulted), this load-type is equivalent to **load-type = 2**.

Note: When GDF is saved, the whole picture space is saved, not just the current viewport. The aspect ratio of the GDF cannot be determined before it is loaded.

- **draw-defaults** *(third element)*

Specifies the action to be taken when loading a GDF object that contains drawing default specifications. The options are:

0 The default; same as 5.

1 Any drawing default definitions within the saved data are ignored.

2 Drawing default definitions within the saved data are appended to the current drawing defaults. A drawing default value within the saved data is used to set the current drawing default value for that attribute, providing that it has not been previously set.

This may affect existing primitives.

3 Drawing default definitions within the saved data are used to over-ride the current drawing defaults. A drawing default value within the saved data is used to set the current drawing default for that attribute. The values for drawing defaults not held in the saved data remain unchanged.

This may affect existing primitives.

4 Drawing default values within the saved data are used to totally replace the current drawing defaults. All current values for drawing defaults not held in the data are discarded, and the value reset to the standard default.

This may affect existing primitives.

5 The drawing defaults within the saved data are incorporated into the segment data to be loaded. The current drawing defaults are not modified. Thus the loaded data reflects the drawing defaults at the time the data was saved, but the GSLOAD does not affect any data currently displayed. This is the default action.

Note: Segments that are both called and chained, do not inherit attribute values from the caller, for those attributes with default values defined in the data.

- **resolve** *(fourth element)*

Specifies the action to be taken when loading a GDF object that contains call segment orders to segments which do not exist in the object. The options are:

0 Default; same as 1.

1 All call segment orders within the GDF object that cannot be resolved are ignored, and a warning message issued.

2 All call segment orders within the GDF object that cannot be resolved remain unchanged.

- **symbol-set** *(fifth element)*

Specifies the action to be taken when loading a GDF object that references symbol sets. The options are:

0 Default; same as 1.

1 GDDM loads the symbol sets that were loaded at the time the segment was saved. It also loads them with the same identifiers, regardless of any symbol sets that might already have been loaded.

2 GDDM loads each symbol set that was loaded at the time the segment was saved, but only if a symbol set of the same name is not already loaded. The symbol sets to be loaded are allocated new identifiers. The largest unused identifiers are used for this purpose. If there are not enough unused identifiers for all the symbol sets to be loaded, the default symbol sets are used instead. GDDM allows only one loaded pattern set and one loaded marker set at a time on internal storage. Additional pattern sets or marker sets are not loaded.

• **seg-zero** *(sixth element)*
Specifies how segments from the segment library with identifier 0 are renumbered.

0 The default; same as 1
1 Do not renumber segments with identifier 0.
2 Renumber segments with identifier 0 according to the **seg-base** element.

| • **primitive tag value** *(seventh element)*
| Specifies a tag value to add to all untagged primitives to
| allow them to be picked with a locator device.

| **0** The default; do not change primitive tags.
| >**0** Any other fullword integer value; tag all primitives which
| currently have a zero tag value with this new value to
| make them detectable and pickable.

Description

Retrieves a complete copy of a graphics data format (GDF) object from the segment library on auxiliary storage and loads it into the current GDDM page.

GSSAVE can be used to save GDF objects.

The segments in the GDF object can be loaded with a set of segment identifiers, which can either be the same, or different, from the ones with which they were saved on the segment library.

A segment must not be open when the GSLOAD call is made; GSLOAD does not leave any open segment.

Segments retain their segment attributes when they are loaded.

GSLOAD loads the GDF object into the current window and viewport. The default graphics field, picture space, viewport, and window are set if they have not been specified.

GDDM loads the symbol sets that the segment used at the time it was saved, and all others that were loaded at the time, whether they were used by the segment or not. The default is to load these symbol sets with their original identifiers even if any loaded symbol sets possess the same identifier. The application can specify, by using **symbol-set** option 2, that symbol sets are to be loaded with distinct identifiers to avoid the over-writing of loaded symbol sets.

It is not possible to determine what symbol sets a GDF uses before it is loaded.

If the symbol sets cannot be loaded, GDDM creates the picture using default symbol sets. If GDDM detects other types of errors, the picture reverts to its original state, without any changes applied.

The application may specify what action GDDM is to take concerning any drawing default definitions contained within the GDF object. (For a description of the drawing defaults definition, see GSDEFS). The default action is to incorporate the values given in the defaults definition into the segment data (option 5). This is done by including, at the start of each chained segment, orders to set those attributes for which default data exists in the file, to the specified default value.

A consequence of this process, is that a segment which is both called and chained does not inherit an attribute value from its caller if a drawing default for that attribute was specified in the file. If the data to be loaded was saved in fixed-point format, the appearance of the loaded data is device-dependent, and subsequent changes to the drawing defaults may not be reflected.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0123 E  SYMBOL SET n1 HAS INVALID FORMAT. REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0128 W  SYMBOL SET n OPTION UNSUPPORTED
ADM0135 E  SYMBOL SET n TYPE UNSUPPORTED
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0143 E  SEGMENT IDENTIFIER n IS DUPLICATE
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0153 E  CONTROL VALUE n IS INVALID
ADM0173 E  STRING LENGTH n IS INVALID
ADM0174 E  INVALID OR UNSUPPORTED GDF ORDER X'xx'
ADM0175 E  INVALID OR UNSUPPORTED LENGTH IN GDF ORDER
           X'xx' OFFSET X'xxxxxxxx'
ADM0182 W  INVALID CHARACTER CODE X'xx' IN STRING
ADM0307 E  FILE 'a' NOT FOUND
ADM0313 E  FILE 'a' HAS INVALID RECORD CONTENT
ADM3157 E  SYMBOL SET IDENTIFIER n ALREADY IN USE
ADM3158 E  NO MATCH IN FONT FOR CODE PAGE INDEX ENTRY
ADM3265 W  CALLED SEGMENT n NOT FOUND
ADM3270 W  MORE SYMBOL SETS THAN SYMBOL-SET IDENTIFIERS
ADM3275 E  INVALID ELEMENT n1 VALUE n2 IN GSLOAD OPTION
           ARRAY
ADM3276 W  {PATTERN|MARKER} SET a1 ALREADY LOADED, a2
           CANNOT BE LOADED
```

GSLSS

Function

To load a graphics symbol set from auxiliary storage.

GSLSS	**(type, symbol-set-name, symbol-set-id)**
APL code	202
GDDM RCP code	X'0C040300' (201589504)

Parameters

type *(specified by user) (fullword integer)*
The type and usage of this symbol set. Possible values are:

- **1** SBCS ISS to be used by GDDM for dot-matrix graphics text.
- **2** SBCS VSS to be used by GDDM for generation of vector graphics text.
- **3** Shading pattern set (ISS only).
- **4** Marker symbol set (ISS or VSS).
- **5** 4250 page printer font. The symbol-set identifier must be different from the identifiers of all loaded type-1 image symbol sets. The symbol set names supplied by IBM are in the form AFTmmnnn. The *DCF SCRIPT/VS Language Reference* manual describes a font library index program that can be used to list all 4250-printer fonts that are held on a particular disk.
- **8** DBCS ISS to be used by GDDM for dot-matrix graphics text.
- **9** DBCS VSS to be used by GDDM for generation of vector graphics text.

symbol-set-name *(specified by user) (8-byte character string)*
The name (left-justified) of the symbol set to be read from auxiliary storage. Symbol sets for various devices can be constructed by the Image Symbol Editor, or the GDDM-PGF Vector Symbol Editor.

If an SBCS symbol-set name ends with a period character, the "." is replaced by another character, depending on the device family and the default graphics cell size, or the pixel resolution, of the current device. To find the device's graphics cell size, use the FSQURY call. For information on the character that replaces the period, see the symbol set naming convention described in Chapter 8, "Symbol set formats" on page 275.

A DBCS symbol set name must consist of 1 through 6 non-blank characters. The last character may be a period character, ".". If present, the period substitution character is replaced by a cell size character in the same way as for SBCS symbol set names. To obtain the names of the GDDM symbol set objects to be loaded, the DBCS ward digits, which are the first two digits of the DBCS character, are appended to the name.

Note: If a symbol set with the same name is loaded more than once, without first being released (see GSRSS), it is not defined whether GDDM uses the copy that it already has, or takes a new copy (even if there is an intervening ESLIB call. This could cause different results if GDDM symbol sets have been modified during a session in which they are used.

symbol-set-id *(specified by user) (fullword integer)*
The identifier by which this symbol set is referred to in later statements. Possible values are:

0	Pattern or marker symbol set
65 through 223	Other symbol sets.

Each loaded symbol set should have a unique identifier with respect to all other symbol sets loaded by GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC calls. This avoids any uncertainty that might arise from a device treating different types of symbol sets as equal candidates for displaying a character string. If, however, a symbol-set identifier is the same as one that has previously been issued for the same type, the new definitions replace the previous ones.

For devices that support image shading, the pattern is padded or truncated to the graphics cell size (see the information for **code=2** under FSQURY) and this graphics cell size pattern is repeated in successive cells.

Note: When using segments, remember that the symbol set belongs to the device and not the segments. Therefore, it is advisable to load symbol sets outside of segments; if a symbol set is loaded within a segment, and that symbol set has already been loaded in a previous segment using the same symbol-set identifier, unexpected output may occur when printing the page.

Description

Loads a set of symbol definitions from auxiliary storage. The symbol set can be either an image symbol set (ISS), a vector symbol set (VSS), or a 4250 page printer font. For a plotter, vector symbol sets may be loaded; image symbol sets can be loaded for characters and markers, but not for shading patterns.

The definitions are retained by GDDM for use by graphics.

For DBCS symbol sets, no GDDM symbol set object will be loaded until a ward digit is known, either by a GSCHAR call or by a GSQSSD call.

Note: When using the 4250 printer, and the National Language is going to be changed (by means of the GSCPG call), the GSLSS call must be made after the change.

For information about restrictions on various devices, see "Graphics area shading" on page 247.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0123 E  SYMBOL SET n1 HAS INVALID FORMAT. REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0127 E  SYMBOL SET NAME 'a' IS INVALID
ADM0128 W  SYMBOL SET n OPTION UNSUPPORTED
ADM0135 E  SYMBOL SET n TYPE UNSUPPORTED
ADM0307 E  FILE 'a' NOT FOUND
ADM0313 E  FILE 'a' HAS INVALID RECORD CONTENT
ADM3157 E  SYMBOL SET IDENTIFIER n ALREADY IN USE
ADM3158 E  NO MATCH IN FONT FOR CODE PAGE INDEX ENTRY
ADM3178 W  PATTERNS CANNOT BE SENT TO DEVICE.   AREA
           SHADING MAY BE INCORRECT
```

GSLT

Function

To set current line type.

GSLT	**(n)**
APL code	516
GDDM RCP code	X'0C0C0703' (202114819)

Parameters

n *(specified by user) (fullword integer)*
The line type. Possible values are:

0 - The drawing default line type
1 - Dotted line
2 - Short-dashed line
3 - Dash-dot line
4 - Double-dotted line
5 - Long-dashed line
6 - Dash-double-dot line
7 - Solid line
8 - Invisible line

For plotters, **n** is interpreted as a hardware line type:

0 - The drawing default line type
1 - Line type 2 (on 1, off 1)
2 - Line type 4 (on 3, off 1)
3 - Line type 6 (on 3, off 1, on 1, off 1)
4 - Line type 9 (on 2, off 2)
5 - Line type 11 (on 6, off 2)
6 - Line type 13 (on 6, off 2, on 2, off 2)
7 - Line type 1 (solid line)
8 - Invisible line

Note: Plotters do not support GDDM line-type 4 (hardware line-type 9); GDDM therefore uses a dotted line with wider spacing.

Description

Sets the current value of the line-type attribute. Subsequent primitives using lines (that is, those constructed by GSLINE, GSARC, GSPLNE, GSELPS, GSPFLT, or GSVECM), have the specified line type until it is changed by another GSLT call.

When a segment is created by the GSSEG call, the line-type attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the line-type attribute is reset to the value that was in effect when the segment was created.

Note: The line-type attribute does not affect mode-3 characters or vector markers that are drawn using GSCHAP, GSCHAR, GSMARK, or GSMRKS. These characters always use the standard drawing default line type.

For restrictions on various devices, see “Graphics line types and widths” on page 247.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSLW

Function

To set current line width.

GSLW	**(linewidth-multiplier)**
APL code	517
GDDM RCP code	X'0C0C0704' (202114820)

Parameters

linewidth-multiplier *(specified by user) (fullword integer)*
Specifies the multiplier to be applied to the standard line width:
0 Drawing default
>**0** Linewidth multiplier (cannot exceed 100)

Description

Sets the current value of the line-width attribute. The standard line width in pixels for the current device (see table below) is multiplied by the line-width multiplier. This value defines, in pixels, the width of the lines subsequently drawn.

If the result is zero, the drawing default line width is used. The initial default line width is the standard line width. The default may be changed by a call to GSDEFS followed by a call to GSFLW or GSLW.

If the result is more than the maximum for the current device, the maximum is used.

If the result is less than the minimum for the current device the minimum is used.

Subsequent primitives using lines have the specified width until it is changed by another GSLW call, or a GSFLW call.

When a segment is created by the GSSEG call, the line-width attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the line-width attribute is reset to the value that was in effect when the segment was created.

Note: The line-width attribute does not affect mode-3 characters or vector markers that are drawn using GSCHAP, GSCHAR, GSMARK, or GSMRKS. These characters always use the standard drawing default line width. See also the GSFLW call.

For restrictions on various devices, see “Graphics line types and widths” on page 247.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
```

GSMARK

Function

To draw a marker symbol.

GSMARK (x, y)	
APL code	527
GDDM RCP code	X'0C0C0406' (202114054)

Parameters

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
The position of the marker in world coordinates.

Description

Draws a single marker at a specified position.

A marker is a symbol used to show a position. It is very like a character drawn in character-mode 2, except that it is positioned by its center.

The marker color is determined by the current value of the color attribute set by GSCOL, and the marker is determined by the current symbol specified by GSMS. The default marker is device-dependent.

The current position is set to **(x, y)**.

The effect of transforming vector markers is device-dependent.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSMB

Function

To set marker-box size.

GSMB (width, depth)	
APL code	636
GDDM RCP code	X'0C0C1307' (202117895)

Parameters

width *(specified by user) (short floating point)*
The width of the marker box in world coordinates.

depth *(specified by user) (short floating point)*
The depth of the marker box in world coordinates.

Description

Sets the marker-box size for subsequent vector markers.

The vector marker is scaled to fill the marker box. The specified marker box overrules any marker scale that has been set by a previous GSMSC call.

The marker-box attribute remains in effect until it is changed by another GSMB call or a GSMSC call.

When a segment is created by the GSSEG call, the marker-box attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the marker-box attribute is reset to the value that was in effect when the segment was created.

Principal errors

```
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM3230 E  MARKER BOX SIZE f IS INVALID
```

GSMIX

Function

To set current foreground color-mixing mode.

GSMIX	**(n)**
APL code	518
GDDM RCP code	X'0C0C0702' (202114818)

Parameters

n *(specified by user) (fullword integer)*
Defines the color-mixing mode. Possible values are:

0 The drawing default.

1 "Mix" mode.
Display points common to two primitives (for example, a line crossing a color-shaded area) assume the color resulting from the mixture of the two colors.Table 1 shows the results of mixing colors on a display device.

Table 1. Example color "mix" mode table (GSMIX)

Primitive color	**Underlying color**						
	B	**R**	**M**	**G**	**C**	**Y**	**N**
Blue (B)	B	M	M	C	C	N	N
Red (R)	M	R	M	Y	N	Y	N
Magenta (M)	M	M	M	N	N	N	N
Green (G)	C	Y	N	G	C	Y	N
Cyan (C)	C	N	N	C	C	N	N
Yellow (Y)	N	Y	N	Y	N	Y	N
Neutral (N)	N	N	N	N	N	N	N

2 "Overpaint" mode.
The color of the current primitive takes precedence. Any underlying color is obscured.

3 "Underpaint" mode.
Display points common to two primitives retain the color specified for the first one drawn, unless that color was "background" (color 8). For example, if an area is shaded blue and then a yellow line is drawn across it, the line is visible only where it lies outside the blue area. However, if the shaded area is specified as the background color (8), the line is yellow throughout its length.

4 "Exclusive-OR" (XOR) mode.
The color attribute of the current primitive is "exclusively-ORed" with the color attributes of the underlying primitives.

This mode is provided for compatibility with picture interchange format (PIF) files. It can also be used by specific, high-function, interactive graphic applications, that require high performance update of generated pictures on a restricted range of devices. *However, there are restrictions in the support of this mode and is not recommended for general applications*.

Table 2 on page 137 shows the results of applying exclusive-OR mode on a display device.

When this mode is active, primitives are exclusively-ORed into the current picture. If the same primitive is subsequently drawn in XOR mode, it will be removed from the picture; the effect is as if it had never been drawn. This gives an application program a way of adding data to a picture, and subsequently removing it, without causing a redraw of the picture. (GDDM normally redraws some or all of the picture when any part is deleted, as this is the only way that it can be sure that the resulting picture is accurate.)

The main restrictions of XOR mode are:

- Use in conjunction with: overlapping partitions, windows, and draft draw mode is not supported, and may give unexpected results.

- If an update of the output device occurs (ASREAD, GSREAD, or FSFRCE is issued) while XOR mode is current in an open segment, the resulting picture may be incorrect. Graphics primitives may "disappear" from the picture as a result of being redrawn in XOR mode.
- It should not be used outside a segment, because screen redraws will give unpredictable results.
- Family-4 output is not fully supported. Some primitives may be incorrectly drawn. In particular, areas with boundaries are not drawn correctly.

Note: This mode must only be used for specific applications that can accept the function as it actually operates within the current release of GDDM. The effects of using this mode may vary from one device to another.

Table 2. Results of exclusive-OR mode (GSMIX)

Primitive color	**Underlying color**							
	B	**R**	**M**	**G**	**C**	**Y**	**N**	¤
Blue (B)	¤	M	R	C	G	N	Y	B
Red (R)	M	¤	B	Y	N	G	C	R
Magenta (M)	R	B	¤	N	Y	C	G	M
Green (G)	C	Y	N	¤	B	R	M	G
Cyan (C)	G	N	Y	B	¤	M	R	C
Yellow (Y)	N	G	C	R	M	¤	B	Y
Neutral (N)	Y	C	G	M	R	B	¤	N
Backgrnd(¤)	B	R	M	G	C	Y	N	¤

5 "Leave alone" or "Transparent" mode.
Primitives drawn in this mode are transparent and therefore, do not appear.

Description

Controls the way that the foreground color of a primitive is combined with the color of any underlying primitive.

For display devices, the single-letter abbreviations in Table 1 on page 136 show the color that results from mixing the current primitive color with any underlying color. For example, if a red primitive overlaps a green one, the overlap region appears as yellow; if a yellow primitive overlaps a red one, the overlap region is also yellow. Note that mixing is symmetrical; the resulting color is independent of the order in which the colors appear.

Not all combinations of foreground and background color-mix modes are allowed on all devices. For information on the combinations of foreground and background color-mix modes allowed, see GSBMIX.

The color-mix attribute remains in effect until it is changed by another GSMIX call.

When a segment is created by the GSSEG call, the color-mix attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the color-mix attribute is reset to the value that was in effect when the segment was created.

For information on color mixing within a shaded area, see GSAREA.

For information about restrictions on various devices, see "Graphics area shading" on page 247.

Principal errors

```
ADM0187 W  MIX MODE 'a' IS NOT SUPPORTED ON CURRENT
           DEVICE
```

GSMOVE

Function

To move without drawing.

GSMOVE (x, y)

APL code	528
GDDM RCP code	X'0C0C0400' (202114048)

Parameters

x *(specified by user) (short floating point)*
y *(specified by user) (short floating point)*
A point, in world coordinates, to which the current position is to be moved. The new value of the current position is (**x,y**).

Description

Moves the current position to the specified point.

Principal errors

```
ADM0154 E  COORDINATE f IS INVALID
```

GSMRKS

Function

To draw a series of marker symbols.

GSMRKS (count, xarray, yarray)	
APL code	529
GDDM RCP code	X'0C0C0407' (202114055)

Parameters

count *(specified by user) (fullword integer)*
The number of markers to be drawn.

xarray *(specified by user) (array of short floating-point numbers)*
An array of length **count** that specifies the x coordinates of the marker positions.

yarray *(specified by user) (array of short floating-point numbers)*
An array of length **count** that specifies the y coordinates of the marker positions.

Description

Draws a series of marker symbols at specified points. The marker color is determined by the current value of the color attribute set by GSCOL. The current position is set to the position of the last marker in the series. The marker symbol used is that specified by GSMS.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0154 E  COORDINATE f IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSMS

Function

To set the current type of marker symbol.

GSMS (n)	
APL code	519
GDDM RCP code	X'0C0C070B' (202114827)

Parameters

n *(specified by user) (fullword integer)*
The marker symbol number. Possible values are:

0 The drawing default

System-defined markers:

0 The drawing default

System-defined markers:

1 ×
2 +
3 ◇
4 □
5 ✳
6 ✳
7 ◆
8 ■
9 •
10 ◯

User-defined markers:

65 through 254

Description

Selects the current marker symbol to be used by calls to GSMARK and GSMRKS. System-defined markers are shown above. User-defined markers can be created with the Image Symbol Editor or the Vector Symbol Editor, and loaded using GSDSS or GSLSS.

The marker-symbol attribute remains in effect until it is changed by another GSMS call.

When a segment is created by the GSSEG call, the marker-symbol attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the marker-symbol attribute is reset to the value that was in effect when the segment was created.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSMSC

Function

To set marker scale.

Note: This call is not recommended for new programs. It is obsolete and has been superseded by GSMB.

GSMSC (scale)	
APL code	563
GDDM RCP code	X'0C0C071D' (202114845)

Parameters

scale *(specified by user) (short floating point)*
Specifies the scaling of the marker symbol with respect to the default marker box.

Description

Sets the scale for subsequent marker symbols if the marker symbol is from a vector set. It is not possible to change the scale of image marker symbols. The parameter determines the scale of the marker symbol with respect to the default marker box. If the marker scale is not specified before outputting the first vector marker, a scale of 1 is assumed.

The marker-scale attribute remains in effect until it is changed by another GSMSC call or a GSMB call.

When a segment is created by the GSSEG call, the marker-scale attribute is set to the default value.

When a segment is closed by the GSSCLS call, the marker-scale attribute is reset to the value that was in effect when the segment was created.

The standard default marker box has the same width as the standard default character box. The height of the marker box is such that the aspect ratio of the marker box is the same as that of the vector symbol set from which the marker is selected.

The size of the marker box set in the GSMSC call overrides any previous marker box that may have been set in the GSMB call.

Principal errors

```
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0168 E  MARKER SCALE f IS INVALID
ADM3260 E  INVALID  FUNCTION  DURING  DRAWING  DEFAULTS
           DEFINITION
```

GSPAT

Function

To set current shading pattern.

GSPAT	**(n)**
APL code	520
GDDM RCP code	X'0C0C070A' (202114826)

Parameters

n *(specified by user) (fullword integer)*
The number of the shading pattern to be used. The actual pattern that appears depends on the pattern set that is loaded. Possible values are:

0	The drawing default.
1 through 16	GDDM-defined patterns (see Figure 9 on page 141).
65 through 254	User-defined patterns; these can include the 64 sample geometric patterns (Figure 8 on page 140), or the 64 color shades shown in the *GDDM Base Application Programming Guide*.

Description

This call sets the current value of the pattern attribute. The pattern attribute remains in effect until it is changed by another GSPAT call. All areas use the specified pattern until it is changed by another call to GSPAT. User-defined patterns created with the Image Symbol Editor can be used, if loaded by GSDSS or GSLSS. The sample geometric patterns or color shades supplied with GDDM can also be loaded in this way.

When a segment is created by the GSSEG call, the pattern attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the pattern attribute is reset to the value that was in effect when the segment was created.

Note: Shaded vector characters or markers that are drawn using GSCHAP, GSCHAR, GSMARK, or GSMRKS calls always use the drawing default shading pattern.

Note: Black areas in this figure represent foreground color, and white areas represent background color. Consequently, if you use a foreground color of white and a background of black, the result will appear to be the reverse of that shown here.

For information about restrictions on various devices, see “Graphics area shading” on page 247.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

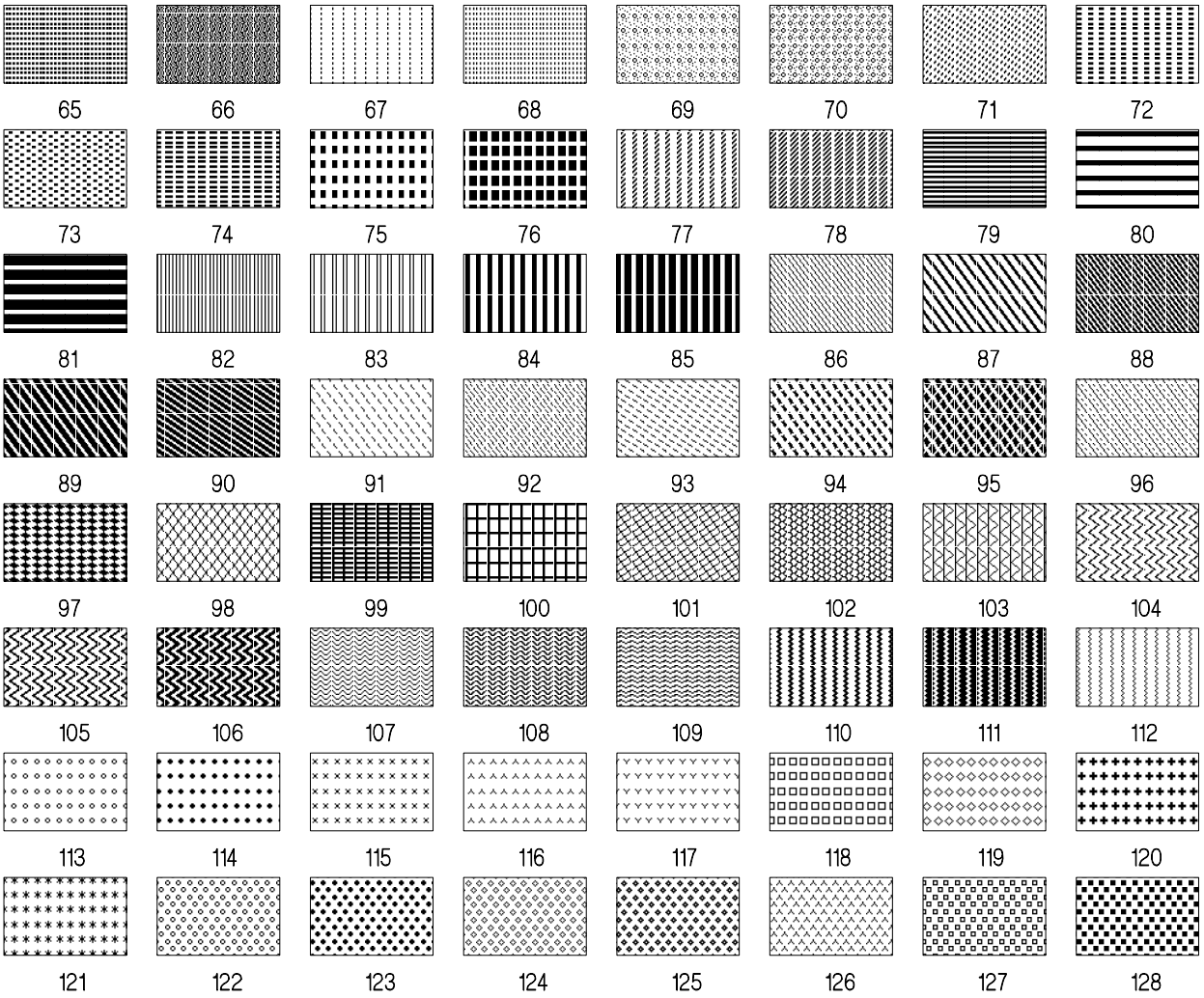


Figure 8. Sample geometric shading patterns (GSPAT)

GSPFLT

Function

To draw a curved fillet.

GSPFLT	**(count, xarray, yarray)**
APL code	557
GDDM RCP code	X'0C0C0602' (202114562)

Parameters

count *(specified by user) (fullword integer)*
The number of points provided in **xarray** and **yarray**.

xarray *(specified by user) (array of short floating-point numbers)*

yarray *(specified by user) (array of short floating-point numbers)*
Two arrays containing the points defining the curve.

Description

Draws a curve starting at current position and defined by the vector of points supplied.

If two points are supplied, an imaginary line is drawn from the current position to the first point and a second line from the first point to the second; see Figure 10 on page 141. A curve is then constructed, starting at the current position and in the direction of the first line. The curve is drawn such that it reaches the last point at a tangent to the second line, having followed a path somewhat like a “curve of pursuit.”

The curve, together with the imaginary lines (which are not drawn) has the appearance of a fillet.

If more than two points are supplied, an imaginary series of lines is constructed through them (as in the GSPLNE call). All the lines except the first and last are then divided in two at their mid-points. A series of curved fillets are then drawn, each starting at the end point of the last. Figure 10 on page 141 shows the curve constructed, given current position A and three points B, C, and D.

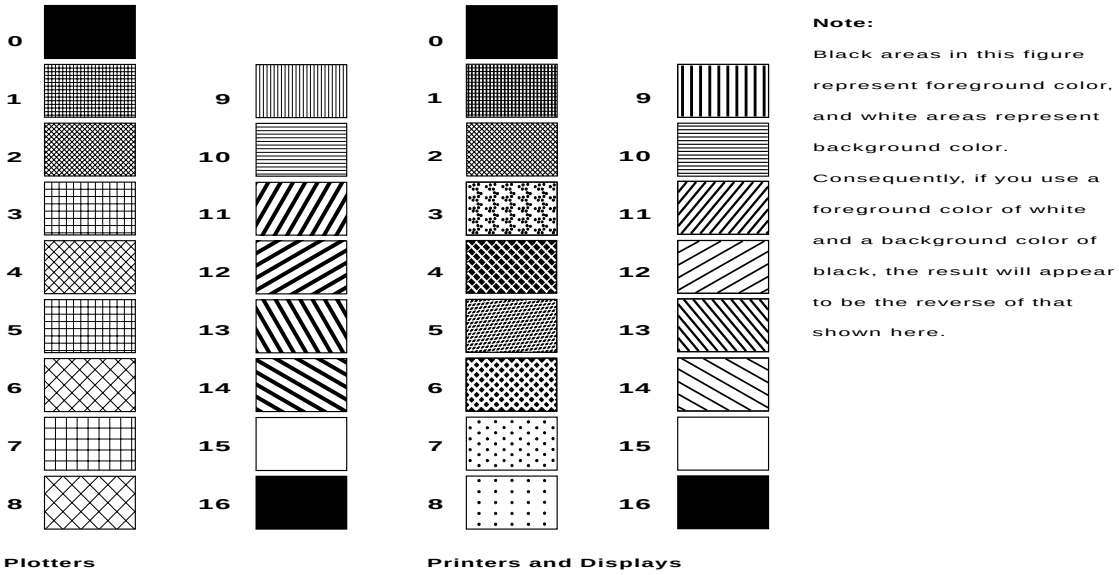


Figure 9. GDDM-defined shading patterns (GSPAT)

The curves have the color, line width, and line type given by the current values of these attributes. The current position is set to the last point.

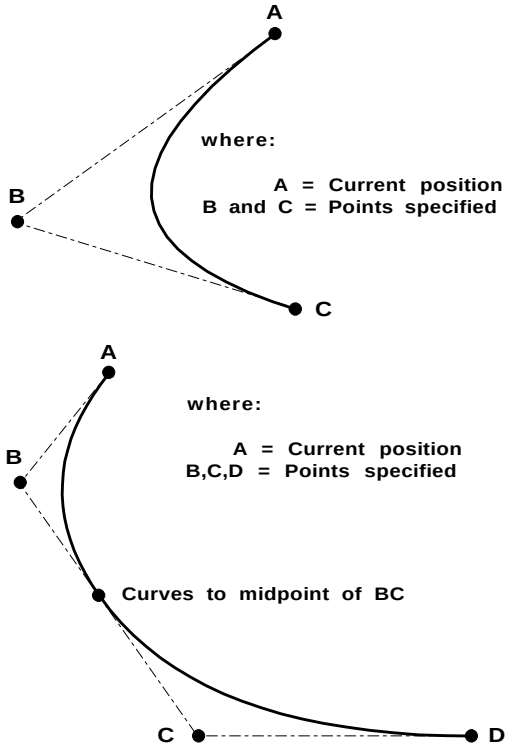


Figure 10. Curved fillets (GSPFLT)

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0154 E  COORDINATE f IS INVALID
```

GSPLNE

Function

To draw a series of lines.

GSPLNE	**(count, xarray, yarray)**
APL code	530
GDDM RCP code	X'0C0C0402' (202114050)

Parameters

count *(specified by user) (fullword integer)*
The number of lines to be drawn.

xarray *(specified by user) (array of short floating-point numbers)*

yarray *(specified by user) (array of short floating-point numbers)*
Arrays containing the end points of the lines. The *nth* line is constructed by drawing a line from current position to the point whose x coordinate is given by the *nth* member of **xarray** and whose y coordinate is given by the *nth* member of **yarray**. After each line has been drawn, the current position is set to the end point.

Description

Draws a series of straight lines starting at the current position and passing through the vector of points specified.

The lines have the color, line width, and line type given by the current values of these attributes. The current position is set to the end point of the last line.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0154 E  COORDINATE f IS INVALID
```

GSPOP

Function

To restore attributes.

GSPOP	**(count)**
APL code	649
GDDM RCP code	X'0C0C1313' (202117907)

Parameters

count *(specified by user) (fullword integer)*
The number of attribute values to be restored.

Description

Restores the primitive attributes that have been saved when new attribute values have been set; see GSAM.

Each time a primitive attribute call (such as color, line type, and so on) is issued when attributes are being saved, the values are put into a “Last in, First out” stack.

The GSPOP call can reset the attribute values (starting with the last one set) to the previous value; this is known as “popping.” This allows a called segment (see GSCALL) to change the values of the attributes, and allows them to be restored on return to the caller (an implicit GSPOP is performed when returning from a called segment). It can also be used to restore a transformation to its previous value after a set current transform; see GSSCT.

When inside an area, GSPOP is only valid if the attribute being popped is valid inside an area. Note that is not possible to check whether the attribute to be popped is valid before issuing the GSPOP call.

Principal errors

```
ADM3225 W  PRIMITIVE ATTRIBUTE STACK EMPTY
ADM3254 E  ATTRIBUTE COUNT n IS INVALID
```

GSPS

Function

To define the picture space.

GSPS	**(width, height)**
APL code	503
GDDM RCP code	X'0C0C0001' (202113025)

Parameters

width *(specified by user) (short floating point)*
height *(specified by user) (short floating point)*
The width and height of the picture space. One of the values must be 1; the other must be greater than zero but must not exceed 1.

Description

Explicitly defines the picture space to be used in the current graphics field. The picture space defines the aspect ratio of the displayed picture unless viewports are explicitly defined with a call to GSVIEW.

The width and height specified are numbers between zero and one that define the aspect ratio (ratio of width to height) of the picture to be constructed.

The center of the picture space is mapped onto the center of the graphics field. The projection ratios are then adjusted so that the entire picture space is displayed in the graphics field as large as possible, while maintaining the aspect ratio specified. Either the left- and right-hand edges, or the top and bottom edges, of the picture space and graphics field coincide. All coincide only if the aspect ratios of picture space and graphics field are identical. If no graphics field is defined or defaulted before GSPS is issued, a default graphics field is set; see GSFLD.

A default picture space is used if no GSPS call is issued before:

- The first viewport is defined or queried
- The first segment is opened
- The first primitive or attribute is drawn or set
- The cursor is queried in graphics coordinates
- Logical input devices are enabled or initialized.

The default equals the actual aspect ratio (in physical units such as millimeters) of the graphics field. Note that this normally alters the aspect ratio from that drawn on the window, if viewports are not explicitly specified. The **default** picture space, therefore, covers the graphics field. For plotters, the default picture space depends on the plotting area

size that is defined in DSOPEN's procopt group 14; see Chapter 19, "Processing options" on page 395.

The aspect ratio of the picture space can be obtained by a call to GSQPS> Once specified or defaulted, the picture space cannot be changed for that graphics field unless the graphics field is cleared by using the GSCLR call. In this case, the picture space can be redefined immediately after the GSCLR call.

Principal errors

```
ADM0162 E  PICTURE  SPACE  PREVIOUSLY  DEFINED  OR
           DEFAULTED
ADM0163 E  PICTURE SPACE SIZE f1{, f2} IS INVALID
```

GSPUT

Function

To restore graphics data.

GSPUT	**(control, length, graphics-data)**
APL code	553
GDDM RCP code	X'0C0C0900' (202115328)

Parameters

control *(specified by user) (fullword integer)*
The coordinate type used in the GDF data. Possible values are:
1 1-byte binary integers
2 2-byte binary integers
4 4-byte short floating point.

length *(specified by user) (fullword integer)*
The length of the GDF string provided. The string of orders is assumed to end when an apparent order code of X'FF' is met, or when the complete string is interpreted, whichever is the sooner.

graphics-data *(specified by user) (character)*
A data area of the indicated byte length containing graphics data in Graphics Data Format. The detailed rules for formatting the GDF string are given in Chapter 10, "GDF order descriptions" on page 281.

Description

Restores the graphics data provided in the **graphics-data** parameter into the current graphics viewport, using the current graphics window coordinates system. The default graphics field, picture space, viewport, and window are defaulted if they were not already specified.

Any series of graphics primitives (such as lines, arcs, areas, character strings) together with their attributes can be coded as a string of data bytes in graphics data format (GDF). This consists of a series of GDF orders; see Chapter 10, "GDF order descriptions" on page 281. Each order defines a graphics primitive or an attribute setting that applies to following primitives.

If graphics data is held by an application program as a series of GDF orders, it can be supplied to GDDM by a single GSPUT call, rather than by many calls to the individual primitives.

The interpretation of a GDF order string is equivalent to the execution of call statements corresponding to the orders in the string. The current position depends on the last order executed, and the current values of the attributes (such as color and line type) depends on the attribute setting orders that have occurred. If the data was generated by previous calls to GSGET, orders exist within that data to set the segment attributes to those that were in effect when the data was obtained. These GDF orders may change any attribute settings that are currently in effect; see GSSATI.

Principal errors

```
ADM0153 E  CONTROL VALUE n IS INVALID
ADM0173 E  STRING LENGTH n IS INVALID
ADM0174 E  INVALID OR UNSUPPORTED GDF ORDER X'xx'
ADM0175 E  INVALID OR UNSUPPORTED LENGTH IN GDF ORDER
           X'xx' OFFSET X'xxxxxxxx'
ADM0182 W  INVALID CHARACTER CODE X'xx' IN STRING
```

Note: Errors can also arise during interpretation of the individual orders, as if the corresponding calls had been made directly.

If such an error occurs, the individual orders before the order in error *will* be processed, but all remaining orders in the graphics-data will be ignored.

GSQAGA

Function

To query all geometric attributes.

GSQAGA	**(id, sx, sy, hx, hy, rx, ry, dx, dy)**
APL code	589
GDDM RCP code	X'0C0C1104' (202117380)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the segment to be queried.

sx *(returned by GDDM) (short floating point)*
sy *(returned by GDDM) (short floating point)*
The x-axis and y-axis scaling factors about the segment origin.

hx *(returned by GDDM) (short floating point)*
hy *(returned by GDDM) (short floating point)*
A pair of values defining a relative vector that denotes a shear parallel to an x-axis through the segment origin.

The values returned represent the coordinates of a point on a vertical line through the segment origin after being sheared. The coordinates are relative to the segment origin.

rx *(returned by GDDM) (short floating point)*
ry *(returned by GDDM) (short floating point)*
A pair of values defining a relative vector that denotes a rotation about the segment origin.

The values returned represent the coordinates of a point on a horizontal line through the segment origin after being rotated. The coordinates are relative to the segment origin.

dx *(returned by GDDM) (short floating point)*
dy *(returned by GDDM) (short floating point)*
The x-axis and y-axis displacements.

Description

Returns a summary of the transform of the segment specified in the **id** parameter expressed in the same terms as those used in the GSSAGA call. The segment must be transformable.

The segment transform is a combination of a scale (**sx,sy**), a shear (**hx,hy**), a rotation (**rx,ry**), and a displacement (**dx,dy**) applied to the segment primitives in that order.

For more information, see the *GDDM Base Application Programming Guide*.

The values returned in this call are expressed in the current world coordinates.

Note: The results from the GSQAGA call **might** not match the geometric attributes of a segment set by means of a single GSSAGA call (because of inherent ambiguity in the attribute specifications). However, the summary produces the same effect as the attributes originally specified.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSQAM

Function

To query the current attribute mode.

GSQAM (n)	
APL code	648
GDDM RCP code	X'0C0C1312' (202117906)

Parameters

n *(returned by GDDM) (fullword integer)*
Receives the current attribute mode. Possible values are:
0 Preserve attributes (the default).
1 Do not preserve attributes.

Description

Returns the current attribute mode, as set by the GSAM call.

Principal errors

None.

GSQATI

Function

To query initial segment attributes.

GSQATI (attribute, value)	
APL code	579
GDDM RCP code	X'0C0C030A' (202113802)

Parameters

attribute *(specified by user) (fullword integer)*
The attribute to be returned by this call. Possible values are:

1 The current detectability status is to be returned.
The values that can be returned in **value** are:

0 Subsequent segments are not detectable.
1 Subsequent segments are detectable.

2 The current visibility status is to be returned.
The values that can be returned in **value** are:

0 Subsequent segments are not visible.
1 Subsequent segments are visible.

3 The current highlight status is to be returned.
The values that can be returned in **value** are:

0 Subsequent segments are not highlighted.
1 Subsequent segments are highlighted.

4 The current transformable status is to be returned.
The values that can be returned in **value** are:

1 Subsequent segments are nontransformable.
2 Subsequent segments are transformable.

5 The current stored/nonstored status is to be returned.
The values that can be returned in **value** are:

0 Subsequent segments are stored.
1 Subsequent segments are nonstored.

6 The current chained/nonchained status is to be returned.
The values that can be returned in **value** are:

0 The segment is to be excluded from the drawing chain.
1 The segment is to be included in the drawing chain (the default).

Note: When a segment is nonchained, it is not added to the drawing chain, and is therefore not drawn unless it is called by another segment; see GSCALL.

value *(returned by GDDM) (fullword integer)*
Receives the current attribute value; see **attribute** above.

Description

Returns the attributes currently assigned to the segments being created. These attributes are modal settings used to determine the initial attributes of new segments as those new segments are created.

Principal errors

```
ADM0184 E  SEGMENT ATTRIBUTE CODE n IS INVALID
```

GSQATS

Function

To query segment attributes.

GSQATS (segment-id, attribute, value)	
APL code	581
GDDM RCP code	X'0C0C030C' (202113804)

Parameters

segment-id *(specified by user) (fullword integer)*
The identification of the segment for which attribute information is to be returned by this call.

attribute *(specified by user) (fullword integer)*
The attribute of the segment, whose current status is to be returned by this call. Possible values are:

1 Detectability
2 Visibility
3 Highlight
4 Transformability
5 Stored/nonstored
6 Chained/nonchained.

Note: When a segment is nonchained, it is not added to the drawing chain, and is therefore not drawn unless it is called by another segment; see GSCALL.

value *(returned by GDDM) (fullword integer)*
Receives the current attribute assigned to the segment. Possible values are:

0 The segment is nondetectable, invisible, not highlighted, stored, or nonchained.
1 The segment is detectable, visible, highlighted, nonstored, nontransformable, or chained.
2 The segment is transformable.

Description

Returns the current value of the specified attribute within the specified segment.

This call does not return the current transform of a transformable segment; for information on this function, see GSQAGA and GSQTFM.

Principal errors

```
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0184 E  SEGMENT ATTRIBUTE CODE n IS INVALID
```

GSQBMX

Function

To query the current background color-mixing mode.

GSQBMX (n)	
APL code	665
GDDM RCP code	X'0C0C1316' (202117910)

Parameters

n *(returned by GDDM) (fullword integer)*
The current value of the background color-mixing mode. Possible values are:
0 The drawing default.
2 “Opaque” mode.
5 “Transparent” mode.

Description

Returns the current value of the background color-mixing mode, as set in the GSBMIX call.

Principal errors

None.

GSQBND

Function

To query the current data boundary definition.

GSQBND (u1, u2, v1, v2)	
APL code	656
GDDM RCP code	X'0C0C000E' (202113038)

Parameters

u1 *(returned by GDDM) (short floating point)*
u2 *(returned by GDDM) (short floating point)*
The left and right extents of the data boundary in world coordinates.
v1 *(returned by GDDM) (short floating point)*
v2 *(returned by GDDM) (short floating point)*
The lower and upper extents of the data boundary in world coordinates.

Description

Returns the definition of the current data boundary in world-coordinate units, as set in the GSBND call.

If a data boundary has not been defined, the values returned in this call are those produced for the current graphics window.

Principal errors

None.

GSQCA

Function

To query character angle.

GSQCA (dx, dy)	
APL code	532
GDDM RCP code	X'0C0C0718' (202114840)

Parameters

dx *(returned by GDDM) (short floating point)*
dy *(returned by GDDM) (short floating point)*
Receives the current values of the character baseline angle coordinates.

Description

Returns the current values of the character baseline angle coordinates, as set by the GSCA call.

Principal errors

None.

GSQCB

Function

To query character-box size.

GSQCB (x, y)	
APL code	533
GDDM RCP code	X'0C0C0717' (202114839)

Parameters

x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
The current dimensions of the character box.

Description

Returns the current values of the character-box size, as set by the GSCB call.

Principal errors

None.

GSQCBS

Function

To query character-box spacing.

GSQCBS (width-multiplier, height-multiplier)	
APL code	650
GDDM RCP code	X'0C0C1310' (202117904)

Parameters

width-multiplier *(returned by GDDM) (short floating point)*
The character box width multiplier.
height-multiplier *(returned by GDDM) (short floating point)*
The character box height multiplier.

Description

Returns the current horizontal and vertical character-box spacing values, as set by the GSCBS call.

Principal errors

None.

GSQCD

Function

To query character direction.

GSQCD (code)	
APL code	534
GDDM RCP code	X'0C0C0719' (202114841)

Parameters

code *(returned by GDDM) (fullword integer)*
The current character direction code.

Description

Returns the code for the current character direction, as set by the GSCD call.

Principal errors

None.

GSQCEL

Function

To query default graphics cell size.

GSQCEL (width, height)	
APL code	535
GDDM RCP code	X'0C0C0202' (202113538)

Parameters

width *(returned by GDDM) (short floating point)*
height *(returned by GDDM) (short floating point)*
Receives the dimensions of the character cell.

Description

Returns the size of the default graphics cell size in current window units.

Principal errors

None.

GSQCH

Function

To query character shear.

GSQCH (dx, dy)	
APL code	559
GDDM RCP code	X'0C0C071C' (202114844)

Parameters

dx *(returned by GDDM) (short floating point)*
dy *(returned by GDDM) (short floating point)*
The current values of the character-shear angle coordinates.

Description

Returns the current values of the character-shear angle coordinates, as set by the GSCH call.

Principal errors

None.

GSQCHO

Function

To query choice device data.

GSQCHO (choice-data)	
APL code	575
GDDM RCP code	X'0C0C0F00' (202116864)

Parameters

choice-data *(returned by GDDM) (fullword integer)*
The choice selected.

For PF keys, it is in the range 1 through 24.

For PA keys, it is in the range 1 through 3.

For Mouse or tablet keys, it is in the range 1 through 3.

For 3270-PC/G and 3270-PC/GX data keys, the value is in the range 1 through 255, and is the character code corresponding to the key that was pressed.

For other choice devices (the ENTER key, the CLEAR key, and the alphanumeric light pen), the choice-data is always set to zero.

Description

Retrieves the data for a choice device from the current input record. The record must correspond to input from a choice device.

Device variations:

For ***3270-PC/G*** and ***3270-PC/GX workstations***, and ***5550-family Multistations***, PA3 is reserved by GDDM (to perform local-mode processing or to redraw the screen).

Principal errors

```
ADM3210 E  REQUIRED DATA NOT FOUND
```

GSQCLP

Function

To query the clipping state.

GSQCLP (state)	
APL code	536
GDDM RCP code	X'0C0C0204' (202113540)

Parameters

state *(returned by GDDM) (fullword integer)*
Receives the current clipping state for the page. Possible values are:
0 Clipping is disabled.
1 Clipping is enabled (precise clip).
2 Clipping is enabled (rough clip).

Description

Returns the current clipping state.

Principal errors

None.

GSQCM

Function

To query the current character mode.

GSQCM (n)	
APL code	537
GDDM RCP code	X'0C0C0715' (202114837)

Parameters

n *(returned by GDDM) (fullword integer)*
The current character mode.

Description

Returns the current value of the character-mode attribute, as set by the GSCM call.

Principal errors

None.

GSQCOL

Function

To query the current color.

GSQCOL (n)	
APL code	538
GDDM RCP code	X'0C0C0711' (202114833)

Parameters

n *(returned by GDDM) (fullword integer)*
Receives the current value of the color attribute.

Description

Returns the current value of the color attribute, as set by the GSCOL call.

Principal errors

None.

GSQCP

Function

To query the current position.

GSQCP	**(x, y)**
APL code	539
GDDM RCP code	X'0C0C0700' (202114816)

Parameters

x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
The x and y world coordinates of the current position.

Description

Returns the current position.

Principal errors

None.

GSQCPG

Function

To query code page.

GSQCPG	**(type, symbol-set-id, code-page-name)**
APL code	216
GDDM RCP code	X'0C040D01' (201592065)

Parameters

type *(specified by user) (fullword integer)*
The type of code-page information that GDDM is to return. The only value that can be specified is:

5 A code page for an IBM 4250 page printer.

symbol-set-id *(specified by user) (fullword integer)*
The identifier of the symbol set for which GDDM is to return the associated code-page name. Possible values are:

0 Requests the name of the current code page.

65 through 223 Requests the name of the code page associated with the specified symbol-set identifier.

code-page-name *(returned by GDDM) (8-byte character string)*
The name (left-justified) of the required code page.

Description

Returns either the name of the current code page or the name of the code page associated with the specified symbol-set identifier.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0120 E  SYMBOL SET n NOT LOADED
```

GSQCS

Function

To query the current symbol-set identifier.

GSQCS	**(n)**
APL code	540
GDDM RCP code	X'0C0C0716' (202114838)

Parameters

n *(returned by GDDM) (fullword integer)*
The current symbol-set identifier.

Description

Returns the current symbol-set identifier, as set by the GSCS call.

Principal errors

None.

GSQCUR

Function

To query the cursor position.

GSQCUR (inwin, x, y)	
APL code	541
GDDM RCP code	X'0C0C0101' (202113281)

Parameters

inwin *(returned by GDDM) (fullword integer)*
Returns information on the position of the alphanumeric cursor. Possible values are:
0 The position returned is outside the window.
1 The position returned is inside the window.
x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
Receives the x and y coordinates of the center of the character cell that contains the cursor.

Description

Returns the coordinates of the current alphanumeric cursor position in current window units. On devices where the alphanumeric cursor is used to emulate a graphics cursor, any movement of the alphanumeric cursor by the operator in response to graphics calls (for example, GSREAD) does not affect the alphanumeric cursor position recorded by GDDM.

The returned values of **x** and **y** are in world coordinates.

For plotters, the **inwin** value is always set to zero because the alphanumeric cursor is always outside the graphics field.

Principal errors

```
ADM0161 E  GRAPHICS FIELD NOT DEFINED
```

GSQFLD

Function

To query the graphics field.

GSQFLD (row, column, depth, width)	
APL code	585
GDDM RCP code	X'0C0C000A' (202113034)

Parameters

row *(returned by GDDM) (fullword integer)*
column *(returned by GDDM) (fullword integer)*
Returns the position on the GDDM page of the top left-hand corner of the graphics field. If no graphics field is currently defined, the parameters return values of 0.
depth *(returned by GDDM) (fullword integer)*
width *(returned by GDDM) (fullword integer)*
Returns the size of the graphics field in terms of rows and columns. If no graphics field is currently defined, the parameters return values of 0.

Description

Returns the position and size of the graphics field, as set by the GSFLD call.

Principal errors

None.

GSQFLW

Function

To query the current fractional line width.

GSQFLW (linewidth-multiplier)	
APL code	562
GDDM RCP code	X'0C0C070F' (202114831)

Parameters

linewidth-multiplier *(returned by GDDM) (short floating point)*
The current value of the fractional line-width multiplier.

Description

Returns the current fractional line width as a floating-point value.

Principal errors

None.

GSQLID

Function

To query logical input device.

GSQLID (device-type, device-id, count, list)	
APL code	643
GDDM RCP code	X'0C0C0C09' (202116105)

Parameters

device-type *(specified by user) (fullword integer)*
The type of logical input device to be queried. For a definition of the valid values, see GSENAB.

device-id *(specified by user) (fullword integer)*
The identification of the logical input device to be queried. For a definition of the valid values, see the GSENAB call description.

count *(specified by user) (fullword integer)*
The number of items of information to be returned in the list.

list *(returned by GDDM) (an array of fullword integers)*
The logical input device information. Possible values are:

1 Enabled state:

–1 The logical input device is not available.
0 The logical input device is not enabled.
1 The logical input device is enabled.

2 Current echo type:

–1 The input device is not available.
n (≥0) The input device is available with the specified echo type. An input device that is not initialized always returns a value of zero.

3 Maximum supported echo type:
The maximum echo type supported for the specified logical input device. Possible values are:

–1 The logical input device does not exist.
0 The logical input device does not support any echoes.
n If the returned value is greater than zero, all echo types with a lower value, including zero, are supported.

Description

Returns information about a specified logical input device, including its availability, its enabled state, and its possible echo types.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
```

GSQLOC

Function

To query graphics locator data.

GSQLOC (inwin, x, y)	
APL code	576
GDDM RCP code	X'0C0C0F01' (202116865)

Parameters

inwin *(returned by GDDM) (fullword integer)*
Indicates whether the position returned is within the window. Possible values are:

0 The position returned is outside the window that was in effect at the time the data was generated

1 The position returned is inside the window that was in effect at the time the data was generated.

x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
The x,y position of the locator from the current input record. The values returned are in window units, using the window that was in effect when the locator data was generated.

Description

Retrieves the locator data from the current input record, which must correspond to input from a locator device.

Note: The locator data generated during GSREAD, and which is awaiting retrieval, is kept by GDDM unmodified, irrespective of whether the particular segment whose identifier appears on the queue has been repositioned or deleted, or if some of the segment attributes have changed.

Principal errors

```
ADM3210 E  REQUIRED DATA NOT FOUND
```

GSQLT

Function

To query the current line type.

GSQLT (n)	
APL code	542
GDDM RCP code	X'0C0C0713' (202114835)

Parameters

n *(returned by GDDM) (fullword integer)*
The current value of the line type.

Description

Returns the current value of the line type, as set by the GSLT call.

Principal errors

None.

GSQLW

Function

To query the current line width.

GSQLW	**(linewidth-multiplier)**
APL code	543
GDDM RCP code	X'0C0C0714' (202114836)

Parameters

linewidth-multiplier *(returned by GDDM) (fullword integer)*
The current value of the line-width multiplier.

Description

Returns the current value of the line width. If a nonintegral line width is queried as an integral value using GSQLW, the integral part is returned.

Principal errors

None.

GSQMAX

Function

To query the number of segments.

GSQMAX	**(n-segments, max-segment)**
APL code	544
GDDM RCP code	X'0C0C0100' (202113280)

Parameters

n-segments *(returned by GDDM) (fullword integer)*
The number of segments currently defined. The number returned does not include segment zero.
max-segment *(returned by GDDM) (fullword integer)*
The maximum segment identifier currently defined. This is zero if there are no segments on the current page.

Description

Returns the number and range of segment identifiers on the current page.

This call can be used to find a unique segment identifier by adding one to the maximum segment identifier returned by this call. This method for finding a unique identifier will not work if the maximum defined segment identifier is the maximum integer that can be represented; that is, 2 147 483 647.

Principal errors

None.

GSQMB

Function

To query marker box.

GSQMB	**(width, depth)**
APL code	637
GDDM RCP code	X'0C0C1308' (202117896)

Parameters

width *(returned by GDDM) (short floating point)*
The width of the marker box in world coordinates.
depth *(returned by GDDM) (short floating point)*
The depth of the marker box in world coordinates.

Description

Queries the value of the current marker box.

Principal errors

None.

GSQMIX

Function

To query the current foreground color-mix mode.

GSQMIX (n)	
APL code	545
GDDM RCP code	X'0C0C0712' (202114834)

Parameters

n *(returned by GDDM) (fullword integer)*
The current value of the foreground color-mix mode. Possible values are:
0 The drawing default.
1 "Mix" mode.
2 "Overpaint" mode.
3 "Underpaint" mode.
4 "Exclusive-OR" mode
5 "Leave alone" or "Transparent" mode.

Description

Returns the current value of the foreground color-mix mode, as set in the GSMIX call.

Principal errors

None.

GSQMS

Function

To query the current marker symbol.

GSQMS (n)	
APL code	546
GDDM RCP code	X'0C0C071B' (202114843)

Parameters

n *(returned by GDDM) (fullword integer)*
The current marker-symbol number.

Description

Returns the current marker-symbol number.

Principal errors

None.

GSQMSC

Function

Note: This call is not recommended for new programs. It is obsolete and has been superseded by GSQMB.

To query marker scale.

GSQMSC (scale)	
APL code	564
GDDM RCP code	X'0C0C071E' (202114846)

Parameters

scale *(returned by GDDM) (short floating point)*
The scale of the marker symbols with respect to the default marker box.

Description

Returns the current marker scale to be used when outputting marker symbols that are defined as vector symbols.

Principal errors

None.

GSQNSS

Function

To query the number of loaded symbol sets.

GSQNSS (n)	
APL code	209
GDDM RCP code	X'0C040102' (201588994)

Parameters

n *(returned by GDDM) (fullword integer)*
The number of symbol sets loaded by GSDSS or GSLSS.

Description

Returns the number of loaded symbol sets. This function is provided to allow the invoker of GDDM to reserve enough storage to perform a subsequent GSQSS request.

Principal errors

None.

GSQORG

Function

To query segment origin.

GSQORG (segment-id, x, y)	
APL code	639
GDDM RCP code	X'0C0C0316' (202113814)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment to be queried.
x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
The x and y coordinates of the segment origin, specified in current world coordinates.

Description

Returns the position of the segment origin of the identified segment in world coordinates. The identified segment need not be transformable. For more information, see GSSPOS and GSSORG.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSQPAT

Function

To query the current shading pattern.

GSQPAT (n)	
APL code	547
GDDM RCP code	X'0C0C071A' (202114842)

Parameters

n *(returned by GDDM) (fullword integer)*
The current shading-pattern number.

Description

Returns the current shading-pattern number.

Principal errors

None.

GSQPIK

Function

To query pick data.

GSQPIK (segment-id, primitive-tag)	
APL code	577
GDDM RCP code	X'0C0C0F02' (202116866)

Parameters

segment-id *(returned by GDDM) (fullword integer)*
The segment chosen by the terminal operator. If it is 0, the detection process was initiated, but no primitive was identified.
primitive-tag *(returned by GDDM) (fullword integer)*
The tag of the primitive within the specified segment chosen by the operator. If it is 0, the detection process was initiated, but no primitive was identified.

Description

Retrieves data from the current input record if the current record contains pick data. If the input record does not contain pick data, an error is raised.

Note: The pick data generated during GSREAD, and which is awaiting retrieval, is kept by GDDM unmodified, irrespective of whether the particular segment whose identifier appears on the queue has been repositioned or deleted, or if some of the segment attributes have changed.

Principal errors

```
ADM3210 E  REQUIRED DATA NOT FOUND
```

GSQPKS

Function

To query pick structure.

GSQPKS (count, segids, tagids, depth)	
APL code	654
GDDM RCP code	X'0C0C0F05' (202116869)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in the segment and tag arrays.
segids *(returned by GDDM) (an array of fullword integers)*
Array for segment identifiers.
tagids *(returned by GDDM) (an array of fullword integers)*
Array for tag identifiers.
depth *(returned by GDDM) (fullword integer)*
The number of segment and tag pairs available in the current input record (depth of pick path).

Description

Retrieves pick structure data from the current input record if the current record contains pick data. If the input record does not contain pick data, an error message is issued.

Note: The pick data generated during GSREAD, and which is awaiting retrieval, is kept by GDDM unmodified, irrespective of whether the particular segment whose identifier appears on the queue has been repositioned or deleted, or if some of the segment attributes have changed.

The data returned consists of pairs of segment identifiers and tags, which indicate the level of nesting. The data returned consists of the segment and tag of the picked data, the segment and tag of all calling segments until the root segment (the segment that was not called by another segment) is reached.

If a segment is called in several places within another segment, a tag before each call would allow the particular instance of the called segment to be identified.

Principal errors

ADM3210 E REQUIRED DATA NOT FOUND

GSQPOS

Function

To query segment position.

GSQPOS (segment-id, x, y)	
APL code	583
GDDM RCP code	X'0C0C030E' (202113806)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment whose position is to be returned.
x *(returned by GDDM) (short floating point)*
y *(returned by GDDM) (short floating point)*
The segment origin, in world coordinates, of the identified segment.

Description

Returns, in world coordinates, the current position of the segment origin of the identified segment. For more information, see GSSPOS and GSSORG.

Principal errors

None.

GSQPRI

Function

To query segment priority.

GSQPRI (ref-seg-id, seg-id, order)	
APL code	635
GDDM RCP code	X'0C0C0313' (202113811)

Parameters

ref-seg-id *(specified by user) (fullword integer)*
The identifier of a reference segment. A value of 0 shows that either the lowest or highest priority segment is to be returned in the **seg-id** parameter, as defined by the value in the **order** parameter.

seg-id *(returned by GDDM) (fullword integer)*
The identifier of the segment that is immediately before or after the segment specified in the **ref-seg-id** parameter. A returned value of 0 shows that the segment specified in the **ref-seg-id** parameter is either the lowest priority segment

(when **order**= –1) or the highest priority segment (when **order**= 1).

order *(specified by user) (fullword integer)*
Shows whether a segment identifier of a higher or lower priority than the segment identified in the **ref-seg-id** parameter is to be returned. Possible values are:

–**1** Query the identifier of the segment with a lower priority than the one named in the **ref-seg-id** parameter or, if **ref-seg-id**=0, query the identifier of the segment with the lowest priority

1 Query the identifier of the segment with a higher priority than the one named in the **ref-seg-id** parameter or, if **ref-seg-id**=0, query the identifier of the segment with the highest priority.

Description

Returns the identifier of the named segment that is before or after a specified named segment. The segment that is before the specified segment is considered to have a lower priority than the specified segment; similarly, the segment that is after the specified segment is considered to have a higher priority than the specified segment.

If all the primitives in all the segments are drawn in overpaint mode (see GSMIX), a segment with a higher priority is always drawn “on top” of all segments of a lower priority.

Note: The GSQPIK call returns the segment with the higher priority if there is more than one detectable segment with a pickable primitive in the pick window.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM3226 E  SEGMENT n IS UNKNOWN
ADM3228 E  SEGMENT ORDERING n IS INVALID
```

GSQPS

Function

To query the picture-space definition.

GSQPS	**(width, height)**
APL code	548
GDDM RCP code	X'0C0C0004' (202113028)

Parameters

width *(returned by GDDM) (short floating point)*
height *(returned by GDDM) (short floating point)*
The width and height of the picture space. One of the values returned is 1; the other is less than 1.

Description

Returns the picture-space definition. This function can be used to adapt a picture layout to suit the shape of the graphics field provided.

GSQPS does not create a default picture space; it returns information about the default picture space if an explicit picture space was not defined.

Principal errors

None.

GSQSEN

Function

To query mixed string attribute of graphics text.

GSQSEN	**(mixed)**
APL code	667
GDDM RCP code	X'0C0C1B01' (202119937)

Parameters

mixed *(returned by GDDM) (fullword integer)*
The mixed string attribute. Possible values are:
0 Default (Mixed with position).
1 Mixed with position.
2 Mixed without position.

Description

Returns the current value of mixed string attribute set by GSSEN.

Principal errors

None.

GSQSIM

Function

To query existence of simultaneous queue entry.

GSQSIM	**(event-flag)**
APL code	574
GDDM RCP code	X'0C0C0E01' (202116609)

Parameters

event-flag *(returned by GDDM) (fullword integer)*
Identifies whether there are any more simultaneous events on the input queue. Possible values are:

- **0** There are no more simultaneous events on the input queue. The next GSREAD updates the screen and waits for input from the enabled logical input devices.
- **1** There are more simultaneous events on the input queue. The next GSREAD call removes the top input record from the queue and returns the information to the program. The screen is not updated.

Description

Shows whether there are any more simultaneous events on the input queue.

Principal errors

None.

GSQSS

Function

To query loaded symbol sets.

GSQSS	**(n, types, symbol-set-names, symbol-set-ids)**
APL code	210
GDDM RCP code	X'0C040103' (201588995)

Note: For information about querying the number of loaded symbol sets, see "GSQNSS – Query the number of loaded symbol sets" on page 153.

Parameters

n *(specified by user) (fullword integer)*
The number of loaded symbol sets to be queried.

types *(returned by GDDM) (an array of fullword integers)*
Identifies the types of symbol definitions. Possible values are:

- **1** Image symbol set
- **2** Vector symbol set
- **3** Shading-pattern set
- **4** Marker symbol set
- **5** 4250 page printer font
- **8** DBCS image symbol set
- **9** DBCS vector symbol set.

symbol-set-names *(returned by GDDM) (array of 8-byte character tokens)*
The files from which the symbol sets were loaded or the names specified in calls to GSDSS. The symbol-set name returned for a symbol set that originally had the substitution character "." in its name is the substituted name. In the case of a DBCS symbol set name, the substitution character will be replaced by its substitute, but ward digits will not be appended.

symbol-set-ids *(returned by GDDM) (an array of fullword integers)*
The identifiers associated with each symbol set.

Description

Returns information about all symbol sets (image, vector, and 4250 fonts) loaded by GSDSS or GSLSS.

Each of the parameters (apart from **n**) is an array with **n** elements. Information about the first **n** symbol sets is returned; if there are fewer than **n**, the types and symbol-set identifiers for the remainder are cleared to zero.

Principal errors

```
ADM0116 E  NUMBER OF SYMBOL SETS n IS INVALID
```

GSQSSD

Function

To query symbol set data.

GSQSSD	**(type, id, count, data)**
APL code	586
GDDM RCP code	X'0C0C0102' (202113282)

Parameters

type *(specified by user) (fullword integer)*
The type of symbol set that is to be queried. The values correspond to the type defined in the GSDSS or GSLSS call, and are:

- **1** Image symbol set
- **2** Vector symbol set

5 4250 page printer font
8 DBCS image symbol set
9 DBCS vector symbol set.

id *(specified by user) (fullword integer)*
The identifier of the symbol set to be queried.

count *(specified by user) (fullword integer)*
The number of elements in the **data** array.

data *(returned by GDDM) (array of short floating-point numbers)*
Information about the symbol set. The values returned depend on the type of symbol set specified in the **type** parameter. Possible values are:

1 The first two elements of the array contain the width and depth (respectively) of a symbol definition, given in world coordinates.

2 The first two elements of the array contain the aspect ratio of the grid on which the vector symbol set is defined. The first element is the width (as a proportion of the depth); the second element is always 1.

An application program can preserve the aspect ratio of the vector symbols by specifying a character box with the same aspect ratio as the symbol set.

5 The first two elements of the array contain the maximum character width and the baseline increment (respectively) of the specified font, given in world coordinates.

8 As for **1**.

9 As for **2**.

Description

Returns information about a specific symbol set that has been loaded by a GSDSS call or a GSLSS call. Using the values returned by GSQSSD, the character cell size can be set so that loadable symbols are drawn with the aspect ratio or size with which they were created.

GSQSSD sets the default graphics field, picture space, viewport, and window if they have not already been specified or defaulted.

A GDDM DBCS symbol set ward must be loaded before information about a DBCS symbol set can be queried. No symbol set wards will have been loaded by a GSLSS call, as there are no ward digits available to this call. However, a DBCS ward may have been loaded by a previous GSCHAR call. If no ward has been loaded, the GSQSSD call will attempt to load a DBCS ward, to obtain the requested information.

Principal errors

```
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0120 E  SYMBOL SET n NOT LOADED
ADM0129 E  ARRAY COUNT n IS INVALID
ADM3279 E  NO DBCS WARD AVAILABLE FOR SYMBOL SET 'a'
           IDENTIFIER n
```

GSQSTK

Function

To query stroke data.

GSQSTK	**(count, draw-flag, x, y, num-points)**
APL code	597
GDDM RCP code	X'0C0C0F04' (202116868)

Parameters

count *(specified by user) (fullword integer)*
The number of elements in the arrays **x** and **y**. If **num-points**, the number of coordinate pairs available in the current input record, is less than **count**, excess elements in the arrays are undefined. If **num-points** is greater than **count**, only **count** coordinate points are returned in the arrays **x** and **y**.

draw-flag *(returned by GDDM) (an array of fullword integers)*
An array of control flags for each x,y pair, specifying:

- For polylocator sampling method:
  –**1** The x,y pair is not defined
  >**0** The number of the tablet or mouse button pressed to record the x,y pair.
- For stream sampling method:
  –**1** The x,y pair is not defined.
  0 The x,y pair represents a point sampled from the trajectory of a moving cursor (that is, within, or at the end of, a polyline).
  1 The x,y pair starts a polyline. This value is recorded, when stream mode is entered, by pressing a tablet or mouse button.

x *(returned by GDDM) (array of short floating-point numbers)*
y *(returned by GDDM) (array of short floating-point numbers)*
Two arrays that return the x,y coordinate pairs. The values returned are in window units, using the window that was in effect when the stroke data was generated.

num-points *(returned by GDDM) (fullword integer)*
The number of coordinate pairs available in the current input record.

Description

Retrieves the stroke data from the current input record, which must correspond to input from a stroke device.

The array in the **draw-flag** parameter shows whether an x,y coordinate pair was generated, and how the x,y coordinate pair was generated.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM3210 E  REQUIRED DATA NOT FOUND
```

GSQSTR

Function

To query string data.

GSQSTR	**(count, string, cursor-pos)**
APL code	596
GDDM RCP code	X'0C0C0F03' (202116867)

Parameters

count *(specified by user) (fullword integer)*
The length of the string that is to be returned.
string *(returned by GDDM) (character)*
The input character string (left-justified, truncated, or padded with blanks).
cursor-pos *(returned by GDDM) (fullword integer)*
The final position of the cursor within the string, or zero if a position cannot be detected.

Description

Returns the string data from the current input record, which must correspond to input from a string device.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM3210 E  REQUIRED DATA NOT FOUND
```

GSQSVL

Function

To query the current segment viewing limits.

GSQSVL	**(u1, u2, v1, v2)**
APL code	659
GDDM RCP code	X'0C0C1315' (202117909)

Parameters

u1 *(returned by GDDM) (short floating point)*
u2 *(returned by GDDM) (short floating point)*
The left and right extents of the segment-viewing limits in world coordinates.
v1 *(returned by GDDM) (short floating point)*
v2 *(returned by GDDM) (short floating point)*
The lower and upper extents of the segment-viewing limits in world coordinates.

Description

Returns the viewing limits of the current segment in world coordinates.

If segment viewing limits have not been explicitly defined, the values returned are those used for overall picture clipping.

Principal errors

None.

GSQTA

Function

To query the current text alignment.

GSQTA	**(horiz, vert)**
APL code	645
GDDM RCP code	X'0C0C130E' (202117902)

Parameters

horiz *(returned by GDDM) (fullword integer)*
The current horizontal text alignment values.
vert *(returned by GDDM) (fullword integer)*
The current vertical text alignment values.

Description

Returns the current horizontal and vertical text alignment values, as set by the GSTA call.

Principal errors

None

GSQTAG

Function

To query current tag.

GSQTAG (tag)	
APL code	567
GDDM RCP code	X'0C0C1001' (202117121)

Parameters

tag *(returned by GDDM) (fullword integer)*
The current tag identifier being assigned within the specified segment.

Description

Returns the current tag as set by the GSTAG call.

Principal errors

None.

GSQTB

Function

To query the text box.

GSQTB (length, string, count, x-array, y-array)	
APL code	560
GDDM RCP code	X'0C0C0502' (202114306)

Parameters

length *(specified by user) (fullword integer)*
The number of characters in the string.

string *(specified by user) (character)*
The character string to be processed.

If GSCS(8) is specified or if GDDM's external defaults contain the MIXSOSI option, the character strings can contain DBCS characters for modes 2 and 3.

count *(specified by user) (fullword integer)*
The number of elements to be returned in **x-array** and **y-array**. If the values specified exceed 5, all values after the fifth are set to zero.

x-array *(returned by GDDM) (array of short floating-point numbers)*
y-array *(returned by GDDM) (array of short floating-point numbers)*
Two arrays containing the relative coordinates of the text box. The elements of each array are defined as:

1 Top-left corner
2 Bottom-left corner
3 Top-right corner
4 Bottom-right corner
5 Concatenation point.

The terms top-left, bottom-right, and so on are well defined when the character angle is such that the baseline is parallel to the x axis and running left to right, and there is no character shear. If the character string is rotated or sheared, the term top-left applies to the corner of the box that appears in the top-left position when no rotation or shear is applied.

This is an example:

```
Set character angle = -1,1
String = ABCDE
Coordinates returned are as shown:
```

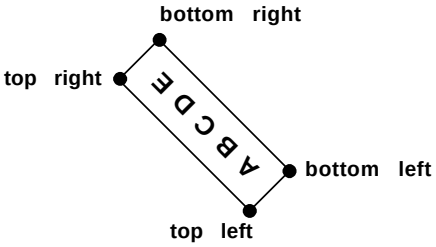


Figure 11. Text box enclosing rotated characters (GSQTB)

Description

Returns the x and y components of the relative coordinates of the four corners of a text box, where the text box is defined as the parallelogram that encloses the specified character string when displayed on the device. Also returned are the relative coordinates of the concatenation point; that is, the position where the next character would have been. All coordinates are relative to the start point, as defined in the GSCD call. The box is evaluated using the current char-

acter mode, box, box spacing, angle, shear, direction, set and text alignment.

For mode-2 text, GSQTB returns the coordinates of a box that encloses the character **boxes** (not the symbols) within the string. Image symbols do not necessarily fill the character boxes, and can also extend outside them. If the boxes are angled, their edges are "staircased." In all instances, the text box is defined as running through the extremities of the character boxes.

More information on how to use this call is given in the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM0111 W  DBCS SYMBOL SET 'a' NOT AVAILABLE
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM0169 E  CHARACTER STRING LENGTH n IS INVALID
ADM3252 W  CHARACTER   X'xx'   REPLACED   BY   SHIFT-IN
           CHARACTER
ADM3253 W  DBCS  CHARACTER  X'xxxx'  IS  INVALID  AND
           REPLACED BY A BLANK
ADM3264 W  DBCS CHARACTER STRING LENGTH n MUST BE EVEN
```

GSQTFM

Function

To query segment transform.

GSQTFM	**(id, n, array)**
APL code	591
GDDM RCP code	X'0C0C1105' (202117381)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the segment.

n *(specified by user) (fullword integer)*
The number of elements that are to be set in the **array** parameter; **n** must be in the range 0 through 9.

array *(returned by GDDM) (array of short floating-point numbers)*
The array into which the elements of the transform are to be returned.

Description

Returns the elements of the transform of the identified segment. The transformation matrix that GDDM returns is expressed in current window coordinates.

For more information on segment transforms, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM0129 E  ARRAY COUNT n IS INVALID
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSQVIE

Function

To query the current viewport definition.

GSQVIE	**(x1, x2, y1, y2)**
APL code	549
GDDM RCP code	X'0C0C0005' (202113029)

Note: This call creates a default graphics field if there is not one already on the current page.

Parameters

x1 *(returned by GDDM) (short floating point)*
x2 *(returned by GDDM) (short floating point)*
The position within the picture space of the left and right boundaries of the viewport.

y1 *(returned by GDDM) (short floating point)*
y2 *(returned by GDDM) (short floating point)*
The position within the picture space of the lower and upper boundaries of the viewport.

Description

Returns the current viewport definition in picture-space units.

Principal errors

None.

GSQWIN

Function

To query the current window definition.

GSQWIN	**(u1, u2, v1, v2)**
APL code	550
GDDM RCP code	X'0C0C0006' (202113030)

Parameters

u1 *(returned by GDDM) (short floating point)*
u2 *(returned by GDDM) (short floating point)*
The left and right boundaries of the window.
v1 *(returned by GDDM) (short floating point)*
v2 *(returned by GDDM) (short floating point)*
The lower and upper boundaries of the window.

Description

Returns the current window definition in world-coordinate units. If a window has not been defined, the values returned in this call are those produced for a default window.

Principal errors

None.

GSREAD

Function

To await graphics input.

GSREAD (delay, input-device-type, input-device-id)	
APL code	120
GDDM RCP code	X'0C100003' (202375171)

Parameters

delay *(specified by user) (fullword integer)*
Identifies whether GDDM should await action from the terminal operator if the input queue is empty. Possible values are:

0 GDDM is not to await action from the terminal operator if the input queue is empty. An input-device-type of 0 is returned to show an empty queue.

1 GDDM is to await action from the terminal operator if the input queue is empty. This option can only be specified if there are currently enabled input devices.

input-device-type *(returned by GDDM) (fullword integer)*
The type of the input data. Possible values are:

0 The input queue is empty. This value is returned only if 0 is specified for the **delay** parameter.
1 Choice device data.
2 Locator data.
3 Pick data.
4 String data.
5 Stroke data.

input-device-id *(returned by GDDM) (fullword integer)*
The source of the input data. Possible values are:

For choice devices:
The value returned identifies the choice device group. Values that can be returned are:

0 The ENTER key.
1 The PF keys.
2 The alphanumeric light pen.
4 The PA keys.
5 The CLEAR key.
8 Data keys
10 The mouse or puck buttons

Note: Not all values can be returned from all devices. For a list of supported values for specific devices, see GSENAB.

More information on the specific choice can be obtained using the GSQCHO call.

For locator devices:

1 The device identifier of the locator device.

The data returned by the locator can be obtained by using the GSQLOC call.

For pick devices:

1 The device identifier of the pick device.

The data returned by the pick can be obtained by using the GSQPIK call.

For string devices:

1 The device identifier of the string device.

The data returned by the string can be obtained by using the GSQSTR call.

For stroke devices:

1 The device identifier of the stroke device.

The data returned by the stroke can be obtained by using the GSQSTK call.

Description

Returns the next graphics event, if necessary by performing all outstanding output and awaiting input from a graphics logical input device.

The processing of the GSREAD call is affected by the state of the input queue, which contains information about the input devices that the operator used during any interactions with the terminal.

If the input queue contains records, the screen images are not updated. The GSREAD call merely removes from the input queue the top input record for a partition and page in the current partition set, and returns the appropriate information (the device type and identifier) to the application program. The top input record becomes the current input record. The partition and page from which the entry was generated becomes the current partition and current page.

If the input queue is empty, and if the application program has requested that GDDM should wait for operator action, and if some input devices are enabled, the current screen images (both graphics and alphanumeric) are updated; GDDM awaits operator action from the enabled devices. If no input devices are enabled, an error message is issued.

The position of the graphics cursor is identified by echoes from the relevant enabled-logical-input device; that is, pick and locator for 3179-G or 3192-G color display stations, or stroke, locator, and pick for 3270-PC/G and 3270-PC/GX workstations, and the 5080 Graphics System.

On other devices, the alphanumeric cursor is located at the current locator or pick position.

For all devices, only one graphics cursor is displayed on the screen. If more than one enabled logical input device requires a graphics cursor, the order of priority for positioning the cursor is: first, the stroke, then the locator, and finally, the pick.

When the operator causes an interrupt, the data returned is added to the input queue. Note that multiple input records might be generated if multiple input devices are enabled. Any changes to alphanumeric fields can be determined by issuing an ASQMOD call.

If the operator-generated interrupt does not result in any data being added to the input queue, the audible alarm is sounded, the terminal interrupt is ignored, and the read operation is tried again.

After the input queue has been generated, the current input record is updated and the input device type and identifier are returned to the application program. The partition from which the interrupt was generated becomes the current partition.

Locator and pick devices are triggered by pressing a key that returns the position of the device.

String and stroke devices are triggered by mouse or puck buttons, data keys, the ENTER key or PF keys, except that stroke devices are not triggered by tablet or mouse buttons.

If a pick is enabled and triggered, an entry is always generated on the queue for the pick device, regardless of whether a primitive was detected. GSQPIK returns an indication as to whether a primitive was actually detected or not.

When the picture is displayed for GSREAD, the page is scrolled (see FSPWIN) to ensure that the alphanumeric cursor position is within the displayed data on the screen. If the alphanumeric cursor is being used as a graphic locator device (as happens on devices such as the 3279), this scrolling positions the page so that the graphic locator position is visible.

For information about restrictions on various devices, see “Graphics logical input devices” on page 247.

Principal errors

```
ADM0270 E  SCREEN FORMAT ERROR
ADM0273 W  PS OVERFLOW
ADM0275 W  GRAPHICS {(IMAGE) }CANNOT BE SHOWN. REASON
           CODE n
ADM0276 W  DEVICE IS OUTPUT ONLY
ADM2864 W  PICTURE IS TOO LARGE FOR 5080 DISPLAY LIST
           BUFFER
ADM3004 E  FIELD LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3005 E  DATA BUFFER n1, ERROR n2 AT INDEX n3
ADM3010 E  BUNDLE LIST n1, ERROR n2 AT ARRAY ELEMENT
           (n3,n4)
ADM3170 E  NO ENABLED INPUT DEVICES
ADM3172 E  INVALID READ DELAY VALUE
ADM3173 W  GRAPHICS CANNOT BE SHOWN.  CELL WIDTH OR
           DEPTH EXCEEDS LOADABLE LIMIT
ADM3175 E  UNEXPECTED ERROR FROM DEVICE. LOG ERROR
           DATA: X'xxxxxxxxxxxx'
ADM3176 W  ECHO SEGMENT NOT STORED IN DEVICE. DEFAULT
           LOCATOR USED
ADM3177 W  INSUFFICIENT SEGMENT STORAGE. STROKE ENTRIES
           REDUCED TO n
ADM3178 W  PATTERNS CANNOT BE SENT TO DEVICE.  AREA
           SHADING MAY BE INCORRECT
ADM3179 W  IMAGE CANNOT BE SHOWN. REASON CODE n
```

GSRSS

Function

To release a graphics symbol set.

GSRSS	**(type, symbol-set-id)**
APL code	207
GDDM RCP code	X'0C040401' (201589761)

Parameters

type *(specified by user) (fullword integer)*
The type of the symbol set to be released. Possible values are:

- **1** Image symbol set
- **2** Vector symbol set
- **3** Shading-pattern set
- **4** Marker symbol set
- **5** 4250 high-resolution printer font
- **8** DBCS image symbol set
- **9** DBCS vector symbol set.

symbol-set-id *(specified by user) (fullword integer)*
The identifier of the symbol set to be released; see GSLSS. Must be specified as zero for patterns and markers.

Description

Releases the specified symbol set. The type must correspond exactly to that specified when the symbol set was loaded or defined. A symbol set may be released, freeing the storage occupied, when the application program no longer needs it. A symbol set should not be released while a graphics picture using it still exists.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0118 E  SYMBOL SET TYPE n IS INVALID
ADM0120 E  SYMBOL SET n NOT LOADED
```

GSSAGA

Function

To set all geometric attributes.

GSSAGA	**(id, sx, sy, hx, hy, rx, ry, dx, dy, type)**
APL code	588
GDDM RCP code	X'0C0C1102' (202117378)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the transformable segment.

sx *(specified by user) (short floating point)*
sy *(specified by user) (short floating point)*
A scale transformation in terms of an x-axis scaling (**sx**) and a y-axis scaling (**sy**). The segment origin is used as a reference point: the axes that are used to scale are parallel to the x and y axes but pass through the segment origin. A scale factor that has magnitude in the range 0 through 1 shrinks primitives; a scale factor of greater than 1 stretches primitives. A negative scale factor reflects primitives about the other axis.

Specifying scale factors **sx=1** and **sy=1** does not perform any scaling. This setting can be used to suppress scaling (to allow a simple rotation, for example).

hx *(specified by user) (short floating point)*
hy *(specified by user) (short floating point)*
A shear transformation in terms of the displacements that a point on the y axis makes after shearing. The axes used for shearing are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used in the GSCH call for character shear. Note that primitives below the x axis are sheared in the opposite direction to those above the x axis. The points on the x axis itself are not moved. If **hx=a** and **hy=b** are used, an identical effect is achieved with **hx=–a** and **hy=–b**.

Specifying shear factors **hx=0** and **hy=1** does not perform any shearing. This setting can be used to suppress shearing (to allow a simple rotation, for example).

Specifying **hy=0** is not valid.

rx *(specified by user) (short floating point)*
ry *(specified by user) (short floating point)*
A rotation transformation in terms of the displacements that a point on the x axis makes after rotating. The axes used for rotating are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used for the character angle in the GSCA call.

Specifying rotation components **rx=1** and **ry=0** does not perform any rotation. This setting can be used to suppress rotation (to allow a simple scaling, for example).

Because two zero values would be ambiguous, specifying **rx=0** and **ry=0** is taken as equivalent to **rx=1** and **ry=0** (no rotation).

The (rx,ry) values below produce these special cases:

(0,0) no rotation; equivalent to (1,0).
(1,0) no rotation.
(0,1) rotation by 90 degrees counterclockwise.
(1,1) rotation by 45 degrees counterclockwise.
(0,–1) rotation by 90 degrees clockwise.
(–1,0) rotation by 180 degrees clockwise (or counterclockwise).

and, in general:

(rx,ry) rotation by theta degrees counterclockwise, where tan(theta)=ry/rx (assuming a uniform world coordinate system).

Note: A rotation of (–1,0) is equivalent to a scale factor of **sx=–1** and **sy=–1**. This inherent ambiguity prevents GSQAGA returning values that are necessarily equal to those specified. However, the values returned in GSQAGA are consistent in that the components (taken together) are equivalent to those specified in GSSAGA.

dx *(specified by user) (short floating point)*
dy *(specified by user) (short floating point)*
A displacement of **dx** parallel to the x axis and **dy** parallel to the y axis. This transformation does not use the segment origin.

Specifying displacements components of **dx=0** and **dy=0** does not perform any displacement. This setting can be used to suppress displacements (to allow a simple rotation, for example).

type *(specified by user) (fullword integer)*
How the existing segment transform is to be modified by the scaling, shear, rotation, and displacement components specified. Possible values are:

0 New/replace
Any transform previously defined for the segment is discarded and is replaced by the combined effect of the specified components.

1 Additive
The combined effect of the specified components is added to the effect already present in the segment transform. The new transform combines both effects in the order (i) old transform, and (ii) GSSAGA parameter values. This option is the most useful for incremental updates to segment transforms.

2 Preemptive
The combined effect of the specified components is added to the effect of the existing segment transform. The new transform combines both effects in the order (i) GSSAGA parameter values, and (ii) the old transform. The effect is as if the GSSAGA parameters modify the primitives of the segment (without transformation) and the existing transformation is applied again.

Description

Sets or modifies the transform of the identified segment by specifying x-axis and y-axis scale factors, and shear, rotation, and displacement values. The parameters are specified in current window coordinates (as defined in the GSWIN call). If the window coordinates are not uniform (see GSUWIN for how to define these), graphics data appears to be sheared as it is rotated.

A segment transform alters the displayed appearance of a segment without altering the primitives within the segment. An existing segment transform can be canceled at any time (by specifying **type=0**), thus reverting to the original appearance.

The combined effect of the scale, shear, rotation, and displacement specified is used to update the segment transform. This is performed in three distinct ways, controlled by the **type** parameter. In all cases, the other parameters are taken as a unit, the effects being combined into a single composite transformation; that is, scale, shear, rotation, and displacement, in that order.

Note: The GSSTFM call also allows the segment transforms to be set or modified using matrix algebra, and is therefore more mathematically oriented.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSSATI

Function

To set initial segment attributes.

GSSATI	**(attribute, value)**
APL code	578
GDDM RCP code	X'0C0C0309' (202113801)

Parameters

attribute *(specified by user) (fullword integer)*
The number of the segment attribute (for example, 2 for visibility).

value *(specified by user) (fullword integer)*
The setting of the segment attribute specified in the **attribute** parameter. Possible values are:

1 Detectability attribute.
A segment can be detectable (**value** = 1) or non-detectable (**value** = 0).

When a segment is made detectable, the segment identifier can be returned as a result of selection through a pick device. A segment cannot be detected by using a pick device if that segment is invisible. The value given in the GSTAG call must not be zero if a segment is to be detectable.

2 Visibility attribute.
A segment can be visible (**value** = 1) or invisible (**value** = 0).

When a segment is visible it is displayed. An invisible segment does not appear on the display, nor can it be detected.

3 Highlight attribute.
A segment can be highlighted (**value** = 1) or nonhighlighted (**value** = 0).

The effect of highlighting varies according to the device, as follows:

On displays it is white,
On printers it is black,
On plotters it takes the color associated with pen 7, or the highest available pen number.

4 Transformability attribute.
A segment can be either transformable (**value** = 2) or nontransformable (**value** = 1).

All segments can be repositioned using the GSSPOS call, transformed (scaled, sheared, rotated, and displaced) using the GSSAGA or GSSTFM call, and used as an echo segment. If a segment is not going to be used for any of these purposes it should be marked as nontransformable as GDDM may use this attribute to optimize the data stream.

5 Not used.

6 Chaining attribute.
A segment can be either chained (**value** = 1) or nonchained (**value** = 0).

When a segment is nonchained, it is not added to the drawing chain, and is therefore not drawn unless it is called by another segment; see GSCALL.

Description

Specifies the attributes to be assumed by subsequently created segments. These attributes are modal settings used to determine the initial attributes of new segments as they are created.

When a graphics field is created, the segment attributes are nontransformable, nondetectable, visible, nonhighlighted, stored, and chained.

The GSSATI call causes the default window, viewport picture space, graphics, page, and so on, to be set up.

Principal errors

```
ADM0184 E  SEGMENT ATTRIBUTE CODE n IS INVALID
ADM0185 E  SEGMENT ATTRIBUTE VALUE n IS INVALID
```

GSSATS

Function

To modify segment attributes.

GSSATS	**(segment-id, attribute, value)**
APL code	580
GDDM RCP code	X'0C0C030B' (202113803)

Parameters

segment-id *(specified by user) (fullword integer)*
The identification of the segment to be modified by this call. A value of zero cannot be specified.

attribute *(specified by user) (fullword integer)*
The segment attribute that is to be modified by this call. Possible values are:

1 Detectability
2 Visibility
3 Highlighting.
The effect of highlighting varies according to the device:
On displays it is white,
On printers it is black,
On plotters it takes the color associated with pen 7, or the highest available pen number.
4 Transformability
5 Not used
6 Chained/nonchained mode.
When a segment is nonchained, it is not added to the drawing chain unless it is called by another segment; see GSCALL.

value *(specified by user) (fullword integer)*
The desired value to be assigned to the segment attribute. Possible values are:

0 The segment is to become nondetectable, invisible, nonhighlighted, or excluded from the drawing chain, as defined by the contents of **attribute**.
1 The segment is to become detectable, visible, highlighted, nontransformable, or included in the drawing chain (default).
2 The segment is to become transformable.

Description

Sets the dynamic attributes that can be assigned to a segment.

The dynamic attributes are detectability, visibility, highlighting, transformability, and chain mode. Segment zero cannot have attributes applied to it.

The highlighting attribute has the effect of setting the color for all primitives in the segment to neutral, which is white for displays, and black for printers. For more information on the effect of this on graphics devices, see GSCOL.

When a segment is modified from nonchained to chained, it is added to the end of the drawing chain.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0184 E  SEGMENT ATTRIBUTE CODE n IS INVALID
ADM0185 E  SEGMENT ATTRIBUTE VALUE n IS INVALID
```

GSSAVE

Function

To save segments.

GSSAVE	**(count1, seg-array, name, count2, parm-array, count3, descriptor)**
APL code	592
GDDM RCP code	X'0C0C1200' (202117632)

Parameters

count1 *(specified by user) (fullword integer)*
The number of elements in the **seg-array** parameter.

seg-array *(specified by user) (an array of fullword integers)*
An array of segment identifiers. If the number of elements or the first segment identifier is zero, all the graphics data in the GDDM page is saved in the file specified in the **name** parameter. If the number of elements is greater than zero, each identified segment is saved in the named file. The segments are stored in the file in the order specified in **seg-array**. If this parameter has 0 after the first element, GDDM does not save any more elements beyond that point. Duplicate segment identifiers are not allowed.

name *(specified by user) (8-byte character string)*
The name (left-justified) that is to be given to the GDF object on auxiliary storage. This must be a valid external object name for the subsystem being used.

count2 *(specified by user) (fullword integer)*
The number of elements in the **parm-array** parameter.

parm-array *(specified by user) (an array of fullword integers)*
An array of control information. The parameter has two elements:

1–Overwrite control
Specifies whether the new GDF object can overwrite an existing object of the same name on auxiliary storage. Possible values are:

0 Overwrite existing file. This is the default.
1 Do not overwrite existing file.

2–Coordinate data type
Controls the type of coordinate data to be saved. Possible values are:

2 Save as 2-byte integers.
4 Save as 4-byte short floating-point values. This is the default.

count3 *(specified by user) (fullword integer)*
The number of characters specified in the **descriptor** parameter.

descriptor *(specified by user) (character)*
The descriptive record, of up to 253 bytes, that is saved with the picture.

Description

Saves segments, or all the graphics data in the current GDDM page, onto auxiliary storage. No segment must be open when the GSSAVE call is issued.

Graphics data format (GDF) objects are loaded with the GSLOAD call.

The segments or graphics data are saved in the GDDM segment library as a GDF object with the name specified in the GSSAVE call's **name** parameter. The saved GDF object contains descriptive information, segments and their attributes, and primitives and their attributes – that is, the same information that is returned to the application program by the GSGET call together with a list of all symbol sets loaded for the current device, whether the segments to be saved refer to them or not; see GSLOAD.

For restrictions on various devices, see "Device-specific saved pictures" in Chapter 4, "Device variations" on page 241.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0143 E  SEGMENT IDENTIFIER n IS DUPLICATE
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0161 E  GRAPHICS FIELD NOT DEFINED
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
ADM0324 E  FILE 'a' ALREADY EXISTS
```

GSSCLS

Function

To close the current segment.

GSSCLS	
APL code	507
GDDM RCP code	X'0C0C0301' (202113793)

Parameters

None.

Description

Causes the current segment to be closed, so that no more primitives can be added to it. After this operation, there is no current segment within the page. Closing a segment does not delete the segment or affect the graphics primitives that are displayed.

If an area is open, GSSCLS closes it as if a GSENDA call had been issued immediately before the GSSCLS call. All graphics attributes are reset to the values for the attributes that were in effect at the time the corresponding segment was created.

Note: It is important to ensure that an open segment is closed before outputting a picture with such calls as ASREAD, GSREAD, MSREAD, or FSFRCE. If this is not done, GDDM closes the open segment, but an unnecessary reshow can result if a GSSCLS call is issued subsequently.

Principal errors

```
ADM0149 E  NO CURRENT GRAPHICS SEGMENT
```

```
ADM0167 W  AREA DEFINITION NOT COMPLETED
```

GSSCPY

Function

To copy a segment.

GSSCPY (segment-id)	
APL code	633
GDDM RCP code	X'0C0C1400' (202118144)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment to be copied.

Description

Copies transformed primitives from the identified segment into the current stream of primitives. This can be either into the open segment or into the stream of primitives outside segments. The copied segment must not be open when the GSSCPY call is issued.

GSSCPY can be issued in the middle of an area, but the primitives that it copies must be valid within an area if GSSCPY is to complete without error.

Each copied primitive is transformed by the segment transform of the copied segment and it is displaced by an amount that brings the segment origin of the copied segment to the current position at the time the GSSCPY call was issued. If clipping is enabled (see GSCLP), the transformed and displaced primitives are clipped to the current viewport.

The primitives inherit the current primitive attributes at the time of the call. After GSSCPY, the current position and values of the primitive attributes are the same as they were before the call.

If the identified segment was created when clipping was enabled, the following current marker and character attributes are not inherited by any marker or character strings within the segment. Instead, the attributes that are used are those that were defaulted when the markers or characters were added to the identified segment.

The current primitive attributes that are **not** inherited are:

Character angle
Character box
Character direction
Character mode
Character set
Character shear.
Marker box
Marker scale
Marker symbol.

Compare the action of the GSSCPY call with that of GSSINC.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
```

GSSCT

Function

To set current transform.

GSSCT (sx, sy, hx, hy, rx, ry, dx, dy, type)	
APL code	651
GDDM RCP code	X'0C0C1107' (202117383)

Parameters

sx *(specified by user) (short floating point)*
sy *(specified by user) (short floating point)*
A scale transformation in terms of an x-axis scaling (**sx**) and a y-axis scaling (**sy**). The segment origin is used as a reference point: the axes that are used to scale are parallel to the x and y axes but pass through the segment origin. A scale factor in the range 0 through 1 shrinks primitives; a scale factor of greater than 1 stretches primitives. A negative scale factor reflects primitives about the other axis.

Specifying scale factors **sx=1** and **sy=1** does not perform any scaling. This setting can be used to suppress scaling (to allow a simple rotation, for example).

hx *(specified by user) (short floating point)*
hy *(specified by user) (short floating point)*
A shear transformation in terms of the displacements that a point on the y axis makes after shearing. The axes used for shearing are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used in the GSCH call for character shear. Note that primitives below the x axis are sheared in the opposite direction to those above the x axis. The points on the x axis itself are not moved. If **hx=a** and **hy=b** are used, an identical effect is achieved with **hx=–a** and **hy=–b**.

Specifying shear factors **hx=0** and **hy=1** does not perform any shearing. This setting can be used to suppress shearing (to allow a simple rotation, for example).

Specifying **hy=0** is not valid (because it would produce an infinite shear).

rx *(specified by user) (short floating point)*
ry *(specified by user) (short floating point)*
A rotation transformation in terms of the displacements that a point on the x axis makes after rotating. The axes used for rotating are parallel to the x and y axes, but pass through the current segment origin. This is similar to the method used for the character angle in the GSCA call.

Specifying rotation components **rx=1** and **ry=0** does not perform any rotation. This setting can be used to suppress rotation (to allow a simple scaling, for example).

Because two zero values would be ambiguous, specifying **rx=0** and **ry=0** is taken as equivalent to **rx=1** and **ry=0** (no rotation).

The (rx,ry) values below produce these special cases:

(0,0) no rotation; equivalent to (1,0)
(1,0) no rotation
(0,1) rotation by 90 degrees counterclockwise
(1,1) rotation by 45 degrees counterclockwise
(0,-1) rotation by 90 degrees clockwise
(–1,0) rotation by 180 degrees clockwise (or counterclockwise).

and, in general:

(rx,ry) rotation by theta degrees counterclockwise, where tan(theta)=ry/rx (assuming a uniform world-coordinate system).

dx *(specified by user) (short floating point)*
dy *(specified by user) (short floating point)*
A displacement of **dx** parallel to the x axis and **dy** parallel to the y axis. This transformation does not use the segment origin.

Specifying displacements components of **dx=0** and **dy=0** does not perform any displacement. This setting can be used to suppress displacements (to allow a simple rotation, for example).

type *(specified by user) (fullword integer)*
How the existing segment transform is to be modified by the scaling, shear, rotation, and displacement components specified. Possible values are:

0 New/replace
Any current transform previously defined is discarded and is replaced by the combined effect of the specified components.

1 Additive
The combined effect of the specified components is added to the effect already present due to previous current transforms. The new transform combines both effects in the order (i) old transform, and (ii) GSSCT parameter values. This option is the most useful for incremental updates to transforms.

2 Preemptive
The combined effect of the specified components is added to the effect of the existing current transform. The new transform combines both effects in the order (i) GSSCT parameter values, and (ii) the old transform. The effect is as if the GSSCT parameters modify the following primitives (without transformation) and the existing transformation is applied again.

Description

Sets the transform used for following primitives by specifying x-axis and y-axis scale factors, and shear, rotation, and displacement values. The parameters are specified in current window coordinates, as defined in the GSWIN call.

If the window coordinates are not uniform, graphics data appears to be sheared as it is rotated. For information on how to define window coordinates, see GSUWIN.

The combined effect of the scale, shear, rotation, and displacement specified is used to update the current primitive transformation. This is performed in three distinct ways, controlled by the **type** parameter. In all cases, the other parameters are taken as a unit, the effects being combined into a single composite transformation; that is, scale, shear, rotation, and displacement, in that order.

If precise clipping is enabled (GSCLP mode 1), primitives are clipped to the window **before** the current transform is applied. Therefore, it is possible for primitives that have been transformed to appear outside the viewport. They are clipped to the graphics field.

Note: The GSSCT call can only be applied when a segment is open; that is, after a GSSEG call and before a GSSCLS call.

Principal errors

```
ADM0149 E  NO CURRENT GRAPHICS SEGMENT
```

GSSDEL

Function

To delete a segment.

GSSDEL (segment-id)	
APL code	508
GDDM RCP code	X'0C0C0302' (202113794)

Parameters

segment-id *(specified by user) (fullword integer)*
Identifies the segment to be deleted.

Description

Causes the specified segment, together with any graphics primitives defined within it, to be deleted.

When a visible segment is deleted, the effect is apparent at the device only when the next device write takes place (that is, when the next ASREAD, FSFRCE, GSREAD, or MSREAD call is processed).

If the segment is current when it is deleted, there will be no current segment after this operation.

Principal errors

```
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
```

GSSEG

Function

To create a segment.

GSSEG	**(segment-id)**
APL code	509
GDDM RCP code	X'0C0C0300' (202113792)

Parameters

segment-id *(specified by user) (fullword integer)*
A number to be associated with the segment. The number must not be negative.

If this parameter is zero, the segment is not named and cannot be deleted explicitly. If the number is nonzero, it must be unique within the current page.

Description

Creates (opens) a segment with the specified identification number.

A segment consists of a group of graphics primitives that share the current viewport and window.

All current primitive attributes are set to the **drawing default** values when a segment is created. The current position is set to the world-coordinate origin.

The segment created becomes the current segment, to which subsequent primitives are added. The window and viewport cannot be changed while a segment is being constructed.

How to renumber a segment: The following example shows how to renumber an existing segment.

```
GSSEG (new)             Open the new segment
GSSINC (old)            Include the old segment
GSSCLS

GSQTFM (old,9,array)    Query the old segment transform
GSSTFM (new,9,array)    Set the new segment transform

GSQORG (old,xorg,yorg)  Query old segment origin
GSSORG (new,xorg,yorg)  Set new segment origin

do i = 1 to 6           Query all the segment
  GSQATS (old,i,value)  attributes and set the
  GSSATS (new,i,value)  new segment attributes
end

GSSPRI (new,old,1)      Make the new segment the
                        same as the old segment

GSSDEL (old)            Delete the old segment
```

The above code only renumbers an existing closed segment; it cannot renumber an open segment.

Also, this code cannot renumber references to a segment from other segments that use the GSCALL function.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0143 E  SEGMENT IDENTIFIER n IS DUPLICATE
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
```

GSSEN

Function

To set mixed string attribute of graphics text.

GSSEN	**(mixed)**
APL code	666
GDDM RCP code	X'0C0C1B00' (202119936)

Parameters

mixed *(specified by user) (fullword integer)*
The mixed string attribute state. Possible values are:
0 Mixed with position; the default.
1 Mixed with position.
2 Mixed without position.

Description

Specifies whether or not the mixed string in the graphics text is to take a one-byte position between single-byte characters and double-byte characters (DBCS, used for Kanji and Hangeul). The setting applies only to the current page and it is defaulted when the page is created.

An application program can present a mixed string using the GSCHAR call. In the mixed string, single-byte and double-byte characters are delimited by shift-out (SO) (X'0E') and shift-in (SI) (X'0F') control codes.

If “mixed with position” is specified by GSSEN, SO and SI control codes in the subsequent GSCHAR call take one byte position when the string is presented.

If “mixed without position” is specified by GSSEN, SO and SI control codes in the subsequent GSCHAR call take no position when the string is presented.

Principal errors

```
ADM3267 E  MIXED  CHARACTER  STRING  ATTRIBUTE  n  IS
           INVALID
```

GSSINC

Function

To include a segment.

GSSINC	**(segment-id)**
APL code	632
GDDM RCP code	X'0C0C1401' (202118145)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment to be included.

Description

Copies the primitives of the identified segment into the current stream of primitives. This can be either into the open segment or into the stream of primitives outside segments.

The copied segment must not be open when the GSSINC call is issued. GSSINC can be issued in the middle of an area, but the primitives that are copied must be valid within an area if GSSINC is to complete without error.

The effect of issuing a GSSINC call is to append the copied primitives to the current primitives, as if they had been drawn as part of the current primitive stream. This means that the copied primitives are drawn precisely where they were originally drawn (although they might be clipped differently). The current position and primitive attributes are inherited and are set by primitives from the copied segment. For a list of the attributes that are **not** inherited, see GSSCPY. GSSINC does not restore the attribute values.

Contrast the action of this call with that of GSSCPY.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0179 E  INVALID FUNCTION DURING GRAPHICS RETRIEVAL
```

GSSORG

Function

To set segment origin.

GSSORG	**(segment-id, x, y)**
APL code	587
GDDM RCP code	X'0C0C0311' (202113809)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment.
x *(specified by user) (short floating point)*
The x coordinate of the segment origin, specified in current world coordinates.
y *(specified by user) (short floating point)*
The y coordinate of the segment origin, specified in current world coordinates.

Description

Defines the segment origin of the identified segment. The identified segment need not be transformable. Setting a segment origin provides a reference point about which subsequent segment transformations will be performed. Note that GDDM does not provide a visible indication of the position of the segment origin.

When a segment is created, all primitives are drawn relative to the window origin identified by (0,0) in world coordinates. By default, the segment origin coincides with the window origin at the time the segment is created.

The GSSPOS call uses the segment origin as the reference point when it repositions the segment.

The GSSCPY call uses the segment origin to determine the placement of the copied segment.

The GSSAGA call uses the segment origin as the center point for rotate, shear, and scale. The results returned in the GSQAGA call also assume this point as the center of rotate, shear, and scale.

The GSQORG call returns the position of the segment origin in current world coordinates.

When a segment is used as an echo for a locator-logical-input device, the segment's origin establishes its position.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSSPOS

Function

To set segment position.

GSSPOS	**(segment-id, x, y)**
APL code	582
GDDM RCP code	X'0C0C030D' (202113805)

Parameters

segment-id *(specified by user) (fullword integer)*
The identifier of the segment to be repositioned. The segment type must be transformable; see GSSATI.

x *(specified by user) (short floating point)*

y *(specified by user) (short floating point)*
The coordinates in current window units to which the segment is to be moved.

Description

Displaces the identified segment and its segment origin.

By default, the segment origin coincides with the window origin at the time the segment was created. The segment origin can be changed by the GSSORG call or by previous GSSPOS calls.

When a segment is created, all primitives are drawn relative to the window origin identified by (0,0) in world coordinates.

When a segment is displaced by GSSPOS, the segment origin is moved to the point identified by the **x,y** parameters in the coordinate system currently in effect (as defined by GSWIN). The segment transform has a displacement added to it; the amount is the same as that by which the segment origin was displaced. This has the effect of displacing the segment. Repositioning a segment does not alter the size or aspect ratio of the image displayed.

The GSQPOS call returns the origin of a transformable segment in the current world coordinates.

If a line or other primitive contained in the segment is moved so that it is outside the boundary of the picture space, the results depend on the device. On the IBM 3279 display and similar devices, parts of vectors that are outside the graphics field boundary will not be visible.

The displacement of the segment from the local origin (0,0) is combined with the segment's transformable attribute; therefore, the new position affects subsequent GSQAGA and GSQTFM calls and the results of GSSAGA calls.

For more information, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSSPRI

Function

To set segment priority.

GSSPRI	**(seg-id, ref-seg-id, order)**
APL code	634
GDDM RCP code	X'0C0C0312' (202113810)

Parameters

seg-id *(specified by user) (fullword integer)*
The identifier of the segment whose priority is to be changed. The priority of segment 0 cannot be changed.

ref-seg-id *(specified by user) (fullword integer)*
The segment that identifies a position in the segment list. The segment specified in the **seg-id** parameter is drawn either immediately before or after this segment, depending on the value specified in the **order** parameter. Specifying 0 for **ref-seg-id** indicates that the position is to be the beginning or the end of the segment list, as defined by the value in the **order** parameter.

order *(specified by user) (fullword integer)*
Specifies whether the segment named in the **seg-id** parameter is to be drawn before or after the segment named in the **ref-seg-id** parameter.

Possible values are:

–1 The segment named in the **seg-id** parameter is to have a lower priority than the segment named in the **ref-seg-id** parameter. The **seg-id** segment is drawn before the **ref-seg-id** segment. If 0 is specified in the **ref-seg-id** parameter, the segment identified in the

seg-id parameter is placed as the highest priority segment.

1 The segment named in the **seg-id** parameter is to have a higher priority than the segment named in the **ref-seg-id** parameter. The **seg-id** segment is drawn after the **ref-seg-id** segment. If 0 is specified in the **ref-seg-id** parameter, the segment identified in the **seg-id** parameter is placed as the lowest priority segment.

Description

Changes the order in which segments are drawn and detected.

Segments are, by default, drawn in order of priority, with the lowest priority segment (the one that was created first) being drawn first. A new segment is given a higher priority than any existing segment and is added to the end of the list of segments to be drawn.

If primitives are drawn in overpaint mode, they appear on top of primitives in lower priority segments. Primitives in the higher priority segments are picked if they lie within the pick aperture.

The GSSPRI call changes the segment-drawing priority, the order in which segments are picked and, in general, it also changes the appearance of the picture.

The GSMIX call controls the mode of drawing (overpaint, mixed, or underpaint mode) for each primitive, but it does not alter the order in which segments are drawn or picked. GSSPRI can be used instead of GSMIX when the same mode of mixing (preferably overpaint mode, the default) is used throughout the range of segments. To overpaint a segment, it should have the highest priority; to underpaint it, a segment should have the lowest priority.

Primitives outside segments, and primitives in segment 0, are always added to the end of the list of segments current at the time they are created; both can be followed, and therefore over-painted, by subsequent named segments. Changing the priority of a segment may have the effect of redrawing the whole picture; therefore, primitives outside segments may be erased.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM3226 E  SEGMENT n IS UNKNOWN
ADM3227 E  REFERENCED SEGMENT n IS UNKNOWN
ADM3228 E  SEGMENT ORDERING n IS INVALID
```

GSSTFM

Function

To set segment transform.

(id, n, array, type)	
GSSTFM	
APL code	590
GDDM RCP code	X'0C0C1103' (202117379)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the segment.

n *(specified by user) (fullword integer)*
The number of elements to be used in the **array** parameter. If **n** is less than 9, the elements omitted default to the corresponding elements of the identity matrix (see above). Specifying **n=0** denotes that the identity matrix is used. Specifying six elements means that the last row is assumed to be (0 0 1).

array *(specified by user) (array of short floating-point numbers)*
The elements of the transformation matrix, in row order. Elements 7, 8, and 9 must be 0, 0, and 1.

type *(specified by user) (fullword integer)*
The way in which the segment is to be set. The options have exactly the same meaning as those in the **type** parameter of the GSSAGA call. Possible values are:

0 New/replace
The call redefines the segment transform; the previous transform is discarded.

1 Additive
The call defines a segment transform that is to be applied after the existing transform is applied to the segment. If **M** is the existing matrix and **N** is the matrix defined by the GSSTFM call, the resulting transformation matrix (**R**) is given by:

R = N • M

where "•" denotes matrix multiplication.

The effect is that the specified transform applies to the primitives as displayed, and the resulting transform combines all previous effects with the new one.

2 Preemptive
The call defines a segment transform that is to be applied before the existing transform is applied to the segment. If **M** is the existing matrix and **N** is the matrix defined by the GSSTFM call, the resulting transformation matrix (**R**) is given by:

R = M • N

where "•" denotes matrix multiplication.

The effect is that the specified transform applies to the original (untransformed) primitives **before** the existing transform is reapplied. The resulting transform summarizes these two effects as a transformation of the original primitives.

Description

Sets or modifies the segment transform.

Note: GSSTFM performs the same function as the GSSAGA call. However, GSSTFM uses a matrix form of the transform and therefore may not be suitable for many types of user.

GSSTFM specifies the transform as a one-dimensional array of n elements, being the first n elements of a 3-row by 3-column matrix ordered in rows. The order of the elements is as follows:

Matrix	Array
$\begin{bmatrix} a & b & c \\ d & e & f \\ 0 & 0 & 1 \end{bmatrix}$	(a,b,c,d,e,f,0,0,1)

The last row, if specified, must be (0,0,1). The transform acts on the coordinates of the primitives in a segment, so that a point with coordinates (x,y) is transformed to the point (a*x+b*y+c,d*x+e*y+f) before display, all expressed in world coordinates.

The initial value of the transform of a segment is the **identity matrix**:

Matrix	Array
$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	(1,0,0,0,1,0,0,0,1)

For more information on using these matrixes, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM0129 E  ARRAY COUNT n IS INVALID
ADM0140 E  SEGMENT IDENTIFIER n IS INVALID
ADM0145 E  SEGMENT n IS UNKNOWN
```

GSSVL

Function

To define segment viewing limits.

GSSVL	**(u1, u2, v1, v2)**
APL code	658
GDDM RCP code	X'0C0C1314' (202117908)

Parameters

u1 *(specified by user) (short floating point)*
u2 *(specified by user) (short floating point)*
The left- and right-hand extents of the segment viewing limits, in world coordinates.
v1 *(specified by user) (short floating point)*
v2 *(specified by user) (short floating point)*
The lower and upper extents of the segment viewing limits, in world coordinates.

Description

Explicitly defines the viewing limits of the current segment in world coordinates.

Viewing limits can be set within a segment as clipping extents for all subsequent primitives in the segment and any segments it calls. They can be changed at any time within the segment and they are not subject to segment transformations. Limits specified in called segment override those set by the limits of the root segment.

The default clipping limit is the graphics field for **all** clip modes.

The limits are reset to their default values by using zeros for **all** clip modes.

The parameters supplied are checked to ensure that, with respect to the direction of the coordinate system of the current graphics window, the right extent is greater than the left and that the upper extent is greater than the lower.

Principal errors

```
ADM0147 E  SEGMENT VIEWING LIMIT f IS INVALID
ADM3258 E  SEGMENT VIEWING LIMIT f1 NOT GREATER THAN f2
ADM3259 E  SEGMENT VIEWING LIMIT f1 NOT GREATER THAN f2
```

GSTA

Function

To set text alignment.

GSTA (horiz, vert)	
APL code	644
GDDM RCP code	X'0C0C130D' (202117901)

Parameters

horiz *(specified by user) (fullword integer)*
The horizontal alignment. Possible values are:

–1 Standard. The alignment assumed depends on the current character direction:

Left to right (0, 1)	Left edge of first character
Top to bottom (2)	Left edge of first character
Right to left (3)	Right edge of first character
Bottom to top (4)	Left edge of first character.

0 The drawing default (initially, –1).

1 Normal. The alignment assumed depends on the current character direction:

Left to right (0, 1)	Left
Top to bottom (2)	Center
Right to left (3)	Right
Bottom to top (4)	Center.

2 Left alignment. The string is aligned on the left edge of its leftmost character.

3 Center alignment. The string is aligned on the arithmetic mean of Left and Right.

4 Right alignment. The string is aligned on the right edge of its rightmost character.

vert *(specified by user) (fullword integer)*
The vertical alignment. Possible values are:

–1 Standard. The alignment assumed depends on the current character direction.

Left to right (0, 1)	Bottom edge of first character
Top to bottom (2)	Top edge of first character
Right to left (3)	Bottom edge of first character
Bottom to top (4)	Bottom edge of first character.

0 The drawing default (initially, –1).

1 Normal. The alignment assumed depends on the current character direction:

Left to right (0, 1)	Base
Top to bottom (2)	Top
Right to left (3)	Base
Bottom to top (4)	Base.

2 Top alignment. the string is aligned on the top edge of its topmost character.

3 Cap alignment. The string is aligned on the cap of its topmost character. Where Cap is not defined by the symbol set, this is the same as Top.

4 Half alignment. The string is aligned on the arithmetic mean of Base and Cap.

5 Base alignment. The string is aligned on the base of its bottom character. Where Base is not defined by the symbol set, this is the same as Bottom.

6 Bottom alignment. The string is aligned on the bottom edge of its bottom character.

Description

Sets the alignment, in the horizontal and vertical directions, of subsequently output text strings. The text alignment attributes remain in effect until they are changed by another GSTA call.

When a segment is created by the GSSEG call, the text alignment attribute is set to the drawing default value.

When a segment is closed by the GSSCLS call, the text alignment attribute is reset to the value that was in effect when the segment was created.

The parameters specify the alignment of character strings horizontally and vertically. Together they define a reference point within the string that is positioned on the starting point specified for the string.

Note: The terms “top left,” “bottom right,” and so on, are well defined when the character angle and the direction of the coordinate system are such that the baseline is parallel to the x axis, running from left to right on the display, and there is no character shear.

If the character is rotated or sheared, the term “top left” applies to the corner of the character box that appears in the top left when no rotation or shear is applied; see GSQTB.

For example, if the direction of the coordinate system is changed so that low x values lie on the right-hand side of the display, the term “top” applies to the side of the display corresponding to high y values, and “left” applies to the side of the display corresponding to low x values.

Principal errors

```
ADM0152 E  ATTRIBUTE VALUE n IS INVALID
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
```

GSTAG

Function

To set current primitive tag.

GSTAG (tag)	
APL code	566
GDDM RCP code	X'0C0C1000' (202117120)

Parameters

tag *(specified by user) (fullword integer)*
Contains a number that is used to identify subsequent primitives. 0 means that primitives are not named, are not detectable, and cannot be correlated.

Description

Specifies a tag by which the following primitives are to be known. When the operator picks a graphics object or uses the GSCORR call to locate an object, both the segment identifier and the primitive tag of the object picked are returned to the application program if the segment has been marked as detectable and the object has been assigned a tag.

If a tag of zero is specified, the primitives have no name and are not detectable by picking or by using the GSCORR call. Initially, the current tag is zero.

Primitives within segment zero cannot be picked or correlated, and any tag applied to them is ignored.

The tag remains in effect until it is changed by another GSTAG call.

When a segment is created by the GSSEG call, the tag is set to the drawing default value.

When a segment is closed by the GSSCLS call, the tag is reset to the value that was in effect when the segment was created.

The GSTAG call is not allowed between GSAREA and GSENDA calls; thus, all primitives within an area have the same tag.

Principal errors

```
ADM0158 E  INVALID FUNCTION IN AREA DEFINITION
ADM3200 E  TAG n IS INVALID
```

GSUWIN

Function

To define a uniform graphics window.

GSUWIN (x1, x2, y1, y2)	
APL code	584
GDDM RCP code	X'0C0C0007' (202113031)

Parameters

x1 *(specified by user) (short floating point)*
x2 *(specified by user) (short floating point)*
The graphics window coordinates of the x axis.
y1 *(specified by user) (short floating point)*
y2 *(specified by user) (short floating point)*
The graphics window coordinates of the y axis.

Description

Defines the graphics window such that either the x axis spans the entire width of the viewport and the y axis is within the height of the viewport, or that the y axis spans the entire height of the viewport and the x axis is within the width of the viewport. The graphics window is mapped to the current viewport such that one x-axis unit physically equals (that is, in terms of distance) one y-axis unit. If either axis is shorter than the width or height of the viewport, that axis is centered within the viewport.

GSQWIN returns the **actual** graphics window bounds (recalculated with either the x axis or y axis centered) **not** the values passed to the GSUWIN call.

GSUWIN allows an application program to ensure that a circle drawn with the GSARC call appears circular without having to compute a graphics window coordinate system of the same aspect ratio as the current viewport.

For information on where the character box at the current position appears if the graphics window is inverted, reversed, inverted and reversed, or in the normal position, see GSCHAR.

If clipping is enabled, primitives that are drawn outside the **actual** graphics window bounds are not visible. For more details, see GSWIN.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0151 E  WINDOW SPECIFICATION f1{, f2} IS INVALID
```

GSVECM

Function

To perform vector operations.

GSVECM (n, vector)	
APL code	531
GDDM RCP code	X'0C0C040A' (202114058)

Parameters

n *(specified by user) (fullword integer)*
The total number of lines or moves to be performed.

vector *(specified by user) (an array of fullword integers)*
An array of operations and coordinate pairs, in the order:

```
(ctrl1,x1,y1,ctrl2,x2,y2,...ctrli,xi,yi,...
                                ctrln,xn,yn)
```

where `ctrli` specifies whether a move is to be performed, or a line drawn, for the *ith* operation. The control values can be:

0 A move is to be performed to (xi,yi)
1 A line is to be drawn to (xi,yi).

Description

Performs a series of operations, each of which is either drawing a line or moving the current position to a specified end point.

If a line is drawn, it has the color, line width, and line type given by the current values of these attributes.

The current position is set to the last end point.

If any specified point lies outside the window boundaries, the line that ends at that specified point, any subsequent lines, and the current position is not defined. If clipping is enabled, only those sections of the lines within the current window are visible.

Principal errors

```
ADM0146 E  ARRAY COUNT n IS INVALID
ADM0153 E  CONTROL VALUE n IS INVALID
```

GSVIEW

Function

To define a viewport.

GSVIEW (x1, x2, y1, y2)	
APL code	504
GDDM RCP code	X'0C0C0003' (202113027)

Parameters

x1 *(specified by user) (short floating point)*
x2 *(specified by user) (short floating point)*
The left and right viewport boundaries in picture-space units. The number specified in **x1** must not exceed that in **x2**.

y1 *(specified by user) (short floating point)*
y2 *(specified by user) (short floating point)*
The lower and upper viewport boundaries in picture-space units. The number specified in **y1** must not exceed that in **y2**.

Description

Explicitly specifies the current viewport boundaries in picture-space units. A GSQPS call can be used to determine what the picture-space units are (they are device-dependent if defaulted). See Figure 12 on page 178. By default, the viewport takes up the whole of the picture space.

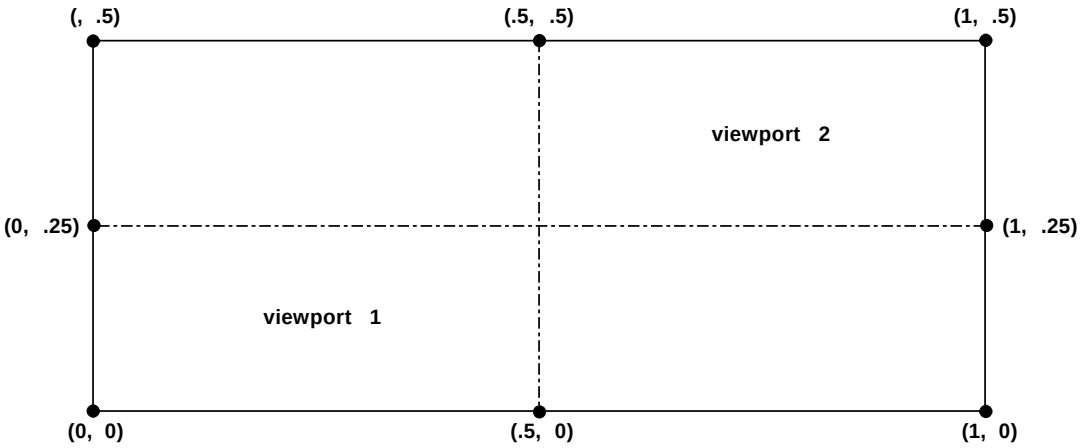
A viewport is a region of the total picture space. Viewports can be used to position the parts of a composite picture. The viewport boundaries are parallel to those of the picture space, and must be entirely within the defined or defaulted picture space.

When the picture space of a graphics field is defined or defaulted, the current viewport is set to cover the complete picture space. By **default**, therefore, the viewport covers the picture space.

Note: GSVIEW does not change the graphics window; see GSWIN.

Principal errors

```
ADM0142 E  VIEWPORT BOUNDARY f OUTSIDE PICTURE SPACE
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0165 E  VIEWPORT  UPPER  BOUNDARY  f1  LESS  THAN  OR
           EQUAL TO f2
ADM0166 E  VIEWPORT  RIGHT  BOUNDARY  f1  LESS  THAN  OR
           EQUAL TO f2
```



Boundary	Viewport 1	Viewport 2
Left	0.0	0.5
Right	0.5	1.0
Lower	0.0	0.25
Upper	0.25	0.5

Figure 12. Example of how a viewport is defined (GSVIEW)

GSWIN

Function

To define a graphics window.

GSWIN	**(u1, u2, v1, v2)**
APL code	505
GDDM RCP code	X'0C0C0002' (202113026)

Parameters

u1 *(specified by user) (short floating point)*
u2 *(specified by user) (short floating point)*
The left-hand and right-hand boundaries of the graphics window. The boundary specified by the **u1** value is mapped to the left-hand edge of the viewport, and the boundary specified by the **u2** value is mapped to the right-hand edge.
v1 *(specified by user) (short floating point)*
v2 *(specified by user) (short floating point)*
The lower and upper boundaries of the graphics window. The boundary specified by the **v1** value is mapped to the bottom of the viewport, and the boundary specified by the **v2** value is mapped to the top.

Description

Explicitly defines the graphics window that corresponds to the viewport. In other words, it specifies the coordinate system (world coordinates) to be used in the viewport.

The graphics window specification is used to position all graphics primitives on the current page until a new graphics window is specified. Before the first GSWIN call is issued, a **default** graphics window of (0,100,0,100) is in force. All graphics primitives in a segment must have the same graphics window.

For information on where the character box at the current position appears if the graphics window is inverted, reversed, inverted and reversed, or in the normal position, see GSCHAR.

If the mapping of the graphics window to the viewport is such that one x-axis unit does not physically equal one y-axis unit, the resultant pictures might appear “squashed” – for example, circles drawn using the GSARC call might appear elliptical.

If clipping is in effect, no part of a primitive (line, arc, character, or area) is visible outside the graphics window, with the exception of mode-1 and mode-2 symbols (see GSCHAR) and images (see GSIMG).

For another method of specifying graphics windows, see GSUWIN.

Note: Graphics defined at the boundaries of the graphics window may not appear.

Principal errors

```
ADM0150 E  GRAPHICS SEGMENT n IS CURRENT
ADM0151 E  WINDOW SPECIFICATION f1{, f2} IS INVALID
```

IMACLR

Function

To clear a rectangle in an image.

IMACLR	**(id, left-edge, right-edge, top-edge, bottom-edge)**
APL code	1604
GDDM RCP code	X'3C010008' (1006698504)

Parameters

id *(specified by user) (fullword integer)*
The image in which the rectangle is to be cleared.
left-edge *(specified by user) (fullword integer)*
right-edge *(specified by user) (fullword integer)*
The columns of pixels that form the left and right edges of the rectangle. The columns are included in the rectangle. The parameter **left-edge** must be in the range 0 through 2^{29}–2 and **right-edge** must be in the range –1 through 2^{29}–2.
top-edge *(specified by user) (fullword integer)*
bottom-edge *(specified by user) (fullword integer)*
The rows of pixels that form the top and bottom edges of the rectangle. The rows are included in the rectangle. The parameter **top-edge** must be in the range 0 through 2^{29}–2 and **bottom-edge** must be in the range –1 through 2^{29}–2.

Note: If **left-edge** is set to “n” and **right-edge** is set to “n–1,” zero width is implied, and, if **top-edge** is set to “n” and **bottom-edge** is set to “n–1,” zero depth is implied.

Description

Resets all pixels in the specified rectangle within the given image to their initial value (all zero for bi-level image).

This call can be used to clear a writable device image (for example, **id**=0), but not a read-only input device.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
ADM3361 E  IMAGE NOT WRITEABLE
ADM3362 E  INVALID RECTANGLE COORDINATE VALUE
ADM3363 W  RIGHT/BOTTOM EDGE EXCEEDS H-PIXELS/V-PIXELS
```

IMACRT

Function

To create an image.

IMACRT	**(id, h-pixels, v-pixels, im-type, res, res-unit, h-res, v-res)**
APL code	1601
GDDM RCP code	X'3C010001' (1006698497)

Parameters

id *(specified by user) (fullword integer)*
The new image. It must be –1, corresponding to a scanner, or greater than zero, corresponding to an application image, and must not be the identifier of an existing image. Zero is reserved for the current device image and cannot explicitly be created.

h-pixels *(specified by user) (fullword integer)*
The horizontal size of the image, in numbers of pixels. If zero is used, an image containing no pixels is created.

v-pixels *(specified by user) (fullword integer)*
The vertical size of the image, in numbers of pixels. If zero is used, an image containing no pixels is created.

im-type *(specified by user) (fullword integer)*
The type of image to be created.

- **0** Default, same as 1
- **1** Bi-level image (one bit per pixel).

res *(specified by user) (fullword integer)*
The resolution flag for the image. This specifies whether the image is to have a defined or undefined resolution. When an image has an undefined resolution, the values of the **h-res** and **v-res** fields of the image are ignored during image manipulations.

When a scanner image is created with an undefined resolution, subsequent scanning is set at the resolution currently set on the scanner, but the images resulting from this have undefined resolution.

Projections containing extracts in real units cannot be applied to images with undefined resolution; see IMREXR.

- **0** undefined resolution; that is, raw data. Image manipulations using this image are performed using a pixel-to-pixel mapping.
- **1** defined resolution; the image has a defined size in real units.

res-unit *(specified by user) (fullword integer)*
The units for the **h-res** and **v-res** parameters. The value of **res-unit** parameter is used only to interpret the **h-res** and **v-res** parameters, and is not subsequently associated with the image in any way.

0 inches
1 meters

h-res *(specified by user) (short floating point)*
v-res *(specified by user) (short floating point)*
The number of pixels in the horizontal or vertical direction per unit of measure defined by **res-unit**. When the **res** parameter is zero, the values of **h-res** and **v-res** are copied into the image, but they are not used in image manipulations until **res** is set to 1.

Values of 0 can be given for these parameters if **res** is 0, providing that **res** is not subsequently changed to 1 using IMARF. It is an error if an image with zero **h-res** and **v-res** is used in this way.

The values of **h-res** and **v-res** must be the same for any image that is to be retrieved by an IMAGT sequence (IMAGTS, IMAGT, IMAGTE) using PPF format and IBM 3800 compression.

Description

Creates an image of the specified horizontal and vertical size, type, and resolution and associates it with the specified identifier.

When created, images contain initial pixel values, defined to be zeros for bi-level image.

The identifier **id** can be any value that does not correspond to an already existing image. If the value of **id** has not been determined using IMAGID, it should be either –1, or a number in the range 1 through 2^{30}–1, so as not to conflict with values reserved by IMAGID, which are in the range 2^{30} through 2^{31}–1. If **id** was returned by IMAGID, it is no longer reserved after IMACRT.

The IMACRT call can be used to define the attributes of a scanner image by specifying **id** equal to –1. The image size usually specifies the paper size.

For a 3118 scanner, the image is defined to be centered over the vertical center line of the paper and aligned with the top edge. Note that the paper-feed mechanism of the 3118 scanner automatically aligns the paper over the center of the scanner sensor array.

For a 3117 scanner, the image is defined to be aligned on the top, and left edges of the scanner bed.

Principal errors

```
ADM3350 E  IMAGE n ALREADY EXISTS
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3352 E  IMAGE n1 HORIZONTAL SIZE  n2 IS INVALID
ADM3353 E  IMAGE n1 VERTICAL SIZE n2 IS INVALID
ADM3354 E  IMAGE n1 TYPE n2 IS INVALID
ADM3355 E  IMAGE n1 RESOLUTION FLAG n2 IS INVALID
ADM3356 E  IMAGE n1 RESOLUTION UNIT n2 IS INVALID
ADM3357 E  IMAGE n INVALID H-RES OR V-RES f FOR RES=1
ADM3470 E  SCANNER DOES NOT EXIST
ADM3474 E  SCANNER  DOES  NOT  SUPPORT  H-RES/V-RES  OF
           f1/f2
ADM3475 E  IMAGE  n1  UNSUPPORTED  HORIZONTAL  SIZE  n2
           PIXELS
ADM3476 E  IMAGE n1 UNSUPPORTED VERTICAL SIZE n2 PIXELS
```

IMADEL

Function

To delete the image associated with the identifier.

IMADEL (id)	
APL code	1603
GDDM RCP code	X'3C010007' (1006698503)

Parameters

id *(specified by user) (fullword integer)*
The image to be deleted.

Description

Deletes the image associated with the specified identifier. The identifier returns to its initial state, that is, before it was associated with the image.

When the **id** is –1 (the scanner), the paper is ejected, and the scanner is disconnected from the current device.

IMADEL cannot be used for image 0 (the current image field).

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
```

IMAGID

Function

To get and reserve a unique image identifier.

IMAGID (id)	
APL code	1600
GDDM RCP code	X'3C010002' (1006698498)

Parameters

id *(returned by GDDM) (fullword integer)*
An identifier for which no image exists and which is not reserved.

Description

Reserves a free image identifier with a value in the range 2^{30} through 2^{31}–1, and returns this value in **id**. A free identifier is one that does not currently have an image associated with it and is not currently reserved. The identifier remains reserved until an image is created specifying that identifier, or until an FSTERM call is made.

The IMAGID call can be used to obtain identifiers that do not conflict with those used by other parts of the application.

Identifiers in the range 2^{30} through 2^{31}–1 should only be used when obtained using the IMAGID call, as GDDM internally-generated images (such as for device emulation) also have identifiers in this range.

Principal errors

None.

IMAGT

Function

To retrieve image data from an image.

IMAGT	**(id, buffer-length, buffer, data-length)**
APL code	1613
GDDM RCP code	X'3C010015' (1006698517)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the image from which data is to be retrieved. This must be the same as on a previous call to IMAGTS.

buffer-length *(specified by user) (fullword integer)*
The number of bytes of **buffer** that are available to receive the image data. This must be at least 80 bytes, except for 3800 compression, when a minimum of 400 bytes is recommended.

buffer *(returned by GDDM) (character)*
A data area, of at least the specified length, to receive the image data.

data-length *(returned by GDDM) (fullword integer)*
The length of image data placed in the buffer by GDDM. If it is zero, all the image data has been returned.

Description

Retrieves image data from the specified image, and places it into a buffer. The image data is returned in the format and compression specified on the call to IMAGTS, which must previously have been issued to start image retrieval from the specified image.

IMAGTx sequences are transfer operations; for more information, see the *GDDM Base Application Programming Guide*.

More than one IMAGT call is usually needed to retrieve all of the image data. If this is the case, the result depends on the format in use:

For unformatted and 3193 data stream format, **buffer-length** bytes of data are placed in each buffer except the last, **data-length** being set to the value of **buffer-length**. The last buffer contains the remaining data, **data-length** being set to indicate the actual number of bytes placed in the buffer; that is, a number in the range 1 through **buffer-length**. The remainder of the last buffer is filled with zeros. Subsequent calls to IMAGT give a **data-length** of zero, indicating “end of data,” in which case the contents of the buffer are unaltered.

The data returned by IMAGT for unformatted and 3193 data-stream format can be entered into another image using the IMAPT call with the same value of **buffer-length**. It is not necessary to use a short length for the final buffer.

Note: It is possible to use an IMAGT call to retrieve data directly from a scanner if specific restrictions are observed; this can reduce the amount of host processing time required. For more information, see the *GDDM Base Application Programming Guide*.

For page printer format, less than **buffer-length** bytes of data are usually placed in each buffer, **data-length** being set to indicate the actual number of bytes placed in the buffer; that is, a value in the range 1 through **buffer-length**. The remainder of the buffer is filled with zeros. When all the data has been returned, calls to IMAGT give a **data-length** of 0, indicating “end of data,” in which case, the contents of the buffer are unaltered.

Note: If the compression on the preceding IMAGTS call is specified as 3800, an image with an ***h-pixels*** value greater than 8000 pixels is truncated to 8000 pixels. A warning message is issued.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3379 E  INVALID BUFFER LENGTH
ADM3384 E  NO PRECEDING IMAGTS CALL
ADM3385 E  BUFFER LENGTH TOO SMALL
```

IMAGTE

Function

To end retrieval of data from an image.

IMAGTE (id)	
APL code	1614
GDDM RCP code	X'3C010016' (1006698518)

Parameters

id *(specified by user) (fullword integer)*
Specifies the identifier of the image for which retrieval is ended. This must be the same as on a previous call to IMAGTS.

Description

Ends the retrieval of image data from the specified image.

The function can be called whether or not all image data has been retrieved by calls to IMAGT.

IMAGTx sequences are transfer operations; for more information, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3384 E  NO PRECEDING IMAGTS CALL
```

IMAGTS

Function

To start retrieval of data from an image.

IMAGTS (id, proj-id, format, compression)	
APL code	1612
GDDM RCP code	X'3C010014' (1006698516)

Parameters

id *(specified by user) (fullword integer)*
The application or device image for which retrieval is started. Device image identifiers are allowed, as follows:
–**1** Scanner input from the current device.
0 The image on the current GDDM page.
>**0** Application images.

proj-id *(specified by user) (fullword integer)*
The projection to be applied.

format *(specified by user) (fullword integer)*
The format of the image. Possible values are:
0 Default (same as 2).
1 Unformatted.
2 3193 data stream format.
3 Page printer format.

The following values of **format** exist to allow the entry of image data coming from a source that defines **1=black**; that is, the opposite of that defined by GDDM. This feature is provided because the supported formats do not uniquely define which values map to black and which to white. If a later release of GDDM supports a level of 3193 data-stream format or page printer format that does allow polarity within the format, the sign of the operand in existing application programs will be ignored.
–**1** Unformatted, where a pixel value of 1 indicates black.
–**2** 3193 data stream formats, where a pixel value of 1 indicates black.
–**3** Page printer formats, where a pixel value of 1 indicates black.

compression *(specified by user) (fullword integer)*
The compression type of the image. Possible values are:
0 Default (same as 1).
1 Uncompressed.
2 MMR (IBM 8815).
3 IBM 4250.
4 IBM 3800.

Only specific combinations of **format** and **compression** are allowed; these are defined as follows:

		1 Unformatted	2 3193DSF	3 PPF
1	**Uncompressed**	√	√	
2	**MMR (IBM 8815)**	√	√	√
3	**IBM 4250**			√
4	**IBM 3800**			√

Where:

MMR = modified-modified read
3193DSF = 3193 data stream format
PPF = page printer format.

Description

Starts a “get” sequence, that is, the retrieval of image data from a specified image. The call is followed by one or more calls to IMAGT, specifying the same image. The image data is retrieved in the specified format and compression. The retrieval process must be ended by a call to IMAGTE.

IMAGTx sequences are transfer operations and therefore a projection is applied to the data; for more information, see the *GDDM Base Application Programming Guide*.

While retrieval is in progress, other images can be accessed, but the specified image can only be accessed by IMAGT or IMAGTE, or it can be deleted by IMADEL.

When the source of the transfer operation is a scanner, an implicit FSFRCE is performed to ensure that the screen contents are up to date and ready for any echoing that may be done. All I/O is performed for the current page. If there is no image field in the current page, one is created. All echoing is to the current image field. The effect of echoing is such that the scanned image is transferred to the image field with the same projection; that is, as if the application had issued:

```
IMXFER (-1,0,proj-id)
```

followed by FSFRCE. Whenever possible, the echoing is performed by the device. The ISCTL values are used to determine the required quality of echoing, and thus whether echoing is performed by the device, or by GDDM; for more information, see the *GDDM Base Application Programming Guide*.

It is not possible to start an IMAGT sequence from image –1 while an IMAPT sequence to the display (image 0) is in progress.

Principal errors

```
ADM3366 E  INVALID SOURCE IMAGE IDENTIFIER
ADM3367 E  SOURCE IMAGE DOES NOT EXIST
ADM3368 E  SOURCE IMAGE NOT READABLE
ADM3369 E  PROJECTION USES INCHES/METERS FOR SOURCE
           WITH NO RESOLUTION
ADM3373 E  INVALID IMAGE FORMAT
ADM3374 E  INVALID IMAGE COMPRESSION
ADM3375 E  COMPRESSION n1 IS NOT VALID FOR FORMAT n2
ADM3378 E  IMAGE ENTRY OR RETRIEVAL ALREADY INITIALIZED
           FOR IMAGE n
ADM3388 E  IMAGE NOT SUITABLE FOR GET SEQUENCE IN
           REQUESTED FORMAT
ADM3390 E  IMAGE (AFTER PROJECTION APPLIED) UNSUITABLE
           FOR REQUESTED FORMAT
ADM3391 W  ONLY n1 OUT OF n2 TRANSFORMS WILL BE
           PROCESSED
ADM3392 I  OVERLAPPED TARGET RECTANGLES MAY GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3403 W  NO IMAGE DATA TRANSFERRED
ADM3477 E  SCANNER NOT READY. MAY BE SWITCHED OFF
ADM3478 E  NO PAPER IN SCANNER
ADM3479 E  SCANNER LAMP INTENSITY IS TOO LOW
ADM3480 E  SCANNER PAPER JAM
ADM3481 E  UNRECOVERABLE SCANNER ERROR OCCURRED
ADM3482 E  PROJECTION WOULD REQUIRE SCANNER TO RESCAN
ADM3483 E  SCANNER DISCONNECTED
ADM3484 W  GRAY-SCALE TRANSFORMS MAY BE SIMPLIFIED TO
           MEET SCANNER CAPABILITIES
```

IMAPT

Function

To enter data into an image.

IMAPT	**(id, buffer-length, buffer)**
APL code	1610
GDDM RCP code	X'3C010012' (1006698514)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the image into which the image data is to be entered. This must be the same as on a previous call to IMAPTS.

If the image does not exist, and the data is formatted or compressed, the image is created implicitly from the data presented on the first call to IMAPT for the image. If such an image is not completely filled by subsequent calls to IMAPT, it is an error and the incomplete image is deleted.

If the value of **id** was not obtained by means of an IMAGID call, it should be in the range 0 through 2^{30}–1, so as not to conflict with the values reserved by IMAGID, which are in the range 2^{30} through 2^{31}–1. If **id** was returned by IMAGID, it is no longer preserved after a call to IMAPTS.

buffer-length *(specified by user) (fullword integer)*
The number of bytes of **buffer** containing the image data. In non-XA environments, the buffer must not be greater than 32Kb.

buffer *(specified by user) (character)*
A data area of at least the specified length containing all or part of the image data, in the format specified by the call to IMAPTS for the image.

Description

Enters image data into the specified image.

The image data must be in the format and compression specified on the call to IMAPTS, which must have been made before this call to start image data entry to the specified image.

Enough calls to IMAPT should be issued until the image is filled. On the call to IMAPT that contains the last part of the image, any data from the final pixel (or for formatted or MMR compressed data, the end of the last structure) to the end of the buffer is ignored, and no error message is issued. Subsequent IMAPT calls in this sequence are an error.

Note: For formatted data, IMAPTx sequences are transfer operations; for more information, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM3370 E  INVALID TARGET IMAGE IDENTIFIER
ADM3371 E  TARGET IMAGE NOT WRITEABLE
ADM3372 E  PROJECTION USES IMRPLR WITH FRACTIONAL
           COORDINATES. TARGET MUST EXIST
ADM3379 E  INVALID BUFFER LENGTH
ADM3380 E  INVALID DATA FOR SPECIFIED FORMAT OR
           COMPRESSION
ADM3381 E  NO PRECEDING IMAPTS CALL
ADM3382 W  IMAGE IS FULL
ADM3387 E  PROJECTION USES INCHES/METERS FOR TARGET
           WITH NO RESOLUTION
ADM3391 W  ONLY n1 OUT OF n2 TRANSFORMS WILL BE
           PROCESSED
ADM3392 I  OVERLAPPED TARGET RECTANGLES MAY GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3402 E  PROJECTION DOES NOT EXIST
```

IMAPTE

Function

To end data entry into an image.

IMAPTE	**(id)**
APL code	1611
GDDM RCP code	X'3C010013' (1006698515)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the image for which entry is ended. This must be the same as on a previous call to IMAPTS.

Description

Ends entry of image data to the specified image.

If the function is called before the image has been filled by calls to IMAPT, a warning message is issued and the remainder of the image is undefined.

Note: For formatted data, IMAPTx sequences are transfer operations; for more information, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM3370 E  INVALID TARGET IMAGE IDENTIFIER
ADM3371 E  TARGET IMAGE NOT WRITEABLE
ADM3372 E  PROJECTION USES IMRPLR WITH FRACTIONAL
           COORDINATES. TARGET MUST EXIST
ADM3381 E  NO PRECEDING IMAPTS CALL
ADM3383 W  IMAGE NOT FILLED
ADM3387 E  PROJECTION USES INCHES/METERS FOR TARGET
           WITH NO RESOLUTION
ADM3389 E  FORMATTED OR COMPRESSED DATA INCOMPLETE. PUT
           SEQUENCE CANCELED
ADM3391 W  ONLY n1 OUT OF n2 TRANSFORMS WILL BE
           PROCESSED
ADM3392 I  OVERLAPPED TARGET RECTANGLES MAY GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3403 W  NO IMAGE DATA TRANSFERRED
```

IMAPTS

Function

To start data entry into an image.

IMAPTS	**(id, proj-id, format, compression)**
APL code	1609
GDDM RCP code	X'3C010011' (1006698513)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the image that receives the data. Output device images can be specified; that is, zero, which identifies the image on the current GDDM page. In this case the projection may be evaluated in the device.

If the image does not exist, and the data is formatted or compressed, the image is created implicitly from the data presented on the first call or calls to IMAPT for the image. If such an image is not completely filled by subsequent calls to IMAPT, it is an error and the incomplete image is deleted. For unformatted-uncompressed data, the image must exist before the IMAPTS call is performed.

If the value of **id** was not obtained using an IMAGID call, it should be in the range 0 through $2^{30}-1$, so as not to conflict with values reserved by IMAGID, which are in the range 2^{30} through $2^{31}-1$. If **id** was returned by IMAGID, it is no longer reserved after IMAPTS.

proj-id *(specified by user) (fullword integer)*
A projection to be applied. For unformatted data that is not MMR-compressed, only 0 (the identity projection) is allowed.

format *(specified by user) (fullword integer)*
The format of the image. Possible values are:

- **0** Default (same as 2).
- **1** Unformatted.
- **2** 3193 data stream format.
- **3** Page printer format, or 8815 data stream format.

The following values of **format** exist to allow the entry of image data coming from a source that defines **1=black**; that is, the opposite of that defined by GDDM. This feature is provided because the supported formats do not uniquely define which values map to black and which to white. If a later release of GDDM supports a level of 3193 data-stream format or page printer format that does allow polarity within the format, the sign of the operand in existing application programs will be ignored.

–**1** Unformatted, where a pixel value of 1 indicates black.
–**2** 3193 data stream formats, where a pixel value of 1 indicates black.
–**3** Page printer formats, where a pixel value of 1 indicates

Only specific combinations of **format** and **compression** are allowed; for further information, see the description of the IMAGTS call.

compression *(specified by user) (fullword integer)*
The compression type of the image. Possible values are:

0 For formatted data, the compression is to be determined by inspection of the data. For unformatted data (that is, **format=1**), this is the same as option 1.
1 Uncompressed.
2 MMR (IBM 8815).
3 IBM 4250.
4 IBM 3800.

Only specific combinations of **format** and **compression** are allowed; for further information, see the description of the IMAGTS call.

Description

Starts a “put” sequence; that is, it initializes the process of data entry to a specified image.

The call is followed by one or more calls to IMAPT, specifying the same image and giving data in the specified format and compression. The process must be ended by a call to IMAPTE.

For formatted data and unformatted-MMR data, IMAPTx sequences are transfer operations, with all the usual rules applying; for more information, see the *GDDM Base Application Programming Guide*.

For data that is unformatted and uncompressed, a **proj-id** of zero must be specified.

While scanning is in progress, and an IMAGT sequence from image –1 is being performed, it is not possible to start a IMAPT sequence to the display (image 0). The scanner echo facility (ISESCA) provides this function and ensures that the maximum amount of processing is off-loaded to the display and scanner, within the quality requirements set by ISCTL and ISXCTL.

Principal errors

```
ADM3370 E  INVALID TARGET IMAGE IDENTIFIER
ADM3373 E  INVALID IMAGE FORMAT
ADM3374 E  INVALID IMAGE COMPRESSION
ADM3375 E  COMPRESSION n1 IS NOT VALID FOR FORMAT n2
ADM3376 E  PROJECTION-ID MUST BE ZERO FOR UNFORMATTED
           DATA
ADM3377 E  IMAGE DOES NOT EXIST. FORMATTED DATA
           REQUIRED
ADM3378 E  IMAGE ENTRY OR RETRIEVAL ALREADY INITIALIZED
           FOR IMAGE n
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
```

IMAQRY

Function

To query attributes of an image.

IMAQRY	**(id, h-pixels, v-pixels, im-type, res, res-unit, h-res, v-res)**
APL code	1619
GDDM RCP code	X'3C010004' (1006698500)

Parameters

id *(specified by user) (fullword integer)*
The image whose attributes are to be returned. If a device image is specified, (for example, **id=0**), the attributes of this are returned.
h-pixels *(returned by GDDM) (fullword integer)*
The horizontal size of the image, in numbers of pixels.
v-pixels *(returned by GDDM) (fullword integer)*
The vertical size of the image, in numbers of pixels.
im-type *(returned by GDDM) (fullword integer)*
The data type of pixels in the image. Only bi-level is currently supported.
1 Bi-level image, (one bit per pixel).
res *(returned by GDDM) (fullword integer)*
Indicates whether or not the image has a defined resolution. Possible values are:
0 undefined resolution; that is, raw data.
Image manipulations using this image are performed using a pixel-to-pixel mapping.
1 defined resolution.
The image has a defined size in real units.
res-unit *(specified by user) (fullword integer)*
The units in which to return the **h-res** and and **v-res** values. This need not be the same as the value supplied on the IMACRT call that created the image.
0 inches.
1 meters.

h-res *(returned by GDDM) (short floating point)*
v-res *(returned by GDDM) (short floating point)*
The number of pixels in the horizontal or vertical direction per unit of measure defined by **res-unit**. When **res** is zero, the values of **h-res** and **v-res** that are returned are the most recent values set, although if **res** is zero, these values will not have been used in image manipulations.

Description

Returns the current attributes of the specified image. These values reflect any changes resulting from operations that alter image attributes.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3356 E  IMAGE n1 RESOLUTION UNIT n2 IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
```

IMARES

Function

To convert the resolution attributes of an image.

IMARES	**(id, res-unit, h-res, v-res, alg)**
APL code	1602
GDDM RCP code	X'3C010006' (1006698502)

Parameters

id *(specified by user) (fullword integer)*
The image for resolution conversion. This must not be zero (image on the current GDDM page) but it can be –1.
res-unit *(specified by user) (fullword integer)*
The units for the **h-res** and **v-res** parameters. Possible values are:
0 Inches.
1 Meters.
h-res *(specified by user) (short floating point)*
v-res *(specified by user) (short floating point)*
The number of pixels in the horizontal or vertical direction per unit of measure defined by **res-unit**.
alg *(specified by user) (fullword integer)*
The resolution-conversion/scaling algorithm to be used when resolution conversion is performed on the image. Possible values are:
0 Default, same as 1.
1 Pixel replication.
Pixels are replicated when new pixels are required on scaling up, and are deleted when scaling down.
2 Black pixel retention.
Pixels are replicated when new pixels are required on scaling up, but when scaling down white pixels are deleted in preference to adjacent black pixels. This algorithm is an improvement over pixel replication for images containing black on white text or graphics.
3 White pixel retention.
Pixels are replicated when new pixels are required on scaling up, but when scaling down black pixels are deleted in preference to adjacent white pixels. This algorithm is an improvement over pixel replication for images containing white on black text or graphics.

Description

Converts the resolution attributes of the specified image to the values given.

If the image has undefined resolution, that is the **res** field of the image has the value 0, the effect of IMARES is to change only the **h-res** and **v-res** fields of the image, while the image data is not changed. The **alg** parameter is ignored.

If the image has defined resolution, that is, the **res** field of the image has the value 1, the effect of IMARES is to scale the image data and modify the **h-res** and **v-res** fields by the corresponding amount. The effect is that the size of the image in real units is unchanged, but that the **h-pixels** and **v-pixels** are scaled by the ratios of the new and old **h-res** and **v-res** values.

A scanner image can only have defined a resolution that it supports. The supported resolutions can be queried with the ISQRES call. If a scanner image is specified, the algorithm parameter of IMARES is ignored, the next image being scanned at the new resolution.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3356 E  IMAGE n1 RESOLUTION UNIT n2 IS INVALID
ADM3357 E  IMAGE n INVALID H-RES OR V-RES f FOR RES=1
ADM3358 E  IMAGE n DOES NOT EXIST
ADM3360 E  INVALID RESOLUTION/SCALING ALGORITHM
ADM3474 E  SCANNER  DOES  NOT  SUPPORT  H-RES/V-RES  OF
           f1/f2
```

IMARF

Function

To change resolution flag of an image.

IMARF	**(id, flag)**
APL code	1620
GDDM RCP code	X'3C01000C' (1006698508)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the image. Zero (the image on the current GDDM page) and –1 (for the 3117 or 3118 scanner) can be specified.
flag *(specified by user) (fullword integer)*
Specifies the new value for the **res** field of the image. Possible values are:
0 Undefined resolution.
1 Defined resolution.
If this is specified, the **h-res** and **v-res** attributes of the image must have valid values. Negative and zero values are not allowed.

Description

Changes an image from undefined-resolution state (that is, raw data) to defined-resolution state, and the converse.
If an image has a **res** attribute of 1, it has defined resolution. If an image has a **res** attribute of 0, it has undefined resolution.

When an image has defined resolution, the **h-res** and **v-res** fields of the image specify the number of pixels per unit length, and hence define real dimensions of the image.
When an image has undefined resolution, the **h-res** and **v-res** fields of the image are not used by GDDM. In this state, GDDM does not alter the values of **h-res** and **v-res**, except when requested using the IMARES call.

A scanner image can only have a defined resolution that it supports. The supported resolutions can be queried with the ISQRES call.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3355 E  IMAGE n1 RESOLUTION FLAG n2 IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
ADM3359 E  IMAGE n HAS INVALID H-RES/V-RES
ADM3471 E  SCANNER DOES NOT SUPPORT AN IMAGE n PIXELS
           {DEEP|WIDE}
ADM3472 E  SCANNER   DOES   NOT   SUPPORT   SPECIFIED
           H-RES/V-RES
```

IMARST

Function

To restore image from auxiliary storage.

IMARST	**(id, proj-id, name, count, descr)**
APL code	1608
GDDM RCP code	X'3C01000B' (1006698507)

Parameters

id *(specified by user) (fullword integer)*
The image to be loaded. A device image can be specified (for example, **id**=0) provided it is writable, in which case the projection can be evaluated in the device.
proj-id *(specified by user) (fullword integer)*
The projection to be applied.
name *(specified by user) (8-byte character string)*
The name (left-justified) of the image to be loaded from auxiliary storage.
count *(specified by user) (fullword integer)*
The number of characters in the **descr** parameter to be used to receive the descriptive record. This must not exceed 253.
descr *(returned by GDDM) (character)*
A character string, of at least **count** bytes, the first **count** of which are to receive, left justified, the descriptive record that was saved with the image. If the length of the descriptor is greater than **count**, it is truncated on the right. If it is less, it is padded on the right with blanks.

Description

Restores an image from the GDDM object library. IMARST is a transfer operation call, and therefore, if the image does not exist before the IMARST call, a new one is created. If the image does exist before the IMARST call, the incoming data is merged with the existing image, according to the usual transfer operation rules.

If the value of **id** is explicitly given, it should be in the range 0 through 2^{30}–1, so as not to conflict with values reserved by IMAGID, which are in the range 2^{30} through 2^{31}–1. If **id** was returned by IMAGID, it is not reserved after a call to IMARST.

Principal errors

```
ADM0307 E  FILE 'a' NOT FOUND
ADM0313 E  FILE 'a' HAS INVALID RECORD CONTENT
ADM3364 E  INVALID DESCRIPTOR COUNT VALUE
ADM3370 E  INVALID TARGET IMAGE IDENTIFIER
ADM3371 E  TARGET IMAGE NOT WRITEABLE
ADM3372 E  PROJECTION  USES  IMRPLR  WITH  FRACTIONAL
           COORDINATES. TARGET MUST EXIST
ADM3387 E  PROJECTION  USES  INCHES/METERS  FOR  TARGET
           WITH NO RESOLUTION
ADM3391 W  ONLY  n1  OUT  OF  n2  TRANSFORMS  WILL  BE
           PROCESSED
ADM3392 I  OVERLAPPED  TARGET  RECTANGLES  MAY  GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3403 W  NO IMAGE DATA TRANSFERRED
```

IMASAV

Function

To save image on auxiliary storage.

IMASAV	(id, proj-id, name, count, descr, protect-flag)
APL code	1607
GDDM RCP code	X'3C01000A' (1006698506)

Parameters

id *(specified by user) (fullword integer)*
The image to be saved. A device image can be specified if it is readable.

proj-id *(specified by user) (fullword integer)*
The projection to be applied.

name *(specified by user) (8-byte character string)*
The name (left-justified) to be given to the image object on auxiliary storage. This must be a valid external object name for the subsystem being used; see Chapter 11, “Image file formats” on page 311.

count *(specified by user) (fullword integer)*
The number of characters in the **descr** parameter to be saved. This must not exceed 253.

descr *(specified by user) (character)*
A character string, of at least **count** bytes, the first **count** bytes of which are to be saved with the image as a descriptive record.

protect-flag *(specified by user) (fullword integer)*
Specifies whether or not to protect an existing file from being overwritten. Possible values are:

0 Do not protect an existing file (allow overwriting).

1 Protect an existing file (do not allow overwriting).

Description

Saves the specified image on auxiliary storage, and also a description of up to 253 characters. The image is saved in the GDDM object library, with the specified name. The image is unchanged by this operation.

Principal errors

```
ADM0324 E  FILE 'a' ALREADY EXISTS
ADM3364 E  INVALID DESCRIPTOR COUNT VALUE
ADM3365 E  INVALID PROTECT-FLAG VALUE
ADM3366 E  INVALID SOURCE IMAGE IDENTIFIER
ADM3367 E  SOURCE IMAGE DOES NOT EXIST
ADM3368 E  SOURCE IMAGE NOT READABLE
ADM3369 E  PROJECTION USES INCHES/METERS FOR SOURCE
           WITH NO RESOLUTION
ADM3391 W  ONLY n1 OUT OF n2 TRANSFORMS WILL BE
           PROCESSED
ADM3392 I  OVERLAPPED TARGET RECTANGLES MAY GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3403 W  NO IMAGE DATA TRANSFERRED
ADM3477 E  SCANNER NOT READY. MAY BE SWITCHED OFF
ADM3478 E  NO PAPER IN SCANNER
ADM3479 E  SCANNER LAMP INTENSITY IS TOO LOW
ADM3480 E  SCANNER PAPER JAM
ADM3481 E  UNRECOVERABLE SCANNER ERROR OCCURRED
ADM3482 E  PROJECTION WOULD REQUIRE SCANNER TO RESCAN
ADM3483 E  SCANNER DISCONNECTED
ADM3484 W  GRAY-SCALE TRANSFORMS MAY BE SIMPLIFIED TO
           MEET SCANNER CAPABILITIES
```

IMATRM

Function

To trim an image down to the specified rectangle.

IMATRM	(id, left-edge, right-edge, top-edge, bottom-edge)
APL code	1605
GDDM RCP code	X'3C010009' (1006698505)

Parameters

id *(specified by user) (fullword integer)*
The image to be trimmed.

left-edge *(specified by user) (fullword integer)*

right-edge *(specified by user) (fullword integer)*
The columns of pixels that form the left and right edges of the rectangle. The columns are included in the rectangle. The **left-edge** parameter must be in the range 0 through $2^{29}-2$ and **right-edge** must be in the range –1 through $2^{29}-2$.

top-edge *(specified by user) (fullword integer)*

bottom-edge *(specified by user) (fullword integer)*
The rows of pixels that form the top and bottom edges of the rectangle. The rows are included in the rectangle. The **top-edge** parameter must be in the range 0 through $2^{29}-2$ and **bottom-edge** must be in the range –1 through $2^{29}-2$.

Note: If **left-edge** is set to “n” and **right-edge** is set to “n–1,” zero width is implied, and, if **top-edge** is set to “n” and **bottom-edge** is set to “n–1,” zero depth is implied.

Description

Trims the specified image to the size of the rectangle defined by the parameters **left-edge**, **right-edge**, **top-edge**, and **bottom-edge**. All data lying outside the rectangle is lost. The h-pixels and v-pixels attributes of the image are altered to reflect the change in size. (It follows, therefore, that IMATRM is not the complement of IMACLR).

The IMATRM call can be used with a writeable device image, in which case all pixels outside the rectangle are reset to their default value (all zero for bi-level image). However, the **h-pixels** and **v-pixels** values of the image are not altered by the call.

The IMATRM call can be used with a 3117 or 3118 scanner (image –1).

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
ADM3361 E  IMAGE NOT WRITEABLE
ADM3362 E  INVALID RECTANGLE COORDINATE VALUE
ADM3363 W  RIGHT/BOTTOM EDGE EXCEEDS H-PIXELS/V-PIXELS
```

IMPCRT

Function

To create an empty projection.

IMPCRT	**(proj-id)**
APL code	1650
GDDM RCP code	X'3C030003' (1006829571)

Parameters

proj-id *(specified by user) (fullword integer)*
The projection to be created. Proj-id 0, the identity projection, cannot be specified.

Description

Creates an empty projection with the specified identifier, to which transforms can be added. If a projection to which no transforms have been added is used on a transfer, no data is transferred, and the source and target images are unchanged.

The **proj-id** parameter can be any value that does not correspond to an already existing projection.

If the value of **proj-id** was not obtained using an IMPGID call, it should be in the range 0 through $2^{30}-1$, so as not to conflict with values reserved by IMPGID, which are in the range 2^{30} through $2^{31}-1$; see IMPGID.

If **proj-id** was returned by IMPGID, it is no longer reserved after an IMPCRT call.

Principal errors

```
ADM3400 E  PROJECTION ALREADY EXISTS
ADM3401 E  INVALID PROJECTION IDENTIFIER
```

IMPDEL

Function

To delete projection.

IMPDEL	**(proj-id)**
APL code	1652
GDDM RCP code	X'3C030004' (1006829572)

Parameters

proj-id *(specified by user) (fullword integer)*
The projection to be deleted. Proj-id 0, the identity projection, cannot be deleted.

Description

Deletes the projection associated with the given identifier. The identifier returns to its initial state, that is, its state before it was associated with the projection.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
```

IMPGID

Function

To get and reserve a unique projection identifier.

IMPGID	**(proj-id)**
APL code	1651
GDDM RCP code	X'3C030001' (1006829569)

Parameters

proj-id *(returned by GDDM) (fullword integer)*
An identifier for which no projection exists and which is not reserved.

Description

Reserves a free projection identifier with a value in the range 2^{30} through $2^{31}-1$, and returns this value in **proj-id**. A free identifier is one that does not currently have a projection associated with it and is not currently reserved. The identifier remains reserved until an image is created specifying that identifier, or until a call to FSTERM.

The IMPGID call can be used to obtain identifiers that do not conflict with those used by other parts of the application.

Projection identifiers in the range 2^{30} through $2^{31}-1$ should only be used when obtained using the IMPGID call, as GDDM internally-generated projections (such as for device emulation) also have projection identifiers in this range.

Principal errors

None.

IMPRST

Function

To restore projection from auxiliary storage.

IMPRST	**(proj-id, name, count, descr)**
APL code	1654
GDDM RCP code	X'3C030006' (1006829574)

Parameters

proj-id *(specified by user) (fullword integer)*
An identifier for the projection being restored. Proj-id 0, the identity projection, cannot be specified.

name *(specified by user) (8-byte character string)*
The name (left-justified) of the projection to be restored from auxiliary storage.

count *(specified by user) (fullword integer)*
The number of characters in the **descr** parameter to be used to receive the descriptive record. This must not exceed 253.

descr *(returned by GDDM) (character)*
A character string, of at least **count** bytes, the first **count** of which are to receive, left justified, the descriptive record that was saved with the projection. If the length of the descriptor is greater than **count**, it is truncated on the right. If it is less, it is padded on the right with blanks.

Description

Restores a projection from the GDDM object library, and loads it with the specified projection identifier. If a projection with that identifier already exists, it is replaced by the incoming projection. If the projection does not exist, a new one is created. The projection on auxiliary storage is unchanged by this operation.

If the value of **proj-id** was not obtained using an IMPGID call, it should be in the range 0 through $2^{30}-1$, so as not to conflict with values reserved by IMPGID, which are in the range 2^{30} through $2^{31}-1$.

If **proj-id** was returned by IMPGID, it is no longer reserved after an IMPRST call.

Principal errors

```
ADM0307 E  FILE 'a' NOT FOUND
ADM0313 E  FILE 'a' HAS INVALID RECORD CONTENT
ADM3364 E  INVALID DESCRIPTOR COUNT VALUE
ADM3402 E  PROJECTION DOES NOT EXIST
```

IMPSAV

Function

To save projection on auxiliary storage.

IMPSAV	**(proj-id, name, count, descr, protect-flag)**
APL code	1653
GDDM RCP code	X'3C030005' (1006829573)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of the projection to be saved. The projection is unchanged as a result of this operation. Proj-id 0, the identity projection, cannot be specified.

name *(specified by user) (8-byte character string)*
The name (left-justified) of the projection to be saved to auxiliary storage. This must be a valid external object name for the subsystem being used; see Chapter 11, “Image file formats” on page 311.

count *(specified by user) (fullword integer)*
The number of characters in the **descr** parameter to be saved. This must not exceed 253.

descr *(specified by user) (character)*
A character string, of at least **count** bytes, the first **count** bytes of which are to be saved with the projection as a descriptive record.

protect-flag *(specified by user) (fullword integer)*
Specifies whether or not to protect an existing file from being overwritten. Possible values are:
0 Do not protect an existing file (allow overwriting).

1 Protect an existing file (do not allow overwriting).

Description

Saves the specified projection on auxiliary storage, together with a description of up to 253 characters. The projection is saved as a GDDM projection in the GDDM object library, with the specified name. The projection is unchanged by this operation.

Principal errors

```
ADM0324 E  FILE 'a' ALREADY EXISTS
ADM3364 E  INVALID DESCRIPTOR COUNT VALUE
ADM3365 E  INVALID PROTECT-FLAG VALUE
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
```

IMRBRI

Function

To define brightness conversion algorithm.

IMRBRI	**(proj-id, alg, count, alg-data)**
APL code	1665
GDDM RCP code	X'3C030202' (1006830082)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of a projection with an incomplete transform, to which this transform element is to be added.

alg *(specified by user) (fullword integer)*
The algorithm to be used for brightness conversion. Possible values are:

0 The default algorithm; this is device-dependent. For scanners this is the same as 1.

1 A simple linear brightness conversion.

```
new = old  + ( ALG–DATA(1) * white)
```

where `white` is the maximum gray level, and `ALG-DATA(1)` specifies the required brightness as a number in the range –1.0 through +1.0 such that:

–1.0	Totally dark,
0.0	No change,
1.0	Totally light.

The image scanners provide three brightness values, the one used depends on the value of `ALG-DATA(1)` as follows:

–1.0 through –0.5	Light original; darken the image.
>–0.5 through <0.5	Normal original; no change.
0.5 through 1.0	Dark original; lighten the image.

count *(specified by user) (fullword integer)*
The number of parameters specified in the **alg-data** array.

alg-data *(specified by user) (array of short floating-point numbers)*
An array of numbers that are the data required by the specified algorithm. For information, see the description of the **alg** parameter.

Description

Defines the brightness conversion algorithm to be applied to a gray-scale sub-image.

The IMRBRI call has no effect if the sub-image is bi-level.

Only one IMRBRI call is allowed for a given transform; subsequent IMRBRI calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3413 E  INVALID ALGORITHM
ADM3414 E  INVALID COUNT
ADM3415 E  INVALID ALGORITHM DATA VALUE
```

IMRCON

Function

To define contrast conversion algorithm.

IMRCON	**(proj-id, alg, count, alg-data)**
APL code	1666
GDDM RCP code	X'3C030203' (1006830083)

Parameters

proj-id *(specified by user) (fullword integer)*
A projection with an incomplete transform, to which this transform element is to be added.

alg *(specified by user) (fullword integer)*
The algorithm to be used for contrast conversion. Possible values are:

0 The default algorithm; this is device-dependent. For scanners this is the same as 1.

1 A simple linear contrast conversion.

```
new=((old–mean) * alg-data(1))+mean.
```

where `mean` is the mid-point between black and white,

and `alg-data(1)` specifies the required contrast as a positive number such that:

0.5 is half the contrast,
1.0 is normal contrast,
2.0 is double the contrast, and so on.

The image scanners provide three contrast values; the one used depends on the value of `alg-data(1)` as follows:

0.0 through 0.5	Decreased contrast
>**0.5 through** <**2.0**	No change
≥**2.0**	Increased contrast.

count *(specified by user) (fullword integer)*
The number of parameters specified in the **alg-data** array.

alg-data *(specified by user) (array of short floating-point numbers)*
An array of numbers that are the data required by the specified algorithm. For information, see the description of the **alg** parameter.

Description

Defines the contrast conversion algorithm to be applied to a gray-scale sub-image.

The IMRCON call has no effect if the sub-image is bi-level.

Only one IMRCON call is allowed for a given transform; subsequent IMRCON calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3413 E  INVALID ALGORITHM
ADM3414 E  INVALID COUNT
ADM3415 E  INVALID ALGORITHM DATA VALUE
```

IMRCVB

Function

To define bi-level conversion algorithm.

IMRCVB	**(proj-id, alg, count, alg-data)**
APL code	1664
GDDM RCP code	X'3C030201' (1006830081)

Parameters

proj-id *(specified by user) (fullword integer)*
A projection with an incomplete transform, to which this transform element is to be added.

alg *(specified by user) (fullword integer)*
The algorithm to be used for conversion to bi-level. Possible values are:

0 The default algorithm; this is device-dependent. For image scanners this is the same as 1.

Note: When writing device-independent code, it is recommended that a ***count*** parameter of zero is used when specifying the default algorithm.

1 Threshold.
A threshold is defined for comparison with each source pixel. Pixels above the threshold specified by **alg-data(1)** become white and below it become black.

alg-data(1) specifies the required threshold in the range 0.0 through 1.0 where 0.0 is black and 1.0 is white. The default threshold is 0.5.

The image scanners provide three threshold levels, the level used depends on the value of **alg-data(1)** as follows:

0.0 through 0.25	Dark original
>**0.25 through** <**0.75**	Normal original
0.75 through 1.0	Light original.

10 Halftoning type A
This is best for intricate pictures.

The **alg-data** array is not used.

11 Halftoning type B
This is best for images in which the gray levels vary gradually.

The **alg-data** array is not used.

12 Compressed data stream
This is best for pure gray-scale documents such as photographs.

The **alg-data** array is not used.

count *(specified by user) (fullword integer)*
The number of parameters defined in the **alg-data** array.

alg-data *(specified by user) (array of short floating-point numbers)*
An array of numbers that are the data required by the specified algorithm. For information, see the **alg** parameter description.

Description

Defines the algorithm to be used during conversion to bi-level in a sub-image. “Thresholding” or “halftoning” are the two types of conversion allowed.

The result of this call is a bi-level sub-image, and so any subsequent gray-scale transform element calls have no effect.

The IMRCVB call has no effect if the sub-image is bi-level.

Only one IMRCVB call is allowed for a given transform; subsequent IMRCVB calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3413 E  INVALID ALGORITHM
ADM3414 E  INVALID COUNT
ADM3415 E  INVALID ALGORITHM DATA VALUE
```

IMREX

Function

To define rectangular sub-image in pixel coordinates.

IMREX	**(proj-id, left-edge, right-edge, top-edge, bottom-edge)**
APL code	1655
GDDM RCP code	X'3C030101' (1006829825)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of a projection to which this transform element is to be added.

left-edge *(specified by user) (fullword integer)*

right-edge *(specified by user) (fullword integer)*
The columns of pixels that form the left and right edges of the rectangle to be extracted. The **left-edge** parameter must be in the range 0 through $2^{29}-2$ and **right-edge** must be in the range –1 through $2^{29}-2$.

top-edge *(specified by user) (fullword integer)*

bottom-edge *(specified by user) (fullword integer)*
The rows of pixels that form the top and bottom edges of the rectangle to be extracted. The **top-edge** parameter must be in the range 0 through $2^{29}-2$ and **bottom-edge** must be in the range –1 through $2^{29}-2$.

Description

Selects a rectangular sub-image, in pixel coordinates. IMREX is a normal transform element call except that when it is used it must be the first transform element in the transform, and it cannot be used in the same transform as an IMREXR call.

For a description of transforms and projections, see the *GDDM Base Application Programming Guide*.

Transforms contain transform element calls (IMREX, IMREXR, IMRSCL, IMRORN, IMRREF, IMRNEG, IMRBRI, IMRCON, and IMRCVB), and are started by the first such call. They are completed by an IMRPL or IMRPLR call, at which time they become available for transfer using the specified projection.

If any part of the specified rectangle falls outside the source image, that part is filled with default values (zeros for bi-level image).

Principal errors

```
ADM3362 E  INVALID RECTANGLE COORDINATE VALUE
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3405 E  IMREX, WHEN CALLED, MUST BE FIRST CALL IN A
           TRANSFORM
```

IMREXR

Function

To define rectangular sub-image in real coordinates.

IMREXR	**(proj-id, coord-type, left-edge, right-edge, top-edge, bottom-edge)**
APL code	1656
GDDM RCP code	X'3C030102' (1006829826)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of the projection to which this transform element is to be added.

coord-type *(specified by user) (fullword integer)*
The coordinate type of the rectangle defined by the following **edge** parameters:

0 Inches
The edges of the rectangle are defined in inches. For this coordinate type, the source image must have a defined resolution.

Note: Negative edge coordinates are not allowed.

1 Meters
The edges of the rectangle are defined in meters. For this coordinate type, the source image must have a defined resolution.

Note: Negative edge coordinates are not allowed.

2 Fractional
The edges of the rectangle are defined as fractions of the **h-pixels** and **v-pixels** dimensions of the source image in the range 0.0 through 1.0.

left-edge *(specified by user) (short floating point)*
right-edge *(specified by user) (short floating point)*
The left and right edges of the rectangle as a distance from the left edge of the image in the units specified by **coord-type**. The values are first converted to a floating-point value in the pixel coordinate range, then **left-edge** is rounded up to the nearest integer and the **right-edge** rounded down, to address the image.

top-edge *(specified by user) (short floating point)*
bottom-edge *(specified by user) (short floating point)*
The top and bottom edges of the rectangle as a distance from the top edge of the image, in the units specified by **coord-type**. The values are first converted to a floating-point value in the pixel coordinate range, then **top-edge** is rounded up to the nearest integer and the **bottom-edge** rounded down, to address the image.

Description

Selects a rectangular subimage, in real coordinates. IMREXR is a normal transform element call except that when it is used it must be the first transform element in the transform, and it cannot be used in the same transform as an IMREX call.

For a description of transforms and projections, see the *GDDM Base Application Programming Guide*.

Transforms contain transform element calls (IMREX, IMREXR, IMRSCL, IMRORN, IMRREF, IMRNEG, IMRBRI, IMRCON, and IMRCVB), are started by the first such call. They are completed by an IMRPL or IMRPLR call, at which time they become available for transfer using the specified projection.

If any part of the specified rectangle falls outside the source image, that part is filled with default values (zeros for bi-level image).

Principal errors

```
ADM3362 E  INVALID RECTANGLE COORDINATE VALUE
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3406 E  IMREXR, WHEN CALLED, MUST BE FIRST CALL IN A
           TRANSFORM
ADM3407 E  INVALID COORD-TYPE VALUE
```

IMRNEG

Function

To negate the pixels of an extracted image.

IMRNEG (proj-id)	
APL code	1663
GDDM RCP code	X'3C030109' (1006829833)

Parameters

proj-id *(specified by user) (fullword integer)*
A projection with an incomplete transform, to which this transform element is to be added.

Description

Negates (inverts) the pixels in an extracted image (for bi-level images, white becomes black and black becomes white). The effect of this call is to convert an image into its photographic negative.

Only one IMRNEG call is allowed for a given transform. Subsequent IMRNEG calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
```

IMRORN

Function

To turn an extracted image clockwise through a number of right angles.

IMRORN (proj-id, orientation)	
APL code	1661
GDDM RCP code	X'3C030107' (1006829831)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of a projection with an incomplete transform, to which this transform element is to be added.

orientation *(specified by user) (fullword integer)*
The number of right angles through which the extracted image is rotated, in the clockwise direction. Possible values are:

0 No rotation.
1 Rotation by 90 degrees clockwise.
2 Rotation by 180 degrees (clockwise or counterclockwise).
3 Rotation by 270 degrees clockwise (90 degrees counterclockwise).

Description

This call rotates an extracted image clockwise through a specified number of right angles.

Only one IMRORN call is allowed for a given transform. Subsequent IMRORN calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3411 E  INVALID ORIENTATION VALUE
```

IMRPL

Function

To define place position in pixel coordinates.

IMRPL	**(proj-id, h-pos, v-pos, mix)**
APL code	1657
GDDM RCP code	X'3C030103' (1006829827)

Parameters

proj-id *(specified by user) (fullword integer)*
The projection to which the transform completed by this call is to be added.

h-pos *(specified by user) (fullword integer)*
v-pos *(specified by user) (fullword integer)*
Where the top-left corner of the transformed image is to be placed in the target image. Negative coordinates are not allowed.

mix *(specified by user) (fullword integer)*
The mode for mixing the pixels of the transformed image into the target image:

0 The default value: same as 1.
1 "Overpaint" mode.
Sets each pixel in the target image to the value of the corresponding pixel in the extracted image.
2 "Merge" mode.
Performs a logical "Or" on each pixel in the target image with the corresponding pixel in the extracted image.
3 "Difference" mode.
Performs a logical "Exclusive-Or" on each pixel in the target image with the corresponding pixel in the extracted image.
4 "And" mode.
Performs a logical "And" on each pixel in the target image with the corresponding pixel in the extracted image.
5 "Subtract" mode.
Performs a logical "And" on each pixel in the target image with the complement of the corresponding pixel in the extracted image.

Description

Completes a transform, defines the position in the target image in pixel coordinates at which the transformed image is placed on a transfer operation, and specifies the mixing algorithm to be used when merging the transformed image into the target image. This call completes a transform and makes it available for use in transfer operations using the specified projection.

If any part of the transformed image falls outside of the target image, clipping occurs such that the image data that falls outside the image is discarded.

Note: See also IMRPLR.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3408 E  INVALID H-POS OR V-POS VALUE
ADM3409 E  INVALID MIX OPTION VALUE
```

IMRPLR

Function

To define place position in real coordinates.

IMRPLR	**(proj-id, coord-type, h-pos, v-pos, mix)**
APL code	1658
GDDM RCP code	X'3C030204' (1006830084)

Parameters

proj-id *(specified by user) (fullword integer)*
The projection to which the transform completed by this call is to be added.

coord-type *(specified by user) (fullword integer)*
The coordinate type of the following **h-pos** and **v-pos** parameters. Possible values are:

0 inches
The location is defined in inches. Negative **h-pos** and **v-pos** parameters are not allowed. For this coordinate type, the target image must have a defined resolution, or be a new target (that is, one that does not exist prior to the transfer operation). A new target inherits the resolution of the first transformed image.

1 meters
The location is defined in meters. Negative **h-pos** and **v-pos** parameters are not allowed. For this coordinate type, the target image must have a defined resolution, or be a new target (that is, one that does not exist prior to the transfer operation). A new target inherits the resolution of the first transformed image.

2 fractional
The location is defined as a fraction of the h-pixels and v-pixels dimensions of the target image in the range 0.0 through 1.0. For this coordinate type, the target image must exist prior to the transfer operation.

h-pos *(specified by user) (short floating point)*
v-pos *(specified by user) (short floating point)*
Where the top-left corner of the transformed image is to be placed in the target image, in the units specified by the **coord-type** parameter. The values are first converted to a floating-point value in the pixel coordinate range, then rounded up to the nearest integer to address the image.

mix *(specified by user) (fullword integer)*
The mode for mixing the pixels of the transformed image into the target image. Possible values are:

0 The default value; same as 1.

1 “Overpaint” mode.
Sets each pixel in the target image to the value of the corresponding pixel in the extracted image.

2 “Merge” mode.
Performs a logical “Or” on each pixel in the target image with the corresponding pixel in the extracted image.

3 “Difference” mode.
Performs a logical “Exclusive-Or” on each pixel in the target image with the corresponding pixel in the extracted image.

4 “And” mode.
Performs a logical “And” on each pixel in the target image with the corresponding pixel in the extracted image.

5 “Subtract” mode.
Performs a logical “And” on each pixel in the target image with the complement of the corresponding pixel in the extracted image.

Description

Completes a transform, defines the position in the target image in real coordinates at which the transformed image is placed on a transfer operation, and specifies the mixing algorithm to be used when merging the transformed image into the target image. This call completes a transform and makes it available for use in transfer operations using the specified projection.

If any part of the transformed image falls outside of the target image, clipping occurs such that the image data that falls outside the image is discarded.

Note: See also IMRPL.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3407 E  INVALID COORD-TYPE VALUE
ADM3408 E  INVALID H-POS OR V-POS VALUE
ADM3409 E  INVALID MIX OPTION VALUE
```

IMRRAL

Function

To set current resolution/scaling algorithm.

IMRRAL	**(proj-id, res-alg)**
APL code	1660
GDDM RCP code	X'3C030106' (1006829830)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of a projection with an incomplete transform, to which this transform element is to be added.

res-alg *(specified by user) (fullword integer)*
The resolution-conversion/scaling algorithm to be used when scaling or resolution conversion is performed on the extracted image. Possible values are:

0 Default, same as 1.

1 Pixel replication.
Pixels are replicated when new pixels are required on scaling up, and are deleted when scaling down.

2 Black pixel retention.
Pixels are replicated when new pixels are required on scaling up, but when scaling down, white pixels are deleted in preference to adjacent black pixels. This algorithm is an improvement over pixel replication for images containing black on white text or graphics.

3 White pixel retention.
Pixels are replicated when new pixels are required on scaling up, but when scaling down, black pixels are deleted in preference to adjacent white pixels. This algorithm is an improvement over pixel replication for images containing white on black text or graphics.

Description

Defines the resolution/scaling algorithm to be used for any resolution conversion or scaling operation performed in the transform in which the IMRRAL call appears. For transforms in which no algorithm is specified, scaling and resolution conversion are performed using the default algorithm.

Only one IMRRAL call is allowed for a given transform. Subsequent IMRRAL calls are an error.

Principal errors

```
ADM3360 E  INVALID RESOLUTION/SCALING ALGORITHM
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
```

IMRREF

Function

To reflect extracted image.

IMRREF	**(proj-id, reflection)**
APL code	1662
GDDM RCP code	X'3C030108' (1006829832)

Parameters

proj-id *(specified by user) (fullword integer)*
The identifier of a projection with an incomplete transform, to which this transform element is to be added.

reflection *(specified by user) (fullword integer)*
Specifies how the extracted image is to be reflected. Possible values are:

1 Left-to-right (horizontal)
2 Top-to-bottom (vertical)
3 Top-to-left (major diagonal)
4 Right-to-top (minor diagonal)

Description

Reflects the extracted image.

Only one IMRREF call is allowed for a given transform. Subsequent IMRREF calls are an error.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3412 E  INVALID REFLECTION VALUE
```

IMRSCL

Function

To scale extracted image.

IMRSCL	**(proj-id, h-scale, v-scale)**
APL code	1659
GDDM RCP code	X'3C030105' (1006829829)

Parameters

proj-id *(specified by user) (fullword integer)*
A projection with an incomplete transform, to which this transform element is to be added.

h-scale *(specified by user) (short floating point)*
The horizontal scale factor by which the **h-pixels** value of the extracted image are to be multiplied. Negative values are not allowed.

v-scale *(specified by user) (short floating point)*
The vertical scale factor by which the **v-pixels** value of the extracted image are to be multiplied. Negative values are not allowed.

Description

Scales the extracted image in the horizontal direction, or vertical direction, or both of these. The scaling factors **h-scale** and **v-scale** are multipliers for **h-pixels** and **v-pixels**.

Note: Negative **h-scale** and **v-scale** values are not allowed, and cause an error message to be issued.

The parameters **h-pixels** and **v-pixels** for the extracted image become **h-pixels*h-scale** and **v-pixels*v-scale**, rounded to the nearest integer.

Only one IMRSCL call is allowed for a given transform. Subsequent IMRSCL calls are an error.

The scaling factors **h-scale** and **v-scale** refer to the extracted image after it has been operated on by any calls that were specified earlier. Therefore, a rotation by 90 degrees followed by a scale with an **h-scale** value of 2 and a **v-scale** value of 1 corresponds to a scale with an **h-scale** value of 1 and a **v-scale** value of 2 followed by a rotation by 90 degrees.

The scaling algorithm to be used can be specified with the IMRRAL call.

Principal errors

```
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3404 E  TRANSFORM ALREADY CONTAINS A CALL TO THIS
           ROUTINE
ADM3410 E  INVALID H-SCALE OR V-SCALE VALUE
```

IMXFER

Function

To transfer data between two images, applying a projection.

IMXFER	**(source-id, target-id, proj-id)**
APL code	1615
GDDM RCP code	X'3C010017' (1006698519)

Parameters

source-id *(specified by user) (fullword integer)*
The source image. Device images can be specified; for example:
- **–1** Identifies the scanner attached to the current primary device.
- **0** Identifies the image on the current GDDM page.
- **>0** Identifies application images.

target-id *(specified by user) (fullword integer)*
The image into which the results of the transfer are placed. Device images can be specified; for example:
- **0** Specifies the image on the current GDDM page, in which case the projection may be evaluated in the device.
- **>0** Specifies application images.

proj-id *(specified by user) (fullword integer)*
The projection used to extract, transform and place image data.
- **0** Specifies the identity projection, which results in the entire source image being copied into the target, with its top left corner at the origin with the right and bottom edges clipped if necessary.
- **>0** Specifies other projections.

Description

Applies a projection to a source image and places the result(s) in the target image. IMXFER is the explicit form of the transfer operation. The following description summarizes the main aspects of this operation.

During a transfer operation, images are extracted from the source image using the definitions in the projection. These extracted images are transformed by the projection, and the results are placed in the target image. Their positioning within the target image is also governed by the projection.

For more information, see the *GDDM Base Application Programming Guide*.

If the target image does not exist before the call, one is created with the following attributes: **h-pixels** and **v-pixels** are the minimum required to contain the right edge of the right-most transformed image, and the bottom-edge of the bottom-most image, **im-type**, **res**, **h-res**, and **v-res**, are those derived from the source image after applying the transform elements.

If the target image already exists, the transformed images are merged into it, using the mixing algorithm(s) specified in the projection.

Resolution conversion of the incoming data take places according to the following rules:

- If the source and target both have a defined resolution (that is, the **res** attribute of the source and target images are both 1) then resolution conversion is performed on the incoming image data, using the resolution/scaling algorithm specified in the projection.
- If either the source or target have undefined resolution (that is, the **res** attribute of either the source or target image is 0) then resolution conversion is not performed, and the transformed data is merged using pixel-to-pixel mapping.

The source and target can be the same image, but all extractions are performed before any data is placed back into the image.

The source or target image can be device images, with the limitation that write-only images cannot be used as sources and that read-only images cannot be used as targets.

IMXFER may cause I/O when device images are referenced. It can be used to transfer image data from the attached scanner by specifying –1 as the source identifier. I/O always occurs when the source is a scanner.

Note that the scanner image must have been created using IMACRT before IMXFER is used. More than one IMXFER call can be issued for a particular sheet of paper (image) in the scanner, if they do not require the scanner to “back up.” When all transfers for a sheet are complete, the IMADEL call is used to cause the sheet to be ejected ready for the next sheet, or ISLDE can be used to load the next sheet.

Whenever an IMXFER from a scanner is performed, an image field is required on the current page. If one does not already exist, a default image field is created. If the image field create fails, it causes the IMXFER to fail.

When the source of the transfer operation is a scanner, an implicit FSFRCE is performed to ensure that the contents of the screen (to which the scanner is connected) is up-to-date and ready for any echoing that may be done. All I/O is performed for the current page. If there is no image field in the current page, one is created. All echoing is to the current image field. The effect of echoing is that the scanned image is transferred to the image field with the same projection; that is, as if the application had issued:

```
IMXFER (–1,0,proj-id)
```

followed by FSFRCE. Whenever possible, the echoing is performed by the device. The ISCTL values are used to determine the required quality of echoing, and thus whether echoing is performed by the device, or by GDDM; for more information, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM3366 E  INVALID SOURCE IMAGE IDENTIFIER
ADM3367 E  SOURCE IMAGE DOES NOT EXIST
ADM3368 E  SOURCE IMAGE NOT READABLE
ADM3369 E  PROJECTION USES INCHES/METERS FOR SOURCE
           WITH NO RESOLUTION
ADM3370 E  INVALID TARGET IMAGE IDENTIFIER
ADM3371 E  TARGET IMAGE NOT WRITEABLE
ADM3372 E  PROJECTION USES IMRPLR WITH FRACTIONAL
           COORDINATES. TARGET MUST EXIST
ADM3387 E  PROJECTION USES INCHES/METERS FOR TARGET
           WITH NO RESOLUTION
ADM3401 E  INVALID PROJECTION IDENTIFIER
ADM3402 E  PROJECTION DOES NOT EXIST
ADM3403 W  NO IMAGE DATA TRANSFERRED
ADM3391 W  ONLY n1 OUT OF n2 TRANSFORMS WILL BE
           PROCESSED
ADM3392 I  OVERLAPPED TARGET RECTANGLES MAY GIVE
           INCORRECT RESULTS
ADM3393 I  SCALE FACTOR n1 APPROXIMATED TO n2
ADM3394 I  SCALING ALGORITHM n1 APPROXIMATED TO n2
ADM3477 E  SCANNER NOT READY. MAY BE SWITCHED OFF
ADM3478 E  NO PAPER IN SCANNER
ADM3479 E  SCANNER LAMP INTENSITY IS TOO LOW
ADM3480 E  SCANNER PAPER JAM
ADM3481 E  UNRECOVERABLE SCANNER ERROR OCCURRED
ADM3482 E  PROJECTION WOULD REQUIRE SCANNER TO RESCAN
ADM3483 E  SCANNER DISCONNECTED
ADM3484 W  GRAY-SCALE TRANSFORMS MAY BE SIMPLIFIED TO
           MEET SCANNER CAPABILITIES
```

ISCTL

Function

To set image quality-control parameters.

ISCTL	**(device, quality)**
APL code	182
GDDM RCP code	X'0C300002' (204472322)

Parameters

device *(specified by user) (fullword integer)*
The image to which the call applies. Possible values are:

–**1** Scanner

0 Current page.

quality *(specified by user) (fullword integer)*
The limit(s) of approximation allowed for transforms.

0 Default, the same as 3.

n A number in the range 1 through 5, that has this meaning:

n	**pae**	**sfm**	**hsa**	**eor**	**air**
1	d/c	any	d/c	d/c	d/c
2	d/c	0.4 – 2.50	d/c	d/c	d/c
3	yes	0.8 – 1.25	d/c	d/c	d/c
4	yes	0.9 – 1.11	yes	yes	yes
5	yes	1.0 – 1.00	yes	yes	yes

Note: In the above table,

d/c means “don’t care”
pae means “process all extractions”
sfm means “scale factor multiplier”
hsa means “honor scaling algorithm”
eor means “emulate overlapped rectangles”
air means “avoid image-size rounding.”

For information on the effects of these values on the different transfer elements, see the Description section below.

Description

Sets the picture quality acceptable to the application, for the current page or scanner device.

Where image transforms can be off-loaded from the host to the device, the ISCTL call can be used by the application to control the trade-off between those host and device capabilities that affect function and performance; for example, the lower the quality requested, the less host manipulation is required.

This call cannot be issued while image entry or retrieval is initialized for the page or scanner specified.

If a given transform element cannot be performed exactly by a device, then GDDM approximates or emulates the function, within the limit(s) specified, to match the device's capability. For example, a scale factor of 2.4 may be approximated to 2. When emulation occurs, the resulting quality is at least as good as that requested.

If a transform element cannot be modified (within the limit(s)) to fall within the device's capability, GDDM performs the function in the host.

The effect of the quality parameter on transform elements is as follows:

Extractions
"Don't care" means that only the number of extractions (transforms) supported by the device need be processed. A warning message is issued on the IMXFER or IMAPT call if one or more extractions is ignored.

The 3193 can perform four extractions.

"Yes" means that all extractions must be processed, with extra processing in GDDM if there are more than the device can process.

Scaling factor
The scale factors can be modified by a multiplier within the range shown in the table above. For example, if a device has scale factors of (1/4, 1/3, 1/2, 2/3, 3/4, 1, 4/3, 3/2, 2, 3, 4) then the (default) multiplier range of 0.8 through 1.25 allows any scale factor in the range 0.2 through 5.0 to be approximated. An information message is issued on the IMXFER, IMAGTx, or IMAPTx calls if approximation occurs.

If the scale factor cannot be approximated, GDDM emulates the function.

Scaling algorithm
"Don't care" means that a device algorithm can be used, even if this is not the same as that requested by the transforms. An information message is issued on the IMXFER or IMAPT call if a different algorithm is used.

"Yes" means that the requested algorithm must be used, with emulation in GDDM if it is not supported by the device.

Overlapped target rectangles
When target rectangles overlap, some devices do not always give correct results in the overlapped parts. This is because the processing of rectangles (possibly in parallel) may be determined by the order the source rectangles occur in the sequential image data, rather than the order the rectangles are specified in the projection; that is, the incorrect result is caused by mixing the (correct) pixels in a different order. Even when overlap occurs, correct results are obtained for those pixels mixed with "symmetric" mix modes, that is, if the logical expression for the result is commutative and associative.

"Unexpected" target rectangle overlap can occur if scale factors or resolutions are modified by approximation (see above).

"Don't care" means that overlapped target rectangles may be processed by the device. An information message is issued on the IMXFER or IMAPT call if overlap can cause incorrect results.

"Yes" means that all extractions must be processed ***as if*** they were sequentially in the correct order, with emulation in GDDM if overlap could cause incorrect results.

Image-size rounding
The scanner can only scan an area of the paper that is a multiple of 8 pixels wide. Also, the left edge of the scanned area must be a multiple of 8 pixels from the left edge of the scanner detector.

When the "avoid image-size rounding" quality value is "don't care," GDDM may round the scanner image size, and the extract sizes (specified in a projection) to suit the scanner limitations. It is best to use the "don't care" value if direct transmission from the scanner is required.

When the "avoid image-size rounding" quality value is "yes," GDDM processes the scanned images to ensure that the effects of the rounding are not noticeable by the application or the user.

Principal errors

```
ADM3386 E  CALL NOT ALLOWED DURING GET/PUT SEQUENCE
ADM3455 E  DEVICE n IS INVALID
ADM3456 E  QUALITY n IS INVALID
```

ISENAB

Function

To enable or disable image cursor.

ISENAB	**(device, control)**
APL code	189
GDDM RCP code	X'0C301200' (204476928)

Parameters

device *(specified by user) (fullword integer)*
The type of image cursor to be enabled or disabled. Possible values are:
- **1** Locator cursor
- **2** Box cursor

control *(specified by user) (fullword integer)*
The new state of the image cursor. Possible values are:
- **0** Disabled.
  No input expected from the image cursor. If the cursor cannot physically be disabled, any input is ignored by GDDM. This is the initial state.
- **1** Enabled.
  Input is expected from the cursor and is to be allowed when subsequent ASREAD calls are issued.

Description

Enables or disables an image cursor and associates it with image 0, the image on the current GDDM page. While an image cursor is enabled, the operator can manipulate it, and its current value is updated on each ASREAD from the terminal. The value can be queried by issuing the appropriate image cursor query call.

Image input can be globally controlled by the FSENAB call. Image cursor input is not possible unless a call of `FSENAB(3,1)` has been issued to enable image input devices.

Initializing an image cursor does not enable it.

If a locator is enabled, the initial position, specified on ISILOC is used to position the image cursor. If the locator has not been initialized before being enabled, the default cursor is used and its position set to the center of the image field.

If a box cursor is enabled, the initial position and size, specified on ISIBOX is used to position the image cursor. If the box cursor has not been initialized before being enabled, the default cursor is used and its position set to the center of the image field. The default box cursor size is one character cell.

When an image cursor is enabled or disabled, the default image field is created if it has not already been specified or defaulted. When the image field is deleted or redefined, all image cursors are reset to their default state; that is, disabled with default initial values.

Enabled cursors can be manipulated by the operator, whenever the current page is displayed.

Principal errors

```
ADM3453 E  CONTROL VALUE n IS INVALID
ADM3490 W  IMAGE CURSOR IS ALREADY ENABLED
ADM3498 E  IMAGE LOCATOR CURSOR IS NOT AVAILABLE
ADM3499 E  IMAGE BOX CURSOR IS NOT AVAILABLE
```

ISESCA

Function

To control echoing of scanner image.

ISESCA	**(control)**
APL code	185
GDDM RCP code	X'0C300B00' (204475136)

Parameters

control *(specified by user) (fullword integer)*
Specifies whether or not to echo scanner input. Possible values are:
0 Do not echo.
1 Echo.

Description

Defines whether input from the scanner is to be echoed (displayed) on the display device to which the scanner is attached.

The default echo state (when ISESCA has not been called) is no echo.

The effect of specifying `control=1` (echo) is the same as performing an IMXFER from image –1 to image 0 at the same time as the transfer call (IMXFER, IMAGTx, IMASAV) that causes scanner input; the same projection is used, and the attributes of the image are not changed.

This call sets the echoing option for the current device only.

Echoing is done by the 3193 display unless the projection contains transformations that the 3193 cannot process, within the quality requirements specified in the ISCTL or ISXCTL calls.

Principal errors

```
ADM3453 E  CONTROL VALUE n IS INVALID
ADM3470 E  SCANNER DOES NOT EXIST
```

ISFLD

Function

To define image field.

ISFLD	**(row, column, depth, width, control)**
APL code	180
GDDM RCP code	X'0C300000' (204472320)

Parameters

row *(specified by user) (fullword integer)*
The row position on the page of the top left-hand corner of the image field. Zero indicates that the image field is to be deleted.
column *(specified by user) (fullword integer)*
The column position on the page of the top left-hand corner of the image field. Zero indicates that the image field is to be deleted.

depth *(specified by user) (fullword integer)*
The depth of the image field. This is specified in rows. Zero indicates that the image field is to be deleted.
width *(specified by user) (fullword integer)*
The width of the image field. This is specified in columns. Zero indicates that the image field is to be deleted.
control *(specified by user) (fullword integer)*
Specifies whether the application can read data from the image. Possible values are:
0 Default. Read-write if User Control is enabled. Otherwise, write-only.
1 Write-only; data cannot be read from the image.
2 Read-write; data can be read from the image.

Description

Defines the position and size of an image field on the current page, in row/column page coordinates, and specifies the image field control.

The image field is used to display image data. It is an image associated with a position on the current page.

The image field can coexist with, but not overlap, a graphics field on the same page. Only one image field can be defined per page. If there is a graphics field on the page and the device does not accept image data streams, the image field is not shown.

When hardware partitions are used (see PTSCRT), image fields can only be defined in as many GDDM partitions as there are hardware image partitions. GDDM assigns these on a “first-come first-served” basis.

When hardware partitions are used and the device supports hardware scrolling, the whole page must fit within the image presentation space of the device – regardless of how small the image field is.

If an image field already exists when a call is made to ISFLD, the existing image field is deleted and its image data is discarded. If no image field has been created when one is required for a requested function, an image field covering the entire page is automatically created.

If any of the position or size parameters is zero, any existing image field is deleted.

Any panning and zooming that has been performed on the picture is reset.

For family-4 devices, the row and column units that apply to the ISFLD parameters depend on the device token in use:

- For AFPDS tokens which are cell-based, alphanumeric rows and columns are used, just as for family-1 and family-2 devices.
- For other family-4 device tokens (which do not specify cell sizes), the row and column units are determined by the FSPCRT call, which divides the available paper area into a grid. If no FSPCRT call is issued, the row and column units default to pixels.

On family-4 devices, the image field with a non-cell-based device token is rounded down to a multiple of 32 pixels in each direction.

If there is no current page when ISFLD is called to define an image field, then a default page is automatically created.

The **control** parameter specifies whether the image is to be write-only or read-write. If reading of image data for the image field is not required, then write-only should be specified so that GDDM may off-load processing to devices that support image processing functions in a write-only mode. If reading is required, then read-write must be specified and GDDM emulates image functions where necessary.

Principal errors

```
ADM3450 E  IMAGE FIELD ALREADY DEFINED IN PARTITION n
ADM3451 E  IMAGE FIELD POSITION n IS INVALID
ADM3452 E  IMAGE FIELD SIZE n IS INVALID
ADM3453 E  CONTROL VALUE n IS INVALID
ADM3454 E  IMAGE FIELD OVERLAPS GRAPHICS FIELD
ADM3462 E  PAGE TOO LARGE FOR IMAGE.  REASON CODE n1,
           LIMIT n2
```

ISIBOX

Function

To initialize image box cursor.

ISIBOX	**(echo, left-edge, right-edge, top-edge, bot-edge)**
APL code	193
GDDM RCP code	X'0C301600' (204477952)

Parameters

echo *(specified by user) (fullword integer)*
An initial value for the echo characteristic of the image box cursor. The only value supported is 0, specifying that the default echo for the current device is used to show the box cursor.

left-edge *(specified by user) (fullword integer)*
right-edge *(specified by user) (fullword integer)*
The columns of pixels which form the left and right edges of the rectangle. The columns are included in the rectangle. The **left-edge** parameter must be in the range 0 through the image-field width, and **right-edge** must be in the range –1 through the image-field width.

top-edge *(specified by user) (fullword integer)*
bot-edge *(specified by user) (fullword integer)*
The rows of pixels which form the top and bottom edges of the rectangle. The rows are included in the rectangle. The **top-edge** parameter must be in the range 0 through the image-field depth, and **bottom-edge** must be in the range –1 through the image-field depth.

Description

Provides an initial value and echo characteristics for an image box cursor. The box is specified in the pixel coordinate system of image 0, the image on the current GDDM page.

An image box cursor is a device used by a terminal operator for identifying a rectangular area of a displayed image. Typically, it consists of a group of keys used to move the box around and change its shape, and an echo that is displayed over the image to show the size, shape, and position of the box.

The cursor can be initialized in the enabled or disabled states – note that this is not the same as graphics.

Initializing the box cursor does not change its enabled/disabled state.

When an image box cursor is initialized, and the image field has not been defined, the default image field is created. When the image field is deleted or redefined, the initial values for the image box cursor are reset to the default.

The default image box cursor is positioned in the center of the image field and has zero size. The default is used when no ISIBOX call has been issued for the current image field.

Note: The initial default box is a box the size of one character cell, and is positioned with its center in the center of the image field.

GDDM does not ensure positioning accuracy to within one pixel on all devices.

Device variations: For the ***3193 display***, the echo is the box outline.

For ***other 3270-family displays***, the image box cursor is not supported.

Principal errors

```
ADM3499 E  IMAGE BOX CURSOR IS NOT AVAILABLE
ADM3491 E  ECHO TYPE n IS NOT SUPPORTED
ADM3362 E  INVALID RECTANGLE COORDINATE VALUE
ADM3494 E  LEFT/TOP EDGE n IS INVALID
ADM3493 E  RIGHT/BOTTOM EDGE n IS INVALID
ADM3492 E  RIGHT/BOTTOM EDGE n1 LESS THAN LEFT/TOP EDGE
           n2
```

ISILOC

Function

To initialize image locator cursor.

ISILOC	**(echo, h-pos, v-pos)**
APL code	191
GDDM RCP code	X'0C301400' (204477440)

Parameters

echo *(specified by user) (fullword integer)*
An initial value for the echo characteristic of the image locator cursor. The only value supported is 0, specifying that the default echo for the current device is used to show the locator cursor position.

For the 3193 display, this is a small cross.

For other 3270-family displays, this is the alphanumeric cursor.

h-pos *(specified by user) (fullword integer)*
The initial horizontal position in pixels of the locator cursor.

v-pos *(specified by user) (fullword integer)*
The initial vertical position in pixels of the locator cursor.

Description

Provides an initial value and echo characteristics for an image locator cursor. The locator is specified in the pixel coordinate system of image 0, the image on the current GDDM page.

An image locator cursor is a device used by a terminal operator for locating a point on a displayed image. It typically consists of a group of keys used to move the point around, and an “echo” displayed over the image to show where the point is located.

The cursor can be initialized in the enabled or disabled states – note that this is not the same as graphics.

Initializing the image locator cursor does not change its enabled/disabled state.

When an image locator cursor is initialized, and the image field has not been defined, the default image field is created. When the image field is deleted or redefined, the initial values for the image locator cursor are reset to the default.

The default location is in the center of the image field. The default position is used when an ISILOC call has not been issued for the current image field.

GDDM does not ensure positioning accuracy to within one pixel on all devices.

It is possible to define an image locator for a device that cannot reasonably support one; for example, a printer or plotter. If this is done, the locator position never changes from the initial position.

Principal errors

```
ADM3491 E  ECHO TYPE n IS NOT SUPPORTED
ADM3497 E  INITIAL CURSOR POSITION n1,n2 IS INVALID
ADM3498 E  IMAGE LOCATOR CURSOR IS NOT AVAILABLE
```

ISLDE

Function

To load external read-only image.

ISLDE	**(id)**
APL code	186
GDDM RCP code	X'0C300C00' (204475392)

Parameters

id *(specified by user) (fullword integer)*
The image to be loaded; it must be –1.

Description

Loads or updates an external read-only image. The only read-only images supported are scanners. This call is used to load a sheet of paper into the scanner. If the scanner is already loaded, it causes the current sheet to be ejected, and the next sheet loaded.

Notes:

1. If paper needs to be loaded when an IMXFER/IMAGTS call is issued, GDDM does this automatically.
2. When this call is used with an IBM 3117 scanner, the scanner is reset so that it is ready to scan from the top of the document.
3. When this call is used with an IBM 3118 Model 2 scanner with an Automatic Document Feed attached, the next sheet of paper is automatically loaded from the document chute and no operator intervention is required.

Principal errors

```
ADM3351 E  IMAGE IDENTIFIER n IS INVALID
ADM3358 E  IMAGE n DOES NOT EXIST
ADM3470 E  SCANNER DOES NOT EXIST
ADM3477 E  SCANNER NOT READY. MAY BE SWITCHED OFF
ADM3478 E  NO PAPER IN SCANNER
ADM3479 E  SCANNER LAMP INTENSITY IS TOO LOW
ADM3480 E  SCANNER PAPER JAM
ADM3481 E  UNRECOVERABLE SCANNER ERROR OCCURRED
ADM3483 E  SCANNER DISCONNECTED
```

ISQBOX

Function

To query image box cursor.

ISQBOX	**(echo, left-edge, right-edge, top-edge, bot-edge, in-image, status)**
APL code	192
GDDM RCP code	X'0C301500' (204477696)

Parameters

echo *(returned by GDDM) (fullword integer)*
This field is reserved for future use and is always set to zero.

0 The default echo for the current device is used to show the box cursor.

For the 3193 display, it is the box outline.

For other 3270-family displays, the image box cursor is not supported.

left-edge *(returned by GDDM) (fullword integer)*
right-edge *(returned by GDDM) (fullword integer)*
The coordinate of the columns of pixels that form the left and right edges of the rectangle. The columns are included in the rectangle.

top-edge *(returned by GDDM) (fullword integer)*
bot-edge *(returned by GDDM) (fullword integer)*
The coordinate of the rows of pixels that form the top and bottom edges of the rectangle. The rows are included in the rectangle.

in-image *(returned by GDDM) (fullword integer)*
Indicates whether all four corners of the box are within the image on the current GDDM page. Coordinates that are outside the image on the current GDDM page are given the appropriate values extrapolated from the pixel coordinate range of the image on the current GDDM page.

Possible values are:

0 All four corners of the box are outside the image on the current GDDM page and none of the image on the current GDDM page is inside the box.

1 All four corners of the box are inside the image on the current GDDM page.

2 One or more corners of the box are outside the image on the current GDDM page, and part or all of the image on the current GDDM page is inside the box.

status *(returned by GDDM) (fullword integer)*
The current status of the image box cursor. Possible values are:

0 Disabled.
The box returned is the last one received from the device when the box cursor was enabled, or the default box if the box cursor has not been enabled since it was initialized.

1 Enabled.
The box returned is the one received from the device on the last ASREAD call, or the default box if no ASREAD call was issued since enabling the box cursor.

Description

The current value, echo characteristics, and status of an image box cursor. The box is specified in the pixel coordinate system of image 0, the image on the current GDDM page.

Principal errors

```
ADM3499 E  IMAGE BOX CURSOR IS NOT AVAILABLE
```

ISQCOM

Function

To query image compressions supported by the device.

ISQCOM	**(device, count, array)**
APL code	194
GDDM RCP code	X'0C301800' (204478464)

Parameters

device *(specified by user) (fullword integer)*
The device whose supported image compressions are to be returned. Possible values are:
-1 an attached scanner
0 the current primary device (display, printer, or plotter)

count *(specified by user) (fullword integer)*
The number of elements in **array**. The number of image compressions supported are given by FSQURY.

array *(returned by GDDM) (an array of fullword integers)*
An array of image compressions supported by the device. Possible values are:
1 uncompressed
2 MMR (IBM 8815)
3 IBM 4250
4 IBM 3800

Note: MMR = modified-modified read.

Description

Returns the image compressions supported by the current primary device, or a scanner attached to it.

Note that this does not limit the compressions that can be specified (on, for example, an IMAGTS(–1,...) call), but defines which compressions can be used without requiring GDDM to perform automatic conversion, with a subsequent increase in host loading.

Principal errors

```
ADM3455 E  DEVICE n IS INVALID
ADM3459 E  COUNT n IS INVALID
```

ISQFLD

Function

To query image field.

ISQFLD	**(row, column, depth, width, control)**
APL code	181
GDDM RCP code	X'0C300001' (204472321)

Parameters

row *(returned by GDDM) (fullword integer)*
column *(returned by GDDM) (fullword integer)*
The row and column position on the page of the top left-hand corner of the image field. Zeros indicate that no image field exists.

depth *(returned by GDDM) (fullword integer)*
width *(returned by GDDM) (fullword integer)*
The size of the image field on the page in terms of rows and columns. Zeros indicate that no image field exists.

control *(returned by GDDM) (fullword integer)*
A value that indicates whether the application can read data from the image. Possible values are:
0 No image field exists.
1 Write-only; data cannot be read from the image.
2 Read-write; data can be read from the image.

Description

Returns the position and size of the image field on the current page, in row/column page coordinates, and returns the image field control.

For a general description of the image field, see ISFLD.

If no image field exists, all parameters return zero values.

Principal errors

None.

ISQFOR

Function

To query image formats supported by the device.

ISQFOR	(device, count, array)
APL code	184
GDDM RCP code	X'0C301700' (204478208)

Parameters

device *(specified by user) (fullword integer)*
The device whose supported image formats are to be returned. Possible values are:
–1 An attached scanner
0 The current primary device (display, printer, or plotter).

count *(specified by user) (fullword integer)*
The number of elements in **array**. The number of image formats supported are given by FSQURY.

array *(returned by GDDM) (an array of fullword integers)*
An array of image formats supported by the device. Each element has the value:
1 Unformatted
2 3193 data stream format
3 Page printer format.

Description

Returns the image formats supported by the current primary device, or a scanner attached to it.

Note that this does not limit the formats that can be specified (on, for example, an IMAGTS(-1,...) call), but defines which formats can be used without requiring GDDM to perform automatic conversion, with a subsequent increase in host loading.

Principal errors

```
ADM3455 E  DEVICE n IS INVALID
ADM3459 E  COUNT n IS INVALID
```

ISQLOC

Function

To query image locator cursor position.

ISQLOC	(echo, h-pos, v-pos, in-image, status)
APL code	190
GDDM RCP code	X'0C301300' (204477184)

Parameters

echo *(returned by GDDM) (fullword integer)*
This field is reserved for future use and always returns zero.
0 The default echo for the current device is used to show the image locator position.

For the 3193 display, this is a small cross.

For the other 3270-family displays, this is the alphanumeric cursor.

h-pos *(returned by GDDM) (fullword integer)*
The horizontal position in pixels of the locator cursor.

v-pos *(returned by GDDM) (fullword integer)*
The vertical position in pixels of the locator cursor.

in-image *(returned by GDDM) (fullword integer)*
Indicates whether the locator position is within image 0. Coordinates that are outside image 0 are given the appropriate values extrapolated from the pixel coordinate range of image 0. Possible values are:
0 The position is outside image 0
1 The position is inside image 0.

status *(returned by GDDM) (fullword integer)*
The current status of the image locator cursor. Possible values are:
0 Disabled
The locator position is the last one received from the device when the locator was enabled, or the default position if the locator was not enabled since initializing the locator cursor.
1 Enabled
The locator returned is the one received from the device on the last ASREAD call, or the default position if no ASREAD call has been issued since enabling the locator cursor.

Description

Returns the current value, echo characteristics, and status of an image locator cursor. The locator is specified in the pixel coordinate system of image 0, the image on the current GDDM page.

Principal errors

```
ADM3498 E  IMAGE LOCATOR CURSOR IS NOT AVAILABLE
```

ISQRES

Function

To query supported image resolutions.

ISQRES	**(device, res-unit, h-ctl, ref-h-res, v-ctl, ref-v-res, h-res, v-res, info)**
APL code	188
GDDM RCP code	X'0C300E00' (204475904)

Parameters

device *(specified by user) (fullword integer)*
The device whose supported resolutions are to be returned. Possible values are:
–**1** An attached scanner
0 The display, printer, or plotter device.

res-unit *(specified by user) (fullword integer)*
The units of resolution for the **ref** values and for which resolutions are to be returned. Possible values are:
0 Inches
1 Meters.

h-ctl *(specified by user) (fullword integer)*
The value that is to be returned in **h-res**. Possible values are:
–**2** Return the value nearest to and less than the **ref-h-res** value.
–**1** Return the value nearest to and less than or equal to the **ref-h-res** value.
0 Return the value nearest to the **ref-h-res** value.
1 Return the value nearest to and greater than or equal to the **ref-h-res** value.
2 Return the value nearest to and greater than the **ref-h-res** value.

ref-h-res *(specified by user) (short floating point)*
The reference value for horizontal resolution. This value is used together with **h-ctl**.

v-ctl *(specified by user) (fullword integer)*
The value that is to be returned in **v-res**. For a description of the valid values, see **h-ctl**.

ref-v-res *(specified by user) (short floating point)*
The reference value for vertical resolution. This value is used together with **v-ctl**.

h-res *(returned by GDDM) (short floating point)*

v-res *(returned by GDDM) (short floating point)*
Pairs of vertical and horizontal resolutions that meet the requirements specified by **h-ctl, v-ctl, ref-h-res**, and **ref-v-res**. If zeros are returned, this indicates that no supported resolution pair met the specification.

info *(returned by GDDM) (fullword integer)*
Gives more information about the resolutions returned in **h-res** and **v-res**. Possible values are:
0 The returned values are a discrete pair of supported resolutions.
1 Any resolution is supported. In this case, the returned values is always equal to the **ref** values.

Description

Returns the nearest supported resolution pair to the pair specified in **ref-h-res, ref-v-res**.

The nearest supported resolution equal to or greater than the **ref** value is returned when the **ctl** value is positive. The nearest supported resolution less than the **ref** value is returned when the **ctl** value is negative.

The aspect ratio (difference between h and v resolutions) is taken into account when choosing the nearest supported resolution pair. When several supported resolution pairs are near the reference point, the pair chosen will give the smallest change in aspect ratio.

Notes:

1. Images of an resolution value can be sent to output devices; the ISQRES call defines those resolutions that do not require conversion in the host.
2. Resolution conversion is subject to approximation as defined by the ISCTL call.

Principal errors

```
ADM3356 E  IMAGE n1 RESOLUTION UNIT n2 IS INVALID
ADM3453 E  CONTROL VALUE n IS INVALID
ADM3455 E  DEVICE n IS INVALID
ADM3470 E  SCANNER DOES NOT EXIST
```

ISQSCA

Function

To query image scanner device.

ISQSCA	**(count, attributes)**
APL code	187
GDDM RCP code	X'0C300D00' (204475648)

Parameters

count *(specified by user) (fullword integer)*
The number of elements to be set in the **attributes** array. This value can be zero, in which case, no attributes are returned.

attributes *(returned by GDDM) (an array of fullword integers)*
Values defining scanner attributes.

1 Current scanner status
Returns a value indicating the current status of the scanner. This value can be used after a failing ISLDE, IMAGTS, or IMXFER call from the scanner to determine what caused

the error. This may be simpler than querying the error message itself. Possible values are:

0 Scanner is either not ready, powered off, or unplugged
1 Scanner is ready
2 Paper is jammed
3 Lamp intensity is too low (bulb needs changing)
4 Some other unexpected hardware error
5 No paper in scanner.

2 ADF (Automatic Document Feeder) Status.
Returns the status of the ADF feature if fitted to the scanner. Possible values are:

0 Scanner does not have the ADF feature.
1 There is more paper in the ADF.
2 The ADF is empty.

Description

Returns the current values of variable scanner attributes.

Notes:

1. The current resolution can be queried with the IMAQRY call.
2. Fixed device characteristics are queried using the FSQURY call.
3. Querying the current scanner status does ***not*** cause communication with the scanner, it simply returns the current known status that was set on any previous scanner I/O call, such as IMXFER, for example.

Principal errors

```
ADM3473 E  ATTRIBUTE COUNT n IS INVALID
ADM3470 E  SCANNER DOES NOT EXIST
```

ISSE

Function

To run the Image Symbol Editor.

ISSE	**(symbol-set-name)**
APL Code	1201
GDDM RCP code	X'18000000' (402653184)

Parameters

symbol-set-name *(specified by user) (8-byte character string)*
Either the name of a new or existing image symbol set, or blanks (in which case the symbol set name will have to be typed in by the operator after the first panel has been presented).

Description

Run the GDDM Image Symbol Editor.

Note: When the Image Symbol Editor is invoked from an application program, the ESLIB routine can be called beforehand, to specify the libraries to be used for retrieving and storing symbol sets and object decks.

Principal errors

None.

ISXCTL

Function

To set extended image quality control parameters.

ISXCTL	**(device, count-1, array-1, count-2, array-2)**
APL code	183
GDDM RCP code	X'0C300003' (204472323)

Parameters

device *(specified by user) (fullword integer)*
The image to which the call applies. Possible values are:

–**1** Scanner
0 Current page

count-1 *(specified by user) (fullword integer)*
The number of elements in **array-1**, from 0 through 4.

array-1 *(specified by user) (an array of fullword integers)*
The limit, or limits, of approximation allowed for transforms. If a given transform element cannot be performed exactly by a device, then GDDM may approximate the function, within the limit, or limits, specified, to match the device's capability. The array elements have the following meanings:

1 Process all extractions
Specifies whether all extractions are to be processed. Possible values are:

–**1** Unchanged (default if element not included)
0 Don't care
1 Yes

2 Honor scaling algorithm
Specifies whether the scaling algorithm is to be used. Possible values are:

–**1** Unchanged (default if element not included)
0 Don't care
1 Yes

3 Emulate overlapped rectangles
Specifies whether overlapped rectangles are to be emulated. Possible values are:

–**1** Unchanged (default if element not included)
0 Don't care
1 Yes

4 Avoid image size rounding
Specifies whether the size of extractions are to be rounded to meet the requirements of the source device (scanner). Possible values are:

–**1** Unchanged (default if element not included)
0 Don't care
1 Yes

count-2 *(specified by user) (fullword integer)*
The number of elements in **array-2**, from 0 through 2.

array-2 *(specified by user) (array of short floating-point numbers)*
The limit, or limits, of approximation allowed for transforms. If a given transform elements cannot be performed exactly by a device, then GDDM may approximate the function, within the limit, or limits, specified, to match the device's capability. The array elements have the following meanings:

1 lower scaling limit

–**1.0** Unchanged (default if element not included)
0.0–1.0 Scale factor lower multiplier

2 upper scaling limit

–**1.0** Unchanged (default if element not included)
≥**1.0** Scale factor upper multiplier

Description

Specifies the picture quality acceptable to the application, for the current page or scanner device.

This call allows individual setting of the quality control parameters described in ISCTL.

This call cannot be issued while image entry or retrieval is initialized for the page or scanner specified.

ISXCTL and ISCTL set the same image control parameters. ISXCTL partially or wholly updates values previously set by ISCTL, or the default values. ISCTL sets **all** of the image control parameters, and therefore overrides the effects of any previous call to ISXCTL.

Principal errors

```
ADM3386 E  CALL NOT ALLOWED DURING GET/PUT SEQUENCE
ADM3455 E  DEVICE n IS INVALID
ADM3457 E  COUNT_n1 n2 IS INVALID
ADM3458 E  INVALID VALUE f FOR ARRAY_n1, ELEMENT n2
```

MSCPOS

Function

To set cursor position.

MSCPOS (position)	
APL code	1112
GDDM RCP code	X'0C280600' (203949568)

Parameters

position *(specified by user) (fullword integer)*
The position of the cursor within the field. A value of 1 indicates the first position following the attribute byte, a value of 2 the second, and so on. If the value is 0, the cursor is set underneath the starting attribute-byte. If the value is greater than the length of the field, the cursor is positioned underneath the last character of the field.

Note: If, when the map was defined, the field in which the cursor is to be positioned was designated as a double-byte character string field (DBCS – used for Kanji and Hangeul), the cursor position is interpreted in units of two-byte characters.

If, when the map was defined, the field in which the cursor is to be positioned was designated as a mixed string field, the cursor position is interpreted in one-byte units, including any SO/SI characters inserted in the application data. Thus, if a field is mixed-without-position, the value required by MSCPOS (or returned by MSQPOS) may differ from the character position of the cursor on the screen.

Description

Positions the cursor in a field that is contained in a map. The field in which the cursor is to be positioned must have its cursor adjunct set in the application data structure for the map. If none of the fields contained in the map has its cursor adjunct set, MSCPOS has no effect.

The MSCPOS call affects only the next MSPUT operation. If MSPUT is used without a preceding MSCPOS, the cursor is positioned under the first character of the field with the cursor adjunct set, if there is one.

Principal errors

```
ADM0966 E  PAGE n IS NOT MAPPED
ADM0984 E  CURSOR POSITION IS NEGATIVE OR TOO LARGE
```

MSDFLD

Function

To create or delete a mapped field.

MSDFLD	**(id, row, column, map-name)**
APL code	1108
GDDM RCP code	X'0C280500' (203949312)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the mapped field. It must be a positive integer greater than zero.

If there is an existing mapped field with the same identifier, the existing mapped field is deleted (see the section below about row or column with zero value).

row *(specified by user) (fullword integer)*
column *(specified by user) (fullword integer)*
The position on the page of the rectangle covered by the mapped field. Usually, the position of the mapped field is taken from the map definition by specifying **row** and **column** as –1. The size of the mapped field is taken from the map definition.

If the value of **row** or **column** is zero, any existing field with the same identifier is deleted.

The field must not overlap any existing mapped field.

map-name *(specified by user) (8-byte character string)*
The name of the map that defines the properties of the field. The maps of all the mapped fields within a page are taken from the same mapgroup; see MSPCRT.

Description

Creates a mapped field and positions it on the current page. The map's constant fields are put onto the page. If the cursor has not previously been positioned and the map has defined a static cursor-position, the cursor is placed at that static cursor-position. The application data area associated with the mapped field is set to its default value.

Notes:

1. The current page must have been created with the MSPCRT call but it can also contain simple alphanumeric fields.
2. The size of the mapped field is found from the map definition. It cannot overlap any other mapped field.
3. A mapped field must not overlap a simple alphanumeric field. Also, care must be taken when placing mapped and alphanumeric fields next to each other, because of the attribute bytes that enclose alphanumeric fields. Undesirable results can occur if the attributes intrude into a mapped field.
4. The area occupied by a mapped field can overlap a graphics or image field. Where such overlaps occur, the mapped data takes precedence, but the results are device-dependent. For devices that support background transparency, the mapped field is transparent, that is, the underlying graphics or image are always visible. For devices that do ***not*** support background transparency, the mapped field is opaque, that is, no graphics or image appears in the cells that are occupied by the mapped field. However, the map definition can contain a graphic area definition, for which a GSFLD call is issued automatically for the appropriate area, and in which the graphics is always visible.
5. The specification of the map position in the map definition can be overridden by the MSDFLD call.
6. Parts of the map that lie outside the page will not be visible to the operator. A warning message is issued if all the map is outside the page. An information message is issued if part of the map is outside the page.
7. If the map is defined as a floating map and is not explicitly positioned, the call is rejected if the floating area is full.
8. Any map can be explicitly positioned within the floating area of the current page. Any attempt to float another map into the same position is rejected.
9. If the page size was defaulted when the MSPCRT call was issued and if the mapgroup associated with the page does not match the current screen or partition, the floating area of the map may extend outside the current page. This can lead to a warning message being issued by MSDFLD.

Principal errors

```
ADM0966 E  PAGE n IS NOT MAPPED
ADM0968 E  ALPHANUMERIC FIELD a1 AND MAPPED FIELD a2
           OVERLAP
ADM0970 E  MAP 'a1' IS NOT IN MAPGROUP 'a2'
ADM0971 E  ROW OR COLUMN IS LESS THAN -1 OR TOO LARGE
ADM0972 E  MAPPED FIELD a1 OVERLAPS MAPPED FIELD a2
ADM0973 E  INSUFFICIENT SPACE LEFT IN FLOATING AREA FOR
           MAPPED FIELD a
ADM0974 E  MAP 'a' FLOATS HORIZONTALLY BUT VERTICALLY
           FLOATING MAPS USED
ADM0975 E  MAP 'a' FLOATS VERTICALLY BUT HORIZONTALLY
           FLOATING MAPS USED
ADM0978 E  MAPPED FIELD ID n IS NOT GREATER THAN ZERO
ADM0990 W  MAP-DEFINED GRAPHIC FIELD IS OUTSIDE PAGE
ADM0991 W  MAPPED FIELD a IS POSITIONED OUTSIDE THE
           PAGE
ADM0992 I  MAPPED FIELD a IS PARTIALLY OUTSIDE THE PAGE
```

MSGET

Function

To retrieve data from a mapped field.

MSGET	**(id, option, length, ads)**
APL code	1110
GDDM RCP code	X'0C280502' (203949314)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the mapped field.

option *(specified by user) (fullword integer)*
The type of data to be returned. Possible values are:

- **0** The data record is returned.
- **1** Reserved.
- **2** Reserved.
- **3** The ADS for highlighting is returned.
- **4** The ADS for color is returned.
- **5** The ADS for programmed symbols is returned.

length *(specified by user) (fullword integer)*
The length of the **ads** parameter.

ads *(returned by GDDM) (character)*
The current value of the contents of the mapped field. The length of the structure must be as least as great as the length defined in the map for the field. The data beyond the end of the map-defined length is unchanged by this operation.

Description

Returns the current value of the contents of a mapped field.

Principal errors

```
ADM0950 W  INPUT FIELD TRUNCATED
ADM0966 E  PAGE n IS NOT MAPPED
ADM0976 E  MAPPED FIELD a DOES NOT EXIST
ADM0978 E  MAPPED FIELD ID n IS NOT GREATER THAN ZERO
ADM0980 E  DATA LENGTH (n) IS SMALLER THAN LENGTH OF
           ADS (a) OF MAP
```

MSPCRT

Function

To create a page for mapping.

MSPCRT	**(id, depth, width, group-name)**
APL code	1102
GDDM RCP code	X'0C280100' (203948288)

Parameters

id *(specified by user) (fullword integer)*
The identifier for the new page. It must be greater than zero (zero is reserved for the default page, which is always available) and unique within the current partition.

depth *(specified by user) (fullword integer)*

width *(specified by user) (fullword integer)*
The size of the page. Usually, the size is taken from the mapgroup; in this case, the depth and width are specified as –1.

If either is specified as zero, the appropriate depth and width according to the area covered by the current partition are used. See FSPCRT for depth and width limitations.

group-name *(specified by user) (8-byte character string)*
The mapgroup name, which is from one through eight characters long, left-justified, and padded with blanks on the right to a total length of 8. Either or both of the last two nonblank characters may be a period (.). If they are, GDDM substitutes code characters for them that are selected according to the capability of the device and the capacity of the screen. These two characters are the device class as shown in Table 3 on page 212. The corresponding generated mapgroup can be read from auxiliary storage.

Note: Unless overridden, the size of the page is determined by the mapgroup definition. The size can be found using the MSPQRY call.

Description

Creates a page in which mapped fields are to be defined, and identifies the mapgroup in which the maps for those mapped fields reside.

Notes:

1. If partitions are in use or if the mapgroup name is given explicitly, the mapgroup may contain maps that are larger than the page. If the page size is given explicitly, the floating area of maps is reduced automatically to fit the page. If, however, the page size is defaulted, the floating area is unchanged and warning messages may be issued by MSDFLD for floating maps.
2. Mapped alphanumerics are supported on family-4 devices if a cell-based AFPDS device token is used.

Principal errors

```
ADM0122 E  PS STORE NUMBER n IS INVALID
ADM0130 E  PAGE n ALREADY EXISTS
ADM0134 E  PAGE IDENTIFIER n IS INVALID
```

Table 3. Device classes for GDDM Interactive Map Definition (MSPCRT)

Default presentation area size	Device class	Typical devices (suitably configured)							
6 x 20	D0					3290			
6 x 40	D1					3290			
12 x 40	D2		3277			3290			
16 x 64	D3					3290			
12 x 80	D4			3278		3290			8775
24 x 80	D5	3179	3277	3278	3279	3290	3270-PC,/G,/GX	5080	8775
32 x 80	D6	3179		3278	3279	3290	3270-PC,/G,/GX	5080	8775
43 x 80	D7			3278		3290	3270-PC,/G,/GX	5080	8775
27 x 132	D8			3278		3290	3270-PC,/G,/GX		
62 x 132	D9					3290	3270-PC,/G,/GX		
66 x 132	P1	Printers (excluding 3283)							
24 x 80	K5	Kanji-Chinese devices: 3278-52, 5550							
32 x 80	K6	Kanji-Chinese devices: 3278-52, 5550							
43 x 80	K7	Kanji-Chinese devices: 3278-52, 5550							
66 x 158	V1	3283-52							

```
ADM0137 E  PAGE SIZE n IS INVALID
ADM0138 E  PAGE DEPTH n1 OR WIDTH n2 IS TOO LARGE
ADM0960 E  PAGE DEPTH OR WIDTH IS LESS THAN -1
ADM0962 E  MAPGROUP 'a' NOT FOUND
ADM0963 E  OBJECT 'a' IS NOT A MAPGROUP
ADM0964 S  MAPGROUP 'a' IS CORRUPTED
```

MSPQRY

Function

To query specified page.

MSPQRY (id, depth, width, group-name)

APL code	308
GDDM RCP code	X'0C040006' (201588742)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the page to be queried.

depth *(returned by GDDM) (fullword integer)*
The depth of the page.

width *(returned by GDDM) (fullword integer)*
The width of the page.

group-name *(returned by GDDM) (8-byte character string)*
The name of the mapgroup associated with the page, if created by the MSPCRT call, or all blanks if created by the FSPCRT call.

The name returned has any substitution characters replaced by their actual values.

Description

Returns information about a page.

Principal errors

```
ADM0132 E  PAGE n DOES NOT EXIST
```

MSPUT

Function

To place data into a mapped field.

MSPUT (id, option, length, ads)

APL code	1109
GDDM RCP code	X'0C280501' (203949313)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the mapped field to be modified.

option *(specified by user) (fullword integer)*
Controls the parts of the map and the associated application data structures to be modified. Possible values are:

0 All data is replaced with data from the application data structure. Character attributes are set to their default values. Any fields in the map that have selector adjuncts indicating that the fields are to be absent, are set to their defaults.

1 Part of the data is replaced. Any fields in the map that have selector adjuncts indicating that the fields are to be absent are unchanged. The selector adjuncts can be used to set fields to their default values.

2 Part of the data is replaced in the same way as for an option value of 1. Also, any fields that were modified by the terminal operator during previous interactions with the application program retain their modification indication.

3 Parts of the shadow application data structure for highlighting are modified. Set the highlighting values required for the characters in each field in the appropriate application data structure field.

4 Parts of the shadow application data structure for color are modified. Set the color values required for the characters in each field in the appropriate application data structure field.

5 Parts of the shadow application data structure for programmed symbols are modified. Set the programmed symbol values required for the characters in each field in the appropriate application data structure field.

Note: For option values 3, 4, and 5, other field adjuncts can be used in the map definition to control the modification of the data. The selector adjunct can be used to indicate that the character attributes for the field are not to be changed or are to be set to the default. The length adjunct can be used to indicate the length of character-attribute data provided. If the field data is longer, the character attributes are padded with default values.

If the field data is shorter, the character attributes are truncated. The cursor adjunct can be used to position the cursor.

length *(specified by user) (fullword integer)*
The length of the application data structure specified in the **ads** parameter. The length can be 0, or it must be at least as great as the length of the application data structure defined to GDDM by the Interactive Map Definition (GDDM-IMD). If the length is 0, the **ads** parameter is not referenced. Instead, GDDM assumes an application data structure of the correct length, filled with blanks.

ads *(specified by user) (character)*
The application data structure. This is a contiguous set of bytes whose layout is determined by the map definer when the map for the field is created.

Description

Modifies the data associated with a map. The whole map, or selected fields within the map, can be replaced with data from the application data structure. The way in which the data in the map appears on the screen can be altered by modifying parts of the shadow application data structures associated with the map. The shadow application data structures contain information about the character attributes that apply to the data in each field of the map.

Principal errors

```
ADM0966 E  PAGE n IS NOT MAPPED
ADM0976 E  MAPPED FIELD a DOES NOT EXIST
ADM0977 E  OPTION IS NOT IN RANGE 0 THROUGH 5
ADM0980 E  DATA LENGTH (n) IS SMALLER THAN LENGTH OF
           ADS (a) OF MAP
ADM0981 E  type ATTRIBUTE SELECTOR FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS NOT BLANK, 1, 2, OR
           3
ADM0982 E  CURSOR SELECTOR FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS NOT BLANK OR 1
ADM0983 E  type ATTRIBUTE FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS INVALID
```

MSQADS

Function

To query application data structure definition.

MSQADS	**(group-name, map-name, format, length, string)**
APL code	1105
GDDM RCP code	X'0C280302' (203948802)

Parameters

group-name *(specified by user) (8-byte character string)*
The name of a mapgroup. It is in the same format as for the MSPCRT call. The mapgroup is not necessarily related to an existing mapped page. The corresponding generated mapgroup may be read from auxiliary storage.

map-name *(specified by user) (8-byte character string)*
The name of a map within the mapgroup. The map is not necessarily related to any existing mapped field.

format *(specified by user) (fullword integer)*
A code indicating the required format of the ADS descriptor. This value must be 1.

length *(specified by user) (fullword integer)*
The number of bytes in the **string** parameter. It must at least equal the ADS descriptor length returned by MSQMAP.

string *(returned by GDDM) (character)*
A byte array into which the ADS descriptor is placed. The format of the descriptor is shown in Figure 13 on page 214 and a detailed description of each section is given below.

Description

Returns a description of the fields that make up the application data structure (ADS) for a specified map. The description includes the names of the fields only if the mapgroup was generated with field names included.

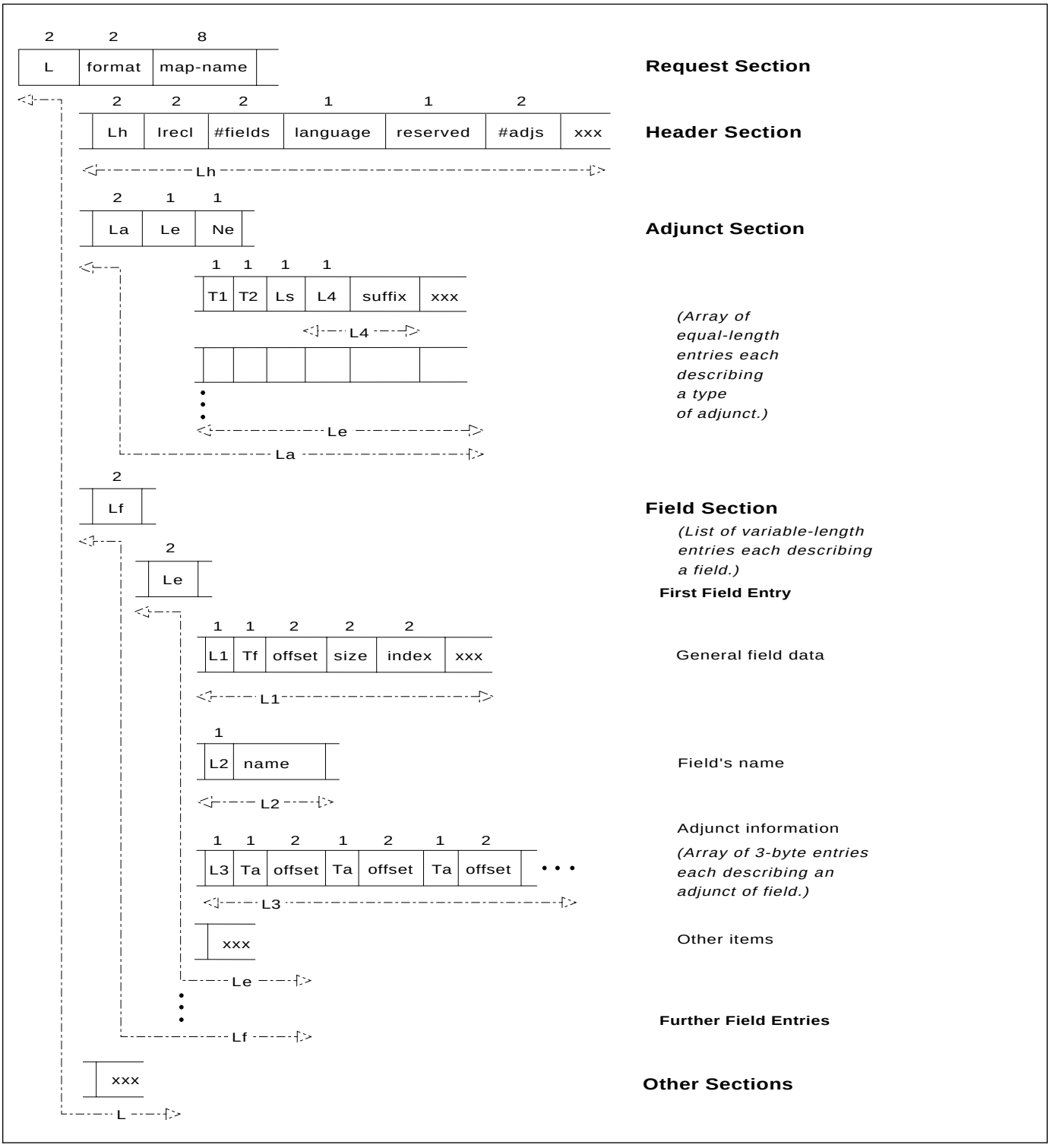


Figure 13. Structure returned from MSQADS

Description of sections: This structure is designed to accommodate more information without impacting an existing application, if the application is written to take account of the following rules:

1. The descriptor is divided into a number of sections and the sections may be divided into sub-sections. Each part is preceded by a length.
2. The meaning and content of a part is defined by its order in the list of parts.
3. New sections may be added in the future, at the end of the list. Applications must be prepared to ignore such parts by using the length of the enclosing section.
4. All lengths always include the length of the length field.

5. New fields may be defined in the future, at the end of the currently defined fields. Applications must be prepared to ignore such parts by using the length of the part.
6. All defined parts and all defined fields in the parts are always present, unless explicitly stated below. Therefore, applications need not cope with the possibility of length fields indicating less data than is expected, unless they need to run on back-levels of GDDM (if, in the future, extensions have been defined).

   However, where the specification allows variable amounts of data, applications must respect the lengths and so on, including the special case of no data.

The sections are:

1. A **Request** section that reflects the information specified by the application. It contains:

   L This field contains the length of the entire structure returned.

   format This indicates the format in which the application is expecting the returned information. The only value supported by GDDM is 1.

   map-name The name of the map, the ADS of which is described.

2. A **Header** section containing general information about the map:

   Lh The length of this section.

   lrecl The length of the ADS for this map.

   #fields The number of data-level fields in the map. This is the number of field entries in the field section of the returned ADS descriptor.

   language A code indicating the programming language for which the ADS was generated.

   0 No language specified
   4 PL/DS (for the IBM 8100)
   32 PL/I
   64 COBOL
   128 Assembler.

   reserved A one-byte reserved field.

   #adjs The total number of adjunct fields in the ADS. Thus, the number of fields of any type in the ADS is: #fields +#adjs.

   xxx Any other fields that may be added from time to time. Applications should be prepared to ignore such fields by using the length Lh.

3. An **Adjunct** section containing general descriptions of adjuncts. This is information that has been factored out of the field descriptions, to reduce the size of the description. It is formatted as an array (that is, equal-length entries) to allow direct addressing, and is referenced by the (zero-origined) array index. The adjunct section is preceded by a header:

   La The length of this section
   Le The length of each entry
   Ne The number of entries.

   Each entry contains:

   T1 A one-byte type-code indicating the data type of the adjunct:

   1 Binary
   2 Character
   3 Two-byte encoded characters.

   T2 A one-byte type-code indicating the type of the adjunct:

   2 Selector
   4 Cursor
   6 Length
   8 Base-attribute selector
   9 Base attribute
   10 Highlight-attribute selector
   11 Highlight attribute
   12 Color-attribute selector
   13 Color attribute
   14 Programmed symbols attribute selector
   15 Programmed symbols attribute
   16 Validation attribute selector
   17 Validation attribute
   18 Outlining attribute selector
   19 Outlining attribute.

   Ls The number of bytes of ADS occupied by the adjunct. If this length is specified as zero, the adjunct has the same length as the field to which it applies.

   L4 The length of the adjunct suffix, including the one-byte length field.

   suffix The adjunct suffix, that is, those characters that should be appended to the right of the field's name to form the adjunct field's name. This value is dependent on the programming language used and is absent (L4=1) if the mapgroup was generated without a name dictionary.

   xxx Any other fields that may be added from time to time. Applications should be prepared to ignore such fields by using the length Le.

4. A **Field** section, consisting of a two-byte section length (Lf) followed by a series of entries, each entry describing a "real" field. The fields are described in *screen* order. Each entry consists of a two-byte entry length (Le) followed by a number of items, each item preceded by a one-byte count of the length of that item. The items are:

   a. General information:

      L1 One-byte length of general information.

      Tf A one-byte type-code indicating the data type of the field:

1 Binary
2 Character
3 Two-byte encoded character
4 Mixed data with position
5 Mixed data without position.

offset The offset in the ADS of the field.

size The number of bytes in the ADS of the field.

index The array index, or 0 if not an array.

xxx Any other fields that may be added from time to time. Applications should be prepared to ignore such fields by using the length L1.

b. The field's name:

L2 One-byte length of field name item.

name Name of field. This is absent (L2=1) if the mapgroup was generated without field names.

c. Adjunct information (a series of three-byte entries each defining an adjunct of the field):

L3 A one-byte field containing the length of the adjunct information. The number of adjunct entries is (L3–1)/3.

Ta The type of the adjunct; that is, an index (zero-origined) into the array of adjunct definitions in the adjunct section.

offset The offset in the ADS of the adjunct.

d. Any other items (**xxx**) that may be added from time to time. Applications should be prepared to ignore such fields by using the length Le.

5. Any other sections (**xxx**) that may be added from time to time. Applications should be prepared to ignore such fields by using the length L.

Principal errors

```
ADM0962 E  MAPGROUP 'a' NOT FOUND
ADM0963 E  OBJECT 'a' IS NOT A MAPGROUP
ADM0970 E  MAP 'a1' IS NOT IN MAPGROUP 'a2'
ADM3090 E  ADS DESCRIPTOR FORMAT IDENTIFIER (n) MUST BE
           1
ADM3091 E  DATA  LENGTH  (n)  IS  TOO  SMALL  FOR  ADS
           DESCRIPTOR HEADER
ADM3092 W  DATA  LENGTH  (n)  IS  TOO  SMALL  FOR  ADS
           DESCRIPTOR. IT IS TRUNCATED
```

MSQFIT

Function

To query map fit.

MSQFIT	**(map-name, count)**
APL code	1106
GDDM RCP code	X'0C280303' (203948803)

Parameters

map-name *(specified by user) (8-byte character string)*
The name of a map within the mapgroup associated with the current page. The map does not necessarily belong to any defined mapped field.

count *(returned by GDDM) (fullword integer)*
The number of times the specified map could be created, using its default position, before the floating area becomes full or the map overlaps an existing map.

For a fixed map the count is zero if its default position is occupied, or one if it is not.

For a floating map, the count is zero or more depending on the size of the floating area that remains, and whether fixed maps are overlapping part of the floating area.

Description

Returns the number of times the map fits in the floating area of the mapgroup associated with the page.

Note: If the mapped page created by the MSPCRT call has the default size taken from the mapgroup (but reduced to fit the device or partition), the floating area of the map may extend outside the page. In this case, the MSQFIT call still refers to the floating area, but floating maps created by the MSDFLD call may be totally or partially outside the page.

Principal errors

```
ADM0970 E  MAP 'a1' IS NOT IN MAPGROUP 'a2'
```

MSQFLD

Function

To query mapped field characteristics.

MSQFLD	**(id, row, column, depth, width, map-name)**
APL code	1111
GDDM RCP code	X'0C280503' (203949315)

Parameters

id *(specified by user) (fullword integer)*
The identifier of the mapped field to be queried.
row *(returned by GDDM) (fullword integer)*
The row position of the mapped field.
column *(returned by GDDM) (fullword integer)*
The column position of the mapped field.
depth *(returned by GDDM) (fullword integer)*
The depth of the mapped field.
width *(returned by GDDM) (fullword integer)*
The width of the mapped field.
map-name *(returned by GDDM) (8-byte character string)*
The name associated with the mapped field.

Description

Returns the properties of a mapped field.

Principal errors

```
ADM0966 E  PAGE n IS NOT MAPPED
ADM0976 E  MAPPED FIELD a DOES NOT EXIST
ADM0978 E  MAPPED FIELD ID n IS NOT GREATER THAN ZERO
```

MSQGRP

Function

To query mapgroup characteristics.

MSQGRP	**(group-name, actual, element-no, count, array)**
APL code	1103
GDDM RCP code	X'0C280300' (203948800)

Parameters

group-name *(specified by user) (8-byte character string)*
The name of a mapgroup. It is in the same format as for the MSPCRT call. The corresponding generated mapgroup may be read from auxiliary storage.

actual *(returned by GDDM) (8-byte character string)*
The name of the mapgroup, with substitutions made for any "." characters at the end of the **group-name** parameter.

element-no *(specified by user) (fullword integer)*
The number of the first element in the returned array. It must have a value in the range 1 through 6.

count *(specified by user) (fullword integer)*
The number of fullwords to be returned in the array. It must have a value in the range 1 through 6.

array *(returned by GDDM) (an array of fullword integers)*
A list of fullword values returned by this call.

1, 2 The depth and width of the page that is created if this mapgroup is used as an operand to the MSPCRT call.

3, 4 The position relative to the top of the page of the floating area in the mapgroup. They are set to 0,0 if no floating area is defined. The row and column values are returned in that order.

5, 6 The size of the floating area of the mapgroup. They are set to 0,0 if no floating area is defined.

Note: The values returned are those in the mapgroup definition. If a mapped page is created using this mapgroup, the sizes might be modified because of the size of the partition in which the page is created.

The maximum number of fullwords returned in the array is given by the value of the **count** parameter.

Description

Returns the properties of a mapgroup. The mapgroup need not be in use by a page.

Principal errors

```
ADM0962 E  MAPGROUP 'a' NOT FOUND
ADM0963 E  OBJECT 'a' IS NOT A MAPGROUP
ADM0964 S  MAPGROUP 'a' IS CORRUPTED
```

MSQMAP

Function

To query map characteristics.

MSQMAP	**(group-name, map-name, element-no, count, array)**
APL code	1104
GDDM RCP code	X'0C280301' (203948801)

Parameters

group-name *(specified by user) (8-byte character string)*
The name of a mapgroup. It is in the same format as for the MSPCRT call. The corresponding generated mapgroup can be read from auxiliary storage.

map-name *(specified by user) (8-byte character string)*
The name of a map within the mapgroup.

element-no *(specified by user) (fullword integer)*
The number of the first element in the returned array. It must have a value in the range 1 through 6.

count *(specified by user) (fullword integer)*
The number of fullwords in the returned array.

array *(returned by GDDM) (an array of fullword integers)*
A list of fullword values returned by this call.

- **1, 2** The origin of the map with respect to the page. These have values of –1 for the SAME floating option and 0 for the NEXT floating option. (SAME and NEXT are the line, or column, or both of these options specified when the map was defined under the Interactive Map Definition).
- **3, 4** The depth and width of the map.
- **5** The length of the ADS as defined in the map.
- **6** The length of the ADS descriptor as returned by the MSQADS call.

Description

Returns information about a map within a mapgroup. Neither the map nor the mapgroup need be in use by a page.

Principal errors

```
ADM0962 E  MAPGROUP 'a' NOT FOUND
ADM0963 E  OBJECT 'a' IS NOT A MAPGROUP
ADM0964 S  MAPGROUP 'a' IS CORRUPTED
ADM0970 E  MAP 'a1' IS NOT IN MAPGROUP 'a2'
```

MSQMOD

Function

To query modified fields.

MSQMOD (count, ids, lengths)	
APL code	1107
GDDM RCP code	X'0C280400' (203949056)

Parameters

count *(specified by user) (fullword integer)*
The number of mapped fields for which information is to be returned. This is also the number of elements in the returned arrays. If this number is greater than the number of modified mapped fields, the unused elements of the returned arrays are set to zero.

ids *(returned by GDDM) (an array of fullword integers)*
The identifiers of the mapped fields that have been changed during the last ASREAD or GSREAD call. The order of the identifiers is the same as that in which the MSDFLD calls that created the mapped fields were executed.

lengths *(returned by GDDM) (an array of fullword integers)*
The lengths of the application data structures associated with each mapped field. Each element in this array corresponds to the same element in the array of mapped field identifiers returned as the **ids** parameter.

Description

Returns the identifiers of the mapped fields that have been changed after an ASREAD call or a GSREAD call. It also returns the lengths of their application data structures.

When modified fields are returned, they become unmodified. For example, if ASREAD indicates that there are ten modified fields, and subsequently two calls are made to MSQMOD, each requesting seven fields, the first returns seven, and the second returns the remaining three (and four zero entries), after which there would be no modified fields on the current page.

Principal errors

None.

MSQPOS

Function

To query cursor position.

MSQPOS (position)	
APL code	1113
GDDM RCP code	X'0C280601' (203949569)

Parameters

position *(returned by GDDM) (fullword integer)*
The position of the cursor in a field that is part of a map, see MSCPOS.

A value of –1 means that the cursor does not lie within a map, or it is within a map but not in a field with a cursor adjunct, or the map is not a cursor receiver; see the *GDDM Interactive Map Definition* manual.

Note: If an ASREAD call was used, the position of the cursor can be found by using the ASQCUR call.

Description

Returns the position of the cursor in a map.

If the cursor lies within a field of the map that has a cursor adjunct in the application data structure, the value returned by the MSQPOS call is set to the position of the cursor and the cursor adjunct is set. To find the cursor position, the application program must issue an MSGET call, inspect the cursor adjuncts, and if one is set, issue an MSQPOS call.

The value returned by MSQPOS is reset by each MSGET call.

Principal errors

None.

MSREAD

Function

To present mapped data.

MSREAD	(group-name, map-name, length, ads, attype, attval)
APL code	1101
GDDM RCP code	X'0C280000' (203948032)

Parameters

group-name *(specified by user) (8-byte character string)*
The name of a mapgroup. It is in the same format as for the MSPCRT call. The corresponding generated mapgroup may be read from auxiliary storage.

map-name *(specified by user) (8-byte character string)*
The name of a map within the mapgroup.

length *(specified by user) (fullword integer)*
The length of the application data structure for the map.

ads *(specified by user and returned by GDDM) (character)*
The application data structure for the map. Values to be displayed with the map are taken from it and, after the terminal interrupt, values from the map are returned into it.

attype *(returned by GDDM) (fullword integer)*
The type of attention interrupt received. See ASREAD for a description of this parameter.

attval *(returned by GDDM) (fullword integer)*
The value, if any, associated with the **attype** parameter. See ASREAD for a description of this parameter.

Note: If MSREAD is issued for a partitioned device, and the input is not in the same partition as that in which the map was output, no data is returned to the application program in the **ads** parameter.

Description

Displays a map, waits for an interrupt from the terminal, and returns data from the map. MSREAD provides a convenient combination of operations to perform simple mapping input and output. It operates within the current partition, but creates its own temporary page, leaving the current page status unchanged. MSREAD is essentially equivalent to the sequence: MSPCRT, MSDFLD, MSPUT, ASREAD, MSGET, and FSPDEL.

Principal errors

```
ADM0962 E  MAPGROUP 'a' NOT FOUND
ADM0963 E  OBJECT 'a' IS NOT A MAPGROUP
ADM0964 S  MAPGROUP 'a' IS CORRUPTED
ADM0970 E  MAP 'a1' IS NOT IN MAPGROUP 'a2'
ADM0973 E  INSUFFICIENT SPACE LEFT IN FLOATING AREA FOR
           MAPPED FIELD a
ADM0980 E  DATA LENGTH (n) IS SMALLER THAN LENGTH OF
           ADS (a) OF MAP
ADM0981 E  type ATTRIBUTE SELECTOR FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS NOT BLANK, 1, 2, OR
           3
ADM0982 E  CURSOR SELECTOR FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS NOT BLANK OR 1
ADM0983 E  type ATTRIBUTE FOR '{FIELD
           n|xxxxxx|xxxxxx(m)}' IS INVALID
ADM0990 W  MAP-DEFINED GRAPHIC FIELD IS OUTSIDE PAGE
ADM0991 W  MAPPED FIELD a IS POSITIONED OUTSIDE THE
           PAGE
ADM0992 I  MAPPED FIELD a IS PARTIALLY OUTSIDE THE PAGE
```

PSDSS

Function

To load a symbol set into a PS store from the application program.

PSDSS	(store-number, symbol-set-name, symbol-set-id, length, data)
APL code	203
GDDM RCP code	X'0C040202' (201589250)

Parameters

store-number *(specified by user) (fullword integer)*
Designates the device PS store into which the symbol set is to be loaded. If the specified PS store is already in use for another symbol set, the new definitions replace the previous ones. The number refers to a PS store previously reserved by PSRSV. If it is specified as zero, GDDM selects a suitable PS store not currently in use or reserved. Other valid values of this parameter are 2 through 7 (if the device is a terminal with the maximum of six PS stores).

symbol-set-name *(specified by user) (8-byte character string)*
Contains the name (left-justified) of the symbol set. This serves only to name the symbol set; no file operations are performed. The name is returned on calls to query loaded symbol sets, and is also used to locate an equivalent set if a copy is made to a printer. If neither of these operations is to be performed, the name can be left blank.

symbol-set-id *(specified by user) (fullword integer)*
Gives the identifier by which the symbol set is to be referenced in subsequent calls using the set. The identifier is

passed to GSCS to select the symbol set for graphics text, or to ASFPSS to select it for alphanumerics text. Allowable symbol-set identifiers are 65 through 223.

Each loaded symbol set should have a unique identifier with respect to all other symbol sets loaded by means of GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC calls. This avoids any uncertainty that might arise from a device treating different types of symbol sets as equal candidates for displaying a character string. If, however, a symbol-set identifier is the same as one that has previously been issued for the same type, the new definitions replace the previous ones.

Note: When using segments, remember that the symbol set belongs to the device and not the segments. Therefore, it is advisable to load symbol sets outside of segments; if a symbol set is loaded within a segment, and that symbol set has already been loaded in a previous segment using the same symbol-set identifier, unexpected output may occur when printing the page.

length *(specified by user) (fullword integer)*
The length of data storage provided for the **data** parameter.

data *(specified by user) (character)*
Contains the symbol-set definitions to be loaded.

Description

Loads a set of symbol definitions from data passed by the application program into a PS store in the device. The definitions are transmitted to the device on the next device update operation.

This call should be issued only when a PS store is available. When used with graphics operations, the call should be issued **before** any graphics drawing calls, or after any pages containing graphics have been deleted. Alternatively, a call to PSRSV can be used to reserve a PS store for later loading.

Note: The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0115 E  SYMBOL SET 'a' LENGTH n IS INVALID
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0121 E  PS STORE n UNAVAILABLE
ADM0122 E  PS STORE NUMBER n IS INVALID
ADM0123 E  SYMBOL SET n1 HAS INVALID FORMAT. REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0904 E  a ARE NOT SUPPORTED FOR THIS DEVICE
```

PSLSS

Function

To load a symbol set into a PS store from auxiliary storage.

PSLSS	**(store-number, symbol-set-name, symbol-set-id)**
APL code	204
GDDM RCP code	X'0C040200' (201589248)

Parameters

store-number *(specified by user) (fullword integer)*
Defines the device PS store into which the symbol set is to be loaded. If the specified PS store is already in use for another symbol set, the new definitions replace the previous ones. The number refers to a PS store previously reserved by PSRSV. If it is specified as zero, GDDM selects a suitable PS store not currently in use or reserved. Other valid values of this parameter are 2 through 7 (if the device is a terminal with the maximum of six PS stores).

symbol-set-name *(specified by user) (8-byte character string)*
Contains the name (left-justified) of the symbol set to be read from auxiliary storage. If the symbol set name ends with the period character ".", the period is replaced by another character, depending upon the device family, the alphanumerics cell size, or the pixel resolution of the current device. The information returned in the FSQURY call for **code=0** can be used to find the device's alphanumerics cell size; see FSQURY. For information on the character that replaces the period, refer to the symbol set naming convention described in Chapter 8, "Symbol set formats" on page 275.

Symbol sets to be loaded must be of the type that matches the hardware cell size.

symbol-set-id *(specified by user) (fullword integer)*
Gives the identifier by which the symbol set is to be referenced in subsequent calls using the set. The identifier is passed to GSCS to select the symbol set for graphics text, or to ASFPSS to select it for alphanumerics text. Allowable symbol-set identifiers are 65 through 223.

Each loaded symbol set should have a unique identifier with respect to all other symbol sets loaded by means of GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC calls. This avoids any uncertainty that might arise from a device treating different types of symbol sets as equal candidates for displaying a character string. If, however, a symbol set identifier is the same as one that has previously been issued for the same type, the new definitions replace the previous ones.

Note: When using segments, remember that the symbol set belongs to the device and not the segments. Therefore, it is advisable to load symbol sets outside of segments; if a

symbol set is loaded within a segment, and that symbol set has already been loaded in a previous segment using the same symbol-set identifier, unexpected output may occur when printing the page.

Description

Loads a set of symbol definitions from auxiliary storage into a PS store in the device. The definitions are transmitted to the device in the next device update operation.

This call should be issued only when a PS store is available. When used with graphics operations, the call should be issued **before** any graphics drawing calls, or after any pages containing graphics have been deleted. Alternatively, a call to PSRSV can be used to reserve a PS store for later loading.

Note: The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0115 E  SYMBOL SET 'a' LENGTH n IS INVALID
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0121 E  PS STORE n UNAVAILABLE
ADM0122 E  PS STORE NUMBER n IS INVALID
ADM0123 E  SYMBOL  SET  n1  HAS  INVALID  FORMAT.  REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0307 E  FILE 'a' NOT FOUND
ADM0904 E  a ARE NOT SUPPORTED FOR THIS DEVICE
```

PSLSSC

Function

To conditionally load a symbol set into a PS store from auxiliary storage.

PSLSSC	**(store-number, symbol-set-name, symbol-set-id)**
APL code	205
GDDM RCP code	X'0C040201' (201589249)

Parameters

store-number *(specified by user) (fullword integer)*
Defines the device PS store into which the symbol set is to be loaded. If the specified PS store is already in use for another symbol set, the new definitions replace the previous ones. The number refers to a PS store previously reserved by PSRSV. If it is specified as zero, GDDM selects a suitable PS store not currently in use or reserved. Other valid values of this parameter are 2 through 7 (if the device is a terminal with the maximum of six PS stores).

symbol-set-name *(specified by user) (8-byte character string)*
Contains the name (left-justified) of the symbol set to be read from auxiliary storage. If the symbol-set name ends with the substitution character (period), the "." is replaced by another character, depending upon the device family, alphanumerics cell size, or pixel resolution of the current device. The information returned in the FSQURY call for **code=0** can be used to find the device's alphanumerics cell size. For information on the character that replaces the ".", see the symbols set naming convention described in Chapter 8, "Symbol set formats" on page 275.

symbol-set-id *(specified by user) (fullword integer)*
Gives the identifier by which the symbol set is to be referenced in subsequent calls using the set. The identifier is passed to GSCS to select the symbol set for graphics text, or to ASFPSS to select it for alphanumerics text. Allowable symbol-set identifiers are 65 through 223.

Each loaded symbol set should have a unique identifier with respect to all other symbol sets loaded by means of GSDSS, GSLSS, PSDSS, PSLSS, or PSLSSC calls. This avoids any uncertainty that might arise from a device treating different types of symbol sets as equal candidates for displaying a character string. If, however, a symbol-set identifier is the same as one that has previously been issued for the same type, the new definitions replace the previous ones.

Description

This is similar to the PSLSS call, except that the loading is conditional. The PS store is loaded **only** if it does not already contain a symbol set with the specified symbol-set identifier. This function is intended for use when a convention exists associating a unique symbol set with a symbol-set identifier on the device. An application program requiring that a PS store be loaded with a particular symbol set can issue a call to PSLSSC immediately after initialization of GDDM. This loads the symbol set only if it is not already present in the device. This can save transmissions if a given symbol-set identifier is always associated with a particular set of symbols.

Note: The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0115 E  SYMBOL SET 'a' LENGTH n IS INVALID
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0119 E  SYMBOL    SET    'a'    HAS    INCONSISTENT
           {IMAGE|VECTOR} TYPE
ADM0121 E  PS STORE n UNAVAILABLE
ADM0122 E  PS STORE NUMBER n IS INVALID
```

```
ADM0123 E  SYMBOL SET n1 HAS INVALID FORMAT. REASON
           CODE n2
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
ADM0125 E  SYMBOL SET n CODE POINT X'xx' IS INVALID
ADM0307 E  FILE 'a' NOT FOUND
ADM0904 E  a ARE NOT SUPPORTED FOR THIS DEVICE
```

PSQSS

Function

To query status of device stores.

PSQSS	**(n, types, states, symbol-set-names, symbol-set-ids)**
APL code	211
GDDM RCP code	X'0C040101' (201588993)

Parameters

n *(specified by user) (fullword integer)*
The number of stores to be queried.

types *(returned by GDDM) (an array of fullword integers)*
Identifies the PS store types. Possible values are:

- **–1** Not present on the device
- **0** Nonloadable character set
- **1** Single-plane PS store (for monochrome sets)
- **2** Triple-plane PS store (for monochrome or multicolor sets).

states *(returned by GDDM) (an array of fullword integers)*
Identifies the allocation states of the sets. Possible values are:

- **0** Not in use, allocated, or reserved
- **1** Allocated by GDDM, either for a symbol set or for graphics
- **2** Reserved by PSRSV.

symbol-set-names *(returned by GDDM) (array of 8-byte character tokens)*
Identifies the files from which the symbol sets were loaded, or the names specified in calls to GSDSS. If the PS store was not loaded by GDDM, the name is blank. If the symbol set was loaded by means of PSLSS, and the substitution character was specified, the name returned is the one after the "." was substituted.

symbol-set-ids *(returned by GDDM) (an array of fullword integers)*
Contains the identifiers associated with the symbol sets. The meaning of each identifier depends on the corresponding value of the **states** parameter, as follows:

- **0** The identifier reflects the most recent use of the symbol set. In particular, immediately after GDDM initialization, the identifiers reflect the terminal status at initialization. Unused PS stores have an identifier of 255 at initialization.
- **1** If the symbol set has been loaded by PSDSS, PSLSS, or PSLSSC, the identifier is that specified in one of those calls. If the PS store is in use for graphics, the identifier is in the range 224 through 239.
- **2** The store is reserved, and the validity of the symbol-set identifier is not ensured, because the store is under control of the application program.

Description

Requests information about the contents of stores (both nonloadable and PS).

Each of the parameters (excluding **n**) is an array with **n** elements. Information about the first **n** symbol sets on the device is returned in order, starting with symbol set 1. Information about the base set (symbol store 0) is not returned.

Note: The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0116 E  NUMBER OF SYMBOL SETS n IS INVALID
```

PSRSS

Function

To release a symbol set from a PS store.

PSRSS	**(symbol-set-id)**
APL code	208
GDDM RCP code	X'0C040400' (201589760)

Parameters

symbol-set-id *(specified by user) (fullword integer)*
The identifier of the symbol set to be released from a PS store in the device.

Description

Releases the designated symbol set from a PS store. A symbol set should not be released until it is no longer needed for alphanumerics or graphics. Note that a symbol set can be released from a PS store even if the store is currently reserved.

Note: The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0117 E  SYMBOL SET IDENTIFIER n IS INVALID
ADM0120 E  SYMBOL SET n NOT LOADED
```

PSRSV

Function

To reserve or release a PS store.

PSRSV	**(control, store-number)**
APL code	206
GDDM RCP code	X'0C040203' (201589251)

Parameters

control *(specified by user) (fullword integer)*
Specifies the new status of the PS store. Possible values are:

- **0** The store is to be released and made available for subsequent allocation by GDDM
- **1** The store is to be reserved.

store-number *(specified by user) (fullword integer)*
The number of the PS store affected. Valid values are 2 through 7.

Description

Reserves or releases a PS store for explicit control and use by the application program. This function serves two purposes:

- It can be used to reserve a PS store in the device for explicit control by the application program, except that a PS store cannot be reserved when it is in use for graphics construction. A reserved store is used by GDDM only if it is explicitly referred to in calls to PSDSS, PSLSS, or PSLSSC. (A call to PSRSS to release a symbol set in a reserved PS store is also valid.)
- The function can also be used to release a previously reserved PS store

Notes:

1. When a PS store is released, GDDM erases from its tables all memory of which symbol set was loaded in that store. This is because the assumption is made that PS store reservation may imply that an application program is loading the store by some means other than using GDDM.
2. The information returned for code=0 (element 10) in the FSQURY call indicates whether the primary device supports PS stores.

Principal errors

```
ADM0121 E  PS STORE n UNAVAILABLE
ADM0122 E  PS STORE NUMBER n IS INVALID
ADM0126 E  CONTROL n IS INVALID
```

PTNCRT

Function

To create a partition.

PTNCRT	**(partition-id, no-of-elements, array)**
APL code	1021
GDDM RCP code	X'0C240000' (203685888)

Parameters

partition-id *(specified by user) (fullword integer)*
The identifier of the new partition. It must be greater than 0 (0 is reserved for the default partition), and unique within the current partition set.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes defined for the new partition. It must be in the range 4 through 6.

array *(specified by user) (an array of fullword integers)*
The attributes for the new partition. Possible values are:

- **1** The row position of the top left-hand corner of the new partition in partition-set grid units.
- **2** The column position of the top left-hand corner of the new partition in partition-set grid units.
- **3** The number of rows in partition-set grid units.
- **4** The number of columns in partition-set grid units.
- **5** The device partition identifier to be used, or –1 (in which case the partition identifier is allocated by GDDM). If specified, this value must correspond to a value supported by the primary device.

  A value of –1 can be used if this value is of no concern to the application. In this instance, GDDM tries to

ensure that increasing values are assigned in an order corresponding to the order in which the partitions are created. The device partition identifier values determine the order in which partitions can be selected by the operator on an appropriate device.

Note: This value is ignored for emulated partitions.

6 Visibility, selected as follows:

0 Not visible.
1 Visible (the default).
2 Protected

A partition with a visibility attribute of protected is visible and all alphanumeric fields in the partition are displayed protected, regardless of whether they are defined as protected or unprotected.

Description

Creates a new partition within the current partition set. The new partition becomes the current partition.

Partitions must always be positioned within the partition-set boundaries.

More than one partition may be visible within the boundaries of the partition-set grid at the same time. Each has its own position and size. The default partition size is determined by the partition set grid; for more information, see FSQURY and PTSCRT.

Partitions can overlap, if the partition-set overlap control value allows it. Partitions are opaque, so the part of a partition that is overlapped by another partition is completely obscured by the top partition.

Each partition has its own set of pages. Any of these pages can be displayed within the partition, but only one at a time. Either pages are created using FSPCRT or a default page 0 is created implicitly as required.

Priority is not listed as one of the partition attributes, but the priority of groups of partitions can be set and queried collectively with the following calls:

- PTSSPP
- PTSQPI
- PTSQPN
- PTSQPP.

A newly created partition has priority of viewing over all existing partitions within the current partition set.

The new partition becomes the current partition. The partition has a new set of pages, including a default page 0, which is the current page.

If the device partition identifier is not specified, GDDM tries to ensure that increasing values are assigned in an order corresponding to the order in which partitions are created. The device partition identifiers determine the order in which partitions can be selected by the operator.

The current partition should always be visible when the screen is updated, for instance, by an ASREAD call. If an invisible partition is current, GDDM changes it to visible.

For information about variations in device support for partitions, see “Partitions” on page 241.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3120 E  PARTITION n ALREADY EXISTS
ADM3122 E  PARTITION ID (n) IS INVALID
ADM3123 E  PARTITION n1 PARTITION SET GRID ROW NUMBER
           (n2) IS INVALID
ADM3124 E  PARTITION n1 PARTITION SET GRID COLUMN
           NUMBER (n2) IS INVALID
ADM3125 E  PARTITION n1 DEPTH ON PARTITION SET GRID
           (n2) IS INVALID
ADM3126 E  PARTITION n1 WIDTH ON PARTITION SET GRID
           (n2) IS INVALID
ADM3127 E  PARTITION n PARTITION SET GRID AREA IS NOT
           AVAILABLE
ADM3128 E  PARTITION n1 DEVICE PARTITION ID (n2) IS
           INVALID
ADM3129 E  PARTITION n1 DEVICE PARTITION ID (n2) IS
           ALREADY IN USE
ADM3130 E  PARTITION n1 MAXIMUM NUMBER OF PARTITIONS
           (n2) EXCEEDED
ADM3131 E  PARTITION n1 MAXIMUM NUMBER OF CHARACTERS
           (n2) EXCEEDED
ADM3134 E  PARTITION n1 VISIBILITY (n2) IS INVALID
```

PTNDEL

Function

To delete a partition.

PTNDEL	**(partition-id)**
APL code	1025
GDDM RCP code	X'0C240101' (203686145)

Parameters

partition-id *(specified by user) (fullword integer)*
The identifier of the partition to be deleted.
–**1** Delete the current partition
0 Delete the default partition.

If the current partition is deleted, the most recently created partition becomes the current partition. If there is no such partition, the default partition becomes the current partition.

Description

Deletes a partition within the current partition set.

Principal errors

```
ADM3121 E  PARTITION n DOES NOT EXIST
ADM3122 E  PARTITION ID (n) IS INVALID
```

PTNMOD

Function

To modify the current partition.

PTNMOD	**(element-no, no-of-elements, array)**
APL code	1023
GDDM RCP code	X'0C240002' (203685890)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the array. It must be in the range 1 through 6. For example, if it is 3, the number of rows is the first element in the array.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes to be modified on the current partition. It is the number of elements in the array and must have a value in the range 1 through 6.

array *(specified by user) (an array of fullword integers)*
The new attributes for the current partition. Possible values are:

- **1** The new row position of the top left-hand corner of the current partition in partition-set grid units. A value of –1 leaves the present position unchanged.
- **2** The new column position of the top left-hand corner of the current partition in partition-set grid units. A value of –1 leaves the present position unchanged.
- **3** The new depth of the partition in partition-set grid units. A value of –1 leaves the existing value unchanged.
- **4** The new width of the partition in partition-set grid units. A value of –1 leaves the existing value unchanged.
- **5** The device partition identifier cannot be changed. Only a value of –1 or the real identifier, as returned by PTNQRY can be used.
- **6** The new visibility for the partition. A value of –1 leaves the existing value unchanged.

Description

Modifies the properties of the current partition within the current partition set.

The number of rows seen for any given page displayed in a partition changes according to the new partition depth and width. The page window depth and width for each page belonging to the partition is altered accordingly.

Note: The character-box size for each page belonging to the partition ***does not change***.

The current partition should always be visible when the screen is updated, for instance, by an ASREAD call. If an invisible partition is current, GDDM changes it to visible.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
ADM3123 E  PARTITION n1 PARTITION SET GRID ROW NUMBER
           (n2) IS INVALID
ADM3124 E  PARTITION n1 PARTITION SET GRID COLUMN
           NUMBER (n2) IS INVALID
ADM3125 E  PARTITION n1 DEPTH ON PARTITION SET GRID
           (n2) IS INVALID
ADM3127 E  PARTITION n PARTITION SET GRID AREA IS NOT
           AVAILABLE
ADM3131 E  PARTITION n1 MAXIMUM NUMBER OF CHARACTERS
           (n2) EXCEEDED
ADM3132 E  THE DEFAULT PARTITION MAY NOT BE MODIFIED
ADM3134 E  PARTITION n1 VISIBILITY (n2) IS INVALID
ADM3156 I  PAGE n1 WINDOW ROW ALTERED TO n2 AND COLUMN
           TO n3
```

PTNQRY

Function

To query the current partition.

PTNQRY	**(element-no, no-of-elements, array)**
APL code	1022
GDDM RCP code	X'0C240001' (203685889)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the returned array. It must be in the range 1 through 7. For example, if it is 5, the first element in the returned array is the width.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes to be returned for the current partition. It is also the number of elements in the returned array.

array *(returned by GDDM) (an array of fullword integers)*
An array containing the **no-of-elements** attributes for the current partition. The array elements are:

1 The identifier of the current partition.

2 The row position of the top left-hand corner of the current partition in partition-set grid units.

3 The column position of the top left-hand corner of the current partition in partition-set grid units.

4 The depth of the current partition in partition-set grid units.

5 The width of the current partition in partition-set grid units.

6 The device partition identifier used for the current partition. A value of –1 indicates that none was specified.

7 The visibility of the current partition:

0 Not visible
1 Visible (the default)
2 Protected.

Description

Returns information about the current partition within the current partition set.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
```

PTNQUN

Function

To query unique partition identifier.

PTNQUN	**(partition-id)**
APL code	1026
GDDM RCP code	X'0C240102' (203686146)

Parameters

partition-id *(returned by GDDM) (fullword integer)*
An identifier for which no partition exists in the current partition set.

Description

Returns a value that has not so far been used for a partition identifier. The application program can then use this value as the identifier for a new partition set.

Principal errors

None.

PTNSEL

Function

To select a partition.

PTNSEL	**(partition-id)**
APL code	1024
GDDM RCP code	X'0C240100' (203686144)

Parameters

partition-id *(specified by user) (fullword integer)*
The identifier of the partition that is to become the current partition. It must be a partition from within the current partition set. A value of –1 does not change the current partition, but it forces the partition that is current at the time of the next output to become the active partition.

Description

Selects a partition from within the current partition-set to be the current partition.

The active partition is the one in which the cursor appears. When using real partitions, GDDM receives data only from the active partition. After input from the terminal, GDDM makes the partition that was active become the current partition. The operator may subsequently enter data into another partition, and, at the time of the next output, GDDM does not force the cursor back into the current partition **unless**:

- PTNSEL(–1) has been called, or
- A different partition is now current, or
- A screen refresh, for example, is required.

The current partition should always be visible when the screen is updated, for instance, by an ASREAD call. If an invisible partition is current, GDDM changes it to visible.

Note: For further information about current partitions, see the *GDDM Base Application Programming Guide*.

Principal errors

```
ADM3121 E  PARTITION n DOES NOT EXIST
ADM3122 E  PARTITION ID (n) IS INVALID
```

PTSCRT

Function

To create a partition set.

PTSCRT	**(partition-set-id, no-of-elements, array)**
APL code	1001
GDDM RCP code	X'0C200000' (203423744)

Parameters

partition-set-id *(specified by user) (fullword integer)*
The identifier for the new partition set. It must be unique and greater than zero (zero is reserved for the default partition set).

no-of-elements *(specified by user) (fullword integer)*
The number of attributes defined for the new partition set in **array**. It must be in the range 0 through 4.

array *(specified by user) (an array of fullword integers)*
The attributes for the new partition set. Possible values are:

1 The number of rows in the partition-set grid. If this is 0, the number of rows in the default partition set grid is used.

2 The number of columns in the partition-set grid. If this is 0, the number of columns in the default partition set grid is used.

3 A partition-control value defines the type of partition that can be created within the new partition set. The possible values are:

- **0** Real partitions on a device supporting hardware partitions, otherwise emulated partitions.

  This is the default if this element is not specified explicitly.
- **1** Emulated partitions, with emulated partition scrolling.

  This partition-control value is used for the default partition set.
- **2** Default partition only, with emulated partition scrolling.

4 Overlap control:

- **0** Partitions do not overlap (the default).
- **1** Partitions can overlap.

  Specifying that partitions can overlap always results in emulated partitions (even when the partitions do not actually overlap) on all devices including those that support real partitions.

Description

Creates a new partition set that also becomes the current partition set. This defines a grid of rows and columns to be used for specifying the size and position of all the partitions in the partition set. For details of the default partition set grid size, see FSQURY.

Only one partition set can be displayed at a time. Each partition set has its own set of partitions and pages. All partitions must belong to a partition set. Partitions can be created using the PTNCRT call, unless the partition control value is set to 2, in which case a default partition is implicitly created.

Once the size of the partition-set grid has been set, it cannot be changed.

Notes:

1. Information on the creation of partition sets, partitions, and pages can be found by using the call FSQURY.
2. A grid of 3 rows and 5 columns on a 24x80 display creates a grid size of 24/3 = 8 screen rows and 80/5 = 16 screen columns. The smallest partition (1x1 grid units) in this case is therefore 8 screen rows by 16 screen columns. Nonexact divisions of the screen cause an uneven grid, but the grid set always fills the screen.
3. The call PTNCRT (see “PTNCRT – Create a partition” on page 223) defines one of several partitions in the current partition set, using the partition set grid units, where (1,1) is the top left corner. The default partition is the whole grid area.
4. For information about variations in device support for partitions, see “Partitions” on page 241.

Principal errors

```
ADM3100 E  PARTITION SET n ALREADY EXISTS
ADM3102 E  PARTITION SET IDENTIFIER (n) IS INVALID
ADM3103 E  PARTITION SET n1 GRID DEPTH (n2) IS INVALID
ADM3104 E  PARTITION SET n1 GRID WIDTH (n2) IS INVALID
ADM3105 E  PARTITION SET n1 PARTITION CONTROL VALUE
           (n2) IS INVALID
ADM3106 E  PARTITION SET n1 OVERLAP CONTROL VALUE (n2)
           IS INVALID
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
```

PTSDEL

Function

To delete a partition set.

PTSDEL (partition-set-id)	
APL code	1004
GDDM RCP code	X'0C200101' (203424001)

Parameters

partition-set-id *(specified by user) (fullword integer)*
The partition set to be deleted. Possible values are:
–1 Delete the current partition set.
0 Delete the default partition set.

Description

Deletes a partition set and, consequently, any partitions within it.

If the current partition set is deleted, the most recently created partition set becomes current. If there is no such partition set, the default partition set becomes current.

Principal errors

```
ADM3101 E  PARTITION SET n DOES NOT EXIST
ADM3102 E  PARTITION SET IDENTIFIER (n) IS INVALID
```

PTSQPI

Function

To query partition identifiers.

PTSQPI (type, no-of-elements, array)	
APL code	1008
GDDM RCP code	X'0C200400' (203424768)

Parameters

type *(specified by user) (fullword integer)*
The category of partition. Possible values are:

1 All partitions
The identifiers of all the partitions for the current partition set returned in viewing priority order.

2 All invisible partitions
The identifiers of all the partitions for the current partition set, that are set to invisible, returned in viewing priority order.

no-of-elements *(specified by user) (fullword integer)*
The number of partition identifiers to be queried.

array *(returned by GDDM) (an array of fullword integers)*
An array of partition identifiers. If there are more elements in **array** than partitions in the specified category, the remaining elements are set to –1.

Description

Returns the identifiers of the partitions that fall into the category defined by the **type** parameter.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3196 E  TYPE n IS INVALID
```

PTSQPN

Function

To query partition numbers.

PTSQPN (element-no, no-of-elements, array)	
APL code	1009
GDDM RCP code	X'0C200401' (203424769)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the array. It must be in the range 1 through 2. For example, if the value is 2, the first element in the array is the number of invisible partitions.
no-of-elements *(specified by user) (fullword integer)*
The number of classifications to be returned in the array.
array *(returned by GDDM) (an array of fullword integers)*
An array of numbers of partitions, by category. Possible values are:
1 The number of partitions.
2 The number of invisible partitions.

Description

Returns the number of partitions, by category, in the current partition set.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
```

```
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
```

PTSQPP

Function

To query partition viewing priorities.

PTSQPP	**(order, partition-id, no-of-elements, array)**
APL code	1007
GDDM RCP code	X'0C200301' (203424513)

Parameters

order *(specified by user) (fullword integer)*
The order of viewing priority. Possible values are:
–**1** Descending order of viewing priority.
1 Ascending order of viewing priority.

partition-id *(specified by user) (fullword integer)*
The identifier of the partition relative to which the query is to take place.

A value of –1 can be used to return, within **array**, the identifiers of all partitions in descending (or ascending) order of priority.

no-of-elements *(specified by user) (fullword integer)*
The number of partition identifiers to be returned in the array parameter. It is the number of elements in **array**.

array *(returned by GDDM) (an array of fullword integers)*
The identifiers of the partitions.

The array is arranged as follows:

- The identifier of the partition that appears behind **partition-id** (if descending order), or in front of **partition-id** (if ascending order) is placed in the first element of the array. If **partition-id**=–**1**, this element contains the identifier of the partition with the highest viewing priority (if descending order), or the lowest viewing priority (if ascending order).
- The identifier of the partition that appears behind (or in front of) the partition identified in the first element is placed in the second element of the array.
- This is repeated until the identities of all the remaining partitions have been entered in the array.
- If there are more elements than partition identifiers to return, the remaining elements are set to a value of –1.

Description

Returns the identifiers of partitions in order of descending or ascending viewing priority, starting from a specified partition.

If descending order of viewing priority is specified, the identifiers of the lower-priority partitions that appear **behind** the specified partition are returned; if ascending order of viewing priority is specified, the identifiers of the higher priority partitions that appear **in front of** the specified partition are returned.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3121 E  PARTITION n DOES NOT EXIST
ADM3122 E  PARTITION ID (n) IS INVALID
ADM3193 E  ORDER n IS INVALID
```

PTSQRY

Function

To query partition set attributes.

PTSQRY	**(element-no, no-of-elements, array)**
APL code	1002
GDDM RCP code	X'0C200001' (203423745)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the returned array. It must be in the range 1 through 5. For example, if it is 2, the first element of the returned array contains the number of rows in the partition-set grid.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes of the current partition set to be returned in **array**. It is also the number of elements in the returned array and must be in the range 1 through 5.

array *(returned by GDDM) (an array of fullword integers)*
The attributes of the current partition set. The elements that can be returned are the:
1 Identifier of the current partition set.
2 Number of rows in the partition-set grid (depth).
3 Number of columns in the partition-set grid (width).
4 Partition control value.
5 Overlap control value.

Description

Returns the identifier of the current partition set, the size of its partition-set grid and the partition and overlap control values.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
```

PTSQUN

Function

To query unique partition set identifier.

PTSQUN	**(partition-set-id)**
APL code	1005
GDDM RCP code	X'0C200102' (203424002)

Parameters

partition-set-id *(returned by GDDM) (fullword integer)*
A value that has not been used for a partition-set identifier.

Description

Returns a value that has not so far been used for a partition-set identifier. The application program can then use this value as the identifier for a new partition set.

Principal errors

None.

PTSSEL

Function

To select a partition set.

PTSSEL	**(partition-set-id)**
APL code	1003
GDDM RCP code	X'0C200100' (203424000)

Parameters

partition-set-id *(specified by user) (fullword integer)*
The identifier of the partition set that is to become current. A value of zero makes the default partition set current.

The current partition for this partition set is the one that was current when this partition set was last selected.

Description

Makes a partition set the current partition set.

Principal errors

```
ADM3101 E  PARTITION SET n DOES NOT EXIST
ADM3102 E  PARTITION SET IDENTIFIER (n) IS INVALID
```

PTSSPP

Function

To set or reset partition viewing priorities.

PTSSPP	**(order, partition-id, no-of-elements, array)**
APL code	1006
GDDM RCP code	X'0C200300' (203424512)

Parameters

order *(specified by user) (fullword integer)*
The order of viewing priority. Possible values are:

–**1** Descending order of viewing priority.
1 Ascending order of viewing priority.

partition-id *(specified by user) (fullword integer)*
The identifier of the partition relative to which the reordering is to take place.

A value of –1 can be used to set the first partition in **array** as either the highest in viewing priority (if descending order), or lowest in viewing priority (if ascending order).

no-of-elements *(specified by user) (fullword integer)*
The number of partitions to be reordered in viewing priority. It is the number of elements in **array**.

array *(specified by user) (an array of fullword integers)*
The identifiers of the partitions to be reordered in viewing priority. A value of –1 in any element terminates the reordering process.

Description

Sets the relative viewing priorities of the specified partitions.

The partitions whose identifiers are specified in **array** are reordered in viewing priority relative to the specified partition, **partition-id**.

The reordering process is as follows:

- The elements of the array parameter are processed one at a time.

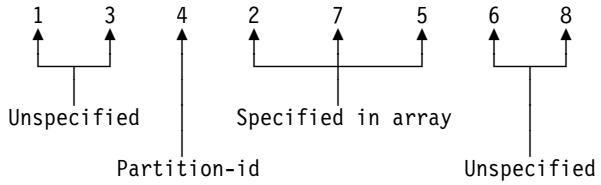
- The partition identified by the first element is placed behind (if descending order) or in front of (if ascending order) the specified partition, **partition-id**.
- The partition identified by the second element is placed behind (or in front of) the partition identified in the first element.
- This process is repeated until all the elements of the array parameter are processed, or until a –1 element is found.
- The priorities of partitions that are not specified in **array** remain unchanged with respect to **partition-id** and each other. Unspecified partitions with a higher priority than **partition-id** retain viewing priorities higher than both **partition-id** and all reordered partitions. Unspecified partitions with a lower priority than **partition-id** retain viewing priorities lower than both **partition-id** and all reordered partitions.

Example: Eight partitions are arranged in descending order of priority:

```
1     2     3     4     5     6     7     8
```

A call to PTSSPP is made with **order**=–1, **partition-id**=4, **no-of-elements**=4, and **array** containing {2 7 5 -1}.

The priorities are reordered thus:



Principal errors

```
ADM3116 E  PARTITION n ALREADY PROCESSED
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3121 E  PARTITION n DOES NOT EXIST
ADM3122 E  PARTITION ID (n) IS INVALID
ADM3193 E  ORDER n IS INVALID
```

SPINIT

Function

To initialize GDDM with SPIB.

Note: SPINIT is an alternative to FSINIT and if used, it must be the first GDDM statement to be executed. It can be invoked only by means of the system programmer interface (SPI).

SPINIT	**(spib-block)**
APL code	115
GDDM RCP code	X'00050000' (327680)

Parameters

spib-block (specified by user) (32-byte character string)
A table giving control information. The contents of this table are processed by GDDM during initialization. Subsequent changes to the contents do not affect GDDM processing. The storage containing the table can be released after initialization has been completed. For details of the system programmer interface block (SPIB), see “The system programmer interface block” on page 432.

Description

Initialize GDDM processing using the System Programmer Interface.

The SPINIT call allows parameters to be passed by a SPIB (SPI Initialization Block). The SPIB contains a number of address words that can be set by an application program.

If the SPINIT call is actually issued in 24-bit mode, GDDM clears the top byte (minus the top bit) of each address word that it processes.

Principal errors

None.

SPMXMP

Function

To control the use of mixed fields by mapping.

Note: This function can only be invoked by the system programming interface (SPI) and should not be used in new application programs. It is included only for compatibility with releases of GDDM before Version 2 Release 2, and is liable to be withdrawn at any time after Version 2 Release 3. The mapping calls, MSCPOS through MSREAD, support mixed fields, and the support they provide should be used rather than this call.

SPMXMP	**(mixed)**
APL code	438
GDDM RCP code	X'0C081401' (201856001)

Parameters

mixed *(specified by user) (fullword integer)*
The status of the mixed fields. Possible values are:
0 Alphanumeric fields not mixed.
1 Alphanumeric fields mixed.

Description

Controls the use of mixed fields in maps when using a primary device that supports them – for example, an IBM 5550-family Multistation.

This control affects all mapped alphanumeric fields; the application is responsible for checking that mixed data placed in the fields, that has not been generated using an IBM 5550-family work station, meets the mixed field requirements.

Principal errors

None.

SSQF

Function

To query a symbol set on auxiliary storage.

SSQF	**(symbol-set-name, length, type)**
APL code	212
GDDM RCP code	X'0C040100' (201588992)

Parameters

symbol-set-name *(specified by user and returned by GDDM) (8-byte character string)*
The name (left-justified) of the symbol set or DBCS ward to be queried. If the name ends with the substitution character "." (for a DBCS ward name, the substitution character is just before the ward digits), GDDM tries to locate the symbol-set file with a name formed by changing the "." to one appropriate to the current device.

GDDM first tries to locate a symbol set whose name is formed by replacing the substitution character with that appropriate for the current alphanumerics cell size. (See the information for **code=0** under FSQURY and also see the symbol set naming convention described in Chapter 7, "Symbol sets" on page 261).

If GDDM cannot find such a symbol set, it then looks for one whose name is formed by replacing its substitution character with the one that is appropriate for the current graphics cell size. (See the information for **code=2** under FSQURY and also see symbol set naming convention described in Chapter 7, "Symbol sets" on page 261.

If a symbol set with a substituted name is located, this parameter contains the changed name (which is returned by GDDM).

length *(returned by GDDM) (fullword integer)*
The length of storage required to contain the symbol-set definitions.

type *(returned by GDDM) (fullword integer)*
The type of symbol definitions. Possible values are:

0 An image symbol set that can be loaded into the device.

1 An image symbol set that cannot be loaded at the device, because the device does not support symbol sets in this format.

2 A vector symbol set.

Description

Returns the length of the definitions and the type of a named symbol set.

This function is provided to allow the application program to find out how much storage is needed to perform a subsequent SSREAD request, and also to find out whether the definitions are for an image symbol set or a vector symbol set.

Principal errors

```
ADM0113 E  SYMBOL SET 'a' HAS UNIDENTIFIABLE TYPE
ADM0124 E  FOR SYMBOL SET 'a' THE DEFINITION LENGTH n
           IS TOO SHORT
```

SSREAD

Function

To read a symbol set from auxiliary storage.

SSREAD	**(symbol-set-name, length, data)**
APL code	213
GDDM RCP code	X'0C040B00' (201591552)

Parameters

symbol-set-name *(specified by user) (8-byte character string)*
The name (left-justified) of the symbol set or DBCS ward to be read from auxiliary storage.

If the name ends with the substitution character "." (for a DBCS ward name, the substitution character is just before the ward digits), GDDM tries to locate the symbol set file

with a name formed by changing the "." to one appropriate to the current device. The **symbol-set-name** parameter is not itself changed upon return. For information on the character that is substituted, see Chapter 7, "Symbol sets" on page 261.

GDDM first tries to locate a symbol set whose name is formed by replacing the substitution character with that appropriate for the current alphanumerics cell size. (See the information for **code=0** under FSQURY and also see the symbol set naming convention described in Chapter 7, "Symbol sets" on page 261.

If GDDM cannot find such a symbol set, it then looks for one whose name is formed by replacing its substitution character with the one that is appropriate for the current graphics cell size. (See the information for **code=2** under FSQURY and also see the symbol set naming convention described in Chapter 7, "Symbol sets" on page 261).

length *(specified by user) (fullword integer)*
The length of storage provided for the **data** parameter. The SSQF call can be used to determine how much storage is needed for the symbol set.

data *(returned by GDDM) (character)*
Contains the symbol set as read from auxiliary storage.

Description

Reads a set of symbol definitions from auxiliary storage. The symbol set can be either an image symbol set or a vector symbol set.

The symbol set does not remain in GDDM storage, and is not available for use by it.

Principal errors

```
ADM0114 E  LENGTH n IS INSUFFICIENT FOR SYMBOL SET 'a'
ADM0115 E  SYMBOL SET 'a' LENGTH n IS INVALID
```

SSWRT

Function

To write a symbol set to auxiliary storage.

SSWRT	**(symbol-set-name, length, data)**
APL code	214
GDDM RCP code	X'0C040B01' (201591553)

Parameters

symbol-set-name *(specified by user) (8-byte character string)*
The name (left-justified) to be assigned to the symbol set or DBCS ward when written to auxiliary storage. No attempt is made to change the name, so it must not include the substitution character ".", either at the end (for SBCS) or just before the ward digits (for DBCS ward).

length *(specified by user) (fullword integer)*
The length of storage provided for **data**.

data *(specified by user) (character)*
Contains the symbol set definitions to be written.

Description

Writes a set of symbol definitions to auxiliary storage. The symbol set can be either an image symbol set or a vector symbol set.

The symbol set does not remain in GDDM storage, and is not available for use by it.

Principal errors

```
ADM0127 E  SYMBOL SET NAME 'a' IS INVALID
```

WSCRT

Function

To create an operator window.

WSCRT	**(operator-window-id, no-of-elements, array, length, string)**
APL code	1040
GDDM RCP code	X'0C2C0000' (204210176)

Parameters

operator-window-id *(specified by user) (fullword integer)*
The identifier of the new operator window. It must be greater than 0 (0 is reserved for the parent operator window) and unique for the primary device.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes defined for the new operator window. It must be in the range 0 through 10.

array *(specified by user) (an array of fullword integers)*
The attributes for the new operator window. If any attribute is not specified, or is specified as 0, the default value is used. The attributes are:

1 The address of the coordination exit routine. The default value is zero. Specifying a 0 for this element indicates that the window can only be used to window a virtual device created by *this* instance of GDDM; it cannot be used to window devices opened with DSOPEN during other instances of GDDM.

2 Exit token. A parameter to be passed to the coordination exit in the UXTOKEN field of UXBLOCK. The default value is zero.

3 The number of rows in the virtual screen. The default is the real screen depth.

4 The number of columns in the virtual screen. The default is the real screen width.

5 The row position of the top left-hand corner of the new operator window on the real screen. This is the position of the top left-hand corner of the window contents, not the position of the window border. The default position is row 1.

6 The column position of the top left-hand corner of the new operator window on the real screen. This is the position of the top left-hand corner of the window contents, not the position of the window border. The default position is column 1.

7 The number of rows in the new operator window. This does not include any rows occupied by the window border. The default is the smaller of the number of rows on the real device, and the number of rows on the virtual screen.

8 The number of columns in the new operator window. This does not include any columns occupied by the window border. The default is the smaller of the number of columns on the real device, and the number of columns on the virtual screen.

9 The row position of the top left-hand corner of the new operator window on the virtual screen. The default position is row 1.

10 The column position of the top left-hand corner of the new operator window on the virtual screen. The default position is column 1.

length *(specified by user) (fullword integer)*
The length of **string** in bytes. It must be in the range 0 through 32, and must not be greater than the width of the real screen or the virtual screen.

string *(specified by user) (character)*
The title to be incorporated into the border of the new operator window. If a length of 0 is specified, the default title is used, namely “XXXXXX”.

Description

Creates a new operator window to be used to window the primary device. Operator windows can only be created for devices opened with the DSOPEN processing option WINDOW.

The new operator window becomes the current operator window, and the candidate operator window.

Subsequent requests to open this (same device name list) device, made from this or any other instance of GDDM, results in the opening of a virtual device, which appears in the candidate operator window.

The rules governing virtual screen and operator window size and position are:

- The virtual screen can be from 1 through 255 columns wide, and from 1 through 255 rows deep.
- When a device is opened with the WINDOW processing option, the real screen size used is the alternate screen size, if there is one, otherwise the default screen size is used.
- The maximum width of the operator window is the smaller of the virtual screen width and the real screen width. The maximum depth of the operator window is the smaller of the virtual screen depth and the real screen depth.
- The operator window must always be positioned so that it is completely on the real screen, although part or all of the window border can be off the screen.
- The operator window must always be scrolled so that it is completely on the virtual screen.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3181 E  OPERATOR WINDOW n ALREADY EXISTS
ADM3182 E  OPERATOR WINDOW ID n IS INVALID
ADM3183 E  OPERATOR WINDOW SCREEN ROW NUMBER n IS
           INVALID
ADM3184 E  OPERATOR WINDOW SCREEN COLUMN NUMBER n IS
           INVALID
ADM3185 E  OPERATOR WINDOW DEPTH n IS INVALID
ADM3186 E  OPERATOR WINDOW WIDTH n IS INVALID
ADM3187 E  OPERATOR WINDOW VIRTUAL SCREEN ROW NUMBER n
           IS INVALID
ADM3188 E  OPERATOR WINDOW VIRTUAL SCREEN COLUMN NUMBER
           n IS INVALID
ADM3189 E  VIRTUAL SCREEN DEPTH n IS INVALID
ADM3190 E  VIRTUAL SCREEN WIDTH n IS INVALID
ADM3191 E  TITLE LENGTH n IS INVALID
```

WSDEL

Function

To delete operator window.

WSDEL	**(operator-window-id)**
APL code	1041
GDDM RCP code	X'0C2C0100' (204210432)

Parameters

operator-window-id *(specified by user) (fullword integer)*
The identifier of the operator window to be deleted. A value of –1 deletes the current operator window. The parent operator window, which has an identifier of zero, cannot be deleted.

Description

Deletes an operator window and, consequently, closes any virtual devices associated with it.

If the current operator window is deleted, the highest viewing priority operator window becomes current.

If the candidate operator window is deleted, the parent operator window becomes the candidate operator window.

Virtual devices in another instance of GDDM which have been associated with the operator window are not closed. They must be closed *before* deleting the operator window. If they are not closed, the results of subsequent operations on these virtual devices are not defined.

Principal errors

```
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3182 E  OPERATOR WINDOW ID n IS INVALID
ADM3192 E  OPERATOR WINDOW n DOES NOT EXIST
ADM3195 E  THE PARENT OPERATOR WINDOW CANNOT BE DELETED
```

WSIO

Function

To control windowed device input/output.

WSIO	**(operator-window-id)**
APL code	151
GDDM RCP code	X'0C100008' (202375176)

Parameters

operator-window-id *(returned by GDDM) (fullword integer)*
The identifier of the operator window that satisfied the I/O request. If hierarchies of operator windows and virtual devices are being used, and the operator window that satisfied the I/O request is at a lower level in the hierarchy than the primary device, the identifier of its ancestor operator window belonging to this device is returned.

Description

Performs output and possibly input for a windowed device.

If any operator window is requesting FSFRCE (either explicitly or implicitly), WSIO performs output but does not wait for input, and returns the identifier of the operator window for which I/O was completed, or its parent operator window identifier, according to the operator window hierarchy.

If no operator windows are requesting FSFRCE, the WSIO call performs the type of I/O requested by the highest viewing priority operator window, and returns the identifier of the operator window for which I/O was completed, or its parent operator window identifier depending upon the operator window hierarchy.

If user control is used to change the highest viewing priority operator window, the type of I/O requested by this window is performed.

WSIO performs an implicit WSSEL for all the parent windows of the operator window that satisfied the I/O request, and also for the operator window that satisfied the I/O request.

Principal errors

```
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
```

WSMOD

Function

To modify the current operator window.

WSMOD	**(element-no, no-of-elements, array, length, string)**
APL code	1043
GDDM RCP code	X'0C2C0200' (204210688)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the array. It must be in the range 1 through 6.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes to be modified for the current operator window. It is the number of elements in the array and must be in the range 0 through 6.

array *(specified by user) (an array of fullword integers)*
The new attributes for the current operator window. If any attribute is not specified, or is specified as –1, the existing value is unchanged. If any attribute is specified as 0, the default value is used. The attributes are:

1 The row position of the top left-hand corner of the current operator window on the real screen.

2 The column position of the top left-hand corner of the current operator window on the real screen.

3 The number of rows in the current operator window. This does not include any rows occupied by the window border.

4 The number of columns in the current operator window. This does not include any columns occupied by the window border.

5 The row position of the top left-hand corner of the current operator window on the virtual screen.
6 The column position of the top left-hand corner of the current operator window on the virtual screen.

length *(specified by user) (fullword integer)*
The length of **string** in bytes. A value of zero leaves the existing string unchanged.

string *(specified by user) (character)*
The title to be incorporated into the border of the current operator window.

Description

Modifies the attributes of the current operator window.

The coordination exit address, exit token, and virtual screen size cannot be modified. Modifications to the other attributes and the window title changes the appearance of the operator window on the device at the next device I/O.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3183 E  OPERATOR WINDOW SCREEN ROW NUMBER n IS
           INVALID
ADM3184 E  OPERATOR WINDOW SCREEN COLUMN NUMBER n IS
           INVALID
ADM3185 E  OPERATOR WINDOW DEPTH n IS INVALID
ADM3186 E  OPERATOR WINDOW WIDTH n IS INVALID
ADM3187 E  OPERATOR WINDOW VIRTUAL SCREEN ROW NUMBER n
           IS INVALID
ADM3188 E  OPERATOR WINDOW VIRTUAL SCREEN COLUMN NUMBER
           n IS INVALID
ADM3191 E  TITLE LENGTH n IS INVALID
```

WSQRY

Function

To query the current operator window.

WSQRY	**(element-no, no-of-elements, array, actual-length, length, string)**
APL code	1044
GDDM RCP code	X'0C2C0300' (204210944)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the returned array. It must be in the range 1 through 11.

no-of-elements *(specified by user) (fullword integer)*
The number of attributes of the current operator window to be returned in the array. It must be in the range 0 through 11.

array *(returned by GDDM) (an array of fullword integers)*
The attributes of the current operator window. They are:
1 The identifier of the operator window.
2 The address of the coordination exit routine.
3 Exit token.
4 The number of rows in the virtual screen.
5 The number of columns in the virtual screen.
6 The row position of the top left-hand corner of the operator window on the screen.
7 The column position of the top left-hand corner of the operator window on the screen.
8 The number of rows in the operator window. This does not include any rows occupied by the window border.
9 The number of columns in the operator window. This does not include any columns occupied by the window border.
10 The row position of the top left-hand corner of the operator window on the virtual screen.
11 The column position of the top left-hand corner of the operator window on the virtual screen.

actual-length *(returned by GDDM) (fullword integer)*
The length of the operator window title in bytes.

length *(specified by user) (fullword integer)*
The length of **string** in bytes.

string *(returned by GDDM) (character)*
The title incorporated into the border of the operator window.

If the actual length is greater than the string length, the returned title is truncated, and if the string length is greater than the actual length, the returned title is padded with blanks (X'40').

Description

Returns the identifier and attributes of the current operator window.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3191 E  TITLE LENGTH n IS INVALID
```

WSQUN

Function

To query unique operator window identifier.

WSQUN	(operator-window-id)
APL code	1045
GDDM RCP code	X'0C2C0400' (204211200)

Parameters

operator-window-id *(returned by GDDM) (fullword integer)*
A value that is not currently in use for an operator window identifier.

Description

Returns a value that is not currently in use for an operator window identifier.

Principal errors

```
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
```

WSQWI

Function

To query operator window identifiers.

WSQWI	(type, no-of-elements, array)
APL code	1046
GDDM RCP code	X'0C2C0500' (204211456)

Parameters

type *(specified by user) (fullword integer)*
The category of operator window:
1 All operator windows
The identifiers of all the operator windows for the primary device are returned in viewing priority order.
no-of-elements *(specified by user) (fullword integer)*
The number of operator window identifiers to be queried.
array *(returned by GDDM) (an array of fullword integers)*
An array of operator window identifiers. If there are more elements in array than operator windows in the specified category, the remaining elements are set to –1.

Description

Returns the identifiers of the operator windows that fall into the category defined by the **type** parameter.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3196 E  TYPE n IS INVALID
```

WSQWN

Function

To query operator window numbers.

WSQWN	(element-no, no-of-elements, array)
APL code	1050
GDDM RCP code	X'0C2C0600' (204211712)

Parameters

element-no *(specified by user) (fullword integer)*
The number of the first element in the array. It must be 1.
no-of-elements *(specified by user) (fullword integer)*
The number of numbers to be returned. It is also the number of elements in the returned array. It must be 0 or 1.
array *(returned by GDDM) (an array of fullword integers)*
An array of numbers of operator windows, by category:
1 All operator windows
The number of operator windows for the primary device is returned.

Description

Returns the number of operator windows, by category, on the primary device.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3119 E  ELEMENT NUMBER (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
```

WSQWP

Function

To query operator window viewing priorities.

WSQWP	(order, operator-window-id, no-of-elements, array)
APL code	1049
GDDM RCP code	X'0C2C0700' (204211968)

Parameters

order *(specified by user) (fullword integer)*
The order of viewing priority. Possible values are:
–**1** Descending order of viewing priority.
1 Ascending order of viewing priority.

operator-window-id *(specified by user) (fullword integer)*
The identifier of the operator window relative to which the query is to take place.

A value of –1 can be used to return, within **array**, the identifiers of all operator windows in descending or ascending order of viewing priority.

no-of-elements *(specified by user) (fullword integer)*
The number of operator window identifiers to be returned in the array parameter. It is the number of elements in **array**.

array *(returned by GDDM) (an array of fullword integers)*
The identifiers of the operator windows.

The array is arranged as follows:

- The identifier of the operator window that appears behind **operator-window-id** (if descending order), or in front of **operator-window-id** (if ascending order) is placed in the first element of the array. If **operator-window-id**=–**1**, this element contains the identifier of the operator window with the highest viewing priority (if descending order), or the lowest viewing priority (if ascending order).
- The identifier of the operator window that appears behind (or in front of) the operator window identified in the first element is placed in the second element of the array.
- This is repeated until the identities of all the remaining operator windows have been entered in the array.
- If there are more elements than operator window identifiers to return, the remaining elements are set to a value of –1.

Description

Returns the identifiers of operator windows in order of descending or ascending viewing priority, starting from a specified operator window.

If descending order of viewing priority is specified, the identifiers of the lower-priority operator windows that appear **behind** the specified operator window are returned; if ascending order of viewing priority is specified, the identifiers of the higher priority operator windows that appear **in front of** the specified operator window are returned.

Principal errors

```
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3182 E  OPERATOR WINDOW ID n IS INVALID
ADM3192 E  OPERATOR WINDOW n DOES NOT EXIST
ADM3193 E  ORDER n IS INVALID
```

WSSEL

Function

To select an operator window.

WSSEL	(operator-window-id)
APL code	1051
GDDM RCP code	X'0C2C0800' (204212224)

Parameters

operator-window-id *(specified by user) (fullword integer)*
The identifier of the operator window to be made current.

Description

Selects an operator window from the primary device to be the current operator window.

The current operator window is the one to which WSQRY calls and WSMOD calls apply. The operator window that is most recently selected, across all devices, is the candidate window for windowing the device.

Principal errors

```
ADM3180 E  PRIMARY DEVICE NOT OPENED WITH THE WINDOW
           PROCOPT
ADM3182 E  OPERATOR WINDOW ID n IS INVALID
ADM3192 E  OPERATOR WINDOW n DOES NOT EXIST
```

WSSWP

Function

To set or reset operator window viewing priorities.

WSSWP	**(order, operator-window-id, no-of-elements, array)**
APL code	1052
GDDM RCP code	X'0C200900' (204212480)

Parameters

order *(specified by user) (fullword integer)*
The order of viewing priority. Possible values are:

–**1** Descending order of viewing priority.
1 Ascending order of viewing priority.

operator-window-id *(specified by user) (fullword integer)*
The identifier of the operator window relative to which the reordering is to take place.

A value of –1 can be used to set the first operator window in **array** as either the highest in viewing priority (if descending order), or lowest in viewing priority (if ascending order).

no-of-elements *(specified by user) (fullword integer)*
The number of operator windows to be reordered in viewing priority. It is the number of elements in **array**.

array *(specified by user) (an array of fullword integers)*
The identifiers of the operator windows to be reordered in viewing priority. A value of –1 in any element terminates the reordering process.

Description

Sets the relative viewing priorities of the specified operator windows.

The operator windows whose identifiers are specified in **array** are reordered in viewing priority relative to the specified operator window, **operator-window-id**.

The reordering process is as follows:

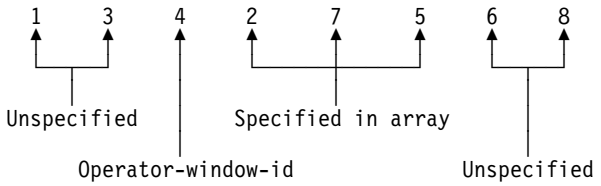
- The elements of the array parameter are processed one at a time.
- The operator window identified by the first element is placed behind (if descending order) or in front of (if ascending order) the specified operator window, **operator-window-id**.
- The operator window identified by the second element is placed behind (or in front of) the operator window identified in the first element.
- This process is repeated until all the elements of the array parameter are processed, or until a –1 element is found.
- The priorities of operator windows that are not specified in **array** remain unchanged with respect to **operator-window-id** and each other. Unspecified operator windows with a higher priority than **operator-window-id** will retain viewing priorities higher than both **operator-window-id** and all reordered operator windows. Unspecified operator windows with a lower priority than **operator-window-id** will retain viewing priorities lower than both **operator-window-id** and all reordered operator windows.

Example: Eight operator windows are arranged in descending order of priority:

```
1    2    3    4    5    6    7    8
```

A call to WSSWP is made with **order**=–1, **operator-window-id**=4, **no-of-elements**=4, and **array** containing {2 7 5 -1}.

The priorities are reordered thus:



Principal errors

```
ADM3116 E  PARTITION n ALREADY PROCESSED
ADM3118 E  NUMBER OF ELEMENTS (n) IS INVALID
ADM3121 E  PARTITION n DOES NOT EXIST
ADM3122 E  PARTITION ID (n) IS INVALID
ADM3193 E  ORDER n IS INVALID
```

Chapter 4. Device variations

The input and output devices supported by GDDM vary widely in their capabilities and in the datastreams required to drive them. The GDDM programming interface shields application programs from many of these device variations. This permits application programs to be written with a degree of device independence, which in turn permits these programs to use a range of different devices without modification to the programs.

The degree of device independence provided by GDDM varies, as follows:

- In some cases GDDM provides complete independence by emulating a function which a device may not have. For example, GDDM emulates graphics data on a 3279 display and image data on a 3472G display.
- In other cases, GDDM effectively ignores an application's requests for functions not available in the device, without rejecting the application calls. In general, it does this for functions that it cannot emulate and which, if ignored, would not normally affect a typical application program. For example, GDDM ignores requests for alphanumeric field highlighting on devices that do not support it.
- In other cases, GDDM is unable to emulate a function not available in a device. However, to ignore requests for the function would probably affect a typical application program adversely. In such a situation, GDDM responds to the function call with an error message. An example of this is a request to enable input from a mouse when the display device does not have one.

For an application program to be able to exploit the full capability of all devices available through GDDM, the program must request different functions, depending upon what functions are available. To support this, GDDM provides a number of query calls, notably FSQURY, which tell the application what function is available.

This chapter describes, for each broad group of GDDM functions, the devices or groups of devices that do not conform to the general description of the function.

Where a function call is subject to device variations, a reference to the appropriate section of this chapter is included in the description of that function call in chapter 3.

Operator windows, partitions, primary, alternate, and dual screens

Operator windows

GDDM operator windows are available only on display devices. They are not available on the 5080 or 6090 graphics system.

Partitions

- On the ***5080 or 6090 Graphics System***, there is a maximum of 15 partitions visible at any one time. The ones that are visible are the 15 most current; that is, the lowest priority partition disappears as each new partition is created, over a value of 15. If partitions are then deleted, the low priority partitions reappear in the reverse order; that is "last out, first in".
- On all other display devices, GDDM can show a maximum of 254 emulated partitions.
- The ***3193, 3290***, and ***8775 displays*** support a number of real nonoverlapping partitions.

Primary and alternate screen sizes

If you use a device with primary and alternate screen sizes (for example, the 3278 Model 5), you can select the required size in the FSPCRT call. Note that in order to obtain the primary screen size, you must suppress User Control, using the CTLMODE procopt, for example, with the nickname:

```
ADMMNICK PROCOPT=((CTLMODE,NO))
```

In order to obtain the primary screen size, use of the WINDOW procopt, partition calls and partition set calls must also be avoided.

Dual screen devices

GDDM alphanumerics are shown separately from graphics, either on a separate 3270-family display unit, or on the 5080 or 6090 screen switched into 3270 mode by the user.

These restrictions apply to the ***5080 Graphics System***:

- The 5080 or 6090 is only supported on VM/CMS or MVS/TSO systems. When using the GDDM image interface, there are no restrictions other than that imposed by the 5080 or 6090 buffer size, which may limit the size of the image data.
- When a 5080 or 6090 Graphics System is queried, the graphics characteristics **(code=2)** refers to the 5080 or 6090 display, and the other information refers to the associated 3270 device.

Dual screen size

On a ***dual-screen 3270-PC/GX workstation***, or ***device running under GDDM-PCLK and GDDM-OS/2 Link***, or on the ***5080 or 6090 Graphics System***, the graphics field is defined in row and column coordinates with respect to the alphanumerics screen. The portion of the graphics screen that is used to display graphics is given by defining a "virtual" grid on the graphics screen, the width and depth of which are the same as the number of columns and rows on the alpha-

numerics screen, and then the placing of the graphics field with respect to this “virtual” grid.

Device-specific saved pictures

GDF saved as 2-byte integers

GDF can be saved as 2-byte integers using the GSSAVE call. Coordinate data cannot be saved as 2-byte integers from the ***5080 or 6090 Graphics Systems***, or ASCII graphics devices. Coordinate data can only be saved as 2-byte integers from ***family 4 page printers*** if the device token or processing option specifies GOCA output.

ADMSAVE files

ADMSAVE files can only be created by the FSSAVE call and shown by the FSSHOW or FSSHOR calls on ***family 1*** devices. Furthermore, they cannot be created from, or shown on, the ***5080 or 6090 Graphics Systems***, or ASCII graphics devices.

To show an ADMSAVE file, the picture must have been created and saved on a device that is compatible with the device on which the picture is to be shown.

If a picture is saved when running against a device for which compressed data streams are being generated, and an attempt is made to reshow this picture when running against a device for which noncompressed data streams are necessary, undefined results occur.

On the ***3270-PC/G*** and ***3270-PC/GX workstations***, if FSSHOW is called, a device that is customized differently (for example, with different screen sizes) is treated as an incompatible device.

The data stream displayed is always in unretained mode, so the amount of segment storage available is not a factor when saving pictures. However, if user pattern sets are used by the saved-data stream, there must be enough symbol-set storage in the device to load these pattern sets; otherwise, undefined results occur.

Note: Because the displayed segments are not retained, local functions (such as local scrolling or change screen) can lead to a loss of the graphics data from the screen.

Screen redraw

The picture is wholly, or partly, reconstructed (and graphics primitives outside segments may be lost) under these device-dependent conditions:

- On ***3270-PC/G, /GX, and /AT workstations*** and ***5550-family workstations***, when local workstation functions are used.
- On ***3270-PC/G, /GX, and /AT workstations, IBM 3179-G***, ***3192-G***, and ***3472-G color graphics displays***, and ***5550-family workstations***:
  - When the screen has to be refreshed (for example, after the subsystem has taken over the screen to display a message, or after PA3 has been pressed).
  - If the visibility or highlighting attribute of a segment is changed by the GSSATS call.
- On ***3270-PC/G, /GX, and /AT workstations,*** when changing to and from unretained mode, where the workstation has been customized with insufficient segment storage for the displayed picture.
- On the ***5080 Graphics System***, when the picture is too large for the 5080 Graphics System display buffer, all data not contained in segments is deleted.

Graphics primitives outside segments

On ***3179-G***, ***3192-G***, and ***3472-G color graphics displays***, ***3270-PC/G, /GX, or AT workstations***, and ***5550-family multistations***, it is recommended that GDDM primitives (graphics calls) be used only after a segment has been opened. This is because, on these devices, the primitives are discarded after they have been displayed on the screen. Therefore, when any local operation takes place at the device (for example, if the screen is scrolled or if a system-issued message is displayed), GDDM does not refresh the primitives, which are consequently lost. Also, when an overlapping partition is deleted or moved, temporary data that was overlapped by the partition is lost.

To avoid these situations, use a GSSEG call to open a segment before using GDDM primitives calls.

Programmed symbol sets (PS) and graphics text

Character mode 1 (hardware character sets)

Hardware character sets are subject to the following device-dependent considerations:

- On a devices such as a ***3279***, it must have been loaded previously as a PS set.
- On ***3179-G, 3192-G***, and ***3472-G color graphics displays*** and on ***3270-PC/G or 3270-PC/GX workstations***, the symbol-set identifier is used to select an image symbol set that has been loaded previously by a GSLSS or GSDSS call, or a PS set that has been loaded previously as a PS set by a PSDSS, PSLSS, or PSRSV call.
- For ***plotters, 3800-3 printers***, and ***3820 printers***, mode-1 character strings are drawn using the default vector character set.

- For ***4250 printers***, mode-1 character strings are drawn using either the default vector character set or the selected printer font.

Character mode 2 (image symbol sets)

Image symbol sets are subject to the following device-dependent considerations:

- On ***3179-G, 3192-G***, and ***3472-G color graphics displays***, and on ***3270-PC/G and 3270-PC/GX workstations***, the device itself controls which symbol set is used to display the string if both a PS set (loaded by a PSDSS, PSLSS, or PSLSSC call) and an image symbol set (loaded by a GSLSS or GSDSS call) with the required identifier have been loaded previously. Equally, the workstation controls which symbol set is used if only a PS set with the required identifier is loaded.
- For ***plotters***, character set 0 characters are scaled, using hardware, to best fit the character box.

PS stores and device cell-size dimensions

Symbol sets that are to be loaded into PS stores must have the same matrix dimensions as the device character cell. These are shown in Table 4.

FSCHEK call

Support for the FSCHEK call is subject to the following device-dependent limitations:

- On the ***5080 Graphics System***, The FSCHEK call is ignored. The amount of data that can be displayed is limited by the size of the display-list buffer in the 5080 system. If the buffer overflows, then the operation is tried once more. If the condition persists, a warning message is returned to the application.
- On ***3179-G***, ***3192-G***, and ***3472-G colour graphics displays***, and ***3270-PC/G and 3270-PC/GX workstations***, the FSCHEK call is ignored because PS overflow cannot occur on these devices.
- On ***5550-family Multistations*** and ***devices running under GDDM-PCLK or GDDM-OS/2 Link***, the FSCHEK call is ignored.

Table 4 (Page 1 of 2). Device cell-size dimensions

Device	Models	Character cell width	Character cell height
3179-G and 3192-G color display	All	9	12 (see note 1)
3472-G color display	24 row screen	9	21
3472-G color display	32 row screen	9	16
3268 printer	2C	10	8
3270-PC display	All	9	14
3270-PC/G workstation	All	9	10 or 16
3270-PC/GX workstation	All	12	20
3278 display	2, 3	9	16
3278 display	4	9	12
3279 display	2B, 3B	9	12
3287 printer	All	10	8
3290 information panel display	All	9	16
3812 printer model 2	All	24	30
3816 printer	All	24	30
3112, 3116, 3912, 3916, and 4028 printers	All	30	37 (see note 2)
4224 and all 4230 printers	All	20	18
4234 printer	11	12	16
PCLK adapter with CGA card	24-row screen	8	8
PCLK adapter with EGA card (64 K)	24-row screen	8	8
PCLK adapter with EGA card (128+ K)	24-row screen	8	14
PCLK adapter with EGA card (128+ K)	32-row screen	8	11
PCLK adapter with MCGA card	24-row screen	8	19

Table 4 (Page 2 of 2). Device cell-size dimensions

Device	Models	Character cell width	Character cell height
PCLK adapter with MCGA card	32-row screen	8	14
PCLK adapter with MCGA card	43-row screen	8	10
PCLK adapter with VGA card	24-row screen	8	19
PCLK adapter with VGA card	32-row screen	8	14
PCLK adapter with VGA card	43-row screen	8	10
PCLK with 8514/A + 8503, 8504, 8512, 8513, 8516, or 8518	24-row screen	8	19
PCLK with 8514/A + 8503, 8504, 8512, 8513, 8516, or 8518	32-row screen	8	14
PCLK with 8514/A + 8503, 8504, 8512, 8513, 8516, or 8518	43-row screen	8	10
PCLK with 8514/A + 8507, 8514, or 8515	24-row screen	12	30
PCLK with 8514/A + 8507, 8514, or 8515	32-row screen	12	23
PCLK with 8514/A + 8507, 8514, or 8515	43-row screen	12	17
PCLK with 8514/A + 8507, 8514, or 8515	27-row x 132-col screen	7	24
PCLK adapter with XGA card + 8503, 8504, 8512, 8513, 8516, or 8518	24-row screen	8	19
PCLK adapter with XGA card + 8503, 8504, 8512, 8513, 8516, or 8518	32-row screen	8	14
PCLK adapter with XGA card + 8503, 8504, 8512, 8513, 8516, or 8518	43-row screen	8	10
PCLK adapter with XGA card + 8507, 8514, or 8515	24-row screen	12	30
PCLK adapter with XGA card + 8507, 8514, or 8515	32-row screen	12	23
PCLK adapter with XGA card + 8507, 8514, or 8515	43-row screen	12	17
PCLK adapter with XGA card + 8507, 8514, or 8515	27-row x 132-col screen	7	24
PCLK with Image Adapter/A + 8503, 8504, 8512, 8513, 8516, or 8518	24-row screen	8	19
PCLK with Image Adapter/A + 8503, 8504, 8512, 8513, 8516, or 8518	32-row screen	8	14
PCLK with Image Adapter/A + 8503, 8504, 8512, 8513, 8516, or 8518	43-row screen	8	10
PCLK with Image Adapter/A + 8506, 8507, 8508, 8514, 8515, or 6091	24-row screen	12	30
PCLK with Image Adapter/A + 8506, 8507, 8508, 8514, 8515, or 6091	32-row screen	12	23
PCLK with Image Adapter/A + 8506, 8507, 8508, 8514, 8515, or 6091	43-row screen	12	17
PCLK with Image Adapter/A + 8506, 8507, 8508, 8514, 8515, or 6091	27-row x 132-col screen	7	24
8775 display	1, 11	9	16
8775 display	1, 12	9	12 or 16

Notes:

1. The alphanumeric cell-size on a 3179-G or 3192-G can be either 9 by 12, or 9 by 16. The actual cell-size is governed by the depth of the GDDM page and subsystem-related factors. Usually, any page with 24 rows or less causes a cell-size of 9 by 16, with other page sizes receiving a cell-size of 9 by 12. The substitution character for a 3179-G or 3192-G is independent of the page size, and always corresponds to the 9 by 12 cell.
2. This is an approximate figure. The actual height is calculated internally by GDDM.

Alphanumerics

Alphanumerics cannot be plotted, nor can they be defined on non-cell-based family-4 devices.

Alphanumeric field attributes

All alphanumeric field attributes may be specified on all devices that support alphanumerics. Attributes not supported on a particular device are ignored. The information returned for “Code=0” in the FSQURY call indicates which attributes are supported by the primary device.

Double-byte character sets (DBCS)

Alphanumeric DBCS characters are supported on the ***3278-52, 3283-52, 5550 family multistations***, and ***PS/55***.

Cursor position within mixed fields For a cursor position within a mixed field, the column indicates the byte position. If this is at a DBCS character position and the device is a DBCS device, the cursor position coincides with the start of the DBCS character.

3278–52

If a field containing DBCS characters is positioned at an even column on the screen, and the device is a ***3278-52***, the DBCS field is protected and filled with the DUP character.

Alphanumeric colors

The standard default color varies according to the device being used:

- For ***monochrome displays***, the default is green (orange on a ***3290***, white on a ***3193***).
- For ***monochrome and most color printers***, the default is black.
- For ***3268 or 3287 printers***, the default is black or green, depending on the Base Color Specify feature.
- For ***3179-G, 3192-G, 3472-G***, and ***3279 color display stations***, and ***3270-PC, 3270-PC/G, and 3270-PC/GX workstations***, ***5550-family displays***, and ***devices using the GDDM-PCLK or GDDM-OS/2 Link program***, the default color varies according to the application:
  - In Extended Color mode, where the screen has defined nondefault colors, or Character Reply Mode has been requested, the default color is green, except for a high-intensity field, which is white. (High intensity has no effect on nondefault color fields.)
  - In Base Color mode, where the screen has not requested any nondefault colors and Field Reply Mode is in operation, the default color depends on the field type, field intensity, and the setting of the operator control (if any). This table summarizes the variations:

Operator control setting: Base Color

Field type	Intensity	Default color
Protected	High	White
Protected	Normal	Blue
Unprotected	High	Red
Unprotected	Normal	Green

Operator control setting: Monochrome

The default color is green for normal-intensity fields, and white for high-intensity fields.

Graphics colors

The following device-dependent variations on the colors displayed by the GSCOL call are observed. The result depends on the color capability of the device.

- On ***eight-color displays*** (for example, ***3279, 3270-PC/G workstations***) and ***5550-family Multistation***, colors 9 through 15 map to colors 1 through 7. Color 16 maps to the default color for the device.
- On ***four-color printers***, the primary colors (red, green, and blue) are interpreted correctly. Other colors in the range 1 through 7 print as black. Colors in the range 9 through 15 are first mapped to colors 1 through 7, color 16 being mapped to default (as for an 8-color display), and are then interpreted as above.
- On ***monochrome 3270-PC/GX workstations***, colors 1 through 7 provide different intensities, as follows:
  - Color 1 displays at one-third intensity
  - Color 7 displays at full intensity
  - All other colors display at two-thirds intensity.
- On ***monochrome displays***, all colors except colors 8 and –1 display identically. Colors 8 and –1 both appear identically as background.
- For a ***monochrome printer***, all colors except colors 8 and –2 display identically. Colors 8 and –2 both appear identically as background.
- On ***family-4 devices***, if color separation is required, colors in excess of the highest color in the selected color table, map to entries in the table. Color –2 (explicit white) maps to the seventh entry in the selected table, and color –1 (explicit black) maps to the eighth entry. Color 8 is always treated as background, even if the eighth entry in the selected table is not defined as background. Color –1 maps to the actual eighth entry, which is not necessarily background. If color separation is not required, all colors except colors 8 and –2 (background and explicit white) appear identically.

  If procopt OFFORMAT is set to GRIMAGE or GRCIMAGE, color wrapping occurs above color 16.

- On ***plotters***, the color numbers map to a pen stall, as follows:

GSCOL color number	2-pen plotter pen number	6-pen plotter pen number	8-pen plotter pen number
-2	no pen used	no pen used	no pen used
-1	1	6	7
0	2	6	8
1	1	1	1
2	2	2	2
3	1	3	3
4	2	4	4
5	1	5	5
6	2	6	6
7	1	6	7
8	no pen used	no pen used	no pen used
9	1	1	1
10	2	2	2
12	2	4	4
13	1	5	5
14	2	6	6
15	1	6	7
16	2	6	8

  If the color number is greater than the highest pen number, GDDM wraps around the set of numbers after the lowest power of 2 that is greater than or equal to the highest pen number (after 8 on a six-pen plotter or after 2 on a two-pen plotter). Numbers between the highest pen number and the next power of two use the highest pen number, so on a six-pen plotter both color 7, and color 6, use pen 6. Color 8 is an exception as it always means “background.”

Note.This color mapping does not apply to plotters connected through GDDM-OS/2 Link In this case, the color mapping depends on how the plotter is configured to OS/2.

Color mixing

The following device-dependent restrictions on foreground (GSMIX) and background (GSBMIX) color mix modes should be noted.

Foreground color mix mode

“Mix” mode: The ***3270-PC/GX workstations*** support 16 colors. The results of mixing these colors are device-dependent, and therefore cannot be shown here.

“Underpaint” mode: On ***3270-PC/G and 3270-PC/GX workstations***, ***5550-family Multistations***, and ***5080 or 6090 Graphics Systems***, “underpaint” mode is not supported. The device itself causes the results to appear in “overpaint” mode.

“Exclusive-OR” mode: On the ***3279***, arcs drawn with line width greater than 1 will be drawn as 2 lines.

On the ***5080 Graphics System***, “exclusive-OR” mode is treated as “overpaint” mode.

On ***IPDS printers***, “exclusive-OR” mode is not supported.

“Transparent” mode: On ***5080 or 6090 Graphics Systems***, “transparent” mode is not supported.

“Mix” mode: On GDDM-PCLK printers, the color white produces no output.

Combinations of foreground and background mix modes

On many devices, only certain combinations of foreground and background mix modes are supported.

If a combination of foreground and background mix modes is requested that is not allowed, the call specifying this combination (GSMIX or GSBMIX) issues a warning message. In this case the requested mode is recorded, but while this combination exists results are device dependent; normally all primitives are drawn with a background mix mode of transparent and with the requested foreground mix mode. If a subsequent change to the foreground mix mode results in the requested combination of modes becoming valid, the requested background mix mode is used.

- On ***IBM devices that do not support background mix orders***, the following combinations of foreground and background color-mix modes are supported:

Foreground	Background
Any	Transparent
Overpaint	Opaque and transparent

- When displaying images on ***Tektronix displays***, the following combinations of foreground and background color-mix modes are supported:

Foreground	Background
Overpaint	Opaque
Mix	Transparent
XOR	Transparent
Transparent	Transparent

- When displaying images on ***DEC displays***, the following combinations of foreground and background color-mix modes are supported:

Foreground	Background
Overpaint	Transparent
Transparent	Transparent

- When displaying lines, shaded areas, and characters on ***Tektronix and DEC displays***, the following combinations of foreground and background color-mix modes are supported:

Foreground	Background
Overpaint	Opaque
Overpaint	Transparent
Transparent	Transparent

Note: When displaying shaded areas on ***Tektronix 42xx series displays with microcode level 11***, *opaque* is the only background mix mode available. When displaying shaded areas on ***Tektronix 4105 and 42xx series displays with microcode earlier than level 11***, *transparent* is the only background mix mode available.

Graphics line types and widths

Line types (GSLT)

On ***family-4 printers***, the same value of the line-type parameter may produce different results on different types of printer. The results on family-4 printers may also differ from the results on ***displays*** and ***family-2 printers***.

Line widths (GSFLW and GSLW)

The standard width (to which the line-width multiplier is applied), together with the minimum and maximum line widths on various devices, is listed below:

	Min.	Std.	Max.
IBM 3270 displays, printers, plotters	1	1	2
IBM 3800-3 Printer	1	3	300
IBM 4250 Printer	1	6	600
IBM 5080 and 6090 Graphics System	1	1	1
ASCII graphics devices	1	1	1

Graphics area shading

The following device-dependent limitations in the implementation of graphics area shading should be noted:

- On ***3270-PC/GX workstations***, when a large area is drawn and there is not enough segment storage configured, the device may not perform the shading. When this happens, the 3270-PC/GX issues an unformatted message to this effect.
- On ***3270-PC/G and 3270-PC/GX workstations***, the shading pattern must be less than or equal to 8 pixels wide and less than or equal to 15 pixels deep to ensure the correct shading. An area is shaded by repeating the pattern every 8 pixels horizontally and is repeated vertically so that if the pattern is less than or equal to 15 rows deep, successive cells are adjacent without blank lines. If the depth is greater than 15 pixels, shading patterns may be truncated to 15.
- On ***devices running under GDDM-PCLK or GDDM-OS/2 Link***, shading patterns are not loadable.
- On the ***IBM 5080 Graphics System***, only the GDDM-defined patterns (patterns 0 through 16) are supported. All shading patterns except 15 (hollow) are opaque; that is the spaces between the shading marks are not transparent.
- Some devices such as:
  - ***IPDS printers***
  - ***ASCII graphics displays***
  - ***Personal computer systems running GDDM-PCLK and GDDM-OS/2 Link***

  may not support the shading patterns you have defined.

On some ASCII graphics displays, complex areas – or areas with more than 255 vertices may be shaded incorrectly. Some of the characters in the high quality DBCS symbol sets may be affected by this device limitation.

Graphics image

When processing the GSIMG call, on the ***5080 Graphics System***, the bits set to zero are black; that is, they are not transparent. Therefore, any image completely replaces the existing data. This means that multicolored images cannot be supported as only the last color is visible.

Graphics logical input devices

String and stroke devices are only supported on the ***IBM 3270-PC/G and 3270-PC/GX workstations***, and ***IBM 5080 Graphics System***.

Choice devices

The following limitations on choice devices should be noted:

- Mice are not supported on all devices.
- Tablets (and therefore puck buttons) and data keys are only supported on the ***IBM 3270-PC/G and 3270-PC/GX workstations***, and ***IBM 5080 Graphics Systems***.
- On any one device, either a mouse or a tablet may be supported, but not both.
- Mice or tablets are only supported when configured on the particular device in use.
- Light pens are only supported on a limited range of devices.

Locator devices (GSILOC)

- For ***3179-G***, ***3192-G***, and ***3472-G color graphics displays***, ***5550-family Multistations***, and ***workstations running under GDDM-PCLK or GDDM-OS/2 Link***, the locator device is a mouse if one is configured.
- For ***3270-PC/G and 3270-PC/GX workstations***, the locator device is a mouse or tablet if one is configured. Tablet and mouse are mutually exclusive.

Echo types: The echo is what the operator sees on the screen when using the locator. Six different echo types may be specified.

Echo type 0 (default echo)

- For ***3179-G***, ***3192-G***, and ***3472-G color graphics displays***, ***3270-PC family workstations***, and ***workstations using the GDDM-PCLK or GDDM-OS/2 Link program***, the locator can be changed by the device; for more information on how this is done, refer to the appropriate user guide.
- For ***other 3270 family displays***, the locator echo is the alphanumeric cursor.
- For the ***5080 Graphics System***, the locator echo is a small cross.
- For the ***5550-family Multistations***, the locator echo is a cross-hair cursor.

Echo type 1 (null echo)

- For ***3179-G, 3192-G***, and ***3472-G color graphics displays***, ***3270-PC/G and 3270-PC/GX workstations***, ***5550-family workstations***, and ***workstations using the GDDM-PCLK or GDDM-OS/2 Link program***, the echo is invisible and the locator position is not shown.
- For ***other 3270-family displays***, the alphanumeric cursor is used. The initial position is described by the **x-coord** and **y-coord** parameters.
- For the ***5080 Graphics System***, the locator echo is a small cross.

Echo type 2

- For the ***3179-G, 3192-G***, and ***3472-G color graphics displays***, ***3270-PC family workstations***, and ***workstations running under GDDM-PCLK or GDDM-OS/2 Link***, the locator can be changed by the device; for more information on how this is done, refer to the appropriate user guide.
- For the ***5080 Graphics System***, the locator is a cross-hair.
- For the ***5550-family Multistation***, the locator is a cross-hair cursor.

Echo types 4 (rubber band) and 5 (rubber box) For the ***5080 Graphics System***, when using rubber-band and rubber-box echo types, if the position of the fixed end or corner is not visible at the time of a GSREAD call, GDDM does not ensure that the initial position and type of the locator echo are correct.

Echo type 6 (transformable graphics segment) For the ***3270-PC/G and 3270-PC/GX workstations***, the copy of the segment is displayed in a form that facilitates drawing by the devices (exclusive-OR mode), and the mixing of the colors of the segment can be different from that for the original segment. This color mixing applies to the mixing with other segments on the screen as well as the primitives of the echoed segment itself.

Trigger keys: Examples of trigger keys are:

Table 5. Triggering keys for locator and pick devices

Display	Triggering keys
3179-G, 3192-G, 3472-G, and 5550-family	ENTER key PF keys Mouse buttons
3270-PC/G and 3270-PC/GX	ENTER key PF keys Mouse or puck buttons
3279	ENTER key PF keys Alphanumeric light pen detect
5080	Data keys (if they are enabled and a string device is not enabled) The mouse or puck buttons (if used for the locator).

Pick devices (GSIPIK)

- For ***3179-G, 3192-G***, and ***3472-G color graphics displays***, ***5550-family Multistations***, and ***workstations running under GDDM-PCLK or GDDM-OS/2 Link***, the locator device is a mouse if one is configured.
- For ***3270-PC/G and 3270-PC/GX workstations***, the locator device is a mouse or tablet if one is configured. Tablet and mouse are mutually exclusive.

Echo

- For ***3270-PC/G or 3270-PC/GX workstations***, the echo is a square that shows the pick aperture. This square is centered on the pick position and superimposed over the echo selected for an enabled locator device. If no locator is enabled, the square is superimposed over a cross-hair cursor. The size of the aperture may be set using the GSIDVF call before enabling the pick device.
- For a ***3279***, the feedback is the position of the alphanumeric cursor.
- For ***3179-G, 3192-G***, and ***3472-G color graphics displays***, ***5550-family Multistations***, and ***workstations using the GDDM-PCLK or GDDM-OS/2 Link program***, the feedback is the position of the graphics cursor, unless a null locator echo has been selected.

Trigger keys: For examples of trigger keys on different devices, see Table 5.

Stroke devices (GSISTK)

The following device-dependent variations should be noted.

- On ***3270-PC/G and PC/GX workstations***, echoing, the mouse or tablet buttons can be used to suspend the stream, move the locator to a new position, and restart the stream if this is desired.
- On the ***5080 Graphics System***, the input is completed the first time the button is released. The value of the **count** parameter may be reduced when the GSREAD

call is made, according to the processing conditions applying at that time.

String devices (GSISTR)

- On the ***5080 or 6090 Graphics System***, only the ENTER key is effective for putting the string data into the input queue. The maximum length of a string on these devices is 80.

  The echo is clipped to the right edge of the screen. If the start of the string is not visible, the string appears with echo-type 2.

  The string device is not triggered by PF keys or data keys.
- On the ***3270-PC/G and PC/GX***, the echo is clipped at the graphics field boundary.
- On ***3270-PC/G and 3270-PC/GX workstations, and 5550-family Multistations***, PA3 is reserved by GDDM (to perform local-mode processing or to redraw the screen).

Image

The following device-dependent variations should be noted.

- Echoing is done by the ***3193 display*** unless the projection contains transformations that the 3193 cannot process, within the quality requirements specified in the ISCTL or ISXCTL calls.
- On ***3270-family displays other than the 3193***, the image box cursor is not supported.
- When using the GDDM image interface on the ***5080 Graphics System***, there are no restrictions other than that imposed by the 5080 buffer size, which may limit the size of the image data.

Chapter 5. APL request codes module

An APL request codes module, which is independent of the subsystem under which GDDM is running, is provided with GDDM. The module defines for each GDDM call statement, the associated APL request code to be used by an APL function when requesting services of GDDM through the APL Auxiliary Processor AP 126.

Although all GDDM call statements have an equivalent APL code assigned, not all codes are supported through AP 126. The APL manuals listed below identify the supported codes for each of the subsystems for which APL is available.

- *VS APL for CICS: Terminal User's Guide*,
- *VS APL for CMS: Terminal User's Guide*,
- *VS APL for TSO: Terminal User's Guide*,
- *APL2 Programming: System Services Reference* manual.

The address of the APL Request Codes Module can be acquired by an application program by using the CALLINF external default option in the SPINIT call; see "Initialization" on page 431.

The APL Request Codes Module is in two sections. The first provides an address table locating the descriptors for a specific range of APL codes. The second section defines the equivalence between APL request codes and GDDM calls for all codes within a specific range.

Note: The APL request codes module may be subject to change between different releases of GDDM, or, as a result of maintenance.

The address table

The address table is located at offset 0 from the entry point of the module. The format of the address table is as follows:

Table 6. APL request codes module – address table

Field name	Field offset	Field length
RCPAIDEN	0	8
RCPAVERS	8	4
RCPATNUM	C	4
RCPAENUM	10	4
RCPATABP(1)	14	8
RCPALOW(1)	0	2
RCPAHIGH(1)	2	2
RCPAPTR(1)	4	4
RCPATABP(2)	1C	8
⋮	⋮	⋮
RCPATABP(n)	14+(n-1)×8	8

RCPAIDEN
Module identifier containing the character string "ADMADAP."

RCPAVERS
A fullword integer identifying the version number of the APL Request Codes Module. The field is currently set to zero.

RCPATNUM
A fullword integer containing the number of assigned APL codes defined in the following tables.

RCPAENUM
A fullword integer containing the number of table indexes to follow. One table index exists for each block of APL codes in the range:

```
100*n : 100*(n+1)-1
```

where `n` is greater than or equal to 0. Thus, the maximum APL code is:

```
RCPAENUM*100-1
```

RCPALOW(n)
A two-byte integer identifying the lowest value in the APL index table pointed to by RCPAPTR(n). The value is currently always set to 100⋆(n-1).

RCPAHIGH(n)
A two-byte integer identifying the highest value in the APL index table pointed to by RCPAPTR(n). The value is always less than 100⋆n.

RCPAPTR(n)
The address of the request code table for those APL codes in the range RCPALOW(n) through RCPAHIGH(n). If the value of the pointer is zero, there are no codes assigned within the range.

The request code table

The Request Code Table is addressed from the address table described in "The address table." There is one entry in the table for each potential APL code in the range RCPALOW(n) through RCPAHIGH(n). The format of the request code table is:

Table 7. APL request codes module – request code table

Field name	Field offset	Field length
RCPAAPLC(1)	0	8
RCPAAPLN(1)	0	2
RCPAAPLG(1)	2	6
RCPAAPLC(2)	8	8
⋮	⋮	⋮
RCPAAPLC(m)	8×(m-1)	8

RCPAPLN(i)
A two-byte integer containing the APL request code. A code of zero indicates that there is no GDDM function assigned to that code.

RCPAAPLG(i)

A six-byte character string containing the name of the GDDM call (for example “ASREAD”) corresponding to the APL function code in RCPAPLN(i).

GDDM Base APL codes, in numeric order

The remainder of this chapter lists APL codes for the GDDM Base calls in numeric order.

Table 8 (Page 1 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
101	ASREAD	Device output/input
102	FSFRCE	Update the display
103	FSREST	Retransmit data
104	FSSAVE	Save current page contents
105	FSSHOW	Display a saved picture
106	FSCHEK	Check picture complexity before output
107	FSQERR	Query last error
108	FSTRCE	Control internal trace
109	FSALRM	Sound the terminal alarm
110	FSQDEV	Query device characteristics
111	ASTYPE	Override alphanumeric character-code assignments
112	ESLIB	Library management
113	ESPCB	Identify program communication block
114	FSEXIT	Specify an error exit, or error threshold, or both
115	SPINIT	Initialize GDDM with SPIB
116	FSTERM	Terminate GDDM processing
117	FSINIT	Initialize GDDM processing
118	FSRNIT	Reinitialize GDDM
119	FSSHOR	Extended FSSHOW
120	GSREAD	Await graphics input
121	FSQURY	Query device characteristics
122	FSQSYS	Query systems environment
123	ESSUDS	Specify source-format user default specification
124	ESEUDS	Specify encoded user default specification
127	ESACRT	Create application group
128	ESADEL	Delete application group
129	ESAQRY	Query the current application group
130	ESASEL	Select an application group
132	FSTRAN	Translate character string
133	ESQCPG	Query code page of a GDDM object
134	ESSCPG	Set code page of a GDDM object
135	ESQEUD	Query encoded user default specification
136	ESQUNL	Query length of user-defined nickname information
137	ESQUNS	Query user-defined nickname information
140	ESQOBJ	Query existence of GDDM object on auxiliary storage
151	WSIO	Windowed device input/output
180	ISFLD	Define image field
181	ISQFLD	Query image field
182	ISCTL	Set image quality-control parameters
183	ISXCTL	Extended set image quality control parameters
184	ISQFOR	Query image formats supported by the device
185	ISESCA	Control echoing of scanner image
186	ISLDE	Load external read-only image
187	ISQSCA	Query image scanner device
188	ISQRES	Query supported image resolutions
189	ISENAB	Enable or disable image cursor
190	ISQLOC	Query image locator cursor position
191	ISILOC	Initialize image locator cursor
192	ISQBOX	Query image box cursor
193	ISIBOX	Initialize image box cursor
194	ISQCOM	Query image compressions supported by the device

Table 8 (Page 1 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
201	GSDSS	Load a graphics symbol set from the application program
202	GSLSS	Load a graphics symbol set from auxiliary storage
203	PSDSS	Load a symbol set into a PS store from the application program
204	PSLSS	Load a symbol set into a PS store from auxiliary storage
205	PSLSSC	Conditionally load a symbol set into a PS store from auxiliary storage
206	PSRSV	Reserving or releasing a PS store
207	GSRSS	Release a graphics symbol set
208	PSRSS	Release a symbol set from a PS store
209	GSQNSS	Query the number of loaded symbol sets
210	GSQSS	Query loaded symbol sets
211	PSQSS	Query status of device stores
212	SSQF	Query a symbol set on auxiliary storage
213	SSREAD	Read a symbol set from auxiliary storage
214	SSWRT	Write a symbol set to auxiliary storage
215	GSCPG	Set current code page
216	GSQCPG	Query code page
280	APDEF	Define a field list
281	APDEL	Delete a field list
282	APMOD	Modify a field list
283	APQIDS	Query field list identifiers
284	APQNUM	Query field list numbers
285	APQRY	Query a field list
286	APQSIZ	Query a field list size
287	APQUID	Query unique field list identifier
301	FSPCLR	Clear the current page
302	FSPCRT	Create a page
303	FSPDEL	Delete a page
304	FSPQRY	Query specified page
305	FSPSEL	Select a page
306	FSQCPG	Query current page identifier
307	FSQUPG	Query unique page identifier
308	MSPQRY	Query current page
309	FSPWIN	Set page window
310	FSQWIN	Query page window
313	FSENAB	Enable/disable device input
401	ASDFLD	Define or delete a single field
402	ASDFMT	Define alphanumeric fields, deleting all existing fields
403	ASDTRN	Define I/O translation tables
404	ASFCLR	Clear fields
405	ASRFMT	Define multiple fields without deleting existing fields
406	ASDFLT	Set default field attributes
407	ASFCOL	Define field color
408	ASFEND	Define field end attribute
409	ASFHLT	Define field highlighting
410	ASFIN	Define input null-to-blank conversion
411	ASFINT	Define field intensity
412	ASFMOD	Change field status
413	ASFOUT	Define output blank-to-null conversion
414	ASFPSS	Define primary symbol set for a field
415	ASFTRN	Assign translation table set to a field
416	ASFTYP	Define field type
417	ASRATT	Define field attributes
418	ASQFLD	Query field attributes
419	ASQMAX	Query the number of fields
420	ASQMOD	Query modified fields
421	ASCCOL	Specify character colors within a field
422	ASCGET	Get field contents
423	ASCHLT	Specify character highlights within a field
424	ASCPUT	Specify field contents
425	ASCSS	Specify character symbol sets within a field
426	ASMODE	Define the operator reply mode
427	ASQCOL	Query character colors for a field
428	ASQHLT	Query character highlights for a field

Table 8 (Page 2 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
429	ASQSS	Query character symbol sets for a field
430	ASFCUR	Position the cursor
431	ASQCUR	Query cursor position
432	ASGPUT	Specify double-character field contents
433	ASGGET	Get double-character field contents
434	ASFTRA	Define field transparency attribute
435	ASQNMF	Query the number of modified fields
436	ASFBDY	Define field outline
437	ASFSEN	Define field mixed-string attribute
438	SPMXMP	Control the use of mixed fields by mapping
439	DSCMF	User Control function
440	DSQCMF	Query user control function
443	ASQLEN	Query length of field contents
501	GSCLP	Enable and disable clipping
502	GSFLD	Define the graphics field
503	GSPS	Define the picture space
504	GSVIEW	Define a viewport
505	GSWIN	Define a graphics window
506	GSCLR	Clear the graphics field
507	GSSCLS	Close the current segment
508	GSSDEL	Delete a segment
509	GSSEG	Create a segment
510	GSCA	Set current character angle
511	GSCB	Set character-box size
512	GSCD	Set current character direction
513	GSCM	Set current character mode
514	GSCOL	Set current color
515	GSCS	Set current symbol set
516	GSLT	Set current line type
517	GSLW	Set current line width
518	GSMIX	Set current foreground color-mixing mode
519	GSMS	Set the current type of marker symbol
520	GSPAT	Set current shading pattern
521	GSARC	Draw a circular arc
522	GSAREA	Start a shaded area
523	GSCHAP	Draw a character string at current position
524	GSCHAR	Draw a character string at a specified point
525	GSENDA	End a shaded area
526	GSLINE	Draw a straight line
527	GSMARK	Draw a marker symbol
528	GSMOVE	Move without drawing
529	GSMRKS	Draw a series of marker symbols
530	GSPLNE	Draw a series of lines
531	GSVECM	Vectors
532	GSQCA	Query character angle
533	GSQCB	Query character-box size
534	GSQCD	Query character direction
535	GSQCEL	Query default graphics cell size
536	GSQCLP	Query the clipping state
537	GSQCM	Query the current character mode
538	GSQCOL	Query the current color
539	GSQCP	Query the current position
540	GSQCS	Query the current symbol-set identifier
541	GSQCUR	Query the cursor position
542	GSQLT	Query the current line type
543	GSQLW	Query the current line width
544	GSQMAX	Query the number of segments
545	GSQMIX	Query the current color mixing mode
546	GSQMS	Query the current marker symbol
547	GSQPAT	Query the current shading pattern
548	GSQPS	Query the picture-space definition
549	GSQVIE	Query the current viewport definition
550	GSQWIN	Query the current window definition
551	GSELPS	Draw an elliptic arc
552	GSIMG	Draw a graphics image
553	GSPUT	Restore graphics data
554	GSGETS	Start retrieval of graphics data
555	GSGET	Retrieve graphics data
556	GSGETE	End retrieval of graphics data
557	GSPFLT	Draw a curved fillet

Table 8 (Page 2 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
558	GSCH	Set current character shear
559	GSQCH	Query character shear
560	GSQTB	Query the text box
561	GSFLW	Set current fractional line width
562	GSQFLW	Query the current fractional line width
563	GSMSC	Set marker scale
564	GSQMSC	Query marker scale
565	GSIMGS	Draw a scaled graphics image
566	GSTAG	Set current primitive tag
567	GSQTAG	Query current tag
568	GSILOC	Initialize locator
569	GSIPIK	Initialize pick device
570	GSIDVI	Initial data value, integer
571	GSIDVF	Initial data value, float
572	GSENAB	Enable or disable a logical input device
573	GSFLSH	Clear the graphics input queue
574	GSQSIM	Query existence of simultaneous queue entry
575	GSQCHO	Query choice device data
576	GSQLOC	Query graphics locator data
577	GSQPIK	Query pick data
578	GSSATI	Set initial segment attributes
579	GSQATI	Query initial segment attributes
580	GSSATS	Modify segment attributes
581	GSQATS	Query segment attributes
582	GSSPOS	Set segment position
583	GSQPOS	Query segment position
584	GSUWIN	Define a uniform graphics window
585	GSQFLD	Query the graphics field
586	GSQSSD	Query symbol set data
587	GSSORG	Set segment origin
588	GSSAGA	Set all geometric attributes
589	GSQAGA	Query all geometric attributes
590	GSSTFM	Set segment transform
591	GSQTFM	Query segment transform
592	GSSAVE	Save a segment
593	GSLOAD	Load segments
594	GSISTR	Initialize string device
595	GSISTK	Initialize stroke device
596	GSQSTR	Query string data
597	GSQSTK	Query stroke data
598	GSARCC	Specify aspect-ratio control (for copy)
601	FSCLS	Close alternate device
602	FSCOPY	Send page to alternate device
603	FSLOG	Send character string to alternate device
604	FSOPEN	Open alternate device
605	GSCOPY	Send graphics to alternate device
606	FSLOGC	Send character string with carriage-control character to alternate device
620	FSGETS	Begin retrieval of family-4 AFPDS datastream
621	FSGET	Retrieve a family-4 AFPDS print-file record
622	FSGETE	End retrieval of family-4 AFPDS datastream
632	GSSINC	Include a segment
633	GSSCPY	Copy a segment
634	GSSPRI	Set segment priority
635	GSQPRI	Query segment priority
636	GSMB	Set marker-box size
637	GSQMB	Query marker box
638	GSCORR	Explicit correlation of tag to primitive
639	GSQORG	Query segment origin
643	GSQLID	Query logical input device
644	GSTA	Set text alignment
645	GSQTA	Query the current text alignment
646	GSCBS	Set character-box spacing
647	GSAM	Set attribute mode
648	GSQAM	Query the current attribute mode
649	GSPOP	Restore attributes
650	GSQCBS	Query character-box spacing
651	GSSCT	Set current transform
653	GSCALL	Call a segment
654	GSQPKS	Query pick structure

Table 8 (Page 3 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
655	GSCORS	Explicit correlation of structure
656	GSQBND	Query the current data boundary definition
657	GSBND	Define a data boundary
658	GSSVL	Define segment viewing limits
659	GSQSVL	Query the current segment viewing limits
660	GSDEFS	Start the drawing defaults definition
661	GSDEFE	End drawing defaults definition
662	FSUPDM	Set update mode
663	FSQUPD	Query update mode
664	GSBMIX	Set current background color-mixing mode
665	GSQBMX	Query the current background color-mixing mode
666	GSSEN	Set mixed string attribute of graphics text
667	GSQSEN	Query mixed string attribute of graphics text
668	GSCP	Set current position
669	CGLOAD	Load a picture from a Computer Graphics Metafile
670	CGSAVE	Save segments in a Computer Graphics Metafile
901	DSOPEN	Open a device
902	DSCLS	Close a device
903	DSUSE	Specify device usage
904	DSDROP	Discontinue device usage
905	DSQUID	Query unique device identifier
906	DSQUSE	Query device usage
907	DSQDEV	Query device characteristics
908	DSRNIT	Reinitialize a device
909	DSCOPY	Send transformed picture to alternate device
910	DSFRCE	Output member to a PDS
1001	PTSCRT	Create a partition set
1002	PTSQRY	Query partition set attributes
1003	PTSSEL	Select a partition set
1004	PTSDEL	Delete a partition set
1005	PTSQUN	Query unique partition set identifier
1006	PTSSPP	Set partition viewing priorities
1007	PTSQPP	Query partition viewing priorities
1008	PTSQPI	Query partition identifiers
1009	PTSQPN	Query partition numbers
1021	PTNCRT	Create a partition
1022	PTNQRY	Query the current partition
1023	PTNMOD	Modify the current partition
1024	PTNSEL	Select a partition
1025	PTNDEL	Delete a partition
1026	PTNQUN	Query unique partition identifier
1040	WSCRT	Create an operator window
1041	WSDEL	Delete operator window
1043	WSMOD	Modify the current operator window
1044	WSQRY	Query the current operator window
1045	WSQUN	Query unique operator window identifier
1046	WSQWI	Query operator window identifiers
1049	WSQWP	Query operator window viewing priorities
1050	WSQWN	Query operator window numbers
1051	WSSEL	Select an operator window
1052	WSSWP	Set operator window viewing priorities
1101	MSREAD	Present mapped data
1102	MSPCRT	Create a page for mapping
1103	MSQGRP	Query mapgroup characteristics
1104	MSQMAP	Query map characteristics
1105	MSQADS	Query application data structure definition
1106	MSQFIT	Query map fit
1107	MSQMOD	Query modified fields
1108	MSDFLD	Create or delete a mapped field
1109	MSPUT	Place data into a mapped field
1110	MSGET	Retrieve data from a map
1111	MSQFLD	Query mapped field characteristics
1112	MSCPOS	Set cursor position
1113	MSQPOS	Query cursor position
1196	CDPU	Control the printing of Composite Documents
1201	ISSE	Run the Image Symbol Editor
1600	IMAGID	Get and reserve a unique image identifier

Table 8 (Page 3 of 3). GDDM Base APL codes, in numeric order

APL code	Call name	Description
1601	IMACRT	Create an image
1602	IMARES	Convert the resolution attributes of an image
1603	IMADEL	Delete the image associated with the identifier
1604	IMACLR	Clear a rectangle in an image
1605	IMATRM	Trim an image down to the specified rectangle
1607	IMASAV	Save image on auxiliary storage
1608	IMARST	Restore image from auxiliary storage
1609	IMAPTS	Start data entry into an image
1610	IMAPT	Enter data into an image
1611	IMAPTE	End data entry into an image
1612	IMAGTS	Start retrieval of data from an image
1613	IMAGT	Retrieve image data from an image
1614	IMAGTE	End retrieval of data from an image
1615	IMXFER	Transfer data between two images, applying a projection
1619	IMAQRY	Query attributes of an image
1620	IMARF	Change resolution flag of an image
1650	IMPCRT	Create an empty projection
1651	IMPGID	Get and reserve a unique projection identifier
1652	IMPDEL	Delete projection
1653	IMPSAV	Save projection on auxiliary storage
1654	IMPRST	Restore projection from auxiliary storage
1655	IMREX	Define rectangular sub-image in pixel coordinates
1656	IMREXR	Define rectangular sub-image in real coordinates
1657	IMRPL	Define place position in pixel coordinates
1658	IMRPLR	Define place position in real coordinates
1659	IMRSCL	Scale extracted image
1660	IMRRAL	Set current resolution/scaling algorithm
1661	IMRORN	Turn an extracted image clockwise through a number of right angles
1662	IMRREF	Reflect extracted image
1663	IMRNEG	Negate the pixels of an extracted image
1664	IMRCVB	Define bi-level conversion algorithm
1665	IMRBRI	Define brightness conversion algorithm
1666	IMRCON	Define contrast conversion algorithm

Chapter 6. GDDM-REXX programming interface

GDDM-REXX enables you to run GDDM in EXECs written for the VM/System Product Interpreter, or the TSO/E Language Processor, using the Restructured Extended Executor Language – REXX.

A GDDM-REXX program may consists of four kinds of components:

- REXX statements
- CMS or TSO commands
- GDDM-REXX commands, subcommands, and utilities
- GDDM calls.

GDDM-REXX commands, subcommands, and utilities

These form the interface between REXX and GDDM.

Summary

The following lists summarize the commands, subcommands, and utility EXEC available in GDDM-REXX.

Commands

GDDMREXX INIT Initializes GDDM-REXX.
GDDMREXX TERM Terminates GDDM-REXX.
GDDMREXX VERSION Displays version number of GDDM-REXX in use.

Subcommands

GXGET AAB Obtains the currently used Application Anchor Block (AAB). (Used in conjunction with `FSINIT` and `GXSET AAB`.)
GXGET CDT Extracts a string of bytes that contains the GDDM call descriptor table (CDT) entry for a given GDDM call. (This is in an encoded form.)
GXGET LASTMSG Extracts the text of the last error message.
GXGET MSG Extracts the current state and level of message display.
GXGET NAMES Extracts a string containing all the GDDM call names, of which, there are several hundred.
GXGET TRACE Extracts the current state of trace control.
GXSET AAB Establish the given AAB as current.
GXSET MSADS In mapping, moves data to the GDDM application data structure (ADS) from the variables that make up the REXX application data structure produced by the ERXMSVAR EXEC.
GXSET MSG Enables or disables display of messages at the specified severity level or higher.
GXSET MSVARS In mapping moves data to the REXX application data structure from the GDDM application data structure.
GXSET TRACE Enables or disables statement and variable fetch or set trace.

Utility EXEC

ERXMSVAR Creates a REXX application data structure for use with mapped alphanumerics.

Syntax conventions

The conventions used in presenting the syntax for commands, subcommands, and utilities are:

- The call name is shown in uppercase.
- The parameters are shown in lowercase.
- Uppercase words and parentheses must be coded exactly as shown.
- Lowercase words should be replaced by appropriate arguments.
- Elements of the call that are optional are shown within square brackets, thus: [...].

GDDMREXX command

GDDMREXX INIT

Function: Initializes GDDM-REXX.

GDDMREXX INIT	**[([NODCSS] [LANG X][)]]**

Parameters

NODCSS
Prevents the discontiguous saved segment (DCSS) being loaded, if there is one. Instead a copy of all of GDDM-REXX is loaded into user storage. If the DCSS is already in use, NODCSS is rejected with an error message. To unload the DCSS you must issue the `GDDMREXX TERM` command and re-initialize. A DCSS is an area of CMS storage available to many users.

This parameter is ignored under MVS.

LANG x
Loads GDDM-REXX message text in the appropriate language (module name ERXTMSGx on CMS or ERXTMSTx on MVS). It has no effect on messages from GDDM or elsewhere. The meaning of the letters is shown below. Some of them may not be available in your installation.

Where no translation language has been specified, maps will remain in U.S. English. `'x'` is one of the following letters:

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.

Notes:

1. If a double-byte character set language is used for GDDM or GDDM-REXX languages, we recommend that you operate with `GXSET MSG OFF`, and use GDDM to display the error messages that you extract with `GXGET LASTMSG`.
2. If this operand is omitted, the default is the same as the GDDM Base language.

GDDMREXX TERM

Function: Terminates GDDM-REXX and frees the storage used by it.

Termination of GDDM-REXX (and, hence, GDDM) also implicitly occurs at CMS command ready. On MVS, however GDDM-REXX is not automatically terminated when an EXEC ends. The programmer must ensure that a GDDMREXX TERM command is always executed before exiting from an EXEC. This includes exits cause by errors and by attention interrupts by the terminal user. Example code to trap such exits is given in "Termination on MVS" on page 11.

GDDMREXX TERM	**[(ALL[])]**

Parameters

ALL
Terminates all instances of GDDM-REXX.

GDDMREXX VERSION

Function: Returns the version and release level, product number, date, and copyright notice of the copy of GDDM-REXX being used. The options allow the use of this command within REXX EXEC files. If they are omitted, the response is returned to the terminal.

GDDMREXX VERSION	**[(action[])]**

Parameters

action
STACK The information is queued onto the stack behind any items that are already on the stack.
LIFO Last in/first out. The information is pushed onto the stack before any items that are already on the stack.
FIFO First in/first out. Same as STACK.

GXGET subcommand

The GXGET subcommand is used to extract information from GDDM-REXX.

GXGET AAB

Function: Extracts a token that relates to the current application anchor block (AAB). This can later be restored by a GXSET AAB subcommand, which will enable the correct instance of GDDM. (Used in conjunction with FSINIT and GXSET AAB.)

GXGET AAB	**.aabtoken**

Parameters

.aabtoken
Variable in which the token relating to the current AAB is returned. You should not tamper with this token in any way. Refer to the *GDDM Base Application Programming Guide* for more information.

GXGET CDT

Function: Gives the contents of the GDDM call descriptor table (CDT) in a byte string. The CDT is described in Chapter 22, "Special-purpose programming in GDDM" on page 431. The bytes need to be interpreted – a method of doing this is shown in the sample ERXPROTO EXEC.

GXGET CDT	**.name .entry**

Parameters

.name
Variable containing the name of the call for which the CDT entry is required. This may be coded as a literal, thus:
`'GXGET CDT GSLOAD .gslcdt'`
.entry
A variable in which the CDT byte string will be returned.

GXGET LASTMSG

Function: Gives text of the last error.

GXGET LASTMSG	**.msg**

Parameters

.msg
Variable in which the text of the last error message is put. If there have been no previous error messages, the string is empty.

GXGET MSG

Function: Gives current state and level of message handling.

GXGET MSG	.state .level

Parameters

.state
Variable in which the state of message handling is returned; its value is ON or OFF.
.level
Variable in which the level of messages shown will be returned. Values are:
0 (Informational) messages and above
4 (Warning) messages and above
8 (Error) messages and above
12 (Severe) messages and above.

GXGET NAMES

Function: Gives a string containing all the GDDM call names. Note that there are several hundred calls, all of which are included.

GXGET NAMES	.namelist

Parameters

.namelist
The variable into which the list of names is placed.

GXGET TRACE

Function: Gives the current state and level of trace.

GXGET TRACE	.state .time

Parameters

.state
Variable in which the state of tracing is returned. It may be ON or OFF.
.time
Variable in which the state of trace timing is returned. It may be TIME or NOTIME.

GXSET subcommand

The GXSET subcommand is used to pass information to GDDM-REXX. Although this syntax shows variable names, literals may be used instead (except where indicated), as in `'GXSET TRACE ON TIME'`.

GXSET AAB

Function: Establishes the given AAB as current. Used in conjunction with GXGET AAB.

GXSET AAB	.aabtoken

Parameters

.aabtoken
Variable containing the anchor block to be established as current, so that the associated instance of GDDM will be used. Refer to the *GDDM Base Application Programming Guide* for more information.

GXSET MSADS

Function: Moves data to the user's application data structure, which must have been created with the ERXMSVAR EXEC. Use before output when using mapped alphanumerics.

GXSET MSADS	.mapgrp .map .prefix .ads

Parameters

.mapgroup
Name of the mapgroup that contains the map
.map
Name of the map
.prefix
The same as the prefix specified in the ERXMSVAR EXEC.
.ads
The name of the application data structure variable. Refer to the *GDDM Base Application Programming Guide* for more information.

GXSET MSG

Function: Enables or disables display of messages at specified severity level or higher. The default is that the state is `ON` with a level of 4, meaning that warning messages and those of a greater severity are shown.

GXSET MSG	.state [.level]

Parameters

.state
Sets message handling state; it may be ON or OFF.

.level
Sets message handling level; it may be:
0 echo (display before processing) user statements and display all messages
4 (Warning) messages and above
8 (Error) messages and above
12 (Severe) messages and above.

GXSET MSVARS

Function: Moves data from the user's application data structure, which was created with the ERXMSVAR EXEC. Use after input when using mapped alphanumerics.

GXSET MSVARS	.mapgrp .map .prefix .ads

Parameters

.mapgrp
Name of the mapgroup that contains the map.
.map
Name of the map.
.prefix
Prefix specified in the ERXMSVAR EXEC. This is used as a stem to give variable names suitable for use with REXX. Note that the dot is required.
.ads
Name of the application data structure. Refer to the *GDDM Base Application Programming Guide* for more information.

GXSET TRACE

Function: Enables or disables statement and variable tracing.

GXSET TRACE	.state [.time]

Parameters

.state
ON
To start tracing
OFF
To end tracing.
.time
Optional parameter used with ON.
TIME
Causes a time-stamp record to be produced with the trace record.
NOTIME
Suppresses time-stamping.

GDDM calls

See "REXX" on page 6 for details of how to access GDDM calls from GDDM-REXX.

ERXMSVAR EXEC

Function: ERXMSVAR produces initialization statements for REXX variables that are associated with a mapgroup and map. It creates a sequential data set or PDS member which can be included in a REXX exec that uses GDDM run-time mapping calls.

Before ERXMSVAR is invoked, the sequential data set or PDS member must be allocated with a ddname of GDDMCOPY. Under MVS, the sequential data set or PDS member must be V format, with a maximum record length of 256 bytes, and allocated with a ddname of GDDMCOPY. Refer to the *GDDM Base Application Programming Guide* for more information.

ERXMSVAR	mapgroupname mapname prefix

Parameters

mapgroupname
The name of the generated mapgroup. The mapgroup must have the filetype (VM/CMS) or ddname (MVS) of ADMGGMAP. This is the filetype or ddname generated by GDDM-IMD. If **mapgroupname** ends with two dots, GDDM will supply the last two characters, which indicate the device class.
mapname
The name of the map.
prefix
The prefix used for the associated REXX variables. To ensure that REXX mapping variables have unique names, specify each prefix with an underscore (_) at the end.

Description: When GDDM Interactive Map Definition has been used to define and generate the map and mapgroup, ERXMSVAR may be executed.

All maps used with GDDM-REXX should be generated with the option to include field names. In GDDM-IMD panel 3.0 specify:

```
FIELD NAMES INCLUDED IN GENERATED MAPGROUP  ==> YES
```

Names of the form NO_NAME_1, NO_NAME_2, and so on are assumed if field names are not included. This is discussed further below.

Field naming rules:

1. Field names (including selector and adjunct names) follow the normal GDDM-IMD rules. The names can be seen in the GDDMCOPY file produced by the ERXMSVAR EXEC.

   Field names and adjunct suffixes are converted into names acceptable to REXX, by changing hyphens (–) to underscores (_). Thus, MY-FIELD becomes MY_FIELD, and if it has the color selector adjunct COL-SEL, that becomes MY_FIELD_COL_SEL.

   Adjunct suffixes depend on the language you selected when you generated the mapgroup (see Chapter 16,

"Application data structure for mapping" on page 357 for details).

2. If you do not use

   ```
   FIELD NAMES INCLUDED IN GENERATED MAPGROUP  ==> YES
   ```

   GDDM-REXX generates field names as follows:

   For fields

   ```
   prefix||"NO_NAME_"||field-number
   ```

   For adjuncts

   ```
   prefix||"NO_NAME_"||field-number||adjunct-suffix
   ```

   where:

   `prefix` Is as previously described.

   `NO_NAME_` Is the standard name given to all fields.

   `field-number` Is the GDDM-IMD sequence number of the field in the map (array indexes, if any, are lost).

   `adjunct-suffix` Is the suffix that appears on all adjunct variables. It takes the PL/I form, using underscores.

   For example:

   ```
   X_NO_NAME_3 = "           "
   X_NO_NAME_4 = "      "
   X_NO_NAME_4_COL_SEL = " "
   ```

Sample output: Assuming there were three arrays of fields called PROD, DESC, and COST (as there are in the sample map ERXORDER), the command

```
ERXMSVAR group1 map1 X_
```

generates:

```
/* GDDM-REXX: output from ERXMSVAR EXEC: . . 16:41:32*/
/* Initialize structure for */
/*     MAPGROUP: ERXORDD6 , MAPNAME: ERXORDER */
X_=""
X_PROD.1 = "           "
X_DESC.1 = "                              "
X_COST.1 = "           "
X_QTY.1 = "      "
X_QTY_COL_SEL.1 = " "
X_QTY_COL.1 = " "
X_TOT.1 = "        "
X_PROD.2 = "           "
 ...
 ...                                      "
 ...
X_TOTAL = "        "
X_MSG = "                                     "
X_ASLENGTH=432 /* length of ADS string */
```

Examples of REXX variable names used for a map without names:

```
X_NO_NAME_3 = "           "
X_NO_NAME_4 = "      "
X_NO_NAME_4_COL_SEL = " "
X_NO_NAME_4_COL = " "
```

Possible pitfalls: If you use a stemmed variable such as `stem.` do not use the same variable names in the EXEC that are used as any field names in your maps, as this could result in unwanted substitutions. For example:

```
title='Overdue orders'
stem.title=title
  ...
```

results in:

```
stem.Overdue orders=Overdue orders
```

The variable in your map `stem.title` will not have been altered.

Chapter 7. Symbol sets

This chapter describes how GDDM processes its various symbol set operations for the different device types.

The chapter also contains descriptions of the GDDM sample Image and Vector symbol sets that are supplied with GDDM. The sample symbol sets can be used by application programs instead of the defaults provided with GDDM.

How GDDM handles symbol sets

GDDM provides facilities for loading and using symbol sets other than the default characters, markers, and shading patterns. These may be image symbol sets (ISS), or vector symbol sets (VSS). Two methods of loading symbol sets are available:

- Loading image symbol sets directly into programmed symbol (PS) stores in the device
- Loading image symbol sets or vector symbol sets into GDDM storage.

PS stores are used for alphanumerics and mode-1 graphics text, GDDM storage for mode-2 and mode-3 graphics text.

For these operations, the symbol sets can be loaded from auxiliary storage, or passed as data from the application program.

In addition to being loaded by these operations, symbol sets can be passed as data between the application program and auxiliary storage.

Symbol sets can be tagged with country-extended code page (CECP) identifiers. CECP sets are automatically converted when they are used. So a set tagged with code page identifier 00037 (for the United States) is converted to represent code page 00297 when it is loaded into a French device.

Where possible, GDDM loads symbol sets into device storage. For the 3270-PC/G and 3270-PC/GX work stations, both image symbol sets and vector symbol sets can be loaded into the device, whereas for the 3179-G, 3192-G, and 3472-G displays, only image sets can be loaded.

Note: No symbol sets can be loaded into the 5550-family work stations or ASCII graphics displays.

Loading programmed symbol stores

Display devices and printers equipped with the programmed symbol (PS) feature contain PS stores that can be loaded with symbol definitions. These PS stores are used for:

- Storing additional or special symbol sets
- Storing symbols or cell definitions used in constructing a picture.

Symbol sets that are to be loaded into PS stores must have the same matrix dimensions as the device character cell. These are shown in Table 4 on page 243.

PS store numbers

The PS store number may optionally be specified as a parameter of the loading call (PSLSS or PSDSS). If specified, it must exist on the device in use at the time, and it must be a triple-plane store if a multicolor symbol set is to be loaded. Call statements are available to determine the number and types of PS stores in the device.

The specified store number controls the function key that can be used by the terminal operator to select the symbol set for data entry. The correspondence between store numbers and keys is:

Table 9. PS store number and PS key relationship

PS store number	PS key and indicator
2	A
3	B
4	C
5	D
6	E
7	F

A store number should always be specified if data entry using the symbol set is expected. The number need not be specified if data entry is not allowed by the application program.

When no store number is specified, the symbol set is loaded into an appropriate PS store, if one is available. Monochrome symbol sets may be loaded into either single-plane or triple-plane stores (for example: numbers 2, 3, 4, 5, 6, and 7 on a 3279 display), but multicolor symbol sets require triple-plane stores (for example: numbers 4, 5, and 7 on a 3279 display).

Symbol-set identification

Displays and printers identify loaded symbol sets by a one-byte symbol-set identifier. Usually, symbol sets are held on auxiliary storage. When a set is loaded into the PS store, a symbol-set identifier specified as a parameter in the loading call is associated with the data. It is then used to identify the symbol set during processing of the application program.

Reference to the symbol-set identifier takes one of two forms:

- A single character

  This form must have a character code greater than X'40', and it is used when identifying the symbol set associated with individual characters, as in the ASCSS

call. If this function or the related query function ASQSS is used, it is likely that the symbol-set identifier chosen will be an alphanumeric character.

- A full-word integer

  This form is used when specifying the symbol-set identifier to be associated with given data, an alphanumeric field, or a graphics character string.

The correspondence between the integer and the character specifications is:

- Characters “0” and “1” correspond to the integers 0 and 1. These refer to “read-only” character sets.
- Other characters correspond to their character codes. For example, “A” corresponds to 193.

Integer symbol-set identifiers in the range 224 through 239 are reserved for graphics use and cannot be assigned to loaded symbol sets.

Using preloaded PS sets

When GDDM is initialized, the current state of the PS stores is determined by a device query, which returns the identifier of any loaded sets. These preloaded sets are noted by the GDDM PS management routines, which maintain knowledge of the contents of the PS stores.

GDDM’s PSLSSC call **conditionally** loads a symbol set into a PS store only if the PS store does not already contain a symbol set with the specified identifier. Conditional loading can be used to optimize PS loading, but it must be used with care, because incorrect results occur if different symbol sets have the same identifier. For example, an application program may load a symbol set with a given identifier, and another program running subsequently on the same device may attempt a conditional load of a different set having the same identifier. This situation can be avoided if a convention is adopted that assigns unique identifiers to specific symbol sets.

Selecting symbol sets by device type

If an application program is designed to be used with different devices, it may be necessary to control symbol set loading on the basis of cell size. This can be done by using a GDDM symbol-set naming convention. The symbol-set name is specified as a parameter of the loading call. If the last character of the name is the period character “**.**”, GDDM replaces it by another character, depending on the current device.

In this way, a symbol set that matches the device in use can be retrieved from auxiliary storage and loaded. As a particular application, if a display containing PS is to be printed, this function allows the selection of a symbol set specific to the printer when printing begins.

For the details of which symbol sets are loaded for a particular device cell size, see Table 10 on page 264.

Using PS with graphics

This section does not apply to 3179-G, 3192-G, or 3472-G color display stations, 3270-PC/G and 3270-PC/GX work stations, IPDS printers, 5550-family work stations, the 5080 graphics system, ASCII graphics displays, and devices supported by GDDM-PCLK or GDDM-OS/2 Link, because PS is not used to construct the graphics for these devices.

When GDDM is constructing a picture, the assumption is made that all PS stores in the device are available for use except those that have either been loaded with symbol sets, or explicitly reserved by the application program. Because the number of PS stores is limited, if an application program uses both additional PS character sets and graphics construction, special attention to PS allocations may be required. This is especially true for printers, because only one PS store can hold a multicolor symbol set.

In general, PS stores should be loaded with any additional symbol sets before graphics picture construction is started, because the PS stores are also used for picture display. An attempt to load a symbol set when graphics are displayed is usually rejected by GDDM. Only when all graphics items are deleted from all pages do the PS stores become released for loading symbol sets.

If the programmer anticipates the need to load a PS store while graphics data is present, the PSRSV call is available to **reserve** a PS store. This must be done before any graphics calls are issued. The specified PS store is not used for graphics data, and is explicitly referred to in the call statement to load the symbol set. When the symbol set is no longer needed, the symbol set can be released from the reserved PS store, and another symbol set can be loaded, or, the PS store itself can be released.

In a windowing environment, the PS stores are allocated in the following order:

1. For symbol sets in the active window
2. For graphics in the active window
3. For graphics for window borders (all windows)
4. Any remaining PS slots are allocated for symbol sets and graphics in non-active windows.

Loading graphics symbol sets

Symbol sets that are not suitable for loading into PS stores can be loaded into GDDM storage. (For 3270-PC/G and 3270-PC/GX workstations, these symbol sets can also be loaded into the device; for the 3179-G, the 3192-G, and the 3472-G, image symbol sets can be loaded into the device.)

Four types of symbol set can be loaded in this way:

- Image symbol sets used as graphics text

- Image symbol sets or vector symbol sets used as marker symbols
- Image symbol sets used for shading graphics areas
- Vector symbol sets used for graphics text.

Unlike PS stores, there is no restriction on symbol size when loading image symbol sets into GDDM storage. Any size that can be created with the Image Symbol Editor can be used. However, when shading patterns are used, the symbol is truncated or padded to the cell size and repeated **at cell intervals**. Therefore, in most circumstances shading patterns should be the same size as the cell.

Devices other than 3179-Gs, 3192-Gs, 3472-Gs, 4224s, ASCII graphics displays, GDDM-PCLK, and GDDM-OS/2 Link devices:

For these devices, in graphics there is occasionally a choice between loading a symbol set into a PS store for use in mode-1, and loading it into GDDM storage and using mode-2. Mode-2 graphics text required if the character set does not match the device, or if exact positioning is required. If neither of these conditions exists, it should be remembered that the PS load transmits all characters in the symbol set to the device once only. Using the characters in mode-2 requires the transmission of only those characters actually used, but more than one cell definition may be transmitted for each.

For details of how to set the graphics text mode, see Chapter 3, “The GDDM calls” on page 21.

Also, for guidance information on mode-1 and mode-2 usage for graphics, see the *GDDM Base Application Programming Guide*.

3270-PC/Gs, 3270-PC/GXs, 3179-Gs, 3192-Gs, and 3472-Gs:

These displays support a maximum of two monoplane PS stores; the precise number depends on how the display has been configured. Programmed symbol sets are not used to construct graphics because the displays have their own graphics capability. In the interests of better performance, the device-provided default character sets should be used whenever possible. Only PS sets can be used for alphanumeric characters.

Note: The 3179-G, 3192-G, or 3472-G display stations, and 3270-PC/G or 3270-PC/GX work stations have a different pixel aspect ratio and default graphics character-box size from displays such as the 3279. Thus, character mode-1 graphics character strings and character mode-2 text and images appear differently on the two types of device.

To prevent storage problems in the display, any symbol sets that have been loaded (by using GSDSS or GSLSS calls) should be released when they are no longer needed. The storage occupied by these symbol sets is common to that used for storing segments, so loading unnecessary symbol sets can cause segment storage to be exhausted (thereby causing GDDM to enter unretained mode with a subsequent effect on performance).

It should also be noted that unless the workstation has enough symbol-set storage to hold the current user-defined pattern sets, the default shading patterns are used (GDDM issues a warning message when this happens).

For details of how to set the graphics text mode, see Chapter 3, “The GDDM calls” on page 21.

For information on using mode-1 and mode-2 graphics, see the *GDDM Base Application Programming Guide*.

PS overflow caused by picture complexity

Note: This section does not apply to the following devices because the problem of PS overflow does not arise with them:

- 3179-G, 3192-G and 3472-G display stations
- 3270-PC/G and 3270-PC/GX work stations
- 5550-family work stations
- The 5080 Graphics System
- Devices supported by GDDM-PCLK
- ASCII graphics displays

When a picture is extremely complex, it may require more PS stores than GDDM and the device can handle. This is known as PS overflow. When PS overflow occurs, message ADM0273 is issued to inform the user that the picture cannot be accurately completed.

In a windowing environment, this message is only issued if the overflow occurs in the active window.

The 4224 printer performs its own vector-to-raster conversion for graphics data. The graphics data stream that is sent to these printers contains GDF orders. The amount of storage available in these printers may not be enough to hold all of the graphics data that defines the picture. When this occurs, message ADM3282 is issued to inform the user that the picture cannot be accurately completed.

The FSCHEK function can be used to discover if PS overflow will occur when a picture is displayed. If PS overflow would occur, the error can be intercepted and action taken to simplify the picture or delete segments until it can be shown.

Using symbol sets in printing

When a call is issued to copy screen data to the printer, the names of symbol sets in use, both on the screen and in GDDM storage, are noted. These names include the final character “.” if it was originally specified, not the suffix that was substituted for it.

When the print operation begins, an attempt is made to reload the symbol sets. The appropriate suffix replaces the “.”, so that a printer symbol set is retrieved, if one exists on

auxiliary storage. If not, the default symbol set is used and an error message is issued.

Note that if the symbol set was loaded into the display by a conditional PS load, a conditional load is also performed before printing. Therefore, the convention associating symbol sets with unique identifiers must apply for both displays and printers.

Because there may be more PS stores available on a display than on a printer, if an application program explicitly uses PS stores, a picture that can be displayed may not print. Also, because only one triple-plane store is available in the 3287 Printer (Models 1C and 2C), if the application reserves this store for a non-graphics symbol set when the print request is processed, multicolor graphics printing is not performed correctly.

Using DBCS symbol sets

For Kanji/Hangeul applications that have double-byte character string (DBCS) symbol sets installed, this type of symbol set can be used directly (by the application program loading the required symbol set and using the definition in the normal manner) or indirectly (by the application program indicating that it requires to use DBCS symbol sets). In the second case, if the GSCS call specifies character set 8 (DBCS) or if mixed (single-byte and double-byte) character strings are enabled (by specifying MIXSOSI=YES in GDDM's external defaults), GDDM recognizes DBCS characters and uses the first byte of the character to identify the symbol set to be loaded and the second byte to retrieve the symbol definition.

Applications access Simplified Chinese vector symbol sets by setting the DBCSDNM(ADMIK,ADMVC) external default.

GDDM's external defaults define whether mixed strings are enabled and indicate the maximum number of DBCS symbol sets of each type that are to be loaded concurrently. When this maximum number is reached, the least recently used symbol set is unloaded to allow the currently required symbol set to be loaded. For details of the external defaults, see Chapter 18, "External defaults" on page 379.

For graphics, DBCS symbol sets are available for mode-2 and mode-3 only.

Naming conventions for sample image symbol sets

Except for shading patterns with a suffix of N or R (and ADMDHIPK), the last character in the name of each image symbol set conforms to the convention for generic retrieval by GDDM, showing the cell size of the symbol set.

Where there is a suffix of N or R, patterns are defined on an 8 by 12 cell size. This allows complete shading, as defined in the GSLSS call.

The following table shows the suffix that GDDM uses to replace the "." substitution character that is used in a GSLSS call (for a graphics symbol-set name), in a PSLSS or PSLSSC call (for an alphanumerics symbol-set name), or in an SSREAD or SSQF call (for either alphanumerics or graphics).

Table 10. Cell sizes for sample image symbol sets

Substituted suffix	Cell size in display points (width by depth)
A	9 by 16
C	9 by 12 (monochrome)
D	9 by 12 (multicolor)
E	9 by 10 (alphanumerics only)
G	10 by 8 (monochrome)
H	10 by 8 (multicolor)
J	Family-4 high-resolution symbol sets (400 pixels per inch or greater). See Note 5
K	20 by 18 (alphanumerics only)
L	Family-4 medium-resolution symbol sets (less than 400 pixels per inch). See Note 5
M	32 by 32. See Note 4
O	32 by 32. See Note 4
N	8 by 16 (graphics only)
Q	24 by 30 (monochrome)
R	12 by 20 12 by 24 (Katakana)
S	9 by 21 (alphanumerics only)
T	12 by 16
U	Plotter symbol sets

Notes:

1. If the device has a cell size that is not one listed above, GDDM selects the suffix that corresponds to the smallest containing cell size. For example, for a device cell size of 9 by 14, GDDM selects an image symbol set with a cell size of 9 by 16 (suffix A).
2. If the device cell size does not fit into any of the cell sizes given in the table, GDDM selects an image symbol set with a cell size of 9 by 16 (suffix A).
3. For a family-3 printer, the character A is always used as the suffix.
4. GDDM provides sample image symbol sets with M and O as the suffix. However, M and O are not characters in the substitution rules.
5. User shading patterns of size 32 by 32 or 64 by 64 may be used for family-4 devices. The sample pattern set ADMPATTJ is 64 by 64 and ADMPATTL is 32 by 32. You need to set OFFORMAT to IMAGE for printers that do not support loading of shading patterns.

Sample image symbol sets

Table 11. Sample image symbol sets

Set name	Contents
ADMCOLSD ADMCOLSN ADMCOLSR	Shading patterns, that create the appearance of 64 color shades.
ADMDHIIA ADMDHIIC ADMDHIIE ADMDHIIG ADMDHIIK ADMDHIIN ADMDHIIQ ADMDHIIR ADMDHIIS ADMDHIIT	The standard CECP set of characters.
ADMDHIMA ADMDHIMC ADMDHIMG ADMDHIMK ADMDHIMN ADMDHIMQ ADMDHIMR ADMDHIMT	Ten standard markers, that correspond to the defaults provided with GDDM. See the description of the GSMS call.
ADMDHIPA ADMDHIPC ADMDHIPG ADMDHIPJ ADMDHIPM ADMDHIPN ADMDHIPO ADMDHIPR	Seventeen standard patterns, that correspond to the defaults provided with GDDM. See the description of the GSPAT call. ADMDHIPJ is for use on an IBM 4250 high-resolution printer, ADMDHIPM is for use on IBM 3800-3 and 3800-8 medium-resolution printers, and ADMDHIPO is for use on low-resolution printers.
ADMDHIPK	Eight patterns, that can be used when producing color masters on high-resolution and medium-resolution printers.
ADMDHIPL	Shading patterns, that can be used for converting colors into shades of gray on high and medium-resolution printers.
ADMIPATA ADMIPATC ADMIPATG ADMIPATN ADMIPATR	Seventeen standard patterns, that correspond to the defaults provided with GDDM. Used with the Image symbol Editor INFILL function. See the description of the GSPAT call.
ADMITALA ADMITALC ADMITALG ADMITALK ADMITALS	Script CECP characters.
ADMPATTA ADMPATTC ADMPATTG ADMPATTJ ADMPATTL ADMPATTN ADMPATTR	Sixty-four sample geometric shading patterns. See the description of the GSPAT call.
ADMIKxx	Double-byte character set 16 x 16 image characters, where “xx” is in the range X'41' through X'68'.
ADMDISJN ADMDISJR	Image symbols for code page 1027 for use with OS/2-J Katakana terminals.
ADMDISKA ADMDISKC ADMDISKG	Image symbols code page 290 for use with Katakana displays and printers.
ADMDISKN ADMDISKR	8x16 and 12x24 image symbols for code page 290 for use with 5550-family multi-stations.
ADMDcIIA ADMDcIIC	Other non-CECP image symbol sets, where c is: • C for Cyrillic (code page 1025) • L for Latin 2 (code page 870) • M for Baltic Multilingual (code page 1112) • O for Estonia (code page 1122) • U for Turkey (code page 1026) • Y for Greece (code page 875)
Note: The symbol sets are only provided as samples. GDDM does not ensure that all styles of characters and patterns are provided for all possible suffix characters.	

Sample vector symbol sets

GDDM’s sample vector symbol sets are as shown below:

Table 12 (Page 1 of 2). Sample vector symbol sets

Set name	Contents
ADMDHIMJ	Contains the GDDM vector marker symbols for use by the Interactive Chart Utility.
ADMDHIMV	Contains ten standard vector markers that correspond to the defaults provided with GDDM.
ADMDHIVJ	Contains the default vector symbol set for the 4250 page printer.
ADMDHIVM	Contains the default vector symbol set for a 3800 Model 3 or a 3800 Model 8 page printer.
ADMDVIH	Contains the default vector symbol set for 3270-PC/G, /GX, or /AT workstations.
ADMDVECP	CECP default vector symbol set
ADMDVSS	The default vector symbol set for code page 00351 (USA version).
ADMDVSSB ADMDVSSD ADMDVSSE ADMDVSSF ADMDVSSG ADMDVSSI ADMDVSSN ADMDVSSS ADMDVSSV	National Language versions of the vector symbol sets for code page 00351: Brazilian Danish U.K.English French German Italian Norwegian Spanish Swedish

Table 12 (Page 2 of 2). Sample vector symbol sets

Set name	Contents
ADMDVSSJ	The vector symbol set for code page 01027.
ADMDVSSK	The default vector symbol set for code page 00290.
ADMDVSSc	Other non-CECP vector symbol sets, where c is: • C for Cyrillic (code page 1025) • L for Latin 2 (code page 870) • M for Baltic Multilingual (code page 1112) • O for Estonia (code page 1122) • U for Turkey (code page 1026) • Y for Greece (code page 875)
	CECP vector symbol sets:
ADMU⋆ARP	Area Filled Roman Principal
ADMU⋆CIP	Complex Italic Principal
ADMU⋆CRP	Complex Roman Principal
ADMU⋆CSP	Complex Script Principal
ADMU⋆DRP	Duplex Roman Principal
ADMU⋆FSS	Filled Sans Serif
ADMU⋆GEP	Gothic English Principal
ADMU⋆GGP	Gothic German Principal
ADMU⋆GIP	Gothic Italian Principal
ADMU⋆KRF	Thick Round Filled
ADMU⋆KRO	Thick Round Outlined
ADMU⋆KSF	Thick Square Filled
ADMU⋆KSO	Thick Square Outlined
ADMU⋆MOD	Modern
ADMU⋆NSF	Thin Filled
ADMU⋆NSO	Thin Outline
ADMU⋆ORP	Outline Roman Principal
ADMU⋆SHD	Shadow
ADMU⋆SRP	Simplex Roman Principal
ADMU⋆TIP	Triplex Italic Principal
ADMU⋆TRP	Triplex Roman Principal
ADMU⋆TSS	Triplex Sans Serif
⋆ is U	– proportionally spaced
⋆ is V	– nonproportionally spaced
⋆ is W	– proportionally spaced (wider spacing for compatibility with GDDM Version 1)
	See “Illustrations of vector typefaces” on page 267.
ADMVCxx	Simplified Chinese vector symbol set characters, where “xx” is in the range X'41' through X'6C'.
ADMVKxx	Sample double-byte Kanji stick character set characters, where “xx” is in the range X'41' through X'68'.
ADMVQxx	Sample double-byte Kanji Mincho character set vector characters, where “xx” is in the range X'41' through X'68'.

Table 12 (Page 2 of 2). Sample vector symbol sets

Set name	Contents
ADMDVctd	Other non-CECP symbol sets, where c is: • C for Cyrillic (code page 1025) • L for Latin 2 (code page 870) • M for Baltic Multilingual (code page 1112) • O for Estonia (code page 1122) • U for Turkey (code page 1026) • Y for Greece (code page 875) t is: • C for Courier • H for Helvetica • T for Times New Roman d is: • R for Roman medium • I for Italic medium • S for Roman bold • J for Italic bold

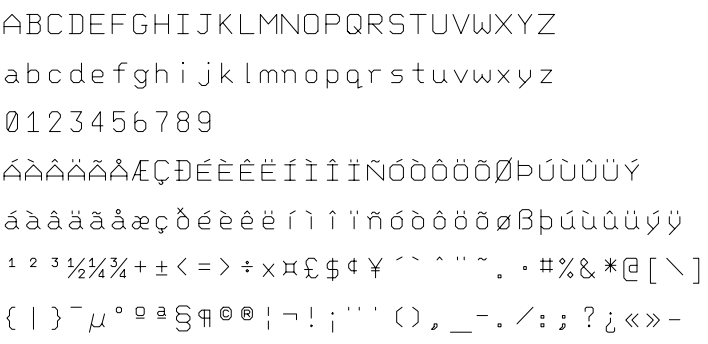
Notes:

1. It is not possible to use the Image Symbol Editor on the sample vector symbol sets. The Vector Symbol Editor is part of GDDM-PGF.
2. As supplied, CECP symbol sets are ordered according to the USA CECP, 00037, and are so tagged. GDDM converts them to the device code page when they are loaded by an application program.
3. All the IBM-supplied sample vector symbol sets have names starting with “ADM”; this aids identification and serviceability. However, installations may find it more convenient to generate copies of these symbol sets, using other names. If necessary, the Image or Vector Symbol Editor can be used to save the symbol sets under different names. The symbol sets are shown in “Illustrations of vector typefaces” on page 267.
4. It should not normally be necessary to alter a CECP set. However, if an editor is used to change a CECP symbol set, the application code page should first be set to be the same as that of the symbol set being edited. GDDM supplies the CECP sets ordered according to code page 00037.

Illustrations of vector typefaces

The vector symbol sets shown on the following pages are supplied as part of GDDM.

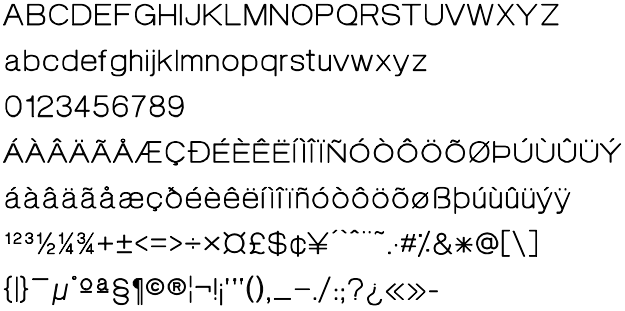
ADMDVECP is the default vector symbol set, used by GDDM if you do not explicitly specify one.



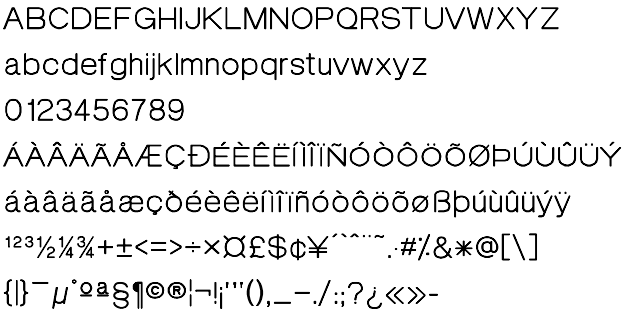
Three variants of each of the following vector symbol sets can be used, where the fifth letter of the name indicates its form (*U* indicates a proportional form, as illustrated; *V* indicates a nonproportional form; *W* indicates a proportional form with a wider space character – so the first one is supplied as ADMUUARP, ADMUVARP, and ADMUWARP). The ADMUW... sets are supplied for compatibility with previous releases of GDDM; the ADMUU... sets are recommended for new applications.

The illustrations on the following pages show the characters at sizes that are consistent within each typeface, although the sizes vary from one typeface to another.

ADMUUFSS – filled sans serif



ADMUUTSS – triplex sans serif



ADMUUMOD – modern

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´`^¨˜ .·#%&*@[\ ]
{ | } ¯µ˚ºª§¶©®¦¬ ! ¡ ” ' ( ) , _ – . / : ; ?¿«»-

ADMUUKRF – thick round filled

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´`^¨˜.·#%&*@[\]
{|}¯µ˚ºª§¶©®¦¬¡"'(),_–./:;?¿«»-

ADMUUKSF – thick square filled

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´`^¨˜.·#%&*@[\]
{|}¯µ˚ºª§¶©®¦¬¡¨'(),_–./:;?¿«»-

ADMUUNSF – thin filled

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´ˆ¨˜.·#%&*@[\]
{|}¯µ°ºª§¶©®¦¬¡¡"'(),_–-./:;?¿«»-

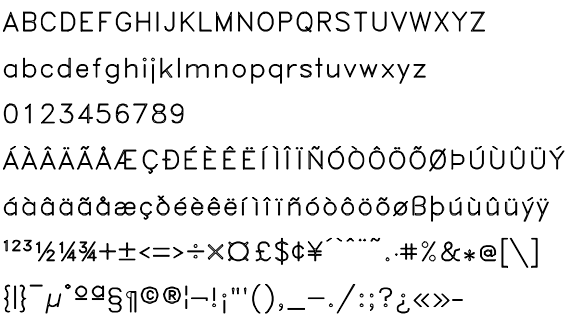
ADMUUKRO – thick round outlined

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´ˆ¨˜.·#%&*@[\]
{|}¯µ°ºª§¶©®¦¬¡¡"'(),_–-./:;?¿«»-

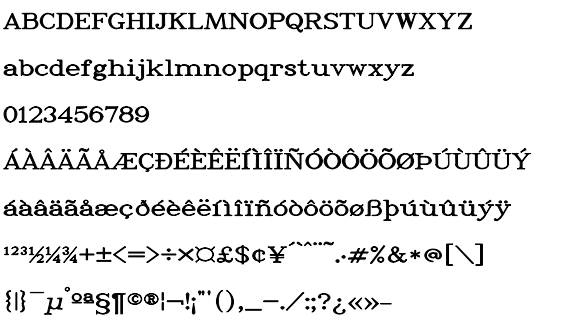
ADMUUKSO – thick square outlined

ABCDEFGHIJKLMNOPQRSTUVWXYZ
abcdefghijklmnopqrstuvwxyz
0123456789
ÁÀÂÄÃÅÆÇÐÉÈÊËÍÌÎÏÑÓÒÔÖÕØÞÚÙÛÜÝ
áàâäãåæçðéèêëíìîïñóòôöõøßþúùûüýÿ
¹²³½¼¾+±<=>÷×¤£$¢¥´ˆ¨˜.·#%&*@[\]
{|}¯µ°ºª§¶©®¦¬¡¡"'(),_–-./:;?¿«»-

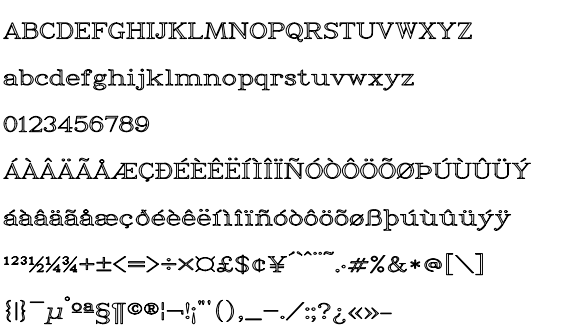
ADMUUNSO – thin outline



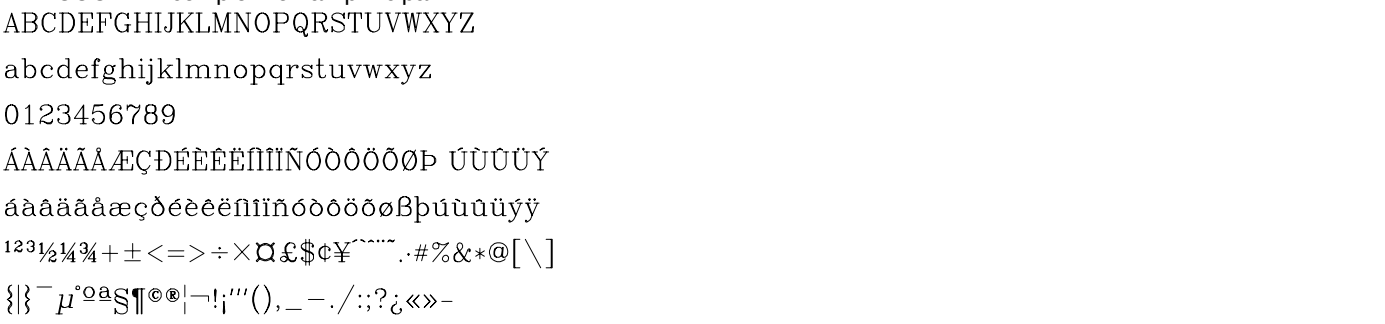
ADMUUARP – area-filled roman principal



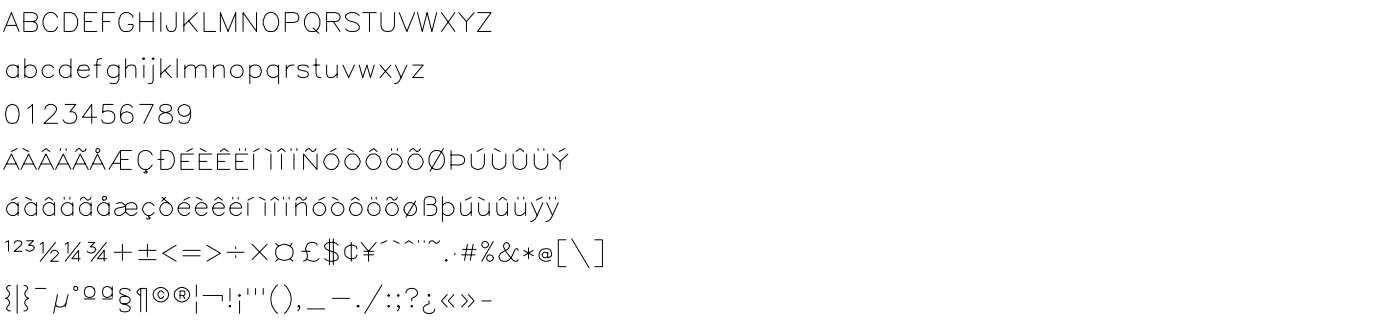
ADMUUORP – outlined roman principal



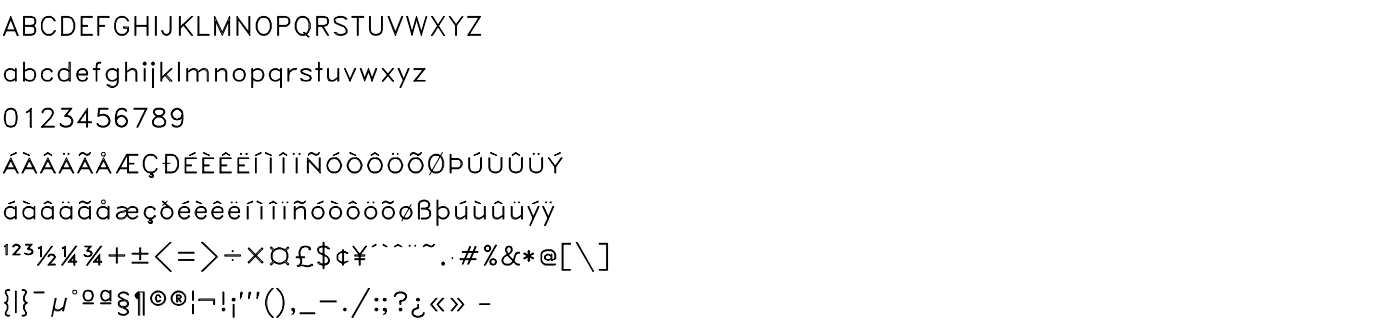
ADMUUCRP – complex roman principal



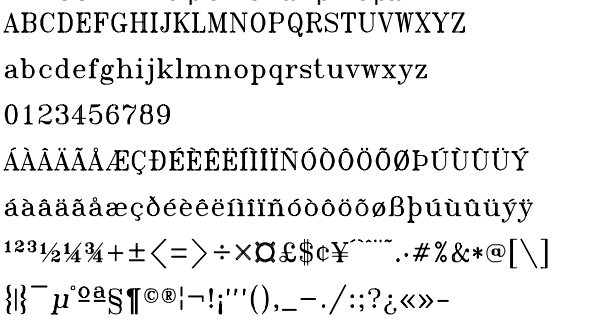
ADMUUSRP – simplex roman principal



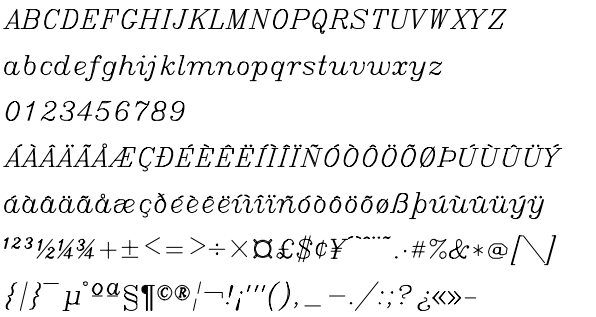
ADMUUDRP – duplex roman principal



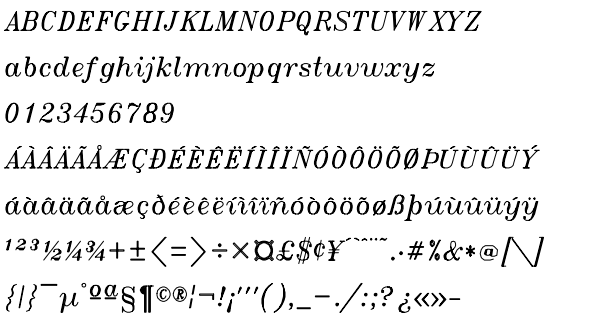
ADMUUTRP – triplex roman principal



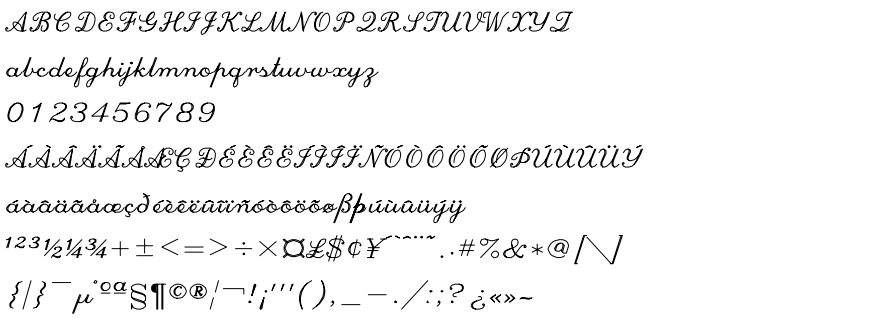
ADMUUCIP – complex italic principal



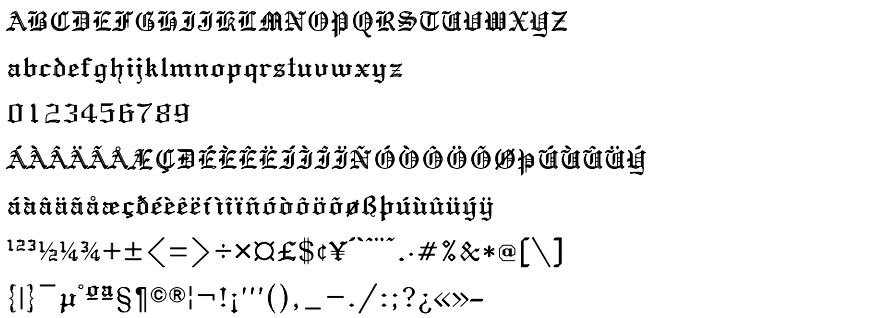
ADMUUTIP – triplex italic principal



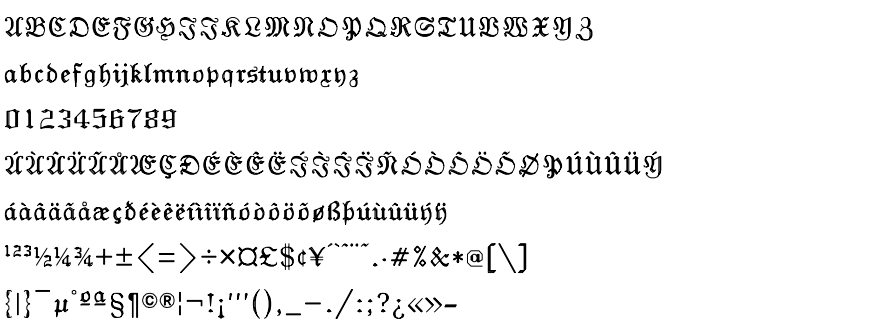
ADMUUCSP – complex script principal



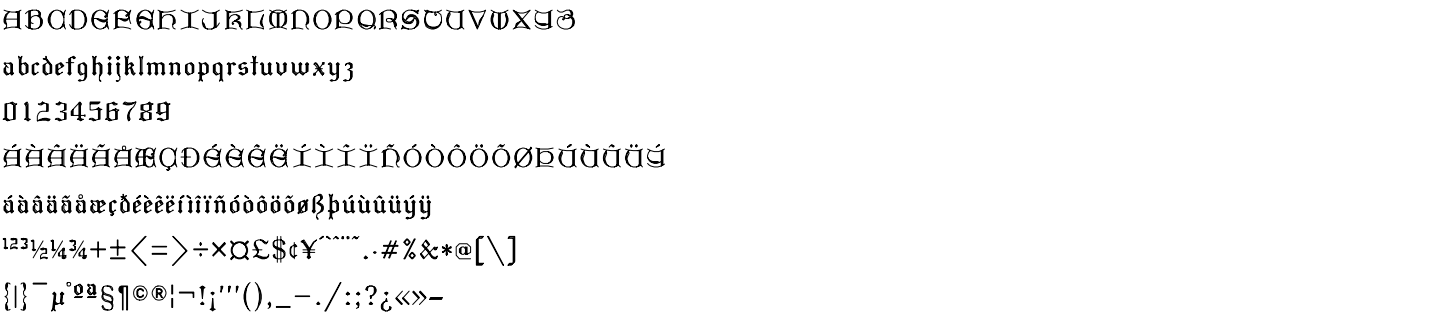
ADMUUGEP – Gothic English principal



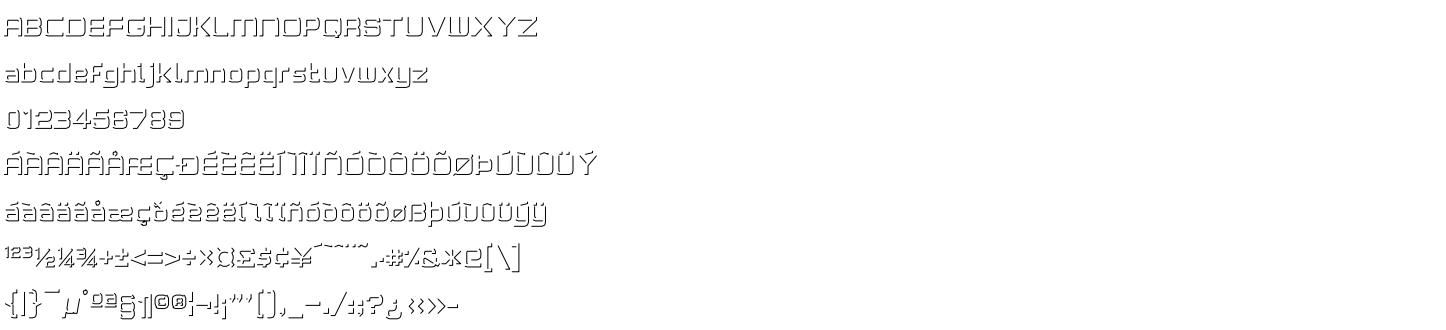
ADMUUGGP – Gothic German principal



ADMUUGIP – Gothic Italian principal



ADMUUSHD – shadow



Chapter 8. Symbol set formats

This chapter describes the formats for vector symbol sets (VSS) and image symbol sets (ISS) held on files (or passed as parameters in symbol-set manipulation calls).

In either case, definitions start with a two-byte field that gives the total length of the definitions (including the length field). Then follows one or more definition components, each of which is in one of the formats described below (depending upon whether the definitions relate to an image symbol set (ISS) or to a vector symbol set (VSS)).

Table 13. Symbol-set definition format

Byte	Field length	Contents
0	2	Definition length
2 2 ⋮	L1 2 ⋮	Component 1 Component length (L1) ⋮
2 + L1 2+ L1 ⋮	L2 2 ⋮	Component 2 Component length (L2) ⋮
⋮	⋮	⋮

The following rules must be observed. For the purpose of these rules, pattern and marker definitions are treated as MODE=2.

1. ISS and VSS components must not be mixed within a definition. ISS definitions cannot be loaded as MODE=3; VSS definitions can be loaded only as MODE=3.
2. For ISS definitions, the following considerations apply to the width (P) and depth (Q) of the cell matrix in display points:

   When either is specified as zero or is not specified, it is assumed to be equal to the cell width or depth of the actual device (except for format type '00001'B, where P is assumed to be 9).

   When the format type is '00001'B, P (specified or assumed) must be 9.

   When P is not a multiple of eight for row-loading format (type '00011'B), the storage occupied by each row must be padded on the right with zero bits to the next byte boundary.

   Similarly, when Q is not a multiple of eight for column-loading format (type '00101'B), the storage occupied by each column must be padded at the bottom with zero bits to the next byte boundary.
3. For MODE=1 definitions, the data format must be one that is supported by the actual device to which the definitions are to be loaded. One or more components may be specified, either to define different color planes for a multicolored definition, or to reduce the total length in cases where only widely-scattered character codes are to be loaded. Although checks are made, it is possible for a symbol set definition to pass these checks and still be rejected by the device or controller when the definitions are actually transmitted.
4. For MODE=2 definitions, only one component may be specified for a monochrome definition, or exactly three components (one for each color) for a multicolored definition. In the latter case, the starting character code and the number of codes defined must be the same for all three color planes.
5. For MODE=3 definitions (VSS), only one component may be specified.
6. The CLEAR bit is not supported by GDDM, although its setting is not altered by GDDM before transmitting MODE=1 definitions.

Image symbol set component format

Table 14. Image symbol set component format

Byte	Field length	Content	Meaning
0	2	LENGTH	Total length of structure (including LENGTH field).
2	1	X'06'	Image Symbol Set type
3	1	FLAGS	
		B'1.......'	EXTENDED – extended format of definition.
		B'.1......'	CLEAR – all definitions in the specified symbol set (plane) are to be cleared before processing the definitions.
		B'..1.....'	SKIPSUPP – skip is to be suppressed after printing a row that contains any symbol from this ISS.
		B'...xxxxx'	TYPE – the data format for the definitions. See Note 1.
		B'...00001'	18-byte form – the first two bytes contain a 16-bit vertical slice; the following 16 bytes contain 8-bit horizontal slices. (For a 9 by 12 cell, the last 4 bytes contain binary zero). Equivalent to cell format 1 for displays.
		B'...00011'	Row loading – bits within each row go from left to right, padded to a byte boundary; successive rows are from top to bottom. Equivalent to cell format 3 for graphics.
		B'...00101'	Column loading – bits within each column go from top to bottom, padded to a byte boundary; successive columns are from left to right. Equivalent to cell format 2 for printers.
4	1		Reserved.
5	1	CP0	Starting character code within this symbol set (in range X'41' — X'FE').
6	1		Reserved.
7	1	LEXT (see Note 2)	Length of extended parameters; gives the length of fields from and including LEXT to the end, but excluding CDEF. Must be specified as 6, if present.
8	1	EXTFLAGS (see Note 2)	
		B'1.......'	APA – all points in the cell are not addressable
		B'.1......'	CB – no LCID compare
		B'..1.....'	OB – no operator selectability
		B'...xxxxx'	Reserved.
9	1	P (see Note 2)	Number of x units in dot matrix.
10	1	Q (see Note 2)	Number of y units in dot matrix.
11	1	SUBSN (see Note 2)	
			Subsection identifiers, as follows:
		X'00'	One-byte codes
		X'42' – X'7F'	Subsection identifiers for two-byte coded data
		Other	Reserved.
12	1	COLOR (see Note 2)	
		B'xxxxx...'	Reserved
		B'.....000'	All color planes to be loaded
		B'.....001'	Blue plane to be loaded
		B'.....010'	Red plane to be loaded
		B'.....100'	Green plane to be loaded
		B'.....xxx'	(Other patterns of bits 5 – 7) Reserved.
7 or 13	V	CDEF (CP0-CPn)	Symbol definitions, starting at character code CP0, in ascending order, and in the format defined by Byte 3.

Notes:

1. "Cell formats" are specified in the Image Symbol Editor. For more information, see the *GDDM Using the Image Symbol Editor* book.
2. Present only if bit 0 (EXTENDED) of FLAGS is set.

Vector symbol set component format

Table 15. Vector symbol set component format

Byte	Field length	Content	Meaning
0	2	LENGTH	Total length of structure (including LENGTH field).
2	1	X'01'	Vector Symbol Set type
3	1	FLAGS B'1.......' B'.x......' B'..1.....' B'...xxxxx' B'...00001' B'...00010' B'...00011'	 EXTENDED – definition is in extended format Ignored. SHADED– all symbols defined are to be shaded using the default shading pattern. This has the effect of surrounding each symbol definition implicitly by GSAREA and GSENDA. TYPE – the data format for the definitions (see Note 1) Type 1 Type 2 Type 3
4	1		Reserved (must be zero).
5	1	CP0	Starting character code within this symbol set (in range X'00' — X'FF').
6	1	FLAGS B'1.......' B'.1......' B'..000000'	 PROPORTIONAL SPACING – each index entry is extended by a halfword value specifying the width of each symbol. If this flag is off, each symbol has the width P. Valid only for type-3 definitions. LINES ONLY – only the following GDF orders are contained within the symbol definitions: {Extended order} line {Extended order} line at current position End of data. Valid only for type-3 definitions. reserved (must be zero)
7	1	LEXT (see Note 2)	Length of extended parameters; gives the length of fields from and including LEXT to the end, but excluding CDEF. Minimum value is 1. Maximum value is 9.
8	1	(see Note 2)	Reserved (must be zero).
9	2	P (see Note 2)	Range of x (0 through P). If this operand is not present, or it is specified as 0, then the value 15 is assumed.
11	2	Q (see Note 2)	Range of y (0 through Q). If this operand is not present, or it is specified as 0, then the value 15 is assumed.
13	2	(see Note 2)	Reserved (must be zero).
15	1	CPn *	Last character code within this symbol set. If this operand is not present, X'FE' is assumed. CPn must not be less than CP0.
7 or 16	V	CDEF (CP0-CPn)	Symbol definitions, starting a character code point CP0, ascending order. See page 277 for format types 1, 2, and 3.

Notes:

1. For VSS, each symbol is formed by lines and (for type 3) curves. Three types are defined:
2. Present only if bit 0 (EXTENDED) of FLAGS is set.

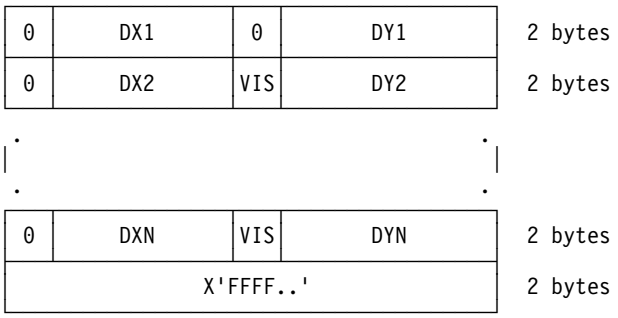
P and Q together define the character box within which a normal symbol fits. The bottom left-hand corner of the box is (0,0) and the top right-hand corner is (P,Q).

Undefined character codes are generally displayed as a blank, but see the description of the GSCHAR call.

Characters are always drawn using the default line type and line width, and with the default area pattern.

Type 1: The definitions start with an index of (CPn + 1 - CP0) two-byte values. Each index value is the offset from the start of CDEF (that is, from the start of the index) of the start of the definition of the corresponding character code (CP0 through CPn). This index must always be present in its entirety, even if not all the characters in the code range are defined. The maximum length of the index, if CP0 is specified as X'00', and CPn as X'FF', is therefore 256 by 2 bytes. Undefined values should be represented by a zero in the index.

Each character is defined as a series of points that define the shape of the character. Each point defines either a line from the preceding point, or a move to be performed to that point. The endpoints of each line (or move) are given by an (x,y) coordinate pair of signed relative values (relative to the previous coordinate, or to the bottom left-hand of the character box for the first coordinate pair). Each coordinate pair occupies two bytes (one byte for the x coordinate, and one for the y). If the first stroke is a line rather than a move, the line is drawn from the bottom left-hand corner of the box. The top bit of the y-coordinate byte is set if the stroke to that point is visible (that is, line rather than move); after the last coordinate pair, two bytes of all 1 bits indicates the end of the definition for that symbol. This format of an individual code point symbol is:



Type 2: For type-2 format, the endpoints of each line (or move) are given by a (dx,dy) coordinate pair of signed relative values. Each coordinate pair occupies four bytes (two bytes for the x coordinate, and two for the y), and is preceded by two bytes of flag bits, so that each point requires a total of six bytes. Each symbol consists of a series of these point definitions, defining the lines and moves needed to draw it. The start is from the bottom left-hand of the character box (0,0). The last point for a particular symbol is recognized by means of a flag, in the flag halfword.

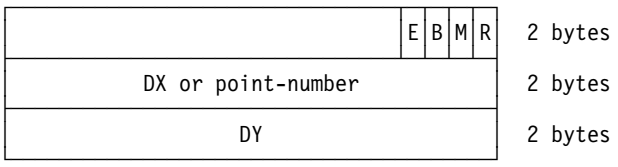
One of the flags designates "branch." This means that, instead of the dx halfword, a point number is specified, to which to branch for the remaining definitions for that symbol. The actual offset within CDEF is given by:

```
offset=point-number*6
```

In the case of a branch, the dy value is ignored.

As with type 1, CDEF starts with an index, with `(CPn+1-CP0)` entries corresponding to symbol codes CP0 through CPn. Each entry is a branch, as defined above, which in effect defines the starting position of a symbol.

This format of an individual point is illustrated below.



where:

E bit (bit 12) is set if this is the last point for the current symbol
B bit (bit 13) is set if this is a branch
M bit (bit 14) is set if this is a move, not a line
R bit (bit 15) is ignored.

Type 3: The definitions start with an index, just the same as the index for type 1. However, if the "PROPORTIONAL SPACING" flag in the header is set, each two-byte index offset entry is followed by a two-byte signed symbol width. This makes the entire index twice its normal size. Each symbol width value must be in the range

```
(-P < w < 0) or (0 < w <= P)
```

where the values `0` and `-P` are reserved. If the width is positive, the boundaries of the symbol (for spacing purposes only) are the left-hand side of the box, and a line "w" to the right of the left-hand side. If the width is negative, the boundaries are the right-hand side, and a line "–w" to the left of the right-hand side. Thus a width of "+P" is the default (full) width.

A type-3 symbol definition consists of a series of GDF (graphics data format) orders. These typically specify lines and curves that make up the symbol. The orders for a given symbol are terminated by an end-of-data marker, which is a single byte with the value `X'FF'`. All orders should be a complete number of halfwords, and, for performance reasons, should be aligned on a halfword boundary.

See Chapter 10, "GDF order descriptions" on page 281 for a description of GDF orders. A symbol generated by the Vector Symbol Editor typically uses the following orders:

- Line (`X'C1'`)
- Line at Current Position (`X'81'`)
- Fillet (`X'C5'`)
- Fillet at Current Position (`X'85'`)
- Area (`X'68'`).

Whenever a type-3 symbol is processed, a particular type of coordinate data is assumed. This depends on the values of P and Q. If both P and Q are less than 128, the default is one-byte signed absolute coordinates. If either P or Q are greater than 127, the default is two-byte signed absolute coordinates.

If the SHADED flag in the header is set, each symbol is drawn, using the default shading pattern, as though that symbol were enclosed in "Begin Area" and "End Area" orders. These orders are **implicit**. If the SHADED flag in the header is **not** set, individual shaded symbols should include an **explicit** "Begin Area" order and an explicit "End Area" order (just before the `X'FF'` marker).

Chapter 9. GDDM object file formats

GDDM supports the following object file types:

Table 16. GDDM object types

File type	Description of object
ADMSYMBL	Symbol set
ADMGGMAP	Generated GDDM mapgroup
ADMSAVE	FSSAVE file
ADMCFORM	Chart format file
ADMCDATA	Chart data file
ADMGDF	GDF file
ADMCDEF	Chart definition file
ADMPROJ	Projection definition file
ADMIMG	Image data file

Record structure

Every object consists of a header record and a number of data records. Every record in a stored GDDM object is 400 bytes long. The first 20 bytes of each record – whether a header or a data record – comprise a record identification field, which is used as a source key when the object is stored on a keyed database (as in CICS or IMS). The remaining 380 bytes of the header record provides more information on the object; the remaining 380 bytes of subsequent records comprise the object data.

Table 17. GDDM stored object file format

Length (bytes)	Content	Record type
20 380	Identification field Information field	Header record
20 380	Identification field Data field	Data record 1
⋮		
20 380	Identification field Data field	Data record n

The record identification field

The record identification field occupies the first 20 bytes of all records – both header record and data records.

The first eight bytes of this field contain the name of the object. This name is the same in all the records of the object.

The second eight bytes of this field contain the object type (see Table 16), and is also the same for all the records in the object.

The last four bytes of the record identification field contain the record sequence number, starting at 1, in fixed binary form.

Table 18. GDDM stored object — record identification field format

Offset	Length	Data type	Content
0	8	CHAR(8)	Object name
8	8	CHAR(8)	Object type
16	4	FIXED(31)	Record sequence number

The header record information field

The remaining 380 bytes of the header record provide extra information about the record, such as the GDDM version and release number, and the date and time the record was encoded. The format is:

Table 19. GDDM stored object — header record information field format

Offset	Length	Data type	Content
20	4	FIXED(31)	GDDM object: V1R1 Length of object V1R2 X'0000 0010' V1R3 and later X'0000 0000'
24	4	CHAR(4)	Reserved
28	4	CHAR(4)	GDDM Version and release (for example: '1030' for Version 1, Release 3.0)
32	4	FIXED(31)	Object major type (same as type in record identification field)
36	4	FIXED(31)	Object minor type: Image symbol sets 1 Vector symbol sets 2 Others 0
40	4	FIXED(31)	Length of supplied user comments
44	8	CHAR(8)	Date and time stored (encoded) – date (00YYDDD+ format) – time (0HHMMSS+ format)
52	20	CHAR(20)	Date and time stored (EBCDIC)
72	8	CHAR(8)	Reserved (must be all X'00')
80	255	CHAR(255)	Up to 255 bytes of user comments
335	63	CHAR(63)	Reserved
398	2	FIXED(16)	Code page identifier

Data records

The second and subsequent records in a GDDM stored object contain the object data. The first 20 bytes of these records constitute record identification fields, as defined in Table 18, leaving 380 bytes for the data proper.

For objects generated by GDDM Version 1 Release 2 or later, the data proper consists of one or more data blocks.

Each data block contains a two-byte length field followed by up to 32000 data bytes as defined for the particular type of object.

For example, a symbol-set object contains just one data block starting with a two-byte length field, followed by data bytes as defined in Chapter 8, “Symbol set formats” on page 275. These data bytes themselves start with two-byte length fields.

Chapter 10. GDF order descriptions

Graphics data format (GDF) is a means of storing pictures. GDDM uses it internally, and also makes it available to application programs. It consists of a set of orders with similar meanings to the GDDM graphics call statements.

In many cases there is a one-for-one mapping between GDF orders and GDDM call statements.

GDDM supports a picture prolog that contains information about the size of the picture and the symbol sets used in the picture. A detailed description of the orders that relate to the picture prolog is given under “Picture prolog” on page 302. The information the picture prolog provides is:

- The coordinate type
- The picture boundary
- The picture scale and aspect ratio
- The symbol sets that are referenced
- The drawing defaults information.

The initial Comment order in the generated GDF is retained for compatibility with previous releases of GDDM.

Compatibility

GDDM ensures upward compatibility of GDF orders from previous releases to the current release. The orders are **not** downward-compatible from the current release to previous releases.

Saving GDF orders

Applications can save GDF orders for later use as follows:

- As application-written GDF files (GDDM Version 1 Release 2 onwards)

  Use GSGET to move GDF orders from GDDM into application-program storage. The application program can then write these to auxiliary storage.

- As GDDM-written ADMGDF objects produced from Version 1 Release 4 onwards.

  Use GSSAVE to save GDF orders as a specially formatted ADMGDF object on auxiliary storage. This object contains the name of the file in columns 1 through 8, and “ADMGDF” in columns 9 through 14 of each record. The ADMGDF objects can be processed by a GDDM application using GSLOAD.

GDF can be retrieved in two formats, fixed or floating point. Floating-point GDF corresponds as closely as possible to the GDDM calls used to generate the picture. The data primitives will have been clipped, only if the application requested clipping using a GSCLP call statement. Fixed-point GDF does not necessarily match the original commands (the data is always clipped).

The GDF data that results may not necessarily resemble the original commands used to generate the picture because these have been processed to suit the primary device in use. For example, coordinates will have been converted to an internal coordinate system with some loss of precision. Complex primitives (such as curved fillets) may have been simplified and approximated. Clipping may have resulted in alterations to the primitives supplied. The data is thus not a substitute for the original. It can, however, be useful in producing an approximate copy of the stored data on another device.

The GDF file conversion utility can also be used to convert the file from the first format to the second.

Figure 14 shows the flow of events:

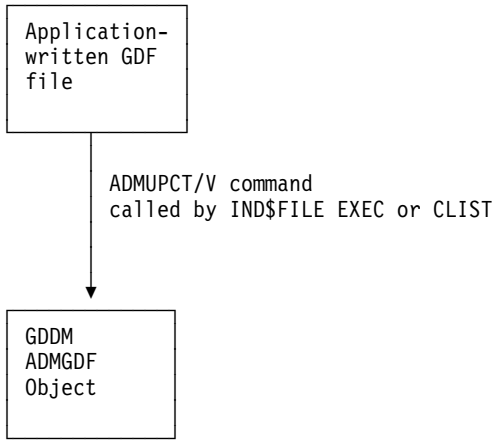


Figure 14. GDF file conversion – format 1 to format 2

To convert an application-written GDF file into an ADMGDF object the command is:

Under TSO:

```
ALLOC F(ADMPIF) DA('pif-dataset-name')SHR
ALLOC F(ADMGDF) DA('admgdf-dataset-name')SHR
CALL 'GDDM.SADMMOD(ADMUPCT)' 'pif-member /
       (PUT admgdf-member options'
```

Where `admgdf-dataset-name` must exist, and must be partitioned. The data set has the attributes LRECL(400), RECFM(FB), and BLKSIZE(400 * n).

If `pif-dataset-name` is sequential, `pifmember` should be omitted.

Under CMS:

```
ADMUPCV gdf-file-id (PUT admgdf-name options
```

Note: The `gdf-file-id` is a standard CMS file identifier.

The options are:

- {*NEWFile*|REPlace} – creates a new ADMGDF object or replaces an existing object of the same name

- {*FIXed*|FLOAT} – creates the ADMGDF file in fixed-point or floating-point format.

Format of GDF objects

The format of the data returned by GSGET is:

```
Comment order, with coordinate information
Begin Symbol-Set Mapping PSC
    Map Symbol-Set Identifier PSC
    ⋮
    ⋮
End Symbol-Set Mapping PSC
Begin Picture Prolog PSC
    Set Drawing Default PSC
    ⋮
    ⋮
End Picture Prolog PSC
Picture GDF (contains GDF orders)
```

The information in this appendix will help to interpret GDDM-created GDF orders that are to be used outside GDDM, or to create new GDF orders that can subsequently be used within GDDM.

For an example program that shows how to handle GDF data, see the *GDDM Base Application Programming Guide*.

Coordinates and aspect ratio

The coordinate values in the Picture Boundary PSC order and the initial Comment order are the upper and lower bounds of the picture space. In fixed-point GDF, these values are the values suitable for the device. In floating-point GDF, they are the continuation of the current window bounds to the picture space boundary. Note that unclipped floating-point GDF can contain orders with coordinates that are outside these limits.

To reshow a GDF picture, the window coordinates should be reset to the picture boundary values. The GDF picture can be reshown at any size. To preserve the aspect ratio of the picture, a GSPS call is required that is based on the coordinate values in the Picture Scale PSC order. This order defines the aspect ratio of the coordinates; the default aspect ratio is 1.

GDF orders: summary

Table 20 shows the GDF orders in the order of their code values and it provides useful information for those who need to interpret the orders.

Table 21 on page 284 shows the GDF orders in alphabetic order as they are described in this appendix. It provides useful information for those who need to write the orders.

Process specific control orders (PSC) are listed and described in more detail on pages 300 through 309.

Table 20 (Page 1 of 2). Summary of GDF orders in order of code values

Code	Name of GDF order	Mnemonic
X'01'	Comment	GCOMT
X'02'	Process Specific Control	GPSC
X'03'	Push And Set Character Box	GPSCC
X'04'	Segment Characteristics	GSGCH
X'07'	Call Segment	GSCALL
X'09'	Push And Set Pattern	GPSPT
X'0A'	Set Color	GSCOL
X'0C'	Set Foreground Color Mix	GSMX
X'0D'	Set Background Color Mix	GSBMX
X'10'	Set Text Alignment	GSTA
X'11'	Fractional Line Width	GSFLW
X'18'	Set Line Type	GSLT
X'19'	Set Line Width	GSLW
X'21'	Set Current Position	GSCP
X'22'	Set Arc Parameters	GSAP
X'23'	Push And Set Pick (Tag) Identifier	GPSPIK
X'24'	Set Model Transform	GSTM
X'26'	Set Extended Color	GSECOL
X'27'	Set Viewing Window	GSVIEW
X'28'	Set Pattern	GSPT
X'29'	Set Marker Type	GSMT
X'33'	Set Character Box	GSCC
X'34'	Set Character Angle	GSCA
X'35'	Set Character Shear	GSCH
X'36'	Set Character Box Spacing	GSCBS
X'37'	Set Marker Box	GSMC
X'38'	Set Character Set	GSCS
X'39'	Set Character Precision	GSCR
X'3A'	Set Character Direction	GSCD
X'3E'	Segment End Prolog	GEPROL
X'3F'	Pop Attribute	GPOP
X'41'	Marker Scale	GSMSC
X'43'	Set Pick (Tag) Identifier	GSPIK
X'4A'	Push And Set Color	GPSCOL
X'4C'	Push And Set Foreground Color Mix	GPSMX
X'4D'	Push And Set Background Color Mix	GPSBMX
X'50'	Push And Set Text Alignment	GPSTA
X'51'	Push And Set Fractional Line Width	GPSFLW
X'53'	Segment Position	GSSPOS
X'58'	Push And Set Line Type	GPSLT
X'59'	Push And Set Line Width	GPSLW
X'60'	End Area	GEAR
X'61'	Push And Set Current Position	GPSCP
X'62'	Push And Set Arc Parameters	GPSAP
X'64'	Push And Set Model Transform	GPSTM
X'66'	Push And Set Extended Color	GPSECOL
X'67'	Push And Set Viewing Window	GPVIEW
X'68'	Area	GBAR
X'69'	Push And Set Marker Type	GPSMT

Table 20 (Page 2 of 2). Summary of GDF orders in order of code values

Code	Name of GDF order	Mnemonic
X'70'	Segment Start	GBSEG
X'71'	Segment End	GESEG
X'72'	Segment Attribute	GISAT
X'73'	Segment Attribute Modify	GMSAT
X'74'	Push And Set Character Angle	GPSCA
X'75'	Push And Set Character Shear	GPSCH
X'76'	Push And Set Character Box Spacing	GPSCBS
X'77'	Push And Set Marker Box	GPSMC
X'78'	Push And Set Character Set	GPSCS
X'79'	Push And Set Character Precision	GPSCR
X'7A'	Push And Set Character Direction	GPSCD
X'81'	Line (at current position)	GCLINE
X'82'	Marker (at current position)	GCMRK
X'83'	Character String (at current position)	GCCHST
X'85'	Fillet (at current position)	GCFLT
X'86'	Arc (at current position)	GCARC
X'87'	Full Arc (at current position)	GCFARC
X'91'	Image Begin (at current position)	GCBIMG
X'92'	Image Data	GIMD
X'93'	Image End	GEIMG
X'A1'	Relative Line (at current position)	GCRLINE
X'C1'	Line	GLINE
X'C2'	Marker	GMRK
X'C3'	Character String	GCHST
X'C5'	Fillet	GFLT
X'C6'	Arc	GARC
X'C7'	Full Arc	GFARC
X'D1'	Image Begin	GBIMG
X'E1'	Relative Line	GRLINE

General structure

A GDF stream consists of a sequence of orders.

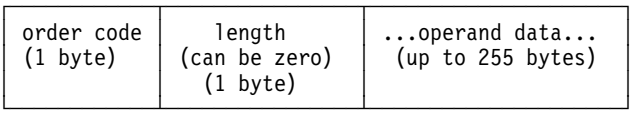
Each order is identified by a one-byte order code and contains one or more bytes of operand data.

Order formats

The order is represented in one of two formats depending on the length of the operand data.

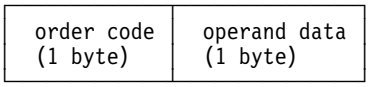
The first format applies to orders with up to 255 bytes of operand data. The second applies only to orders that have a single byte of operand data.

Normal format: In the normal format, there is a one-byte order code and a one-byte length field, followed by “length” bytes of operand data:



Therefore, the maximum possible length of a GDF order is 257 bytes.

Short format: If the first hexadecimal digit of an order code is less than 8, and the second hexadecimal digit is 8 or greater, the GDF is a short-format order. This consists of two bytes; the first one is the order code (as just defined), and the second one contains the operand data.



Padding

Orders can be followed by padding bytes X'00' so that the next order aligns on a convenient boundary.

Coordinate data: Many of the orders contain coordinate data or coordinate-related data. Coordinates may use different representations, either fixed or floating point.

When integer coordinates are used, the integers can be:

- Two-byte (halfword)
- One-byte.

These coordinate values are normal 15-bit or 7-bit numbers with sign. When negative, they are in twos-complement notation.

When floating-point coordinates are used, they are in standard four-byte (short floating-point) format. The type and length of coordinates must be specified on the GSPUT call. This is constant for the string.

Primitives

The following graphics primitives can be represented:

- Line (relative or absolute)
- Marker
- Character string
- Curved “fillet”
- Arc (circular, elliptical, or full)
- Image.

The orders have a close correspondence with many of the GDDM functions.

Table 21. Alphabetic summary of GDF order codes and usage

Order name	Primitives	Primitives at current position	Set	Push and set	Others
Arc	X'C6'	X'86'			
Arc Parameters			X'22'	X'62'	
Area					X'68'
Background Color Mix			X'0D'	X'4D'	
Call Segment					X'07'
Character Angle			X'34'	X'74'	
Character Box			X'33'	X'03'	
Character Box Spacing			X'36'	X'76'	
Character Direction			X'3A'	X'7A'	
Character Precision			X'39'	X'79'	
Character Set			X'38'	X'78'	
Character Shear			X'35'	X'75'	
Character String	X'C3'	X'83'			
Color			X'0A'	X'4A'	
Comment					X'01'
Current Position			X'21'	X'61'	
End Area					X'60'
Extended Color			X'26'	X'66'	
Fillet	X'C5'	X'85'			
Foreground Color Mix			X'0C'	X'4C'	
Fractional Line Width			X'11'	X'51'	
Full Arc	X'C7'	X'87'			
Image Begin	X'D1'	X'91'			
Image Data					X'92'
Image End					X'93'
Line	X'C1'	X'81'			
Line Type			X'18'	X'58'	
Line Width			X'19'	X'59'	
Marker	X'C2'	X'82'			
Marker Box			X'37'	X'77'	
Marker Scale			X'41'		
Marker Type			X'29'	X'69'	
Model Transform			X'24'	X'64'	
Pattern			X'28'	X'09'	
Pick (Tag) Identifier			X'43'	X'23'	
Pop					X'3F'
Process Specific Control					X'02'
Relative Line	X'E1'	X'A1'			
Segment Attribute					X'72'
Segment Attribute Modify					X'73'
Segment Characteristics					X'04'
Segment End					X'71'
Segment End Prolog					X'3E'
Segment Position					X'53'
Segment Start					X'70'
Set Viewing Window			X'27'		
Text Alignment			X'10'	X'50'	

Current position: The GDF order formats given below contain all relevant coordinates. However, for brevity the start position of the graphics primitive can be omitted. When omitted, current position is used in its place.

Current position is set by each of the orders. It is set to the end point of a line or arc and, except for character strings, the rule is the same as for the corresponding GDDM function.

The difference between an order that specifies the first coordinate pair, and an order that assumes the current position as the starting position is shown by the state of bit 1 of the order code, thus:

0 The order specifies the first coordinate pair.
1 The order assumes the current position as the starting position.

Attributes

GDDM provides two forms of attribute order; these are:

- Push And Set
- Set.

GDDM maintains a stack of attributes, which can be removed from the stack by using the Pop order.

A Push And Set attribute order puts the current value of the attribute being set onto the attribute stack and sets the value of the attribute to the value in the order. The Pop order unstacks the most recently pushed attribute on the stack and sets the popped attribute to the value restored from the stack.

The difference between a Set attribute and a Push And Set attribute order is generally shown by the state of bit 1 of the order code, thus:

0 The order is a Set attribute.
1 The order is a Push And Set attribute.

There are three exceptions to this rule; they are:

Order	Set	Push And Set
Pattern	X'28'	X'09'
Character Box	X'33'	X'03'
Pick (Tag) Identifier	X'43'	X'23'

Both the Set and the Push And Set orders correspond to GDDM attribute setting functions, according to the current attribute mode; see the description of the GSAM call. For example, the GPSLT order corresponds to the GSLT call when the attribute mode is 0 (preserve attributes).

As with the equivalent call statements, attribute setting orders change the current values of the attributes. An attribute setting applies to all subsequent primitives (to which it is relevant) until a new setting is made.

Attribute-setting orders appearing in a GDF string argument to GSPUT affect the current attribute settings after the call. The effects are not purely local to primitives within the string, but may affect subsequent primitives.

Full information on the effects of the orders is not given. For more explanation, see the corresponding call statement descriptions in Chapter 3, “The GDDM calls” on page 21.

GDF orders: full descriptions

This section describes the content and format of the GDF orders, which are presented in alphabetic order.

Format of tables: Where applicable, the Set and Push And Set forms of an order are included in the same table to reduce duplication. Only the hexadecimal order-code values are different for the two forms. The contents of the remaining fields are the same for the Set and Push And Set forms.

Some orders are available as “order”, or “order at current position”; only some of the following fields apply to “order at current position” and the table is annotated accordingly.

Where the Content of a field is given as “LEN”, it means that the order length is variable. Variable length orders occur when they include fields that contain coordinate and coordinate-related data. The length of such fields is indicated by a “*” in the Field length column. Variable length orders can also occur where a variable number of data items follows.

When the content of a field can take the form of various bit patterns with different meanings, they are represented in the form:
B'xxxxxxxx'
where

. Bit setting is not relevant.
1 Bit is set (1).
0 Bit is reset (0).
x Bit may be set to either value.

In general, “Reserved” bit positions should be reset (0).

Format of examples: The examples are given in hexadecimal, usually assuming halfword coordinates. Blanks in the hexadecimal strings are to aid readability; they have no other significance.

For detailed information about the GDDM calls mentioned, see Chapter 3, “The GDDM calls” on page 21.

Arc

This order constructs an arc starting at (x0,y0), passing through (x1,y1), and ending at point (x2,y2).

The intermediate point (x1,y1) should, for greatest accuracy, lie midway along the arc. (If it coincides with either end point the arc becomes undefined.) The initial point and the final point must not coincide.

The arc may be part of a circle or part of the ellipse defined by the previous “arc parameters” order. a length proportional to “a”; the axis parallel to the y axis has a length proportional to “b”.

The initial coordinate pair (x0,y0) may be omitted. Current position is then used as the starting point of the arc and the order code becomes X'86'.

The current position is set to point (x2,y2).

Fld len	Content	Meaning
1	X'C6' or X'86'	Arc order code (GARC) or Arc (at current position) (GCARC)
1	LEN	Length of following data
⋆	x0	x coordinate of start of arc (omitted for order X'86')
⋆	y0	y coordinate of start of arc (omitted for order X'86')
⋆	x1	x coordinate of intermediate point
⋆	y1	y coordinate of intermediate point
⋆	x2	x coordinate of end of arc
⋆	y2	y coordinate of end of arc

Arc parameters

This order determines the shape of subsequent arcs. The full parameters give a transformation that maps the unit circle to an ellipse of the required shape:

```
x' = Px + Ry
y' = Sx + Qy
```

A circle results if P=Q and R=S=0.

If P=a, Q=b, an ellipse results. The axis parallel to the x axis has a length proportional to “a”; the axis parallel to the y axis has a length proportional to “b.”

If R and S are nonzero, the ellipse is tilted. Usually, for an ellipse with major and minor axes proportional to “a” and “b”, tilted at angle “theta” to the x axis:

```
P = a.cos(theta)
Q = b.cos(theta)
R = -b.sin(theta)
S = a.sin(theta)
```

Fld len	Content	Meaning
1	***Set*** X'22'	Arc Parameters order code
	Push & set X'62'	Arc Parameters order code
1	LEN	Length of following data
⋆	P	x coordinate of major axis end
⋆	Q	y coordinate of minor axis end
⋆	R	x coordinate of minor axis end
⋆	S	y coordinate of major axis end

Area

The Area order approximates to the GSAREA and GSENDA calls.

Fld len	Content	Meaning
1	X'68'	Area order code
1	Flags:	
	B'1.....'	Start of an area
	B'0.....'	End of an area
	B'.1.....'	The boundary lines are to be drawn

Note: The End Area (see page 291) order has the same meaning as the Area order with the “end area” bit set. GDDM accepts both forms of order, but only generates the X'60' End Area order.

Examples:

```
68 80
C1 0A 00 00  05 00  05 05  00 05  00 00
68 00
```

Draws a rectangular area 5 units square. Boundary lines are not drawn.

```
68 C0
C1 0A 00 00  05 00  05 05  00 05  00 00
68 00
```

Draws the same area, but includes boundary lines.

Background color mix

The Background Color Mix order corresponds to the GSBMIX call.

Fld len	Content	Meaning
1	***Set*** X'0D'	Background Color Mix order
	Push & set X'4D'	Background Color Mix order code
1	Background color mix mode:	
	X'00'	Default
	X'01'	OR
	X'02'	Overpaint
	X'03'	Underpaint
	X'04'	Exclusive-OR (implemented as overpaint)
	X'05'	Leave alone

Call segment

The Call Segment order corresponds to the GSCALL call.

Fld len	Content	Meaning
1	X'07'	Call Segment order code
1	X'06'	Length of following data
2	X'0000'	Reserved
4	SEGID	Identifier of segment to be called

Character angle

The Character Angle order corresponds to the GSCA call. It controls the angle of subsequent character strings.

Fld len	Content	Meaning
1	***Set*** X'34'	Character Angle order code
	Push & set X'74'	Character Angle order code
1	LEN	Length of following data
⋆	Ax	x coordinate of a point that defines the angle of the text
⋆	Ay	y coordinate of the point

Note: Ax and Ay specify a relative vector that defines the angle of the baseline of the string. When the coordinate (x,y) is on the baseline, (x + Ax, y + Ay) is also on the baseline.

When both Ax and Ay are zero, the current character angle attribute is set to the drawing default value.

Character box

The Character Box order corresponds to the GSCB call. The order specifies the size of characters in following character strings.

For 1-byte or 2-byte integer coordinates, the order can optionally be extended to provide a fractional portion of the character box; see the FRACTWIDTH and FRACTDEPTH fields.

Fld len	Content	Meaning
1	***Set*** X'33'	Character Box order code
	Push & set X'03'	Character Box order code
1	LEN	Length of following data
⋆	CHARWIDTH	Width of character box
⋆	CHARHEIGHT	Height of character box
⋆	FRACTWIDTH	Fractional portion of character box width, specified as multiples of 1/65536 for 2-byte format or 1/256 for 1-byte format
⋆	FRACTDEPTH	Fractional portion of character box depth, specified as multiples of 1/65536 for 2-byte format or 1/256 for 1-byte format

Note: When either the fractional width or depth is to be specified, both must be included. The integer and fractional character widths (depths) together form a character width (depth) that defines the character box required.

Character box spacing

The Character Box Spacing order corresponds to the GSCBS call.

The order specifies the spacing of characters in following character strings.

Fld len	Content	Meaning
1	***Set*** X'36'	Character Box Spacing order
	Push & set X'76'	Character Box Spacing order code
1	LEN	Length of following data
1	Flags:	
	B'0.....'	Set char box spacing
	B'1.....'	Set default char box spacing
	B'.0000000'	Reserved – must be as shown
1	X'00'	Reserved
*	HSPACE	Horizontal character box spacing
*	VSPACE	Vertical character box spacing

Character direction

The Character Direction order corresponds to the GSCD call.

Fld len	Content	Meaning
1	Set X'3A'	Character Direction order code
	Push & set X'7A'	Character Direction order code
1	Character direction:	
	X'00'	Default
	X'01'	Left to right
	X'02'	Top to bottom
	X'03'	Right to left
	X'04'	Bottom to top

Note: The character direction gives the placement of each character relative to the previous one, either along or perpendicular to the baseline.

Character precision

The Character Precision order corresponds to the GSCM call.

Fld len	Content	Meaning
1	***Set*** X'39'	Character Precision order code
	Push & set X'79'	Character Precision order code
1	Character precision mode:	
	X'00'	Default
	X'01'	String precision
	X'02'	Character precision
	X'03'	Stroke precision
	Other	Not defined

Character set

The Character Set order corresponds to the GSCS call.

Fld len	Content	Meaning
1	***Set*** X'38'	Character Set order code
	Push & set X'78'	Character Set order code
1	Local identifier (LCID) for the character set:	
	X'00'	Default
	X'01'	APL
	X'41' – X'DF'	User-defined set
	X'F8'	Default DBCS
	Other	Not defined

Character shear

The Character Shear order corresponds to the GSCH call. It controls the shear of subsequent characters.

Fld len	Content	Meaning
1	***Set*** X'35'	Character Shear order code
	Push & set X'75'	Character Shear order code
1	LEN	Length of following data
⋆	Hx	Hx and Hy specify a relative vector that defines the angle at which characters are to be sheared.
⋆	Hy	y increment: see above

Notes:

1. Hx and Hy specify a vector that defines the angle of the upright strokes of a character relative to the baseline. If the lower left-hand corner of a character is placed at (0,0) and the character baseline lies along the x axis, the line from (0,0) to (Hx,Hy) gives the direction of upright strokes.
2. If both Ax and Ay are zero, the current shear attribute is set to the drawing default value.

Character string

The Character String order corresponds to the GSCHAR and GSCHAP calls.

Fld len	Content	Meaning
1	X'C3' or X'83'	Character String order code or Character String (at current position) order code
1	LEN	Length of following data
⋆	x0	x coordinate at which character string is to be placed (omitted for order X'83')
⋆	y0	y coordinate of character string (omitted for order X'83')
V	STRING	EBCDIC character code of each character in the string. All characters above and including X'40' are valid.

Note: The character string is placed at the indicated coordinate. The attributes of the string (for example, mode, size, angle) are taken from the current values.

If the character string has a length that is odd, the length field in the order contains an odd number. The order must be padded with padding characters to an even number of bytes.

The position (x0,y0) may be omitted, in which case the order code becomes X'83' and the string is placed at the current position.

Current position is not changed. (This is different from GSCHAR.)

Examples:

```
C3 08 0002 0003 C1C2C3C4
```

Draws the character string “ABCD” at coordinate (2,3).

```
83 03 C5C6C7
```

Draws the character string “EFG” at the current position.

Color

There are two orders approximating to the GSCOL call.

The Color order code takes this form:

Fld len	Content	Meaning
1	***Set*** X'0A'	Color order code
	Push & set X'4A'	Color order code
1	Color:	
	X'00'	Default
	X'01'	Blue
	X'02'	Red
	X'03'	Magenta (pink)
	X'04'	Green
	X'05'	Turquoise (cyan)
	X'06'	Yellow
	X'07'	Neutral: white on displays, black on hardcopy
	X'08'	Background: black on displays, white on hardcopy
	Other	Not defined

Note: The GSCOL call and the X'0A' Set Color order may be mapped to the X'26' Set Extended Color order as follows:

- Colors 0 through 8 are mapped to X'FF00' through X'FF08' in the Extended Color order on page 290.
- All other values map directly to a two-byte value.

The Extended Color order code takes this form:

Fld len	Content	Meaning
1	***Set*** X'26'	Extended Color order code
	Push & set X'66'	Extended Color order code
1	X'02'	Length of following data
2	Color:	
	X'0000' or X'FF00'	Default
	X'0001' or X'FF01'	Blue
	X'0002' or X'FF02'	Red
	X'0003' or X'FF03'	Magenta (pink)
	X'0004' or X'FF04'	Green
	X'0005' or X'FF05'	Turquoise (cyan)
	X'0006' or X'FF06'	Yellow
	X'0007'	White
	X'0008'	Black
	X'0009'	Dark blue
	X'000A'	Orange
	X'000B'	Purple
	X'000C'	Dark green
	X'000D'	Dark turquoise (cyan)
	X'000E'	Mustard
	X'000F'	Gray
	X'0010'	Brown
	X'FF07'	Neutral/multicolor (white on displays, black on hardcopy)
	X'FF08'	Background (black on displays, white on hardcopy)
	Other values	Not defined

Notes:

1. If a color value is outside the range of color values supported by a device, the color displayed is device-dependent (see GSCOL).
2. For color separation on family-4 devices, the color values depend on the loaded color table.
3. All subsequent primitives have the color given until this is reset.

Comment

The Comment order holds GDDM or application-program data within a GDF stream. Comments are stored in floating-point GDF.

The first GDF order returned by GSGET (and, by convention, in GDF files) contains the coordinate range and coordinate type in the following form. This convention is maintained but has been superseded by Process Specific Control (PSC) orders.

Fld len	Content	Meaning
1	X'01'	Comment order code
1	LEN	Length of following data
2	Coordinate type:	
	2	2-byte integer
	4	floating-point
⋆	xL	x lower boundary of picture space
⋆	xU	x upper boundary of picture space
⋆	yL	y lower boundary of picture space
⋆	yU	y upper boundary of picture space

It is recommended that Comment orders, created by an application program to contain application-specific information, should take the following form. (GDDM suggests the following convention but does not enforce it.)

Fld len	Content	Meaning
1	X'01'	Comment order code
1	LEN	Length of following data
2	X'0000'	Reserved
8	IDENT	Application identifier
N	DATA	User data

Current position

The Current Position orders approximate to the GSCP call.

Fld len	Content	Meaning
1	***Set*** X'21'	Current Position order code
	Push & set X'61'	Current Position order code
1	LEN	Length of following data
⋆	x	x coordinate of new current position
⋆	y	y coordinate of new current position

End area

The End Area order has the same meaning as the Area order (see page 286), with the “end area” bit set. GDDM accepts both forms of order, but only generates the X'60' End Area order.

Fld len	Content	Meaning
1	X'60'	End Area order code
1	LEN	Length of following data
LEN	X'00'	Reserved (must be all nulls)

Fillet

The Fillet order approximates to the GSPFLT call.

Fld len	Content	Meaning
1	X'C5'or	Fillet order code or
	X'85'	Fillet (at current position) order code
1	LEN	Length of following data
⋆	x0	x coordinate of line start (omitted for order X'85')
⋆	y0	y coordinate of line start (omitted for order X'85')
⋆	x1	x coordinate of first line end
⋆	y1	y coordinate of first line end
⋆	x2	x coordinate of second line end
⋆	y2	y coordinate of second line end
	⋮	⋮

Note: The order shown generates a single fillet. More coordinate pairs may be added to form a polyfillet. The points are joined in order by imaginary straight lines. A curve is then fitted to the lines as follows. The curve is tangential to the first line at its starting point and to the last line at its end point. If there are intermediate lines, the curve is tangential to these lines at their center points. In the special case when only two points are supplied, a straight line results.

The initial coordinate pair (x0,y0) may be omitted. The current position is then used as the starting point of the arc, and the order code becomes X'85'. The current position is set to the last point specified.

Examples:

```
C5 08 0002 0003  0004 0006
```

Draws a line from coordinate (2,3) to coordinate (4,6).

```
C5 0C 0000 0000  0004 0000  0004 0004
```

Draws a curve, beginning at coordinate (0,0) and tangential to the line from (0,0) to (4,0). Initially the curve is horizontal. The curve then takes an approximately circular path to meet the line from (4,0) to (4,4) at (4,4).

```
C5 08 00 00  04 00  04 08  00 08
```

assuming byte coordinates, draws two curves. The first is that in the previous example and the second completes an approximation to a semicircular arc.

Foreground color mix

The Foreground Color Mix order corresponds to the GSMIX call.

Fld len	Content	Meaning
1	***Set*** X'0C'	Foreground Color Mix order
	Push & set X'4C'	Foreground Color Mix order code
1	Foreground color mix mode:	
	X'00'	Default
	X'01'	Mix
	X'02'	Overpaint
	X'03'	Underpaint
	X'04'	Exclusive-OR
	X'05'	Leave alone
	Other	Not defined

Note: Mix mode controls how an inserted primitive affects the existing picture. In all modes, generated 0 bits leave the underlying features untouched; new 1 bits become whatever the current color is if that bit was previously of background color. The effect of a new 1 bit over an existing 1 bit depends on the particular mode: the old color (underpaint), the new color (over-paint), or a mixture (mix) may result.

Fractional line width

The Fractional Line Width order corresponds to the GSFLW call.

Fld len	Content	Meaning
1	***Set*** X'11'	Fractional Line Width order code
	Push & set X'51'	Fractional Line Width order code
1	X'02'	Length of following data
1	INTEGRAL LINE WIDTH	The integer portion of the line-width multiplier
1	FRACTIONAL LINE WIDTH	The fractional portion of the line-width multiplier, specified as multiples of 1/256

Note: The integral and fractional line widths together form a line-width multiplier that defines the line width required. For an explanation of the interpretation of the multiplier, see description the GSLW and GSFLW calls.

Full arc

The Full Arc order allows a complete circle or ellipse to be specified in one order. The size and shape of the circle or ellipse are determined by the Set Arc Parameters order; see page 286. Note that the Set Arc Parameters order sets the relative lengths of the major and minor axes for three-point arcs, but for the Full Arc order it sets the absolute size, in world coordinates, of a full circle or ellipse.

The coordinate pair may be omitted. The order code then becomes X'87', and the arc is drawn with its center at the current position. The current position is unchanged. (The X'87' version of the order draws the arc at the current position.)

The major and minor axes of the arc are defined as

`M.a  M.b`

where

`M` is the two-byte unsigned fractional fixed-point multiplier; the first eight bits are the integral part and the second eight bits are the fractional part.

`a` and `b` are the lengths of the major and minor axes obtained from the Arc Parameters order; see page 286.

A Full Arc order is allowed in an area definition and causes the area to be closed.

Fld len	Content	Meaning
1	X'C7' or	Full Arc (at given position) order code or
	X'87'	Full Arc (at current position) order code
1	LEN	Length of following data
*	x	x Coordinate value (omitted for order X'87')
*	y	y Coordinate value (omitted for order X'87')
2	M	Multiplier

Note: The arc is drawn with its center at point (x,y), which becomes the current position.

Image – begin

The Begin Image order, together with the Image Data and End Image orders approximate to the GSIMG and GSIMGS calls.

An image consists of a rectangular array of display points.

It is represented by a sequence of orders. The first is a Begin Image order and the last is an End Image order (see below). Between these delimiters, several Image Data orders may occur, giving the array of display points in the image.

The initial coordinate pair (x0,y0) can be omitted. The current position is then used to place the image data, and the order code becomes X'91'. The current position is not changed by a series of Image orders.

The size of the display point array and its representation are given by the Begin Image order. The fields IMAGEWIDTH and IMAGEDEPTH are optional and may be either both specified, or both omitted. When specified, the image is scaled to fill the area identified by the fields, using the rules defined in the GSIMGS call. When omitted, each display point is represented by one bit in the display point array.

Fld len	Content	Meaning
1	X'D1' or X'91'	Begin Image (at given position) order code or Begin Image (at current position) order code
1	LEN	Length of following data
⋆	x0	The x position at which the image is to be placed (omitted for order X'91')
⋆	y0	The y position at which the image is to be placed (omitted for order X'91')
2	FORMAT	The format of the image data. This field must have the value 0.
2	WIDTH	The width of the image in display points
2	DEPTH	The depth of the image in display points
⋆	IMAGEWIDTH	The desired width of the image in coordinate units
⋆	IMAGEDEPTH	The desired depth of the image in coordinate units

Image – data

For image FORMAT 0, each Image Data order contains the display points for one row of the display point array. Thus for an image with a DEPTH of N, there are N Image Data orders between the Begin and End Image orders. Each Image Data order contains data for WIDTH display points. Each display point is represented by a single bit. When the bit is one, the display point is "on"; when the bit is zero, the display point is "off".

Fld len	Content	Meaning
1	X'92'	Image Data order code
1	LEN	Length of following data
LEN	PIXELDATA	The display points of the image

Example:

```
91 06 0000 0009 0004
92 02 FF80
92 02 8080
92 02 8080
92 02 FF80
93 02 0000
```

Draws an image, whose size is nine display points wide by four deep, at the current position. The image consists of a small square of "on" display points (which appear in the current color) surrounding an "off" center.

Image – end

This order ends the construction of an image.

Fld len	Content	Meaning
1	X'93'	End Image order code
1	X'02'	Length of following data
2	X'0000'	Reserved

Line

The Line order approximates to the GSPLNE call.

Fld len	Content	Meaning
1	X'C1' or	Line order code or
	X'81'	Line (at current position) order code
1	LEN	Length of following data
⋆	x0	x coordinate of line start order code (omitted for order X'81')
⋆	y0	y coordinate of line start order code (omitted for order X'81')
⋆	x1	x coordinate of first line end
⋆	y1	y coordinate of first line end
	⋮	⋮

Note:

A line is drawn from the first coordinate given (x0,y0) to the second (x1,y1).

The order shown is for a single line, but usually any number of coordinates can be present. Consecutive coordinates in the order are joined by straight lines. The data length must be an even multiple of the coordinate length.

The initial coordinate pair (x0,y0) may be omitted. Current position is then used as the starting point of the first line and the order code becomes X'81'.

Current position is set to the last point specified.

Note that a line order with only an initial position is permitted. This serves only to move current position.

Examples

```
C1 08 0002 0003  0004 0006
```

Draws a line from coordinate (2,3) to coordinate (4,6).

```
C1 0C 0002 0003  0004 0006  0009 0009
```

Draws a line from coordinate (2,3) to coordinate (4,6) and a line from (4,6) to (9,9).

```
C1 06 02 03  04 06  09 09
```

Draws the same lines, but 1-byte coordinates are used.

```
C1 04 0002 0003
```

Draws no lines. However, current position is changed to the last coordinate (2,3).

```
81 04 0004 0006
```

Draws a line from current position to point (4,6). Thus, the pair of orders:

```
C1 04 0002 0003
81 04 0004 0006
```

has the same effect as the first:

```
C1 08 0002 0003  0004 0006.
```

Line type

The Line Type order corresponds to the GSLT call.

Fld len	Content	Meaning
1	***Set*** X'18'	Set Line Type order code
	Push & set X'58'	Set Line Type order code
1	Line type:	
	X'00'	Default
	X'01'	Dotted line
	X'02'	Short dashed line
	X'03'	Dash-dot line
	X'04'	Double-dotted line
	X'05'	Long-dashed line
	X'06'	Dash-double-dot line
	X'07'	Solid line
	X'08'	Invisible line
	Other	Not defined

Line width

The Line Width order corresponds to the GSLW call.

Fld len	Content	Meaning
1	***Set*** X'19'	Set Line Width order code
	Push & set X'59'	Set Line Width order code
1	LINEWIDTH	Value for line-width attribute

Marker

The Marker order approximates to the GSMRKS call.

Fld len	Content	Meaning
1	X'C2' or	Marker order code or
	X'82'	Marker (at current position) order code
1	LEN	Length of following data
⋆	x0	x coordinate of marker (omitted for order X'82')
⋆	y0	y coordinate of marker (omitted for order X'82')
	⋮	⋮

Note:

The order is shown for a single marker. More coordinate pairs may be added. The current marker is placed at each point specified.

The first (or only) coordinate pair may be omitted. The order code then becomes X'82', and a marker is placed at the current position in addition to any points specified.

The current position is set to the last coordinate specified, or, if none, is unchanged.

Examples:

```
C2 04 0002 0003
```

Draws the current marker at coordinate (2,3).

```
C2 0C 0002 0003  0004 0006  0009 0009
```

Draws markers at (2,3) (4,6) and (9,9).

```
82 00
```

Draws the current marker at current position.

Marker box

The Marker Box order specifies the size of the cell used for scaling vector markers.

Fld len	Content	Meaning
1	***Set*** X'37'	Marker Box order code
	Push & set X'77'	Marker Box order code
1	LEN	Length of following data
⋆	MARKER-WIDTH	Width of marker cell
⋆	MARKER-HEIGHT	Height of marker cell

Marker scale

The Marker Scale order approximates to the GSMSC call. It sets the scale of the marker box with respect to the default marker box.

Fld len	Content	Meaning
1	X'41'	Marker Scale order code
1	LEN	Length of following data
4	SCALE	Scale of marker box with respect to the default marker box in floating-point format

Marker type

The Marker Type order corresponds to the GSMS call.

Fld len	Content	Meaning
1	***Set*** X'29'	Marker Type order code
	Push & set X'69'	Marker Type order code
1	Marker type:	
	X'00'	Default
	X'01'	Cross
	X'02'	Plus
	X'03'	Diamond
	X'04'	Square
	X'05'	6-point star
	X'06'	8-point star
	X'07'	Filled diamond
	X'08'	Filled square
	X'09'	Dot
	X'0A'	Small circle
	X'0B' through X'40'	Not defined
	X'41' through X'EF'	User defined

Note: The marker number determines which symbol is displayed by the marker primitive.

Model transform

The Model Transform order is a transformation matrix.

Fld len	Content	Meaning
1	***Set*** X'24'	Model Transform order code
	Push & set X'64'	Model Transform order code
1	LEN	Length of following data
1	X'00'	Reserved
1	Flags:	
	B'000000..'	Reserved
	B'......00'	Replace
	B'......01'	Postmultiply
	B'......10'	Premultiply (equivalent to GSSTFM preemptive)
	Other bit patterns	Not supported.
2	MASK	Load Mask Values
N	MATRIX	Transformation Matrix

Note: A segment transformation is defined by a matrix

$$M = \begin{bmatrix} M11 & M12 & M13 & M14 \\ M21 & M22 & M23 & M24 \\ M31 & M32 & M33 & M34 \\ M41 & M42 & M43 & M44 \end{bmatrix}$$

which is applied to primitives p to give p' as follows:

$(x',y',z',1) = (x,y,z,1).M$

The GDF order defines the matrix elements in the order

M11,M12,...,M14,M21,...,M44

This differs from the GSSTFM call, which specifies the matrix elements in the order

M11,M21,M31,M12,...,M33

The MASK field identifies those elements of the transformation defined by the MATRIX field. The bits within MASK correspond to, in order, elements

M11,M12,...,M14,M21,...M44

of the transformation. The values provided in MATRIX correspond, in order, to those elements of the transformation identified by bits set to 1 within MASK. All uninitialized values within the transformation matrix are set from the identity transformation.

Only elements M11, M12, M21, M22, M41, and M42 are processed by GDDM. All other elements must be zero or one (as in the identity matrix).

The transformation elements may be specified in one-byte, two-byte, or four-byte form, corresponding to the data type GDF coordinates.

The fixed-point representation of the matrix elements is: M41, M42 are twos complement numbers (8-bit or 16-bit). Elements M11, M12, M21, and M22 are twos complement numbers in the following form:

`SB.bb bbbb (bbbb bbbb)`

where S is the sign bit (1 = negative), B is the integer bit, and b the fractional bits (6 or 14 of them).

Pattern

The Pattern order corresponds to the GSPAT call.

Fld len	Content	Meaning
1	***Set*** X'28'	Pattern order code row.
	Push & set X'09'	Pattern order code
1	Pattern type:	
	X'00'	Default
	X'01' through X'08'	Decreasing density
	X'09'	Vertical lines
	X'0A'	Horizontal lines
	X'0B'	Diagonal lines 1 (bottom left to top right)
	X'0C'	Diagonal lines 2 (bottom left to top right)
	X'0D'	Diagonal lines 1 (top left to bottom right)
	X'0E'	Diagonal lines 2 (top left to bottom right)
	X'0F'	No shading
	X'10'	Solid shading
	X'11' through X'40'	Not defined
	X'41' through X'FE'	User-defined

Notes:

1. See the description of the GSPAT call.
2. This attribute determines which pattern (either built-in or defined through the GSLSS call) is to be used to shade the interior of subsequent areas.

Pick (tag) identifier

The Pick order corresponds to the GSTAG call.

A value of X'00000000'is considered a “null” value. Any output primitive that is given a null tag does not take part in correlation.

Fld len	Content	Meaning
1	***Set*** X'43'	Pick (Tag) Identifier order code
	Push & set X'23'	Pick (Tag) Identifier order code
1	X'04'	Length of following data
4	TAG	Tag value

Pop

The Pop order pops the top attribute of the current attribute stack and sets the popped attribute to the restored value.

The order is valid outside segments but the results can be unpredictable.

Fld len	Content	Meaning
1	X'3F'	Pop order code
1	X'00'	Reserved

Process specific control

GDDM uses the Process Specific Control (PSC) order to store picture prolog and symbol-set information in a GDF object. Because it does not add any graphics data to the picture, the order is ignored by the GSPUT call. For information on the different PSC orders, see pages 301 to 309.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'xx'	Process Identifier
1	X'xx'	Function Identifier
LEN –2	DATA	Process-specific Controls

Note: GSGET returns two types of PSCs:

- X'01' – for symbol-set names
- X'02' – for the picture prolog, including the defaults.

Relative line

The Relative Line order defines one or more straight lines. The end point of each line is given as a one-byte signed offset from the start of the line. Note that the offsets are always one-byte fixed, even in the floating-point form.

Order code X'A1' omits the current position, and draws lines from the current position.

Fld len	Content	Meaning
1	X'E1' or	Relative Line order code or
	X'A1'	Relative Line (at current position) order code
1	LEN	Length of following data
*	x0	x coordinate of line start (omitted for order X'A1')
*	y0	y coordinate of line start (omitted for order X'A1')
1	x1	x1 coordinate of first line end point
1	y1	y1 coordinate of first line end point
	⋮	⋮

Notes:

1. The current position is updated to the last point specified.
2. The data length must be an even multiple of the coordinate length.

Segment attribute

The Segment Attribute order approximates to the GSSATI call. It sets the attributes to be assigned to subsequently generated segments.

Fld len	Content	Meaning
1	X'72'	Segment Attribute order code
1	X'02'	Length of following data
1	The attribute to be set for subsequent segments:	
	X'01'	Detectability
	X'02'	Visibility
	X'03'	Highlighting
	X'04'	Transformability
	X'05'	Reserved
	X'06'	Chained
1	The value to be assigned to the specified attribute:	
	X'00'	Not detectable, invisible, not highlighted, or unchained
	X'01'	Detectable, visible, highlighted, nontransformable, or chained
	X'02'	Transformable

Segment attribute modify

The Segment Attribute Modify order approximates to the GSSATS call. It modifies the attributes that are currently assigned to a segment.

Fld len	Content	Meaning
1	X'73'	Segment Attribute Modify order code
1	X'06'	Length of following data
1	The attribute to be modified:	
	X'01'	Detectability
	X'02'	Visibility
	X'03'	Highlighting
	X'04'	Transformability
	X'05'	Reserved
	X'06'	Chained
1	The value to be assigned to the specified attribute:	
	X'00'	Not detectable, invisible, not highlighted, or unchained
	X'01'	Detectable, visible, highlighted, nontransformable, or chained
	X'02'	Transformable
4	IDENTIFIER	The identifier of the segment for which the attributes are to be modified.

Segment characteristics

The Segment Characteristics order adds more attributes to a segment. It is valid only within the prolog of a segment.

The general format is as follows:

Fld len	Content	Meaning
1	X'04'	Segment Characteristics order code
1	LEN	Length of following data
1	CHID	Identifier code for characteristics
LEN –1	DATA	Data

Notes:

1. GDDM sets X'80' in the CHID field. All other values are reserved. Values above X'80' are allocated to applications other than GDDM.
2. GDDM preserves all Segment Characteristics orders with values of other than X'80' in the CHID field. The use of the Segment Characteristics order with a CHID value of X'80' is defined below.

The order can be used to provide information that corresponds to the GSSORG call, as follows:

Fld len	Content	Meaning
1	X'04'	Segment Characteristics order code
1	LEN	Length of following data
1	X'80'	Identifier for GDDM
1	X'00'	Reserved
1	X'04'	Segment origin
1	LEN–4	Length of coordinate data
⋆	x0	x coordinate origin
⋆	y0	y coordinate origin

Segment end

This order corresponds to the GSSCLS call.

Fld len	Content	Meaning
1	X'71'	End Segment order code
1	X'00'	No data

Segment end prolog

The Segment End Prolog order shows the end of the prolog section of each segment. See also page 299.

Fld len	Content	Meaning
1	X'3E'	Segment End Prolog order code
1	X'00'	Reserved

Segment position

The Segment Position order approximates to the GSSPOS call. It sets the position of a transformable segment.

Fld len	Content	Meaning
1	X'53'	Segment Position order code
1	LEN	Length of following data
⋆	X0	x coordinate of the segment position
⋆	Y0	y coordinate of the segment position
4	IDENTIFIER	The identifier of the transformable segment that is to be positioned.

Segment start

This order corresponds to the GSSEG call. In the short form, the order is truncated immediately after the SEGMENT-ID field. In this case, all segment attributes are taken from the current initial segment attributes as set by the Segment Attribute order (see page 297), or by the GSSATI call.

Segment attribute information can be extended by using a segment prolog. The presence of a segment prolog is shown by a flag bit in the Segment Start order.

Fld len	Content	Meaning
1	X'70'	Segment Start order code
1	X'0C' or X'04'	Length of following data
4	SEGMENT-ID	The identifier to be given to the following segment, or 0 if unnamed. A four-byte (fullword) positive or zero integer (as in GSSEG).
2	Flags (omitted in short form):	
	B'0....... ....'	Visible
	B'1....... ....'	Invisible
	B'.1...... ....'	Reserved
	B'..0..... ....'	Nondetectable
	B'..1..... ....'	Detectable
	B'...1.... ....'	Reserved
	B'....0... ....'	No highlighting
	B'....1... ....'	Highlighting
	B'.....100 ....'	Reserved
	B'........ 0.......'	Chained
	B'........ 1.......'	Nonchained
	B'........ .00.....'	Reserved
	B'........ ...0....'	No prolog
	B'........ ...1....'	Prolog
	B'........ ....0...'	Nontransformable
	B'........ ....1...'	Transformable
	B'........ .....000'	Reserved
2	L2	Length of segment (see Note) (omitted in short form)
4	X'00000000'	Reserved (omitted in short form)

Notes:

1. GDDM returns the length of a fixed-point GDF segment in the Segment Start order retrieved using GSGET. The length of segment is ignored on GSPUT; segments must be closed by an explicit Segment End order. When the length of a segment cannot be represented as a 2-byte unsigned number, a length of zero is set.
2. The segment attributes in the Segment Start order override the initial segment attributes that are in effect at the time the segment is created. The segment attributes can be altered by GSSATS in the usual way.
3. Within the prolog of a segment, only the following orders are valid:
   - A no-operation (X'00')
   - Comment (X'01')
   - Process Specific Control (X'02')
   - Segment Characteristics (X'04')
   - Pop (X'3F')
   - Marker Scale (X'41')
   - The attribute orders shown below:

Arc Parameters	(X'22' or X'62')
Character Angle	(X'34' or X'74')
Character Box	(X'03' or X'33')
Character-Box Spacing	(X'36' or X'76')
Character Direction	(X'3A' or X'7A')
Character Precision	(X'39' or X'79')
Character Set	(X'38' or X'78')
Character Shear	(X'35' or X'75')
Color and Extended Color	(X'0A', X'4A', X'26', or X'66')
Fractional Line Width	(X'11' or X'51')
Line Type	(X'18' or X'58')
Line Width	(X'19' or X'59')
Marker Box	(X'37' or X'77')
Marker Type	(X'29' or X'69')
Foreground Color Mix	(X'0C' or X'4C')
Model Transform	(X'24' or X'64')
Pattern	(X'28' or X'09')
Pick (Tag) Identifier	(X'43' or X'23').
Segment Viewing Window	(X'27')
Text Alignment	(X'10' or X'50').

Primitive attributes in the segment prolog are treated as being ordinary primitive attributes. GDDM does not create any primitive attributes apart from the transform in the segment prolog. For upward compatibility of GDF, it is advisable not to place primitive attribute orders (other than the Model Transform order) within the segment prolog.

Segment viewing window

This order corresponds to the GSSVL call.

Fld len	Content	Meaning
1	***Set*** X'27'	Set Segment Viewing Window
	Push & set X'67'	Set Segment Viewing Window
1	LEN	Length of following data
1	X'00'	Reserved
1	Mask:	
	B'xx.....'	Reserved
	B'..1.....'	x left limit included in list of WW values
	B'...1....'	x right limit included in list of WW values
	B'....1...'	y bottom limit included in list of WW values
	B'.....1..'	y top limit included in list of WW values
	B'......1.'	z near limit included in list of WW values
	B'.......1'	z far limit included in list of WW values
0– 6★	WW	Window values

Text alignment

This order corresponds to the GSTA call.

Fld len	Content	Meaning
1	***Set*** X'10'	Set Text Alignment
	Push & set X'50'	Set Text Alignment
1	X'02'	Length of following data
1	Horizontal text alignment:	
	X'00'	Default
	X'01'	Normal
	X'02'	Left
	X'03'	Center
	X'04'	Right
	X'FF'	Standard
1	Vertical Text Alignment:	
	X'00'	Default
	X'01'	Normal
	X'02'	Top
	X'03'	Cap
	X'04'	Half
	X'05'	Base
	X'06'	Bottom
	X'FF'	Standard

Process specific control orders (PSC)

The Process specific control orders (order code X'02') fall into two sub-classes:

- Symbol-set orders. These precede all other orders except the Comment order. The third byte of the Symbol-set orders is always X'01'.
- Picture prolog orders, including default PSC orders. These follow the Symbol-set orders (if present). Default process specific control orders are only valid within the picture prolog; they always follow a Begin picture prolog order and are terminated by an End picture prolog order. The third byte of the Picture prolog orders is always X'02'.

Symbol-set process specific control orders: These orders precede all other orders except the Comment order. The third byte of the Symbol-set process orders is always X'01'. The fourth byte identifies the individual orders. They are summarized in numerical order of the fourth byte in Table 22 on page 301. Symbol-set PSCs are listed in alphabetic order of order name on pages 301 through 302.

Table 22. Numeric list of Symbol-set process specific control orders.

Order name	Fourth byte
Begin Symbol-set Mapping	X'7E'
Map Symbol-set Identifier	X'40'
End Symbol-set Mapping	X'7F'

Picture prolog process specific control orders: These include the Default PSC orders. These orders follow the Symbol-set orders (if present). The third byte of the Picture prolog process specific control orders is always X'02'. The fourth byte identifies the individual orders. They are summarized in numerical order of the fourth byte in Table 23. Picture prolog PSCs are listed in alphabetic order of name on pages 302 through 309.

Table 23. Numeric list of Picture prolog process specific control orders.

Order name	Fourth byte
Set Picture Origin	X'01'
Set Default Foreground Color Mix	X'0C'
Set Default Background Color Mix	X'0D'
Set Default Coordinate Type	X'0E'
Set Default Text Alignment	X'10'
Set Default Fractional Line Width	X'11'
Set Default Line Type	X'18'
Set Default Picture Scale	X'20'
Set Default Arc Parameters	X'22'
Set Default Extended Color	X'26'
Set Default Viewing Window	X'27'
Set Default Pattern	X'28'
Set Default Marker Symbol	X'29'
Set Picture Boundary	X'32'
Set Default Character Box	X'33'
Set Default Character Angle	X'34'
Set Default Character Shear	X'35'
Set Default Character-Box Spacing	X'36'
Set Default Marker Box	X'37'
Set Default Character Set	X'38'
Set Default Character Precision	X'39'
Set Default Character Direction	X'3A'
Set Default Pick Identifier	X'43'
Begin Picture Prolog	X'7E'
End Picture Prolog	X'7F'

Symbol-set PSC orders

The symbol-set PSCs contain the names and types of the symbol sets that are currently loaded. When an IBM 4250 printer is the primary device, the code page is returned, as well as the symbol-set name and type.

The symbol-set types and names are recorded in the Map Symbol-Set Identifier PSC. There is one control for each symbol set. All the controls are bracketed between a Begin Symbol-Set Mapping and End Symbol-Set Mapping PSC, as defined below.

GSGET returns one group of symbol-set mapping information.

Begin symbol-set mapping

The Begin Symbol-Set Mapping PSC precedes the picture prolog and the segments.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'02'	Length of following data
1	X'01'	GDDM Symbol-Set Process Identifier
1	X'7E'	Begin Symbol-Set Mapping

Map symbol-set identifier

Several Map Symbol-Set Identifier PSC orders follow the Begin Symbol-Set Mapping PSC order.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'0D' or X'15'	Length of following data
1	X'01'	GDDM Symbol-Set Process Identifier
1	X'40'	Map symbol-set ID to name and type
1	Type of symbol set:	
	X'01'	Image symbol set
	X'02'	Vector symbol set
	X'03'	Shading pattern
	X'04'	Marker (image) symbol set
	X'05'	4250 printer image symbol set
	X'06'	Marker (vector) symbol set
	X'07'	Pattern (vector) symbol set
	X'08'	DBCS image symbol set
	X'09'	DBCS vector symbol set
1	NUMBER	Number of symbol-set definition. Always returned as zero by GSGET.
1	ID	Symbol-set identifier (LCID)
8	SS-NAME	Symbol-Set Name
8	CP-NAME	Code Page Name (optional)

End symbol-set mapping

The End Symbol-Set Mapping PSC order appears at the end of the Map Symbol-Set Identifier PSC orders. Within the symbol-set mapping structure, any orders other than PSCs, comments, and no operations cause an implicit end to symbol-set mapping. A warning message is issued.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'02'	Length of following data
1	X'01'	GDDM Symbol-Set Process Identifier
1	X'7F'	End Symbol-Set Mapping

Picture prolog

GDDM uses PSCs to define the coordinate type, the extent of GDF, and the default attribute values. They are returned in the GDF after the map symbol-set PSC orders (if these are present).

All the Picture Prolog PSC orders, including the Default PSC orders, are bracketed between a Begin Picture Prolog PSC order and an End Picture Prolog PSC order.

Begin picture prolog

The Begin Picture Prolog PSC order precedes the Picture Prolog PSC order. Only one picture prolog is returned by the GSGET call.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'02'	Length of following data
1	X'02'	GDDM Picture Prolog Identifier
1	X'7E'	Begin Picture Prolog

End picture prolog

The End Picture Prolog PSC follows the Picture Prolog PSC orders.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'02'	Length of following data
1	X'02'	GDDM Picture Prolog Identifier
1	X'7F'	End Picture Prolog

Set default arc parameters

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'22'	Set Default Arc Parameters
1	LEN–3	Length of parameter data
⋆	P	P parameter of the transform
⋆	Q	Q parameter of the transform
⋆	R	R parameter of the transform
⋆	S	S parameter of the transform

Set default background mix

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'0D'	Set Default Background Mix
1	Default background color mix-mode attribute:	
	X'00'	Standard default
	X'02'	Opaque
	X'05'	Transparent

Set default character angle

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'34'	Set Default Character Angle
1	LEN–3	Length of coordinate data
*	AX	x coordinate of a point that defines the angle of the string
*	AY	y coordinate of a point that defines the angle of the string

Note: AX and AY specify a relative vector that defines the angle of the baseline of the string.

Set default character box

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'33'	Set Default Character Box
1	LEN–3	Length of width and height data
*	CHARWIDTH	Width of character box
*	CHARHEIGHT	Height of character box

Set default character-box spacing

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'09'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'36'	Set Default Character-Box Spacing
1	X'06'	Length of remaining data
1	Flags:	:
	B'0.....'	Set character-box spacing
	B'1.....'	Set default character-box spacing
	B'.0000000'	Reserved
1	X'00'	Reserved
2	HSPACE	Horizontal character-box spacing
2	VSPACE	Vertical character-box spacing

Set default character direction

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'3A'	Set Default Character Direc-tion
1	Default character direction:	
	X'00'	Standard default
	X'01'	Left to right
	X'02'	Top to bottom
	X'03'	Right to left
	X'04'	Bottom to top
	Other	Reserved

Set default character precision

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'39'	Set Default Character Precision
1	Default character precision mode:	
	X'00'	Standard default
	X'01'	String precision
	X'02'	Character precision
	X'03'	Stroke precision
	Other	Reserved

Set default character set

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'38'	Set Default Character Set
1	Local identifier (LCID) for the default character set:	
	X'00'	Standard default
	X'01' through X'FF'	Specified symbol set

Set default character shear

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'35'	Set Default Character Shear
1	LEN–3	Length of vector data
⋆	HX	HX and HY specify a relative vector that defines the angle at which characters are to sheared
⋆	HY	y increment: see above

Set default coordinate type

The Set Default Coordinate Type PSC defines the coordinate type of the primitive coordinates in the segments that follow. The default coordinate type is 2-byte fixed point.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	GDDM Picture Prolog Identifier
1	X'0E'	Set Default Coordinate Type
1	Default coordinate type:	
	B'0.......'	2-byte fixed-point number
	B'1.......'	4-byte floating-point number
	B'.0000000'	Reserved

Set default extended color

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'05'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'26'	Set Default Extended Color
1	X'02'	Length of following data
2	Default color:	
	X'0000' or X'FF00'	Default
	X'0001' or X'FF01'	Blue
	X'0002' or X'FF02'	Red
	X'0003' or X'FF03'	Magenta (pink)
	X'0004' or X'FF04'	Green
	X'0005' or X'FF05'	Turquoise (cyan)
	X'0006' or X'FF06'	Yellow
	X'0007'	White
	X'0008'	Black
	X'0009'	Dark blue
	X'000A'	Orange
	X'000B'	Purple
	X'000C'	Dark green
	X'000D'	Dark turquoise (cyan)
	X'000E'	Mustard
	X'000F'	Gray
	X'0010'	Brown
	X'FF07'	Neutral/multicolor (white on displays, black on hardcopy)
	X'FF08'	Background (black on displays, white on hardcopy)
	Other values	Not defined.

Set default foreground mix

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'0C'	Set Default Foreground Mix
1	Default foreground color mix-mode:	
	X'00'	Standard default
	X'01'	OR (Mix)
	X'02'	Opaque
	X'03'	Underpaint
	X'04'	Exclusive-OR
	X'05'	Transparent

Set default fractional line width

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'04'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'11'	Set Default Fractional Line Width
1	INTEGRAL LINE WIDTH	The integer portion of the line-width multiplier
1	FRACTIONAL LINE WIDTH	The fractional portion of the line-width multiplier, specified as multiples of 1/256

Set default line type

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'18'	Set Default Line Type
1	Default line type:	
	X'00'	Standard default
	X'01'	Dotted
	X'02'	Short dashed
	X'03'	Dash-dot
	X'04'	Double dotted
	X'05'	Long dashed
	X'06'	Dash-double-dot
	X'07'	Solid
	X'08'	Invisible
	Other	Reserved

Set default marker box

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'37'	Set Default Marker Box
1	LEN–3	Length of width and height data
⋆	MARKER-WIDTH	Width of marker box
⋆	MARKER-HEIGHT	Height of marker box

Set default marker type

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'29'	Set Default Marker Symbol
1	Default marker type:	
	X'00'	Default
	X'01'	Cross
	X'02'	Plus
	X'03'	Diamond
	X'04'	Square
	X'05'	6-point star
	X'06'	8-point star
	X'07'	Filled diamond
	X'08'	Filled square
	X'09'	Dot
	X'0A'	Small circle
	X'0B' through X'40'	Not defined
	X'41' through X'EF'	User defined

Set default pattern

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'03'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'28'	Set Default Pattern
1	Default pattern type:	
	X'00'	Default
	X'01' through X'08'	Decreasing density
	X'09'	Vertical lines
	X'0A'	Horizontal lines
	X'0B'	Diagonal lines 1 (bottom left to top right)
	X'0C'	Diagonal lines 2 (bottom left to top right)
	X'0D'	Diagonal lines 1 (top left to bottom right)
	X'0E'	Diagonal lines 2 (top left to bottom right)
	X'0F'	No shading
	X'10'	Solid shading
	X'11' through X'40'	Not defined
	X'41' through X'FE'	User-defined

Set default pick identifier

This order is only valid within the picture prolog.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'07'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'43'	Set Default Pick Identifier
1	X'04'	Length of following data
4	PICKID	Pick Identifier

Set default picture scale

The format of the Set Default Picture Scale PSC is as follows:

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'0B'	Length of following data
1	X'02'	Common Picture Prolog
1	X'20'	Set Default Picture Scale order code
1	X'08'	Length of scaling factor data
4	x	x scaling factor (see text)
4	y	y scaling factor

Notes:

1. When the coordinate type is two-byte fixed, the first halfword encodes the integer part of the scaling factor and the second halfword encodes the fractional part.
2. The scaling factors specify the number of coordinate units per millimeter. The default scaling factor is 20 per millimeter. Zero and negative scaling factors are not valid.

Set default text alignment

The format of the Set Default Text Alignment PSC is as follows:

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	X'04'	Length of following data
1	X'02'	Picture Prolog Identifier
1	X'10'	Set Default Text Alignment
1	Horizontal text alignment.	
	X'00'	Default
	X'01'	Normal
	X'02'	Left
	X'03'	Center
	X'04'	Right
	X'FF'	Standard
1	Vertical Text Alignment.	
	X'00'	Default
	X'01'	Normal
	X'02'	Top
	X'03'	Cap
	X'04'	Half
	X'05'	Base
	X'06'	Bottom
	X'FF'	Standard

Set default viewing window

The Set Default Viewing Window PSC defines the picture space. For fixed-point GDF, this is in device coordinates. For floating-point GDF, it is in world coordinates. This PSC contains the same data as the Set Picture Boundary PSC and should be considered as defining a view of a picture.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Common Picture Prolog
1	X'27'	Set Default Viewing Window
1	LEN–3	Length of remaining data
1	X'00'	Reserved
1	Mask:	
	B'xx.....'	Reserved
	B'..1.....'	x left limit included in list of WW values
	B'...1....'	x right limit included in list of WW values
	B'....1...'	y bottom limit included in list of WW values
	B'.....1..'	y top limit included in list of WW values
	B'......1.'	z near limit included in list of WW values
	B'.......1'	z far limit included in list of WW values
0– 6★	WW	Window values

Note: The coordinates can be 2-byte fixed-point data or 4-byte floating-point data.

Set picture boundary

The Set Picture Boundary PSC defines the picture space. For fixed-point GDF, this is in device coordinates. For floating-point GDF, it is in world coordinates. This PSC contains the same data as the Set Default Viewing Window PSC and should be used for setting window coordinates when redisplaying a GDF picture.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	Common Picture Prolog
1	X'32'	Set Picture Boundary
1	LEN–3	Length of remaining data
1	X'00'	Reserved
1	Mask:	
	B'xx.....'	Reserved
	B'..1.....'	x left limit included in list of WW values
	B'...1....'	x right limit included in list of WW values
	B'....1...'	y bottom limit included in list of WW values
	B'.....1..'	y top limit included in list of WW values
	B'......1.'	z near limit included in list of WW values
	B'.......1'	z far limit included in list of WW values
0– 6★	WW	Window values

Note: The coordinates can be 2-byte fixed-point data or 4-byte floating-point data.

Set picture origin

The Set Picture Origin PSC defines the lower left-hand corner of the graphics field.

Fld len	Content	Meaning
1	X'02'	Process Specific Control order code
1	LEN	Length of following data
1	X'02'	GDDM Picture Prolog Identifier
1	X'01'	Set Picture Origin order code
1	LEN–3	Length of coordinate data
★	x0	x coordinate of picture origin
★	y0	y coordinate of picture origin

Note: The values returned in x0,y0 are the coordinates of the lower left-hand corner of the graphics field.

Chapter 11. Image file formats

Image data is entered (or "put") into images using the IMAPTS, IMAPT, and IMAPTE calls. The reverse process of retrieving image data from an image (or "get") is done by using the IMAGTS, IMAGT, and IMAGTE calls. If the image is self-defining, the "put" process is a transfer operation and so a projection can be used.

The "get" process is always a transfer operation.

Images can also be entered into and retrieved from the application program using the IMASAV and IMARST calls. These store complete images into, and retrieve complete images from, a database or GDDM object library.

For detailed information on all image calls, see Chapter 3, "The GDDM calls" on page 21.

Formats and compression types

Image data must have a valid combination of format and compression type. The following image data formats are allowed:

1 Unformatted
2 3193 data stream format
3 Page printer format (PPF).

The following image data compression types are allowed:

1 Uncompressed
2 MMR (IBM 8815)
3 IBM 4250
4 IBM 3800.

Only specific combinations of format and compression are allowed; these are indicated by an "X" in the following table:

Table 24. Valid combinations of format and compression

	Unformatted	**3193 DSF**	**PPF**
Uncompressed	X	X	
MMR (IBM 8815)	X	X	X
IBM 4250			X
IBM 3800			X

Notes:

1. MMR = modified-modified read format (8815 compatible)
2. 3193DSF = 3193 data stream format
3. PPF = page printer format.

3193 data stream and page printer formats

Self-defining data comprises a data stream containing a list of image objects. These are indicated in Figure 15 on page 312 by suitable mnemonics.

For entry of formatted data (IMAPT) into either of these formats, GDDM processes only the first image object in the data stream (from BIC to EIC for 3193 data stream, and from BIM to EIM for page printer format) and ignores all others as shown in Figure 15 on page 312.

For retrieval of formatted data (IMAGT), GDDM constructs an image object either in 3193DSF or in PPF.

For 3193DSF data, GDDM constructs BIC, ISP, IEP, ID, and EIC structured fields, but it does not return ILP.

For more information on the 3193 data stream format, see the *IBM 3193 Display Station Description* manual.

For PPF data, GDDM constructs BIM, IID, IRD, and EIM structured fields, but it does not return IOC – except for 120 ppi 3800 image data, which is returned as 240 ppi with a scale factor of 2 in the IOC.

For more information on the page printer format, see the

- *Print Management Facility: User Guide and Reference* manual
- *Composed Document Printing Facility: General Information* manual.

Unformatted data

Unformatted binary image data is defined as follows:

Compression Uncompressed data; 1 bit per pixel, 8 bits per byte.

Compressed data; as defined by the compression algorithm.

Padding Uncompressed data; the end of each row is padded to the byte boundary, if it does not fall on one.

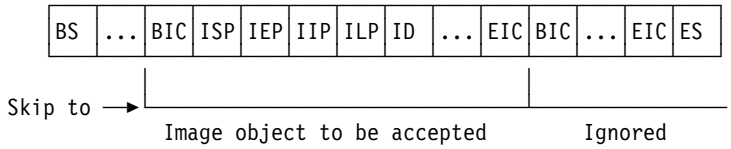
Compressed data; padding is defined by the compression algorithm.

Structure No headers, trailers, or imbedded control fields, other than those defined by the compression algorithm The pixels (and trailing pad values) occupy contiguous storage.

Row 0 (that is, the top of the picture) comes first, followed by the other rows in order.

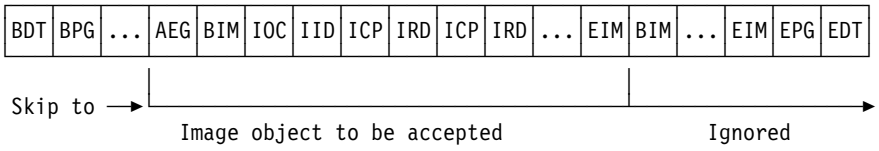
Within a row, the pixels with the lower index (that is, the left of the picture) come first.

3193DSF data stream



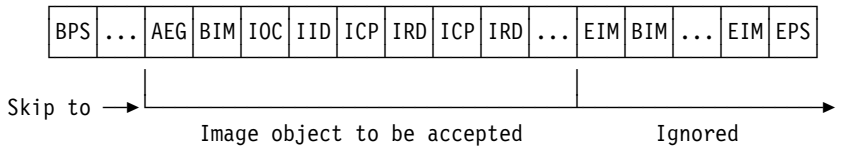

```
BS  :  Begin Segment
BIC :  Begin Image Content
ISP :  Image Size Parameter
IEP :  Image Encoding Parameter
IIP :  Image IDE Size Parameter
ILP :  Image LUT—ID Parameter
ID  :  Image Data
EIC :  End Image Content
ES  :  End Segment
```

PPF document




```
BDT :  Begin Document
BPG :  Begin Page
AEG :  Active Environment Group (optional)
BIM :  Begin Image
IOC :  Image Output Control (optional)
IID :  Image Input Descriptor
ICP :  Image Cell Position
IRD :  Image Raster Data
EIM :  End Image
EPG :  End Page
EDT :  End Document
```

PPF page segment




```
BPS :  Begin Page Segment
EPS :  End Page Segment
```

Figure 15. Accepted data streams (3193DSF and PPF)

3193DSF output data stream

BIC		(Begin Image Content)
ISP		(Image Size Parameter)
IEP		(Image Encoding Parameter)
ID	////////Image Data/////////	(Image Data)
⋮		
ID	//Image Data//	(Image Data)
EIC		(End Image Content)

PPF output data stream

BIM		(Begin Image)
IOC (32 bytes long)		(Image Output Control)
IID (44 bytes long)		(Image Input Descriptor)
ICP		(Image Cell Portion)
IRD	//////Image Data///////	(Image Raster Data)
ICP		(Image Cell Portion)
IRD	//Image Data//	(Image Raster Data)
⋮		
EIM		(End Image)

Figure 16. IMAGT data streams from GDDM

Chapter 12. Picture interchange format files

Application programs can transfer picture information between GDDM running in a host system and the 3270-PC/G or 3270-PC/GX work station as **picture interchange format (PIF) files** by using the GDDM-supplied GDF conversion utility (ADMUPCT/V) and the 3270-PC Graphics Control Program file transfer function.

A PIF file can also be generated on a workstation that uses GDDM-PCLK, through the User Control facility. (For details, refer to the *GDDM User's Guide*.)

As the PIF files on the host have different internal formats from those on a workstation, when files are transferred from one system to the other, they must also be converted to the relevant format before they can be used.

This conversion can be done at the same time as the transfer operation or as a separate operation.

The methods used to process PIF files vary according to the subsystem that the GDDM host session is running under. This chapter explains:

- Processing PIF files under TSO
- Processing PIF files under VM/CMS.

Note: GDDM does not support PIF files under CICS or IMS.

These topics are discussed for each subsystem:

- How PIF data relates to GDF data
- How to create PIF information under GDDM
- How to create PIF information at a workstation
- What a PIF file must contain if it is to be used under GDDM
- The structure of a PIF file
- Base PIF files.

The commands needed to convert and transfer PIF files are defined in the sections that follow; for more information, refer to the *GDDM User's Guide*.

Processing PIF files under TSO

The conversion operation

The GDF file-conversion utility: The conversion utility is distributed as a module called ADMUPCT. This utility converts GDDM ADMGDF objects into PIF files, or converts PIF files from the work station into ADMGDF objects.

The conversion utility also converts files (created by applications from GSGET calls and often named GDDM Version 1 Release 2 and 3 GDF files) into ADMGDF files; see “Saving GDF orders” on page 281.

Figure 17 on page 316 shows the flow of events.

When the IND$FILE CLIST executes, the ADMUPCT command is invoked to run the conversion utility if the ADMGDF option has been specified in a SEND or RECEIVE command.

The transfer operation

If the commands described in “Commands to use under TSO” on page 316 did not work, check that the IND$FILE CLIST is available at your installation, and that the library search order searches CLISTs before searching commands.

GDF data files must be converted into PIF files before they can be sent from GDDM to the workstation. There are four components in the procedure for transferring and converting the files:

- The SEND and RECEIVE commands that are issued at the workstation.

  These commands generate the IND$FILE command on the current host session, with the first parameter set to either PUT or GET.
- The IND$FILE CLIST that is issued at the host (GDDM).

  This CLIST controls the file transfer program and the conversion utility (see below).
- The IND$FILE file transfer command.
- The GDF conversion utility, which converts GDDM ADMGDF object files to PIF files, and *vice versa*.

Of these four components, the SEND and RECEIVE commands are described in the *GDDM User's Guide*. The other components are described in greater detail below.

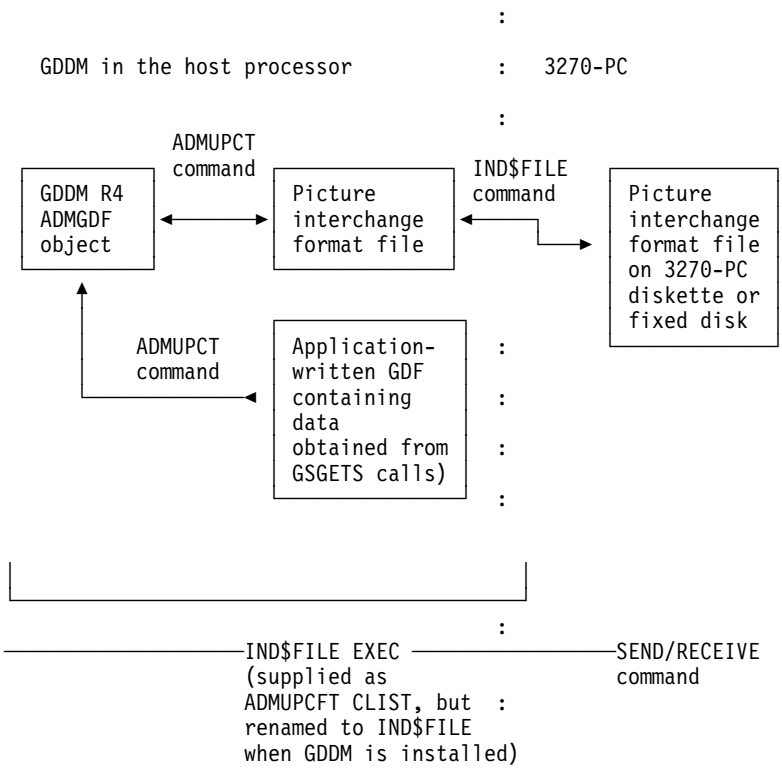


Figure 17. GDF file conversion procedure under TSO

The IND$FILE CLIST: These examples of the commands work with the IND$FILE CLIST that is supplied with GDDM.

Note: The CLIST is distributed with the name ADMUPCFT CLIST; it is recommended that it is renamed to IND$FILE CLIST by the systems programmer, after GDDM has been installed.

The IND$FILE CLIST invokes the IND$FILE file transfer program at the workstation.

Notes:

1. On heavily-loaded systems, it may be advisable to perform the file transfer separately from the conversion; for details, see “Commands to use under TSO”; for further information, refer to the *GDDM User's Guide*.
2. If the ADMUPCFT CLIST has been renamed to a name other than IND$FILE CLIST, the workstation SET command can be used to invoke the appropriate CLIST when a SEND or RECEIVE command is issued. For details of the SET command, refer to the *IBM Personal Computer Disk Operating System* manual.

The IND$FILE file transfer command: This is the command that transfers files between a workstation and the host processor.

Note: The file transfer command requires the File Transfer Program (licensed program number 5665-311), which runs on MVS/TSO.

Commands to use under TSO

To transfer a PIF file from the workstation to host

1. Ensure that the host session is ready to receive an operator command (that is, it is in a READY state).
2. From the PC session of the workstation enter:

   ```
   SEND picture.pif 'pif-dataset-name'
   ```

   The “pif-dataset-name” data set is automatically allocated if it does not already exist, and is created as a sequential data-set with fixed-length 80-byte records (unblocked). The “pif-dataset-name” if it already exists may be sequential or partitioned. If partitioned, the member-name must be included in “pif-dataset-name.”

To transfer a GDDM GDF picture from the host to the workstation

1. Enter the RECEIVE command from the workstation (in a PC session) as follows:

   ```
   RECEIVE picture.pif 'pif-dataset-name'
   ```

   This sends the file “pif.dataset-name” from the host (GDDM) system to the current workstation directory, converting it from the ADMGDF format to a PIF format.

   If the SEND or RECEIVE command was not successful, there may be some options not set up on your system, and you should consider this:

To convert a PIF file into a GDDM ADMGDF object

1. Use the commands:

```
ALLOC F(ADMPIF) DA('pif-dataset-name') SHR
ALLOC F(ADMGDF) DA('admgdf-dataset-name') SHR
CALL 'GDDM.SADMMOD(ADMUPCT)'
     'pif-member (PUT admgdf-member options'
```

   Where “admgdf-dataset-name” must exist, and must be partitioned. The data set has the attributes LRECL(400), RECFM(FB), and BLKSIZE(400 * n).

   If “pif-dataset-name” is sequential, pifmember should be omitted.

To convert a GDDM ADMGDF object into a PIF file

1. Use the commands:

```
ALLOC F(ADMPIF) DA('pif-dataset-name') SHR
ALLOC F(ADMGDF) DA('admgdf-dataset-name') SHR
CALL 'GDDM.SADMMOD(ADMUPCT)'
     'pif-member(GET admgdf-member,options'
```

   Where “admgdf-dataset-name” must exist, and must be partitioned. The data set has the attributes LRECL(400), RECFM(FB), and BLKSIZE(400 * n).

   If “pif-dataset-name” is sequential, pifmember should be omitted.

Notes:

1. The admpif-member-name is either a member name of the ADMPIF data set or blank if a sequential data set is being used.
2. The ADMPIF data set defaults are LRECL=400, RECFM=(FB), and BLKSIZE(400 * n).
3. The GDDM-supplied IND$FILE CLIST accepts the SEND and RECEIVE commands from the workstation, or it can run independently when invoked from GDDM in the host.

The format of a PIF file

The format of a PIF file under GDDM in the host processor depends on the subsystem being used; under TSO, it can be a sequential data set or a member of a partitioned data set.

In a 3270-PC/G or 3270-PC/GX work station, and devices supported by GDDM-PCLK, the PIF file is a standard PC-DOS 2.1 file.

In both the host and the workstation, the orders in a PIF file can span records.

Processing PIF files under VM/CMS

The conversion operation

The GDF file-conversion utility: The conversion utility is distributed as a module called ADMUPCV. This utility converts ADMGDF objects into PIF files, or converts PIF files from the work station into ADMGDF objects.

The conversion utility also converts files (created by applications from GSGET calls and often named GDDM Version 1 Release 2 and 3 GDF files) into ADMGDF files; see “Saving GDF orders” on page 281.

Figure 18 on page 318 shows the flow of events.

When the IND$FILE EXEC executes, the ADMUPCV command is invoked to run the conversion utility if the ADMGDF option has been specified in a SEND or RECEIVE command.

The transfer operation

If the commands described in “Commands to use under VM/CMS” on page 318 did not work, check that the IND$FILE EXEC is available at your installation.

GDF data files must be converted into PIF files before they can be sent from GDDM to the workstation. There are four components in the procedure for transferring and converting the files:

- The SEND and RECEIVE commands that are issued at the workstation.

  These commands generate the IND$FILE command on the current host session, with the first parameter set to either PUT or GET.
- The IND$FILE EXEC that is issued at the host (GDDM).

  This EXEC controls the file transfer program and the conversion utility (see below).
- The IND$FILE file transfer command.
- The GDF conversion utility, which converts GDDM ADMGDF object files to PIF files, and conversely.

Of these four components, the SEND and RECEIVE commands are described in the *GDDM User's Guide*. The other components are described in greater detail below.

The IND$FILE EXEC: These examples of the commands work with the IND$FILE EXEC that is supplied with GDDM.

Note: The EXEC is distributed with the name ADMUPCFV EXEC; it is recommended that it is renamed to IND$FILE EXEC by the systems programmer, after GDDM has been installed.

The IND$FILE EXEC invokes the IND$FILE file transfer program at the workstation.

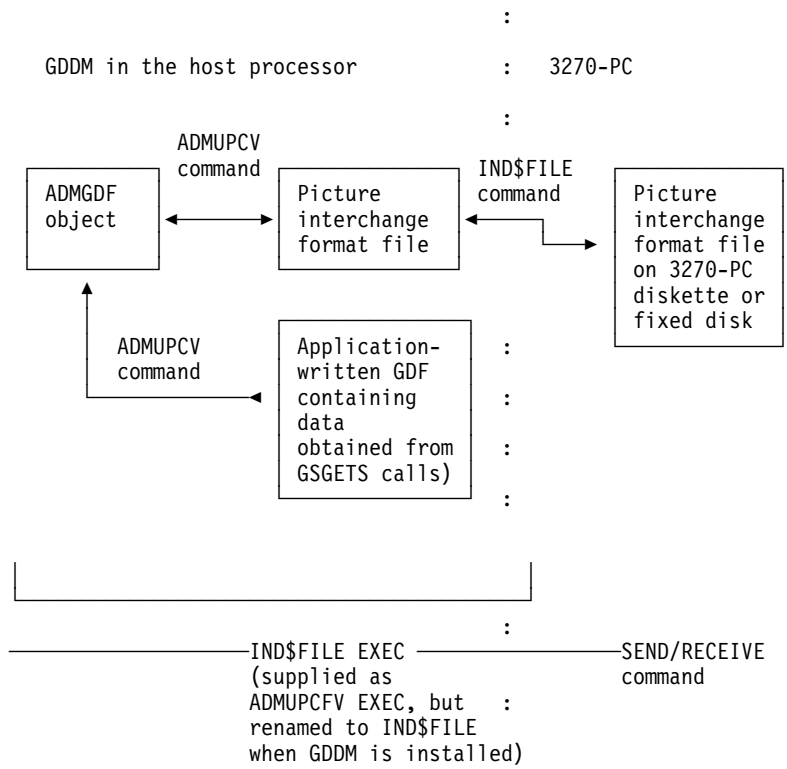


Figure 18. GDF file conversion procedure under VM/CMS

Notes:

1. On heavily-loaded systems, it may be advisable to perform the file transfer separately from the conversion; for details, see "Commands to use under VM/CMS" on page 318; for more information, refer to the *GDDM User's Guide*.
2. If the ADMUPCFV EXEC has been renamed to a name other than IND$FILE EXEC, the workstation SET command can be used to invoke the appropriate EXEC when a SEND or RECEIVE command is issued. For details of the SET command, refer to the *IBM Personal Computer Disk Operating System* manual.

The IND$FILE file transfer command: This is the command that transfers files between a workstation and the host processor.

Note: The file transfer command requires the File Transfer Program (licensed program number 5664-281 for VM/SP) which runs on VM/SP Release 3 or later.

Commands to use under VM/CMS

To transfer a PIF file from the workstation to host

1. Ensure that the host session is ready to receive an operator command (for example, ensure that the host session is not running the Interactive Chart Utility).
2. Ensure that the CMS default SET IMPEX ON is in operation.
3. Enter the SEND command from the workstation (in a PC session) as follows:

   ```
   SEND picture.PIF picture (ADMGDF
   ```

   This sends the file picture.PIF from the current workstation directory, converts it to GDDM format (because of the ADMGDF keyword), and stores the file as a GDDM ADMGDF picture in the host.

   If you want to transmit the file again unchanged (for back-up or transmission to another workstation), do not use the keyword option ADMGDF as this option may result in some details of the picture being lost.

To transfer a GDDM GDF picture from the host to the workstation

1. Ensure that the host session is ready to receive an operator command (for example, ensure that the host session is not running the Interactive Chart Utility).
2. Ensure that the CMS default SET IMPEX ON is in operation.

3. Enter the RECEIVE command from the workstation (in a PC session) as follows:

   ```
   RECEIVE picture.PIF picture (ADMGDF
   ```

   This sends the GDDM ADMGDF picture file from the host ADMGDF object library to the current workstation directory.

   If you want to transmit the file again unchanged (for back-up or transmission to another workstation), do not use the keyword option ADMGDF, as this option may result in some details of the picture being lost.

To convert a PIF file into a GDDM ADMGDF object

1. Use the command:

   ```
   ADMUPCV admpif-file-id (PUT admgdf-name options
   ```

   The options are:

   - {*NEWFile*|REPlace} – creates a new GDF object or replaces an existing object of the same name.
   - {*FIXed*|FLOAT} – creates the GDF object in fixed- or floating-point format.

To convert a GDDM ADMGDF object into a PIF file

1. Use the command:

   ```
   ADMUPCV admpif-file-id (GET admgdf-name options
   ```

   The options are:

   - {*NEWFile*|REPlace} – creates a new PIF file or replaces an existing file of the same name.
   - {*FIXed*|FLOAT} – creates the PIF file in fixed- or floating-point format. If the PIF file is to be sent to a workstation, this parameter must be specified as FIXed.
   - LRECL {*400*|n} – specifies the length of each record for fixed-length files, or the maximum record length for variable-length files. The value of n must be in the range 16 through 2000.
   - RECFM {*F*|V} – specifies the record format as fixed length or variable length.

Note: The admpif-file-id is a standard CMS file identifier.

The format of a PIF file

The format of a PIF file under GDDM in the host processor depends on the subsystem being used; under VM/CMS, it is a normal VM/CMS file, conventionally of filetype PIF.

In a 3270-PC/G or 3270-PC/GX work station, and devices supported by GDDM-PCLK, the PIF file is a standard PC-DOS 2.1 file.

In both the host and the workstation, the orders in a PIF file can span records.

Creating PIF data under GDDM

The graphics data in PIF files is essentially the same as that in fixed-point GDF files. Using GDDM's GSGETS call, see Chapter 3, "The GDDM calls" on page 21, with the options for returning fixed-point coordinate data with a picture prolog, produces PIF orders.

Creating PIF data at a workstation

There are two ways of creating PIF data at a workstation:

1. By capturing alphanumerics or alphanumerics and graphics data that is displayed on a monitor. This is done by:

   a. Pressing the Ws Ctrl key

   b. Pressing the Print or Print and Shift keys.

   This spools a file called INDPRTnn.PIF to the user's INDPRT directory for printing at the workstation.

2. By writing an application program to create and save alphanumerics or graphics data, or both of these.

If they are to be transferred to GDDM, the PIF files created at a workstation must contain only those drawing orders that are recognized as GDF orders; the GDF orders are listed and described in Chapter 10, "GDF order descriptions" on page 281. The GDF utility converts orders where possible and diagnoses any changes made.

Note: Spooling a GDDM picture locally causes structural information to be lost because GDDM optimizes the data stream for display. Therefore, if possible you should create your PIF files at the host rather than spooling them locally into PC disk storage and retrieving them from the workstation.

How PIF data relates to GDF data

The formats of data in PIF files and in files created by applications from the results of GSGET calls differ, in some respects, from those of Version 1 Release 4 GDF (ADMGDF) files created from GSSAVE calls. The conversion utility converts from one form to the other. The differences are:

- PIF files contain special control information as detailed below.
- Fixed-point GDF is, usually, a subset of PIF function. However, some GDF orders before Version 1 Release 4 are ignored by the workstations. The GDF utility makes the appropriate conversions. The orders are:

  X'11' Fractional Line Width
  X'41' Marker Scale
  X'53' Segment Position
  X'71' Segment End
  X'72' Segment Attribute
  X'73' Segment Attribute Modify.

The workstation treats all these orders as no operations.

- Fixed-point GDF End Area (X'6800') is treated as a Begin Area order by workstations. End Area should be shown using X'6000'.

For a full list of the drawing orders supported by the workstation, see the *IBM 3270 Personal Computer/G or /GX: Reference Information for Picture Interchange Format* manual. See also the *IBM 3270 Personal Computer/G or /GX: Supplementary Reference Information for Picture Interchange Format* manual.

Pictures created at the workstation for use under GDDM should contain only those GDF orders listed in Chapter 10, "GDF order descriptions" on page 281 and should adhere to the restrictions that GDDM places on their use.

The conversion utility removes or changes orders in the PIF file that are not accepted by GDDM. In particular, note that symbol-set definitions are removed by the GDF conversion utility. For example, if a chart that uses symbol sets is created under GDDM's Interactive Chart Utility (ICU), and is stored using the Print Spool function, GDDM may use different symbol sets when the chart is sent to GDDM and displayed at the host. This is because PIF files created in this way do not reference the original symbol sets and because the symbol-set definitions in the PIF file are discarded.

Base PIF

For GDDM Version 2, there is a subset of GDF orders known as Base PIF. **All** Base PIF files can be imported into GDDM.

Restrictions and considerations

To ensure that ADMGDF files convert to Base PIF so that they can be exported, the following must be borne in mind:

Creating files: Avoid any GDDM calls involved with:

- Multiple-connected areas; for example a ring
- Image data
- Image symbols
- Loaded marker and pattern sets
- Foreground color mixing other than overpaint.

The spool print function: The same restrictions listed above must be observed when the Spool Print function is used to produce a PIF file from a picture originally created by a GDDM application.

The GDDM sample program ADMUSP4: PIF files imported into GDDM cannot be edited directly by the GDDM sample program ADMUSP4; refer to the program described in the *GDDM Base Application Programming Guide*.

ADMUPCV and ADMUPCT utilities: When using these utilities to create PIF files, avoid generating files that have a floating-point format.

LCLMODE processing option: Ensure that the LCLMODE processing option is enabled. This ensures that the maximum amount of picture detail is present in a PIF file resulting from Spool Print. In the absence of local mode, GDDM optimizes the data stream (for example, an arc is expanded into a series of line segments), such that, at the original scale, a picture is displayed correctly. However, exporting the resulting PIF file to another product such as DisplayGraphics, would not give the intended result.

GGXA file conversion: PIF files created by GGXA that contain pictures drawn with black lines will not be visible when imported into GDDM and viewed using a GDDM application, such as the ICU. They will, however, be plotted and printed successfully by GDDM.

DisplayGraphics: PIF files created by DisplayGraphics should be drawn white with black background. They, when imported into GDDM and viewed using a GDDM application, such as the ICU, display correctly as a white image on a black background, and print as black on white background.

The structure of a PIF file

A PIF file consists of the GDF orders that are listed and described in Chapter 10, "GDF order descriptions" on page 281. Also, it can contain specific orders from the workstation. The structure of a PIF file created at a workstation is as follows:

```
File Descriptor order
Begin Symbol Set Mapping order
     Map Symbol Set Identifier order       One for each identifier
          ...                               ...
End Symbol Set Mapping order

Begin Line Type Mapping order
     Map Line Type Identifier order
End Line Type Mapping order

Begin Picture Prolog order
     Set Picture Coordinates order
     Set Picture Boundary order
     Set Page Color order
     "picture default" orders
End Picture Prolog order

Begin Segment order                        Repeated for each segment
     "segment attribute" orders           of the picture
     End Segment Prolog order
     "drawing" orders
End Segment order

Begin Symbol Set Definition order          See Note 2 below
     Load Symbol Set structured field      Repeated as needed for
     Continue Symbol Set Definition order  multiplane image symbol
     Load Symbol Set structured field      sets
          ...
End Symbol Set Definition order

Begin Line Type Definition order
     Load Line Type structured field
End Line Type Definition order
```

Figure 19. The structure of a PIF file

Notes:

1. Where present, the File Descriptor, Symbol Set Mapping, Line Type Mapping, Picture Prolog, Picture Segments, Symbol Set Definition, and Line Type Definition orders must be in the sequence shown.
2. The symbol-set definition orders are repeated for each internal symbol-set definition.
3. COMMENT and NOOP orders can be placed anywhere in the file except between the Begin Symbol Set Definition and End Symbol Set Definition orders, and between the Begin Line Type Definition and End Line Type Definition orders.
4. The GDDM-supplied conversion utility (ADMUPCT/V) removes these orders when the PIF file is converted to GDDM format:
   - The Line Type Mapping and Line Type Definition orders
   - The Symbol-Set Definition orders
   - The Set Page Color order.

The File Descriptor and Line Type Mapping orders, and the Set Page Color order, have no corresponding GDDM GDF orders. The format of these orders is described in the *IBM 3270 Personal Computer/G or /GX: Reference Information for Picture Interchange Format* manual.

Chapter 13. Computer Graphics Metafiles

A computer graphics metafile (CGM) is a file that contains information about the content of a picture, and conforms to the International Standard, *ISO 8632*, 1987(E), or later, or is of a similar format.

CGM files can be output to a number of devices, such as plotters. They can also be modified by any of the editors that accept such files.

Note: CGM support is only available in the CMS, TSO, and MVS/Batch environments.

GDDM allows you to save (export) pictures in CGM format, and to load (import) CGM files into GDDM storage. You can also convert CGM files to ADMGDF format, so that CGM files can be modified or processed by GDDM applications, and you can convert ADMGDF files to CGM.

CGM is a fairly broad standard and, as a consequence, applications that generate CGM files do so in their own way. The CGM conversion functions of GDDM are sufficiently general to handle CGM files produced by various graphical applications. Table 25 shows the conversion profiles supplied with GDDM to aid conversion between the ADMGDF format and the CGM format produced by each of the listed graphics applications:

Table 25. GDDM-supplied conversion profiles for conversion of data between ADMGDF and CGM formats.

Conversion profile	Graphics application
ADM	General Purpose
ADMCD	Corel Draw
ADMFP2	Freelance Plus V2
ADMFP3	Freelance Plus V3
ADMHG	Harvard Graphics
ADMMD	Micrografx Designer

Note: In GDDM Version 2 Release 3, the names of these conversion profiles began with the characters CGM. If you specify a profile beginning with CGM, but GDDM cannot find it, GDDM looks for the corresponding profile beginning with ADM.

The parts of the conversion process that are specific to applications are defined in a ***CGM Conversion Profile***. GDDM supplies a profile tailored to each of the applications listed above, although, depending on usage, further tailoring may be necessary.

In a number of instances, the general-purpose profile will produce acceptable output without further tailoring (especially with enhancements added for GDDM V3.2). You may want to use this profile as the basis for your own tailored conversion profiles for applications used by your enterprise. You may need to write your own profiles for other applications. See “Conversion profiles” on page 329 and the information on retrieving pictures for CGM in the *GDDM Base Application Programming Guide* for more detail.

When GDDM converts files between CGM and ADMGDF formats, an exact correspondence of the picture is not always possible, because the two formats do not map exactly onto each other. Within the limitations described later, GDDM makes the pictures correspond as closely as possible. However, converted pictures are not guaranteed to be identical to the original. The way in which specific orders are handled is described in “GDF order processing (CGSAVE call)” on page 339 and “CGM order processing (CGLOAD call)” on page 340. Note the following general restrictions on the conversions:

- GDDM supports only the Binary Encoding as defined in ISO 8632-3.
- Some CGM Version 1 orders (such as cell arrays) and all Version 2 and 3 orders are ignored on input. See Table 36 on page 340.
- CALS (the US-MIL-D standard) restricts fonts to the Hershey range and certain other fonts. The Hershey fonts are not provided as GDDM fonts, and so, on conversion from CGM to GDF, available GDDM symbol sets are substituted for them. For further details of CALS, see *MIL-D-28003A*, November 1991 (Federal Information Processing Standard publication 128).
- CALS *additional pattern sets* are not supported on input or output. Some additional linetypes (such as 6, single arrow) are not correctly converted on input.
- Double-byte character strings (DBCS) are not supported.
- There is no specific support for APL or Katakana characters.

The conversion functions are provided by two GDDM base calls, CGLOAD and CGSAVE. The PL/I declarations for the API calls are in ADMUPINK and ADMUPIRK.

Application program calls

Full descriptions of the CGLOAD and CGSAVE calls are given in Chapter 3, “The GDDM calls” on page 21.

CGLOAD

The CGLOAD call retrieves one or all pictures from a Computer Graphics Metafile (CGM) on auxiliary storage and loads it into the graphics field of the current GDDM page. It can create a graphics segment for each individual primitive, to assist with later editing of the picture, or can load the entire picture into one graphics segment.

The permitted formats of the CGM files handled by this call are defined in “CGM file format” on page 327.

CGSAVE

The CGSAVE call saves specific graphics segments or all the graphics segments in the current GDDM page, into a CGM file on auxiliary storage.

The format of the CGM files created by this call is defined in "CGM file format" on page 327.

Utility programs

GDDM provides two CGM-ADMGDF conversion utilities:

- ADMUCG
- ADMUGC

ADMUCG

This utility converts a CGM file to a floating-point ADMGDF file.

If you simply want to display the file on screen, and if you are calling it using ADMUCG, under CMS just enter:

```
ADMUCG cgmfilename
```

All other parameters are optional.

If you want to convert a CGM file to an ADMGDF file, the CGM filename, filetype, and filemode, and the GDF filename are mandatory. For example, under CMS:

```
ADMUCG cgmfilename cgmfiletype cgmfilemode gdffilename
```

The GDF filetype and filemode are optional.

Format

CMS:	
ADMUCG	**cgm-file gdf-file (keyword_parms**
TSO:	
CALL	**'data_set_name(ADMUCG)' 'FROM(cgm-file) TO(gdf-file) keyword_parms'**

Under CMS, the keywords following the opening parenthesis can occur in any order. If a keyword is specified more than once, the last specification is used.

Under TSO, all of the keywords (including FROM and TO) can occur in any order. If a keyword is specified more than once, the last specification is used. The parameters following all of the keywords (including FROM and TO) must be enclosed in parentheses.

Keywords: Permitted keywords with their parameters (with the minimum abbreviation shown in uppercase) are:

```
PICture picture-number
PROFile profname
View
NEW | REPlace
Codepage cpn
```

PICture The sequence number within the CGM of the picture to be converted. Use -1 to choose all the pictures (the default).

PROFile The name of the conversion profile to be used. If the profile name is not specified, the default is ADM, except on CMS. On CMS, the filetype of the CGM file is used as the profile name. For example, on CMS:

```
ADMUCG MYCGM MYTYPE A MYADF ADMGDF A
```

would use a default profile name of MYTYPE. However, if the CGM filetype is CGM, profile ADM is used (if found). Otherwise, profile CGM is used.

On TSO, you must allocate DD name ADMCGM to point to your SADMSAM data set to use the GDDM-supplied profiles, or to your own data set if you want to use your own profile.

View Displays the converted picture on the screen.

NEW|REPlace Specifies how to deal with the output file.

Codepage The code page that was used by the CGM generating application.

Parameters

data_set_name The name of the data set containing the GDDM load library.

cgm-file The input CGM file to be converted.

Under CMS, this parameter is a CMS fileid, as follows:

- **fn** The CMS filename of the file – any valid CMS file name.
- **ft** The CMS filetype of the file – any valid CMS file type. The default is "CGM" (which GDDM maps onto the ADM profile). If you use the appropriate conversion-profile name as the filetype, GDDM uses that profile automatically for the conversion and there is no need to specify it as a keyword.
- **fm** The CMS filemode of the file – any valid CMS file mode, or "*" (default), indicating the first occurrence of the file in the CMS search order.

Under TSO, this parameter specifies a fixed-block data set of record length 400 containing the CGM file to be loaded. The data set can be specified in one of the following ways:

- As a pre-allocated DD name pointing to a single PDS member
- As a pre-allocated DD name pointing to a sequential data set
- As a data-set name pointing to a sequential data set. The data-set name can either be

fully qualified in which case it must be specified in single quotes, for example:

```
'FROM(MYGDF)  TO(''MYHLQ.CGM.DATA'')'
```

or, if the quotes are omitted, the TSO PROFILE PREFIX is used as a prefix to the data-set name.

- As a data-set name pointing to a single PDS member. The data-set name can either be fully qualified, in which case it must be specified in single quotes, or if the quotes are omitted, the TSO PROFILE PREFIX is used as a prefix to the data-set name.

gdf-file The floating point ADMGDF file to be produced. If this parameter is omitted, no output file is created and the VIEW option is assumed.

Under CMS, this parameter is a CMS fileid, as follows:

fn The CMS filename of the file – any valid CMS file name or "=", indicating the same file name as the *cgm-file*.

ft The CMS filetype of the file. This parameter is allowed but ignored. The file type used is the current GDDM external default value (usually "ADMGDF").

fm The CMS filemode of the file. This can be omitted but must be "A", if specified.

Under TSO, **gdf-file** is the name of a member within the partitioned data set defined by the ddname ADMGDF or the GDDM external default value. The new member created by this call to ADMUCG must not exist already.

Description: The input CGM file is converted to GDF format and may be viewed, saved in an ADMGDF file, or both. If the VIEW option is selected or no output ADMGDF file is specified, the converted picture is shown on the screen and a "read" is issued. Any PF key causes the VIEW function to end, except for the key defined to invoke GDDM *User Control*. *User Control* can be entered and the picture manipulated and saved in the usual way. Manipulations have no effect on a picture subsequently saved as a GDF file. Other keys such as CLEAR and the PA keys have their usual effect.

The permitted formats of the CGM files handled by this program are defined in "CGM file format" on page 327.

The CGM descriptors are handled in the following way:

- The text from the *begin metafile* and *metafile descriptor* elements of the metafile are discarded.
- The name from the *begin picture* element is placed in the ADMGDF file as the GDF descriptor, after appropriate code page translation as defined in "National language code pages" on page 328.

ADMUCG uses the *picture_info_parms* provided in the conversion profile (see page 336), except that a *picture-number* or *cpn* (code page number) parameter will take precedence over the values in the profile.

Return codes

0 Normal completion
100 Too many (>6) name-parts: '(' not found when expected
101 Invalid option: unexpected character string after '('
102 Option value not numeric (CODEPAGE or PICTURE)
103 Keyword >8 chars or value in ( ) too long (TSO only)
104 CGLOAD failed
105 GSSAVE failed
106 ASREAD failed
107 Specified GDF file filetype is not 'A'
108 DSOPEN failed
Other Return code from failed FSINIT or FSTERM.

ADMUGC

This utility converts an ADMGDF file to a CGM file.

If you simply want to display the file on screen, and if you are calling it using ADMUGC, under CMS just enter:

```
ADMUGC gdffilename
```

All other parameters are optional.

If you want to convert an ADMGDF file to a CGM file, the ADMGDF file name, type, and mode, and CGM file name are mandatory. For example, under CMS:

```
ADMUGC gdffilename gdffiletype gdffilemode cgmfilename
```

The CGM file type and mode are optional.

Format

CMS:	
ADMUGC	**gdf-file cgm-file (keyword_parms**
TSO:	
CALL	**'data_set_name(ADMUGC)'** **'FROM(gdf-file) TO(cgm-file)** **keyword_parms'**

Under CMS, the keywords following the opening parenthesis can occur in any order. If a keyword is specified more than once, the last specification is used.

Under TSO, all of the keywords (including FROM and TO) can occur in any order. If a keyword is specified more than once, the last specification is used. The parameters following all of the keywords (including FROM and TO) must be enclosed in parentheses.

Keywords: Permitted keywords with their parameters (with the minimum abbreviation shown in upper case) are:

```
PROFile profname
View
NEW | REPlace
Codepage cpn
```

PROFile The name of the conversion profile to be used. If the profile name is not specified, the default is ADM, except on CMS. On CMS, the filetype of the CGM file is used as the profile name. For example, on CMS:

```
ADMUGC MYADF ADMGDF A MYCGM MYTYPE A
```

would use a default profile name of MYTYPE. However, if the CGM filetype is CGM, profile ADM is used (if found). Otherwise, profile CGM is used.

On TSO, you must allocate DD name ADMCGM to point to your SADMSAM data set to use the GDDM-supplied profiles, or to your own data set if you want to use your own profile.

View Displays the input ADMGDF picture on the screen.

NEW|REPlace Specifies how to deal with the output file.

Codepage The code page that the application receiving the CGM is expecting.

Parameters

data_set_name The name of the data set containing the GDDM load library.

gdf-file The name of the ADMGDF file to be converted. The file can be in either fixed or floating-point format.

Under CMS this parameter is defined as follows:

fn The CMS filename of the file – any valid CMS file name.

ft The CMS filetype of the file – any valid file type, normally "ADMGDF." If **ft** is omitted, the current GDDM default (normally ADMGDF) is used. If specified as "*" or omitted, the filetype "ADMGDF" or the current GDDM external default value is assumed.

fm The CMS filemode of the file. Any file mode may be specified here but it is ignored and assumed to be "*".

Under TSO, **gdf-file** is the name of a member within the partitioned data set defined by the ddname ADMGDF or the GDDM external default value.

cgm-file The name of the CGM file to be produced. If this parameter is omitted, no output file is produced and the VIEW option is assumed.

Under CMS this parameter is defined as follows:

fn The CMS filename of the file – any valid CMS file name, or "=", indicating that the same file name is used as given in **gdf-file**.

ft The CMS filetype of the file – any valid CMS file type. If omitted, the default is "CGM" (which GDDM maps onto the ADM conversion profile). If you specify as the filetype the name of the conversion profile for the type of CGM file you want, GDDM automatically uses that profile for the conversion.

fm The CMS filemode of the file – any valid CMS file mode, or "=". The default is "A". "*" is not permitted.

Under TSO, this parameter specifies a fixed-block data set of record length 400 containing the CGM file to be loaded. The data set can be specified in one of the following ways:

- As a pre-allocated DD name pointing to a single PDS member
- As a pre-allocated DD name pointing to a sequential data set
- As a data-set name pointing to a sequential data set. The data-set name can either be fully qualified in which case it must be specified in single quotes, for example:

  ```
  'FROM(MYGDF)  TO(''MYHLQ.CGM.DATA'')'
  ```

  or, if the quotes are omitted, the TSO PROFILE PREFIX is used as a prefix to the data-set name.
- As a data-set name pointing to a single PDS member. The data-set name can either be fully qualified, in which case it must be specified in single quotes, or if the quotes are omitted, the TSO PROFILE PREFIX is used as a prefix to the data-set name.

Note: If you use a DD name, the data set or PDS member is created if it does not already exist or is overwritten if it already exists. If you specify a data-set name that points to a sequential data set, the data set must not already exist.

Description: ADMUGC converts an ADMGDF file to a CGM file and/or allows it to be viewed at the user's display terminal. If the VIEW option is selected (or defaulted), the ADMGDF picture *prior to conversion* is shown on the screen and the user may interact with it by means of *User Control* in the same manner as for ADMUCG. Any changes made do not affect the saved picture.

The format of the CGM files created by this call is defined in "CGM file format" on page 327.

The CGM descriptors are generated in the following way:

- The text of the *begin metafile* and *metafile descriptor* elements of the metafile are generated automatically and contain information about the date and time of the conversion, the name of the source file, and the name and version of the program performing the conversion.
- The name on the *begin picture* element is copied from the ADMGDF file descriptor, after appropriate code page translation as defined in "National language code pages" on page 328.

ADMUGC uses the *picture_info_parms* provided in the conversion profile (see page 336), except that a *picture-number*

or *cpn* (code page number) parameter will take precedence over the values in the profile.

Return codes

0 Normal completion
100 Too many (>6) name-parts: '(' not found when expected
101 Invalid option: unexpected character string after '('
102 Option value not numeric for CODEPAGE
103 Keyword >8 chars or value in ( ) too long (TSO only)
104 GSLOAD failed
105 CGSAVE failed
106 ASREAD failed
107 ESEUDS failed
108 DSOPEN failed
Other Return code from failed FSINIT or FSTERM.

SEND and RECEIVE

These utilities let you transfer files between a workstation and a host session. They use the *PROFile* and *Codepage* options as used by the ADMUCG and ADMUGC utilities. They have an extra option (*ADMCGM*) to cater for the conversion between CGM and GDF formats.

The syntax for SEND and RECEIVE is as follows:

```
SEND pcfile hostfile
    ( ADMCGM PROFILE profname Codepage cpn

RECEIVE pcfile hostfile
    ( ADMCGM PROFILE profname Codepage cpn
```

External defaults

The default filetype (on CMS) or ddname (on TSO) of the conversion profiles may be changed using the external default options CMSCPT and TSOCPT respectively. The GDDM default for both is ADMCGM. The syntax is:

```
DEFAULT CMSCPT=aaaaaaaa
```

for CMS, and

```
DEFAULT TSOCPT=aaaaaaaa
```

for TSO.

CGM file format

CGM files used as input to CGLOAD or ADMUCG can be in fixed length or variable length record format, and can have a record length of up to 8000 bytes. The files must be in CGM binary format.

CGM files created by ADMUGC and CGSAVE are in fixed length record format with a record length of 400 bytes.

```
* General-purpose conversion profile for CGM/ADMGDF conversion
*
* 5695-167, 5684-168, 5686-057, 5646-001   (c) Copyright IBM Corp. 1990, 1996
*             Licensed Materials - Property of IBM
*
 PICTURE_INFO_PARMS 0 0 4 2 0 0;
 *CGM_BACKGROUND_COL -1;
 *SCALE_MODE_FACTOR 300;
 *WINDOW_LIMITS 0 0 32767 32767;
 LINE_WIDTH_FACTOR 30  1.0;
 MARKER_SIZE_FACTOR .5;

 * These colors are those defined in the CALS standard
 CGM_COLOR_INDEX    0  1 2 3 4 5 6 7 8 9 0 1 3 9 10 11 12 13 14 15 16;
 GDF_COLOR_INDEX   -2 -1 2 4 1 6 3 5 -2 -1 8 7 0 9 10 11 12 13 14 15 16;

 CGM_LINE           -2 -1 1 2 3 4 5 6 7;
 GDF_LINE            5  2 7 2 1 3 6 8 4;
 LINE_CONVERT_MODE   1  1 1 1 1 1 1 1 1;

 CGM_MARKER          5 5 2 -1 -2 3 -3 -4 -5 1  1  4  4  0 ;
 GDF_MARKER          0 1 2  3  4 5  6  7  8 9 75 10 96 74 ;
 MARKER_CONVERT_MODE 1 1 1  0  0 1  0  0  0 1  1  1  1  0 ;

 CGM_PATTERN   -12 -11 -10 -9 -8 -7 -6 -5 -4 -3 -2 -1 0  1 2  3  4 5 6 0  0;
 GDF_PATTERN    12  14  10  9  8  7  6  5  4  3  2  1 0 10 9 11 13 5 6 15 16;
 PATTERN_CONVERT_MODE 1 1 1 1 1 1  1  1  1  1 1 1 1  1 1  1  1 1 1  1  1;

 CGM_FONT_INDEX 1 2 3 4 5 6 7 8 9 10 11 12;
 CGM_FONT_NAME  TIMES_ROMAN TIMES_ITALIC TIMES_BOLD TIMES_BOLD_ITALIC
                HELVETICA HELVETICA_OBLIQUE HELVETICA_BOLD
                HELVETICA_BOLD_OBLIQUE COURIER COURIER_BOLD COURIER_ITALIC
                COURIER_BOLD_ITALIC;
 GDF_FONT_NAME  ADMUUT ADMUUTI ADMUUTB ADMUUTBI ADMUUH ADMUUHI ADMUUHB
                ADMUUHBI ADMUVC ADMUVCB ADMUVCI ADMUVCBI;
 GDF_FONT_INDEX 101 102 103 104 105 106 107 108 109 110 111 112;
 CHAR_WIDTH_FACTOR  1 1 1 1 1 1 1 1 1 1 1 1;
 CHAR_HEIGHT_FACTOR 1 1 1 1 1 1 1 1 1 1 1 1;
 FONT_CONVERT_MODE  1 1 1 1 1 1 1 1 1 1 1 1;
```

Figure 20. Example of a conversion profile.

National language code pages

Wherever character strings appear in ADMGDF or CGM source files they are subject to code page translation during conversion. Three types of code pages are involved in the process:

CGM code page
The code page defining the text string content of the CGM input or output file. Any code page in the user-modifiable module ADMDATRN is allowed. Code pages 437 (United States), 850 (multilingual), and 819 (ISO/ANSI Multilingual), as shown in the *GDDM System Customization and Administration* book, have been specifically provided for CGM support. The code-page is defined by one of the following means:

- The *picture_info_parms* keyword of the conversion profile
- The *opt-array* parameter of CGLOAD or CGSAVE
- The *Codepage* keyword in ADMUCG or ADMUGC.

Application code page
ADMUCG creates an ADMGDF file using the application code page defined (or defaulted) by the GDDM application program.

On CGLOAD, all character strings, including descriptors, are translated to the application code page. (If the application code page is 351, the translation is only approximate.)

Installation code page
In common with other GDDM file names, the names of the conversion profile and CGM source files are translated according to the GDDM installation code page. Symbol set names in conversion profiles are also translated according to the installation code page. (This is specified by the INSCPG external default value.)

Conversion profiles

The way in which graphics orders are converted from CGM to GDF and from GDF to CGM is controlled by a *Conversion Profile*. The same profile is used for conversion in either direction. Several profiles are supplied with GDDM, tailored to the CGM files produced by specific graphics applications.

The conversion profile must be an F-format or V-format file with a record length no longer than 256. Under CMS, the profile has default filetype ADMCGM (for example, the GDDM-supplied profile for Freelance Plus V2 is ADMFP2 ADMCGM). Under TSO, the ADMCGM ddname must be allocated (using the ALLOC command) to the partitioned data set containing the conversion profiles.

Format of a conversion profile

The conversion profile is a free-format file which consists of:

- Lines containing specified key-words followed by their parameters
- Continuation lines (where the previous line contained an incomplete keyword/parameter sequence).
- Comment lines (those starting with a "★").

The contents of a conversion profile can be in mixed, lower, or upper case, and leading blanks are not significant. Keywords are followed by their parameters, and they are terminated by a semicolon. Any text following the semicolon until the end of the line is treated as a comment. An example is given in Figure 20 on page 328.

The information specified in the conversion profile is of two types:

- **Picture mapping information**, which specifies how fonts, lines, patterns, and so on, are mapped during conversion.
- **Picture adjustment factors**, which define various adjustments necessary to allow for differences in CGM files from different sources.

Picture mapping information

Table 26. Picture Mapping Keywords

Attribute	**Keywords used in** conversion profile
Background color	CGM_BACKGROUND_COL
Line type	CGM_LINE GDF_LINE
Marker type	CGM_MARKER GDF_MARKER
Pattern type	CGM_PATTERN GDF_PATTERN
Color index	CGM_COLOR_INDEX or CGM_COLOUR_INDEX GDF_COLOR_INDEX or GDF_COLOUR_INDEX
Color mapping	COLOR_MAPPING or COLOUR_MAPPING
Font index and name	CGM_FONT_INDEX CGM_FONT_NAME GDF_FONT_INDEX GDF_FONT_NAME

Table 26 shows the picture mapping keywords. There are usually two of each type and they are usually specified in pairs, though the order is immaterial.

During a GDF to CGM conversion, the value of the GDF order (such as line type) is searched for in the parameters to the appropriate GDF keyword (in this case, GDF_LINE). If it is found, the corresponding parameter in the matching CGM keyword (in this case CGM_LINE) is taken as the CGM value. If the value being scanned for occurs more than once in the GDF keyword, the first occurrence is used.

The converse process is followed when converting from CGM to GDF.

For example, if line type 7 is encountered in a GDF order, this is the third item in the GDF_LINE entry in Figure 20 on page 328. The corresponding third entry in the CGM_LINE entry is 1, and the CGM line type is therefore set to 1.

Although the keyword prefix "GDF" is used, the parameters to these keywords correspond to GDDM API call parameters and not to their encoded form in the GDF orders.

Picture mapping information is subject to the following rules:

1. Keywords and their parameters can be continued on multiple lines up to a parameter count of 256.
2. Numeric values must be in the range –255 through +255.
3. If the same keyword appears more than once in the conversion profile, the final specification is the one that is used.
4. If a keyword is missing (either a pair or one of a pair), the GDDM-defined default values are used for the missing parts. These defaults are defined in Table 29 on page 331.

5. The two keywords of a pair need not appear together in the conversion profile, but it aids readability if they do.
6. If one keyword of a pair has fewer parameters supplied than the other, excess parameters on the longer one are ignored.
7. If a value in the input order stream is not found in the corresponding keyword parameters, the value inserted in the output order stream is the *conversion default* for the target format. These values are defined in Table 27.
8. If a value is not specified in the input order stream and it is required to be generated in the output order stream, the value inserted in the output is the *conversion default*.
9. The GDDM default value (0) can be used as a GDF_ keyword parameter and it is processed in the same way as any other value.

The following additional rules apply to CGM_FONT_NAME:

1. A maximum of 32 parameters may be used.
2. CGM_FONT_NAME normally has no default, but is HELVETICA if the CALS_COMPLIANCE keyword value is 1.
3. CGM font names may be enclosed in "quotes" if special characters, such as space or semicolon, are to form part of the CGM font name.
4. CGM font names may be no more than 32 characters long (excluding any enclosing quotes).

Table 27. Conversion default values

Attribute	GDF conversion default	CGM conversion default
Color	–1	1
Marker	5	3
Line	0	1
Pattern	10	1
Font	0	1
Background color	–2	White

The following sections give details specific to some of the mappings.

Note: The mapping of fonts, lines, patterns, and markers is under the control of the *convert_mode* keywords that can be used to stroke out these higher-level primitives to simple vectors. These keywords are described on page 338.

Background Color: The default CGM background color is white.

If the imported CGM does not specify the background color, white is assumed. When a CGM is exported, the background color is set to white, unless another color is specified by the CGM_BACKGROUND_COL keyword, which allows the exported CGM to have a defined color.

Color Index: CGM indexed colors (without a color table definition) and GDDM colors are mapped to each other in the standard way, with the difference that the default values are cyclical.

The sequence between the braces, {}, in Table 29 on page 331 is extended as necessary for any required input color with CGM_COLOR_INDEX increasing in steps of one, and GDF_COLOR_INDEX repeating the cycle:

```
{2 4 1 6 3 5 8 7 }
```

On CGM to GDF conversion, for CGM indexed color, color tables given in the metafile are ignored, except where the mapped GDF value is 0. In that case, the resultant color is the closest match to the color table's RGB value in the GDDM colors –2 through 16.

Table 28. Color Table created by CGSAVE/ADMUGC

Color	Red	Green	Blue
neutral	255	255	255
red	255	0	0
green	0	255	0
blue	0	0	255
yellow	255	255	0
magenta	255	0	255
cyan	0	255	255
background	0	0	0
dark blue	0	0	170
orange	255	170	0
purple	170	0	170
dark green	0	170	0
dark turquoise	0	170	170
mustard	170	170	0
grey	85	85	85
brown	170	85	10

Table 29. Keyword Defaults.

These are defined such that lines, markers, colors, etc, in one standard map to the same or similar entities in the other.

Keyword	Default values when keyword is absent from profile
CGM_LINE GDF_LINE	1 2 3 4 5 1 1 1 7 2 1 3 6 0 4 5
CGM_MARKER GDF_MARKER	1 2 3 4 5 3 3 3 3 3 5 9 2 6 10 1 3 4 5 7 8 0
CGM_PATTERN GDF_PATTERN	1 2 3 4 5 6 1 1 1 1 1 1 1 1 10 9 11 13 12 14 0 1 2 3 4 5 6 7 8
CGM_COLOR_INDEX GDF_COLOR_INDEX	0 1 2 3 4 5 6 7 8 9 {10 11 12 13 14 15 16 17} 0 1 3 9 10 11 12 13 14 15 16 -2 -1 2 4 1 6 3 5 -2 -1 { 2 4 1 6 3 5 8 7} -2 -1 0 9 10 11 12 13 14 15 16
CGM_FONT_INDEX GDF_FONT_INDEX	0 1 2 3 4 5 6 7 8 9 1 100 101 102 103 104 105 106 107 108 109 0
GDF_FONT_NAME CGM_FONT_NAME	(none) (none)

Note: for a description of the COLOR_INDEX default see “Color Index”

For CGM direct color, RGB values are approximated to the closest match in the GDDM colors 0-16, except where the RGB value is used in a non-background area fill, in which case its closest match in the 64 color pattern set ADMCOLSD is used.

On CGSAVE, a color table is placed in the metafile and entries are selected according to the COLOR_INDEX values in the conversion profile. The entries in this color table are shown in Table 28 on page 330.

The order of the entries in the color table depends on the *color_index* values in the conversion profile. The *n*th entry in the table has a color value equivalent to the GDF color that CGM color *n* maps to in the profile, where 1≤n≤16.

Color mapping: The *COLOR_MAPPING* keyword is an alternative to the COLOR_INDEX keyword as way of specifying color mapping for the creation and interpretation of CGM data. The COLOR_MAPPING keyword takes a single parameter that controls the way GDDM interprets colors in a CGM and also controls the size of the color table generated in a GDDM CGM. The values that this parameter can have are shown in Table 30 on page 332.

The color is determined by the Red-Green-Blue (RGB) values specified in the CGM. These RGB values are in the range 0-255 and are shown in Table 31 on page 332. (The color names are not intended to accurately indicate the color shown; you should not try to select a color based on its descriptive name.)

The mapping of the RGB values to a GDDM color is explained in Table 30 on page 332.

Table 30. CGM color_mapping parameter

Color_mapping parameter	Description
0	This is the default color mapping mode and is compatible with GDDM Version 2 Release 3 and GDDM Version 3 Release 1.
16	When interpreting a CGM, the RGB specified in the CGM is mapped to a GDDM color in the range –2 to 16. The color chosen has the RGB value closest to that specified in the CGM. The GDDM color RGB values used are the first 16 shown in Table 31 on page 332. If a CGM using indexed color selection mode omits the RGB values for a color, the CGM color index is used as the GDDM color number. When generating a CGM, a 16-element color table is generated. The CGM color index values correspond to the GDDM color numbers. The only exception to this is for Black (–1), White (–2), Neutral (7) and Background (8). The generated color table specifies CGM color index 7 as White and 8 as black. GDDM colors –1 and 8 map to black, and GDDM colors –2 and 7 map to white. Any CGM_COLOR_INDEX or GDF_COLOR_INDEX profile entries are ignored.
255	When interpreting a CGM, the RGB specified in the CGM is mapped to a GDDM color in the range –2 to 255. The color chosen has the RGB value closest to that specified in the CGM. The GDDM color RGB values used are shown in Table 31 on page 332. If a CGM using indexed color selection mode omits the RGB values for a color, the CGM color index is used as the GDDM color number. When generating a CGM, a 255-element color table is generated. The rules described for *COLOR_MAPPING=16* apply. Any CGM_COLOR_INDEX or GDF_COLOR_INDEX profile entries are ignored.

Table 31 (Page 1 of 3). GDDM colors used for CGM interpretation

Color	Red	Green	Blue
–2 white	255	255	255
–1 black	0	0	0
1 blue	0	0	255
2 red	255	0	0
3 magenta	255	0	255
4 green	0	255	0
5 cyan	0	255	255
6 yellow	255	255	0
7 neutral			
8 background			
9 dark blue	0	0	170
10 orange	255	128	0
11 purple	170	0	170
12 dark green	0	146	0
13 dark turquoise	0	146	170
14 mustard	192	160	32
15 gray	131	131	131
16 brown	144	48	0
17 shadowy green	0	36	0
18 shadowy blue	0	36	85
19 dreary blue	0	36	170
20 happy blue	0	36	255
21 drab green	0	73	0
22 shadowy cyan	0	73	85
23 twilight blue	0	73	170
24 fresh blue	0	73	255
25 muddy green	0	109	0
26 muddy cyan	0	109	85
27 numb cyan	0	109	170
28 light blue	0	109	255
29 stormy green	0	146	85
30 gloomy cyan	0	146	255
31 dusky green	0	182	0
32 murky green	0	182	85
33 twilight cyan	0	182	170
34 wintry cyan	0	182	255
35 ominous green	0	219	0

Table 31 (Page 1 of 3). GDDM colors used for CGM interpretation

Color	Red	Green	Blue
36 mysterious green	0	219	85
37 faded cyan	0	219	170
38 mysterious cyan	0	219	255
39 glowing green	0	255	85
40 scintillating green	0	255	170
41 shadowy red	43	0	0
42 shadowy pink	43	0	85
43 deep blue	43	0	170
44 hot blue	43	0	255
45 shadowy yellow	43	36	0
46 muddy blue	43	36	85
47 dense blue	43	36	170
48 glowing blue	43	36	255
49 drab yellow	43	73	0
50 drab cyan	43	73	85
51 wintry blue	43	73	170
52 warm blue	43	73	255
53 dull green	43	109	0
54 stagnant cyan	43	109	85
55 sombre cyan	43	109	170
56 brilliant blue	43	109	255
57 sombre green	43	146	0
58 twilight green	43	146	85
59 dim cyan	43	146	170
60 scintillating blue	43	146	255
61 gloomy green	43	182	0
62 wintry green	43	182	85
63 evening cyan	43	182	170
64 mellow cyan	43	182	255
65 mellow green	43	219	0
66 happy green	43	219	85
67 sweet green	43	219	170
68 faint cyan	43	219	255
69 perky green	43	255	0
70 bright green	43	255	85
71 shining green	43	255	170
72 light cyan	43	255	255

Table 31 (Page 2 of 3). GDDM colors used for CGM interpretation

Color	Red	Green	Blue
73 drab red	85	0	0
74 drab pink	85	0	85
75 faded blue	85	0	170
76 stunning blue	85	0	255
77 dull red	85	36	0
78 muddy pink	85	36	85
79 flat blue	85	36	170
80 bright blue	85	36	255
81 muddy yellow	85	73	0
82 dull pink	85	73	85
83 faint blue	85	73	170
84 vibrant blue	85	73	255
85 dull yellow	85	109	0
86 dreary green	85	109	85
87 cheerful blue	85	109	170
88 dazzling blue	85	109	255
89 deep green	85	146	0
90 dense green	85	146	85
91 deep cyan	85	146	170
92 shining blue	85	146	255
93 faded green	85	182	0
94 flat green	85	182	85
95 murky cyan	85	182	170
96 flat cyan	85	182	255
97 faint green	85	219	0
98 fresh green	85	219	85
99 vibrant green	85	219	170
100 bleary cyan	85	219	255
101 stunning green	85	255	0
102 dazzling green	85	255	85
103 pastel green	85	255	170
104 dazzling cyan	85	255	255
105 stagnant red	128	0	0
106 stagnant pink	128	0	85
107 sombre pink	128	0	170
108 dusky pink	128	0	255
109 sombre red	128	36	0
110 numb pink	128	36	85
111 dreary pink	128	36	170
112 radiant blue	128	36	255
113 muddy orange	128	73	0
114 dim red	128	73	85
115 stormy pink	128	73	170
116 sparkling blue	128	73	255
117 stagnant yellow	128	109	0
118 numb yellow	128	109	85
119 perky blue	128	109	170
120 soft blue	128	109	255
121 drab gray	1	1	1
122 dull gray	8	8	8
123 stagnant gray	16	16	16
124 sombre gray	24	24	24
125 stormy gray	32	32	32
126 dim gray	41	41	41
127 dusky gray	49	49	49
128 evening gray	57	57	57
129 gloomy gray	65	65	65
130 murky gray	74	74	74
131 foggy gray	82	82	82
132 ominous gray	90	90	90
133 mellow gray	98	98	98
134 dowdy gray	106	106	106
135 mysterious gray	115	115	115
136 bleary gray	123	123	123
137 hot gray	139	139	139
138 perky gray	148	148	148
139 glowing gray	156	156	156
140 sweet gray	164	164	164
141 light gray	172	172	172
142 bright gray	180	180	180
143 vibrant gray	189	189	189

Table 31 (Page 2 of 3). GDDM colors used for CGM interpretation

Color	Red	Green	Blue
144 radiant gray	197	197	197
145 dazzling gray	213	213	213
146 sparkling gray	222	222	222
147 soft gray	230	230	230
148 pastel gray	238	238	238
149 faded gray	246	246	246
150 faint gray	254	254	254
151 dreary yellow	128	146	85
152 sweet blue	128	146	170
153 pale blue	128	146	255
154 dim yellow	128	182	0
155 bleary green	128	182	85
156 foggy cyan	128	182	170
157 gentle blue	128	182	255
158 hot green	128	219	0
159 light green	128	219	85
160 sparkling green	128	219	170
161 glowing cyan	128	219	255
162 brilliant green	128	255	0
163 soft green	128	255	85
164 hazy green	128	255	170
165 soft cyan	128	255	255
166 evening red	170	0	85
167 faded pink	170	0	255
168 stormy red	170	36	0
169 foggy red	170	36	85
170 deep pink	170	36	170
171 ominous pink	170	36	255
172 dusky red	170	73	0
173 mellow red	170	73	85
174 twilight pink	170	73	170
175 flat pink	170	73	255
176 sombre orange	170	109	0
177 bleary red	170	109	85
178 gloomy pink	170	109	170
179 pastel blue	170	109	255
180 stormy yellow	170	146	0
181 deep yellow	170	146	85
182 murky pink	170	146	170
183 hazy blue	170	146	255
184 twilight yellow	170	182	0
185 gloomy yellow	170	182	85
186 warm green	170	182	170
187 misty blue	170	182	255
188 faded yellow	170	219	0
189 radiant green	170	219	85
190 pale green	170	219	170
191 vibrant cyan	170	219	255
192 mellow yellow	170	255	0
193 faded green	170	255	85
194 misty green	170	255	170
195 faded cyan	170	255	255
196 murky red	213	0	0
197 mysterious red	213	0	85
198 evening pink	213	0	170
199 mysterious pink	213	0	255
200 ominous red	213	36	0
201 hot red	213	36	85
202 dense pink	213	36	170
203 bleary pink	213	36	255
204 dowdy red	213	73	0
205 fresh red	213	73	85
206 foggy pink	213	73	170
207 blunt pink	213	73	255
208 happy red	213	109	0
209 light red	213	109	85
210 mellow pink	213	109	170
211 fresh pink	213	109	255
212 dusky orange	213	146	0
213 vibrant red	213	146	85
214 sparkling red	213	146	170

Table 31 (Page 3 of 3). GDDM colors used for CGM interpretation

Color	Red	Green	Blue
215 bright pink	213	146	255
216 murky yellow	213	182	0
217 foggy yellow	213	182	85
218 pale red	213	182	170
219 steaming blue	213	182	255
220 wintry yellow	213	219	0
221 flat yellow	213	219	85
222 faint yellow	213	219	170
223 blizzard blue	213	219	255
224 mysterious yellow	213	255	0
225 blunt yellow	213	255	85
226 blizzard green	213	255	170
227 blizzard cyan	213	255	255
228 sweet red	255	0	85
229 wintry pink	255	0	170
230 perky red	255	36	0
231 bright red	255	36	85
232 dowdy pink	255	36	170
233 sweet pink	255	36	255
234 stunning red	255	73	0
235 scintillating red	255	73	85
236 faint pink	255	73	170
237 radiant pink	255	73	255
238 warm red	255	109	0
239 dazzling red	255	109	85
240 faded red	255	109	170
241 sparkling pink	255	109	255
242 dense orange	255	146	0
243 shining orange	255	146	85
244 hazy red	255	146	170
245 pale pink	255	146	255
246 ominous orange	255	182	0
247 pastel orange	255	182	85
248 faint red	255	182	170
249 hazy pink	255	182	255
250 dowdy orange	255	219	0
251 bleary orange	255	219	85
252 blizzard red	255	219	170
253 blizzard pink	255	219	255
254 sparkling yellow	255	255	85
255 blizzard yellow	255	255	170

```
*
* The color_mapping keyword can be used to provide an
* alternative to the color_index keywords and
* color_shading patterns used by default.
*
* Mapping to and from 16 or 255 GDDM colors is
* available
*
*COLOR_MAPPING      16;
*COLOR_MAPPING     255;
```

Figure 21. CGM color_mapping keyword

Depending on the application for which you are providing a profile, the value you specify on the *COLOR_MAPPING* parameter gives different results. To determine whether 16- or 255-color mapping gives you better results, perform the conversion twice, modifying the sample profile each time to use a different setting of the parameter. Remove the asterisk preceding either parameter setting in the sample profile to use it in the conversion. The appearance of the colors also depends on the device on which they are output. For example, an eight-color display shows the color orange as red, because of color mapping.

Note: GDDM accepts the keyword with the alternative spelling COLOUR_MAPPING.

Marker: On CGM to GDF conversion, marker values in the CGM input may be mapped to either the system or user defined markers of GDDM. The user-defined markers that the picture will reference are those current when the CGLOAD call was issued. If no user-defined set has been loaded, CGLOAD loads ADMDHIMJ.

On GDF to CGM conversion, markers are mapped or stroked out under the control of the *marker_convert_mode*. If stroked out, they are converted to vector form using the symbols in the currently-loaded vector marker set, or, if none, in ADMDHIMJ, which is loaded by GDDM if needed.

Pattern: The term “pattern” in this context refers to both GDDM patterns and CGM hatches that approximately correspond to each other. CGM patterns are converted in the same way as CGM hatches, and are used for interior style 2 areas. Default-hatched areas in imported CGMs are treated

as if hatch index 1 were specified. On CGM import, shading patterns from the 64 geometric pattern symbol set ADMPATT• are used for GDF patterns greater than 64 (unless a shading pattern symbol set is already loaded or 64 colour pattern set ADMCOLSD is being used in an area fill). This allows the use of patterns such as 100 and 98 for CGM hatches 5 and 6 on devices that can load patterns.

The precise way that GDF patterns are mapped to CGM (when the *pattern_convert_mode* specifies **non-stroked-out** mode) is as follows:

Table 32. GDF patterns and CGM mapping

GDF pattern number	CGM elements generated: area with interior style
15	4 (empty)
0, 16	1 (solid)
other values	3 (hatch)

Hence, most GDF pattern values map to CGM hatch values, and vice versa, except for 0, 15 and 16, which map to empty and solid areas. Consequently, the CGM_PATTERN keyword parameters corresponding to GDF_PATTERN 0, 15 and 16 are ignored. For **stroked-out** mode (*pattern_convert_mode* = 0) CGM areas are filled with vectors to correspond to the original GDF pattern.

Font Index and Font Name: Fonts (symbol sets) are handled in a special way using four picture mapping keywords and the FONT_CONVERT_MODE adjustment factor.

On CGM export, if any FONT_CONVERT_MODE is 0, the corresponding font is stroked out to vectors if the font is available in vector form. If FONT_CONVERT_MODE is 1, or the font is not available in vector format, the following font mapping process is used:

- If CGM_FONT_NAME has at least one font name, “font mapping by name” is performed:
  - A font list is placed in the CGM containing all the font names in the CGM_FONT_NAME list, indexed from 1 to number-of-fonts. (But if CALS_COMPLIANCE is specified, the list will only contain allowed font names. See the CALS_COMPLIANCE keyword.)
  - The GDF symbol-set name used is looked up in the GDF_FONT_NAME list and its position determines the CGM_FONT_NAME to be indexed in the CGM.
  - If there is no matching CGM font name, CGM font index 1 is used (that is, the first named CGM font name).
- If CGM_FONT_NAME is absent, or has no font names, “font mapping by index” is performed:
  - No font list is placed in the CGM (unless CALS_COMPLIANCE is specified, in which case a font list **will** be placed in the CGM, containing CGM font names matching GDDM core interchange symbol set names found in GDF_FONT_NAME, or HELVETICA. See the CALS_COMPLIANCE keyword.)
  - The GDF symbol-set name used is looked up in the GDF_FONT_NAME list and its position determines the CGM_FONT_INDEX to be used in the CGM.
  - If there is no matching CGM font index, CGM font index 1 is used.

On CGM import, GDDM saves an internal copy of the Font List if one is found in the CGM (and records the font names, and any truncation, in a GDDM trace if tracing is activated with the CGMREP trace keyword). If a text element occurs **before** any font index element, font index 1 is assumed. The following mapping process is then used:

- If CGM_FONT_NAME has at least one font name, “font mapping by name” is performed:
  - The CGM font index in the CGM is used to select:
    - A CGM font name from the saved list
    - The first font name if the index is out of range
    - HELVETICA, if there was no font list in the CGM.
  - The selected CGM font name is then looked up in the CGM_FONT_NAME list, and its position determines the GDF_FONT_INDEX and the GDF_FONT_NAME in the usual way (ignoring the CGM_FONT_INDEX entries). If no match is found, but the font name started “HERSHEY:”, the search is repeated as if the name started “HERSHEY/”. If there is still no match, the font index found in the CGM is looked up in the CGM_FONT_INDEX list, and its position determines the GDF_FONT_INDEX and GDF_FONT_NAME in the usual way.
- If CGM_FONT_NAME is absent, or has no font names, “font mapping by index” is performed. The font index found in the CGM is looked up in the CGM_FONT_INDEX list, and its position determines the GDF_FONT_INDEX and GDF_FONT_NAME.

In either case, the default font ADMDVECP is loaded if there is no valid GDDM font name in the conversion profile corresponding to the chosen LCID (or the defaults above) or if the symbol set does not exist. If the LCID is less than 65, the device default font is used. If a vector symbol set has already been loaded for a particular LCID (greater than or equal to 65) before the CGLOAD call, the way this is handled depends on the setting of the *symbol-set* parameter of CGLOAD. There is an upper limit of 60 symbol sets that can be handled during conversion.

Picture adjustment factors: Several adjustment factors may be specified in a conversion profile. These have in most cases been provided in order to cater for the different types of CGM file commonly found and the inconsistencies sometimes found between them. They are:

Picture_Info_Parms
Window_Limits

| Scale_Mode_Factor
| CALS_Compliance
Line_Width_Factor
Marker_Size_Factor
Char_Width_Factor
Char_Height_Factor
Line_Convert_Mode
Marker_Convert_Mode
Pattern_Convert_Mode
Font_Convert_Mode

Picture_Info_Parms: The *picture_info_parms* adjustment factor supplies defaults for the *opt-array* parameters of the CGLOAD and CGSAVE calls. The ADMUCG and ADMUGC utilities also use the *picture_info_parms* (except where overridden by explicit utility keywords). If the call is coded with the appropriate *opt-array* parameter as the default (that is, 0 or omitted), the corresponding *picture_info_parms* parameter is used in its place. In other words, *picture_info_parms* supplies defaults for these calls.

When a value from *picture_info_parms* is used, it has the same meaning and allowed range of values as that defined for the CGLOAD or CGSAVE call (including the use of 0 to take the default value). The following items can be set:

- **picture-number**: If the *picture-number* parameter on the CGLOAD call is set to default (0) or omitted, the value supplied here is used.
- **seg-base**: If the *seg-base* parameter on the CGLOAD call is set to default (0) or omitted, the value supplied here is used.
- **load-type**: If the *load-type* parameter on the CGLOAD call is set to default (0) or omitted, the value supplied here is used.
- **symbol-set**: If the *symbol-set* parameter on the CGLOAD call is set to default (0) or omitted, the value supplied here is used.
- **seg-use**: If the *seg-use* parameter on the CGLOAD call is set to default (0) or omitted, the value supplied here is used.
- **code-page**: If the *code-page* parameter on the CGLOAD or CGSAVE call is set to default (0) or omitted, the value supplied here is used. Unlike the other *picture_info_parms*, *code-page* applies to both calls.

Window_Limits: The *window_limits* allow the specification of a coordinate range (window) to override the extent of the CGM Virtual Device Coordinates (VDC). This only has an effect on CGM to GDF conversion, and is used to overcome the problem that CGM files from some applications use only a small fraction of the VDC extent that they define. *Window_limits* also allows a limited zoom and crop capability. It is ignored on GDF to CGM conversion.

If the *window_limits* are absent, the window defaults first to the CGM VDC extent. If the VDC extent is not defined in the CGM file, the default values used are:

–32768, 32767, –32768, 32767

| ***Scale_Mode_Factor:*** Exported CGMs can contain a scaling factor in order to produce results with fixed measurements. The value specified for the Scale Mode Factor is used for the size, in millimeters, of the longer side of the picture. The value must be positive. A number of CGM applications do not use this metric mode and may respond with errors for CGMs that contain it.

| If a value is not specified, or if a value of 0 is used, exported CGMs do not have a scaling factor.

| ***CALS_Compliance:*** This takes a single parameter that is used to select whether or not exported CGMs should be constrained to the limits specified by MIL-D-2803A. When the parameter is 0 (the default), or the keyword omitted, no constraints are imposed. When the parameter is 1, the following constraints are imposed on exported CGMs:

- The string “MIL-D-28003A/BASIC-1” is included in the Metafile Descriptor element.
- Polygon elements are limited to 1024 points. Additional points are truncated (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- Graphical text strings have any control codes (X'01'-X'1F' and X'80 - X'9F') changed to ASCII space characters (X'20').
- Line types out of the range 1 through 15 are changed to 1 (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- Line and edge widths greater than 10% of the smaller VDC range are reduced to 10% (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- Marker types out of the range 1 through 5 are changed to 1 (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- A font list is always placed in the CGM, containing only font names from this list:

```
HERSHEY/CARTOGRAPHIC_ROMAN
HERSHEY:CARTOGRAPHIC_ROMAN
HERSHEY/CARTOGRAPHIC_GREEK
HERSHEY:CARTOGRAPHIC_GREEK
HERSHEY/SIMPLEX_ROMAN
HERSHEY:SIMPLEX_ROMAN
HERSHEY/SIMPLEX_GREEK
HERSHEY:SIMPLEX_GREEK
HERSHEY/SIMPLEX_SCRIPT
HERSHEY:SIMPLEX_SCRIPT
HERSHEY/COMPLEX_ROMAN
HERSHEY:COMPLEX_ROMAN
HERSHEY/COMPLEX_GREEK
HERSHEY:COMPLEX_GREEK
HERSHEY/COMPLEX_SCRIPT
HERSHEY:COMPLEX_SCRIPT
HERSHEY/COMPLEX_ITALIC
HERSHEY:COMPLEX_ITALIC
HERSHEY/COMPLEX_CYRILLIC
HERSHEY:COMPLEX_CYRILLIC
HERSHEY/DUPLEX_ROMAN
HERSHEY:DUPLEX_ROMAN
HERSHEY/TRIPLEX_ROMAN
HERSHEY:TRIPLEX_ROMAN
HERSHEY/TRIPLEX_ITALIC
HERSHEY:TRIPLEX_ITALIC
HERSHEY/GOTHIC_GERMAN
HERSHEY:GOTHIC_GERMAN
HERSHEY/GOTHIC_ENGLISH
HERSHEY:GOTHIC_ENGLISH
HERSHEY/GOTHIC_ITALIC
HERSHEY:GOTHIC_ITALIC
TIMES_ROMAN
TIMES_ITALIC
TIMES_BOLD
TIMES_BOLD_ITALIC
HELVETICA
HELVETICA_OBLIQUE
HELVETICA_BOLD
HELVETICA_BOLD_OBLIQUE
COURIER
COURIER_BOLD
COURIER_ITALIC
COURIER_BOLD_ITALIC
```

If a font list was forced, this fact is reported in a GDDM trace statement if tracing is active with the CGMREP keyword.

- Out-of-range font indexes (that is, which do not point to a font in the font list) are changed to 1 (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- Hatch indexes out of the range 1 through 6 are changed to 1 (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).
- Edge types out of the range 1 through 5 are changed to 1 (and this fact reported in a GDDM trace statement if tracing is active with the CGMREP keyword).

Line_Width_Factor: This takes two positive parameters, the second of which is optional. The default values are 1,1.

The first parameter applies in nearly all circumstances. It is used as a divisor in converting from CGM to GDF and as a multiplier in the reverse direction. The CGM line width value is divided by this parameter to obtain the GDDM line width value. On 3270 displays, width values less than 2 give the standard line width (1 pixel). A value of 2 or more gives the doubled line width (2 pixels). For further details, see the GSLW call in Chapter 3, “The GDDM calls” on page 21.

The second parameter is used only on CGM-to-GDF conversion, where the CGM specifies absolute values for line and edge width. This value is used as a divisor to influence the value that GDDM normally sets for imported lines and edges. Where the imported CGM specifies a scaling mode with a metric scale factor and uses absolute values for lines and edges, GDDM uses this to determine the required thickness of lines in millimeters. The output should maintain the correct values for printing on a 300dpi IPDS printer such as the IBM 4028, which uses a 3 dot unit line width. Where the final output is intended for a 240dpi printer, the second parameter needs to be changed from 1 to 1.25. Similarly, adjustments need to be made to the second parameter where the unit line width is not 3 dots. Where no metric scale factor is specified in the CGM, GDDM bases the standard line width on .001 of the longer side of the VDC extent specified in the CGM.

Marker_Size_Factor: This takes a single parameter that is used to adjust sizes of all markers of all marker types. It is analogous in operation to *line_width_factor*. The default value is 1.0.

Char_Width_Factor, Char_Height_Factor: These factors affect the shape of text characters when GDDM generates or interprets CGM data.

Figure 22 on page 338 shows how these factors can be specified in combination with other font-related keywords in the ‘general’ CGM profile supplied with GDDM.

The character width and height adjustments correspond in position to their respective *font-name* and *font-index* values.

- When GDDM is importing a CGM, the character box width and height of each font are divided by the appropriate factors.
- When GDDM is exporting a CGM, the character box width and height of each font are multiplied by the appropriate factors.

This modification of the shape of the characters assists in matching the appearance of converted character strings because character aspect ratios can differ from font to font.

```
CGM_FONT_INDEX     1         2         3       4        5;
GDF_FONT_INDEX     101       102       103     104      105;
GDF_FONT_NAME      ADMUUDRP  ADMUUKSF  ADMUVC  ADMUUFSS ADMUUCIP;
CHAR_WIDTH_FACTOR  0.90      0.77      1.2     0.85     1;
CHAR_HEIGHT_FACTOR 0.83      0.83      1.1     0.83     1;
FONT_CONVERT_MODE  1         1         1       1        1;
```

Figure 22. Character-height and character-width factors for conversion of fonts between ADMGDF and CGM formats.

Line_, Pattern_, Font_, and Marker_Convert_Mode: These each have the same number of parameters as the corresponding CGM_ or GDF_ keyword, each of which should be 0 or 1. On GDF to CGM conversion, these control whether lines, patterns, markers, and fonts should be stroked out (when 0) or converted to corresponding CGM index values (when 1). On CGM to GDF conversion, lines, patterns, markers, and fonts are never stroked out.

If there are fewer parameters supplied than on the corresponding CGM_ or GDF_ keyword, missing values are assumed to be 0.

Note: If a *convert_mode* keyword is not specified, the default values used are 0s (the corresponding primitives are stroked out).

Stroked out mode (0) should be used when you want an exact replication of the original picture at the expense of generating many short vector orders, resulting in a larger graphics file. Non-stroked-out mode (1) should be used when you are more concerned with generating a concise file and when you are intending to edit the converted file. As an example, the double dot line type does not exist in CGM; it can be emulated by stroking out the short dots (at considerable cost in terms of processing time and file size).

Pattern_Convert_Mode has no effect on GDF patterns 0, 15, or 16; these are never stroked out.

Conversion Profiles supplied with GDDM: GDDM supplies a set of standard conversion profiles for use in importing and exporting CGM files to and from various applications. These are listed in Table 33. There is not a 1:1 mapping of all CGM to GDF orders, and so it is not possible to guarantee that all pictures will convert perfectly in all circumstances. Conversion of pictures from one format to another and then back again is unlikely to produce exactly the same graphics orders as in the original, and users should check carefully the restrictions listed in “CGM order processing (CGLOAD call)” on page 340 and “GDF order processing (CGSAVE call)” on page 339 before discarding any original data.

Table 33. GDDM-supplied conversion profiles

Name of conversion profile	Use
ADMFP2	Freelance Plus V2
ADMFP3	Freelance Plus V3
ADMHG	Harvard Graphics
ADMCD	Corel Draw
ADMMD	Micrografx Designer
ADM	General purpose

Table 34. CGM test patterns for conversion profile creation

Name	Description	Instructions
COLORxxx	color test pattern	Draw a set of filled rectangles each with a different fill color. Label each as it is drawn with text describing the color.
PATNSxxx	hatch test patterns	Using the same set of rectangles as for color specification, vary the fill pattern in an orderly fashion with notations describing the appearance (for example, solid, empty, horizontal hatch, 45 degree diagonal hatch, large diagonal hatch, vertical hatch, and so on).
LINESxxx	linetypes	Draw a set of lines, each with a different line type and appropriate description (for example, solid, dotted, dashed, dash dot, and so on).
WIDTHxxx	linewidths	Draw a set of lines, each with a different line width and appropriate description (for example, thinnest, thin, medium, thick, thickest).
MARKSxxx	markers	Create a metafile depicting the available marker symbols with appropriate description (for example, cross, triangle, filled triangle, and so on).
FONTSxxx	fonts	Create a metafile that lists the available fonts with the font description in that same font (for example, Swiss Light, Roman Bold, Italic, Italic bold, and so on). Use vertical reference lines to delimit the string length. This is useful for character aspect ratio adjustment.

Creating conversion profiles for other applications: Conversion profiles for applications other than those catered for above can fairly easily be created and tested. The technique involves creating CGM test patterns using the application program for which a conversion profile is required and then adjusting the new conversion profile until the correct picture is obtained after conversion. Table 34 lists the recommended test patterns for creating a conversion profile CGMxxx where “xxx” is a mnemonic for the CGM-generating application. They are not provided with GDDM.

In addition, place a border around the application drawing space to allow for the possible requirement of window adjustment.

It is not recommended that you create all these picture elements in the same CGM, because it would become too confused to analyze. If the naming convention described is

adhered to, it allows you to keep track of the test patterns for many different applications.

GDF order processing (CGSAVE call)

This section defines how GDF orders are dealt with during the GDF to CGM conversion process. Orders described as "ignored" are skipped in the input datastream and the next order is dealt with. Orders described as "equivalent" have CGM orders generated to produce the equivalent effect.

Pushing and Popping: There is no direct equivalence in CGM of the GDF pushing and popping of environments, and so this is handled by generating appropriate CGM orders to set and reset the environment.

Area Fill: The definitions of the area fill algorithms for GDF and CGM are different, and the way GDF areas are converted to CGM may result in a different appearance in some cases.

The type of CGM area-fill implemented by most CGM receiving applications limits the area definition to using only the polygon order. Hence, areas containing only complete polygons (regardless of the number of line crossings) are drawn the same in CGM as in GDF, but those containing move orders are subject to restrictions.

If a *move* or *set current position* order is found within a GDF area definition, and the move is to outside the smallest rectangle aligned with the X/Y axes that encloses all of the graphics already drawn, the current CGM polygon is closed and a new polygon started at the point after the move. Hence, each GDF area may give rise to several CGM areas.

However, if the move is to within the enclosing rectangle, the move is converted to a line so that the polygon before the move and the one after are joined together to form one polygon. At the end of the second polygon, a second line is generated back over the previously generated line to join the two polygons.

The above effect causes some GDF areas to appear incorrectly after conversion. If more than one CGM polygon is created from one GDF area and these CGM polygons overlap, the result depends on the order that they are drawn (the last one appears on the top), whereas the appearance of areas of overlap in GDDM areas are defined by the GDDM line crossing fill algorithm.

Multi-line text strings: The position of multi-line GDF text strings may not be exactly replicated by the CGM orders. If positioning problems are found, they can often be solved by specifying a *font_convert mode* of 0 (stroked out). However, the position of the text may still change, particularly if the horizontal alignment of the multi-line string was defined as `'centered'` or `'right'`.

Table 35 (Page 1 of 2). GDF order processing

GDF order name	Comments
Arc	Generates CGM arc order, except when within an area fill, when the arc is stroked out to line orders.
Arc parameters	Equivalent.
Area	Generates CGM filled polygon order(s). Subject to restrictions as described in "Area Fill."
Background color mix	Equivalent.
Call segment	Ignored.
Character angle	Equivalent.
Character box	Equivalent. Height converts to CGM character height. Width converts to CGM character expansion factor.
Character-box spacing	Equivalent. (character spacing)
Character direction	Equivalent. (character path)
Character precision	Precisions 1 and 2 are converted to CGM stroked precision text with a suitable CGM font index as defined in the conversion profile. Precision 3 is either stroked out to vector or mapped to a CGM font index under control of the profile. Precisions 1 and 2 are never stroked out.
Character set	Equivalent. The mapping of character set to font index is controlled by the profile.
Character shear	Equivalent. (baseline angle and up-vector orders).
Character string	Equivalent. Result depends on character precision. See under *character precision*. Character string is translated to the required CGM code page.
Color	Colors are mapped under control of the profile.
Comment	Ignored.
Current position	Equivalent.
End area	Equivalent.
Extended color	Equivalent.
Fillet	Same as arc. orders.
Foreground color mix	Ignored. CGM only supports overpaint.
Fractional line width	Maps to CGM line width order after rounding to integer. CGM linewidth may be *absolute* or *scaled*. CGSAVE always generates absolute CGM linewidths. See also *line_width_factor* in the conversion profile.
Full arc	Same as *arc*
Image begin Image data Image end	Ignored.

Table 35 (Page 2 of 2). GDF order processing

GDF order name	Comments
Line	Generates CGM polyline order.
Line type	Maps to CGM linetype under control of the conversion profile.
Line width	Same as for *fractional line width*.
Marker Marker box Marker scale Marker type	Maps to CGM markers under control of the conversion profile.
Model transform	There is no corresponding CGM transform order, and so CGM coordinates that are generated subsequent to this GDF order have the model transform pre-applied to them. The *additive* and *preemptive* forms of this order as processed are not specifically handled; they are processed in the same ways as the *replace* form. Combinations of character angle, spacing, direction and shear with a model transform applied may result in text misplacement.
No-op	Order is skipped.
Pattern	Maps to CGM hatch (pattern) types under control of the conversion profile.
Pick (tag) identifier	Ignored.
Pop	No CGM pop order, so pop is handled by generating appropriate color, width, etc. orders when needed.
Process specific control	PSC defines GDF's symbol sets and defaults for mix, width, etc. Symbol sets are mapped to CGM fonts using the font mapping defined in the conversion profile. PSC specifying defaults are ignored.
Relative line	Generates a polyline order.
Segment attribute	Ignored.
Segment attribute modify	Ignored.
Segment characteristics	Ignored unless CHID is X'80'.
Segment end Segment end prolog	CGM does not support segments so no specific segment end can be generated. However, the pushing and popping of segment attributes, viewing window, transform etc are converted to CGM orders.
Segment position	Ignored.
Segment start	Invisible segments are not drawn. Other aspects of order are ignored (i.e. detectability, highlight, transformability, chaining).
Set viewing Window	Maps to CGM clipping window order.
Text Alignment	Equivalent.

CGM order processing (CGLOAD call)

This section defines how CGM orders are dealt with during the CGM to GDF conversion process. Orders described as "Ignored" are skipped in the input datastream and the next order is dealt with. Orders described as "Equivalent" are converted to equivalent GDF orders and/or affect the generation of subsequent GDF orders.

Table 36 (Page 1 of 3). CGM order processing

CGM code and order name	Comments
CGM Class 0 orders: Delimiter elements	
0 No-op	Ignored.
1 Begin metafile	Text returned as first part of CGLOAD *descriptor1* parameter
2 End metafile	Terminates picture definition
3 Begin Picture	Text returned as CGLOAD *descriptor2* parameter
4 Begin Picture Body	Causes GDDM picture to be started and first segment opened
5 End Picture	
6 Begin Segment 7 End Segment 8 Begin Figure 9 End Figure 13 Begin Protection Region 14 End Protection Region 15 Begin Compound Line 16 End Compound Line 17 Begin Compound Text Path 18 End Compound Text Path 19 Begin Tile Array 20 End Tile Array	CGM Version 2 and 3 elements. Ignored.
CGM Class 1 orders: Metafile Descriptor elements	
1 Metafile Version	Value checked (only Version 1, 2, or 3 CGM permitted, but all Version 2 and 3 elements are ignored).
2 Metafile Description	Text returned as second part of CGLOAD *descriptor1* parameter.
3 VDC Type	Equivalent.
4 Integer Precision	Equivalent.
5 Real Precision	Equivalent.
6 Index Precision	Equivalent.
7 Color Precision	Equivalent.
8 Color Index Precision	Equivalent.
9 Maximum Color Index	Equivalent.

Table 36 (Page 2 of 3). CGM order processing

CGM code and order name	Comments
10 Color Value Extent	Equivalent.
11 Metafile Element List	Ignored.
12 Metafile Defaults Replacement (metafile elements)	Equivalent.
13 Font List	Equivalent. Up to 32 font names of up to 32 characters each.
14 Character Set List	Ignored.
15 Character Coding Announcer	Ignored.
16 Name Precision 17 Maximum VDC Extent 18 Segment Priority Extent 19 Color Model 20 Color Calibration 21 Font Properties 22 Glyph Mapping 23 Symbol Library List	CGM Version 2 and 3 elements. Ignored.
CGM Class 2 orders: Picture Descriptor elements	
1 Scaling Mode	Equivalent.
2 Color Selection Mode	Equivalent. See “Color Index” on page 330 for more details.
3 Line Width Specification Mode	Equivalent. GDDM accepts only Absolute and Scaled modes. Other modes are rejected as errors.
4 Marker Size Specification Mode	Equivalent. GDDM accepts only Absolute and Scaled modes. Other modes are rejected as errors.
5 Edge Width Specification Mode	Equivalent. GDDM accepts only Absolute and Scaled modes. Other modes are rejected as errors.
6 VDC Extent	Equivalent.
7 Background Color	Equivalent.
8 Device Viewport 9 Device Viewport Spec. Mode 10 Device Viewport Mapping 11 Line Representation 12 Marker Representation 13 Text Representation 14 Fill Representation 15 Edge Representation 16 Interior Line Spec. Mode 17 Line and Edge Type Definition 18 Hatch Style Definition 19 Geometric Pattern Definition	CGM Version 2 and 3 elements. Ignored.
CGM Class 3 orders: Control elements	
1 VDC Integer Precision	Equivalent.
2 VDC Real Precision	Equivalent.

Table 36 (Page 2 of 3). CGM order processing

CGM code and order name	Comments
3 Auxiliary Color	Ignored.
4 Transparency	Equivalent.
5 Clip Rectangle	Equivalent.
6 Clip Indicator	Equivalent.
7 Line Clipping Mode 8 Marker Clipping Mode 9 Edge Clipping Mode 10 New Region 11 Save Primitive Context 12 Restore Primitive Context 17 Protection Region Indicator 18 Generalized Text Path Mode 19 Mitre Limit 20 Transparent Cell Color	CGM Version 2 and 3 elements. Ignored.
CGM Class 4 orders: Graphical Primitive elements	
1 Polyline	Equivalent.
2 Disjoint Polyline	Equivalent.
3 Polymarker	Equivalent.
4 Text	Equivalent.
5 Restricted Text	Equivalent, using basic restriction and (if appended) using the final text attributes for the complete text string
6 Append Text	Equivalent.
7 Polygon	Equivalent.
8 Polygon Set	Equivalent.
9 Cell Array	Ignored.
10 Generalized Drawing Primitive	Ignored.
11 Rectangle	Equivalent.
12 Circle	Equivalent.
13 Circular Arc 3-point	Equivalent.
14 Circular Arc 3-point Close	Equivalent.
15 Circular Arc Center	Equivalent.
16 Circular Arc Center Close	Equivalent.
17 Ellipse	Equivalent.
18 Elliptical Arc	Equivalent.
19 Elliptical Arc Close	Equivalent.
20 Circular Arc Center Reversed 21 Connecting Edge 22 Hyperbolic Arc 23 Parabolic Arc 24 Non-Uniform B-Spline 25 Non-Uniform Rational B-Spline 26 Polybezier 27 Bitonal Tile 28 Tile 29 Polysymbol	CGM Version 2 and 3 elements. Ignored.
CGM Class 5 orders: Attribute elements	

Table 36 (Page 3 of 3). CGM order processing

CGM code and order name	Comments
1 Line Bundle Index	Implemented as per CALS defaults in Table XI of the MIL-D-28003A specification.
2 Line Type	Equivalent.
3 Line Width	Equivalent.
4 Line Color	Equivalent.
5 Marker Bundle Index	Implemented as per CALS defaults in Table XI of the MIL-D-28003A specification.
6 Marker Type	Equivalent.
7 Marker Size	Equivalent.
8 Marker Color	Equivalent.
9 Text Bundle Index	Implemented as per CALS defaults in Table XI of the MIL-D-28003A specification.
10 Text Font Index	Equivalent.
11 Text Precision	Ignored.
12 Character Expansion Factor	Equivalent.
13 Character Spacing	Equivalent.
14 Text Color	Equivalent.
15 Character Height	Equivalent.
16 Character Orientation	Equivalent.
17 Text Path	Equivalent.
18 Text Alignment	Equivalent (except for “continuous text alignment,” which is ignored.)
19 Character Set Index 20 Alternate Character Set Index	Ignored.
21 Fill Bundle Index	Implemented as per CALS defaults in Table XI of the MIL-D-28003A specification.
22 Interior Style	Equivalent (except for “geometric pattern” and “interpolated”, which are treated as “solid”).
23 Fill Color	Equivalent.
24 Hatch Index	Equivalent. Maps to GDDM patterns
25 Pattern Index	Equivalent.
26 Edge Bundle Index	Implemented as per CALS defaults in Table XI of the MIL-D-28003A specification.
27 Edge Type	Equivalent.
28 Edge Width	Equivalent.
29 Edge Color	Equivalent.
30 Edge Visibility	Equivalent.
31 Fill Reference Point	Ignored.
32 Pattern Table	Ignored.
33 Pattern Size	Ignored.
34 Color Table	Order converted under certain conditions. See “Color Index” on page 330 for more details.
35 Aspect Source Flags	Ignored.
36 Pick Identifier 37 Line Cap 38 Line Join 39 Line Type Continuation 40 Line Type Initial Offset 41 Text Score Type 42 Restricted Text Type 43 Interpolated Interior 44 Edge Cap 45 Edge Join 46 Edge Type Continuation 47 Edge Type Initial Offset 48 Symbol Library Index 49 Symbol Color 50 Symbol Size 51 Symbol Orientation	CGM Version 2 and 3 elements. Ignored.
CGM Class 6 orders: Escape element	
1 Escape	Ignored.
CGM Class 7 orders: External elements	
1 Message	Ignored.
2 Application Data	Ignored.
CGM Class 8 orders: Segment elements	
1 Copy Segment 2 Inheritance Filter 3 Clip Inheritance 4 Segment Transformation 5 Segment Highlighting 6 Segment Display Priority 7 Segment Pick Priority	CGM Version 2 and 3 elements. Ignored.

Chapter 14. Graphics Interchange Format (GIF) files

The graphics interchange format (GIF) is a commercial format for interchanging graphical information between computer systems. GIF files contain bitmap (graphics) data that is compressed using the LZW (Lempel-Ziv Welch) compression algorithm. The files can be expanded and viewed by using a variety of graphical application programs at a user's workstation. GIF is the main format used to include images on World Wide Web (WWW) Internet pages.

GIF file structure

GIF allows for more than one color image to be defined in a file. It has the concept of a logical screen area, within which all the images are displayed. Each image is offset by a number of pixels from the top left-hand corner of the logical screen. Any part of the logical screen not covered by an image is filled with the background color (as specified by the GIF file or the GIF viewer itself).

GIF images are a sequence of color numbers (one for each pixel). The color numbers are in the range 0 to 255 and are used as an index into a color table to get the real RGB value for that pixel. GIF also has the concept of a transparency color. This has the effect of not displaying that color (so the background color shows through). Only one color number may be specified as the transparency color. The color table also forms part of the GIF.

ADMUGIF

ADMUGIF is a GDDM utility that runs under VM/CMS and MVS/TSO. It converts an ADMGDF file to a GIF file.

ADMUGIF needs parameters to identify the input ADMGDF file, the output GIF file name, and keyword parameters that affect the format of the GIF.

The input ADMGDF file (fixed or floating point) is converted to GIF, and the output is saved in a file in ASCII binary format. If a file with the same name exists, it is overwritten (without issuing any warning message).

GIF files produced by ADMUGIF have only one color image and have a logical screen size that is the same as the image size. So, in effect, offsets for the color image are redundant, and background color is only seen through transparent pixels (if the GIF viewer does not force its own background color).

The color table placed in the GIF is derived from the default GDDM color table used originally in GDDM 3.1.1 for PostScript support. See Table 31 on page 332 for mapping color numbers to real RGB values.

The values can be changed by using the ADMMCLTB or COLORTAB UDS (see the *GDDM System Customization and Administration* book).

ADMUGIF uses GDDM color separation facilities and, during its execution, creates eight color separation files (which have a maximum size of 1MB). Also, to encode the color image, ADMUGIF allocates virtual storage of about 2MB. All allocated files and storage are freed after execution finishes.

Some GIF viewers or decoders are for specific GIF versions. GIF files produced by ADMUGIF are version 87a. If a transparency color other than -1 is used, they are version 89a.

Keyword parameters

The keyword parameters are as follows (abbreviations shown in uppercase):

Width The range is 8 through 1024; the default is 400. GDDM rounds up the number you specify to a multiple of 8 pixels.

Depth The range is 8 through 1024; the default is 400.

The Width and Depth values specify the maximum size, in pixels, of the output GIF image.

If a Width value is specified without a Depth, the Depth is set to the Width. Similarly, the Width is set to the Depth if only a Depth value is specified.

Aspect The range is 0 through 2; the default is 1.

- **0** The GIF image exactly fits the Width and Depth area. The aspect ratio of the original ADMGDF file is **not** maintained. (In effect, the original ADMGDF is stretched to fit the Width and Depth area of the GIF.)
- **1** The GIF image is produced in the same aspect ratio as the original ADMGDF. The GIF image may **not** fill the specified Width and Depth area. The GIF image has an actual width **or** depth that is less than the values specified in the Width and Depth keyword parameters.
- **2** The GIF picture is produced in the same aspect ratio as the original ADMGDF. The GIF image fills the Width and Depth area. This may mean that parts of the GIF image are 'unused' and default to black image (color number 0).

Transcol The range is -2 through 255; the default is -1. Specifies the transparency color number, where:

-2 Transparency is the color of the left-most edge of the image.

-1 No transparency.

0-255 Transparency color, as specified by the color table.

Backcol The range is -2 and 0 through 255; the default is 0. Specifies the background color and is output in the GIF file but, as described above, may have no effect.

-2 Swaps white and black so that the output GIF has a white background with black lines.

Keyword parameter values that are out of range are set to the highest or lowest values, as applicable.

Invoking ADMUGIF under VM/CMS

The ADMGDF file name is required. The GIF file name and keyword parameters are optional.

```
ADMUGIF gdf-file <gif-file> <( keyword-parameters>
```

where:

```
gdf-file = gdf-filename <gdf-filetype <gdf-filemode>>
gif-file = <gif-filename <gif-filetype<gif-filemode>>>
```

The defaults are:

gdf-filetype ADMGDF
gdf-filemode A
gif-filename Same as gdf-filename
gif-filetype GIFBIN
gif-filemode A

Example VM/CMS invocations

```
ADMUGIF ADMTEST
```

Gets an ADMGDF file called ADMTEST ADMGDF (the first one in the CMS disk search order) and converts it to a GIF output file called ADMTEST GIFBIN A (of width 400 pixels and depth 400 pixels, depending on the aspect ratio).

```
ADMUGIF ADMTEST GIFOUT
```

Gets an ADMGDF file called ADMTEST ADMGDF (the first one in the CMS disk search order) and converts it to a GIF output file called GIFOUT GIFBIN A (of width 400 pixels and depth 400 pixels, depending on the aspect ratio).

Note: After ADMUGIF, two names (with no other values) are treated as filenames. For example, ADMUGIF ADMTEST ADMGDF produces an output GIF file called ADMGDF GIFBIN.

```
ADMUGIF ADMTEST GDF GIFOUT GIF
```

Gets an ADMGDF file called ADMTEST GDF (the first one in the CMS disk search order) and outputs the corresponding GIF file called GIFOUT GIF A (of width 400 pixels and depth 400 pixels, depending on the aspect ratio).

```
ADMUGIF ADMTEST GDF T GIFOUT GIF B
```

Gets an ADMGDF file called ADMTEST GDF from the T disk and outputs the corresponding GIF file called GIFOUT GIF to the B disk (of width 400 pixels and depth 400 pixels, depending on the aspect ratio). ADMUGIF expects that any output filemode used for the GIF output file will be accessed in write mode.

```
ADMUGIF ADMTEST ( Width 800 Depth 800
```

Creates a GIF file called ADMTEST GIFBIN of approximately 800 by 800 pixels, depending on the aspect ratio of the GDF).

```
ADMUGIF ADMTEST ( W 800 D 800 B 1 T 2 A 0
```

Creates a GIF file called ADMTEST GIFBIN of exactly 800 by 800 pixels, with the picture stretched to fit the area and having a background color of 1 (blue) and a transparency color of 2 (red).

Invoking ADMUGIF under MVS/TSO

The following allocations are needed before invoking ADMUGIF:

```
ALLOC F (ADMGDF)   DSN(gdf-pds-dataset-name)
ALLOC F (ADMSYMBL) DSN(symbolset-pds-name)
```

where gdf-pds-dataset contains all the ADMGDF files that are to be converted and symbolset-pds contains ADMDHIPK, which is needed for color image separation processing.

The ADMGDF file name is required. The GIF file name and keyword parameters are optional.

```
CALL 'dataset-name (ADMUGIF)'
'From(gdf-file) <To(gif-file)> <keyword-parms>'
```

where:

gdf-file Is the name of a member within the partitioned data set defined by the ddname ADMGDF.

gif-file Is either an allocated ddname or a data set name to which a data set qualifier is added to the front (in the usual TSO way). If you do not enter a gif-file name, the name defaults to:

```
'TSO-prefix.gdf-file.GIFBIN'
```

| **keyword-parms** Parentheses are required around the parameter values: for example, Width(200).

| ADMUGIF can also be used in batch under the MVS/TSO command processor IKJEFT01. Here is some sample JCL:

```
| //ADMUGIF  JOB
| //IKJEFT01 EXEC PGM=IKJEFT01,REGION=3M
| //ADMSYMBL DD DSN=GDDM.SADMSYM,DISP=SHR
| //ADMGDF   DD DSN=GDDM.SADMGDF,DISP=SHR
| //SYSTSPRT DD SYSOUT=*
| //SYSPRINT DD SYSOUT=*
| //SYSTSIN  DD *
|  CALL 'GDDM.SADMMOD(ADMUGIF)' 'FROM(ADMTEST)'
| /*
```

| The userid running the batch job must be defined for TSO logon and have a valid TSO PREFIX value assigned.

| Example MVS/TSO invocations

```
| CALL 'dataset-name(ADMUGIF)' 'F(ADMTEST)'
```

| Gets an ADMGDF file called ADMTEST from the partitioned data set defined by ddname ADMGDF and converts it to a GIF sequential output file called TSO-prefix.ADMTEST.GIFBIN.

```
| CALL 'dataset-name(ADMUGIF)' 'F(ADMTEST) T(GIFOUT)'
```

| Gets an ADMGDF file called ADMTEST from the partitioned data set defined by ddname ADMGDF and converts it to a GIF sequential output file called TSO-prefix.GIFOUT.

```
| CALL 'dataset-name(ADMUGIF)' 'F(ADMTEST)
|   T(GIFOUT) W(800) T(0) A(2)'
```

| Creates a GIF output file that will best fit an 800 by 800 pixel output area and have a transparency color of 0 (black).

| For both VM/CMS and MVS/TSO

| The following return codes and notes apply to both VM/CMS and MVS/TSO.

| Return codes

| **100** Too many (>6) name-parts: '(' not found when expected

| **101** Invalid option: unexpected character string after '('

| **102** Option value not numeric

| **103** Keyword >8 chars or value in ( ) too long (TSO only)

| **104** FSFRCE failed

| **105** GSLOAD failed

| **106** File open failed

| **107** File read failed

| **108** File close failed

| **109** ESEUDS failed

| **110** Allocate storage failed

| **111** DSOPEN failed

| **Other:** Return code from failed FSINIT or FSTERM

| **Notes:**

| 1. Text quality: GDDM graphics text mode 1 and mode 2 are not sized correctly because of the way the image is reduced by using a GDDM viewport. Mode 3 text is sized correctly but may be indistinct for small output GIF files.
| 2. MVS Batch invocation needs to run under IKJEFT01 and run with a userid that has a valid TSO profile prefix.
| 3. ADMMCLTB or COLORTAB for color number 0 should be used to change the background color of a converted ADMGDF file.
| 4. GIF is a service mark of Compuserve Inc.

Chapter 15. Format of a Composite Document Presentation Data Stream

This chapter describes the structure of a Composite Document Presentation Data Stream (CDPS) document that can be processed by the Composite Document Print Utility (CDPU). A list of the Advanced Function Presentation Data Stream (AFPDS) structured fields supported by the CDPU is given in “AFPDS structured fields supported by the CDPU” on page 356.

The physical organization of the file varies depending on the environment:

CICS Temporary data file

VSE Batch ESDS data set

MVS Batch V-format sequential data set

TSO V-format sequential data set

CMS V-format sequential file.

In each case each record contains a complete structured field. The structured fields must be in the order shown, except where it is stated that the order is optional.

Structured fields are described in detail below.

Document structure

In the syntax structure below, the following conventions apply:

::= Precedes the definition of an item

[ ] Square brackets indicate optional items

. . . The item may be repeated.

document::=
begin-document
[invokable-master-environment-group] . . .
[page] . . .
end-document

The file cannot contain multiple documents. Anything after the end-document structured field is ignored.

The formats of individual structured fields, such as “begin-document” and “end-document”, are defined in the next section, under the heading “Structured field formats” on page 349.

invokable-master-environment-group::=
begin-master-environment-group
[medium-descriptor]
[medium-modification-control] . . . (up to two)
[medium-copy-count]
[map-medium-overlay]
[page-descriptor]
[page-position]
end-master-environment-group

page::=
[master-environment-group-invocation] . . .
begin-page
[active-environment-group]
[presentation-text-object]
[image-object] . . .
[graphics-object] . . .
[bar-code-object] . . .
end-page

Note: The presentation-text-object, image-object, graphics-object, and bar-code-object may occur in any order.

master-environment-group-invocation::=
begin-master-environment-group
invoke-master-environment-group
end-master-environment-group

active-environment-group::=
begin-active-environment-group
[map-coded-font]
[page-descriptor]
[page-position]
[object-area-descriptor]
[object-area-position]
[presentation-text-descriptor]
[object-area-position]
end-active-environment-group

presentation-text-object::=
begin-presentation-text
[presentation-text-data] . . .
end-presentation-text

graphics-object::=
begin-graphics-object
begin-object-environment-group
object-area-descriptor
object-area-position
[map-coded-font]
[graphics-data-descriptor]
end-object-environment-group
[graphics-data] . . .
end-graphics-object

bar-code-object::=
begin-bar-code-object
begin-object-environment-group
object-area-descriptor
object-area-position
[map-bar-code]
[map-coded-font]
bar-code-data-descriptor
end-object-environment-group
[bar-code-data]...
end-bar-code-object

image-object::=
begin-image-object
begin-object-environment-group
object-area-descriptor
object-area-position

[image-data-descriptor]
end-object-environment-group
[image-picture-data] . . .
end-image-object

In addition, **no-operation** structured fields may appear anywhere in the document and are ignored.

Structured fields

A document consists of a sequence of structured fields, each of which has the following format:

0 – 1 Length of the structured field in bytes. This is the length of the parameters specific to the type of structured field, plus the 8-byte introducer. In no case may the length of a structured field be more than 8200. (This differs from AFPDS documents, for which the maximum is 8202.)

2 – 4 String identifying the type of structured field. The hexadecimal value for each type is shown in the heading for each structured field; see "Structured field formats" on page 349.

5 – 7 X'000000'.

8 – n Parameter information as described for each structured field under "Structured field formats" on page 349.

Offsets, for example in error messages, are shown in hexadecimal and are calculated from the start of the structured field, including the two length bytes.

Summary of structured fields

Table 37. Structured fields in code order

Hex code	Meaning	
D3A66B	Object area descriptor	OBD
D3A688	Medium descriptor	MDD
D3A69B	Presentation text descriptor	PTD
D3A6AF	Page descriptor	PGD
D3A6BB	Graphics data descriptor	GDD
D3A6EB	Bar code data descriptor	BDD
D3A6FB	Image data descriptor	IDD
D3A788	Medium modification control	MMC
D3A89B	Begin presentation text	BPT
D3A8A8	Begin document	BDT
D3A8AF	Begin page	BPG
D3A8BB	Begin graphics object	BGR
D3A8C7	Begin object environment group	BOG
D3A8C8	Begin master environment group	BMG
D3A8C9	Begin active environment group	BAG
D3A8EB	Begin bar code object	BBC
D3A8FB	Begin image object	BIM
D3A99B	End presentation text	EPT
D3A9A8	End document	EDT
D3A9AF	End page	EPG
D3A9BB	End graphics object	EGR
D3A9C7	End object environment group	EOG
D3A9C8	End master environment group	EMG
D3A9C9	End active environment group	EAG
D3A9EB	End bar code object	EBC
D3A9FB	End image object	EIM
D3AB8A	Map coded font/2	MCF/2
D3ABDF	Map medium overlay	MMO
D3ABEB	Map bar code	MBC
D3AC6B	Object area position	OBP
D3AFC8	Invoke master environment group	IMG
D3B188	Medium copy count	MCC
D3B1AF	Page position	PGP
D3EE9B	Presentation text data	PTX
D3EEBB	Graphics data	GAD
D3EEEB	Bar code data	BDA
D3EEEE	No operation	NOP
D3EEFB	Image picture data	IPD

Table 38. Structured fields in alphabetic order

	Meaning	Hex code
BAG	Begin active environment group	D3A8C9
BBC	Begin bar code object	D3A8EB
BDA	Bar code data	D3EEEB
BDD	Bar code data descriptor	D3A6EB
BDT	Begin document	D3A8A8
BGR	Begin graphics object	D3A8BB
BIM	Begin image object	D3A8FB
BMG	Begin master environment group	D3A8C8
BOG	Begin object environment group	D3A8C7
BPG	Begin page	D3A8AF
BPT	Begin presentation text	D3A89B
EAG	End active environment group	D3A9C9
EBC	End bar code object	D3A9EB
EDT	End document	D3A9A8
EGR	End graphics object	D3A9BB
EIM	End image object	D3A9FB
EMG	End master environment group	D3A9C8
EOG	End object environment group	D3A9C7
EPG	End page	D3A9AF
EPT	End presentation text	D3A99B
GAD	Graphics data	D3EEBB
GDD	Graphics data descriptor	D3A6BB
IDD	Image data descriptor	D3A6FB
IMG	Invoke master environment group	D3AFC8
IPD	Image picture data	D3EEFB
MBC	Map bar code	D3ABEB
MCC	Medium copy count	D3B188
MCF/2	Map coded font/2	D3AB8A
MDD	Medium descriptor	D3A688
MMC	Medium modification control	D3A788
MMO	Map medium overlay	D3ABDF
NOP	No operation	D3EEEE
OBD	Object area descriptor	D3A66B
OBP	Object area position	D3AC6B
PGD	Page descriptor	D3A6AF
PGP	Page position	D3B1AF
PTD	Presentation text descriptor	D3A69B
PTX	Presentation text data	D3EE9B

Structured field formats

For each of the structured fields below, offsets are shown within the parameter information section which starts at byte 8 in the structured field.

Begin active environment group (D3A8C9) BAG: Indicates the beginning of an active environment group.

0 – 7 Active environment group name (0 through 8 characters).

Begin bar code object (D3A8EB) BBC: Indicates the beginning of a bar code object.

0 – 7 Data-object name (0 through 8 characters)

Bar codes can be printed on IPDS printers and viewed on graphics displays. They are not supported on page printers.

GDDM does not check the suitability of the bar-code font ID, module width, element height, height multiplier, or wide-to-narrow ratio for the target IPDS printer. Unsuitable values cause an error message to be issued and can cause the printer to stop.

When a bar code is viewed on a display, GDDM shows a generalized and stylized picture of the bar code, rather than an accurate representation. The following defaults are used if requested by the application:

Module width	014-inch	
Element height	UPCA, UPCE, EAN13	1-inch
	3-of-9, 2-of-5, MSI	¼-inch
	others	⅞-inch
Wide-to-narrow ratio	2.5 : 1	

Bar code data (D3EEEB) BDA: Contains data parameters for positioning, encoding, and presenting a bar code symbol in the bar code object presentation space. The data to be presented is encoded in accordance with the parameter definitions in the Bar-code-data descriptor (BDD) structured field.

0 Flags to control the presence and position of the human-readable interpretation (HRI).

- B'0.....' The HRI is printed.
- B'1.....' The HRI is not printed.
- B'.00....' The printer default is used for positioning the HRI.
- B'.01....' The HRI is printed below the bar code symbol.
- B'.10....' The HRI is printed above the bar code symbol.
- B'...0....' No asterisk is printed as the HRI for 3-of-9 code start and stop.
- B'...1....' An asterisk is printed as the HRI for 3-of-9 code start and stop.

1 – 2 x coordinate of the top left corner of the bar code symbol (in the range 1 through 32767).

3 – 4 y coordinate of the top left corner of the bar code symbol (in the range 1 through 32767).

5 – n bar coded data.

The coordinates in bytes 1-2 and 3-4 are expressed using the measurement units defined in the BDD structured field. The format and length of the data is determined by the bar code type specified in the BDD (see Table 39).

Table 39. Format of bar code data

Type	Length	Data
3-of-9	variable	Alphanumeric characters
MSI	variable	Numeric characters
UPC/CGPC-Version A	11	Numeric characters
UPC/CGPC-Version E	10	Numeric characters (5th to 9th must be 0)
UPC 2-digit supplement	2	Numeric characters
UPC 5-digit supplement	5	Numeric characters
EAN 8 (include JAN short)	7	Numeric characters
EAN 13 (includes JAN standard)	12	Numeric characters
2-of-5	variable	Numeric characters
EAN 2-digit add-on	2	Numeric characters
EAN 5-digit add-on	5	Numeric characters
Note: Numeric characters are in EBCDIC from X'FO' through X'F9'.		

Bar-code-data descriptor (D3A6EB) BDD: Specifies the size of the bar-code-object presentation space, the bar code type, and various parameters and attributes for presenting bar code symbols within the object.

0 – 1 X'0000'
2 – 3 Number of measurement units in 10 inches in the x-direction
4 – 5 Number of measurement units in 10 inches in the y-direction
6 – 7 Presentation-space size in the x-direction
8 – 9 Presentation-space size in the y-direction
10 – 11 Reserved
12 – 13 Bar code type and modifier

The following types are supported.
X'01' 3-of-9 code, with modifier X'01'–'02'
X'02' MSI, with modifier X'01'–X'09'
X'03' UPC/CGPC-Version A, with modifier X'00'
X'05' UPC/CGPC-Version E, with modifier X'00'
X'06' UPC 2 digit supplement, with modifier X'00'

X'07' UPC 5 digit supplement, with modifier X'00'
X'08' EAN-8, with modifier X'00'
X'09' EAN-13, with modifier X'00'
X'0A' 2-of-5 industrial, with modifier X'01'–X'02'
X'0B' 2-of-5 matrix, with modifier XX'01'–X'02'
X'0C' Interleaved 2-of-5, with modifier X'01'–X'02'
X'16' EAN 2-digit add-on, with modifier X'00'
X'17' EAN 5-digit add-on, with modifier X'00'

14 Local font identifier

This allows control of the font to be used for any Human Readable Information (HRI) associated with the bar code. Setting it to X'FF' causes the printer to use the default font appropriate to the particular bar code type. A value in the range X'00' through X'7F' matching the Resource Local Identifier in the preceding MCF/2 causes the IPDS font specified by that MCF/2 to be used for the HRI.

The Resource Local Identifier on the MCF/2 must not clash with that used on any other MCF/2.

Printers may impose restrictions on the font that may be used with a particular bar code type. It is therefore recommended that you use the default X'FF'.

The UPC and EAN bar code types can only be printed with OCR-B.

15 – 16 Color identifier

X'0000' or X'FF00'	Printer default
X'0001' or X'FF01'	Blue
X'0002' or X'FF02'	Red.
X'0003' or X'FF03'	Magenta (pink)
X'0004' or X'FF04'	Green
X'0005' or X'FF05'	Turquoise (cyan)
X'0006' or X'FF06'	Yellow
X'0008'	Black
X'0010'	Brown
X'FF07'	Printer default
X'FF08'	Color or medium

17 Module width

X'FF' Printer default
Others The width in thousandths of an inch of the smallest defined bar code dimension

18 – 19 Element height

X'FFFF' Printer default height
Others The height of bar and space elements in units of 1/1440 of an inch.

20 Height multiplier

The number of vertically contiguous, identical bar and space patterns printed in a bar code symbol.

21 – 22 Wide-to-narrow ratio

The ratio of the wide element to the narrow element dimension.

X'FFFF'	Printer default
X'0014'–X'00IE'	2.0 – 3.0 in units of 0.1
X'00C8'–X'012C'	2.00 – 3.00 in units of 0.01

This does not apply to UPC/CGPC, EAN, JAN, or UPC codes.

Note: Printers impose restrictions on the values accepted in bytes 17 through 22. These restrictions also depend on the bar code type (byte 12). Check your printer documentation before using non-default values.

Begin document (D3A8A8) BDT: Indicates the beginning of the document. It contains the following fields:

0 – 7 Document name.

8 – 9 X'0000'.

10 – n Groups of optional, additional information, in any order. These groups are reserved to describe the level of function in the document.

Begin graphics object (D3A8BB) BGR: Indicates the beginning of a graphics object.

0 – 7 Data Object name (0 through 8 characters).

Begin image object (D3A8FB) BIM: Indicates the beginning of an image object.

0 – 7 Image name (0 through 8 characters).

Begin master environment group (D3A8C8) BMG: Indicates the beginning of a master environment group (MEG). This may be either an invokable MEG, or an invocation of such a MEG.

0 – 7 Master environment group name (0 through 8 characters).

Begin object environment group (D3A8C7) BOG: Indicates the beginning of an object environment group.

0 – 7 Object environment group name (0 through 8 characters).

Begin page (D3A8AF) BPG: Indicates the beginning of a page.

0 – 7 Page name (0 through 8 characters).

Begin presentation text (D3A89B) BPT: Indicates the beginning of a presentation text object.

0 – 7 Data object name (0 through 8 characters).

End active environment group (D3A9C9) EAG: Indicates the end of an active environment group.

0 – 7 Active environment group name (0 through 8 characters).

End bar code object (D3A9EB) EBC: Indicates the end of a bar code object.

0 – 7 Data-object name (0 through 8 characters).

End document (D3A9A8) EDT: Indicates the end of the document.

0 – 7 Document name (0 through 8 characters).

End graphics object (D3A9BB) EGR: Indicates the end of a graphics object.

0 – 7 Data object name (0 through 8 characters).

End image object (D3A9FB) EIM: Indicates the end of an image object.

0 – 7 Image name (0 through 8 characters).

End master environment group (D3A9C8) EMG: Indicates the end of a master environment group.

0 – 7 Master environment group name (0 through 8 characters).

End object environment group (D3A9C7) EOG: Indicates the end of an object environment group.

0 – 7 Object environment group name (0 through 8 characters).

End page (D3A9AF) EPG: Indicates the end of a page.

0 – 7 Page name (0 through 8 characters).

End presentation text (D3A99B) EPT: Indicates the end of a presentation text object.

0 – 7 Data object name (0 through 8 characters).

Graphics data (D3EEBB) GAD: Contains the graphics orders to be drawn.

0 – n Up to 8192 bytes of graphics data. This, combined with successive graphics data structured fields if required, contains one or more complete graphics segments. It must not have drawing orders outside segments.

The format is a sequence of orders suitable for processing by GSPUT, with the following exceptions:

- Segment start is one of two formats.
  - The longer of the two forms defined in Chapter 10, “GDF order descriptions” on page 281.
  - An extended form of the above. The second byte contains X'0E' as the length of following data. The length of segment field contains the low-order 2 bytes of the length of segment. Two extra bytes at the end contain the high-order bytes.

  In each case the length of segment must be specified exactly, and is not assumed to end when a X'FF' order code is met.
- Segment end is not accepted as indicating the end of a segment, and is ignored. Instead, the length given in the segment start order is used to show the position of the end.

Coordinates must match the format specified in the graphics data descriptor.

A graphic order may span successive graphics data structured fields that make up an object.

Graphics data descriptor (D3A6BB) GDD: Specifies the limits of coordinates in the graphics data. It contains one or two groups of data as follows:

- Drawing order subset, optional.

  0 X'F7'.
  1 Length of following data.
  2 – n Reserved.
- Window specification, required.

  0 X'F6'
  1 Length of following data
  2 – 3 X'0000'
  4 Format of coordinates:
  X'00' 2 byte integers
  X'01' 4 byte floating-point
  5 X'00' Reserved.
  6 – 11 (or 6 – 17 if floating-point coordinates) Reserved
  12 – 13 (or 18 – 21 if floating-point coordinates) x-coordinate of left edge in graphics data
  14 – 15 (or 22 – 25 if floating-point coordinates) x-coordinate of right edge in graphics data
  16 – 17 (or 26 – 29 if floating-point coordinates) y-coordinate of bottom edge in graphics data
  18 – 19 (or 30 – 33 if floating-point coordinates) y-coordinate of top edge in graphics data
  20 – 23 (or 34 – 41 if floating-point coordinates) Reserved.

The graphics data is drawn scaled to fit the object area specified in the object area descriptor and object area position fields. If the object area is partly off the page, the object is clipped at the page boundary.

Preservation of aspect ratio or size can be achieved by appropriate selection of parameters when creating this file.

Image data descriptor (D3A6FB) IDD: Specifies the size of the image to be included.

0 X'00'
1 – 2 Number of pixels in 10 inches in the x-direction
3 – 4 Number of pixels in 10 inches in the y-direction
5 – 6 Image size in the x-direction
7 – 8 Image size in the y-direction.

The image is drawn without scaling, and is trimmed to fit the object area specified in the object area descriptor and object area position fields. If the object area is partly off the page, the image is trimmed at the page boundary.

Invoke master environment group (D3AFC8) IMG: This indicates a change for the current state master environment group (MEG) parameters. It contains the following field:

0 – 7 The name of the invokable MEG whose parameters are to become the current state MEG values.

Image picture data (D3EEFB) IPD: Contains the image orders to be drawn.

0 – n Up to 8192 bytes of image data. It must be in a format suitable for processing by IMAPT. The contents of a sequence of image picture data structured fields must be a valid sequence that would follow a call to IMAPTS specifying default compression and a format value of –2. This implies that the image data must follow the convention that **1 = black**.

Map bar code (D3ABEB) MBC: Specifies how the bar code data object is to be positioned in an output area of the logical page.

0 – 4 X'0005030400'

This specifies the only valid option - position, no trim or scale. The top-left corner of the bar code object presentation space is mapped to the object-content origin that is specified in the object-area position (OBP) structured field.

This is the default action if this structured field is omitted.

Medium copy count (D3B188) MCC: Specifies medium modification group references.

0 – 4 X'0001000100'.
5 Medium modification group reference for the front of the paper. It must match the group identifier in a medium modification control structured field.
6 Medium modification group reference for the reverse of the paper, applicable to duplex printing. It must match the group identifier in a medium modification control structured field, or is zero for simplex printing.

Map coded font (D3AB8A) MCF/2: Identifies the correspondence between external font names and a resource local identifier. It consists of a repeating group for each font. Each has the following fields:

0 – 1 Length of this repeating group
2 – n Groups of additional information, in any order, as follows:

- Fully qualified name, required

  0 – 1 X'0C02'– identifies the group
  2 Type of name as follows:
  X'84' Coded font name (GRID)
  X'85' Code page name
  X'86' Font name
  3 X'00' Reserved
  4 – 11 External name of the font.

  For a text map-coded font, one of the following is required:

  – a coded font name (GRID)
  – both a code page name and a font name.

  For a graphics map-coded font, one of the following is required:

  – a coded font name (GRID)
  – a font name.

  For both text and graphics DBCS fonts, a coded font name (GRID) is required.

  Code page information is ignored for graphics map-coded fonts.

  The external name of the font (byte 4 through 11) takes one of the following forms:

  – Coded font name (X'84'–global resource identifier or GRID):

    4 – 5 A graphic character-set global identifier (GCSGID).
    6 – 7 A code-page global identifier (CPGID).
    8 – 9 A font global identifier (FGID).
    10 – 11 A 2-byte character-width field. This is the width of the space character in 1/1440 inch units.

  – Font name (X'86')

    For a presentation-text map-coded font (MCF/2) the external name is an 8 byte PSF member name. For a graphics-text MCF/2, it must be a GDDM symbol-set name.

- Resource local identifier, required.

  0 – 2 X'042405' – identifies the group.
  3 Resource local identifier. It must be in the range 1 through 127 for a text font. When used in a graphics object the value is as follows:
  0 pattern or marker symbol set
  65 – 223 other symbol sets.

- Font descriptor, optional

  0 – 1 X'0D1F' – identifies the group.
  2 Font weight class. It must be in the range 1 through 9.
  3 Font width class. It must be in the range 1 through 9.
  4 – 5 Font vertical point size. It must be in the range 0 through 360.
  6 – 7 Average character width. It must be in the range 0 through 360.
  8 Font descriptor flags, assigned as follows:
  X'80' Italic
  X'40' Underscored
  X'10' Outline characters
  X'08' Overstruck
  X'04' Proportional spaced.
  These flags are ignored by GDDM.
  9 Font usage. This is defined in a graphics object only, and is a reserved byte for a text font. The values 1–5, 8, and 9 correspond to

the type parameter on a call to GSLSS. Other values are reserved.

10 Font family – reserved.

11 Font class – reserved.

12 Font quality.

Font descriptor for IPDS printers

GDDM uses the following interpretation of the font descriptor when driving IPDS printers:

1. Font weight class

   X'07' Bold characters (when supported by printer)

   other values Normal characters.

2. Font width class

   X'07' Double wide characters (when supported by printer)

   other values Normal characters.

3. Font descriptor flags

   X'80' Italic characters (when supported by printer)

   X'08' Overstruck characters (when supported by printer).

4. Font quality

   X'01' Low quality, high speed (when supported by printer)

   X'02' Medium quality, medium speed (when supported by printer)

   X'03' High quality, low speed (when supported by printer).

   GDDM changes font quality only when it changes pages. For consistent results, all fonts used on any one page should all have the same font quality.

Medium descriptor (D3A688) MDD: Specifies the size of the medium. The default is the size of the target device known to GDDM. It contains the following fields:

0 – 1 X'0000'.

2 – 3 Number of medium measurement units in 10 inches in the x-direction. It must be in the range 2400 through 14400.

4 – 5 Number of medium measurement units in 10 inches in the y-direction. It must be in the range 2400 through 14400.

6 – 8 Medium size in the x-direction (in the range 1 through 8388607).

9 – 11 Medium size in the y-direction (in the range 1 through 8388607).

Medium modification control (D3A788) MMC: Specifies the modifications of a medium copy group.

0 Medium modification group identifier, in the range 1 through 127.

1 X'FF'.

2 – n Modifications, in any order, from the list below. Each type of modification may be specified at most once.

- First source location selector.

  0 X'E1'. Keyword identifier.

  1 Source selection for the first form in the group. It must be in the range 1 through 100. The default is 1.

- Subsequent source location selector.

  0 X'E2'. Keyword identifier.

  1 Source selection for subsequent forms in the group. It must be in the range 1 through 100. The default is 1.

- Medium overlay local identifier.

  0 X'F2'. Keyword identifier.

  1 Local identifier for the overlay required. It must match the local identifier in a map medium overlay structured field. The default is no overlay.

Map medium overlay (D3ABDF) MMO: Identifies the correspondence between an external overlay name and a resource local identifier. It contains one or two groups, each specifying such a pair. If a second group is included, it applies to the reverse of the paper in duplex printing. The format of each group is:

0 – 1 X'0012'.

2 – 17 Two groups of additional information, in any order, as follows:

- Fully qualified name, required.

  0 – 3 X'0C028400'. Identifies the group.

  4 – 11 External name of the overlay.

- Resource local identifier, required.

  0 – 2 X'042401'. Identifies the group.

  3 Resource local identifier. It must be in the range 1 through 127.

No operation (D3EEEE) NOP: This may be used to add comments to the data stream. It can appear in any position.

0 – n Up to 8192 bytes of comment data, not examined.

Object area descriptor (D3A66B) OBD: Specifies the size of an object. It contains the following fields:

0 – n Three groups of additional information, in any order, as follows:

- Descriptor position identifier, required.

  0 – 1 X'0343'. Identifies the group.

  2 Descriptor position identifier in range 1 through 127.

- Object area measurement units, required.

  0 – 3 X'084B0000'. Identifies the group.

4 – 5 Number of object area units in 10 inches in the x-direction. It must be in the range 2400 through 14400.
6 – 7 Number of object area units in 10 inches in the y-direction. It must be in the range 2400 through 14400.

- Object area size, required.

  0 – 2 X'094C02'. Identifies the group.
  3 – 5 Object area size in the x-direction (in the range 1 through 8388607).
  6 – 8 Object area size in the y-direction (in the range 1 through 8388607).

The information for the text object area descriptor must match that of the page descriptor, which is the default if this field is missing.

Object area position (D3AC6B) OBP: Specifies the position of an object on the page.

0 Object area position identifier in range 1 through 127.
1 X'17'.
2 – 4 Object area origin, x (in the range 0 through 8388607).
5 – 7 Object area origin, y (in the range 0 through 8388607).
8 – 11 Object orientation. The orientations are respectively:
X'00002D00' North
X'2D005A00' East (permitted for bar code objects only)
X'5A008700' West (permitted for bar code objects only)
X'87000000' South (permitted for bar code objects only).
12 X'00'
13 – 15 Object content origin, x (in the range 0 through 8388607).
16 – 18 Object content origin, y (in the range 0 through 8388607).
19 – 23 X'00002D0000'.

The text object area position, if present, must have both origins zero (the default). The origins are expressed in object measurement units, as defined in the OBD structured field.

Parts of a graphics or image object that lie outside the page size are not printed.

Note: Non-zero object rotation may only be used for bar code objects. Non-zero object rotation is not supported by all IPDS printers.

Page descriptor (D3A6AF) PGD: Specifies the size of the page, which must not be zero and must fit on the GDDM device size at the position given by the page-position structured field. The default is found from the size in the medium descriptor.

0 – 1 X'0000'.
2 – 3 Number of page measurement units in 10 inches in the x-direction. It must be in the range 2400 through 14400.
4 – 5 Number of page measurement units in 10 inches in the y-direction. It must be in the range 2400 through 14400.
6 – 8 Page size in the x-direction (in the range 1 through 8388607).
9 – 11 Page size in the y-direction (in the range 1 through 8388607).

Page position (D3B1AF) PGP: Specifies the position of the page on the form. The defaults are zero. It contains the following fields:

0 – 1 X'0109'.
2 – 4 Page origin in the x-direction (in the range 0 through 8388607).
5 – 7 Page origin in the y-direction (in the range 0 through 8388607).
8 – 9 Page orientation. The orientations are respectively:
X'0000' North
X'2D00' East
X'5A00' South
X'8700' West.

Note: The page origin is ignored for family-4 devices if it is found within an active environment group.

Presentation text descriptor (D3A69B) PTD: Gives the size of the text block on the page, and may specify defaults to be used for text controls. The fields are as follows:

0 – 1 X'0000'.
2 – 3 Number of text measurement units in 10 inches in the x-direction. It must be in the range 2400 through 14400.
4 – 5 Number of text measurement units in 10 inches in the y-direction. It must be in the range 2400 through 14400.
6 – 8 Text block size in the x-direction, the same as the size in the Page Descriptor.
9 – 11 Text block size in the y-direction, the same as the size in the Page Descriptor.
12 – 13 Presentation text flags.
14 – n Optional groups of additional information, in any order, specifying initial defaults in place of printer defaults. The format is the same as bytes 2 – n of the presentation text data structured field. The subset that may be included in the presentation text descriptor structured field is as follows:

- Set baseline increment
- Set coded font local
- Set intercharacter increment
- Set inline margin
- Initial addressable position (absolute move baseline and absolute move inline)
- Set text color.

Presentation text data (D3EE9B) PTX: This contains a chain of text controls. It contains the following fields:

0 – 1 X'2BD3'.
2 – n A sequence of text controls as described below. Byte 1 of the last text control contains the value shown minus 1, to mark the end of the chain.

- Absolute move baseline.

  Sets the distance from the top margin of the paper.

  0 – 1 X'04D3'.
  2 – 3 Baseline print position.

- Absolute move inline.

  Sets the distance from the left margin of the paper.

  0 – 1 X'04C7'.
  2 – 3 Inline print position.

- Begin line.

  Sets the print position for a new line. See also **set baseline increment** and **set inline margin**.

  0 – 1 X'02D9'.

- Draw baseline rule.

  Draws a rule perpendicular to the top of the page.

  0 – 1 X'07E7'.
  2 – 3 Length. If the length is positive, the line is drawn in the sequential baseline direction. If the length is negative, the line is drawn away from the sequential baseline direction.
  4 – 5 Width. If the width is positive, pixels are added in the positive inline direction. If the width is negative, pixels are added in the negative inline direction.
  6 X'00'.

- Draw inline rule.

  Draws a rule parallel to the top of the page.

  0 – 1 X'07E5'.
  2 – 3 Length. If the length is positive, the line is drawn in the sequential inline direction. If the length is negative, the line is drawn away from the sequential inline direction.
  4 – 5 Width. If the width is positive, pixels are added in the positive baseline direction. If the width is negative, pixels are added in the negative baseline direction.
  6 X'00'.

- No operation.

  Used to include variable length comments, or to terminate chaining (by changing byte 1 to X'F8').

  0 – 1 X'xxF9'. xx denotes the length of the comment.
  2 – (xx+1) Variable length comment.

- Relative move baseline.

  Used to change the current baseline position.

  0 – 1 X'04D5'.
  2 – 3 Baseline print position move. It can be positive or negative.

- Relative move inline.

  Used to change the current inline position.

  0 – 1 X'04C9'.
  2 – 3 Inline print position move. It can be positive or negative.

- Repeat string.

  Specifies the repetition of a character string.

  0 – 1 X'xxEF'. xx denotes the length of the string to be repeated.
  2 – 3 Number of characters to be repeated. For example, if the resulting length is 7 and the string is **ABC**, then **ABCABCA** is printed.
  4 – (xx+3) Variable length string to be repeated.

  For example, to generate the string ABCABCA the field would be:

  ```
  03EF0007ABC
  ```

- Set baseline increment.

  Sets the baseline movement to be used by the begin line control.

  0 – 1 X'04D1'.
  2 – 3 Baseline increment amount.

- Set coded font local.

  Identifies the coded font to be used for printing subsequent text.

  0 – 1 X'03F1'.
  2 Local identifier of the coded font to be used. It must match one of those in a map coded font structured field.

- Set inline margin.

  Sets the inline margin to be used by the begin line control.

  0 – 1 X'04C1'
  2 – 3 Inline margin.

- Set intercharacter increment.

  Used to set intercharacter spacing.

  0 – 1 X'04C3'
  2 – 3 Increment.

- Set text color.

  Specifies the color of text that follows.

  0 Length of set text color control, either 4 or 5.
  1 X'75'.
  2 – 3 Foreground color.
  4 If present, indicates the color precision.

- Set text orientation.

  Only one text orientation may be specified, although the whole page may be rotated.

0 – 5 X'06F700002D00'.

- Set variable space character increment.

  Establish the width of blank characters that appear in transparent data controls.

  0 – 1 X'04C5'.
  2 – 3 Width of variable space character.

- Transparent data.

  Identifies text to be printed that does not contain any text controls.

  0 – 1 X'xxDB'. xx denotes the length of the text.
  2 – (xx+1) Variable length text to be printed.

AFPDS structured fields supported by the CDPU

The Composite Document Print Utility (CDPU) supports printing and viewing of a document, a page segment or an overlay in Advanced Function Presentation Data Stream (AFPDS) format. AFPDS formats supported by the CDPU include LIST3820, LIST38PP, LIST4250, LISTAPA, PSEG38PP, PSEG4250, OVLY38PP, and OVLY4250.

The following restrictions apply to CDPU support of AFPDS documents:

- The AFPDS document cannot contain multiple documents or page segments. (A document can contain page segments inline.)
- The structured field length must not exceed 8202 bytes.
- Input formatted for the 4250 can be viewed but not printed.
- Secondary input is not supported.
- Page segments that contain text cannot be printed unless they are imbedded in a document.

GDDM supports the AFPDS enhancements provided by Print Services Facility (PSF) Version 2.1 on VM and MVS systems. These include support for GOCA graphics orders and IO compressed image.

The formats of individual structured fields, such as “begin-document”, are defined in the *AFPDS Data Stream Reference*, S544-3202.

Summary of AFPDS structured fields supported by the CDPU

AFPDS structured fields supported by the CDPU are listed in Table 40.

The include-page-segment structured field is not supported and is treated as an error. Structured-field-introducer (SFI) extensions (bit 0, flag byte 5) are not supported and are treated as errors. Padding bytes (bit 4, flag byte 5 of SFI) are not supported.

Table 40. AFPDS structured fields supported by the CDPU

Hex code	Meaning
D3A66B	Object area descriptor (OBD)
D3A67B	Image input descriptor (IID)
D3A69B	Composed/Presentation text descriptor - format 1 (CTD/PTD)
D3A6AF	Page descriptor (PGD)
D3A6BB	Graphics data descriptor (GDD)
D3A6FB	Image date descriptor (IDD)
D3A77B	Image output control (IOC)
D3A79B	Composed text control (CTC)
D3A85F	Begin page segment (BPS)
D3A87B	Begin image (IM) object (BIM)
D3A89B	Begin composed/presentation text object (BCT/BPT)
D3A8A8	Begin document (BDT)
D3A8AF	Begin page (BPG)
D3A8BB	Begin graphics object (BGR)
D3A8C7	Begin object environment group (BOG)
D3A8C9	Begin active environment group (BAG)
D3A8DF	Begin medium/multifunction overlay (BMO)
D3A8FB	Begin image (IO) object (BIM)
D3A95F	End page segment (EPS)
D3A97B	End image (IM) block (EIM)
D3A99B	End composed/presentation text (ECT/EPT)
D3A9A8	End document (EDT)
D3A9AF	End page (EPG)
D3A9BB	End graphics object (EGR)
D3A9C7	End object environment group (EOG)
D3A9C9	End active environment group (EAG)
D3A9DF	End medium/multifunction overlay (EMO)
D3A9FB	End image (IO) object (EIM)
D3AB8A	Map coded font - format 2 (MCF/2)
D3ABBB	Map graphics object (MGO)
D3AC6B	Object area position (OBP)
D3AC7B	Image cell position (ICP)
D3B18A	Map coded font - format 1 (MCF/1)
D3B19B	Presentation text descriptor - format 2 (PTD)
D3EE7B	Image raster data (IRD)
D3EE9B	Composed/Presentation text data (CTX/PTX)
D3EEBB	Graphics data (GAD)
D3EEEE	No operation (NOP)
D3EEFB	Image picture data (IPD)
D3A67B	Image input descriptor (IID)
D3A69B	Composed text descriptor (CTD)
D3A6AF	Page descriptor (PGD)
D3A77B	Image output control (IOC)
D3A79B	Composed text control (CTC)
D3A85F	Begin page segment (BPS)
D3A87B	Begin image block (BIM)
D3A89B	Begin composed text block (BCT)
D3A8A8	Begin document (BDT)
D3A8AF	Begin page (BPG)
D3A8C9	Begin active environment group (BAG)
D3A8DF	Begin medium overlay (BMO)
D3A95F	End page segment (EPS)
D3A97B	End image block (EIM)
D3A99B	End composed text block (ECT)
D3A9A8	End document (EDT)
D3A9AF	End page (EPG)
D3A9C9	End active environment group (EAG)
D3A9DF	End medium overlay (EMO)
D3AC7B	Image cell position (ICP)
D3B18A	Map coded font (MCF)
D3EE7B	Image raster data (IRD)
D3EE9B	Composed text data (CTX)

Chapter 16. Application data structure for mapping

The basic purpose of the application data structure is to define an input/output area for use in transferring data between your application program and GDDM. You include the application data structure declaration created by GDDM-Interactive Map Definition (GDDM-IMD) in your application to define the layout of one or more areas of storage. GDDM also keeps a copy in its own storage of the data area associated with each mapped field that you define, and it uses its copy to create the display that the operator sees, and to record the changes made by the operator.

Your program modifies the GDDM data area by filling in values in its own area, then passing the area to GDDM using an MSPUT call. It finds out the values in the GDDM data area by using an MSGET call, which copies the GDDM area into the program's data area. Usually, MSGET is used so that the program gets access to the operator's input, though it can be used at other times; for example, after MSDFLD, to initialize the program's data area to the default values.

When you have finished the GDDM-IMD map-definition and generation processes, you will not only have one or more generated mapgroups, but you will also have an application data structure for each map. The data structure and the fields that it defines depends on the selections made during the map-definition process. Further information on this process is given in the *GDDM Interactive Map Definition* book.

The application data area can be used for these purposes:

- Most of an application data structure is data fields, each data field corresponding to a map-defined display field. You place into the data fields the character data that you want to be displayed.
- You can position the cursor in a display field by setting the field's **cursor adjunct**. By default, the cursor is placed under the first character of the field, but you can change this by using the MSCPOS call before you use MSPUT.
- **Selector adjuncts** provide additional control over, and information about, a field's data value. You can selectively update a field, reset a field to its map-defined default value, and determine whether a field has been modified by the operator.
- **Length Adjuncts** show the length of the data in the field. If the data in a field is shorter than the map-defined display field length, GDDM pads the data with nulls when it displays the field. After operator input the length adjunct is set to the number of characters provided by the operator.
- Usually, field attributes are specified for the various fields on a map during map definition. However, at run time the application program can change these attributes by placing attribute values in **attribute adjunct fields** in the application data structure. One or more adjunct fields can be associated with a given data field in the application data structure during map definition. Each attribute adjunct controls a different type of attribute.
- Some devices allow different attributes to be applied to individual characters in the same field. Character attributes are controlled using a separate copy of the application data area. The data fields in this copy contain the character attribute data instead of the normal character data. Each character in the character attribute data area determines the attributes of the corresponding character in the normal application data area.
- The application program can be designed to allow detection (or selection) of fields in a displayed panel by a light pen or, on some devices, the Cursor Select (CURSR SEL) key. The type of detection that occurs is determined by the first data character in the field; this character is called a **designator character**.
- If specified in the map during map definition, GDDM edits input data entered by the terminal operator. To process this edited input, you need to know how GDDM presents it in the application data structure.

This chapter gives valid settings and explanations of adjunct fields, character attributes, and designator characters, and describes the format of edited input. It also describes how to copy the application data structure into the application program.

Adjunct fields

Each data field in the application data structure may have associated adjunct fields, depending on the options selected during the Field Naming step of map definition. The possible adjunct fields for a data field are shown below. They appear in the data structure in the order given, immediately before the data field.

Adjunct	Length (bytes)
Selector	1
Cursor	1
Base attribute	2
Extended highlighting	2
Color	2
Programmed symbols (PS)	2
Validation	2
Outlining	2
Length	2

The base attribute, extended highlighting, color, PS, validation, and outlining adjunct fields shown above are each subdivided into two one-byte fields. In each case, the first byte acts as a selector to let GDDM know whether or not the value held in the second byte is to be used during program execution.

COBOL example

Suppose that in the Map Characteristics frame (2.1) of GDDM-IMD, you entered:

```
PROGRAM LANGUAGE ==> COBOL
```

Next, suppose that in the Application Structure Review frame (2.5), you are defining the characteristics of a data field that you have named SPECNAME. You want to be able to:

1. Set the cursor in the field under application program control
2. Have dynamic control of extended highlighting
3. Specify the length of data in the field.

You therefore enter "#HL" in the ADJUNCT column against the field name.

As a result of this entry, the application data structure contains, for the field SPECNAME, a cursor adjunct (1 byte), a highlighting adjunct (2 bytes), a length adjunct (2 bytes), plus the data field itself, whose length is as defined in the map (say 25 bytes).

GDDM-IMD names the adjunct fields by suffixing the data field name supplied by the user. So, for example, the cursor adjunct field is named SPECNAME-CURSOR.

The portion of the application data structure that is generated for SPECNAME is:

```
10  SPECNAME-CURSOR      PIC X.
10  SPECNAME-HI-SEL      PIC X.
10  SPECNAME-HI          PIC X.
10  SPECNAME-LENGTH      PIC 999 COMP.
10  SPECNAME             PIC X(25).
```

Assembler language example

Assume that instead of entering COBOL as the program language in the above example, you enter ASM, and, to comply with Assembler-language length restrictions, you name the data field SPEC. The generated code (assuming the other selections were the same as those given above) is:

```
SPECCR    DS    X
SPECHS    DS    X
SPECH     DS    X
SPECL     DS    AL2
SPEC      DS    XL25
```

PL/I example

Similarly, if you use PL/I as the program language and call the data field SPECNAME, the generated code is:

```
10 SPECNAME_CURSOR CHAR(1),
10 SPECNAME_HI_SEL CHAR(1),
10 SPECNAME_HI CHAR(1),
10 SPECNAME_LENGTH FIXED BIN(15),
10 SPECNAME CHAR(25),
```

Adjunct field names

The above examples show that GDDM-IMD suffixes the name you have given to a data field to create unique names for each adjunct field in the application data structure. The full set of suffixes that GDDM-IMD uses for COBOL, Assembler, and PL/I data structures is shown in Table 41 on page 358.

Adjunct values

Table 42 on page 359 summarizes valid settings for adjunct fields. Information for each type of adjunct is given in the following text.

Table 41. Adjunct field naming conventions

Adjunct	Length	COBOL name	Assembler name	PL/I name
Selector	1	XXX-SEL	XXXS	XXX_SEL
Cursor	1	XXX-CURSOR	XXXCR	XXX_CURSOR
Base attribute	1 1	XXX-ATTR-SEL XXX-ATTR	XXXAS XXXA	XXX_ATTR_SEL XXX_ATTR
Extended highlighting	1 1	XXX-HI-SEL XXX-HI	XXXHS XXXH	XXX_HI_SEL XXX_HI
Color	1 1	XXX-COL-SEL XXX-COL	XXXCS XXXC	XXX_COL_SEL XXX_COL
PS	1 1	XXX-PS-SEL XXX-PS	XXXPS XXXP	XXX_PS_SEL XXX_PS
Validation	1 1	XXX-VAL-SEL XXX-VAL	XXXVS XXXV	XXX_VAL_SEL XXX_VAL
Outlining	1 1	XXX-OUT-SEL XXX-OUT	XXXOS XXXO	XXX_OUT_SEL XXX_OUT
Length	2	XXX-LENGTH	XXXL	XXX_LENGTH

Table 42 (Page 1 of 2). Values used in adjunct fields

Adjunct	Value (See Note 1)	Meaning
Selector	C' '	MSPUT: Any data value is ignored. The field is unchanged (see Note 2).
Selector	C' '	MSGET: Neither the application nor the operator has put a value in it.
Selector	C'1'	MSPUT: The field contains a value.
Selector	C'1'	MSGET: The field contains a value that the operator has just modified.
Selector	C'2'	MSPUT: The field is to be reset to its map-defined default value. The data value is ignored. On a subsequent MSGET, the field contains its default value and the selector is C'3'.
Selector	C'2'	MSGET: not used.
Selector	C'3'	MSPUT: The field contains a value. (that is, the same as C'1').
Selector	C'3'	MSGET: The field contains a value that has not just been modified by the operator.
Cursor	C' '	MSPUT: The cursor is not in this field.
Cursor	C' '	MSGET: The cursor is not in this field.
Cursor	C'1'	MSPUT: The cursor is in this field.
Cursor	C'1'	MSGET: The cursor is in this field (set only if map is a cursor rece iver).
The cursor position within the field can be controlled by using the MSCPOS call, and verified by using the MSQPOS call.		
Attribute Selector (first byte of adjunct)	C' '	The attribute is unchanged (see Note 2). The attribute byte (the second byte) is ignored.
Attribute Selector (first byte of adjunct)	C'1'	Change the attribute to the value in the second byte.
Attribute Selector (first byte of adjunct)	C'2'	Reset the attribute to the map-defined default value.
Attribute Selector (first byte of adjunct)	C'3'	Change the attribute to the value in the second byte (same as C'1'). After an MSGET, the attribute selector is set to C'3' and the attribute byte set to the current attribute value.
Attribute Value (second byte of attribute adjunct)	Ignored unless the attribute selector is C'1' or C'3'. Otherwise, the valid value depends on the attribute type, as follows: C' ' Default for all attributes. X'00' Default for all attributes.	
Base attribute	A valid 3270 attribute if used. These values are defined mnemonically in ADMUAIMC (Assembler), ADMUCIMC (COBOL), and ADMUPIMC (PL/I). **For example:** C' ' Unprotected C'H' Unprotected, Intensified C'-' Protected C'Y' Protected, Intensified C'0' Autoskip B'xx.....' Ignored (set by GDDM) B'..1.....' Protected B'..0.....' Unprotected B'...0....' Alphanumeric B'..01....' Unprotected numeric B'..11....' Autoskip B'....00..' Normal B'....01..' Selectable B'....10..' Intensified selectable B'....11..' Nondisplay B'......x.' Ignored (set by GDDM) B'.......1' Modified data tag set B'.......0' Modified data tag not set	

Table 42 (Page 2 of 2). Values used in adjunct fields

Adjunct	Value (See Note 1)	Meaning
Extended high-lighting attribute	C' '	No extended highlighting.
Extended high-lighting attribute	C'1'	Blinking.
Extended high-lighting attribute	C'2'	Reverse video.
Extended high-lighting attribute	C'4'	Underscore.
Color attribute	C' '	Default.
Color attribute	C'1'	Blue.
Color attribute		
Color attribute	C'2'	Red.
Color attribute	C'3'	Magenta (pink).
Color attribute	C'4'	Green.
Color attribute	C'5'	Turquoise (cyan).
Color attribute	C'6'	Yellow.
Color attribute	C'7'	White/Neutral.
PS attribute	C' '	Default character set.
PS attribute	X'41'–X'DF'	PS code of any symbol set specified in PS Set Management in GDDM-IMD, or loaded using PSDSS, PSLSS, or PSLSSC.
Validation attribute	C' '	No validation.
Validation attribute	X'00'	No validation.
Validation attribute	X'01'	Trigger.
Validation attribute	X'02'	Mandatory enter.
Validation attribute	X'04'	Mandatory fill.
Validation attributes can be ORed together to give two or more validation attributes to the same field. For example, specify X'03' to give a field the mandatory enter and trigger attributes (X'02' OR X'01'=X'03').		
Outlining attribute	C' '	No outlining.
Outlining attribute	X'00'	No outlining.
Outlining attribute	X'01'	Underline.
Outlining attribute	X'02'	Vertical line on right.
Outlining attribute	X'04'	Overline.
Outlining attribute	X'08'	Vertical line on left.
Outlining attributes can be ORed together to give two or more outlining attributes to the same field. For example, specify X'03' to give a field with underlining and a vertical line on the right (X'02' OR X'01'=X'03').		
Length	Binary value	Length, in characters, of the data.

Notes:

1. In this table, C'c' indicates character data type, X'x' indicates hexadecimal data type, and B'bbbbbbbb' indicates a binary pattern.
2. On an MSPUT call with option 0 (“WRITE”), all fields and attributes are reset to their map-defined default value, before the application data area is processed. Therefore, an attribute selector or field selector of C' ' has the net effect of resetting the value to default, when used on an MSPUT call with option 0, or of leaving the value unchanged, when used on an MSPUT call with option other than zero.

The application program sets the values required for a send request. GDDM sets the values associated with input data returned for a receive request. On a send request, each field **must** contain one of the settings given for it in Table 42.

Selector adjunct: The selector adjunct provides additional control over an individual field in the application data structure, and shows, after an MSGET, whether the data field has just been modified by the operator.

The control function is most useful when using MSPUT with option 1 (REWRITE) or 2 (REJECT), particularly if the application program does not maintain a complete copy of the application data area. A partially completed application structure can be used. Fields whose selector is blank are ignored and so need not be set by the application. Their value is unchanged. Fields whose selector is C'1' or C'3' are processed by placing the current data value in GDDM's copy of the data area with the value from the program's data area.

Note: Fields that do not have a selector are always processed.

The control function can also be used to set a field to its map-defined default value. This is the constant text placed into the field during GDDM-IMD's Field Definition or Field Initialization steps. This is the value of the field immediately after a mapped field is defined by MSDFLD. Note that if the field has no selector, any MSPUT call replaces this default value with the value from the application data area (even if the field is all blanks). If the field has a selector adjunct, its value can be reset to the map-defined default value by specifying a selector of C'2' on any MSPUT call.

Notes:

1. The map-defined default character attributes are always "default." GDDM-IMD does not support character attributes.
2. An MSPUT with option 0 (WRITE) sets all fields (attributes and so on) back to their default value before processing the application data area.

When a selector value of C'2' is specified, GDDM converts it into C'3', and places the default field value into the data field in GDDM's copy of the data area so that the program can access it using MSGET. (The program's data area is **not** modified during an MSPUT.)

After an MSGET, a selector adjunct shows whether the field has just been modified by the operator. A value of C'1' shows that the field has been modified by one of these events:

- The operator has typed into the field
- The operator has selected the field with a light-pen (if the field is selectable)
- The field has been set by AID translation.

Note: "Modified" includes the degenerate case of the operator modifying the field back to its original value.

Usually, modification indicators are reset when the operator is next given an opportunity to enter data (for example, an ASREAD). Your program can avoid this resetting by issuing an MSPUT call with option 2 (REJECT) on any map within the page, before the ASREAD call.

Cursor adjunct: The cursor adjunct is used to set the cursor in a field dynamically (thus overriding any static cursor setting specified in the map), and to show whether the cursor was left in a field on input.

Static setting of the cursor is specified in the Field Attribute Definition step of GDDM-IMD; for further information, see the *GDDM Interactive Map Definition* book.

To set the cursor in a field dynamically, the application program sets the associated cursor adjunct to 1. This causes the cursor to be placed in the field when the field is displayed. By default, the cursor is placed under the first character of the field. To position the cursor elsewhere, use the MSCPOS call to specify the position, just before issuing the MSPUT call. The position is a number between 0 and the length of the field, thus:

0 Means "under the attribute byte"
1 Means "under the first character"
2 Means "under the second character" and so on.

When the position specified is greater than the length of the field, GDDM places the cursor under the last character in the field.

GDDM places the cursor at the last dynamic setting it meets for a page. In the absence of any dynamic settings, GDDM places the cursor at the first static setting.

To determine the position of the cursor on input, the map must have been defined as a cursor-receiver map in the Map Characteristics step of map definition. If the map has been so defined, GDDM sets the cursor adjunct of the field in which the cursor lies to C'1' when the field has a cursor adjunct. The position of the cursor within the field can be found using the MSQPOS call **after** the MSGET call.

Note: The "cursor-receiver" map characteristic is provided so that applications that use cursor adjuncts only for output cursor control do not have to search for, and turn off, cursor adjuncts after an MSGET call. If the cursor adjuncts were left on, GDDM might misinterpret the application's intention when the application data area is next used in an MSPUT call.

Attribute adjuncts: An Attribute adjunct is used to change the attribute of a field from its map-defined default value. There are several types of attribute adjuncts; one "base" attribute that controls a compound set of basic field properties, and one attribute type for each of a set of "extended" properties.

Each attribute adjunct field consists of two subfields; an attribute selector byte, and an attribute value byte. The valid values for the attribute selector are the same for all attribute adjuncts (and the same as those for a Field Selector):

C' ' Ignore the value provided. Leave the attribute at its current value. If the mapped field has just been defined, or if the operator is an MSPUT with option 0 (WRITE), the current value is the map-defined value. Otherwise, the value is that set by previous MSPUT operations.

C'1' Change the attribute to that specified in the attribute adjunct.

C'2' Reset the attribute to the map-defined value.

C'3' Change the attribute to that specified in the attribute adjunct (that is, the same as C'1'). After an MSGET, all attribute adjunct selectors are set to C'3', and the attribute value byte is set to the current attribute value.

The second byte of an attribute adjunct is the value to be used (for selector value C'1' and C'3'). The range of valid values is dependent on the attribute type.

GDDM provides, as part of GDDM-IMD, a set of declarations in Assembler, COBOL, and PL/I, for the values that can be used in attribute adjuncts. These are in the files ADMUAIMC (Assembler), ADMUCIMC (COBOL) and ADMUPIMC (PL/I) in the GDDM Sample Library.

Note that all attribute adjunct types can be used on all devices supported by GDDM for mapping, but they have no effect on the presentation if the device does not support the corresponding function.

Base attribute adjunct: Base attributes are the basic (as opposed to extended) field attributes that are supported by all display devices supported by GDDM. They can be specified for individual fields on a map during map definition and reset during program execution by base-attribute adjuncts. They include:

- Protected/unprotected/autoskip
- Intensified-display/normal-display/nondisplay
- Detectable/nondetectable
- MDT bit set/reset
- Alphanumeric/numeric.

The attribute adjunct value byte can contain any valid IBM 3270 basic attribute code. GDDM sets the reserved and meaningless bits of the attribute correctly, so all one-byte values are accepted.

The base attribute adjunct value byte completely specifies the combination of base attributes to be used for the field on the device. It is not merged in any way with previous base-attribute specifications for the field, or with the value specified in the associated map.

Extended highlighting adjunct: The extended highlighting adjunct can be used by the application program to override any extended highlighting attribute defined for a field in the map. Extended highlighting is available only on specific devices, and can be used in addition to the intensification control of the base attribute. It lets you specify whether a field should blink, be underscored, or be displayed in reverse video.

Possible settings in the attribute adjunct value byte are as shown in Table 42 on page 359.

Color adjunct: Possible settings in the attribute adjunct value byte are again as shown in Table 42 on page 359.

Note that this adjunct cannot be used to control color on devices whose color is determined by means other than the color extended attribute. For example, it can be used to control color on seven-color display devices, but not on four-color display devices.

Programmed symbols adjunct: The programmed symbols (PS) adjunct lets you specify that the special characters and symbols defined in a given symbol set apply for the field associated with the PS adjunct in the application data structure. You can define your own symbol sets using the Image Symbol Editor, as described in the *GDDM Using the Image Symbol Editor* book. You can also use the predefined symbol sets supplied by IBM.

Your application program can use characters from a particular symbol set only if that symbol set is loaded into a PS store in the device. A symbol set can be loaded when defining a mapgroup containing maps that use symbol sets. You can specify that the symbol sets are to be loaded automatically by GDDM when a MSPCRT request naming the mapgroup is issued; for more information, see the *GDDM Interactive Map Definition* book. Symbol sets loaded in this way are available to the application program for the life of the page.

The required symbol set is identified by the PS code (or PSID), which is a single-character identifier in the range X'41' through X'DF', designated by you or your installation. The PS code is designated when the mapgroup is defined, if GDDM is to handle symbol-set loading, or during the loading operation, if your application program or your installation is handling symbol-set loading directly.

Validation adjunct: The validation attribute is supported only by the IBM 8775 Display Terminal (with the appropriate feature). On all other devices it is ignored.

Possible settings in the attribute adjunct value byte are as shown in Table 42 on page 359. The IBM 8775 handles operator input according to the validation attribute, as follows:

1 Mandatory Enter Attribute
If the operator tries to transmit data (for example, by pressing the ENTER key) while there is a mandatory enter field that has not had data entered into it, the

transmission fails and input is inhibited. The cursor is repositioned to the start of the first empty mandatory enter field. The operator can proceed by pressing the RESET key. Then, the operator can either enter data in the mandatory enter field, or use the ERASE EOF or Error Override key to set the MDT. For the Error Override key, an error value (X'3F') is returned to the application program in the mandatory enter field.

2 Mandatory Fill Attribute
If data is entered into a mandatory fill field, the field must be completely filled before the cursor can be moved out of it. If an attempt is made to move the cursor out of the field before it has been filled, further input is inhibited.

The operator can proceed by pressing the RESET key, and completing the entry of data into the mandatory fill field. Or, the Error Override key can be used to fill the field with error values (X'3F') before continuing.

3 Trigger Fill Attribute
The trigger field attribute enables the application program to receive data entered into a particular field as soon as the data entry for that field is complete and the cursor leaves the field. The operator can continue keying data while the trigger field is being checked, but the data entered is placed on a queue in the device (and is not displayed).

Cursor exit from a modified trigger field causes the inbound transmission of this single field with a “trigger” AID. The application can access the trigger field data in the usual way using MSGET.

The application program must then decide whether to accept the trigger field (and hence the operator’s queued keystrokes) by issuing a positive acknowledgment, or to reject the field (and lose the operator’s queued keystrokes) by issuing a negative acknowledgment.

A positive acknowledgment is generated by issuing an MSPUT call specifying that the keyboard is to be unlocked. By default, this is true of options 0 (WRITE) and 1 (REWRITE).

A negative acknowledgment is generated by issuing an MSPUT call specifying that the keyboard is to remain locked. By default, this is true of option 2 (REJECT).

Note: The relationship between the MSPUT option and locking the keyboard is defined in GDDM-IMD’s Map Characteristics step.

Field outlining: Outlining is only available on specific devices; if the device does not support outlining the adjunct is ignored. Possible settings in the attribute adjunct byte are shown in Table 42 on page 359.

Length adjunct: The length adjunct is a two-byte field that can contain values in the range 0 through the length of the field. It indicates the length of the data in the data field. GDDM treats a value greater than the field length as if it were equal to the field length.

When a field is displayed, GDDM pads the data with nulls, from the length specified in the length adjunct, to the length of the display field.

After the operator modifies a field, the length adjunct specifies the number of bytes of data placed in this field by the input operation.

If right-hand justification has been specified for the field during map definition, the length adjunct is set on input to the length of the field in the application data structure. If left-hand justification has been specified, the length adjunct is set to the number of characters in the field up to the first padding character.

Character attributes

Highlighting, color, and PS attributes can be specified for individual characters within a field. Usually, character attributes are used to emphasize a particular character string in a field.

Note: GDDM supports character attributes in mapped variable fields, but not in constant or initial values held in the map.

To control any type of character attribute, the program needs an additional application data area. This area has the same structure as the usual application data area (including adjunct fields), but the data fields are interpreted as character attributes rather than character data.

To declare several data areas using the same structure, you can use an array of structures or (in PL/I) the LIKE attribute.

COBOL

```
01  ALLAREAS.
 02  DATA-AREA OCCURS 3 TIMES.
    COPY MAP.
```

PL/I

```
Declare
    1 DATA_AREA,
       %INCLUDE MAP;
Declare
    1 COLOR_AREA LIKE DATA_AREA;
```

In the former case, the individual application areas (and fields and adjuncts within them) can be referred to using an array index. In the second case, they can be referenced using name qualification (DATA_AREA.FIELD1, COLOR_AREA.FIELD1, and so on).

The character attribute data areas are filled in the same way as are the usual application data areas, except that the data

fields contain characters representing attributes. For example:

```
DATA_AREA.FIELD1 ='data value';
COLOR_AREA.FIELD1='1111111121';
```

Adjunct fields in the character attribute application data area have the same meaning as in the normal data area. Selector and Length adjuncts apply to the character attribute data field.

Each application data area is passed to GDDM with a separate MSPUT call. The character attribute type is specified as an option on MSPUT. The character attributes should be MSPUT **after** the data values, because changing the data value of any field automatically resets the character attributes of the field to the default value (C' '). Also, an MSPUT with option 0 (WRITE) resets all the character attributes of all fields in the map to default.

The allowable attribute types and attribute values are listed in Table 43. GDDM checks attribute types and does not transmit those that the device does not support. Invalid attribute values are rejected.

Table 43. Character attribute types and values

Type	Value	Meaning
All	X'00' C' '	Default. Take the attribute value from field's attribute.
Extended highlighting	C'1' C'2' C'4'	Blinking Reverse video Underscore
Color	C'1' C'2' C'3' C'4' C'5' C'6' C'7'	Blue Red Magenta (pink) Green Turquoise (cyan) Yellow White/Neutral
Programmed symbols	X'41' ⋮ X'DF'	PS code. Note that a symbol-set must be loaded before any reference to it is made. See "Programmed symbols adjunct".

Note: In the above table, C'c' indicates character data type, and X'x' indicates hexadecimal.

Setting character attributes from the terminal

If the application program uses the ASMODE call and an appropriate keyboard is in use, the terminal operator can set the attributes of data characters entered from the terminal. The program can read these attributes using MSGET with the correct option.

The procedure for setting character attributes from the terminal can be found in the appropriate terminal operator's guide.

Designator characters for light-pen or cursor selection

You specify that a field can be selected by a light pen, or, on some terminals, the CURSR SEL key, by giving it a "detectable" attribute at map-definition time. The "detectable" attribute can be defined for a field using GDDM-IMD's Field Attribute Definition step, and can be controlled dynamically using the base attribute adjunct.

However, the type of selection that occurs on using the light pen is determined by the first character (the designator character) in the data field. You must set the required designator character in the first byte of the data field. If the field contains constant data, the designator character is set in the map; otherwise, it is set in the application data structure. When the field is displayed, the designator character appears on the screen along with the rest of the data in the field.

A field having a "detectable" attribute but not starting with a valid designator character is not selectable.

The types of selection that can be set are:

1. Delayed detection. When selected by the operator, the field is marked as "modified" but nothing is transmitted until the operator performs another action associated with field modification (such as selecting an "immediate detection" field or pressing ENTER). The designator character for this type of field is "?" (X'6F'). If the field is detected, the designator character changes to ">" (X'6E'); another detection restores it to "?" and cancels the modification indication.
2. Immediate detection without data. The designator character is a blank (X'40'). Selection of this type of field causes immediate input transmission. No data from any of the fields is transmitted, however. The effect is thus:
   a. The ASREAD (or MSREAD) returns an Attention Type of 2 indicating light-pen selection.
   b. If the application issues an MSGET, any field that was modified or delay-detected has its selector set to C'1'; its data value, however, is unchanged even if the operator typed into the field.
   c. GDDM restores all display fields to their original value at the next FSFRCE, ASREAD, or GSREAD.
3. Immediate detection with data. (Not possible with the IBM 3277 Display Terminal). The designator character is "&" (X'50'). The effect is the same as pressing ENTER.

For more information on the mechanics of light-pen detection and the use of designator characters, refer to the appropriate component description manual.

Map-defined input editing

Using GDDM-IMD's Field Naming or Application Data Structure Review steps, you can specify that the following transformations are to be performed automatically by GDDM on input data passed to the application program. The transformations are specified for individual fields.

- Folding: translation to uppercase of all alphabetic input entered into the field.
- Justification and padding: right- or left-alignment and padding of data entered into the field.
- Attention identifier translation: translation of the AID associated with the input transmission into a predetermined character string.

For information on how to specify these transformations on a map, see the Application Data Structure Review step of GDDM-IMD in the *GDDM Interactive Map Definition* book. The information given in the remainder of this section relates to the application program's view of the transformed fields returned in response to a receive request.

Notes:

1. The transformations take place on input from the operator, for receipt by the application on an MSGET. Data that is placed into the application data area by the application's MSPUT and map-defined default data is not transformed, even though it may be read back using MSGET.

   The effect of the transformations is not immediately visible to the operator. However, if the application does not modify the field, delete the mapped field, or delete the page, the transformed data is displayed to the operator on the next ASREAD, FSFRCE, or GSREAD call.
2. If more than one of these transformations have been specified for a given field, processing is done in this order:
   a. AID translation
   b. Folding
   c. Justification and padding.

AID translation

At map definition time, you can associate an AID translation table with an input field on a map. This field is called an "AID receiver" field.

The translation table is set up during map definition. It defines character strings for the various terminal function keys (and the light pen, trigger fields, operator ID card reader, and magnetic slot reader, if required).

When the operator uses the corresponding key, GDDM places the corresponding character string into the designated field.

AID translation is not restricted to a single field on the map. You can associate several fields with the same or different translation tables and thus receive different character strings in the fields on input.

AIDs can be specified as "do not translate," in which case, the existing field value remains unchanged. For AIDs not explicitly named in the table, a default translation value can be specified; on the other hand, these AIDs can be specified as "do not translate."

An AID receiver field can have a corresponding display field, although this is not mandatory. If the receiver field has a corresponding unprotected display field, operator input into that field is overwritten by the translated AID value unless the operator uses an interrupt key that is designated (explicitly or implicitly) "do not translate."

Folding

When specified, folding always occurs irrespective of what other attributes have been specified for the field.

The folding transformation uses the Lowercase-to-Uppercase Translation Table in the GDDM Alphanumerics Defaults Table (ADMDATRN).

Justification and padding

During map definition, you can specify that a field should be right-justified, left-justified or not justified, and, if you want, that it should be padded with a particular character. If you do not specify a padding character, defaults are used; that is, character zero for right-justified fields, blank for left-justified fields.

For right-justified fields:

1. The rightmost significant (that is, nonblank, nonnull) character is aligned with the rightmost boundary of the field in the application data structure. Leading blanks or nulls are then changed to the padding character.
2. The length adjunct (if one was specified for the field) is set to the application data structure field length.

For left-justified fields:

1. The leftmost significant (that is, nonblank, nonnull) character is aligned with the leftmost boundary of the field in the application data structure. Trailing padding characters are then added to fill the field.
2. The length adjunct (if one was specified) is set to the number of characters in the field up to the first padding character.

For fields for which no justification is specified, the input data is left unchanged (that is, leading and trailing blanks are **not** removed), and the rest of the field is filled with blanks. The length adjunct, if specified, is set to the number of characters

(including leading and trailing blanks) entered by the terminal operator.

If the input data is longer than the field in the application data structure, it is truncated on the right, irrespective of any justification specification, before leading and trailing blanks are suppressed, and a warning message is issued when MSGET is used on the map.

Copying the application data structure into the program

When you have finished the map definition and generation processes, you will have an application data structure for each map, each having the same name as the associated map. You can copy these application data structures into your application program, if it is a COBOL or PL/I program.

For an Assembler program, you must include macro instructions in your program having the same names as the maps. These expand into DSECTs at assembly time.

An example showing the code that might be used for a COBOL program is given below. For illustration, assume that there is a page that is constructed from three separate maps named HEADER, DATAREC, and TRAILER. The maps belong to a mapgroup called MAPGRP.

```
01 HEADER.
   COPY HEADER.
01 DATAREC.
   COPY DATAREC.
01 TRAILER.
   COPY TRAILER.
```

Note: As part of the application structure declaration, GDDM-IMD generates a declaration of a variable with name “mapname-ASLENGTH” (COBOL) “mapname_ASLENGTH” (PL/I) that is initialized with the length, in bytes, of the application structure. This variable can be used as the length parameter in MSPUT and MSGET calls.

Overlaying application data areas

Sometimes, for programming reasons such as conserving storage, it is convenient to overlay the storage used by one of several application data structures. Generally, the structures are not the same length. In this situation, COBOL requires that the longest record description occurs first. To avoid needing to know in advance which record is the longest, you can specify

```
LARGE STRUCTURE ===> YES
```

in frame 3.0 of the generation step of GDDM-IMD. This causes GDDM-IMD to generate an additional structure in a file with the same name as the mapgroup containing a single data item of length equal to that of the largest record.

The following code in the relevant section of the COBOL program then creates the necessary overlaid record descriptions:

```
01 MAPGRP.
   COPY MAPGRP.
01 HEADER REDEFINES MAPGRP.
   COPY HEADER.
01 DATAREC REDEFINES MAPGRP.
   COPY DATAREC.
01 TRAILER REDEFINES MAPGRP.
   COPY TRAILER.
```

COBOL also has the restriction on the placement of declarations using REDEFINES. To satisfy this restriction GDDM-IMD does not generate variables initialized to the application structure length, if you request

```
LARGE STRUCTURE=YES
```

Note: If one of the maps has a name that is the same as the mapgroup name, the application data structure for that map is expanded by a dummy data item (if necessary) to make it as long as the longest application data structure.

Double-byte character string fields

Double-byte character strings (DBCS) fields are specially treated in some cases. (Double-byte character string fields are used for Kanji and Hangeul applications).

A field can be designated as DBCS by using GDDM-IMD’s Field Definition steps, or Field Attribute Definition steps, or both of these.

A field can also be changed to or from DBCS by using a PS attribute adjunct and specifying a value of X'F8' (C'8') for DBCS, or C' ' (or any other valid value) for EBCDIC.

However, the special treatment of length adjuncts and cursor positioning provided for DBCS fields depends only on how the fields were defined to GDDM-IMD. Dynamically changing a field to or from DBCS does not change this treatment.

The special treatment is:

Length adjuncts
If a field is designated **at map definition time** as a DBCS field, the field’s length adjunct is always interpreted as several two-byte characters. Hence, the length of the data in bytes is twice the value of the length adjunct.

Cursor position
If a field is designated **at map definition time** as a DBCS field, the cursor position specified by MSCPOS and returned by MSQPOS is interpreted as several two-byte characters. Hence, the position within the field in bytes is twice the value specified (minus 1).

Mixed double-byte and single-byte character fields in maps

Some Asian languages, including Chinese, Kanji, and Hangeul are displayed and printed using double-byte character sets (DBCS), which means that each character is represented by two bytes. European languages use Latin single-byte character sets (SBCS). The IBM 5550 Multistation and Personal System/55 workstations will display and print both SBCS and DBCS characters.

Sometimes, the two types need to be mixed in a single alphanumeric field. The 5550 and Personal System/55 allow this.

The internal representation of mixed character strings makes use of shift-out (SO) and shift-in (SI) control characters, `X'0E'` and `X'0F'`, to indicate the start and end of a DBCS substring.

There are two ways of displaying mixed character strings, called mixed-with-position and mixed-without-position. The display method to be used is specified in the map definition for each field.

- Mixed-with-position

  The SO/SI codes occupy one character position each, and are displayed as either a blank or a special character – the terminal user can select which.

- Mixed-without-position.

  The SO/SI codes do not occupy a character position on the screen.

The initial input mode of the workstations is SBCS. To enter DBCS characters, the operator presses a special shift key to change the mode. After entering the DBCS string, pressing another shift key returns the terminal to SBCS mode, so further single-byte characters can be entered.

GDDM-supplied mapping constants

The following table lists the GDDM-supplied declarations that contain mapping constants. By including these declarations in your program, you can simplify the setting of the second byte of attribute adjuncts by using a mnemonic name rather than a bit value.

The declarations contain mnemonically-named variables for every attribute, and for combinations of attributes. The variables are initialized to the bit patterns required in the 3270 attribute bytes.

The method of including the declarations in your program varies according to the subsystem and programming language that are being used.

Table 44. GDDM mapping constants tables

Mapping constants table name	Language
ADMUAIMC	Assembler
ADMUBIMC	C
ADMUCIMC	COBOL
ADMUPIMC	PL/I

Chapter 17. GDDM high-performance alphanumerics

High-performance alphanumerics (HPA) is another way of doing alphanumerics in GDDM, and is intended for complex applications that require minimum instruction path length within GDDM.

The application program may not mix mapped and procedural alphanumeric field definitions with HPA field definitions on the same GDDM page.

The style of application programming interface used by HPA differs from that used by other parts of GDDM, such as procedural alphanumerics. When using procedural alphanumerics, application programs use many API calls to describe the data to GDDM for output, and also to determine the data input by the device operator. In contrast, the HPA application builds a data structure to describe all the data, and passes that to GDDM for output. Also, the data input by the device operator is returned to the HPA application in the same data structure. Changes to the data are indicated through status indicators that are part of the structure.

Note: Although HPA can be used from REXX, its use is not recommended, except for prototyping, because the instruction-path length in the interpreter interface to GDDM is significant.

HPA data structure

The data structure consists of three distinct objects. These are:

The field list
The data buffer
The bundle list.

The field list

The **field list** groups together all information about the layout of alphanumeric data on one GDDM page. New fields can be added to an existing GDDM page, or old ones deleted, by modifying the field list. To give additional flexibility, there may be more than one field list in any GDDM page, so that if an existing field list is used up, further field definitions can be added by creating a new one.

A field list consists of a header followed by field definitions.

The header contains:

The status of the field list
The number of field definitions in the list
The size of the field definitions
The cursor position on the page.

Each field definition contains:

The status of the field definition
The size and position of the field on the GDDM page
A reference to the field attribute bundle definition in the bundle list
A reference to the character data
Optional length of character data
Optional references to character attributes.

The field list is represented as a rectangular array of half-word integers, in which the first row is the header and the following rows contain field definitions.

It can be declared as a structure, or as a two-dimensional array stored in row-major order. Programming languages that use column-major ordering of two dimensional arrays will have to exchange rows and columns in the description which follows. Below is a sample PL/I declaration for a field list, where “depth” and “width” are the array dimensions used in the API call APDEF:

```
DCL FIELD_LIST(depth,width) FIXED BIN(15);
```

The numbers beside each component description below are the indexes of each item in the row. See Figure 23 on page 370.

The field list header row

1 – List Status
The status of the field list.

Values that can be assigned to list status are the same as field status; in fact, list status must always be equal to the value obtained by ORing together the values of all the field statuses in the field list. For example, if any field has the indicator set to indicate that the field is to be “output” because the character data has been changed by the application, the corresponding indicator in list status must also be set. This means that whenever the application changes a field status indicator, it must ensure that the list status indicator is correct. Whenever GDDM changes a field status indicator it will also do this.

2 – Used depth
The number of rows in the field list used by GDDM.

This value must be in the range 1 through list depth. It may be changed by the application to add new fields or to remove deleted fields from the list.

Note: If this number is increased to add new fields to the list, the create indicator must be set in the new field-definition status elements. Also, if deleted fields are removed from the list, the deletions must first have been processed by GDDM, which sets the status element in the field definitions to zero.

3 – Used width
The number of elements in the header and each field definition used by GDDM.

This value must be less than or equal to the list width, and must be in the range 6 through 10. If the value is less than the list width, any extra elements in the header and each field definition are ignored by GDDM, and may be used by the application to record its own data. It may be changed by the application to extend or reduce the field definitions. An example of this might be increasing the used width to 9 to specify character color. If the used width is changed, the output indicator must be set in the field definition status elements of all the field definitions altered by this change.

If this value is less than 10, then the omitted parts of the field definition are described as being "not present," and assume default values.

Note: Even though GDDM may not use as many elements in the header as in the field definitions, only those elements beyond the used width may be used for application data. The rest must be zero.

4 – Cursor row
This is the row position of the alphanumeric cursor on the GDDM page.

When used, it must be in the range 1 through page depth, otherwise it must be zero. If the field list is designated as the one used for cursor positioning, then the cursor row and cursor column are used to position the alphanumeric cursor on output, and also to return its position on input. This designation is made by setting the mode parameter of the APDEF or APMOD call.

This cursor position overrides any cursor position specified by calling ASFCUR. During I/O, if the cursor position specified lies outside the page window, then the cursor is placed at the closest position within the page window.

5 – Cursor column
This is the column position of the alphanumeric cursor on the GDDM page.

When used it must be in the range 1 through page width, otherwise it must be zero.

The field definition row

1 – Field Status
The status of the field definition. The list of values below shows both numerical value and corresponding bit position of the indicator. If your use of HPA requires complex testing and setting of these status indicators then you may prefer to declare the status element as a bit string.

Values that can be assigned to the field status are:

1 — Bit 15 — Process
If this indicator is not set, none of the other indicators in the field status element may be set.

Only those field definitions that have this indicator set are processed. This allows space for future field definitions to be reserved in the field list, in which case the application program must set both this indicator and the create indicator before the first use of the field. If a field has been indicated to be deleted, GDDM sets the field status element to zero on the next I/O to the primary device involving the GDDM page.

Note: An I/O involving the page is any I/O operation, ASREAD, FSFRCE, and so on, for the primary device to which the page belongs during which the page is the current one for its partition, and the partition set is the current one for the device.

2 — Bit 14 — Create
Indicates a new field to be created.

If it is set, GDDM resets it on the next I/O to the primary device involving the GDDM page. When a field list is first defined to GDDM all its fields are assumed to be new, so this indicator need not be set.

4 — Bit 13 — Delete
Indicates a field to be deleted.

When the application sets this indicator, it informs GDDM that the field is to be deleted. GDDM resets the entire status element, including the Process indicator, on the next I/O to the primary device involving the GDDM page. The field definition may not be reused to define another field until after GDDM has reset this indicator.

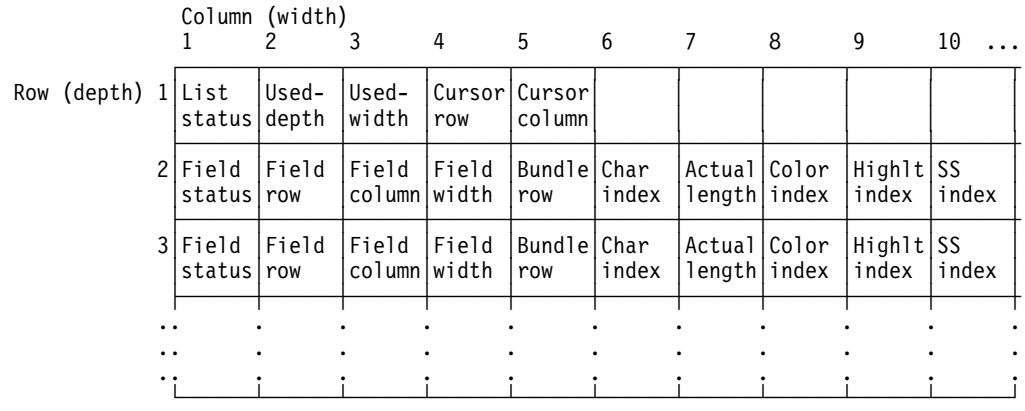


Figure 23. Field list array

8 — Bit 12 — Output
Indicates a field to be output.

It must be set by the application whenever it changes one of the following:

Character data
Character attributes
Character index
Color index
Highlight index
Symbol-set index
Actual-length
Bundle-row.

This indicator is reset by GDDM on the next I/O to the primary device involving the GDDM page.

Notes:

1. This indicator is set by GDDM if the device operator updated the field. This causes the field to be output on the next I/O to ensure that any input data editing is reflected back on the device.
2. This indicator should not be set to indicate changes in the bundle definition, it only indicates changes in the field definition.

16 — Bit 11 — Input
Indicates a field has been input.

This indicator is set by GDDM, during input involving the GDDM page, to indicate changes to character data and possibly character attributes, made by the device operator. It should be reset by the application once the changes have been processed.

If more than one status indicator is required, the element must be set to the sum of the numbers corresponding to the indicators required.

2 – Row
This is the row for the top left-hand corner of the field within the GDDM page.

Rows are numbered from top to bottom of the page, starting with 1. This is the position of the field contents, not the field attribute. Once the field has been defined the application may not change the field row until the field has been deleted. For best performance it is recommended that fields are defined in order of their positions on the page.

3 – Column
This is the column for the top left-hand corner of the field within the GDDM page. Columns are numbered from left to right across the page, starting with 1. This is the position of the field contents, not the field attribute. Once the field has been defined, the application may not change the field column until the field has been deleted.

4 – Width
This is the number of columns that the field occupies.

The width may cause the field to extend beyond the right-hand side of the page, in which case it wraps to the left-hand side of the page on the next row. A field may not extend below the bottom of the page, neither may fields overlap. Once the field has been defined, the application may not change the field width until the field has been deleted.

Width also defines the data-area length. For mixed-without-position fields the data-area length is twice Width bytes, and for other fields the data-area length is Width bytes. The data-area length is the length of the data areas in the data buffer, where the data for the field is held. There may, optionally, be data areas for:

Character data
Character color attributes
Character highlight attributes
Character symbol-set attributes.

The data areas as defined by the character index and width, the color index and width, the highlight index and width, and the symbol-set index and width, must be contained totally within the data buffer.

5 – Bundle row
This is the row number in the bundle list of the field attribute bundle definition. It must be in the range 2 through the number of rows in the bundle list.

6 – Character Index
This is the index in the data buffer of the data area containing the characters that occupy the field. An index of 0 indicates that there are no character codes for the field. The character data area must be present if color, highlight, or symbol-set data areas are present. The character data area must also be present if the field is unprotected or has the MDT attribute.

Note: It is possible for more than one field to be associated with the same data area or overlapping data areas, within the data buffer. This does not cause any difficulty if all the fields are protected.

In the instance where one or more of the fields is unprotected, the application must set the output indicators of **all** the fields involved if the data area has been changed as a result of device operator input. If this is not done, the corresponding fields on the screen may not be updated on the next I/O.

In the instance where two or more unprotected fields share the same data area, and the device operator enters updates into two or more such fields in the same I/O operation, the resulting contents of the data area are undefined.

7 – Actual Length
This is the length of the data in the data area(s).

When the application changes the data, it must set this to the length of data in the character, color, highlight, and symbol-set data areas in the data buffer. If not present an actual length of data-area length is assumed. If a value greater than data-area length is specified, then only data area length bytes are output. If the number of bytes output does not fill the field, then the rest of the field is filled with the pad character. (The pad character is null for character data and blank, meaning inherit the field attributes, for character attributes.)

If the device operator enters data into the field, GDDM sets actual length to the length of data, in bytes, now in the field, up to and including the last nonpad character.

GDDM only sets actual length if the field status indicates that changes to field contents have been input.

8 – Color Index
This is the index in the data buffer of the data area containing the color codes for individual characters that occupy the field. If not present, an index of 0 is assumed. An index of 0 indicates that there are no character color codes for the field.

9 – Highlight index
This is the index in the data buffer of the data area that contains the highlight codes for individual characters that occupy the field. If not present an index of 0 is assumed. An index of 0 indicates that there are no character highlight codes for the field.

10 – Symbol-set index
This is the index in the data buffer of the data area that contains the symbol-set codes for individual characters that occupy the field. If not present, an index of 0 is assumed. An index of 0 indicates that there are no character symbol-set codes for the field.

Fields that do not have character attributes should specify indexes of 0. Omitting character attribute data areas, when not required, significantly improves the performance characteristics of an application.

Example: This is an example of a field list declaration in PL/I (compare with Figure 23 on page 370).

```
DCL FL(5,10) FIXED BIN(15) STATIC INIT

/*STA  DEP  WID  CSR  CSC                          */
 ( 1,   5,  10,   2,   5,   0,   0,   0,   0,   0,
/*STA  ROW  COL  WID  BLR  CHI  ACT  COI  HII  SSI*/
   1,   2,   5,   4,   2,   1,   4,   0,   0,   0,
   1,   4,  10,  11,   3,   5,  11,   0,   0,   0,
   1,   6,  15,  13,   4,  16,  13,   0,   0,   0,
   1,   8,  20,   3,   5,  29,   3,  32,  35,  38);
```

The data buffer

The **data buffer** consists of data areas containing the data and character attributes for each field defined in the field list. The position and size of each data area within the data buffer is defined in the field list. Each field-list entry contains the length and index into the data buffer of its character-data area. Optionally, it may also contain indexes to a character color data area, a character highlight data area, and a character symbol set data area.

Mixed double-byte and single-byte character fields: The internal representation of mixed character strings makes use of shift-out (SO) and shift-in (SI) control characters, X'0E' and X'0F', to indicate the start and end of a DBCS substring.

There are two ways of displaying mixed character strings, called mixed-with-position and mixed-without-position. The display method to be used is specified in the bundle definition for each field.

- Mixed-with-position

  The SO/SI codes occupy one character position each, and are displayed as either a blank or a special character – the terminal user can select which.

- Mixed-without-position.

  The SO/SI codes do not occupy a character position on the screen.

Character attributes Character attributes are represented by these codes:

Color

blank	X'40'	Inherit the field color (the default)
1	X'F1'	Blue
2	X'F2'	Red
3	X'F3'	Magenta (pink)
4	X'F4'	Green
5	X'F5'	Turquoise (cyan)
6	X'F6'	Yellow
7	X'F7'	Neutral (white on displays, black on printers).

Highlight

blank	X'40'	Inherit the field highlight (the default)
1	X'F1'	Blink
2	X'F2'	Reverse video
4	X'F4'	Underscore.

Symbol-set

X'00' or X'40'	Inherit the field symbol set (the default)
X'01' through X'03'	Loadable symbol set (3800 system printer)
X'41' through X'DF'	Loadable symbol set (3270 family devices)
X'F1'	Alternative nonloadable symbol set (3270-family devices).

Notes:

1. The two character attributes, corresponding to the two bytes of a DBCS character, must both be the same.
2. Symbol-set character attributes, corresponding to DBCS characters, must be blank.

Example: The data buffer to go with the field lists in the earlier example might be:

```
DCL DB CHAR(40) STATIC INIT
     ('HighPerformanceAlphanumericsAPI356124 &&');
     /*↑   ↑          ↑            ↑ ↑ ↑ ↑  */
```

The field list has four fields defined, corresponding to the words `High`, `Performance`, `Alphanumerics`, and `API`. No color, highlight, or symbol set indexes have been specified for the first three fields. The field definition for the fourth defines a color index that selects the `'356'`, a highlight index that selects the `'124'`, and a symbol-set index that selects the `' &&'` in the data buffer. (The blank specifies inheritance of the field symbol set, and the two `'&'` characters (X'50', decimal 80) request the use of a symbol set with identifier 80.)

The bundle list

The field attributes that are used with the alphanumeric fields defined in the field list, are themselves defined in the **bundle list**. Each field definition in the field list contains a bundle row, which is the row number of the bundle definition in the bundle list.

The first row of the bundle list is a header, and following rows contain field attribute bundle definitions. Each bundle definition consists of a status element, and the number of type-and-value pairs in the definition, followed by pairs of attribute types and attribute values describing the attributes of the bundle. It may also contain application data.

Figure 24 on page 374 illustrates the layout of a bundle list.

The bundle list can be declared as a structure, or as a two-dimensional array stored in row-major order. Programming languages that use column-major ordering of two dimensional arrays have to exchange rows and columns in the description that follows. Below is a sample PL/I declaration for a bundle list, where “depth” and “width” are the array dimensions used in the API call APDEF:

```
DCL BUNDLE_LIST(depth,width) FIXED BIN(15);
```

The components of the bundle list are:

Bundle list header row

1 – List Status
The status of the bundle list.

Values that can be assigned to list status are the same as bundle status; in fact list status must always be equal to the value obtained by ORing together the values of all the bundle statuses in the bundle list. For example, whenever the application changes a bundle status indicator it must also change the list status.

2 – Used depth
The number of rows that GDDM uses in the bundle list.

Its value must be in the range 1 through list depth. It may be changed by the application to add new definitions or to remove unused definitions from the list. If this value is increased, the new bundle definitions must have the bundle changed indicator set in the bundle definition status element.

3 – Used width
The maximum number of elements in the header and each bundle definition used by GDDM.

This value must be less than or equal to the list width, and the minimum value is 4. If the value is less than the list width, then extra elements in the header and each bundle definition will be ignored by GDDM, and may be used by the application to record its own data. It may be changed by the application to extend or reduce the maximum number of type-and-value pairs in the bundle definitions.

Note: Although GDDM may not use as many elements in the header as in the bundle definitions, only those elements beyond the used width may be used for application data, the rest must be zero.

Bundle definition row

1 – Bundle status
The status of the bundle definition. The list of values below shows both numerical value and corresponding bit position of the indicator.

1 — Bit 15 — Bundle changed
This must be set by the application to tell GDDM of changes made to the bundle definition, and, if set by the application, is reset by GDDM on the next I/O to the primary device involving the GDDM page. Set it if the number of pairs, the attribute types, or the attribute values, have been changed.

Note: All the other status indicators in the halfword must be zero.

2 – Pairs
The number of type-and-value pairs in the bundle definition.

The minimum value is 0 and the maximum value is (Used_width–2)/2. Elements in the bundle definition beyond this specified number are ignored by GDDM.

3 – Type-and-value pairs
Type is a code for the attribute type, such as “color” and value is a code for the corresponding value such as “blue”.

The permitted type codes and their associated value codes are:

0 Dummy

This is a special type code that causes the type-and-value pair is to be ignored by GDDM. It effectively reserves space within the bundle definition for future use by the application. The associated value is ignored.

8 Field type

The permitted values are:

0 Unprotected alphanumeric (the default)
1 Alphanumeric output, numeric input
2 Protected alphanumeric.

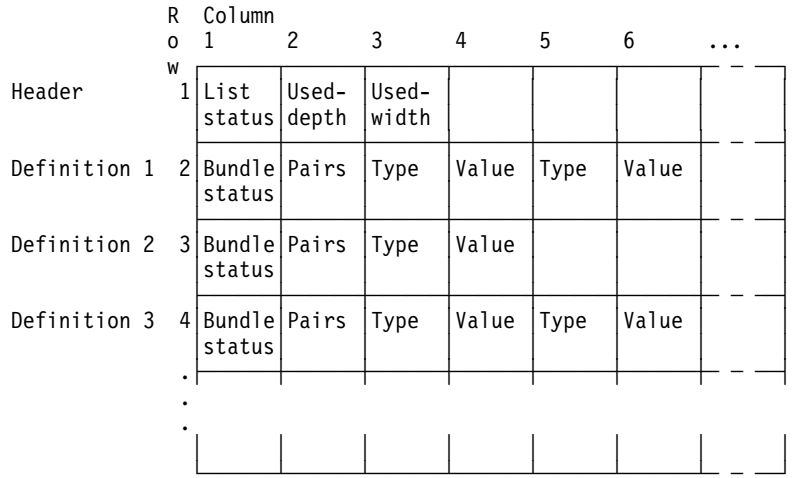


Figure 24. The bundle list array

16 Intensity

The permitted values are:

0 Invisible
1 Normal (the default)
2 Bright.

24 Color

The permitted values are:

0 Default
1 Blue
2 Red
3 Magenta (Pink)
4 Green
5 Turquoise (cyan)
6 Yellow
7 Neutral (white on color displays, black on printers).

32 SBCS Primary symbol set alias

The permitted values are:

0 Default. For a 3270-family device, the base nonloadable symbol set; for a 3800-system printer, the first loadable symbol set (use the CHARS parameter to specify the loaded symbol sets when printing).
1 through 3. For a 3800-system printer, the second, third, and fourth loadable symbol sets respectively (use the CHARS parameter to specify the loaded symbol sets when printing).
65 through 223. For a 3270 family device, loadable symbol sets corresponding to X'41' through X'DF'. The alias must be made known to GDDM with a call to PSDSS, PSLSS, or PSLSSC to load the symbol set.

40 Highlight

The permitted values are:

0 Normal (the default)
1 Blink
2 Reverse video
4 Underscore.

48 End

The permitted values are:

0 Autoskip (the default)
1 Notautoskip.

56 Transparency

The permitted values are:

0 Opaque (the default)
1 Transparent.

64 SBCS/DBCS

The permitted values are:

0 SBCS (the default)
1 Mixed-with-position
2 Mixed-without-position
3 DBCS.

Note: On 5550-family displays all unprotected fields on the device that are not DBCS, or mixed-with-position are enabled for mixed-without-position input if any bundle list on the device specifies mixed-without-position. If the device operator enters mixed-without-position data into a field, GDDM only places the correct shift-in, shift-out, and DBCS characters into the data buffer if mixed-without-position is specified for the field.

72 Outlining

The permitted values are:

0 None (the default)
1 Underline
2 Vertical line on right
4 Overline
8 Vertical line on left.

For an outlining attribute that is composed of more than one of these lines, specify the sum of the numbers corresponding to the lines required.

80 Modified data tag (MDT)

This defines the field MDT setting. It causes the physical MDT bit to be set so that the fields can be returned as input to a subsequent application program after GDDM terminates. This function is intended primarily for use under CICS and IMS.

The permitted values are:

0 Reset the MDT (the default)
1 Set the MDT.

88 Reply

This defines the character reply attribute. It specifies whether the device operator is able to enter color, highlight, or symbol-set character attributes into the field. If the field definition also specifies data areas for character attributes, GDDM will update the data areas with the attributes input.

The permitted values are:

0 Character reply mode off (the default)
1 Enable color character reply mode
2 Enable highlight character reply mode
4 Enable symbol-set character reply mode.

To enable combinations of color, highlight, and symbol-set character reply modes, specify the sum of the numbers corresponding to the enablements required.

Note: On 3270-family displays all unprotected fields in the real partition (or on the real screen if emulated partitions are being used) are enabled for character-attribute input if any bundle list on the page sets this attribute. If the device operator enters character attributes into a field, GDDM only places the character attributes input into the data buffer if the appropriate reply mode is enabled for the field.

96 Pen detectable

This attribute permits selection of fields by a light pen or cursor select key.

The permitted values are:

0 Not pen detectable (the default)
1 Pen detectable.

The type of selection that occurs is determined by the first data character in the field; this character is called a designator character. A field having a “pen detectable” attribute but not starting with a valid designator character is not selectable.

The types of selection that can be set are:

Delayed detection. When selected by the device operator, the field is marked as “modified” but nothing is transmitted until the device operator performs another action associated with field modification (such as selecting an “immediate detection” field or pressing ENTER). The designator character for this type of field is “?” (X'6F'). If the field is selected the designator character changes to “>” (X'6E'); another selection restores it to “?” and cancels the modification indication.

Immediate detection without data. The designator character is a blank (X'40'). Selection of this type of field causes immediate input transmission. No data from any of the fields is transmitted, however. The effect is thus:

1. The ASREAD returns an Attention Type of 2 indicating light pen selection. All changes typed in by the device operator are lost.
2. GDDM restores all fields on the display to their original value at the next ASREAD (or other I/O call).

Immediate detection with data. The designator character is “&” (X'50'). The effect is the same as pressing ENTER. (Not possible with the IBM 3277 display terminal.)

Except for dummy, the same type may not appear more than once in the same bundle definition.

Example: Below is an example of a declaration for a bundle list in PL/I:

```
DCL BL( 5,10) FIXED BIN(15) STATIC INIT

  /*STA DEP  WID                            */
   ( 0,  5, 10,  0,  0,  0,  0,  0,  0,  0,
  /*STA PRS TYP VAL COL VAL BDY VAL PSS VAL*/
     0,  3,  8,  0, 24,  3, 72,  1,  0,  0,
     0,  3,  8,  0, 24,  5, 72,  3,  0,  0,
     0,  4,  8,  0, 24,  6, 72, 15, 32, 80,
     0,  4,  8,  0, 24,  3, 72,  7, 88,  7);
```

How to use high-performance alphanumerics

Move mode and locate mode: There are two modes in which data can be transferred between GDDM and the application program, which are the **move** and **locate** modes. The mode is specified through the “mode” parameter of the APDEF call.

If move mode is specified, the field list, data buffer, and bundle list are copied by GDDM when APDEF is called. Subsequent output and input processing, done by GDDM, use the GDDM copies. When the application needs to retrieve updates made by the device operator, or modify the fields, it must query the field list, data buffer, and bundle list by calling APQRY. This returns copies of the field list, data buffer, and bundle list held by GDDM. When the application has modified the field list, data buffer, and bundle list, it must pass the modified versions back to GDDM by calling APMOD.

If locate mode is specified, GDDM does not copy the field list, data buffer, or bundle list. Subsequent output and input processing, by GDDM, use the copies in application storage. The application must not release the storage that these objects occupy until the field list has been deleted. The contents of the field list, data buffer, and bundle list must be valid whenever GDDM is called. When using locate mode, it is not necessary to call APQRY to determine device operator updates, nor to call APMOD to inform GDDM of changes made by the application.

The choice of move mode or locate mode will affect any application data embedded in the field list, data buffer, or bundle list. If move mode is used, this application data is copied by GDDM on APDEF and subsequent calls to APMOD. The value copied on the most recent APDEF or APMOD call is returned by GDDM on APQRY. This means that any changes made after APDEF or APMOD will be lost on the next call to APQRY. If locate mode is used this application data is not altered by GDDM.

Output: To display a page of alphanumeric fields:

- Construct the field list and associated data buffer and bundle list to describe the page of alphanumerics. Set the field definition statuses for all the fields to be shown to 1. Set the field list status to 1. Set the bundle list status, and all bundle definition statuses to 0.
- Call APDEF to define the field list and associated data buffer and bundle list to GDDM.
- Call ASREAD, or another GDDM I/O call as required.

Input: To retrieve device operator updates to the page of alphanumeric fields following an I/O operation:

- If using move mode, retrieve the field list, data buffer, and bundle list from GDDM by calling APQRY.
- Test the field list status input indicator to determine if any fields have been updated by the device operator. If they have, test field definition input indicators to determine which fields have been changed, and process the input found in the data buffer.
- If the alphanumerics are not to be reshown, they should be cleared by calling APDEL.

Reshow: The application may need to reshow the page of alphanumeric fields just input, which should be done as follows:

- Reset the field list status input indicator and the field definition input indicators.
- Change the data or character attributes in the data buffer as required, and set the corresponding output indicators in the field definition and header status.
- Change the bundle definitions in the bundle list as required, and set the corresponding bundle definition and header status indicators.
- Change the field definitions in the field list as required, and set the corresponding status indicators to specify what has changed.
- If using move mode, return the modified field list, data buffer, and bundle list to GDDM by calling APMOD.
- Call ASREAD, or another GDDM I/O call as required.

Field list update rules: The rules for altering a field list are:

- The input indicators, which indicate device operator updates, should be reset by the application after each I/O. If this is not done, the application will not be able to detect further updates on a subsequent I/O.
- Field row, field column, and field width may not be changed, except when using a previously-unused field definition entry to define a new field. Fields may be defined in any order, but must not overlap. They may wrap from row to row, but must not extend beyond the end of the page.
- Bundle row may be changed by the application, in which case the application must also set the output indicator to indicate to GDDM that this is changed. It is not necessary to set this indicator if only the bundle definition has changed and the field definition has not changed.
- If the character index, color index, highlight index, symbol-set index or actual length are changed, then the application must set the Output indicator to indicate to GDDM that the field has changed and is therefore to be output on the next I/O.
- When a previously unused field definition is activated, the process indicator and the create indicator must be set by the application. These indicators should never be reset by the application, only by GDDM.
- If an existing field is to be deleted, the field delete indicator should be set by the application. This indicator should never be reset by the application, only by GDDM, and the field definition entry may only be reused to define a new field after GDDM has reset the entire field status element.
- Changes to any field definition status indicator may also require changes to the corresponding header status indicator. The header status must always be set to the value obtained by ORing together all the field status elements.

Data buffer update rule: The rule for altering a data buffer is:

- If a character data area, or a character attribute data area is modified, then the output indicators in the corresponding field definition status and field list status must be set.

Bundle list update rule: The rule for altering a bundle list is:

- If a bundle definition is modified, the bundle changed indicator in the bundle definition status and bundle list status must be set.

Dynamic fields: Dynamic alphanumeric fields, using HPA, may be obtained by reserving space in the field list, data buffer, and bundle list for the fields to be added later. Reserved field definitions in the field list may be made by leaving the process indicator off. Reserved space may be left in the data buffer by not referring to it in existing field definitions. Reserved bundle definitions in the bundle list may be made by setting the number of type-and-value pairs to zero, or by using the dummy attribute type.

It may become necessary at some stage to enlarge the structures. When this happens, the APMOD call may be used to change the size of the field list, data buffer, or bundle list and also their location if using locate mode. The application must allocate new-larger data structures to replace the old ones, initialize them from the old ones (or by calling APQRY), call APMOD to define the enlarged versions to GDDM, and throw the old ones away.

Note: If APMOD is used in this way, any differences between the contents of the old and new structures must be indicated by change indicators as defined in the rules above.

Interpreted languages: In general, locate mode cannot be used by applications written in interpreted languages such as REXX. When using these languages move mode must be used. See also the restrictions on shared storage below.

Read-only storage: In certain circumstances it may be desirable to use HPA with the field list, data buffer, or bundle list in read-only storage. An example might be an application that is used by many users at the same time. In this instance, it would be more efficient if fixed panel layouts were placed in shared storage. To use HPA from read-only storage, ensure that GDDM does not write to it by adhering to the rules below:

- Neither APDEF nor APMOD alters the storage of the field list, data buffer, or bundle list.
- In move mode, ASREAD does not alter the objects in user storage.
- In locate mode, ASREAD only alters:

The field list	If any of the create, delete, or output indicators are set, or if any field is unprotected or has the MDT attribute
The data buffer	If any field is unprotected or has the MDT attribute
The bundle list	If any status indicators are set.

Shared storage: When using locate mode, it is possible for an application to define more than one field list using the same storage. Field lists, data buffers, and bundle lists could all share storage. The rules for sharing storage are:

- Field lists may not share storage unless they are read only. See the section on Read-only storage on page 377.
- Bundle lists may be shared between more than one field list on the same device. They may not be shared between field lists on different devices unless they are read only.
- Data buffers may be shared between more than one field list only if unprotected data areas (that is, data areas corresponding to fields that are unprotected or have the MDT attribute) are not shared.

Note: Violations of these rules are not detected, and the results of such a violation are undefined.

Validation: To enable GDDM to be used as the device driver for fully tested program products, it is necessary to be able to run HPA without validation. (Validation is not necessary for tested applications and the performance advantages are significant.)

Validation checks the API parameters such as identifiers and lengths, as well as the field list, data buffer, and bundle list. The field list, data buffer, and bundle list are not validated during the API call processing as other parameters are, instead they are validated during processing for each I/O call involving the GDDM page.

If the writers of an application choose to use HPA without validation, they do so at their own risk. Incorrect use may result in device checks.

Validation is controlled by an external default as follows:

FRCEVAL — Force validation.

The default is NO. When FRCEVAL=YES is specified, the validation indicator in the mode parameter is overridden so that validation is always performed. The other indicators in the mode parameter are not affected.

For example, when a tested application (for instance, a shipped program product that does not use validation), is suspected of a bug, validation can be turned on to determine whether the application or GDDM is at fault by specifying:

```
ADMMDFT FRCEVAL=YES
```

in the external defaults file. This default may not be specified in the external defaults module, on SPINIT calls, or by API call.

Alternatively validation may be controlled by the mode parameter on the APDEF and APMOD API calls. It may be used during application development, but once an application is fully tested validation should be turned off.

Chapter 18. External defaults

This chapter contains information on the following:

- Changing GDDM's default values
- Format of GDDM external defaults, listed alphabetically.
- Alphabetical list of default descriptions on pages 384 through 393.

GDDM's default values

This section describes the options you can specify to change defaults for your GDDM and subsystem environment. The information is presented in tabular form.

Full descriptions of the defaults are given under "Alphabetical list of GDDM external defaults" on page 384, where they are listed in alphabetical order of the user default specification parameter.

The first four columns of each table give a brief meaning of the option, the source format of the user default specification (UDS) to change that option, the GDDM default for that option, and the encoded format of the UDS. The fifth column shows the methods of implementing the UDS you have specified; it shows where the UDS can be specified, as follows:

M in the External Defaults Module,
F in the External Defaults File,
S in the SPINIT call,
C in the ESEUDS and ESSUDS calls.

Note that not all defaults can be specified by all of the methods; some defaults can be specified by only one of the methods. The final column shows to which subsystems the external defaults apply.

Changing GDDM's default values

The default values supplied by GDDM can be changed to allow for variations in such things as specific operating environments, equipment availability, or user requirements. For full information, refer to the *GDDM System Customization and Administration* book.

If a default keyword is specified without a value, the current default value is not changed. For example, in:

```
DEFAULT ERRTHRS=,NATLANG=F
```

the ERRTHRS keyword has no effect.

A default value of blanks can be defined by specifying it as a null string enclosed in parentheses. For example:

```
DEFAULT TSOS99U=()
```

The tables that follow list, in alphabetical order of default function, the GDDM defaults you can change for each subsystem environment, together with their source-format and equivalent encoded-format user default specifications.

Note that in defaults files, the "ADMMDFT" keyword can be replaced by "DEFAULT".

External defaults: format

The syntax of the external defaults are listed here in alphabetic order of keyword.

Table 45 (Page 1 of 5). 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

Table 45 (Page 2 of 5). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
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
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, cccccccc,.....)	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

Table 45 (Page 3 of 5). GDDM external defaults

Source syntax of the ADMMDFT options	Description	GDDM default	Encoded values – list of fullwords	Valid in: M F S C	System
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
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
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

Table 45 (Page 4 of 5). 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,....)	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		
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		

Table 45 (Page 5 of 5). 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,....)	Filetypes for: Symbol sets Generated mapgroups Saved pictures Chart formats Chart data GDDM-IMD tutorial pages GDF files (reserved) Chart data definition Projection definition Image data Reserved for GDDM-PCLK and GDDM-OS/2 Link files	 ADMSYMBL ADMGGMAP ADMSAVE ADMCFORM ADMCDATA ADMTUTPG ADMGDF — ADMCDEF ADMPROJ ADMIMG ADMPC	4-24,107, aaaa,aaaa, bbbb,bbbb, cccc,cccc, dddd,dddd, eeee,eeee, ffff,ffff, gggg,gggg, hhhh,hhhh, iiii,iiii, jjjj,jjjj, kkkk,kkkk, llll,llll	Y Y Y Y	VM
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
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

Alphabetical list of GDDM external defaults

This section lists the GDDM external defaults and their values in alphabetical order of the user external default description parameter. For example, for the "always-unlock-keyboard" external default you would look up AUNLOCK in this list.

In this list, the values of the external defaults, as supplied by GDDM, are underlined thus: **NO**.

Note: Where an operand is defined as a 4- or 8-character string, it may 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 Chapter 19, "Processing options" on page 395.

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. See "Call format descriptor module" on page 438. The second four bytes receive the address of the APL request code module. See Chapter 5, "APL request codes module" on page 251.

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. See the *GDDM System Customization and Administration* book.

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. For information on how to define these devices, see Chapter 19, “Processing options” on page 395.

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. For information on how to define these devices, see Chapter 19, “Processing options” on page 395.

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. For information on how to define these devices, see Chapter 20, “Name-lists” on page 415.

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 non-blank 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 (“GSCPG – Set current code page” on page 113).

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.

In the encoded UDS format, the default value must be coded as X'xx000000', where “xx” is the hexadecimal equivalent of the character “c”.

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. For more information, refer to the *GDDM System Customization and Administration* book.

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 non-conversational 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. For information on how to define system printers, see Chapter 20, “Name-lists” on page 415.

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 384.

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.

Except on devices that support mixed alphanumeric fields (such as the IBM 5550 and 5553), alphanumeric fields that are to contain mixed strings must also be

defined as “mixed” by the ASFSEN call. On devices that support mixed alphanumeric fields, it is not necessary to specify MIXSOSI=YES, unless mixed graphic character string support is also required. (See also the SOSIEMC external default.)

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 PM)	(U.S. convention; XM=AM or
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.

Chapter 19. Processing options

Processing options (procopts) allow you to specify precisely how the input or output of a device is to be processed, with regard to the devices available, the devices' capabilities, and the subsystem under which they run.

Name-lists are a means of grouping devices according to the device family, and the subsystem under which the application is running. For information on these, see Chapter 20, "Name-lists" on page 415.

Processing options

Processing options can be specified on DSOPEN calls, and in nicknames.

The processing options are summarized in numeric order of option group code in Table 46.

Processing options: format

The processing options are listed here in numeric order of processing option group code.

Table 46 (Page 1 of 2). Summary of processing options and nickname keywords

Procopt group code	Nickname keyword	Arguments	Examples
1	BMSCOORD	{**NO**|YES}	(BMSCOORD,NO)
2	OUTONLY	{**NO**|YES}	(OUTONLY,NO)
3	AUNLOCK	{NO|YES}	(AUNLOCK,NO)
4	PRINTCTL	n,n,n,n,.....	(PRINTCTL, **1,1,66,0,0,0,80,0**)
5	OFDSTYPE	{**PS**|EPS}	(OFDSTYPE,EPS)
5	OFDSTYPE	{**DOC**|PSEG|OVLY}	(OFDSTYPE,PSEG)
5	CDPFTYPE	{**PRIM**|SEC|OVLY} Alternative to OFDSTYPE. Retained for compatibility with earlier releases of GDDM.	(CDPFTYPE,SEC)
6	HRISPILL	{**YES**|NO}	(HRISPILL,YES)
7	HRISWATH	n	(HRISWATH,**1**)
8	PRTPSIZE	w,d,{TENTHS|MILLS}	(PRTPSIZE,80,110,TENTHS)
8	HRIPSIZE	w,d,{TENTHS|MILLS}	(HRIPSIZE,50,30,TENTHS)
9	OFFORMAT	{BITMAP|**IMAGE**|GRIMAGE|GRCIMAGE}	(OFFORMAT,IMAGE)
9	HRIFORMT	{BITMAP|**CDPF**|GRIMAGE|GRCIMAGE} Alternative to OFFORMAT. Retained for compatibility with earlier releases of GDDM.	(HRIFORMT,CDPF)
10	PLTFORMF	{NO|YES}	(PLTFORMF,NO)
11	PLTPENV	n	(PLTPENV,30)
12	PLTPENW	n	(PLTPENW,**3**)
13	PLTPENP	n	(PLTPENP,10)
14	PLTAREA	xmin,xmax,ymin,ymax	(PLTAREA,0,70,0,70)
15	PLTPAPSZ	{ *|A4|A3|...|A|B|...}	(PLTPAPSZ, *)
16	PLTROTAT	{**NO**|YES}	(PLTROTAT,NO)
17	SEGSTORE	{**YES**|NO}	(SEGSTORE,NO)
18	STAGE2ID	xxxxxxxx,xxxxxxxx,...	(STAGE2ID,*,AUX2)
19	LOADDSYM	{**NO**|YES}	(LOADDSYM,YES)
20	ORIGINID	{**NO**|YES}	(ORIGINID,YES)
21	LCLMODE	{**NO**|YES}	(LCLMODE,NO)
22	HRIDOCNM	xxxxxxxx	(HRIDOCNM,FIGURE9)
23	SPECDEV	{IBM5080|*}, {ddname|*}	(SPECDEV,IBM5080)
24	WINDOW	{**NO**|YES}	(WINDOW,YES)
25	PSCNVCTL	{**NO**|START|CONTINUE}	(PSCNVCTL,START)
26	FASTUPD	n	(FASTUPD,**0**)
27	CTLFAST	{**NO**|YES}	(CTLFAST,YES)
28	CTLMODE	{ *|YES|NO}	(CTLMODE,NO)
29	CTLKEY	type,value	(CTLKEY,**4,3**)

Table 46 (Page 2 of 2). Summary of processing options and nickname keywords

Procopt group code	Nickname keyword	Arguments	Examples
30	CTLPRINT	{**YES**|NO}	(CTLPRINT,NO)
31	CTLSAVE	{YES|NO}	(CTLSAVE,YES)
32	INRESRCE	{YES|**NO**}	(INRESRCE,YES)
33	PCLK	{YES|**NO**}	(PCLK,YES)
34	DEVCPG	n	(DEVCPG,00273)
35	IPDSQUAL	{ ***|DP|DPQ|DPT|DPTQ|NLQ|LLL|MLL|HLH|LLH}	(IPDSQUAL,NLQ)
36	PCLKEVIS	{YES|**NO**}	(PCLKEVIS,YES)
37	PRTROT	{**0**|90|180|270}	(PRTROT,90)
37	IPDSROT	{**0**|90|180|270}	(IPDSROT,180)
38	IPDSTRUN	{YES|NO}	(IPDSTRUN,NO)
39	IPDSLPI	{6|**8**}	(IPDSLPI,8)
40	IPDSBIN	m,n	(IPDSBIN,1,2)
41	IPDSIMSW	{**YES**|NO}	(IPDSIMSW,NO)
42	IMGINIT	{**BLACK**|WHITE|BACKGND}	(IMGINIT,BLACK)
43	IPDSCPI	{**100**|120|133|150|167|170|180|200}	(IPDSCPI,100)
44	PATTRAN	m,n	(PATTRAN,5,1)
45	GINKEY	type,value	(GINKEY,**0**,**0**)
46	TOFILE	{**NO**|YES},{**REP**|NOREP}	(TOFILE,NO,REP)
47	PLTDELAY	n	(PLTDELAY,**0**)
48	GRAYLINE	{**NO**|YES}	(GRAYLINE,YES)
49	PSCHAR	{**7**|8}	(PSCHAR,8)
50	POSTPROC	xxxxxxxx	(POSTPROC,COLORPRI)
51	DEVCSET	n	(DEVCSET,1172)
1000	CMSINTRP	{**PA1PA2**|PA2|PA1|NONE}	(CMSINTRP,PA1PA2)
1001	CMSATTN	{**BASIC**|EXTENDED},n,addr	(CMSATTN,BASIC,0,0)
1002	CPSPOOL	xxxxxxxx,xxxxxxxx,**...**	(CPSPOOL,TO,RSCS)
1003	CPTAG	xxxxxxxx,xxxxxxxx,**...**	(CPTAG,OUR3287,PRT,=, GRAPH)
1004	INVKOPUV	{**NO**|YES}	(INVKOPUV,YES)
2000	TSOINTRP	{**PA1**|NONE}	(TSOINTRP,NONE)
2001	TSORESHW	n	(TSORESHW,**0**)
2002	PRINTDST	{class| ***}, [destname|ddname| ***|=], [writer],[forms])	(PRINTDST,G,=,MYPRT)
2003	FRCETYPE	{**FSFRCE**|DSFRCE}	(FRCETYPE,DSFRCE)
3000	COLORMAS	n	(COLORMAS,**0**)

Note: Default settings, shown in either the Arguments or Examples columns, are shown like this: **BLACK**. Where no value is shown like this, either there is no default setting or the default setting is determined by the device being used.

Detailed descriptions, in numeric order of option group code, are given on pages 396 through 413.

Processing options: full descriptions

The processing options are listed here in numerical order of option group code. A full description is given of each processing option, in this format:

- The processing option group code and nickname keyword
- A definition of the nickname syntax
- A brief description of the function of the processing option
- The applicable subsystems
- The applicable device families
- The length of the processing option group, expressed in fullwords
- A breakdown of the function of each fullword.

In this list, the default values, as supplied by GDDM are underlined thus: **NO**.

The processing options are summarized on page 395.

---

0 Dummy
Nickname syntax: not applicable

A dummy processing option, which is ignored. It can be used to pad processing option-lists to any desired length.

Subsystems: All
Devices: All

Length: 1 fullword.

1 The option group code: 0

---

1 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. Coordination mode is more fully described in the *GDDM System Customization and Administration* book.

Subsystems: CICS
Devices: Family-1
Length: 2 fullwords.

1 The option group code: 1
2 The type of coordination:
- **0 (NO)** Not in coordination mode (default)
- **1 (YES)** In coordination mode.

---

2 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
Length: 2 fullwords.

1 The option group code: 2
2 Normal or output-only mode:
- **0 (NO)** Not output-only mode (default)
- **1 (YES)** Output-only mode.

---

3 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 Chapter 18, "External defaults" on page 379), 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
Length: 2 fullwords.

1 The option group code: 3
2 The type of keyboard mode:
- **0 (NO)** Normal mode (default for CICS, TSO, VM)
- **1 (YES)** Always-unlock-keyboard mode (default for IMS).

---

4 Print control options
Nickname syntax: (PRINTCTL,n,n,n,n,....)

(where n,n,n,n,... represents the values of Fullword 3 onwards, as defined below). This option controls printing and copy functions. The group has this format:

Fullword 1	Option code = 4
2	No. of fullwords following
3	Heading indicator
4	Number of copies
5	Page depth
6	Top margin
7	Left margin
8	Bottom margin
9	Max FSLOG characters/line
10	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
Length: 2+N fullwords.

1 The option group code: 4.

2 Number (N) of fullword values that follow (can be 0 through 8).

3 The heading indicator (family-2 only):
- **0** Do not print a heading page
- **1** Print a heading page (default).

4 The number of copies (applicable to family-2 only). The default is 1. If 0 is specified, 1 is assumed.

5 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.

6 The depth of the top margin. 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.

7 The width of the left margin. The default is 0.

See the description for the top margin, under Fullword 6.

8 The depth of the bottom margin (FSLOG and FSLOGC only). The default is 0.

9 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.

10 Alphanumeric device type for translation. The default is 0.

For information about the values that can be specified, see the description of ASTYPE in Chapter 3, “The GDDM calls” on page 21.

---

5 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

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family 4
Length: 2 full-words

1 The option group code 5

2 Data-stream type:
- **0 (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.
- **1 (EPS)** or **(PSEG)** or **(SEC)** Produce secondary data stream (encapsulated PostScript or page segment). Not supported for the CDPU.
- **2 (OVLY)** Produce an overlay segment (secondary data stream). Not supported for the CDPU or PostScript.

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.

---

6 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
Length: 2 fullwords.

1 The option group code: 6
2 Spill file usage:
- **0 (YES)** Store internal picture description on disk in a spill file (the default)
- **1 (NO)** Store internal picture description in main storage.

---

7 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
Length: 2 fullwords.

1 The option group code: 7
2 The number of swathes to be used: The default is 1, which means generate the output image with just one pass through the internal picture description.

---

8 Output paper size
Nickname syntax: (PRTPSIZE,w,d,{TENTHS|MILLS})
Alternative syntax: (HRIPSIZE,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, defined for the device token being used.

Subsystems: TSO, VM, MVS, and VSE Batch
Devices: Family-4
Length: 4 fullwords.

1 The option group code: 8
2 The paper width in the units defined in Fullword 4
3 The paper depth in the units defined in Fullword 4
4 The units used in Fullword 2 and Fullword 3:
- **0 (TENTHS)** Units are tenths of an inch
- **1 (MILLS)** Units are millimeters.

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 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. For compatibility with previous releases of GDDM, the option name HRIPSIZE can also be used in nickname statements.

---

9 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
Length: 2 full-words

1 The option group code 9
2 Output format
- **0 (BITMAP)** Produce unformatted output.

1 (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).

2 (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.

3 (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.

---

10 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
Length: 2 fullwords.

1 The option group code: 10
2 The plotter form feed option:
- **0** Page feed (default for those devices that support page feed)
- **1 (NO)** No page feed
- **2 (YES)** Page feed.

---

11 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
Length: 2 fullwords.

Note: Refer to the information about the velocity-select (VS) instruction in the appropriate color plotter programming manual.

1 The option group code: 11
2 The pen velocity:
- **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.

---

12 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
Length: 2 fullwords.

1 The option group code: 12
2 The 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.

---

13 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
Length: 2 fullwords.

Note: Refer to the information on the pressure-select instruction in the appropriate color plotter programming manual.

1 The option group code: 13
2 The pen pressure:
- **0** The pressure, as 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.

---

14 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
Length: 5 fullwords.

1 The option group code: 14.

2 The minimum x value as a percentage of the maximum paper width. The default is 0.

3 The maximum x value as a percentage of the maximum paper width. The default is 100.

4 The minimum y value as a percentage of the maximum paper height. The default is 0.

5 The maximum y value as a percentage of the maximum paper height. The default is 100.

---

15 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
Length: 3 fullwords.

1 The option group code: 15.
2 The paper-size code:
- **0 (*)** The default (whatever paper size is loaded)
- **1** A or A4 size
- **2** B or A3 size

	3	C or A2 size
	4	D or A1 size
	5	E or A0 size.
3	The dimension-type code:	
	0 (*)	ISO dimensions (the default)
	1	ISO dimensions (A4, A3, A2, A1, or A0)
	2	ANSI dimensions (A, B, C, D, or E).

---

16 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
Length: 2 fullwords.

1	The option group code: 16	
2	The orientation value:	
	0	No rotation (the default)
	1 (NO)	No rotation
	2 (YES)	Rotate the picture by 90 degrees.

---

17 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
Length: 2 fullwords.

1	The option group code: 17	
2	Retained or unretained mode:	
	0 (YES)	Retained mode (the default)
	1 (NO)	Unretained mode.

---

18 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.

The default is a zero value in fullword 2. If this processing option is not specified or if fullword 2 is zero, 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
Length: 2+2xN fullwords.

1	The option group code: 18.
2	The number (N, in the range 0 through 2) of pairs of fullwords that follow.
3 through 2+2xN	"N" pairs of fullwords. Each pair forms an 8-byte name-part.

Note: This procopt is used to determine the values returned when you are plotting via a family-2 device. (See the FSQURY call.)

---

19 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 the *GDDM System Customization and Administration* book.

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
Length: 2 fullwords.

1 The option group code: 19
2 The default symbol sets option:
- **0 (NO)** Use the workstation's default mode-2 and mode-3 symbol sets (the default)
- **1 (YES)** Load GDDM's mode-2 and mode-3 symbol sets, replacing the device's default symbol sets.

---

20 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 alpha-numeric 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
Length: 2 + N fullwords.

1 The option group code: 20
2 The number (N, in the range 0 through 1) of fullword values that follow
3 The identification value:
- **0 (NO)** No origin identification (the default)
- **1 (YES)** Origin identification required.

---

21 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
Length: 2 fullwords.

1 The option group code: 21
2 The local interactive graphics mode option:
- **0 (NO)** Local interactive graphics mode not allowed (the default)
- **1 (YES)** Local interactive graphics mode allowed.

---

22 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
Length: 3 fullwords.

1 The option group code: 22
2 and 3 One pair of fullwords, forming an 8-byte name part.

---

23 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
Length: 2+2xN fullwords.

1 The option group code: 23.
2 The number (N, in the range 0 through 2) of pairs of words that follow.
3 and 4 One pair of fullwords specifying an 8-byte special device name:
- **'IBM5080 '** To use the IBM 5080 Graphics System for graphics
- **'* '** To turn off the use of the IBM 5080 Graphics System.

5 and 6 Information specific to this device.

For a special device name for the IBM 5080 Graphics System, there are only two fullwords of device-specific information, which are ddname or blank.

Notes:

1. The device name, ddname, is the same as the name in the command that you issue before you use the IBM 5080 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 need be attached.

---

24 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
Length: 2 fullwords.

1 The option group code: 24
2 The type of window mode:
- **0 (NO)** Not in window mode (the default)
- **1 (YES)** In window mode.

---

25 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
Length: 2 fullwords.

1 The option group code: 25.
2 The use of pseudoconversational mode:
- **0 (NO)** Do not use pseudoconversational mode (the default).
- **1 (START)** Start use of pseudoconversational mode.
- **2 (CONTINUE)** Continue use of pseudo-conversational mode.

---

26 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 “FSUPDM – Set update mode” on page 95.

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.
Length: 2 fullwords.

1 The option group code: 26
2 The type of window mode:
- **0** No degradation of picture fidelity (default)
- **1** Picture degradation acceptable using GDDM's chosen method for the picture.

---

27 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
Length: 2 fullwords.

1 The option-group code: 27.
2 The availability of fast-path mode for User Control functions that require pointings:
- **0 (NO)** Fast path mode is not selected (the default).
- **1 (YES)** Fast path mode is selected.

---

28 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
Length: 2 fullwords.

1 The option-group code: 28.
2 The availability of control mode:
- **0 (*)** User Control is available for devices not capable of supporting real partitions (the default).
- **1 (YES)** User Control is always available, forcing emulated partitions.
- **2 (NO)** User Control is not allowed.

---

29 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
Length: 3 fullwords.

1 The option-group code: 29.
2 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.

3 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.

---

30 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
Length: 2 fullwords.

1 The option-group code: 30.
2 The ability to print from the screen:
- **0 (YES)** Printing is allowed in User Control (the default).
- **1 (NO)** Printing is not allowed in User Control.

---

31 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 Chapter 18, "External defaults" on page 379), and is subsystem-dependent.

Subsystems: TSO, VM, CICS
Devices: All family-1 displays
Length: 2 fullwords.

1 The option-group code: 31.
2 The ability to save the picture:
- **0 (NO)** Saving is not allowed from User Control.
- **1 (YES)** Saving is allowed from User Control.

---

32 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
Length: 2 fullwords.

1 The option-group code: 32.
2 Inline resources supported:
- **0 (NO)** Inline resources are not supported (the default).
- **1 (YES)** Inline resources supported.

---

33 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
Length: 2 fullwords.

1 The option-group code: 33
2 GDDM-PCLK availability:
- **0 (NO)** GDDM-PCLK not available (the default)
- **1 (YES)** GDDM-PCLK available.

---

34 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
Length: 2 fullwords.

1 The option-group code: 34
2 Device code-page:
- **n** The global code-page identifier

---

35 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
Length: 2 fullwords.

1 The option-group code: 35
2 Print quality:
- **0 (*)** Printer hardware setting (the default)
- **1 (DP or DPQ)** Data-processing quality text, high-density graphics, and high contrast bar codes
- **2 (DPT or DPTQ)** Data-processing text quality text, high-density graphics, and high contrast bar codes
- **3 (NLQ)** Near-letter quality text, high-density graphics, and high contrast bar codes
- **4 (LLL)** Data-processing quality text, low density graphics, and low contrast bar codes
- **5 (MLL)** Data-processing text quality text, low-density graphics, and low contrast bar codes
- **6 (HLH)** Near-letter quality text, low-density graphics, and high contrast bar codes
- **7 (LLH)** Data-processing quality text, low-density graphics, and high contrast bar codes.

---

36 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
Length: 2 fullwords.

1 The option-group code: 36
2 Encoded data fields to be displayed:
- **0 (NO)** Encoded data fields to be nondisplayable (the default)
- **1 (YES)** Encoded data fields to be displayed.

(PCLKEVIS,YES) must be used with GDDM-PCLK if your terminal emulator normally discards nondisplayable characters.

---

37 Rotation of print output
Nickname syntax: (PRTROT,{**0**|90|180|270})
Alternative syntax: (IPDSROT,{**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
Length: 2 fullwords.

1 The option group code: 37
2 The clockwise rotation, in degrees:
- **0 (0)** Portrait – top of page is drawn at upper edge of paper (the default)
- **1 (90)** Landscape – top of page is drawn at right edge of paper
- **2 (180)** Portrait – top of page is drawn at lower edge of paper
- **3 (270)** Landscape – top of page is drawn at left edge of paper.

Notes:

1. For compatibility with previous releases of GDDM, the option name IPDSROT 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.

---

38 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
Length: 2 fullwords.

1 The option group code: 38
2 Data stream truncation:
- **0 (NO)** Do not truncate the data stream (the default for 4234)
- **1 (YES)** Truncate the data stream (the default for other IPDS printers).

---

39 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
Length: 2 fullwords.

1 The option group code: 39
2 The number of lines to the inch to be printed:
- **0 (8)** Print at 8 lines to the inch (default)
- **1 (6)** Print at 6 lines to the inch.

---

40 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
Length: 3 fullwords.

1 The option group code: 40
2 Paper feed bin for main document
- Can be in the range 0 through 100

3 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.

---

41 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
Length: 2 fullwords.

1 The option group code: 41.
2 Swathing control:
- **0 (YES)** Data is sent to the printer as a series of compressed subimages (swathed). This is the default.
- **1 (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.

---

42 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
Length: 2 fullwords.

1 The option group code: 42
2 Initial value:
- **0 (BLACK)** Black (the default)
- **1 (WHITE)** White
- **2 (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.

---

43 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
Length: 2 Fullwords.

1 The option group code: 43
2 The number of print characters per ten inches:

- **100 (100)** 10.0 characters per inch (the default)
- **120 (120)** 12.0 characters per inch
- **133 (133)** 13.3 characters per inch
- **150 (150)** 15.0 characters per inch
- **167 (167)** 16.7 characters per inch
- **170 (170)** 17.0 characters per inch
- **180 (180)** 18.0 characters per inch
- **200 (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.

---

44 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.
Length. 3 fullwords.

1 The option group code: 44.
2 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.
3 The table number to be used for triplane (color) shading patterns. The format of this table number is as above.

Three tables are provided by GDDM:

ADM00001 Translates user-defined patterns 65 through 254 to system patterns 1 through 16 in a repeated fashion, with no change in color. Patterns 65 through 80 are translated to 1 through 16 respectively, patterns 81 through 96 are translated to 1 through 16, and so on; finally patterns 241 through 254 are translated to 1 through 14.

ADM00002 Translates user-defined patterns 65 through 128 in symbol set ADMPATTC to similar-looking system patterns. Where two or more user-defined patterns are translated to the same system pattern, a different color distinguishes between them. User-defined patterns 129 onwards are translated to system patterns as in table ADM00001.

ADM00003 Translates the user-defined patterns to combinations of colors and system-defined patterns to simulate the use of ADMCOLSD. Each user-defined pattern is translated to a unique combination of a color in the range 1 through 256, and one of the system-defined patterns. This is suitable for devices using display adapters which support 256 or more colors (8514/A, for example), running under GDDM-OS/2 Link.

---

45 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
Length: 3 fullwords.

1 option group code: 45
2 Type of key selected:
- **0** The ENTER key (default)
- **1** PF key (see value below)
- **4** PA key (see value below)

3 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.

---

46 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
Length 3 fullwords.

1 option group code: 46
2 Specifies whether or not plotter output is to be stored in a file
- **0 (NO)** Plotter output is not to be stored in file but drawn on an attached plotter device (default).
- **1 (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.

3 Specifies whether the new file can overwrite an existing file with the same name.
- **0 (REP)** Overwrite existing file (default).
- **1 (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.

---

47 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
Length: 2 fullwords.

1 The option group code: 47
2 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.

---

48 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)
Length: 2 fullwords.

1 The option group code: 48
2
- **0 (NO)** All characters and lines are drawn in black. (Default)
- **1 (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**.

---

49 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)
Length: 2 fullwords.

1 The option group code: 49
2
- **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.

---

50 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.
Length: 2 fullwords.

1 The option group code: 50
2
- **xxxxxxxx** The name of a procedure or program that is to perform post processing on the PostScript 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.

---

51 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
Length: 2 fullwords.

1 The option group code: 51
2 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.

1000 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
Length: 2 fullwords.

1 The option group code: 1000
2 The type of PA1/PA2 protocol:
- **0 (PA1PA2)** PA1 causes entry to CP mode; PA2 causes entry to CMS subset mode (default).

1 (PA2) PA1 is returned to the application; PA2 causes entry to CMS subset mode.
2 (PA1) PA1 causes entry to CP mode; PA2 is returned to the application.
3 (NONE) PA1 and PA2 are returned to the application.

1001 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
Length: 4 fullwords.

This option group always contains four fullwords. (If basic attention handling is requested, the third and fourth fullwords must still be present even though they are not inspected.)

1 The option group code: 1001.
2 The type of attention handling:

0 (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.

1 (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 by this option group. However, if either the length or the address of the block is zero, the feedback block is not filled in.

3 The length of the attention feedback block. See the description of extended attention handling, above (Fullword 2).
4 The address of the attention feedback block. See the description of extended attention handling, above (Fullword 2).

1002 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. The default is a zero value in fullword 2. If this processing option is not specified or if fullword 2 is zero, no CP SPOOL command is issued.

Subsystems: VM
Devices: Family-1 device 'PUNCH'
Family-4 device 'PRINTER'
Length: 2+2xN fullwords.

1 The option group code: 1002
2 The number (N, in the range 0 through 16) of pairs of fullwords that follow
3 – 2+2xN “N” pairs of fullwords, giving the appropriate spooling information as 8-character tokens.

1003 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. The default is a zero value in fullword 2. If this processing option is not specified or if fullword 2 is zero, no CP TAG command is issued.

Subsystems: VM
Devices: Family-1 device 'PUNCH'
Family-4 device 'PRINTER'
Length: 2+2xN fullwords.

1 The option group code: 1003
2 The number (N, in the range 0 through 16) of pairs of fullwords that follow
3 – 2+2xN "N" pairs of fullwords, giving the appropriate routing (tag) information as 8-character tokens.

1004 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
Length: 2 fullwords.

1 The option group code: 1004
2 Print utility control:
0 (NO) Do not invoke print utility (the default)
1 (YES) Invoke print utility automatically.

2000 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.

For a more detailed discussion of the use of the PA1 and CLEAR keys in an TSO environment, see the *GDDM System Customization and Administration* book.

Subsystems: TSO
Devices: Family-1
Length: 2 fullwords.

1 The option group code: 2000.
2 The type of attention handling:
0 (PA1) PA1 causes attention, CLEAR is ignored (TSO default action)
1 (NONE) PA1 and CLEAR are returned to the GDDM application (PA1 does not cause an attention).

2001 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
Length: 2 fullwords.

1 The option group code: 2001.
2 The keys treated as "reshow" AIDs:
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.

2002 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
Length: 2+2xN fullwords.

1 The option group code: 2002.

2 The number (N, in the range 1 through 2) of pairs of fullwords that follow.

3 and 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.

5 and 6 An 8-character token containing one of:
- **destname** The JES Remote workstation name, associated through JES/328X, with the required target printer.
- **ddname** The ddname of a DD statement describing the output data set to be used.
- ***** Output is to go to the ADMPRINT queue. Not valid for Family 4.
- **=** 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.

   See the section on the format of the nickname UDS in the *GDDM System Customization and Administration* book for more information.

2003 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
Length: 2 fullwords.

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.

3000 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 processing option 9 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
Length: 2 fullwords.

1 The option group code: 3000.

2 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 on color separations, see the *GDDM Base Application Programming Guide*.

Chapter 20. Name-lists

This chapter describes the **name-list** values that can be specified for each subsystem and for each GDDM device family.

A **name-list** is a means of identifying which physical device is to be opened for use by a GDDM application program. It can be a parameter of the DSOPEN call or it can be specified as a nickname. The naming convention of the **name-list** varies according to the subsystem and device family in use.

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 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 giving a length of 0 in DSOPEN)
- By setting all name-parts to "*".

Also, (under CICS, TSO, or VMS), the name-list parameter 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 GDDM-PCLK-attached printer; see the *GDDM User's Guide*.
4. Use ADMPMOP for the GDDM-OS/2 Link 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 (DCT).

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

Applicable to family-4 files only.

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

Note: If output has been redirected using the TOFILE procopt, the name-list format will be different from the description in this section. Refer to page 409.

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).

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...'
```

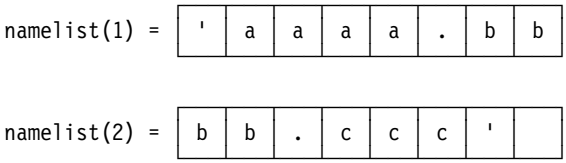
where qualifier(s) is the active dsn-prefix, or user ID, or both of these.

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.

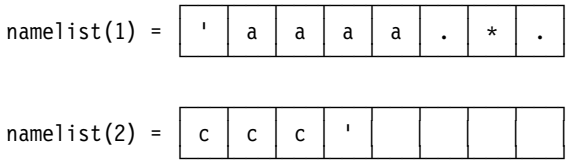
If contained in quotes, the derived names are taken as complete DSNAMEs. If not contained in quotes, they are formed into complete DSNAMEs of the form "qualifier(s).name1.name2..." (where qualifier(s) is the active dsn-prefix, or user ID, or both of these).

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

Note: If output has been redirected using the TOFILE procopt, the name-list format will be different from the description in this section. Refer to page 409.

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".

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

Note: If output has been redirected using the TOFILE procopt, the name-list format will be different from the description in this section. Refer to page 409.

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 the *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).

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.

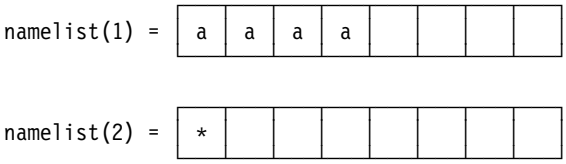
Monochrome master

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
```

Chapter 21. Device characteristics tokens

This chapter describes the device characteristic tokens (usually called device tokens) supplied by GDDM. Many Input/Output tasks in GDDM can be done without using device tokens at all; however, there are some instances where they should be used. For example:

- Under IMS/VS, to define the database that links the characteristics of terminals with logical terminal names, and so determine the type of data stream that GDDM sends.
- To specify some special types of device, such as AFPDS printers, in a DSOPEN call.
- To override the information obtained by GDDM about a particular device so that device information is taken from the token rather than from the device itself, which is the usual source.

Notes:

1. The paper sizes quoted in this chapter generally refer to the usable area, not the actual sheet size.
2. Tokens for some obsolete devices have been omitted from the tables in this chapter. These tokens are still available in the ADMLSYS*n* modules.

Creating your own device tokens

The GDDM-supplied device tokens are designed to cater for most requirements. For further information, refer to the *GDDM System Customization and Administration* book.

Device tokens for ASCII graphics displays

Device tokens are required to define the characteristics of ASCII graphics displays. GDDM normally selects the relevant token based on information received from the 3174 controller attached to the ASCII graphics display. Alternatively, a token may be specified in a DSOPEN call or by means of a nickname. ASCII device tokens are contained in the module ADMLSYSA and any extra ones supplied at installation must be added to this module.

GDDM-supplied device tokens

The GDDM-supplied device tokens are shown in the tables below.

Note: Your installation may have changed the GDDM-supplied device tokens or created tokens of their own.

The meanings of the tokens are shown as the macro definitions used to create them. You may need to study the meanings of the macro operands to understand the tokens. The operands are explained in the *GDDM System Customization and Administration* book.

The GDDM-supplied device tokens are grouped as follows:

- Table 47: queriable terminals, plotters, and printers (family 1 or family 2)
- Table 48 on page 425: Kanji devices, and 8775 and 3290 displays (family 1)
- Table 49 on page 426: ASCII devices (family 1)
- Table 50 on page 426: Nonqueriable diplays and printers
- Table 51 on page 427: GDDM-PCLK displays, printers, and plotters (family 1)
- Table 52 on page 428: system printers (family 3)
- Table 53 on page 428: AFPDS printers (family 4)
- Table 54 on page 429: PostScript printers (family 4)
- Table 55 on page 430: page printers (family 4).

Table 47 (Page 1 of 4). GDDM-supplied device tokens. Device tokens are provided for queriable terminals, plotters, and printers (family 1 or family 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

Table 47 (Page 2 of 4). GDDM-supplied device tokens. Device tokens are provided for queriable terminals, plotters, and printers (family 1 or family 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 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 424.

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 424.

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	*Plotter*
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 424.)

Device token
L3179A71
L3179A72
L3179A80
L3179A82
L3179A84
L3179A85
L3179A86

Table 47 (Page 3 of 4). GDDM-supplied device tokens. Device tokens are provided for queriable terminals, plotters, and printers (family 1 or family 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.*

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*
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 424.

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)
S3812L	LU-1 (SCS)	112 by 85	Legal

3816 IPDS printer with 3270 attachment feature.

Device token	*Protocols*	*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

Table 47 (Page 4 of 4). GDDM-supplied device tokens. Device tokens are provided for queriable terminals, plotters, and printers (family 1 or family 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.*

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
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.

Table 48. GDDM-supplied 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 (non-graphics)

Device token	*Model*	*Protocols*
L5550A	5550 display	
L5553A	5553 printer	LU-3
L5553A1	5553 printer	LU-1 (SCS)
L5550G4	3270-PC Version 4.0	
L5550H4	3270-PC Version 4.0 color	
L5553B34	5553 printer	LU-3
L5553B14	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 cols*	*cell size*
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

8775 terminals with partitions and scrolling

Device token	*Model*
ADMK7510	model 1
ADMK7520	model 2
ADMK7530	model 3
ADMK7540	model 4

8775 terminals with partitions and programmed symbols

Device token	*Model*
ADMK751S	model 1
ADMK752S	model 2
ADMK753S	model 3
ADMK754S	model 4

Table 49. GDDM-supplied device tokens for support of 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 cols*	*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

Table 50. GDDM-supplied device tokens for nonqueriable displays and printers (family 1). This set of token definitions is part of ADMLSYS1.

3277 displays	
Device token	*Model*
ADMK7710	3277 Model 1
ADMK771A	3277 Model 1 with APL
ADMK7720	3277 Model 2
ADMK772A	3277 Model 2 with APL
3277 displays	
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	
ADMKQUEP	default token for family-2 (queued) printers
ADMK8710	3287 Model 1
ADMK871A	3287 Model 1 with APL

Table 51. GDDM-supplied device tokens for GDDM-PCLK displays, printers, and plotters (family 1). 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

Table 52. GDDM-supplied 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

Table 53 (Page 1 of 2). GDDM-supplied 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, refer to the GDDM System Customization and Administration *book.*

3800 printer at 240 pixels per inch: IM. See note 1 on page 429.

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 pixels per inch: 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 pixels per inch: 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 pixels per inch: GOCA, IOCA. See note 2 on page 429.

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

Table 53 (Page 2 of 2). GDDM-supplied 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, refer to the GDDM System Customization and Administration *book.*

3835 printer at 240 pixels per inch: GOCA, IOCA. See note 2.

Device token	*Rows by columns*	*Pixels*	*Lines per inch*
A3835S	60 by 110	1800 by 2640	8
B3835S	45 by 110	1800 by 2640	6

3112, 3116, 3912, 3916, and 4028 printers at 300 pixels per inch: 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 pixels per inch: 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.

Table 54 (Page 1 of 2). GDDM-supplied 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 of the ADMMPSCR macro in the *GDDM System Customization and Administration* book.

Device token	*Paper size - inches width by depth*	*Resolution pixels per inch*	*Type*	*Paper*
Generic level-1 monochrome printer				
PPS1MA4	8.0 × 11.7	300	PS1M	A4
PPS1MQ	8.5 × 11.0	300	PS1M	Quarto
Generic level-2 monochrome printer				
PPS2MA4	8.0 × 11.7	300	PS2M	A4
PPS2MQ	8.5 × 11.0	300	PS2M	Quarto
Generic level-1 color printer				
PPS1CA4	8.0 × 11.7	300	PS1C	A4
PPS1CQ	8.5 × 11.0	300	PS1C	Quarto
Generic level-2 color printer				
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

Table 54 (Page 2 of 2). GDDM-supplied device tokens for PostScript printers (family 4). This set of token definitions is part of ADMLSYS4.

4029 monochrome printer				
P4029A4	8.0 × 11.7	600	PS1M	A4
P4029Q	8.5 × 11.0	600	PS1M	Quarto
4079 color printer				
P4079A3	8.0 × 16.0	360	PS1C	A3
P4079B	11.0 × 17.0	360	PS1C	B

Table 55. GDDM-supplied 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.

Device token	*Paper size - inches width by depth*	*Resolution pixels per inch*	*Pixels per unit line width*	*Paper*
3800, 3812, 3816, or 3820 printer				
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, and 4028 printers				
IMG300A4	8.0 × 11.3	300	3	A4
IMG300Q	8.1 × 10.6	300	3	Quarto
IMG300L	8.1 × 13.6	300	3	Legal
4224 and 4230 printers				
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 and 2.				
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 and 4.

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 (bitmap) 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.

Chapter 22. Special-purpose programming in GDDM

The System Programmer Interface (SPI) is provided for programmers who want to use GDDM as the basis for a graphics system of their own. It enables GDDM functions to be written in a coded form, it gives greater control over the subsystem environment, and it allows greater programming flexibility within the subsystem environment.

This chapter describes:

- "Using the system programmer interface," below,

and

- "Specifying user exits" on page 432.

Using the system programmer interface

The system programmer interface is a special interface available to "system programming" types of applications. It is available only in reentrant form, and shares many features with the application-programmer reentrant interface. The reentrant interfaces are described in Chapter 1, "GDDM programming interface" on page 1.

In the simplest case, the system programmer interface merely provides a means of accessing a GDDM function by a function code (the Request Control Parameter, RCP) rather than by selecting an entry point. Assembler-language macros defining mnemonics for these function codes are provided.

Each call takes the form:

```
CALL ADMASP (aab,rcp,component parameters,...)
```

where `ADMASP` is the defined system programmer interface entry point. ADMASP is a single entry point resolved by the GDDM interface modules that are link-edited with the application.

Note: The sample PL/I declarations do not include this entry point, because it can only be called using the system programmer interface. The PL/I application programmer using this call must, therefore, supply an entry-point declaration for the system programmer interface, as described in Chapter 3, "The GDDM calls" on page 21. For example:

```
DECLARE ADMASP EXTERNAL ENTRY OPTIONS (ASM,INTER);
```

Parameters

aab (*specified by user***) (***8-byte control block***)**
An Application Anchor Block, as described in Chapter 1, "GDDM programming interface" on page 1.

rcp (*specified by user***) (***fullword integer***)**
The Request Control Parameter, a 4-byte, fullword-aligned function code defining the GDDM function to be called. The GDDM RCP code is given, for each GDDM call listed and described in Chapter 3, "The GDDM calls" on page 21, in both hexadecimal and decimal format. Also, the *GDDM Diagnosis* book contains a table defining the RCP codes for all GDDM calls.

RCP codes are the function codes to be specified when invoking GDDM and GDDM-PGF by means of the system programmer interface (SPI). The codes are also set by GDDM in error records passed to user error exits, or produced in response to FSQERR calls.

Assembler-language tables ADMURCPB and ADMURCPO define RCP codes. Table ADMURCPB defines the RCP codes for GDDM only. ADMURCPO defines the RCP codes for both GDDM and GDDM-PGF. (On CMS these tables are in ADMLIB MACLIB; on MVS they are in the SADMSAM data set; and on VSE they are supplied as member type Z.)

The RCP codes are defined as symbolic Assembler-language EQUATE statements of mnemonics (QQxxxxxx) to numeric values. The "xxxxxx" of the mnemonic is the name of the GDDM or GDDM-PGF function; for example,

```
QQFSINIT EQU X'0C000001'
```

component parameters
The parameters for the function specified in the RCP. These are as described for the specific function being called.

Calls to the system programmer and reentrant interfaces can be mixed, provided that the same application anchor block is passed on each call.

Initialization

This interface provides an alternative initialization function (known as SPINIT) that allows control of environmental aspects. SPINIT is an alternative to FSINIT and, if used, must be the first GDDM statement to be run.

Note that your program would not use an explicit call to an entry point called SPINIT. Instead, like all other system programmer interface calls, you would code a call to ADMASP. The function is described for consistency as a SPINIT call, as it behaves like the other GDDM calls, but it can only be specified through the system programmer interface. The GDDM Assembler language tables ADMURCPB and ADMURCPO include the mnemonic QQSPINIT.

For the syntax of the SPINIT call, see Chapter 3, "The GDDM calls" on page 21. The details of the **spib-block** parameter are listed below.

Table 56. SPIB format

Offset (hex)	Length (bytes)	Label	Usage
0	4	SPIBHEAD	Spare. Reserved for the application program to use as an eye-catcher.
4	4	SPIBLENG	Length of SPIB.
8	4	SPIBUDSL	Length of user default specification list.
C	4	SPIBUDSP	Address of user default specification list.
10	4	SPIBGSXP	Address of application-defined GET STORAGE exit.
14	4	SPIBGSXK	User-defined parameter to be passed to the application program's GET STORAGE exit.
18	4	SPIBFSXP	Address of application-defined FREE STORAGE exit.
1C	4	SPIBFSXK	User-defined parameter to be passed to the application program's FREE STORAGE exit.

The system programmer interface block

The system programmer interface block (SPIB or SPIB-block) is a table giving control information. The contents of this table are processed by GDDM during initialization. Subsequent changes to the contents do not affect GDDM processing. The storage containing the table can be released after initialization has been completed.

Note: Since Version 1 Release 4, GDDM supports an abbreviated format of the SPIB. This is described below. A number of the functions that were previously available in the SPIB are now available through other GDDM calls, which can be issued immediately after the SPINIT call. For example, the functions of the SPIBOPNF, SPIBPA2F, SPIBXFBF, SPIBXFBL, and SPIBXFBP fields can now be specified as DSOPEN processing options; the functions of many other fields can be specified as input to the SPIB by means of items in a user default specification list.

The previous format of the SPIB is retained for reasons of compatibility; it does not contain or provide access to new function provided since GDDM Version 1 Release 4. It is described in the edition of the *Base Programming Reference* manual for the Release of GDDM for which your program was written.

Format of the system programmer interface block

The labels are defined here in more detail:

SPIBLENG
Specifies the length of the SPIB. Must be in the range 16 through 32, which identifies this as a GDDM Version 1 Release 4 SPIB. The fields after offset X'10' can be omitted (and thus allowed to default) by specifying the minimum value of 16.

SPIBUDSL
Specifies the length (in bytes) of an encoded user default specification list (UDSL). Must be set to 0 if no UDSL is to be passed.

SPIBUDSP
Specifies the address of an encoded user default specification list (UDSL). Must be set to 0 if no UDSL is to be passed.

SPIBGSXP (TSO, MVS batch, VSE batch, and VM/CMS)
Specifies (if present and if not zero) the address of an application-defined storage exit to be called for GET STORAGE requests.

SPIBGSXK (TSO, MVS batch, VSE batch, and VM/CMS)
Specifies (if present) a user-defined parameter that GDDM is to pass when calling a GET STORAGE exit.

SPIBFSXP (TSO, MVS batch, VSE batch, and VM/CMS)
Specifies (if present and if not zero) the address of an application-defined storage exit to be called for FREE STORAGE requests.

SPIBFSXK (TSO, MVS batch, VSE batch, and VM/CMS)
Specifies (if present) a user-defined parameter that GDDM is to pass when calling a FREE STORAGE exit.

The interface specifications for GDDM storage exits are described under "Storage exit routines – interface specifications" on page 437.

Specifying user exits

User exits allow a system program to trap specific events whenever an application program uses a GDDM or system resource. Such events include task switching in TSO, intercepting some or all GDDM calls, and so on.

A limited number of user exits can be specified using User Default Specifications (UDSs). UDSs are described in the *GDDM System Customization and Administration* book. The user exits are:

- A Task Switch exit
- A Call Intercept exit
- A Coordination exit.

Table 57. GDDM exits — options

Meaning of default	Source syntax of the ADMMEXIT macro options	Encoded values - list of fullwords	Valid in: M F S C
Call intercept user exit address	CALLINT=(addr)	3,3005,A(CI-UX)	N N Y N
Call intercept user exit token value	CALLINT=(,token)	3,3006,CI-token	N N Y N
Default user exit address	DEFAULT=(addr)	3,3001,A(DFT-UX)	N N Y N
Default user exit token value	DEFAULT=(,token)	3,3002,DFT-token	N N Y N
Task switch user exit address (TSO only)	TASKSWI=(addr)	3,3003,A(TSW-UX)	N N Y N
Task switch user exit token value (TSO only)	TASKSWI=(,token)	3,3004,TSW-token	N N Y N

Note: In the source-format forms, corresponding pairs can be combined in this way: DEFAULT=(address,token).

This section describes how you specify user exits, the conventions that your exits must follow, and the function of each type of exit.

It also describes the storage exit routines that can be defined by using the System Programmer Interface Block (SPIB) in the SPINIT call. For more information about the SPIB, see “Initialization” on page 431.

Table 57 shows the defaults that you can specify for GDDM exits using the SPINIT call. The figure also describes the corresponding user default specifications (in source and encoded format). These UDSs must be passed to GDDM using the SPINIT call in the form of an encoded-UDS list. The last column shows where the UDS can be specified, as follows:

M in the External Defaults Module,
F in the External Defaults File,
S in the SPINIT call,
C in the ESEUDS and ESSUDS calls.

Exit values

The descriptions of these options are:

CALLINT=(address,token)

address gives the fullword address of the Call Intercept exit.

token provides four bytes of data that are passed from the application program to the exit.

DEFAULT=(address,token)

address gives a fullword address for all user exits. Specifying an address in this option is equivalent to specifying it for each user exit explicitly.

token provides four bytes of data that are passed from the application program to any exit. Specifying a token in this option is equivalent to specifying it for each user exit explicitly.

TASKSWI=(address,token)

address gives the fullword address of the Task Switch exit.

token provides four bytes of data that are passed from the application program to the exit.

GDDM user-exit conventions

Unless otherwise noted, user exits defined by means of UDSs must conform to these rules:

- The contents of the registers on entry to the exit are:

```
R13 -> A 72-byte save area
R14 -> The return address
R15 -> The entry point of the exit
R1  -> The parameter address list, in standard
       variable-list format:
       ADDR1 -> AAB     (Char(8))
       ADDR2 -> UXBLOCK ((3) Fixed(31))
       .
       .
       additional parameters as defined for the
       specific exit
```

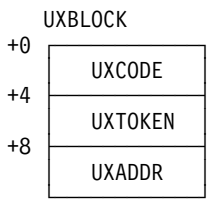
AAB

The application’s AAB (application anchor block) (or in the case of the coordination exit, the GDDM-provided dummy AAB if the application is using the nonreentrant interface).

The exit must not use the AAB to issue a GDDM call. That is to say, the GDDM instance that caused the exit must not be entered recursively.

UXBLOCK

A user-exit control block of this format:



The contents of UXBLOCK are:

UXCODE

The fullword user-exit code. This code is the same as the UDS-code used to define the user exit address. The exit must **not** change this parameter.

UXTOKEN
The fullword user-exit token. This field is initialized to 0. An explicit value for this token can be specified when the exit is specified. The exit or application program can change this parameter; in which case, subsequent calls to the exit are passed in the changed parameter.

UXADDR
The fullword user-exit address. On entry to an exit, this parameter has the same value as R15 (the address of the exit entry point). The exit can change this parameter; in which case, subsequent calls to the exit are to the new address. If the address is set to 0, GDDM stops using the exit for as long as the address remains 0. If the address is subsequently reset to nonzero (by the application program or by another exit), GDDM resumes invocation of the exit.

- The parameter address list is in variable parm-list format (that is, with the high-order bit of the last address word set to "1"), and GDDM may pass parameters in addition to those defined below. Therefore, the exit must **not** rely on the high-order bit of a specific parameter address word always being set to "1."
- Unless otherwise noted, the exit must not modify any parameter passed to it. (The only exception is the UXBLOCK parameter.)
- On return, the exit must set R15 to one of the specified completion codes.

  If any other value is returned, the results are undefined (nonzero values may be diagnosed, ignored, or abended).
- It is recommended that you make the exit **reentrant** and **read-only**. Otherwise, careful thought must be given as to how the operation of GDDM and its calling application(s) is affected.
- The exit must conform to standard System/370 calling conventions (including the use of save areas and restoring registers).
- Under MVS/XA, MVS/ESA, or VM/XA, the exit must be AMODE(ANY); that is, it must be prepared to accept control in 24-bit or 31-bit mode, and must return control in the same mode. If called in 31-bit mode, all addresses (including R13) must be treated as 31-bit addresses and may be greater than 16 megabytes.

  Under MVS/XA, MVS/ESA, or VM/XA, a 24-bit mode application program must ensure that the top byte of an initial value for a user-exit token is cleared to zero if it intends that this token is to be interpreted as an address. GDDM considers this token to be a FIXED(31) variable, and does **not** clear the top byte of the token before invoking the exit.

The task switch exit

A Task Switch exit can be defined in an ADMMEXIT UDS. This exit is valid under TSO only. If it is specified in other environments, the results are undefined.

Function: By providing a Task Switch exit, an TSO tasking application program can call GDDM both from its main task and from any of a number of subtasks. The Task Switch exit should be coded to switch to a standard task (typically, the main task) under which specific subsystem-dependent task-sensitive functions can be performed.

If enabled, the Task Switch exit is invoked before GDDM performs selected task-sensitive functions. The Task Switch exit has passed to it the address of a GDDM subroutine to be called after switching tasks, plus the parameters to be passed to the routine.

The Task Switch exit is returned to when the GDDM subroutine has performed the task-sensitive functions. The Task Switch exit should then switch tasks back before returning to GDDM.

The system-dependent functions that are "task protected" in this manner are:

- Explicit GETMAIN and FREEMAIN requests. (Indirect requests by means of storage exits or other system-dependent functions are **not** "task protected.")
- DASD OPEN and CLOSE requests. (READ, WRITE, PUT, and GET requests are **not** "task protected.")
- Explicit LOAD and DELETE requests.

Exceptionally, some of the GETMAIN, FREEMAIN, LOAD, and DELETE requests that are issued by GDDM routines at initialization and termination are not "task protected." These requests should be separately "task protected" by the application program, by ensuring that the GDDM FSINIT (or SPINIT) and FSTERM calls are always issued from the standard task.

A Task Switch exit should be prepared to be invoked in a recursive manner in some circumstances. For example:

```
GDDM invokes the Task Switch exit before OPEN.
--> The Task Switch exit calls a GDDM subroutine.
------> The OPEN macro is called, resulting in an
        OPEN error.
----------> The DCB ABEND exit receives control.
----------> Diagnostic processing is initiated.
----------> GDDM invokes the Task Switch exit
            before a LOAD for diagnostic routines.
--------------> The Task Switch exit calls a GDDM
                subroutine.
------------------> The LOAD macro is called for
                    diagnostic routines.
------------------> The subroutine returns to the
                    Task Switch exit.
--------------> The Task Switch exit returns to
                GDDM after the LOAD.
----------> Diagnostic processing completes.
----------> The DCB ABEND exit returns to NSI
            after the OPEN.
------> The OPEN macro completes.
------> The subroutine returns to the Task Switch
        exit.
--> The Task Switch exit returns to GDDM after
    the OPEN.
```

However, a Task Switch exit can prevent such recursion by disabling itself on entry, by setting the UXADDR field in the UXBLOCK parameter to 0. GDDM still ensures a return through the Task Switch exit, which should then reset the UXADDR field to the address of its entry point, before returning to GDDM.

How to specify a task switch exit: A Task Switch exit is specified as follows:

```
ADMMEXIT TASKSWI=([address][,token])
```

Parameters: The parameters for Task Switch exits are as follows:

```
R13 -> A 72-byte save area
R14 -> The return address
R15 -> The entry point of the exit
R1  -> The parameter address list, in standard
       variable parm-list format:
             ADDR1 -> AAB     (Char(8))
             ADDR2 -> UXBLOCK ((3) Fixed(31))
             ADDR3 -> SUBADDR (Ptr(31))
             ADDR4 -> SUBPARM (Format reserved to
                               GDDM)
```

Parameters `AAB` and `UXBLOCK` are described under “GDDM user-exit conventions” on page 433. Additional parameters are as follows:

SUBADDR
: The address of the GDDM subroutine to be called after switching tasks.

  The GDDM subroutine must be called according to full System/370 calling conventions. Specifically, Register 13 on entry to the subroutine must locate a register save area, which must **not** be the same as that passed to the exit by GDDM. Also, Register 1 on entry to the subroutine must be the same as was passed to the exit by GDDM.

  The GDDM subroutine saves and restores the exit's registers as normal, but does **not** necessarily conform to other System/370 calling conventions.

  On return from the subroutine, the exit must return to GDDM according to full System/370 calling conventions. Specifically, the exit **must** reload Register 14 from GDDM's save area in order to return. The exit must **not** rely on the contents of GDDM's save area being the same as on entry (specifically, all saved registers, including Register 14, and the RSA forward chain, may have been modified by the GDDM subroutine).

SUBPARM
: Additional parameter(s) that may be supplied by GDDM, for the use of the GDDM subroutine.

  The exit should not assume the existence of, nor try to examine, these parameters. The exit should call the GDDM subroutine with Register 1 locating the **same** parameter address-list as that passed to the exit by GDDM.

The exit must be AMODE(ANY); that is, it must be prepared to accept control in 24-bit or 31-bit mode, and must return control in the same mode. Also, it must call the GDDM subroutine in the same mode.

Feedback values: On return, the exit must set R15 as follows:

0 Successful completion.

The call intercept exit

A Call Intercept exit may be defined by using an ADMMEXIT UDS. This exit is valid in all environments.

Function: The Call Intercept exit provides a mechanism whereby a controlling process can monitor the calls issued by an application program. Other than for its specification by means of the SPIB, this exit is transparent to an application program at the API.

The Call Intercept exit is invoked from within GDDM, **before** each application-program call is processed (though after some housekeeping has been performed). Application-program calls that are grossly in error may be rejected without giving control to the exit.

The exit has passed to it the parameters provided by the application program. It cannot change the request or the parameters, but it can have some control over the subsequent execution, as described below.

The exit could operate in a pass-through mode, whereby it passes the specified requests through to a secondary instance of GDDM that had been separately initialized. In this mode, the exit could change or add more calls to the

secondary instance of GDDM in response to a single call from the application program. However, in this mode the exit may have difficulty passing-back error diagnostics from the GDDM secondary instance.

How to specify a call intercept exit: The Call Intercept exit is specified as follows:

```
ADMMEXIT CALLINT=([address][,token])
```

Parameters: The parameters for the Call Intercept exit are as follows:

```
R13 -> A 72-byte save area
R14 -> The return address
R15 -> The entry point of the exit
R1  -> The parameter address list, in standard
       variable parm-list format:
         ADDR1 -> AAB     (Char(8))
         ADDR2 -> UXBLOCK ((3) Fixed(31))
         ADDR3 -> RCP     (Fixed(31))
         ADDR4 -> NPARMS  (Fixed(31))
         ADDR5 -> PLIST(NPARMS) (Array of Ptr(31))
```

Parameters AAB and UXBLOCK are described under "GDDM user-exit conventions" on page 433. Additional parameters are as follows:

RCP The RCP code defining the call issued by the application program.

NPARMS The number of functional parameters provided by the application program (excluding the AAB for RACI, and the AAB and RCP for SPI).

PLIST(NPARMS)
The addresses of the functional parameters provided by the application program. These addresses are **not** in variable parameter-list format. These addresses should be treated as read-only. ADDR5 is undefined (and hence PLIST is not addressable) if NPARMS = 0.

Feedback values: On return, the exit must set R15 as follows:

0 GDDM is to continue processing the call
8 GDDM is to ignore the call, with no message
12 GDDM is to reject the call, issuing the message:

```
ADM0056 E  REQUEST REJECTED BY USER EXIT.
           REASON n
```

If R15 = 12, the exit should set R0 as follows:

n The reason-code to be inserted into message ADM0056.

Otherwise, R0 should be restored to its value on entry.

The coordination exit

A coordination exit can be defined by specifying the coordination exit address in the **array** parameter of the WSCRT call; for a description of this, see Chapter 3, "The GDDM calls" on page 21.

Function: By providing a coordination exit when creating an operator window, a task manager allows the use of that window by independent applications running their own instances of GDDM.

Whether a GDDM instance is being used by a task manager, or by a single application, the basics of a windowing program are the same:

- The first DSOPEN in a GDDM program opens the real display device with the (WINDOW,YES) processing option. This automatically creates a default operator window, and associates the real display device with it.
- The program then divides the screen of the real display device into one or more operator windows.
- Subsequent DSOPEN calls open one or more virtual display devices and associate each with an operator window. (Under a task manager, the subsequent DSOPENs would be in each application.)
- Each application (under a task manager) or each function (under a single application) then communicates with the terminal user through an operator window conceptually situated in front of a virtual screen, and can behave as if it had complete control of a real screen.

A virtual device can itself be opened with the (WINDOW,YES) processing option. Operator windows created for this virtual device are further subdivisions of the real screen. So, although you can conceptually define hierarchies of operator windows, they do not appear inside each other. Rather, they are displayed as peers, according to their priorities.

When you first create a number of overlapping operator windows in an application, the viewing order depends on the order that you create the operator windows in. The operator window that you create first is at the bottom of the viewing order, and the operator window that you create last is at the top. On the display screen, each operator window appears in front of the operator windows that are below it in the viewing order. The topmost window is the active operator window.

In a single application not running under a task manager, the current operator window is always the candidate operator window; which is the operator window with which the next virtual device to be opened will be associated.

When you have several applications running concurrently under a task manager, only one of those applications is actually executing, while the others are waiting because they have unsatisfied reads outstanding. Each of the applications can have a current operator window. But no matter how many devices or applications there may be, only that operator window made current by the most recently executed WSCRT, WSSEL, or WSIO call is the candidate operator window; which is the operator window with which the next virtual device to be opened will be associated.

The way that GDDM makes it possible for several application programs to share the screen is by allowing the task manager to intervene in the execution of the program's input/output calls. When each operator window is created, the task manager specifies (in the first array element of the last parameter of the WSCRT call) the address of a coordination exit routine. This runs in the application program subtask, and is invoked by GDDM whenever the application calls a function that requires input/output for the terminal – an ASREAD call, typically. The numbers in the figure represent the following events:

1. An input/output call is issued by the application, causing GDDM to invoke the coordination exit routine.
2. The exit routine, when invoked, must post the task-manager task and wait. The task manager must then call WSIO, the coordinated output/input call. The WSIO call updates all the windows on the screen. WSIO also returns the identifier of the topmost window on the screen. The task manager uses this to find out which subtask to post. It then posts that subtask and waits.
3. When the task manager posts the subtask, control passes back to the coordination exit routine, which in turn returns control to GDDM.
4. Control then returns to the ASREAD (or other application input/output call). GDDM completes the processing of this call, and passes control back to the application program. Any input data entered by the terminal user is then available to the application.

The purpose of the coordination exit routine is to switch control from the subtask to the main task, or the other way round. There is a direction parameter to tell it which way to switch.

How to specify a coordination exit: A coordination exit is specified as part of the WSCRT call. For more information, see the description of the WSCRT call in the Chapter 3, "The GDDM calls" on page 21.

Parameters: The parameters for coordination exits are as follows:

```
R13 -> A 72-byte save area
R14 -> The return address
R15 -> The entry point of the exit
R1  -> The parameter address list, in standard
       variable parm-list format:
             ADDR1 -> AAB     (Char(8))
             ADDR2 -> UXBLOCK ((3) Fixed(31))
             ADDR3 -> DIRECTN (Fixed(31))
```

Parameters `AAB` and `UXBLOCK` are described under "GDDM user-exit conventions" on page 433. Additional parameters are as follows:

DIRECTN The direction in which the exit is to pass control. Possible values are:

0 Pass control from the sub-task to the main task
1 Pass control from the main task to the sub-task.

The exit may not change this parameter.

Feedback values: On return, the exit must set R15 as follows:

0 Successful completion
8 Sub-task terminated abnormally.

Storage exit routines – interface specifications

Storage exit routines can be defined using explicit fields in the System Programmer Interface Block (SPIB) passed as a parameter to GDDM in the SPINIT call.

The following section references fields defined in the Version 1 Release 4 format of the SPIB, but equivalent fields exist in the pre-Version 1 Release 4 format. For more information, see "Initialization" on page 431.

Under VM/CMS and TSO, GDDM calls application exit routines, identified by fields SPIBGSXP and SPIBFSXP (if defined and nonzero), to GET and FREE storage. The interface to these storage exits is as follows:

Register 0 contains the number of bytes of storage requested (GET) or to be released (FREE). The high-order bit of this register is set to indicate a conditional request. This value is passed to the storage exits for both GET and FREE.

Register 1 contains the address of the block of storage. This address is returned by the application exit on GET and passed to the application exit on FREE.

Register 14 contains the GDDM return address.

Register 15 contains the user-defined parameter specified in either field SPIBGSXK (GET) or field SPIBFSXK (FREE). This is passed by GDDM to the appropriate application exit on each call. Before returning to GDDM, the application exit should set a return code in register 15: 0 indicating that the request was successful, and, for conditional requests only, 4 indicating that the request was unsuccessful.

All other registers must be preserved across the call.

Application storage exits must operate without corrupting any of the registers on entry other than as described above. On entry to the exit routines, register 13 does not locate a register save area. If necessary, the exits should provide for their own save area, possibly by "anchoring" a user area by means of the SPIBGSXK or SPIBFSXK, or both, fields passed in register 15.

Application storage exits must not assume that their entry point is located by register 15 on entry. Register 15 is set as described above.

The application GET storage exit must return storage that is double-word aligned.

GDDM abnormally ends on receiving a return code other than as described above.

GDDM requests for blocks of local, last-in-first-out, or instance storage are restricted to a maximum length of 32K bytes. When storage and exit routines are defined (that is, "active"), this restriction also applies to extended storage requests. GDDM never releases "merged" or "split" blocks; storage is always released in blocks as acquired from the application GET exit routine.

Under MVS/XA, MVS/ESA, or VM/XA, the top bit of the specified storage exit address is taken to identify the AMODE of the exit and causes the exit to be called accordingly (that is, a top bit of `'1'B` causes the corresponding exit to be called in 31-bit addressing mode).

Call format descriptor module

A GDDM call format descriptor module, which is independent of the subsystem under which GDDM is running, is provided with GDDM. The module contains information for each GDDM Call statement, describing the number of parameters required on the call, and the type of each parameter.

The address of the call format descriptor module can be acquired by an application program by using the CALLINF external defaults option in a SPINIT call.

The call format descriptor module is in three sections. The first section provides an address table locating the descriptors for call statements with a given first two characters. The second section (the call descriptor table) provides descriptors for all GDDM calls that have the same first two letters in their name. The third section (the parameter descriptor table) provides descriptors for the parameters for a specific call statement.

The address table

The address table is located at offset 0 from the entry point of the module. The format of the address table is:

Table 58. Call format descriptor module – address table

Field name	Field offset	Field length
RCPPIDEN	0	8
RCPPVERS	8	4
RCPPTABP(1)	C	8
RCPPFTWO(1)	C	2
(reserved)	E	2
RCPPSPTR(1)	10	4
RCPPTABP(2)	14	8
RCPPFTWO(2)	14	2
(reserved)	16	2
RCPPSPTR(2)	18	4
⋮	⋮	⋮
RCPPTABP(n)	4+(8×n)	8
RCPPFTWO(n)	4+(8×n)	2
(reserved)	6+(8×n)	2
RCPPSPTR(n)	8+(8×n)	4

RCPPIDEN
An eight-byte table identifier containing the character string "ADMADCP ". Note the mandatory terminating blank.

RCPPVERS
A four-byte (fullword) integer identifying the version number of the call format descriptor module. If this field is set to '1', the extended call descriptor table is present. Applications that use the calls in the extended call descriptor table should test the version code, and if this is set to '1', they should scan the call descriptor table until they reach `X'FFFE'`. Applications that do not use the extension scan the call descriptor table only until they reach `X'FFFF'`.

RCPPTABP(h)
This consists of:

RCPPFTWO(h)
A two-byte character string containing the first two characters of the GDDM call statements described by the call descriptor table addressed by RCPPSPTR(h).

A value of `X'FFFF'` in this field indicates the end of the address table.

(Reserved field)
Two bytes.

RCPPSPTR(h)
The address of the call descriptor table, contained in four bytes, which defines all GDDM call statements that start with the two characters identified by field RCPPFTWO(h).

The call descriptor table

The call descriptor tables are addressed from the address table, There is a call descriptor table for each group of GDDM call statements that have the first two letters in common. The structure of a call descriptor table is illustrated in Table 59 on page 439.

Table 59. Call format descriptor module – call descriptor table

Field name	Field offset	Bit pattern	Field length
RCPPSTAB(1)	0		RCPPLENG(1)
RCPPLENG(1)	0		2
RCPPFLAG(1)	2		2
RCPPIO (1)		x...	bit within RCPPFLAG
RCPPOGP(1)		.x..	bit within RCPPFLAG
RCPPGIO(1)		..x.	bit within RCPPFLAG
RCPPDIO(1)		...x	bit within RCPPFLAG
RCPPIIO(1)		.... x...	bit within RCPPFLAG
RCPPCPAG(1)		.... .x..	bit within RCPPFLAG
RCPPHCNG(1)		.... ..x.	bit within RCPPFLAG
RCPPAPLS(1)		x	bit within RCPPFLAG
RCPPNAME(1)	4		4
RCPPRCP(1)	8		4
RCPPDESC(1)	C		RCPPNARG(1)–12
RCPPSTAB(2)	RCPPLENG(1)		RCPPLENG(2)
RCPPLENG(2)	RCPPLENG(1)		4
)	.		.
) as above,	.		.
) with ...(2)	.		.
)	.		.
.	.		.
.	.		.
.	.		.
RCPPSTAB(n)	&Sigma(RCPPLENG(1) ... (n-1))		RCPPLENG(n)

RCPPSTAB(j)

There is one entry in the call descriptor table for each GDDM call statement.

RCPPLENG(j)

A two-byte field containing the length of this entry in the call descriptor table. The next entry in the table is at offset RCPPLENG from this field.

A value of X'FFFF' indicates the end of the version 0 call descriptor table. If RCPPVERS is set to 1, the call descriptor table extension is present; a value of X'FFFE' indicates the end of the call descriptor table extension.

RCPPFLAG(j)

A set of two-byte flags to indicate features of the CALL statement.

RCPPIO(j) (bit 0)

0 The call cannot cause a terminal I/O.

1 The call may cause I/O to the terminal. This flag is set if any of the flags RCPPGIO, RCPPDIO, or RCPPIIO are set to 1.

RCPPOGP(j) (bit 1)

0 The call is available in the base function of GDDM.

1 The call is available only through another licensed program in the GDDM family of licensed programs.

RCPPGIO(j) (bit 2)

0 No I/O is performed to the device (unless either flag RCPPDIO or flag RCPPIIO is set to 1).

1 The call causes I/O to the terminal. For example, FSFRCE outputs data to the device, ASREAD outputs data and awaits terminal operator input.

RCPPDIO(j) (bit 3)

0 No data-set I/O that causes terminal activity can result from this call.

1 The call may cause I/O activity to a data set. It may result in a terminal I/O operation on some subsystems; for example, a password prompt in opening a data set.

RCPPIIO(j) (bit 4)

0 Data is never sent to the terminal for the call (unless either flag RCPPGIO or flag RCPPDIO is set to one).

1 The call may cause data to be output to the terminal by GDDM if specific conditions are met. Currently, this can only occur if the device is a 3270-PC/G, /GX, or /AT workstation, and the application is drawing graphics primitives outside segments. Implicit I/O occurs whenever too much data stream is accumulated, or a change is made to primitives within segments when primitives outside segments have been drawn.

RCPPCPAG(j) (bit 5)

0 The call applies to GDDM pages other than the current one.

1 The call applies to the current GDDM page only.

RCPPHCNG(j) (bit 6)

0 The call does not cause any change to the hierarchical structure.

1 The call may cause a change to the hierarchical structure of GDDM. One or more of a page, a partition, a partition set, or a device are affected. The flag is set if any of the current elements in the hierarchy may be changed, or an entry may be added to or deleted from the set of hierarchical entities.

RCPPAPLS(j) (bit 7)

0 The call does not require any special processing by APL.

1 The call may require special processing by APL.

RCPPNAME(j)
The last four characters (four bytes) of the GDDM call statement name.

RCPPRCP(j)
A four-byte (fullword) integer specifying the GDDM request control parameter (RCP) code associated with the call statement.

RCPPDESC(j)
This variable-length array contains the descriptors for the arguments that may be passed on to GDDM. See “The parameter descriptor table” for information about this array.

The parameter descriptor table

The parameter descriptor tables are imbedded within the call descriptor tables as described above. The structure of a parameter descriptor table is shown in Table 60 on page 441.

RCPPPDES(n)

For each GDDM call statement, there are one or more sets of parameter descriptors, one for each parameter. Multiple descriptors are also provided when the contents of a parameter list may vary depending upon the contents of the first argument in the parameter list.

RCPPNARG(n)
A one-byte field containing the number of elements in the array RCPPDARG described below. This field contains a value of zero if no parameters are passed to, or received from, GDDM.

The value of this field may be greater than the number of parameters passed to or received from GDDM. In this case, the argument descriptors contain dummy entries used to copy length information between the accumulators used to determine the length of passed or returned data.

With the exception of the dummy entries, each successive element in the array RCPPDARG(n) describes successive arguments passed to or received from GDDM on the call statement.

RCPPDFLG(n)
A one-byte set of flags, only one of which is currently used, to indicate features of the parameters passed to or received from GDDM.

RCPPMATC(n) (bit 0)

0 The parameter descriptors provided in the array RCPPDARG(n) are valid regardless of the contents of the first argument passed to GDDM.

1 The parameter descriptors provided in the array RCPPDARG described below are only valid if the contents of the first parameter, which are always a fixed-point number, are the same as the value specified in the field RCPPMVAL(n). If the contents of the passed parameter do not match those in RCPPMVAL, the next set of parameter descriptors, RCPPPDES(n+1), must be used to test for a matching argument value, or to describe the argument list, depending upon the value of flag RCPPMATC(n+1).

Other flags are reserved for future use.

RCPPMVAL(n)
If flag RCPPMATC(n) = 1, this two-byte field contains the value that the first parameter passed to GDDM must

Table 60. Call format descriptor module – parameter descriptor table

Field name	Field offset	Bit pattern	Field length
RCPPPDES(1)	0		4×(1+RCPPNARG(1))
RCPPNARG(1)	0		1
RCPPDFLG(1)	1		1
RCPPMATC		x...	bit within RCPPDFLG
(reserved)		.xxx xxxx	7 bits within RCPPDFLG
RCPPMVAL(1)	2		2
RCPPDARG(1,1)	4		4
RCPPAFLG	4		1
(parameter data-type flags)		xxxx xx..	6 bits within RCPPAFLG
RCPPINP		.... ..x.	bit within RCPPAFLG
RCPPOUT		x	bit within RCPPAFLG
RCPPLACC(1,1)	5		1
RCPPLVAL(1,1)	6		2
RCPPDARG(1,2)	8		4
)	.		.
) as above,	.		.
) with ...(1,2)	.		.
)	.		.
RCPPDARG(1,m)	4×m		4
)	.		.
) as above,	.		.
) with ...(1,m)	.		.
)	.		.
RCPPPDES(2)	4×(1+RCPPNARG(1))		4×(1+RCPPNARG(2))
)	.		.
) as above,	.		.
) with ...(2)	.		.
)	.		.
.	.		.
.	.		.
.	.		.
RCPPPDES(n)	4×(n+Σ(RCPPNARG(1)...(n)))		4×(1+RCPPNARG(n))

match if the parameter descriptors in array RCPPDARG(n) are the correct descriptors for the instance of the call statement.

RCPPDARG(n,m)

This is an array of dimension RCPPNARG(n). Each element of the array is four bytes long, and is either a descriptor for an argument passed to or received from GDDM, or is a dummy entry used to prime the length accumulators.

RCPPAFLG(n,m)

A one-byte set of flags to indicate the type of the data passed to or received from GDDM.

The parameter data-type flags (in RCPPAFLG, bits 0..5) are set as combinations of these bits, with the meaning shown below:

`100000..` The parameter contains character data.

`010000..` The parameter contains fullword fixed point data.

`010100..` The parameter contains halfword fixed point data.

`001000..` The parameter contains floating point data.

`000100..` The parameter contains undefined format data. The structure of the argument is too complex to describe with a control block structure, probably because the length of the data item cannot be determined without knowledge of the values of the contents of one or more fields imbedded within a prior argument passed to GDDM.

`000010..` The data passed in this parameter is a fullword fixed-point number that is used as either a length or an array dimension. Field RCPPLACC(n,m) contains the number of an accumulator into which the length should be multiplied.

`000001..` The parameter being described contains a fullword length or dimension. Field RCPPLACC(n,m) contains an accumulator number into which the length should be multiplied. Parameter descriptor RCPPDARG(n,m+1) also describes the same passed parameter. This parameter descriptor is therefore used to prime two or more length accumulators from the same argument passed to GDDM.

All unlisted combinations are reserved for future use.

RCPPINP(n,m) (bit 6)

0 The data passed in the parameter is not input to GDDM.

1 The data passed in this parameter is input to GDDM.

RCPPOUT(n,m) (bit 7)

0 The data passed in the parameter is not output from GDDM.

1 The data passed in this parameter is output from GDDM.

RCPPLACC(n,m)
This one-byte field contains an accumulator number. Accumulators are used to define the length of character strings or the number of elements in an array of numbers. Nine accumulators are provided, and all accumulators are assumed to start with an initial value of one.

If either bit 2 or bit 3 in RCPPAFLG(n,m) is set to 1, this field contains the accumulator number that the argument passed to GDDM should be multiplied into.

If both bit 2 and bit 3 in RCPPAFLG(n,m) are set to 0, the accumulator contains the number of characters in a character argument, or the number of fullwords in a numeric array. If an accumulator number of zero is specified, the length or dimension is assumed to be 1. This length or dimension is subject to modification by the contents of field RCPPLVAL(n,m).

RCPPLVAL(n,m)
This two-byte field contains a modifier to be applied to the length of character strings or the dimension of numeric arrays. The total length of the character string, or dimension of a numeric array is obtained by multiplying the contents of the accumulator specified in field RCPPLACC(n,m) with the value of the field RCPPLVAL(n,m).

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

3112 and 3116 IPDS printers 424
3270-PC/G and /GX work stations
  character set usage 263
  GDF files 315
  PIF files 315
  symbol set usage 263
3270-PC/G and /GX workstations
  LCLMODE procopt 403
  load device or GDDM default symbol sets 402
  LOADDSYM procopt 402
  local interactive graphics mode 403
  retained/nonretained mode 402
  zooming and panning pictures 403
3912 and 3916 IPDS printers 424
4224 printer picture overflow 263
4250 high-resolution printer
  spill file usage 399
5080 graphics system 403
5550 multistation 367
6090 graphics system 403

A

ABNDRET, external default 384
address table, APL 251
addresses for call intercept exit 436
adjunct fields
  base attribute 362
  color 362
  cursor 361
  example of specification of 358
  extended highlighting 362
  introduction 357
  length 363
  names 358
  programmed symbols 362
  selector 361
  summary 357
  validation 362
  values 358—363
ADMDVECP default vector symbol set, illustration of 267
ADMOPUV, the GDDM/VM Print Utility
  automatic invocation of 412
  INVKOPUV processing option 412
ADMUCG 324
  return codes 325
ADMUGC 325
  return codes 327
ADMUGIF 343
  return codes 345
ADMUPC
  GDF-ADMGDF conversion utility 281
  GDF-PIF conversion utility 315
ADMUPCT
  GDF-ADMGDF conversion utility under TSO 281
  GDF-PIF conversion utility for TSO 315
ADMUPCV
  GDF-ADMGDF conversion utility under VM/CMS 281
  GDF-PIF conversion utility for VM/CMS 317
ADMUPINx (PL/I declare statements) 6
ADMUUxxx sample CECP vector symbol sets
  illustrations 267
ADMUUxxx sample CECP vector symbol sets, illustrations of 267
Advanced Function Printer Datastream
  See AFPDS
advanced function printers
  device tokens 428
AFPDS
  CDPU support
    structured fields 356
  structured fields
    CDPU support 356
alphanumeric field attributes 245
alternate device
  close (FSCLS) 72
  copy picture to (DSCOPY) 57
  open (FSOPEN) 78
  send character string to (FSLOG) 77
  send character string with carriage-control character to (FSLOGC) 77
  send graphics to (GSCOPY) 109
  send page to (FSCOPY) 72
always-unlock-keyboard mode 397
AM3270, external default 384
APL
  address table 251
  CMSAPLF external default, VM APL feature 385
  codes 252—254
  request code table 251
  request codes module 251
  using with GDDM 2
APPCPG, external default 384
application anchor block 1, 433
application anchor block anchor pointer 1
application data structure (ADS)
  adjunct field names 358
  adjunct fields 357
  adjunct values 358
  assembler-language example 358
  attribute adjunct fields 361
  base attribute adjuncts 362
  character attributes 363

application data structure (ADS) *(continued)*
COBOL example 358
copying into program 366
cursor adjunct 361
description 357
designator characters 364
editing, mapped data 365
generating large structure 366
PL/I example 358
query definition (MSQADS) 213
selector adjunct 361
application group
create (ESACRT) 64
delete (ESADEL) 65
query current (ESAQRY) 65
select (ESASEL) 65
arc
draw circular (GSARC) 96
draw elliptical (GSELPS) 117
draw, GDF order 286
parameters, GDF orders 286
set default parameters, PSC GDF order 302
area
GDF orders 286
shaded, end (GSENDA) 119
ASCII
GINKEY processing option 408
graphics-input key 408
aspect-ratio control (for copy), specify (GSARCC) 97
assembler language
example application data structure 358
using with GDDM 3
assign translation table set to field (ASFTRN) 37
attention-handling for VM 411
attribute adjunct fields
introduction 357
usage 361
AUNLOCK, external default 384
AUNLOCK, processing option 397

B

background color-mix mode
GDF orders 286
query (GSQBMX) 145
set (GSBMIX) 98
set default, PSC GDF order 302
background color, GDF and CGM conversion 330
badge reader 45
bar codes 349
BASIC, using with GDDM 3
bilevel images, initial value 408
blank-to-null conversion on output, define (ASFOUT) 35
BMSCOORD, processing option 397

C

C programming language
pragma linkage 3—5
using with GDDM 3—5
call intercept user exit option 433
call statement syntax
assembler 3
C programming language 3
COBOL programming language 5
FORTRAN programming language 5
GDDM_REXX 6
PL/I programming language 6
call statements
detailed descriptions 21—239
syntax conventions 21
CALLINF, external default 384
CDPDS
document structure 347—348
structured field formats 347—356
CDPFTYPE, processing option 398
CDPU (Composite Document Print Utility)
AFPDS support 356
API call 50
file format 347—356
CECPINP, external default 385
cell-size dimensions for devices 243
CGM
applications 323
content 323
conversion between CGM and ADMGDF formats 323
file format 327
load a picture from (CGLOAD) 323
output 323
picture, load from 51
save segments in (CGSAVE) 324
segments, save in (CGSAVE) 54
support 323
CGM profile
customizing
character-height factor 337
character-width factor 337
color mapping 331
chained attributes, set initial (GSSATI) 166
change field status (ASFMOD) 34
change resolution flag of an image (IMARF) 186
changing GDDM supplied values
defaults 379
character angle
GDF order 287
query (GSQCA) 146
set (GSCA) 99
set default, PSC GDF order 302
character attributes
introduction 357
setting from the terminal 364

character attributes *(continued)*
setting within a program 363
character box
GDF order 287
query size (GSQCB) 146
query spacing (GSQCBS) 147
set default, PSC GDF order 303
set size (GSCB) 102
set spacing (GSCBS) 103
character colors for field, query (ASQCOL) 39
character direction
Arabic text (GSCD) 104
Chinese text (GSCD) 104
GDF order 288
query (GSQCD) 147
Roman text (GSCD) 104
set (GSCD) 103
set default, PSC GDF order 303
character mode
query (GSQCM) 148
set (GSCM) 108
character pitch, processing option 408
character precision
GDF order 288
set default, PSC GDF order 304
character shear
GDF order 288
query (GSQCH) 147
set (GSCH) 105
set default, PSC GDF order 304
character string
draw at current position (GSCHAP) 105
draw at specified point (GSCHAR) 106
GDF order 289
send to alternate device (FSLOG) 77
send to alternate device with carriage-control character (FSLOGC) 77
character-code assignments, override (ASTYPE) 48
CICAUD, external default 385
CICDECK, external default 385
CICDFPX, external default 385
CICGIMP, external default 385
CICIADS, external default 385
CICIFMT, external default 385
CICPRNT, external default 385
CICS
ADMGIMP name, CICGIMP external default 385
ADS name, CICIADS external default 385
audit trail anchor block 385
coordination mode 397
defaults file temporary storage 385
device query temporary storage prefix, CICTQRY external default 385
GDDM-IMD staging data file name, CICSTGF external default 385
print utility, CICPRNT external default 385
CICS *(continued)*
pseudoconversational mode control 404
staging data file-type, CICIFMT external default 385
system printer name, CICSYSP external default 385
temporary storage prefix, CICTSPX external default 385
trace transient data name, CICTRCE external default 385
CICSTGF, external default 385
CICSYSP, external default 385
CICTIF, external default 385
CICTQRY, external default 385
CICTRCE, external default 385
CICTSPX, external default 385
clear
current page (FSPCLR) 79
field (ASFCLR) 31
graphics field (GSCLR) 108
graphics input queue (GSFLSH) 121
rectangle in image (IMACLR) 179
CLEAR/PA1 protocol in TSO 412
clipping
enable and disable (GSCLP) 107
query state (GSQCLP) 148
close
alternate device (FSCLS) 72
current segment (GSSCLS) 167
device (DSCLS) 56
CMSAPLF, external default 385
CMSATTN, processing option 411
CMSCOLM, external default 386
CMSDECK, external default 386
CMSDFTS, external default 386
CMSIADS, external default 386
CMSIFMT, external default 386
CMSINTRP, processing option 410
CMSMONO, external default 386
CMSMSLT, external default 386
CMSPRNT, external default 386
CMSSYSP, external default 386
CMSTEMP, external default 386
CMSTRCE, external default 386
COBOL programming language
example application data structure 358
using with GDDM 5
code conversion on character string (FSTRAN) 94
code page
for CGM applications 328
GDDM object, set (ESSCPG) 70
query (GSQCPG) 149
set (GSCPG) 113
color
adjunct and attribute 362
characters within field, query (ASQCOL) 39
characters within field, specify (ASCCOL) 25
field, define (ASFCOL) 32
graphics, query (GSQCOL) 148

color *(continued)*
graphics, set (GSCOL) 109
color mapping
customizing CGM profile 331
keyword, COLOR_MAPPING 331
color master table identifier 413
COLOR_MAPPING, keyword 331
color, graphics
extended, set default, PSC GDF order 305
GDF order 289
COLORMAS, processing option 413
COMMENT, external default 386
complex pictures 263
Composite Document Presentation Data Stream (CDPDS)
see CDPDS
composite documents
file format 347—356
print (CDPU) 50
Computer Graphics Metafiles
see CGM
Computer Graphics Metafiles (CGM) 323
conditional load (PSLSSC) 221
contents of data buffer 372
control echoing of scanner image (ISESCA) 201
control internal trace (FSTRCE) 95
control use of mixed fields by mapping (SPMXMP) 231
conversion profile, CGM
example format 327
picture adjustment factors 335
picture mapping information 329
convert
ADMGDF to CGM file (ADMUGC utility) 325
ADMGDF to GIF file (ADMUGIF utility) 343
CGM to ADMGDF file (ADMUCG utility) 324
resolution attributes of an image (IMARES) 186
coordination exit control direction parameter 437
coordination exit routine 436, 437
coordination mode for CICS BMS 397
copy a segment (GSSCPY) 168
correlate segments and tags (GSCORS) 111
correlate tag to primitive (GSCORR) 110
CPN4250, external default 386
CPSPOOL, processing option 411
CPTAG, processing option 411
create
application group (ESACRT) 64
empty projection (IMPCRT) 189
image (IMACRT) 179
mapped field (MSDFLD) 210
operator window (WSCRT) 233
page (FSPCRT) 79
page for mapping (MSPCRT) 211
partition (PTNCRT) 223
partition set (PTSCRT) 227
segment (GSSEG) 170
CTLFAST, processing option 404
CTLKEY, processing option 405
CTLMODE, processing option 405
CTLPRINT, processing option 405
CTLSAVE, external default 386
CTLSAVE, processing option 405
current position
GDF order 290
move without drawing (GSMOVE) 137
query (GSQCP) 149
set (GSCP) 112
cursor
alpanumeric, query position (ASQCUR) 40
alphanumeric, query position (GSQCUR) 150
alphanumerics, position within field (ASFCUR) 32
control with mapping requests 361
image box, initialize (ISIBOX) 202
image locator, initialize (ISILOC) 203
image locator, query position (ISQLOC) 206
image, enable or disable (ISENAB) 200
query position in map (MSQPOS) 218
selection 364
set position in mapped field (MSCPOS) 209
cursor adjunct fields
introduction 357
usage 361
customization
color mapping 331

D

data boundary
define (GSBND) 99
query (GSQBND) 146
data streams
truncation processing option, IPDS printers 407
DATEFRM, external default 386
DATRN, external default 387
DBCS fields
control use of mixed fields by mapping (SPMXMP) 231
get contents (ASGGET) 38
in GDDM-IMD 366
in mapped data 367
in mapping 366
MIXSOSI external default 390
selection 387
SOSI emulation character, SOSIEMC external default 391
specify contents (ASGPUT) 39
symbol set component threshold 387
symbol set language option 387
symbol sets 264
DBCSDFT, external default 387
DBCSDNM, external default 387
DBCSLIM, external default 387

DBCSLNG, external default 387
default user exit option 433
defaults
  changing GDDM-supplied values 379
  drawing, end definition (GSDEFE) 114
  drawing, start definition (GSDEFS) 114
  encoded UDS, query (ESQEUD) 68
  GDDM-supplied values 379
  source-format UDS, specify (ESSUDS) 71
  UDS, specify encoded (ESEUDS) 66
  user exits 432
deferred device name-list for print utility 402
define
  bi-level conversion algorithm (IMRCVB) 192
  brightness conversion algorithm (IMRBRI) 191
  contrast conversion algorithm (IMRCON) 191
  field attributes (ASRATT) 45
  field color (ASFCOL) 32
  field end action (ASFEND) 33
  field mixed-string attribute (ASFSEN) 36
  field type (ASFTYP) 38
  graphics field (GSFLD) 120
  graphics window (GSWIN) 178
  image field (ISFLD) 201
  picture space (GSPS) 142
  place position in pixel coordinates (IMRPL) 195
  place position in real coordinates (IMRPLR) 195
  primary symbol set for field (ASFPSS) 35
  rectangular sub-image in pixel coordinates (IMREX) 193
  rectangular sub-image in real coordinates (IMREXR) 193
  segment viewing limits (GSSVL) 174
  single field (ASDFLD) 28
  uniform graphics window (GSUWIN) 176
  viewport (GSVIEW) 177
delete
  application group (ESADEL) 65
  image (IMADEL) 180
  mapped field (MSDFLD) 210
  operator window (WSDEL) 234
  page (FSPDEL) 80
  partition (PTNDEL) 224
  partition set (PTSDEL) 228
  projection (IMPDEL) 189
  segment (GSSDEL) 169
descriptor modules for call formats 438
designator characters
  introduction 357
  values 364
DEVCPG, processing option 406
DEVCSET, processing option 410
device
  close (DSCLS) 56
  input, enable or disable (FSENAB) 73
  query unique identifier (DSQUID) 63
  query usage (DSQUSE) 63
  reinitialize (DSRNIT) 63
Device character set
  overriding with procopt 410
device characteristic tokens
  see device tokens
device characteristics
  query (DSQDEV) 62
  query (FSQDEV) 82
  query (FSQURY) 85
device code page
  processing option (DEVCPG) 406
device stores, query status (PSQSS) 222
device tokens
  advanced function printers 428
  ASCII devices (family 1) 426
  defining 421
  displays 421
  GDDM-PCLK displays 427
  GDDM-PCLK plotters 427
  GDDM-PCLK printers 427
  IBM 3290 displays 425
  IBM 8775 displays 425
  Kanji devices 425
  nonqueriable displays 426
  nonqueriable printers 426
  page printers (cell-based family 4) 428
  page printers (family 4) 430
  plotters 421
  PostScript (family 4) 429
  printers 421
  queriable terminals 421
  scanners 421
  system printers (family 3) 428
device usage
  discontinue (DSDROP) 58
  specify (DSUSE) 64
device variations
  alphanumerics 245
  area, shading 247
  background color-mix mode 246
  check picture complexity before output (FSCHEK) 243
  color wrapping 245
  color-mix mode 246
  double-byte character sets (DBCS) 245
  dual screen devices 241
  field color 245
  foreground color-mix mode 246
  fractional line width, set (GSFLW) 247
  GDF, saved as 2-byte integers (GSSAVE) 242
  graphics image 247
  graphics text 242
  image 249
  line type (GSLT) 247
  line width, set (GSLW) 247
  locator device, initialize (GSILOC) 247
  logical input devices 247
  pick device, initialize (GSIPIK) 248

device variations *(continued)*
picture, display saved (FSSHOW or FSSHOR) 242
picture, save 242
programmed symbol sets 242
PS overflow check 243
screen redraw 242
screen size, alphanumeric 241
screen size, alternate 241
screen size, primary 241
segments, primitives outside 242
string device, initialize (GSISTR) 249
stroke device, initialize (GSISTK) 248
DFTXTNA, external default 387
disable 200
clipping (GSCLP) 107
device input (FSENAB) 73
image cursor (ISENAB) 200
logical input device (GSENAB) 118
display
mapped data (MSREAD) 219
saved picture 242
saved picture (FSSHOW) 93
saved picture – extended FSSHOW (FSSHOR) 93
update (DSFRCE) 58
update (FSFRCE) 74
update (FSGETE) 75
update (FSGETS) 76
document name 403
double-byte character set
See DBCS fields
draw
character string at current position (GSCHAP) 105
character string at specified point (GSCHAR) 106
circular arc (GSARC) 96
curved fillet (GSPFLT) 140
elliptical arc (GSELPS) 117
graphics image (GSIMG) 126
scaled graphics image (GSIMGS) 127
series of lines (GSPLNE) 141
series of markers (GSMRKS) 138
single marker (GSMARK) 135
straight line (GSLINE) 130

E

EBCDIC character codes 49
enable 200
clipping (GSCLP) 107
device input (FSENAB) 73
image cursor (ISENAB) 200
logical input device (GSENAB) 118
end
data entry into image (IMAPTE) 184
drawing defaults definition (GSDEFE) 114
retrieval of data from an image (IMAGTE) 182
retrieval of graphics data (GSGETE) 122
end *(continued)*
shaded area (GSENDA) 119
enter data into image (IMAPT) 183
ERRFDBK, external default 387
error exits
error record structure 83
specify (FSEXIT) 74
error last, query (FSQERR) 83
error messages 21
ERRTHRS, external default 388
ESLIB (library management) 208
external default options 384—393
external defaults
abend/return processing 384
ABNDRET 384
alphanumeric defaults module control 387
always-unlock-keyboard 384
AM3270 384
APL specification for TSO 392
APPCPG 384
application code-page 384
AUNLOCK 384
call information block 384
CALLINF 384
CECPINP 385
CGM conversion filetype for TSO 392
CGM conversion filetype for VM 386
CICAUD 385
CICDECK 385
CICDFPX 385
CICGIMP 385
CICIADS 385
CICIFMT 385
CICPRNT 385
CICS ADMGIMP name 385
CICS ADS name 385
CICS audit trail anchor 385
CICS deck name 385
CICS defaults file temporary storage 385
CICS device query temporary storage prefix 385
CICS GDDM-IMD staging data file name 385
CICS GDDM-IMD staging data file-type 385
CICS print utility name 385
CICS system printer name 385
CICS temporary storage prefix 385
CICS transaction independence 385
CICSTGF 385
CICSYSP 385
CICTIF 385
CICTQRY 385
CICTRCE 385
CICTSPX 385
CMSAPLF 385
CMSCOLM 386
CMSCPT 386
CMSDECK 386

external defaults *(continued)*
CMSDFTS 386
CMSIADS 386
CMSIFMT 386
CMSMONO 386
CMSMSLT 386
CMSPRNT 386
CMSSYSP 386
CMSTEMP 386
CMSTRCE 386
color master ddname/high-level qualifier for TSO 392
color master file name 393
color master filetype for VM 386
COMMENT 386
compressed PS loads 390
conversion between CGM and ADMGDF 327
CPN4250 386
CTLSAVE 386
DATEFRM 386
dates punctuation convention 386
DATRN 387
DBCS selection 387
DBCSDFT 387
DBCSDNM 387
DBCSLIM 387
DBCSLNG 387
ddname for TSO 392
deck output LTERM for IMS 388
default symbol-set names 387
device attachment 384
DFTXTNA 387
dynamic allocation size 393
ERRFDBK 387
error exit 387
error thresholds 388
ERRTHRS 388
exit character string for IMS 389
FF3270P 388
File Control dataset names 391
file name for VSE 393
force evaluation of HPA 388
form feed 388
format 379
FRCEVAL 388
FSSAVE buffer size 391
IBM 4250 code page name 386
ICU format 388
ICU transaction name for IMS 389
ICUFMDF 388
ICUFMSS 388
ICUISOL 388
ICUISOL, isolate value for ICU 388
ICUPANC 388
ICUPANC, use of panel color value for ICU 388
IMS shutdown LTERM name 389
IMSDECK 388

external defaults *(continued)*
IMSEXIT 389
IMSICU 389
IMSISE 389
IMSMAST 389
IMSMODN 389
IMSPRNT 389
IMSSDBD 389
IMSSEGS 389
IMSSHUT 389
IMSSYSP 389
IMSTRCE 389
IMSUISZ 389
IMSUMAX 389
IMSVSE 389
IMSWTOD 389
IMSWTOR 389
in-storage trace table size 392
input area size for IMS 389
INSCPG 389
installation code-page 389
IOBFSZ 390
IOCOMPR 390
IOSYNCH 390
ISE transaction name for IMS 389
keyboard input of CECP characters 385
mapgroup storage threshold 390
MAPGSTG 390
maximum number of users for IMS 389
message output descriptor name 389
mixed fields 390
MIXSOSI 390
monochrome file name for VSE 393
national language support 391
NATLANG 391
numbering convention 391
NUMBFRM 391
OBJFILE 391
parameter verification 391
PARMVER 391
print utility name 389
SAVBFSZ 391
segment names for IMS 389
short-on-storage processing 392
shutdown string for IMS 389
SOSI emulation character 391
SOSIEMC 391
STGRET 392
symbol set component threshold 387
symbol set language 387
synchronized I/O 390
system printer name for IMS 389
system-definition database definition (DBD) name 389
time punctuation convention 392
TIMEFRM 392
TRACE 392

external defaults *(continued)*
trace control 392
trace ddname for IMS 389
trace ddname for TSO 393
trace file name for VSE 393
trace output width control 392
trace share 392
trace word value 392
transmission buffer size 390
TRCESTR 392
TRCEWID 392
TRTABLE 392
TSO ADMGIMP ddname 393
TSO ADS ddname 393
TSO deck ddname 392
TSO export utility ddname 393
TSO I/O Emulation 392
TSO monochrome ddname or low-level qualifier 393
TSO print data-set qualifier 393
TSO reserve master print queue DASD 393
TSO system printer ddname 393
TSOAPLF 392
TSOCOLM 392
TSOCPT 392
TSODECK 392
TSODFTS 392
TSOEMUL 392
TSOGIMP 393
TSOIADS 393
TSOIFMT 393
TSOMONO 393
TSOPRNT 393
TSORESV 393
TSOS99S 393
TSOS99U 393
TSOSYSP 393
TSOTRCE 393
unit specification for TSO 393
use of symbol sets in formats value for ICU 388
User Control SAVE function control 386
Vector Symbol Editor transaction name for IMS 389
VM ADS filetype 386
VM APL feature 385
VM deck filetype 386
VM export utility filetype 386
VM filename and filetype 386
VM monochrome filetype 386
VM MSL filetype 386
VM print filetype 386
VM system printer filetype 386
VM trace filename/filetype 386
VM work-file filetype 386
VSE batch printing 387
VSECOLM 393
VSEDFTS 393
VSEMONO 393
external defaults *(continued)*
VSETRCE 393
write-to-operator descriptor codes for IMS 389
write-to-operator routing codes for IMS 389
external interfaces
nonreentrant 1
reentrant 1
system programmer 2
system programmer interface 431

F

family-2 and -4 print-file destination in TSO 412
fast update mode 404
FASTUPD, processing option 404
feedback values
call intercept exit 436
task switch exit 435
FF3270P, external default 388
field
assign translation table set to (ASFTRN) 37
change status (ASFMOD) 34
character colors, specify within (ASCCOL) 25
character highlights within, specify (ASCHLT) 26
character symbol sets, specify within (ASCSS) 27
clear (ASFCLR) 31
contents, specify (ASCPUT) 27
define attributes (ASRATT) 45
define color (ASFCOL) 32
define input null-to-blank conversion (ASFIN) 34
define mixed-string attribute (ASFSEN) 36
define or delete single (ASDFLD) 28
define output blank-to-null conversion (ASFOUT) 35
define primary symbol set for (ASFPSS) 35
define type (ASFTYP) 38
end action, define (ASFEND) 33
get contents (ASCGET) 26
get double-character contents (ASGGET) 38
graphics, clear (GSCLR) 108
highlight, define (ASFHLT) 33
image, query (ISQFLD) 205
intensity, define (ASFINT) 34
modified, query number (ASQNMF) 44
multiple, define (ASDFMT) 29
outline, define (ASFBDY) 31
outlining in maps 363
position cursor within (ASFCUR) 32
query attributes (ASQFLD) 41
query character colors (ASQCOL) 39
query character highlights for (ASQHLT) 42
query character symbol sets (ASQSS) 44
query length of contents (ASQLEN) 42
query maximum number (ASQMAX) 43
query modified (ASQMOD) 43
redefine (ASRFMT) 47
set default attributes (ASDFLT) 29

field *(continued)*
specify double-character contents (ASGPUT) 39
transparency attribute, define (ASFTRA) 37
field attributes for mapping
alphanumeric 362
autoskip 362
base 362
blinking 362
color 362
detectable 362
extended highlighting 362
intensified-display 362
introduction 357
mandatory enter 362
mandatory fill 363
MDT 362
nondetectable 362
nondisplay 362
normal-display 362
numeric 362
programmed symbols 362
protected 362
reverse video 362
trigger 363
underscore 362
unprotected 362
validation 362
field list
define (APDEF) 21
delete (APDEL) 22
modify (APMOD) 23
query (APQRY) 24
query identifiers (APQIDS) 23
query numbers (APQNUM) 24
query size (APQSIZ) 25
query unique identifier (APQUID) 25
field, mapped
create or delete (MSDFLD) 210
mixed, control use of (SPMXMP) 231
place data into (MSPUT) 212
query (MSQFLD) 216
query modified (MSQMOD) 218
retrieve data from (MSGET) 211
fillet
curved, draw (GSPFLT) 140
GDF order 291
foreground color-mix mode
GDF order 291
query (GSQMIX) 153
set (GSMIX) 136
set default, PSC GDF order 305
FORTRAN programming language, using with GDDM 5
FRCETYPE, processing option 413
FRCEVAL, external default 388
FSLOG
maximum characters for each line 398
FSLOG *(continued)*
page sizes for 398
FSLOGC
maximum characters for each line 398
page sizes for 398
functions
control 14
copy 15
detailed descriptions 21—239
device 15
graphics 15
graphics segments 16
high-performance alphanumerics 17
image 17
interactive graphics 17
mapped alphanumerics 18
operator window 18
page 19
partition 19
procedural alphanumeric 19
summary 13—20
symbol set 20
utility call 20

G

GDDM
changing default user exits
call intercept exit 435
task switch exit 434
changing supplied default values 379
object file format 279—280
supplied declarations for mapping constants tables 367
supplied device tokens 421, 429
user-exit conventions 433
GDDM Internet home page xix
GDDM-REXX
command and subcommands summary 255
differences from other programming languages
FSINIT 10
FSTERM 10
ERXMSVAR utility EXEC 258
GDDMREXX command 255
GXGET subcommand 256—257
GXSET subcommand 257—258
programming interface 255—258
restrictions in
CHART 10
SPINIT 10
subcommands 256—258
using with GDDM 6—11
GDF
description of orders 281
end retrieval of (GSGETE) 122
format of objects 282
handling of orders during CGM to GDF conversion 340

GDF *(continued)*
handling of orders during GDF to CGM conversion 339
list of orders 282
loading objects from library (GSLOAD) 130
restore (GSPUT) 143
retrieve fam-4 datastream (FSGET) 75
retrieve graphics data as (GSGET) 121
start retrieval of (GSGETS) 122
GDF orders
arc 286
arc parameters 286
arc parameters, set default 302
area 286
attributes 285
background color mix mode, set default 302
background color-mix mode 286
begin image 292
begin picture prolog 302
call segment 287
character angle 287
character angle, set default 302, 303
character box 287
character box spacing 287
character direction 288
character direction, set default 303
character precision 288
character precision, set default 304
character shear 288
character shear, set default 304
character string 289
character-box spacing, set default 303
color 289
comment 290
coordinate type, set default 304
current position 290
default PSC 302—309
draw fillet 291
draw full arc (circle or ellipse) 292
draw line 293
draw relative line 297
end area 291
end image 293
end picture prolog 302
end symbol-set mapping 302
extended color, set default, PSC GDF order 305
foreground color-mix mode 291
foreground color-mix mode, set default 305
format 283
fractional line width 292
line type 294
line type, set default 306
line width 294
line width, fractional, set default 305
map symbol-set identifiers 301
marker 294
marker box 294
GDF orders *(continued)*
marker box, set default 306
marker scale 295
marker type 295
marker type, set default 306
model transformation 295
padding 283
pattern, set default 306
pick identifier, set default identifier 307
picture boundary, set default 308
picture origin, set default 309
picture scale, set default 307
pop 296
PSC definition 300—309
representation of graphics primitives 283
segment attributes 297
segment attributes, modify 297
segment end 298
segment origin 298
segment position 298
segment prolog, end 298
segment viewing window 300
shading patterns 296
start segment 299
symbol set mapping 301
symbol set, set default, PSC GDF order 304
symbol sets 288
tag identifier 296
text alignment 300
text alignment, set default 307
transform graphics 295
use of current position 285
viewing window, set default 308
write image data 293
GDF-ADMGDF conversion utility 281
get and reserve, image (IMAGID) 180
get and reserve, projection (IMPGID) 189
get field contents (ASCGET) 26
GIF
content 343
output 343
support 343
GIF (graphics interchange format files) 343
GINKEY, processing option 408
GL (graphics language) plot file 409
graphics
loading symbol sets 262
using PS with 262
graphics attributes
query mode (GSQAM) 144
restore (GSPOP) 142
set mode (GSAM) 96
graphics data format
See GDF
graphics field
define (GSFLD) 120

graphics field *(continued)*
query (GSQFLD) 150
graphics image
begin, GDF order 292
draw (GSIMG) 126
draw scaled (GSIMGS) 127
end, GDF order 293
write data, GDF order 293
graphics input key on ASCII graphics devices 408
graphics input queue, clear (GSFLSH) 121
graphics input, wait for (GSREAD)
graphics interchange format files (GIF) 343
graphics primitives
outside segments 242
representation through GDF 283
graphics segments
call or transform (GSCALL) 100
close current (GSSCLS) 167
copy (GSSCPY) 168
correlate to tags (GSCORS) 111
create (GSSEG) 170
define viewing limits (GSSVL) 174
delete (GSSDEL) 169
geometric attributes, set (GSSAGA) 164
geometric attributes, set for current (GSSCT) 168
include (GSSINC) 171
modify attributes (GSSATS) 166
query all geometric attributes (GSQAGA) 143
query attributes (GSQATS) 145
query initial attributes (GSQATI) 144
query number (GSQMAX) 152
query origin (GSQORG) 154
query position (GSQPOS) 155
query priority (GSQPRI) 155
query viewing limits (GSQSVL) 159
save (GSSAVE) 166
set geometric attributes (GSSTFM) 173
set initial attributes (GSSATI) 165
set origin (GSSORG) 171
set position (GSSPOS) 172
set priority (GSSPRI) 172
set transform (GSSTFM) 173
transform current (GSSCT) 168
graphics text
alignment, GDF order 300
alignment, set default, PSC GDF order 307
query alignment (GSQTA) 159
query box (GSQTB) 160
query mixed-string attribute (GSQSEN) 156
set alignment (GSTA) 175
set mixed-string attribute (GSSEN) 170
graphics window
define (GSWIN) 178
uniform, define (GSUWIN) 176
GRAYLINE, processing option 410

H

Hangeul fields (see DBCS fields)
heading page 398
high-performance alphanumerics
bundle definition 373
bundle-list contents 373
bundle-list header 373
contents of field-list header 369
data buffer 372
data structure 369
data-buffer attributes
color 372
highlight 372
symbol set 372
data-buffer contents 372
dynamic fields 376
enlarging structures 377
example of bundle list 375
example of data buffer 373
example of field list 372
field list 369
field-definition contents 369
functions 17
how to use 375
input 376
locate mode 375
move mode 375
output 376
reshow 376
restrictions
FRCEVAL call 377
on running with validation set 377
on use of shared storage 377
on use with interpreted languages 377
on use with read-only storage 377
updating a bundle list 376
updating a data buffer 376
updating a field list 376
highlight field, define (ASFHLT) 33
home page for GDDM xix
HRIDOCNM, processing option 403
HRIFORMT, processing option 399
HRIPSIZE, processing option 399
HRISPILL, processing option 399
HRISWATH, processing option 399

I

I/O translation tables, define (ASDTRN) 30
IBM 4250 printer
code page name 386
IBM-GL output 409
ICU (Interactive Chart Utility)
isolate value, ICUISOL external default 388
use of panel color, ICUPANC external default 388

ICUFMDF, external default 388
ICUFMSS, external default 388
ICUISOL, external default 388
ICUPANC, external default 388
identify program communication block (ESPCB) 68
IEEE customization panel 415
image
  box cursor, query (ISQBOX) 204
  clear rectangle (IMACLR) 179
  control echoing of scanner image (ISESCA) 201
  convert resolution attributes (IMARES) 186
  create (IMACRT) 179
  create an empty projection (IMPCRT) 189
  cursors, query device characteristics (FSQURY) 90
  data compression types 311
  define bi-level conversion algorithm (IMRCVB) 192
  define brightness conversion algorithm (IMRBRI) 191
  define contrast conversion algorithm (IMRCON) 191
  define field (ISFLD) 201
  define place position in pixel coordinates (IMRPL) 195
  define place position in real coordinates (IMRPLR) 195
  define rectangular sub-image in pixel coordinates (IMREX) 193
  define rectangular sub-image in real coordinates (IMREXR) 193
  delete (IMADEL) 180
  delete projection (IMPDEL) 189
  displays, query device characteristics (FSQURY) 89
  enable or disable cursor (ISENAB) 200
  end data entry (IMAPTE) 184
  end retrieval of data (IMAGTE) 182
  enter data (IMAPT) 183
  field, query (ISQFLD) 205
  file formats 311
  format, query (ISQFOR) 206
  initial value of bilevel 408
  initialize box cursor (ISIBOX) 202
  initialize locator cursor (ISILOC) 203
  negate pixels of extracted image (IMRNEG) 194
  orientate (IMRORN) 194
  projection, unique identifier, get and reserve (IMPGID) 189
  query attributes (IMAQRY) 185
  query compressions (ISQCOM) 205
  query locator cursor position (ISQLOC) 206
  query scanner (ISQSCA) 207
  query supported resolutions (ISQRES) 207
  read-only, load external (ISLDE) 204
  reflect (IMRREF) 197
  resolution flag, change (IMARF) 186
  restore from auxiliary storage (IMARST) 187
  restore projection from auxiliary storage (IMPRST) 190
  retrieve data (IMAGT) 181
  save on auxiliary storage (IMASAV) 188
  save projection on auxiliary storage (IMPSAV) 190
  scale (IMRSCL) 197
image *(continued)*
  scanners, query device characteristics (FSQURY) 90
  set current resolution/scaling algorithm (IMRRAL) 196
  set extended quality control parameters (ISXCTL) 208
  set quality-control parameters (ISCTL) 199
  start data entry (IMAPTS) 184
  start retrieval of data (IMAGTS) 182
  swathing 407
  transfer data between, applying a projection (IMXFER) 198
  trim to rectangle (IMATRM) 188
  unique identifier, get and reserve (IMAGID) 180
Image Symbol Editor API call (ISSE) 208
image symbol sets 276
IMGINIT, processing option 408
IMS
  deck output LTERM, IMSDECK external default 388
  exit character string, IMSEXIT external default 389
  ICU transaction name, IMSICU external default 389
  input area size, IMSUISZ external default 389
  ISE transaction name, IMSISE external default 389
  maximum number of users, IMSUMAX external default 389
  message output descriptor name, IMSMODN external default 389
  print utility name, IMSPRNT external default 389
  segment names, IMSSEGS external default 389
  shutdown LTERM name, IMSMAST external default 389
  shutdown string, IMSSHUT external default 389
  system printer name, IMSSYSP external default 389
  system-definition database definition (DBD) name, IMSSDBD external default 389
  trace ddname (IMSTRCE default) 389
  Vector Symbol Editor transaction name, IMSVSE external default 389
  write-to-operator descriptor codes, IMSWTOD external default 389
  write-to-operator, routing codes, IMSWTOR external default 389
IMSDECK, external default 388
IMSEXIT, external default 389
IMSICU, external default 389
IMSISE, external default 389
IMSMAST, external default 389
IMSMODN, external default 389
IMSPRNT, external default 389
IMSSDBD, external default 389
IMSSEGS, external default 389
IMSSHUT, external default 389
IMSSYSP, external default 389
IMSTRCE, external default 389
IMSUISZ, external default 389
IMSUMAX, external default 389
IMSVSE, external default 389
IMSWTOD, external default 389

IMSWTOR, external default 389
include segment (GSSINC) 171
IND$FILE
  CLIST 316
  EXEC 317
initial data value for device
  float (GSIDVF) 123
  integer (GSIDVI) 124
initial value for bilevel images 408
initialize
  GDDM processing (FSINIT) 76
  GDDM with SPIB
    call intercept exit 433
    task switch exit 433
    user exit definition 433
  GDDM with SPIB (SPINIT) 231
  image box cursor (ISIBOX) 202
  image locator cursor (ISILOC) 203
  locator (GSILOC) 125
  pick device (GSIPIK) 128
  string device (GSISTR) 129
  stroke device (GSISTK) 128
input/output, device (ASREAD) 45
INRESRCE, processing option 405
INSCPG, external default 389
intensity, field, define (ASFINT) 34
Interactive Map Definition (GDDM-IMD) 357
interfaces
  external 1
  nonreentrant 1
Internet home page for GDDM xix
INVKOPUV, processing option 412
invoking VM print utility automatically 412
IOBFSZ, external default 390
IOCOMPR, external default 390
IOSYNCH, external default 390
IPDS printers
  characters per inch 408
  data stream truncation processing option 407
  image swathing 407
  paper feed bin selection 407
  quality processing option 406
  rotation processing option
    IPDSROT 406
IPDSBIN, processing option 407
IPDSCPI, processing option 408
IPDSIMSW, processing option 407
IPDSLPI, processing option 407
IPDSQUAL, processing option 406
IPDSROT, processing option 406
IPDSTRUN, processing option 407
ISS
  see image symbol sets

J

Japan (Latin) extended character codes 50

K

Kanji fields (see DBCS fields)
Katakana character codes 49
keyboard, unlocking in DSOPEN 397

L

LCLMODE, processing option 403
length adjunct
  introduction 357
  usage 363
library management (ESLIB) 67
line
  draw relative, GDF order 297
  draw, GDF order 293
line type
  GDF order 294
  query (GSQLT) 151
  set (GSLT) 134
  set default, PSC GDF order 306
line width
  fractional, GDF order 292
  fractional, set default, PSC GDF order 305
  GDF order 294
  query (GSQLW) 152
  query fractional (GSQFLW) 150
  set (GSLW) 134
  set fractional (GSFLW) 121
line-pitch processing option 407
lines-per-inch processing option 407
load
  external read-only image (ISLDE) 204
  graphics segments (GSLOAD) 130
  graphics symbol set from auxiliary storage (GSLSS) 133
  graphics symbol sets from application program (GSDSS) 117
  picture from a CGM (CGLOAD) 323
  PS sets, using mapping 362
  symbol set into PS store from application program (PSDSS) 219
  symbol set into PS store from auxiliary storage (PSLSS) 220
LOADDSYM, processing option 402
loading
  workstation or GDDM default symbol sets 402
local interactive graphics mode 403
locator device
  initialize (GSILOC) 125
  query data (GSQLOC) 151
lock keyboard mode 397

logical input device
enable or disable (GSENAB) 118
initial data value, float (GSIDVF) 123
initial data value, integer (GSIDVI) 124
initialize pick device (GSIPIK) 128
query (GSQLID) 151

M

magnetic stripe (badge) reader 45
MAPGSTG, external default 390
mapped alphanumerics
application data structures 357—367
create or delete mapped field (MSDFLD) 210
create page for mapping (MSPCRT) 211
display data (MSREAD) 219
place data into mapped field (MSPUT) 212
query application data structure definition (MSQADS) 213
query cursor position in map (MSQPOS) 218
query map characteristics (MSQMAP) 217
query map fit (MSQFIT) 216
query mapgroup characteristics (MSQGRP) 217
query mapped field (MSQFLD) 216
query modified fields (MSQMOD) 218
query page (MSPQRY) 212
retrieve data from mapped field (MSGET) 211
set cursor position in mapped field (MSCPOS) 209
mapping
adjunct field names 358
adjunct fields 357
adjunct values 358
attribute adjunct fields 361
base attribute adjunct fields 362
character attributes 363
color adjunct 362
copy application data structure into program 366
create or delete mapped field (MSDFLD) 361
cursor adjunct 361
detectable fields 364
extended highlighting 362
field attributes 362
generating large application data structures 366
Hangeul (DBCS) fields 366
Kanji (DBCS) fields 366
left-justify fields 365
length adjunct 363
locating cursor (MSQPOS) 361
mandatory enter attribute 362
mandatory fill attribute 363
map-defined input editing
AID translation 365
AID-receiver field 365
folding input data 365
justify/pad fields 365
MSCPOS call 361
mapping *(continued)*
MSGET and MSPUT calls 358
new functions 367
overlaying application data areas 366
PS adjunct 362
receive requests 360
REWRITE and REJECT requests 361
right-justify fields 365
selector adjunct 361
send requests 360
setting character attributes from terminal 364
supplied declarations 367
transforms 365
trigger field attribute 363
validation adjunct 362
WRITE requests 361
margin sizes 398
marker box
query size (GSQMB) 152
set size (GSMB) 135
markers
box, GDF order 294
box, set default, PSC GDF order 306
draw a series of (GSMRKS) 138
draw single (GSMARK) 135
draw, GDF order 294
query scale (GSQMSC) 153
query type (GSQMS) 153
scale, GDF order 295
set scale (GSMSC) 138
set type (GSMS) 138
type, GDF order 295
type, set default, PSC GDF order 306
mixed double-byte and single-byte character 372
mixed fields
control use by mapping (SPMXMP) 231
introduction 367
with position 372
with-position 367
without position 372
without-position 367
MIXSOSI, external default 390
modify
current operator window (WSMOD) 235
partition (PTNMOD) 225
segment attributes (GSSATS) 166
move current position without drawing (GSCP) 112
move without drawing (GSMOVE) 137
multi-line text strings 339
multiple fields define (ASDFMT) 29
MVS/XA, usage of FSEXIT 74

N

name-lists
family-1 415

name-lists *(continued)*
name-count values in DSOPEN
CICS 415
IMS 416
MVS/Batch 417
TSO 416
VM 418
VSE 416
reserved names “*” and blanks 415
NATLANG, external default 391
negate pixels of extracted image (IMRNEG) 194
nicknames
encoded-UDS format 66
ESQUNL call 69
ESQUNS call 69
list 69
nonchained attributes, set initial (GSSATI) 166
nonreentrant interface 1
null-to-blank conversion on input, define (ASFIN) 34
number of copies printed 398
NUMBFRM, external default 391

O

OBJFILE, external default 391
OFDSTYPE, processing option 398
OFFORMAT, processing option 399
open
alternate device (FSOPEN) 78
device (DSOPEN) 59
operator reply mode (ASMODE) 39
operator windows 404
create (WSCRT) 233
creating default 436
delete (WSDEL) 234
device variations 241
modify (WSMOD) 235
query (WSQRY) 236
query identifiers (WSQWI) 237
query numbers (WSQWN) 237
query unique identifier (WSQUN) 237
query viewing priorities (WSQWP) 238
select (WSSEL) 238
set viewing priorities (WSSWP) 239
virtual devices 436
virtual screen 436
windowed deviceinput/output (WSIO) 235
orientate extracted image (IMRORN) 194
origin-identification option 403
ORIGINID, processing option 403
outline, field, define (ASFBDY) 31
overflow
caused by picture complexity 263
PS, check (FSCHEK) 71

P

PA1 usage
TSO 412
VM 410
PA2 usage under CMS 410
page
clear current (FSPCLR) 79
create (FSPCRT) 79
delete (FSPDEL) 80
query (FSPQRY) 80
query current identifier (FSQCPG) 82
query unique identifier (FSQUPG) 85
query window (FSQWIN) 91
save current contents (FSSAVE) 92
select (FSPSEL) 81
set window (FSPWIN) 81
sizes 398
page feed for plotters
processing option 400
panning and zooming 403
paper feed bin selection processing option 407
paper-size option, plotters 401
parameters for call intercept exit 436
PARMVER, external default 391
partition sets
create (PTSCRT) 227
delete (PTSDEL) 228
query attributes (PTSQRY) 229
query unique identifier (PTSQUN) 230
select (PTSSEL) 230
partitions
create (PTNCRT) 223
delete (PTNDEL) 224
device variations 241
modify (PTNMOD) 225
query current (PTNQRY) 225
query device characteristics (FSQURY) 86
query identifiers (PTSQPI) 228
query numbers (PTSQPN) 228
query unique identifier (PTNQUN) 226
query viewing priorities (PTSQPP) 229
select (PTNSEL) 226
set viewing priorities (PTSSPP) 230
patterns
set default, PSC GDF order 306
shading, set (GSPAT) 139
PATTRAN, processing option 408
PCLK, processing option 406
PCLKEVIS, processing option 406
pens for plotters
pressure option 401
velocity option 400
width option 400
personal computers processing option 406

Personal System/55 367
pick devices
  correlate tag to primitive (GSCORR) 110
  initialize (GSIPIK) 128
  query data (GSQPIK) 154
  query structure (GSQPKS) 155
  query tag (GSQTAG) 160
  set default identifier, PSC GDF order 307
  set tag (GSTAG) 176
  tag identifier, GDF order 296
picture adjustment factors 335
picture complexity before output, check (FSCHEK) 71
picture mapping information 329
picture overflow, 4224 printer 263
picture prolog
  begin, PSC GDF order 302
  end, PSC GDF order 302
  PSC GDF orders 301, 302—309
picture space
  define (GSPS) 142
  query (GSQPS) 156
picture-orientation option, plotters 402
PIF (picture interchange format) files 315
PL/I programming language
  example application data structure 358
  using with GDDM 6
place data into mapped field (MSPUT) 212, 358
plotters
  page feed 400
  paper size 401
  pen pressure 401
  pen velocity 400
  pen width 400
  picture orientation 402
  plotting-area size 401
  query device characteristics (FSQURY) 89
plotting-area option 401
PLTAREA, processing option 401
PLTDELAY, processing option 409
PLTFORMF, processing option 400
PLTPAPSZ, processing option 401
PLTPENP, processing option 401
PLTPENV, processing option 400
PLTPENW, processing option 400
PLTROTAT, processing option 402
position cursor (ASFCUR) 32
POSTPROC, processing option 410
PostScript
  grayline 410
  post processing 410
  pschar 410
PostScript printers
  rotation processing option
    PRTROT 406
preloaded PS sets 262
print control option (PRINTCTL) 397
PRINTCTL, processing option 397
PRINTDST, processing option 412
printing bar codes 349
processing options
  dummy 396
  format 395
  full descriptions 396—413
  STAGE2ID 402
  summary 395
  using with DSOPEN 395
  using with nicknames 395
processing options (procopts)
  AUNLOCK 397
  BMSCOORD 397
  CDPFTYPE 398
  CMSATTN 411
  CMSINTRP 410
  COLORMAS 413
  CPSPOOL 411
  CPTAG 411
  CTLFAST 404
  CTLKEY 405
  CTLMODE 405
  CTLPRINT 405
  CTLSAVE 405
  DEVCPG 406
  DEVCSET 410
  FASTUPD 404
  FRCETYPE 413
  GINKEY 408
  GRAYLINE 410
  HRIDOCNM 403
  HRIFORMT 399
  HRIPSIZE 399
  HRISPILL 399
  HRISWATH 399
  IMGINIT 408
  INRESRCE 405
  INVKOPUV 412
  IPDSBIN 407
  IPDSCPI 408
  IPDSIMSW 407
  IPDSLPI 407
  IPDSQUAL 406
  IPDSROT 406
  IPDSTRUN 407
  LCLMODE 403
  LOADDSYM 402
  OFDSTYPE 398
  OFFORMAT 399
  ORIGINID 403
  OUTONLY 397
  PATTRAN 408
  PCLK 406
  PCLKEVIS 406

processing options (procopts) *(continued)*
PLTAREA 401
PLTDELAY 409
PLTFORMF 400
PLTPAPSZ 401
PLTPENP 401
PLTPENV 400
PLTPENW 400
PLTROTAT 402
POSTPROC 410
PRINTCTL 397
PRINTDST 412
PRTPSIZE 399
PRTROT 406
PSCHAR 410
PSCNVCTL 404
SEGSTORE 402
SPECDEV 403
STAGE2ID 402
TOFILE 409
TSOINTRP 412
TSORESHW 412
WINDOW 404
procopts
See processing options
programmed symbols (PS)
adjunct 362
attribute 362
code 362
loading 362
overflow check (FSCHEK) 71
overflow of complex pictures 263
sets 262, 362
store 362
programmed symbols set identifier (PSID) 362
programming language considerations 2—11
projection
apply, while transfering data between images (IMXFER) 198
create an empty (IMPCRT) 189
image, delete (IMPDEL) 189
restore from auxiliary storage (IMPRST) 190
save on auxiliary storage (IMPSAV) 190
unique identifier, get and reserve (IMPGID) 189
PRTPSIZE, processing option 399
PRTROT, processing option 406
PS overflow check 243
PS stores
conditionally load symbol set into, from auxiliary storage (PSLSSC) 221
load symbol set into, from application program (PSDSS) 219
load symbol set into, from auxiliary storage (PSLSS) 220
loading 261
numbers 261
query status (PSQSS) 222
PS stores *(continued)*
release symbol sets from (PSRSS) 222
reserve or release (PSRSV) 223
PSCHAR, processing option 410
PSCNVCTL, processing option 404

Q

quasi-reentrancy 1, 2
query
all geometric attributes (GSQAGA) 143
application data structure definition (MSQADS) 213
attributes of image (IMAQRY) 185
background color-mix mode (GSQBMX) 145
character angle (GSQCA) 146
character colors for field (ASQCOL) 39
character direction (GSQCD) 147
character highlights for field (ASQHLT) 42
character shear (GSQCH) 147
character symbol sets for field (ASQSS) 44
character-box size (GSQCB) 146
character-box spacing (GSQCBS) 147
choice device data (GSQCHO) 148
clipping state (GSQCLP) 148
code page (GSQCPG) 149
code page of GDDM object (ESQCPG) 68
current application group (ESAQRY) 65
current character mode (GSQCM) 148
current color (GSQCOL) 148
current page identifier (FSQCPG) 82
current partition (PTNQRY) 225
current position 149
current symbol-set identifier (GSQCS) 149
current viewport definition (GSQVIE) 161
cursor position (MSQPOS) 361
cursor position in map (MSQPOS) 218
cursor position, alphanumeric (ASQCUR) 40
cursor position, alphanumeric (GSQCUR) 150
data boundary definition (GSQBND) 146
default graphics cell size (GSQCEL) 147
device characteristics (DSQDEV) 62
device characteristics (FSQDEV) 82
device characteristics (FSQURY) 85
device stores status (PSQSS) 222
device usage (DSQUSE) 63
encoded user default specification (ESQEUD) 68
existence of GDDM object (ESQOBJ) 69
field attributes (ASQFLD) 41
field list (APQRY) 24
field list identifier (APQUID) 25
field list identifiers (APQIDS) 23
field list numbers (APQNUM) 24
field list size (APQSIZ) 25
foreground color-mix mode (GSQMIX) 153
fractional line width (GSQFLW) 150
graphics attribute mode (GSQAM) 144

query *(continued)*
graphics field (GSQFLD) 150
image box cursor (ISQBOX) 204
image compressions (ISQCOM) 205
image field (ISQFLD) 205
image formats (ISQFOR) 206
image locator cursor position (ISQLOC) 206
image scanner device (ISQSCA) 207
initial segment attributes (GSQATI) 144
last error (FSQERR) 83
length of field contents (ASQLEN) 42
line type (GSQLT) 151
line width (GSQLW) 152
loaded symbol sets (GSQSS) 157
locator data (GSQLOC) 151
logical input device (GSQLID) 151
map characteristics (MSQMAP) 217
map fit (MSQFIT) 216
mapgroup characteristics (MSQGRP) 217
mapped field (MSQFLD) 216
mapped page (MSPQRY) 212
marker box size (GSQMB) 152
marker scale (GSQMSC) 153
marker type (GSQMS) 153
mixed-string attribute of graphics text (GSQSEN) 156
modified fields (ASQMOD) 43
modified mapped fields (MSQMOD) 218
number of fields (ASQMAX) 43
number of loaded symbol sets (GSQNSS) 153
number of modified fields (ASQNMF) 44
number of segments (GSQMAX) 152
operator window (WSQRY) 236
operator window identifiers (WSQWI) 237
operator window numbers (WSQWN) 237
operator window viewing priorities (WSQWP) 238
page window (FSQWIN) 91
partition identifiers (PTSQPI) 228
partition numbers (PTSQPN) 228
partition set attributes (PTSQRY) 229
partition viewing priorities (PTSQPP) 229
pick data (GSQPIK) 154
pick structure (GSQPKS) 155
picture space (GSQPS) 156
segment attributes (GSQATS) 145
segment origin (GSQORG) 154
segment position (GSQPOS) 155
segment priority (GSQPRI) 155
segment transform (GSQTFM) 161
segment viewing limits (GSQSVL) 159
shading pattern (GSQPAT) 154
simultaneous queue entry (GSQSIM) 157
specified page (FSPQRY) 80
string data (GSQSTR) 159
stroke data (GSQSTK) 158
supported image resolutions (ISQRES) 207
symbol set data (GSQSSD) 157
query *(continued)*
symbol set on auxiliary storage (SSQF) 232
systems environment (FSQSYS) 84
tag (GSQTAG) 160
text alignment (GSQTA) 159
text box (GSQTB) 160
unique device identifier (DSQUID) 63
unique operator window identifier (WSQUN) 237
unique page identifier (FSQUPG) 85
unique partition identifier (PTNQUN) 226
unique partition-set identifier (PTSQUN) 230
update mode (FSQUPD) 84
user control status (DSQCMF) 62
user defined nicknames (ESQUNL) 69
user defined nicknames (ESQUNS) 69
window definition (GSQWIN) 161

R

RCP
call intercept exit 436
in ADMASP call 2, 431
introduction 2, 431
read symbol set from auxiliary storage (SSREAD) 232
redefine fields (ASRFMT) 47
reentrant interface 1
reflect extracted image (IMRREF) 197
reinitialize GDDM (FSRNIT) 91
release a symbol set from a PS store (PSRSS) 222
release graphics symbol set (GSRSS) 163
reply mode, operator, define (ASMODE) 39
request code table, APL 251
request codes module, APL 251
reserve or release PS store (PSRSV) 223
reshow protocol in TSO 412
restore
attributes (GSPOP) 142
graphics data (GSPUT) 143
image from auxiliary storage (IMARST) 187
projection from auxiliary storage (IMPRST) 190
screen contents (FSREST) 91
retrieve
data from mapped field (MSGET) 211, 358
fam-4 datastream(FSGET) 75
graphics data (GSGET) 121
image data from an image (IMAGT) 181
return codes
ADMUCG 325
ADMUGC 327
ADMUGIF 345
REXX
See GDDM-REXX
rotation processing option, PostScript printers 406
run the Image Symbol Editor (ISSE) 208

S

sample program ERXPROTO 9
SAVBFSZ, external default 391
save
  current page contents (FSSAVE) 92
  graphics segment (GSSAVE) 166
  image on auxiliary storage (IMASAV) 188
  projection on auxiliary storage (IMPSAV) 190
  segments in a CGM (CGSAVE) 324
saved picture, display
  FSSHOR 93
  FSSHOW 93
SBCS (single-byte character set) 367, 372
scale extracted image (IMRSCL) 197
scanner
  control echoing of image (ISESCA) 201
  query (ISQSCA) 207
screen contents, restore (FSREST) 91
segment, graphics
  attributes, GDF order 297
  attributes, modify, GDF order 297
  end prolog, GDF order 298
  end, GDF order 298
  origin, GDF order 298
  position, GDF order 298
  query transform (GSQTFM) 161
  start, GDF order 299
  viewing window, GDF order 300
SEGSTORE, processing option 402
select
  operator window (WSSEL) 238
  page (FSPSEL) 81
  partition (PTNSEL) 226
  partition set (PTSSEL) 230
selector adjunct fields
  introduction 357
  usage 361
send
  character string to alternate device
    with carriage-control character (FSLOGC) 77, 398
    without carriage-control character (FSLOG) 77, 398
  graphics to alternate device (GSCOPY) 109
  page to alternate device (FSCOPY) 72
set
  all geometric attributes (GSSAGA) 164
  character angle (GSCA) 99
  character attributes from terminal 364
  character direction (GSCD) 103
  character mode (GSCM) 108
  character shear (GSCH) 105
  character-box size (GSCB) 102
  character-box spacing (GSCBS) 103
  code page (GSCPG) 113
  code page of GDDM object (ESSCPG) 70
  color (GSCOL) 109
set *(continued)*
  current line type (GSLT) 134
  current position (GSCP) 112
  current primitive tag (GSTAG) 176
  current resolution/scaling algorithm (IMRRAL) 196
  current symbol set (GSCS) 113
  current transform (GSSCT) 168
  cursor position (MSCPOS) 209, 361
  cursor with mapping request 361
  default field attributes (ASDFLT) 29
  extended image quality control parameters (ISXCTL) 208
  foreground color-mix mode (GSMIX) 136
  fractional line width (GSFLW) 121
  graphics attribute mode 96
  image quality-control parameters (ISCTL) 199
  initial segment attributes (GSSATI) 165
  line width (GSLW) 134
  marker scale (GSMSC) 138
  marker type (GSMS) 138
  marker-box size (GSMB) 135
  mixed string attribute of graphics text (GSSEN) 170
  operator window viewing priorities (WSSWP) 239
  page window (FSPWIN) 81
  partition viewing priorities (PTSSPP) 230
  segment origin (GSSORG) 171
  segment position (GSSPOS) 172
  segment priority (GSSPRI) 172
  segment transform (GSSTFM) 173
  shading pattern (GSPAT) 139
  text alignment (GSTA) 175
  update mode (FSUPDM) 95
set default PSC GDF orders
  arc parameters, set default 302
  background color-mix mode 302
  character angle 302
  character box 303
  character direction 303
  character precision 304
  character shear 304
  character-box spacing 303
  color, graphics 305
  coordinate type 304
  foreground color-mix mode 305
  graphics text 307
  line type 306
  line width, fractional 305
  marker box 306
  marker type 306
  pattern 306
  pick devices 307
  picture boundary 308
  picture origin 309
  picture scale 307
  symbol sets 304
  viewing window 308

shaded area, start (GSAREA) 97
shading patterns
  GDF order 296
  query (GSQPAT) 154
shading patterns, user-defined 408
simultaneous queue entry, query (GSQSIM) 157
SOSIEMC, external default 391
sound terminal alarm (FSALRM) 71
source-format user default specification (ESSUDS) 71
SPECDEV, processing option 403
special device 403
specify
  device usage (DSUSE) 64
  encoded user default specification) (ESEUDS) 66
  error thresholds (FSEXIT) 74
  field contents (ASCPUT) 27
SPI interface entry point 2, 431
spill file usage (4250 printers) 399
start
  data entry into image (IMAPTS) 184
  drawing defaults definition (GSDEFS) 114
  retrieval of data from an image (IMAGTS) 182
  retrieval of graphics data (GSGETS) 122
  shaded area (GSAREA) 97
STGRET, external default 392
stored object file format 279—280
string devices, query data (GSQSTR) 159
stroke devices, query data (GSQSTK) 158
structure correlation (GSCORS) 111
structured field formats
  bar codes 349—352
  documents 350
  environment groups 349—351
  graphics data 351
  graphics objects 350—351
  image data 351
  image objects 350—351
  image pictures 352
  map coded fonts 352
  map medium 353
  no operation 353
  object area 353
  pages 350—354
  presentation text 350—356
substitution character in symbol-set name 262
swathes, number of 399
symbol sets
  character, query for field (ASQSS) 44
  character, set default, PSC GDF order 304
  character, specify within field (ASCSS) 27
  color shading pattern 265
  component threshold for DBCS, DBCSLIM external default 387
  conditional loading of 262
  conditionally load into PS store from auxiliary storage (PSLSSC) 221
symbol sets *(continued)*
  DBCS 264
  default names DBCS, DBCSDNM external default 387
  double-byte character 265
  end mapping, PSC GDF order 302
  format of VSS 277—278
  GDF order 288
  geometric shading pattern 265
  graphics, load from application program (GSDSS) 117
  graphics, load from auxiliary storage (GSLSS) 133
  graphics, release (GSRSS) 163
  handling by GDDM 261
  identification 261
  ISS format 276
  ISS samples 265
  Katakana character 265
  language for DBCS, DBCSLNG external default 387
  load into PS store from application program (PSDSS) 219
  load into PS store from auxiliary storage (PSLSS) 220
  loaded, query (GSQSS) 157
  loading from workstation or GDDM defaults 402
  loading graphics sets 262
  loading PS stores 261
  map identifier, PSC GDF order 301
  mapping, PSC GDF order 301
  naming convention 264
  primary, define for fields (ASFPSS) 35
  PS overflow 263
  PS store numbers 261
  PSC GDF orders 300, 301—302
  query current identifier (GSQCS) 149
  query data (GSQSSD) 157
  query number loaded (GSQNSS) 153
  query on auxiliary storage (SSQF) 232
  read from auxiliary storage (SSREAD) 232
  release from a PS store (PSRSS) 222
  script CECP character 265
  selecting by device type 262
  selection for DBCS, DBCSDFT external default 387
  set current (GSCS) 113
  standard CECP character 265
  standard marker 265
  standard shading pattern 265
  using in printing 263
  using preloaded PS sets 262
  using PS with graphics 262
  VSS samples 265
  VSS typeface illustrations 267—274
  write to auxiliary storage (SSWRT) 233
syntax conventions 21
  assembler-language linkage 3
  C programming language 3
  COBOL programming language 5
  FORTRAN programming language 5
  GDDM-REXX 6—10

syntax conventions *(continued)*
in manuals 10
PL/I programming language 6
system programmer interface 2, 431
system programmer interface block (SPIB)
SPINIT call 231, 431
user exit definition 433
systems environment, query (FSQSYS) 84

T

task switch user exit option 433
terminal alarm, sound (FSALRM) 71
terminate GDDM processing (FSTERM) 94
TIMEFRM, external default 392
TOFILE, processing option 409
trace, control internal (FSTRCE) 95
TRACE, external default 392
transaction work area 2
transfer data between two images, applying a projection (IMXFER) 198
transfer file
host to workstation (RECEIVE) 327
workstation to host (SEND) 327
translate character string (FSTRAN) 94
translating AID values 365
translating user-defined shading patterns 408
translation tables for I/O, define (ASDTRN) 30
transparency, field, define (ASFTRA) 37
TRCESTR, external default 392
TRCEWID, external default 392
trim image to rectangle (IMATRM) 188
TRTABLE, external default 392
TSO
APL external default specification, TSOAPLF external default 392
CGM conversion profile filetype, TSOCPT external default 392
CLEAR/PA1 protocol 412
color master ddname/high-level qualifier 392
dynamic allocation size, TSOS99S external default 393
family-2 and -4 print-file destination 412
I/O Emulation, TSOEMUL external default 392
reshow protocol 412
storage exit routines 437
task switch exit 434
TSOTRCE external default, trace ddname for TSO 393
unit specification, TSOS99U external default 393
TSOAPLF, external default 392
TSOCOLM, external default 392
TSOCPT, external default 392
TSODECK, external default 392
TSODFTS, external default 392
TSOEMUL, external default 392
TSOGIMP, external default 393
TSOIADS, external default 393
TSOIFMT, external default 393
TSOINTRP, processing option 412
TSOMONO, external default 393
TSOPRNT, external default 393
TSORESHW, processing option 412
TSORESV, external default 393
TSOS99S, external default 393
TSOS99U, external default 393
TSOSYSP, external default 393
TSOTRCE, external default 393

U

UDS (user default specification)
encoded, specify (ESEUDS) 66
query encoded (ESQEUD) 68
source-format, specify (ESSUDS) 71
unique identifier
update mode, query (FSQUPD) 84
user control 405
automatic entry (DSCMF) 57
fast path mode 404
query status (DSQCMF) 62
user default specification
See UDS
user exits 432
call intercept exit 435
control block, UXBLOCK 433
GDDM conventions 433
storage exit routines 437
UDS (user default specification) 433
user exits 432
user-defined shading patterns, translation of 408

V

vector operations (GSVECM) 177
vector symbol sets
format 277—278
samples of 265
typeface illustrations 267—274
viewport
define (GSVIEW) 177
query current definition (GSQVIE) 161
VM
APL feature, CMSAPLF external default 385
attention handling 411
automatic invocation of print utility 412
CGM conversion profile filetype, CMSCPT external default 386
color master filetype, CMSCOLM external default 386
CP SPOOL parameters 411
CP TAG parameters 411
filename and filetype, CMSDFTS external default 386
monochrome color master, CMSMONO external default 386

VM *(continued)*
MSL (map specification library) name, CMSMSLT external default 386
PA1 and PA2 protocol 410
print filetype, CMSPRNT external default 386
storage exit routines 437
trace filename/filetype, CMSTRCE external default 386
work-file filetype, CMSTEMP external default 386
VSE
batch printing default, DFTXTNA external default 387
color master file name, VSECOLM external default 393
file name, VSEDFTS external default 393
monochrome file name, VSEMONO external default 393
trace file name, VSETRCE external default 393
VSECOLM, external default 393
VSEDFTS, external default 393
VSEMONO, external default 393
VSETRCE, external default 393
VSS
see vector symbol sets

W

wait for graphics input (GSREAD) 162
WINDOW, processing option 404
windowed device input/output (WSIO) 235
write symbol set to auxiliary storage (SSWRT) 233

Z

zooming and panning pictures 403

Sending your comments to IBM

GDDM

Base Application Programming Reference

SC33-0868-02

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: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.

Readers' Comments

GDDM

Base Application Programming Reference

SC33-0868-02

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 Base Application Programming Reference SC33-0868-02

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.

SC33-0868-02